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

文件目录

文件目录　对大量的文件实施有效的管理。按文件名对所有文件进行管理，实现从文件符号名到文件实体的映射

目录管理要求　实现“按名存取”；提高目录的检索速度；文件共享；允许文件重名

文件控制块（FCB）（文件与FCB一一对应）

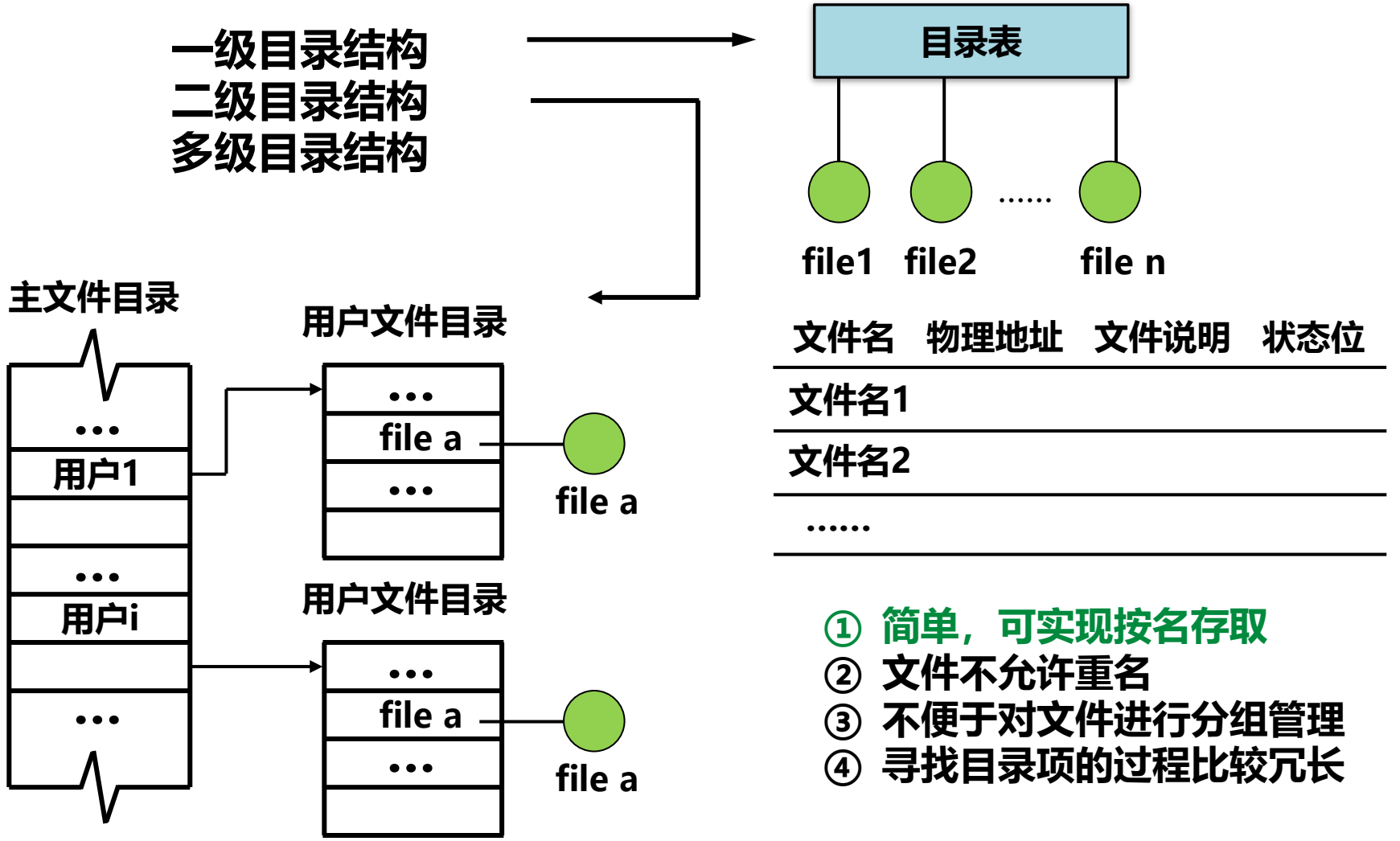
- 基本信息类：文件名、文件物理位置（设备名、起始盘号、长度等用于文件逻辑到物理地址变换）等
- 存取控制类：文件主、核准用户及一般用户对文件的存取权限等
- 使用信息类：建立及上次修改时间、当前打开该文件的进程数、在内存中是否修改等

FCB的集合构成文件目录（一个FCB就是一条目录项），同样保存在外存，称为目录文件。

文件目录

目录结构

一级目录结构
二级目录结构
多级目录结构

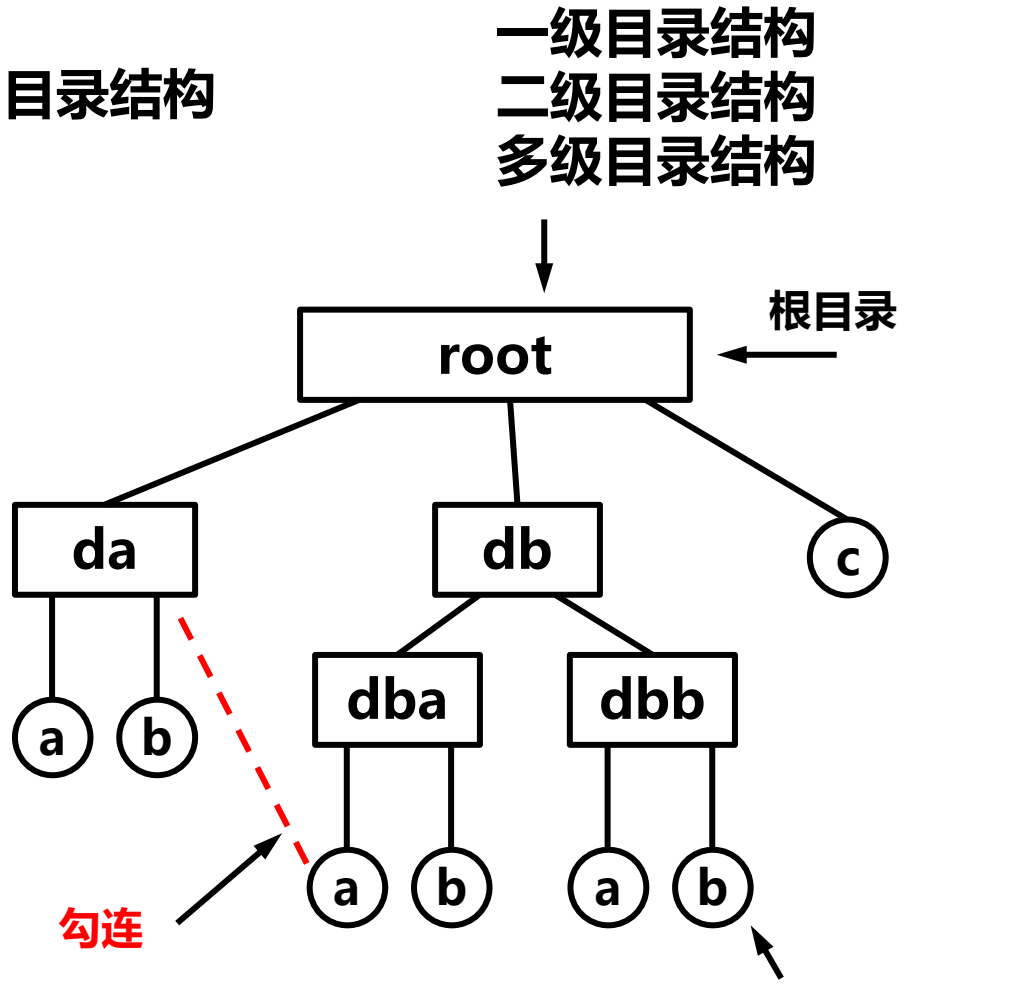


文件名	物理地址	文件说明	状态位
文件名1			
文件名2			
……			

① 检索速度提高
② 不同用户可以使用相同的文件名
③ 缺乏灵活性

① 简单，可实现按名存取
② 文件不允许重名
③ 不便于对文件进行分组管理
④ 寻找目录项的过程比较冗长

文件目录



- **路径名**：从根目录开始到任何数据文件存在的一条通路。在该路径上经过的全部目录文件名和数据文件名，用“/”连接起来。
- **当前目录**：可为每个进程设置一个“当前目录”，又称为“工作目录”。进程对各文件的访问都相对与“当前目录”。
- **相对路径**：从当前目录开始到数据文件构成的路径。
- **绝对路径**：从根目录到数据文件构成的路径。

UNIX的文件目录

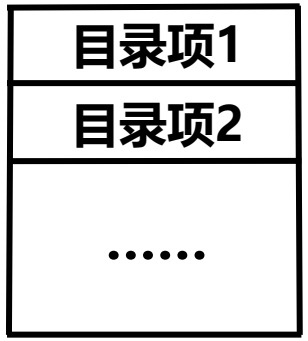
文件控制块（FCB）（文件与FCB一一对应）

- 基本信息类：文件名、文件物理位置（设备名、起始盘号、长度等用于文件逻辑到物理地址变换）等
- 存取控制类：文件主、核准用户及一般用户对文件的存取权限等
- 使用信息类：建立及上次修改时间、当前打开该文件的进程数、在内存中是否修改等

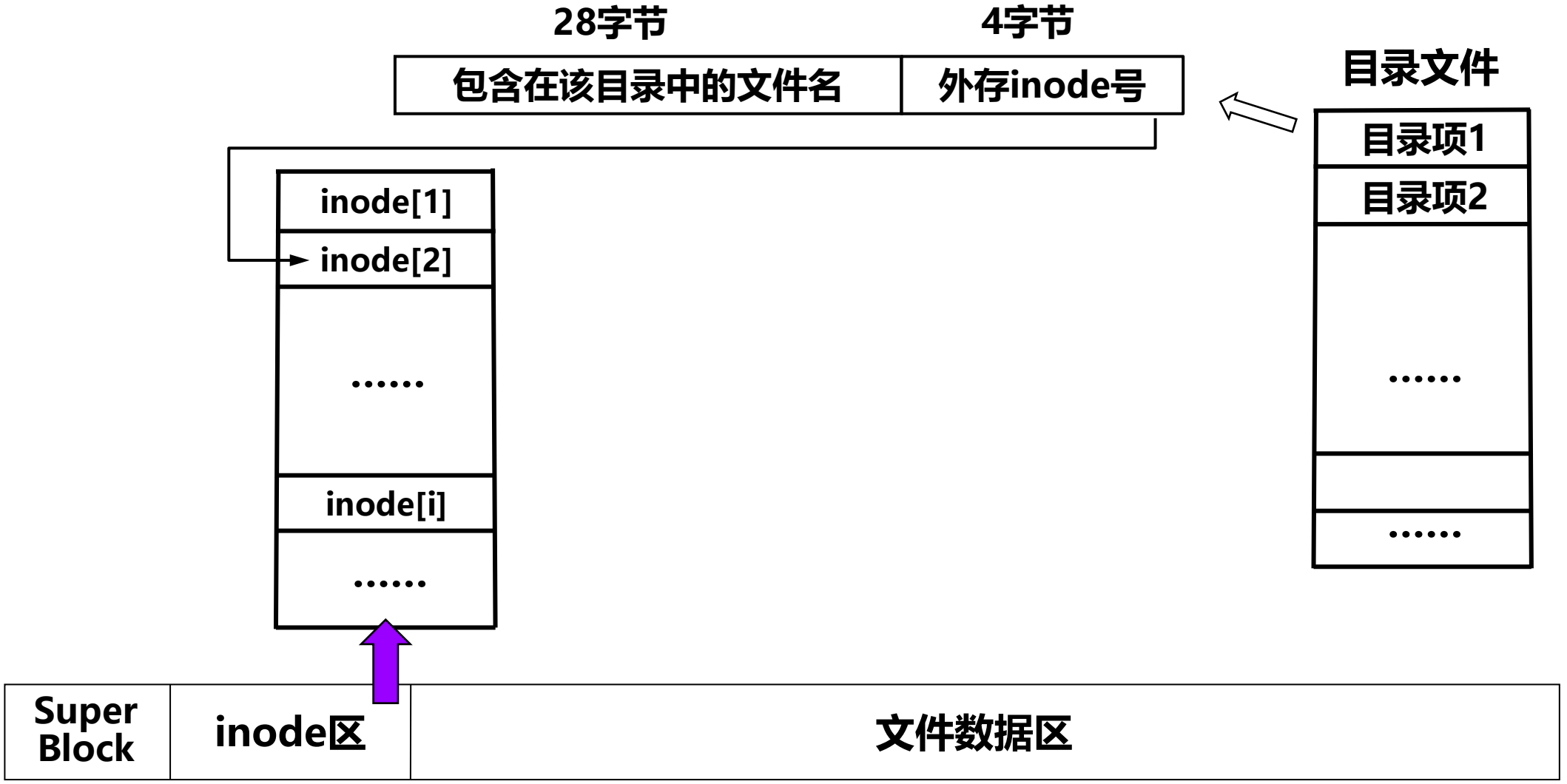
FCB的集合构成文件目录（一个FCB就是一条目录项），同样保存在外存，称为目录文件。

FCB内容过于复杂，导致目录文件非常庞大。UNIX 引入i_node，简化目录文件结构。

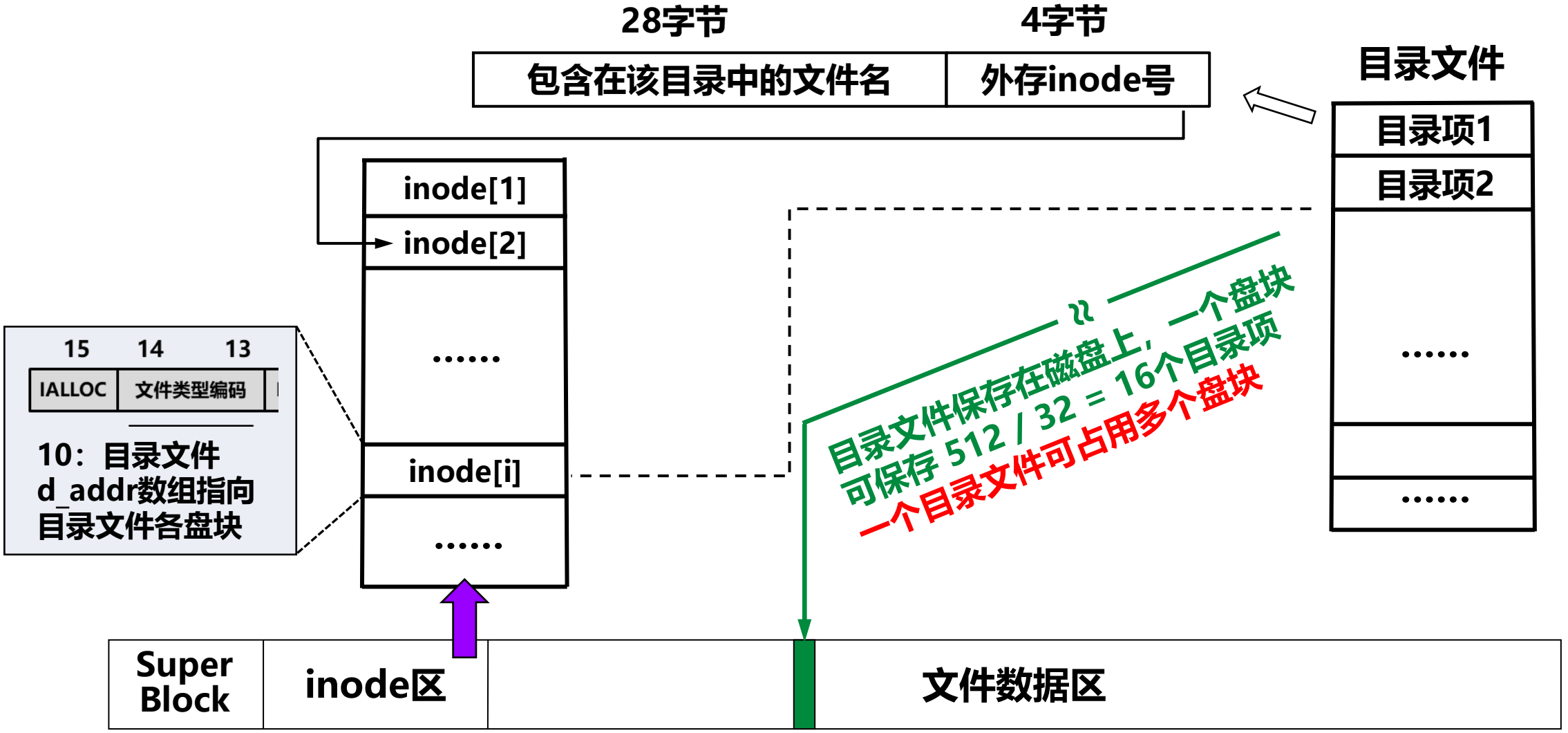
目录文件



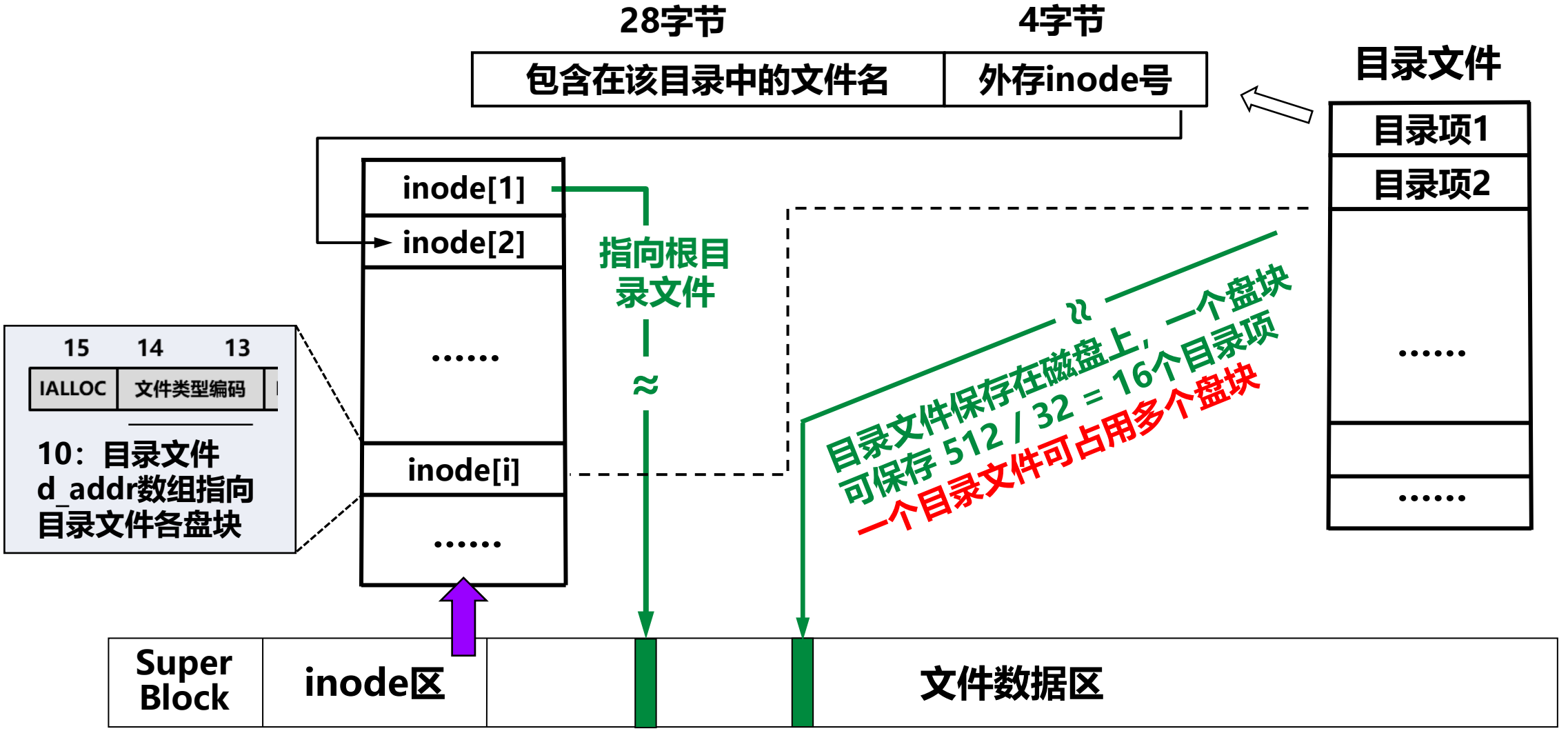
UNIX的文件目录


28字节
4字节
包含在该目录中的文件名
外存inode号
目录文件
目录项1
目录项2
.....
.....
inode[1]
inode[2]
.....
inode[i]
.....
Super Block
inode区
文件数据区

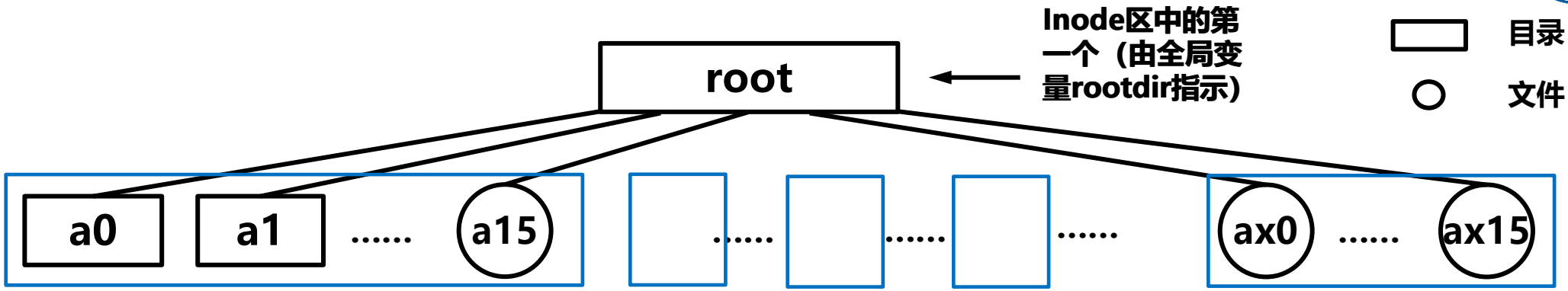
UNIX的文件目录


28字节
4字节
包含在该目录中的文件名
外存inode号
目录文件
目录项1
目录项2
.....
.....
inode[1]
inode[2]
.....
inode[i]
.....
15 14 13
IALLOC
文件类型编码
10：目录文件
d_addr数组指向
目录文件各盘块
目录文件保存在磁盘上，一个盘块
可保存 512 / 32 = 16个目录项
一个目录文件可占用多个盘块
Super Block
inode区
文件数据区

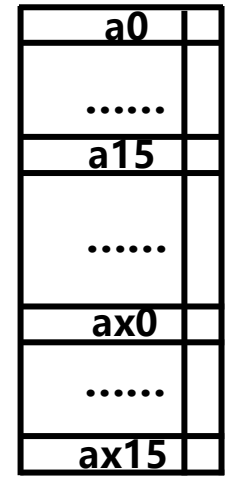
UNIX的文件目录


28字节
4字节
包含在该目录中的文件名
外存inode号
目录文件
目录项1
目录项2
.....
.....
inode[1]
inode[2]
.....
inode[i]
.....
指向根目录文件
≈
≈
目录文件保存在磁盘上，一个盘块可保存 512 / 32 = 16个目录项
一个目录文件可占用多个盘块
15 14 13
IALLOC
文件类型编码
10：目录文件
d_addr数组指向
目录文件各盘块
Super Block
inode区
文件数据区

UNIX的文件目录

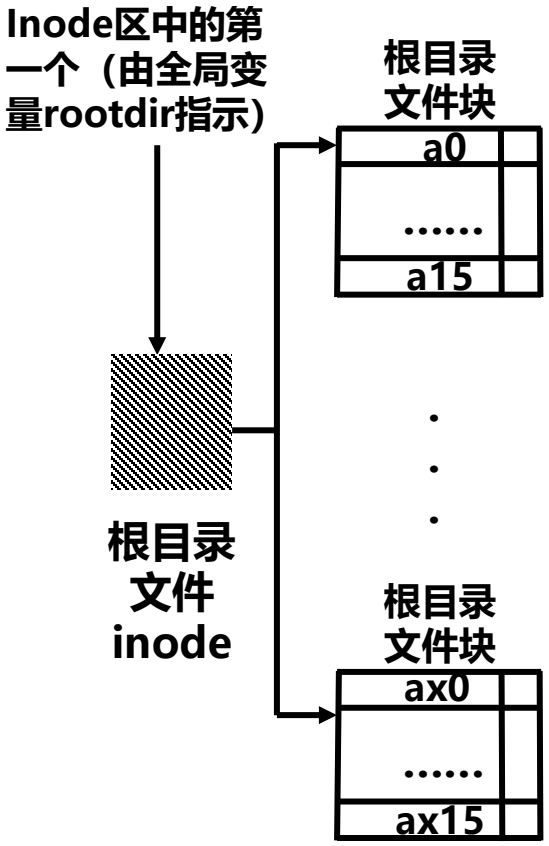


根目录文件



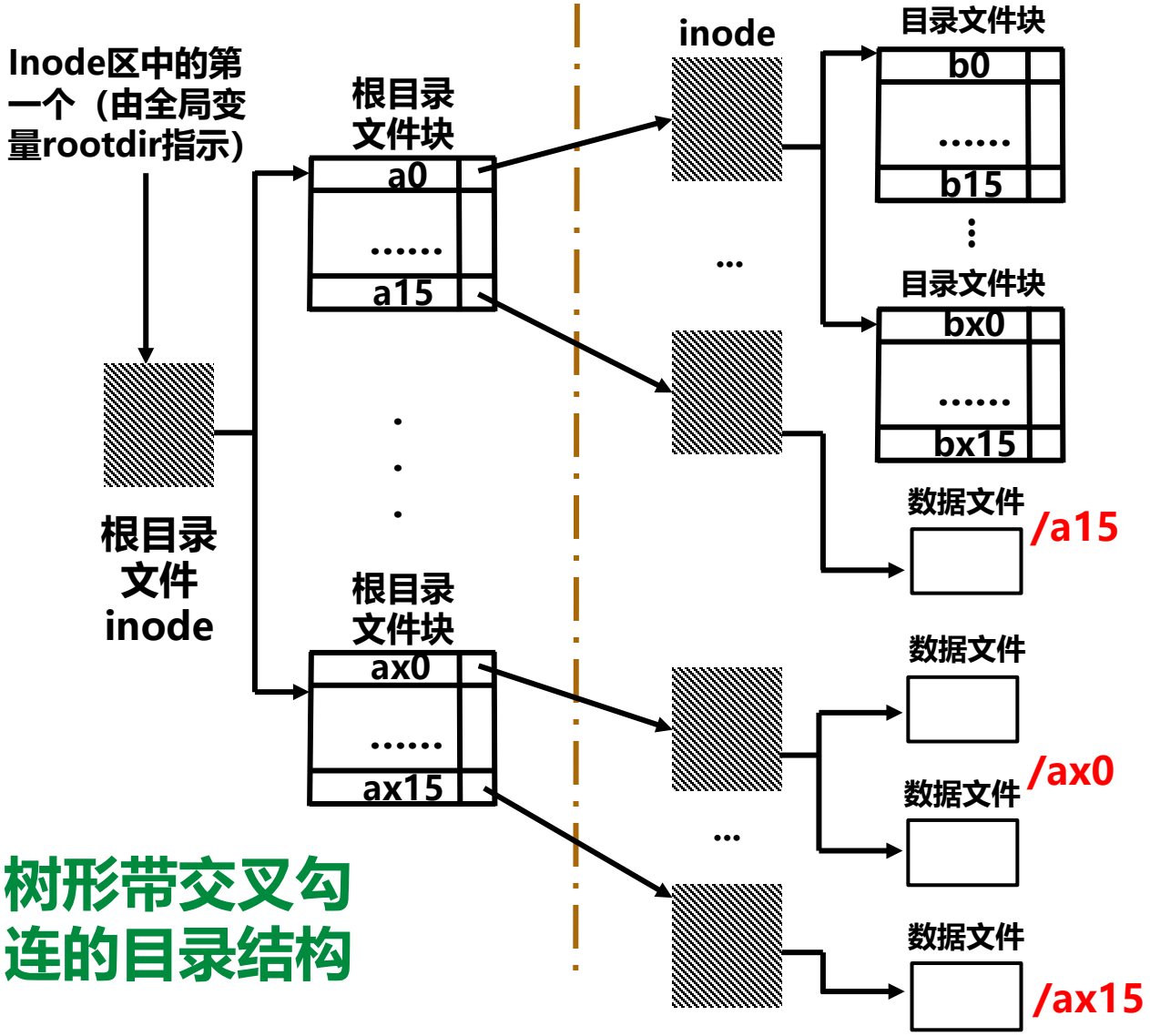
目录结构

UNIX的文件目录



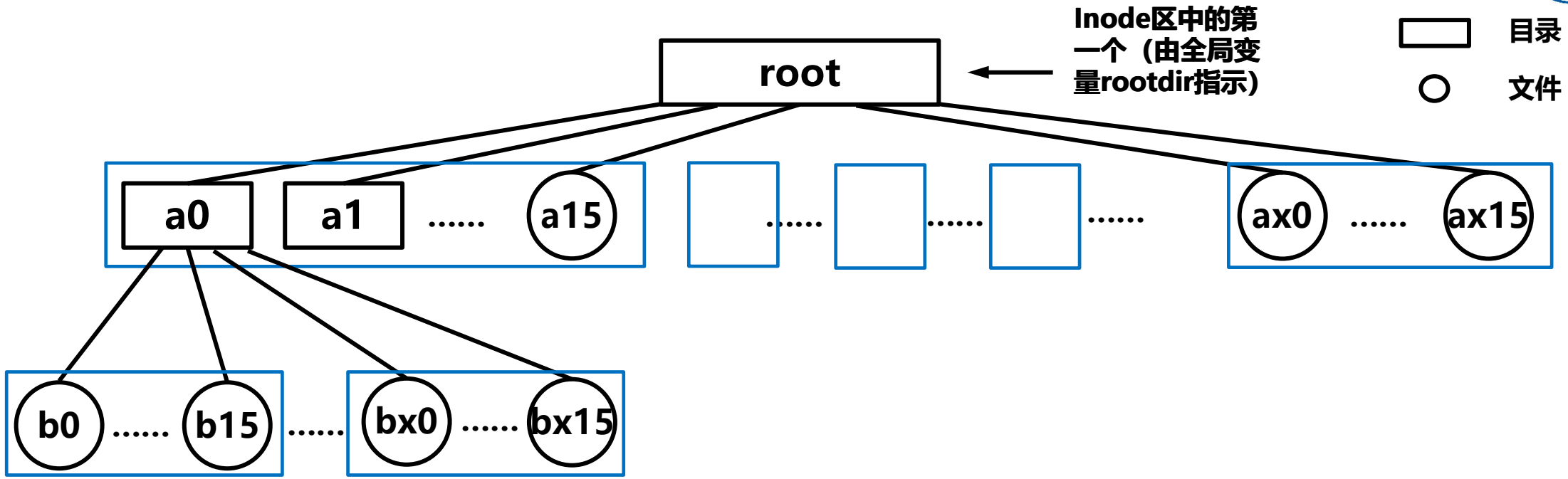
树形带交叉勾连的目录结构

UNIX的文件目录



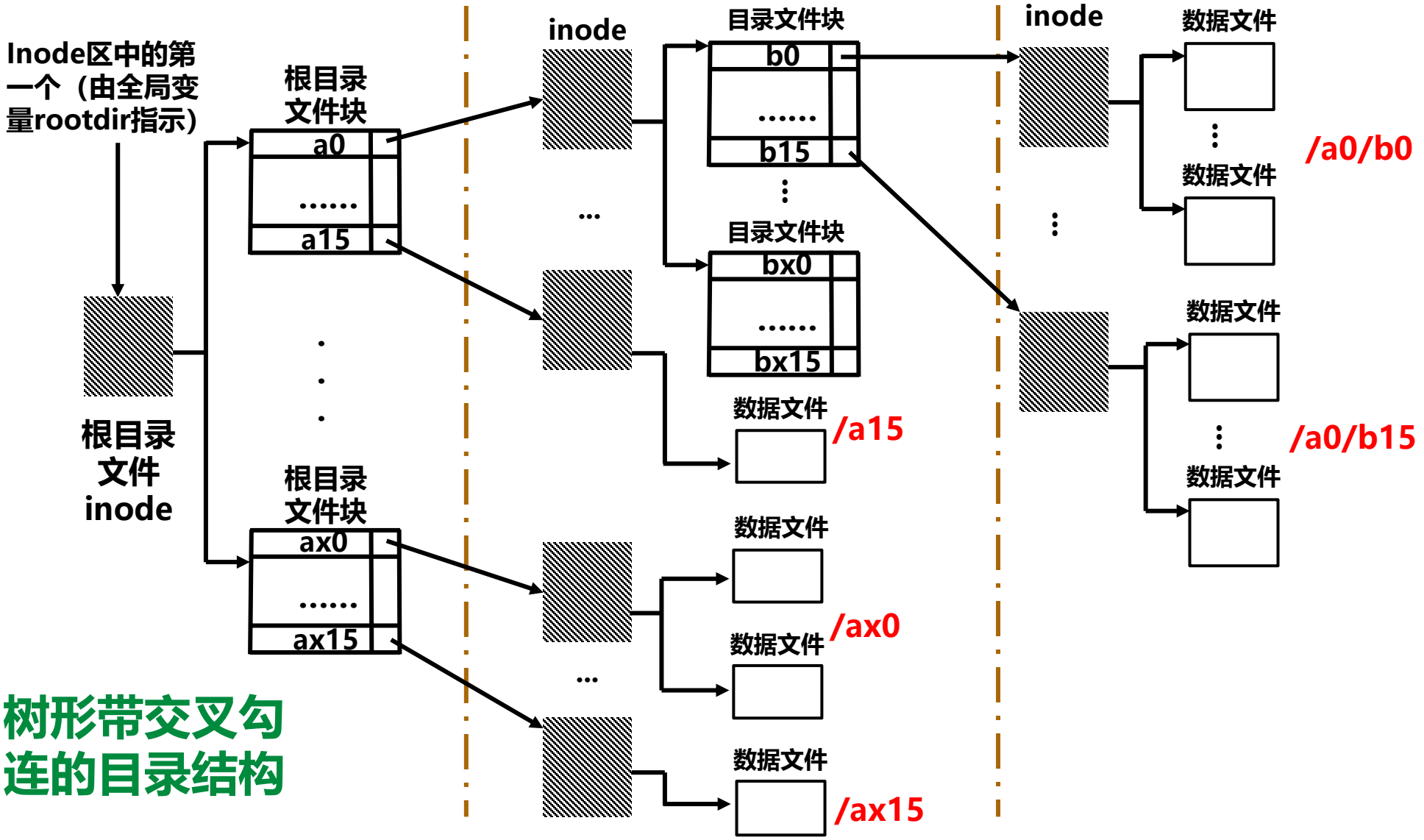
树形带交叉勾连的目录结构

UNIX的文件目录



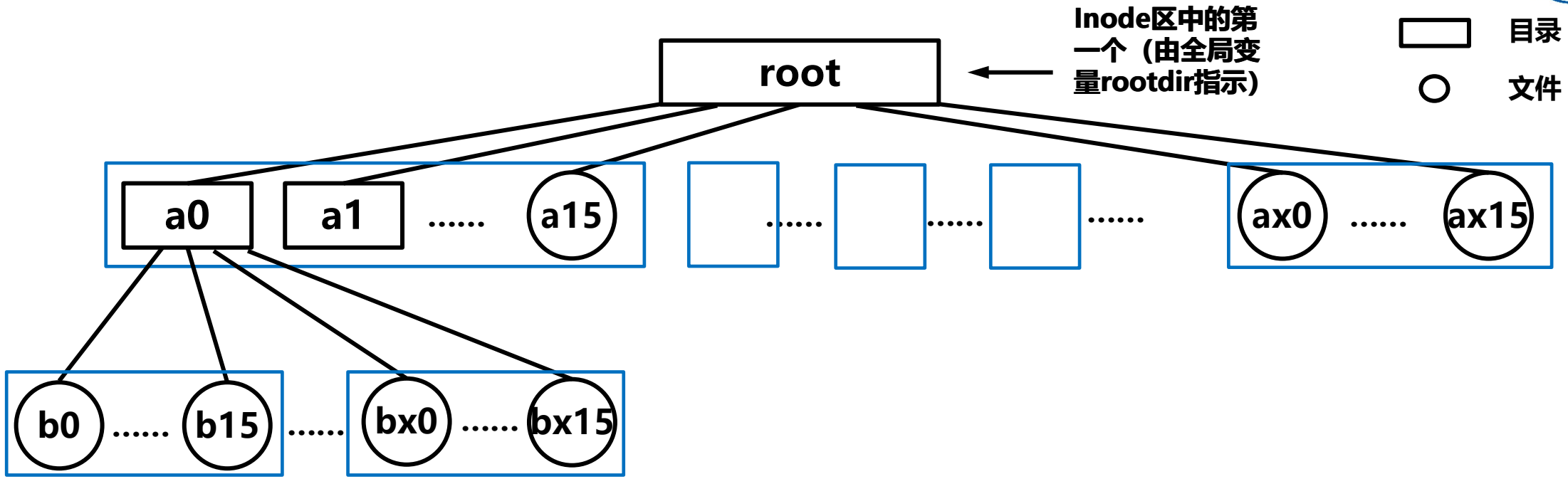
目录结构

UNIX的文件目录



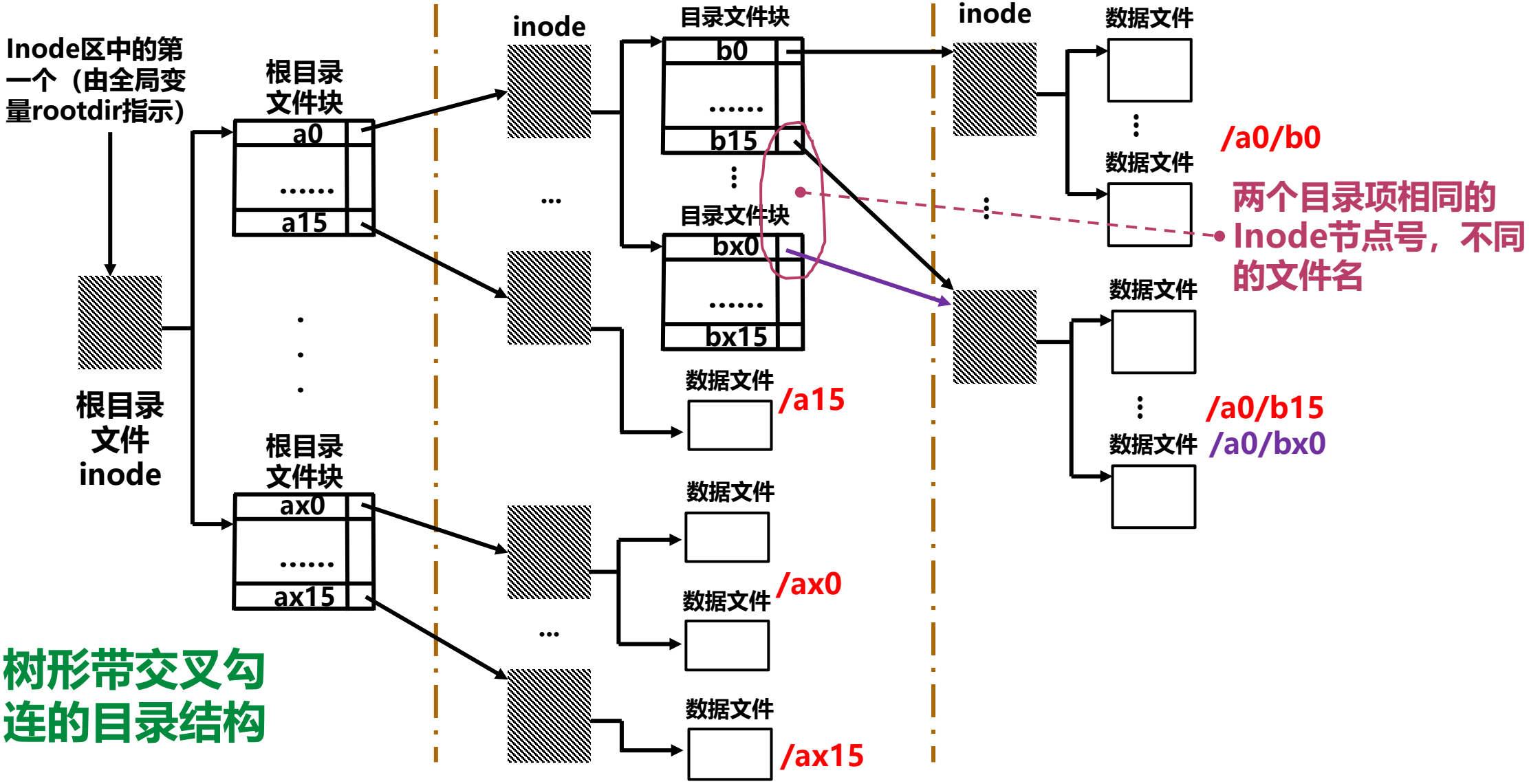
树形带交叉勾连的目录结构

UNIX的文件目录



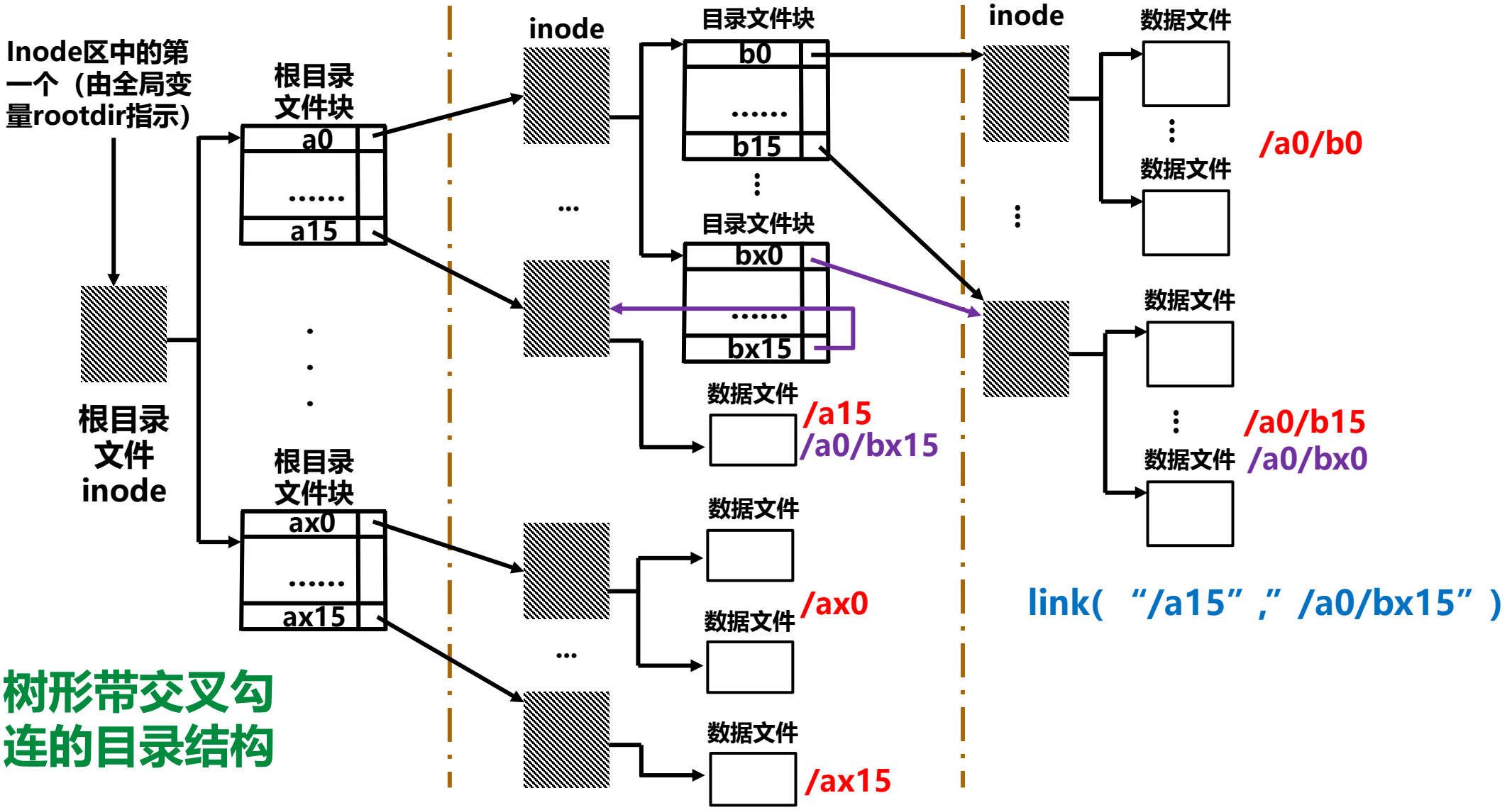
如果文件” /a0/bx0”不是通过creat创建的，而是通过：
link(“/a0/b15”,” /a0/bx0”)
创建的。。。

UNIX的文件目录

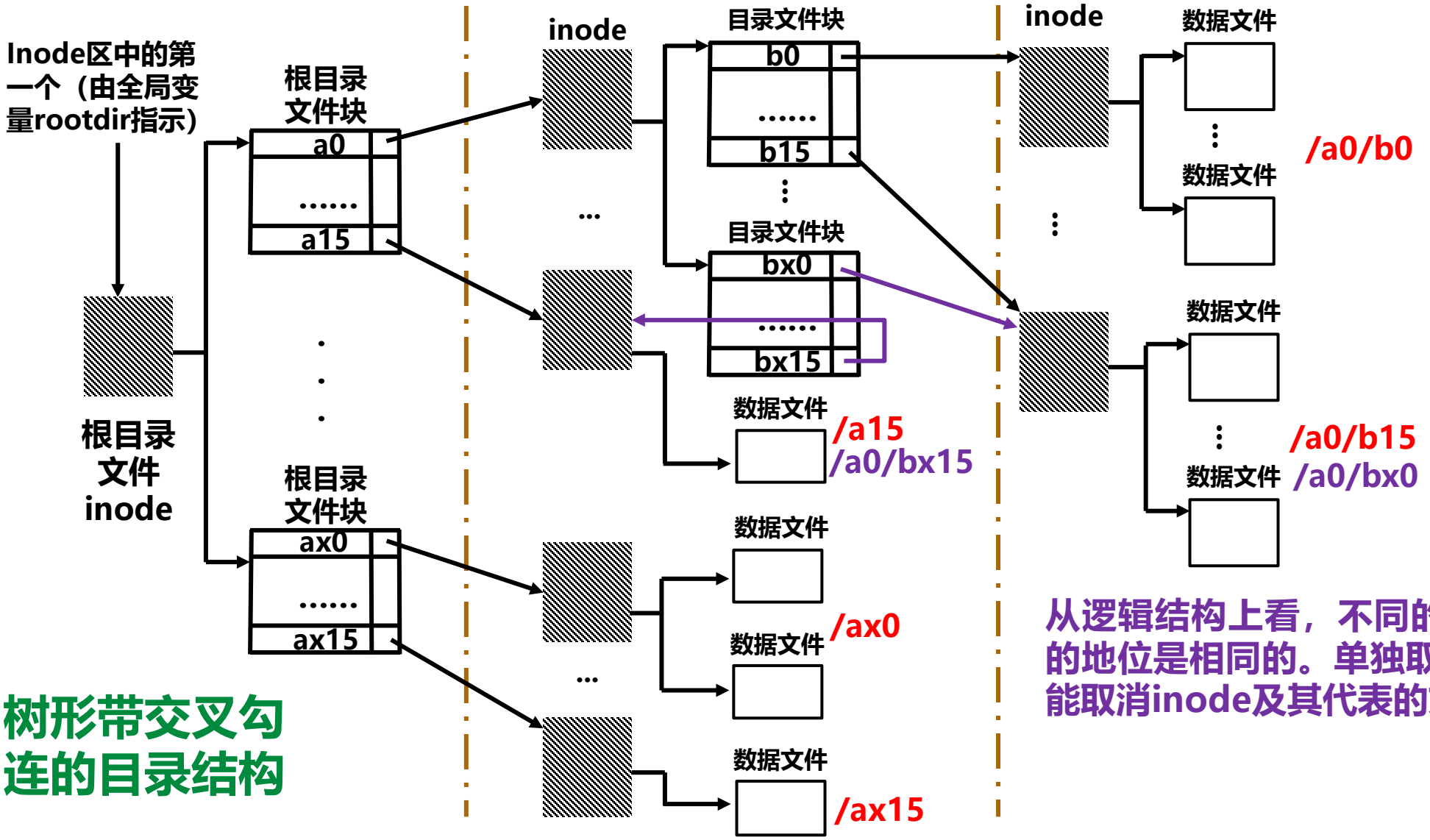


树形带交叉勾连的目录结构

UNIX的文件目录


Inode区中的第一个（由全局变量rootdir指示）
根目录文件inode
根目录文件块
a0
……
a15
根目录文件块
ax0
……
ax15
inode
目录文件块
b0
……
b15
目录文件块
bx0
……
bx15
数据文件
/a15
/a0/bx15
数据文件
/ax0
数据文件
/ax15
inode
数据文件
/a0/b0
数据文件
数据文件
/a0/b15
数据文件
/a0/bx0
link(“/a15”,” /a0/bx15”)
树形带交叉勾连的目录结构

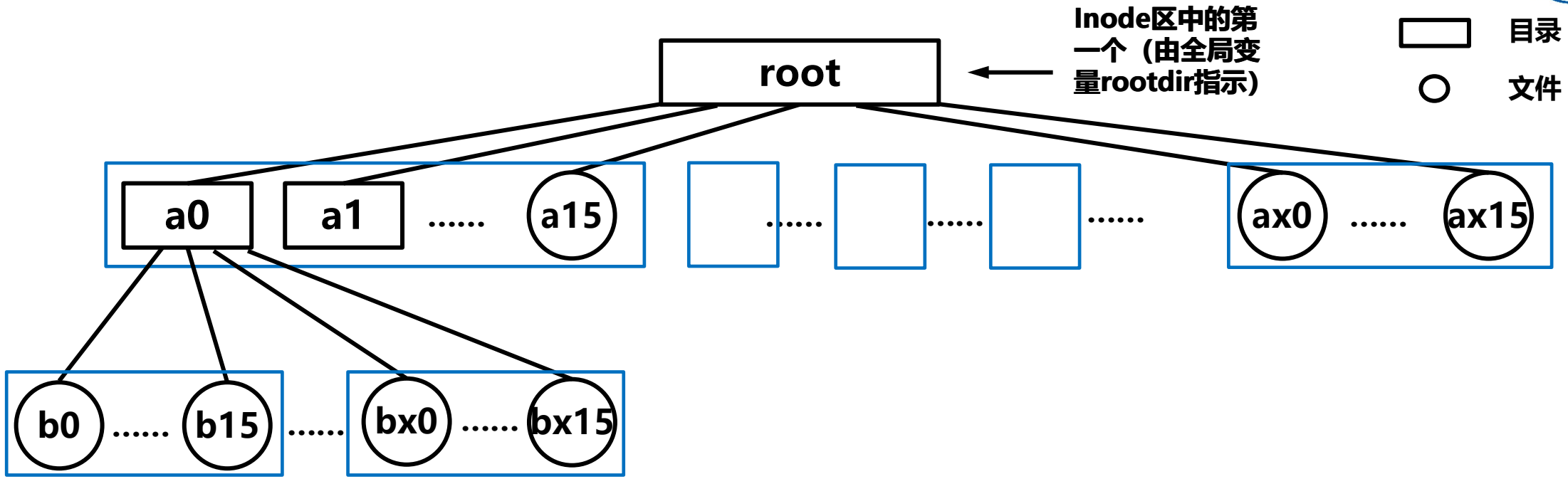
UNIX的文件目录



树形带交叉勾连的目录结构

从逻辑结构上看，不同的路径对inode的地位是相同的。单独取消一个路径不能取消inode及其代表的文件实体。

UNIX的文件目录



1. 目录树中的每一个节点对应一个文件，具有唯一的Inode。
2. 如果是目录节点，Inode中标注为目录文件；如果是文件节点，Inode中标注为数据文件或设备文件。
3. 同一级的所有节点，在上一级目录文件中登记。

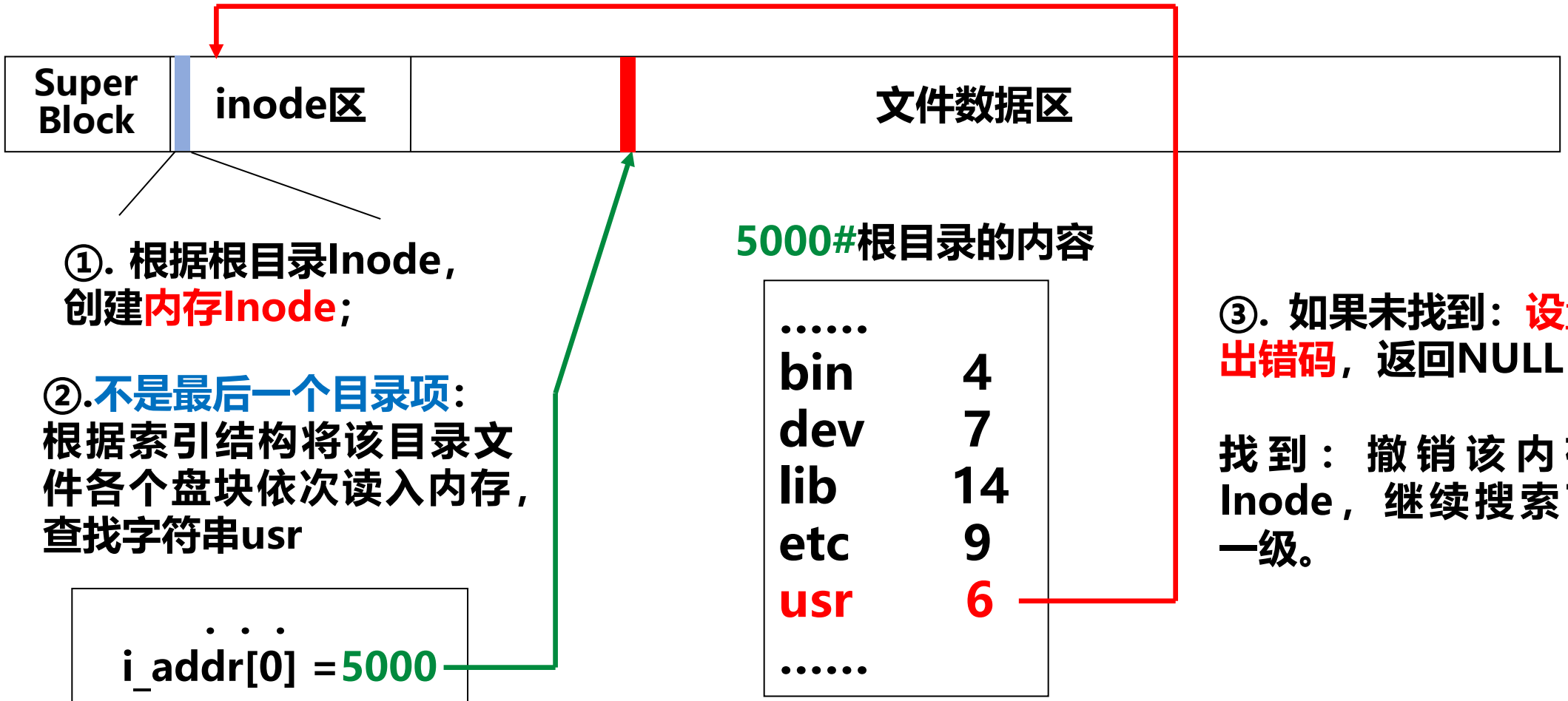
UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```

```
enum DirectorySearchMode
{
    OPEN = 0,/* 以打开文件方式搜索目录 */
    CREATE = 1,/* 以新建文件方式搜索目录 */
    DELETE = 2/* 以删除文件方式搜索目录 */
};
```

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```



文件“/usr/ast/mbox”的搜索过程

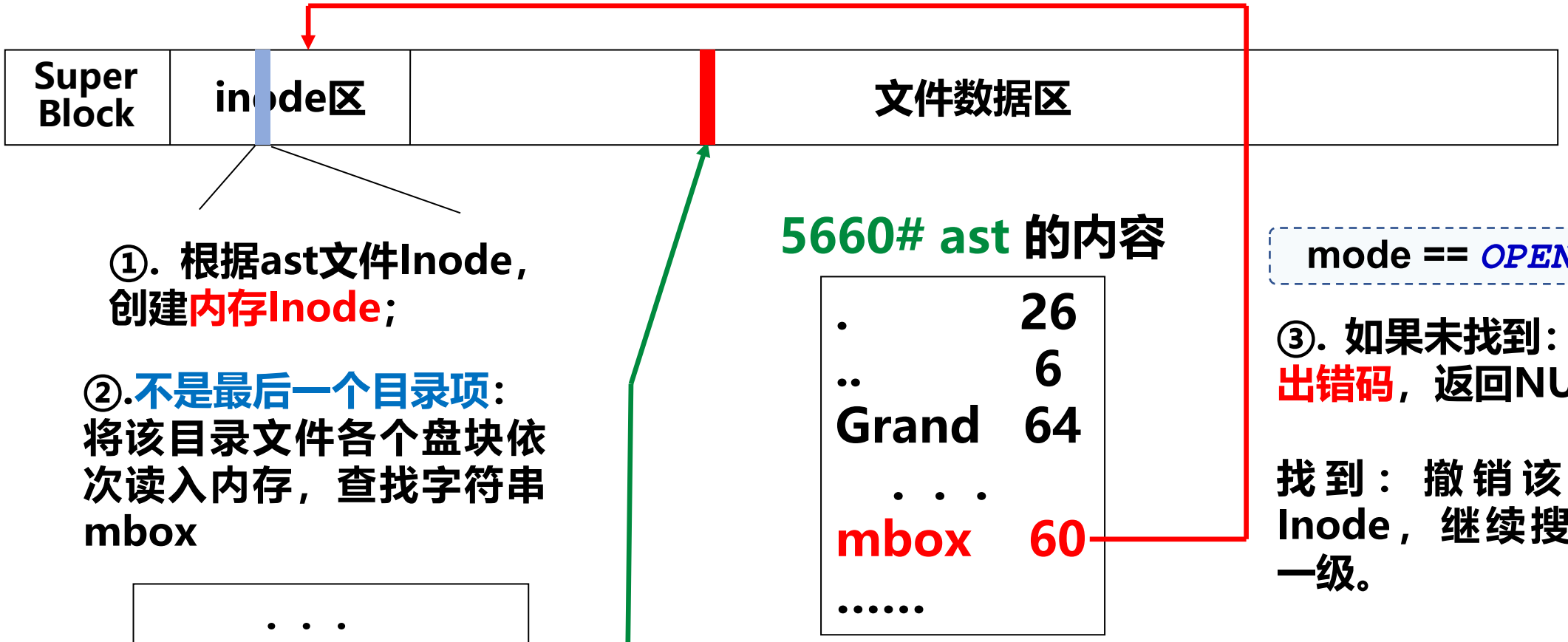
UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```

文件“/usr/ast/mbox”的搜索过程

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```



文件“/usr/ast/mbox”的搜索过程

UNIX的文件目录

文件“/usr/ast/mbox”的搜索过程

目录检索过程

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```

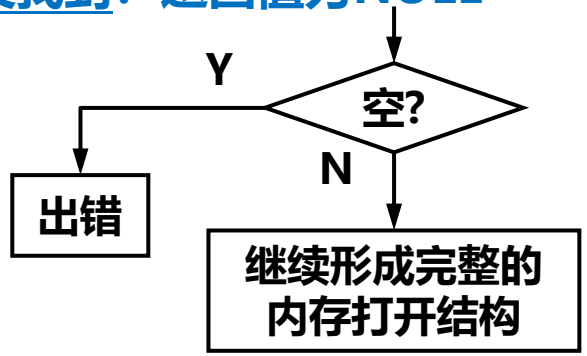
```
enum DirectorySearchMode
{
    OPEN = 0,/* 以打开文件方式搜索目录 */
    CREATE = 1,/* 以新建文件方式搜索目录 */
    DELETE = 2/* 以删除文件方式搜索目录 */
};
```

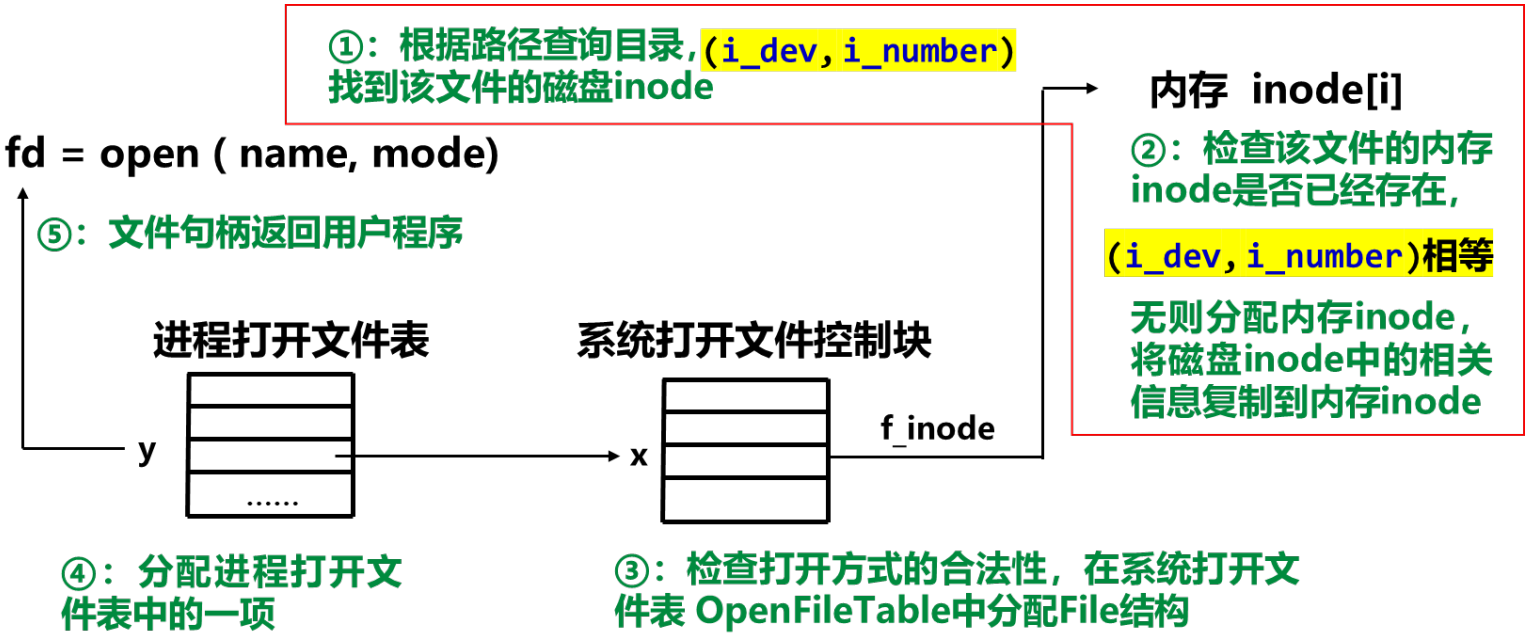
Open

Namei（文件名， FileManager::OPEN）

找到： 返回找到文件的内存Inode；

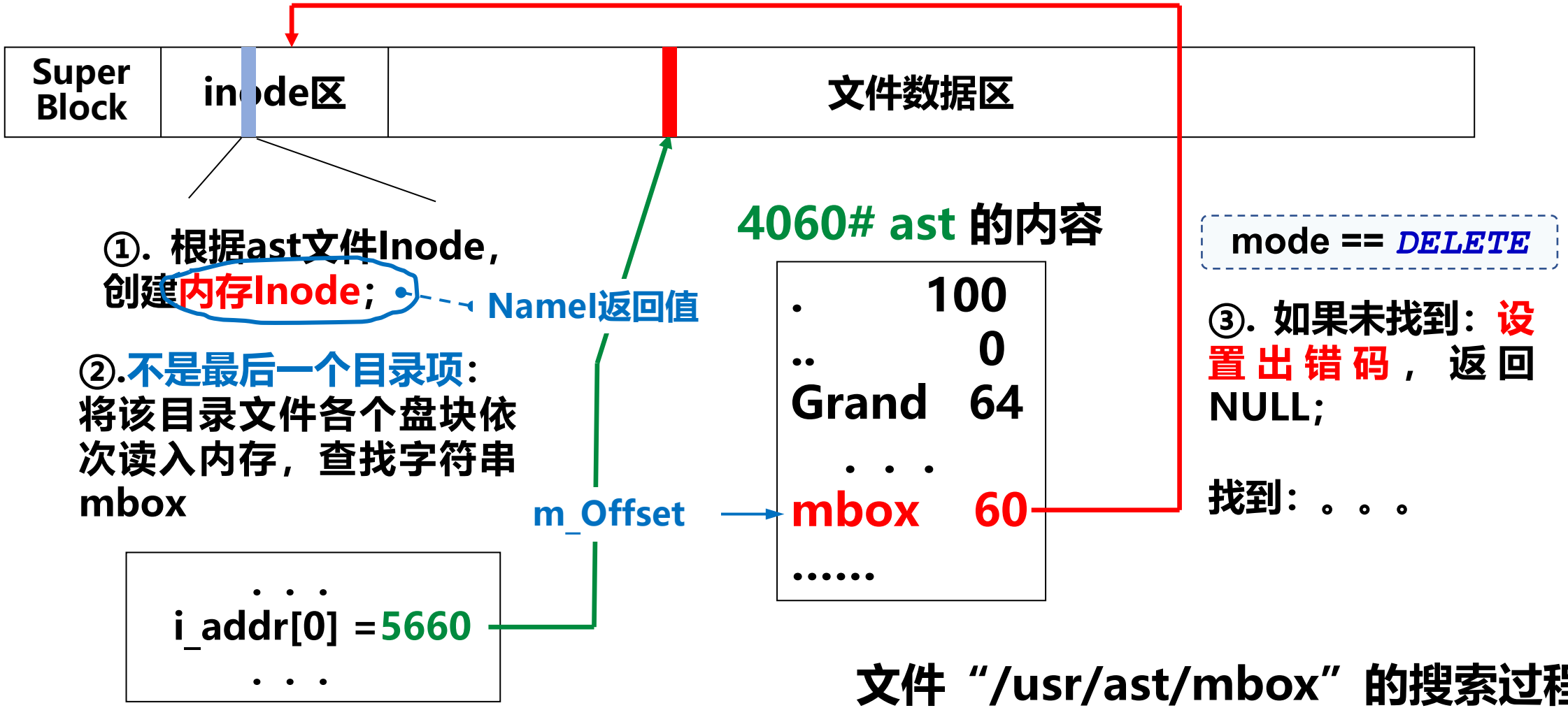
没找到：返回值为NULL





UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```



文件“/usr/ast/mbox”的搜索过程

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```

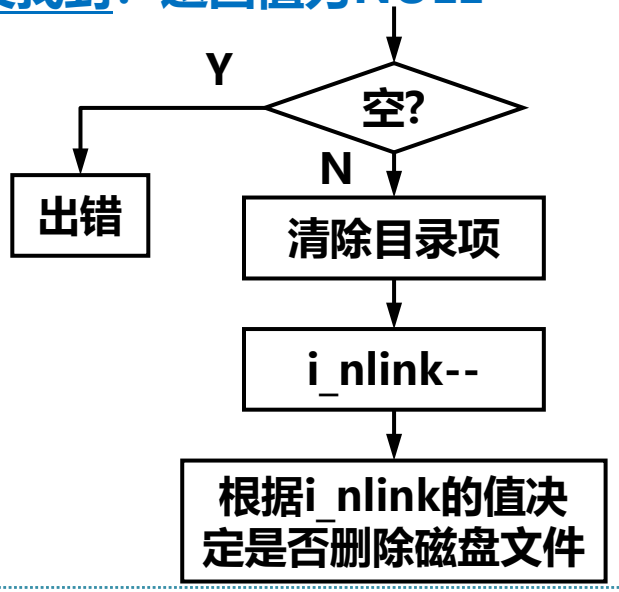
```
enum DirectorySearchMode
{
    OPEN = 0,/* 以打开文件方式搜索目录 */
    CREATE = 1,/* 以新建文件方式搜索目录 */
    DELETE = 2/* 以删除文件方式搜索目录 */
};
```

Unlink

NameI（文件名，FileManager::DELETE）

找到：返回上级目录文件的内存Inode；且m_Offset记录目录项位置

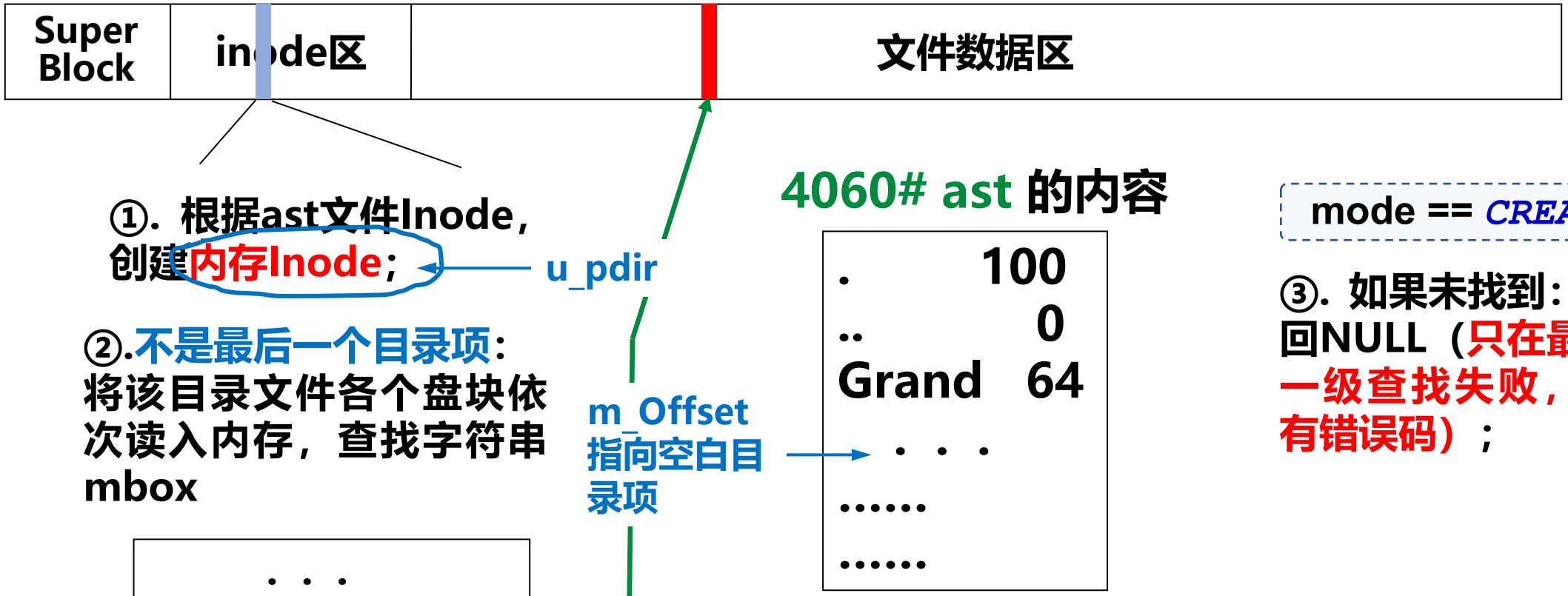
没找到：返回值为NULL



unlink只是取消文件的一条路径，取消最后一条路径时才真正删除一个文件。

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```



文件“/usr/ast/mbox”的搜索过程

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```

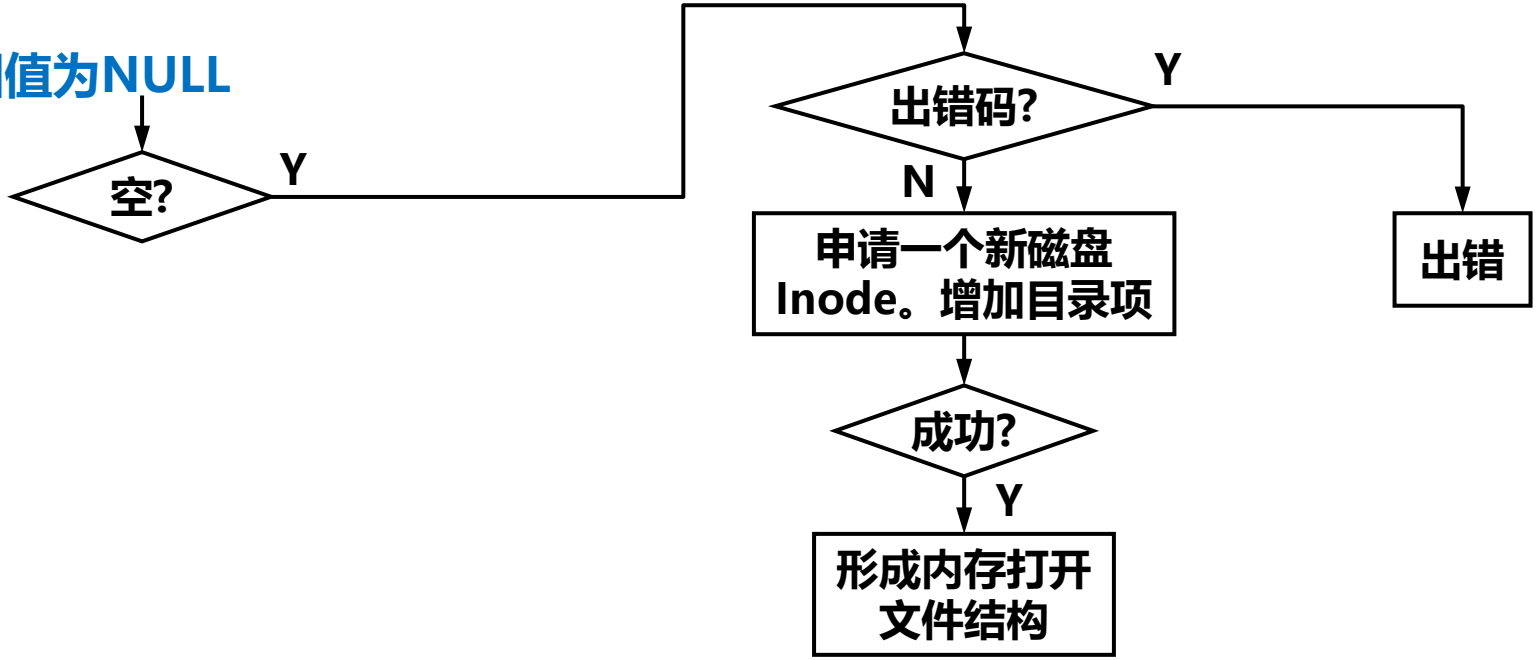
```
enum DirectorySearchMode
{
    OPEN = 0,/* 以打开文件方式搜索目录 */
    CREATE = 1,/* 以新建文件方式搜索目录 */
    DELETE = 2/* 以删除文件方式搜索目录 */
};
```

Creat

NameI (文件名， FileManager::*CREATE*)

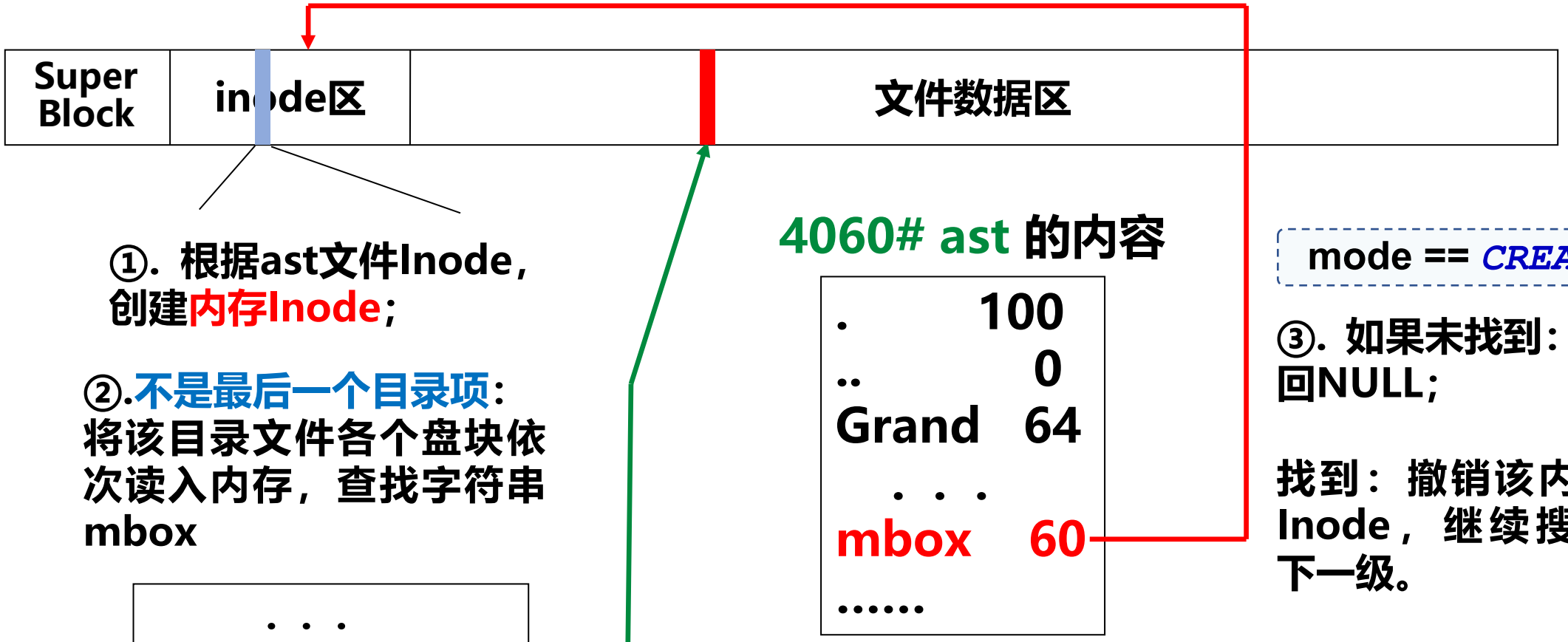
找到： 返回值找到文件的内存Inode;

没找到：返回值为NULL



UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```



文件“/usr/ast/mbox”的搜索过程

UNIX的文件目录

文件“/usr/ast/mbox”的搜索过程

目录检索过程

UNIX的文件目录


```
Inode* FileManager::NameI( char (*func)(), enum DirectorySearchMode mode )
```

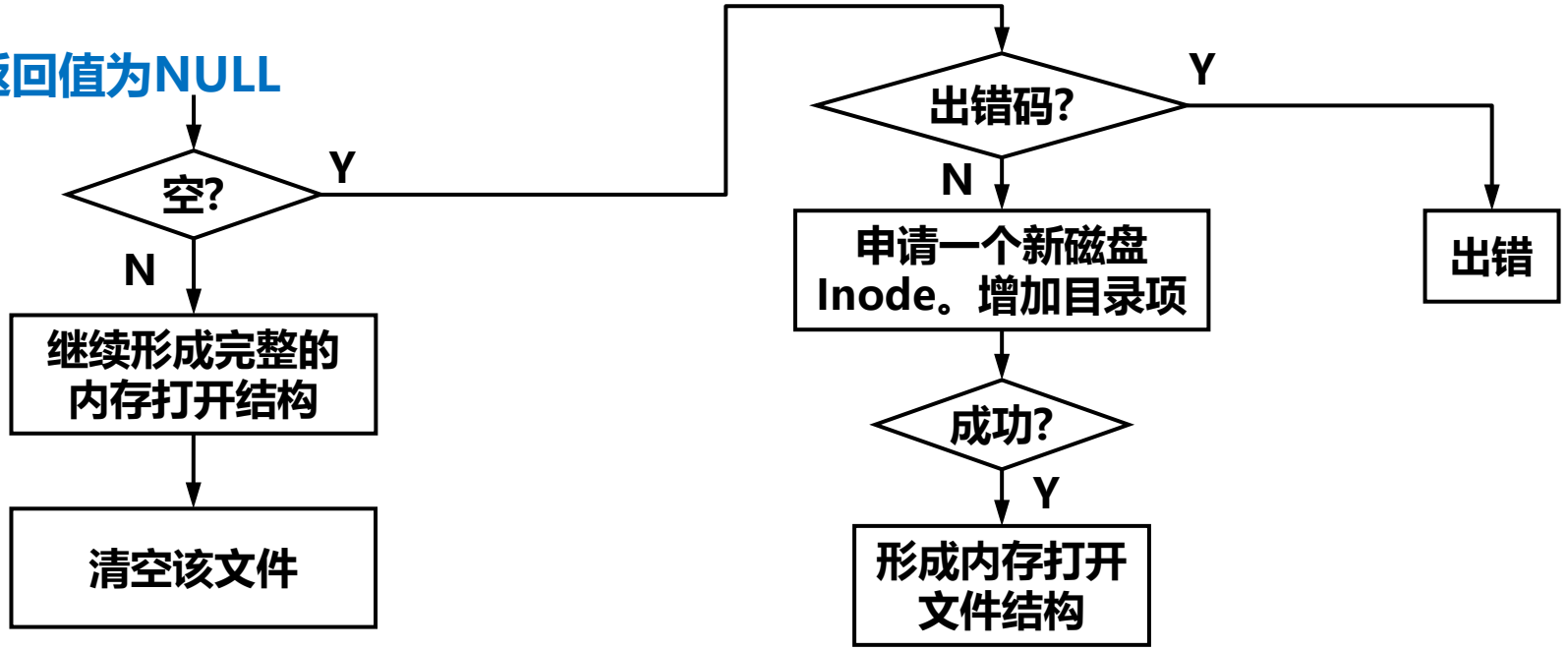
```
enum DirectorySearchMode
{
    OPEN = 0,/* 以打开文件方式搜索目录 */
    CREATE = 1,/* 以新建文件方式搜索目录 */
    DELETE = 2/* 以删除文件方式搜索目录 */
};
```

Creat

NameI（文件名，FileManager::CREATE)

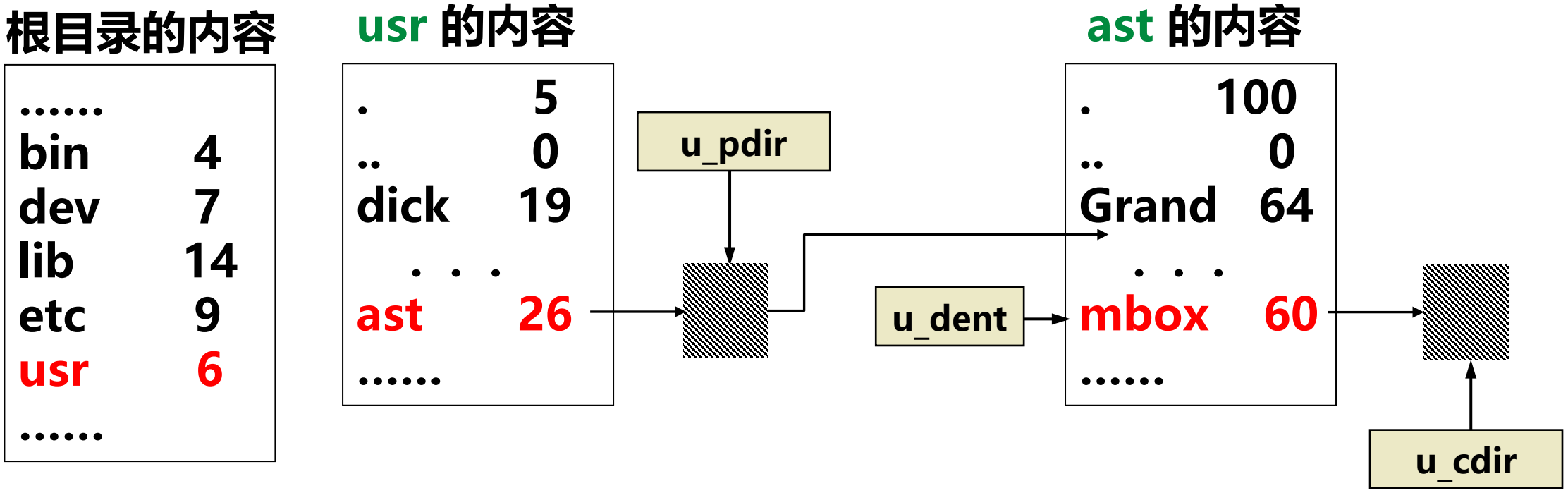
找到：返回值找到文件的内存Inode;

没找到：返回值为NULL



UNIX的文件目录

如果设置“/usr/ast/mbox”为当前工作目录

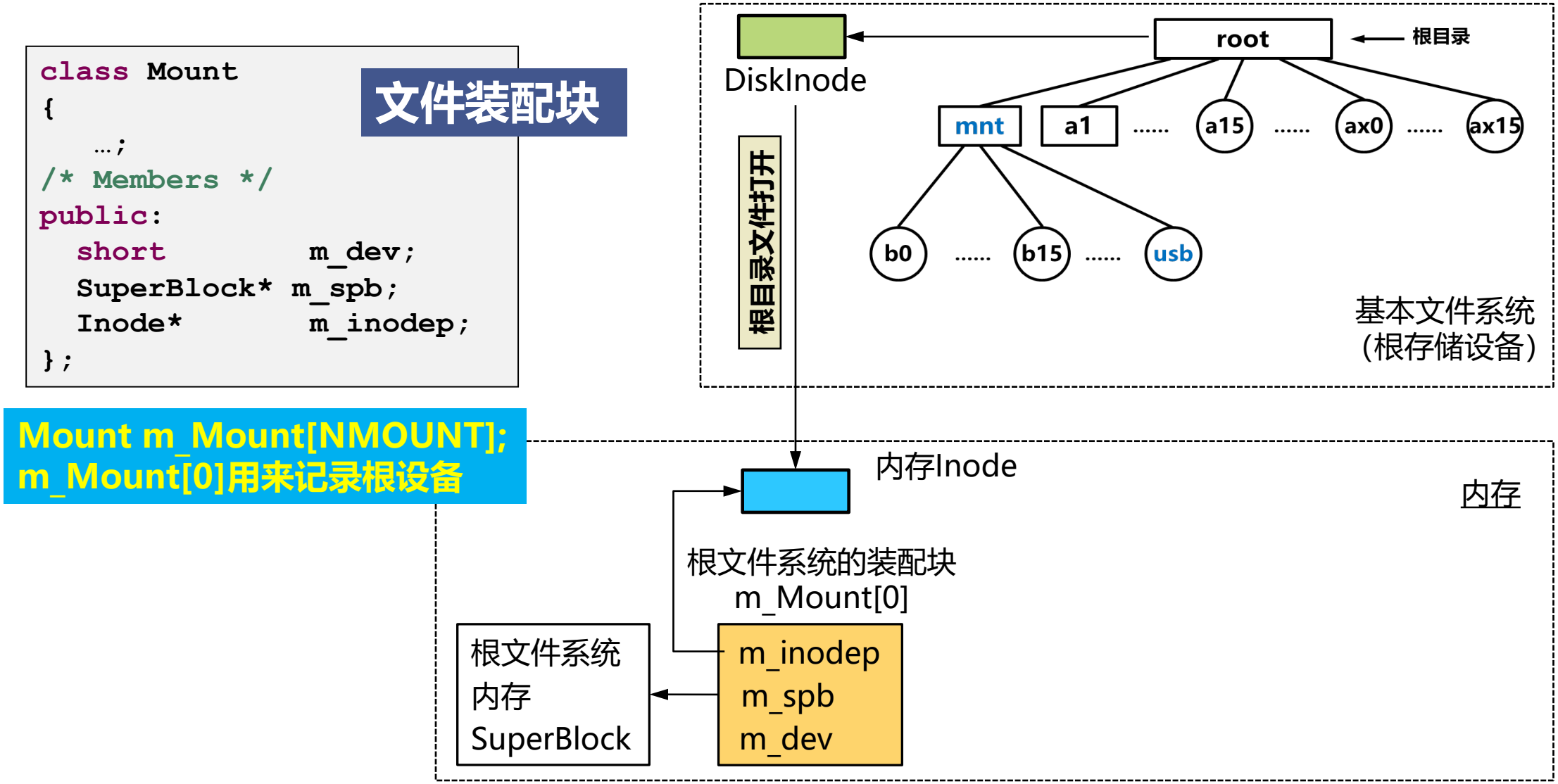


UNIX的文件目录

```
class Mount
{
    …;
/* Members */
public:
   short          m_dev;
   SuperBlock* m_spb;
   Inode*         m_inodep;
};
```

文件装配块

Mount m_Mount[NMOUNT];
m_Mount[0]用来记录根设备



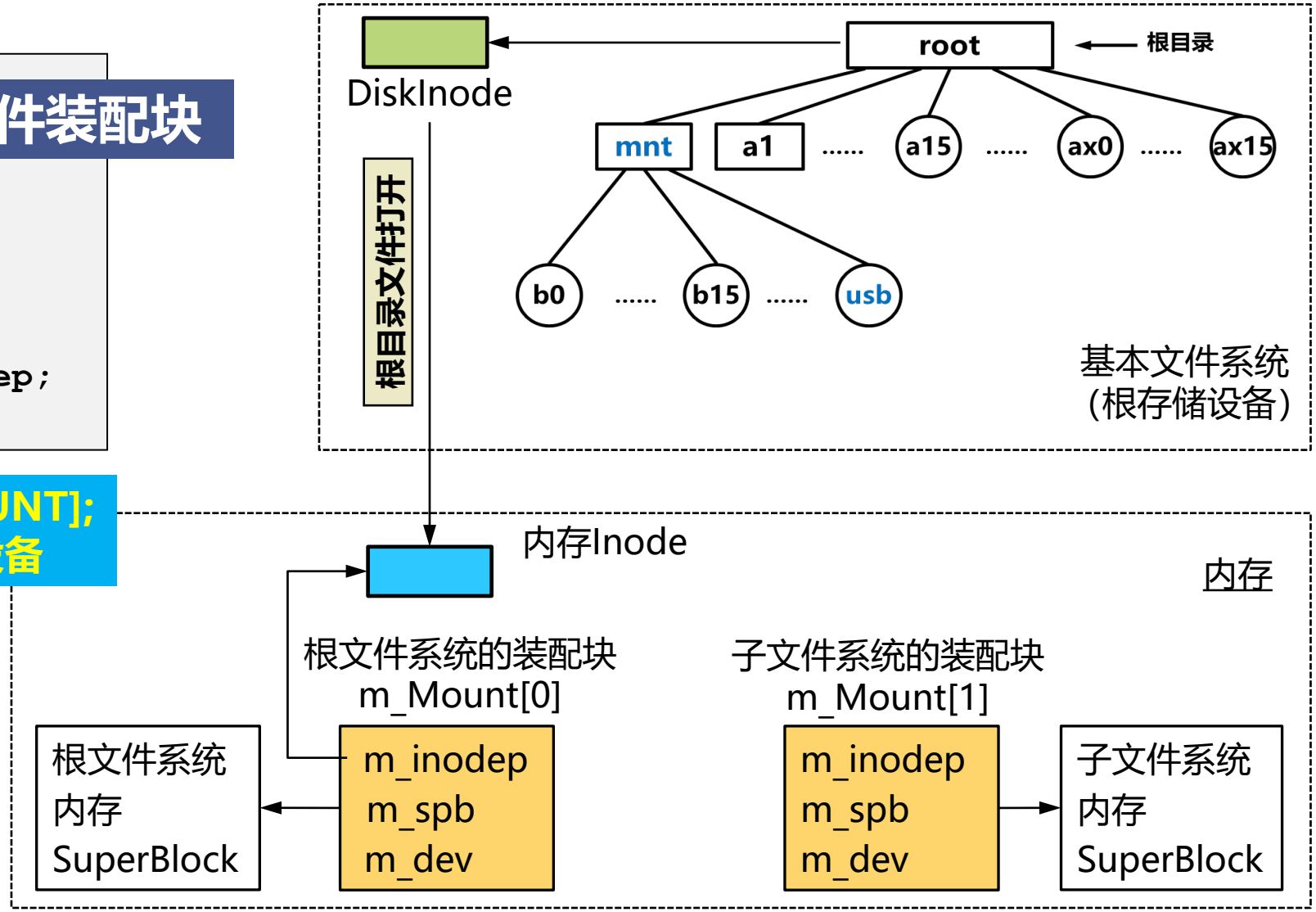
UNIX的文件目录

```
class Mount
{
    …;
/* Members */
public:
   short          m_dev;
   SuperBlock* m_spb;
   Inode*         m_inodep;
};
```

文件装配块

Mount m_Mount[NMOUNT];
m_Mount[0]用来记录根设备

插入一块U盘后



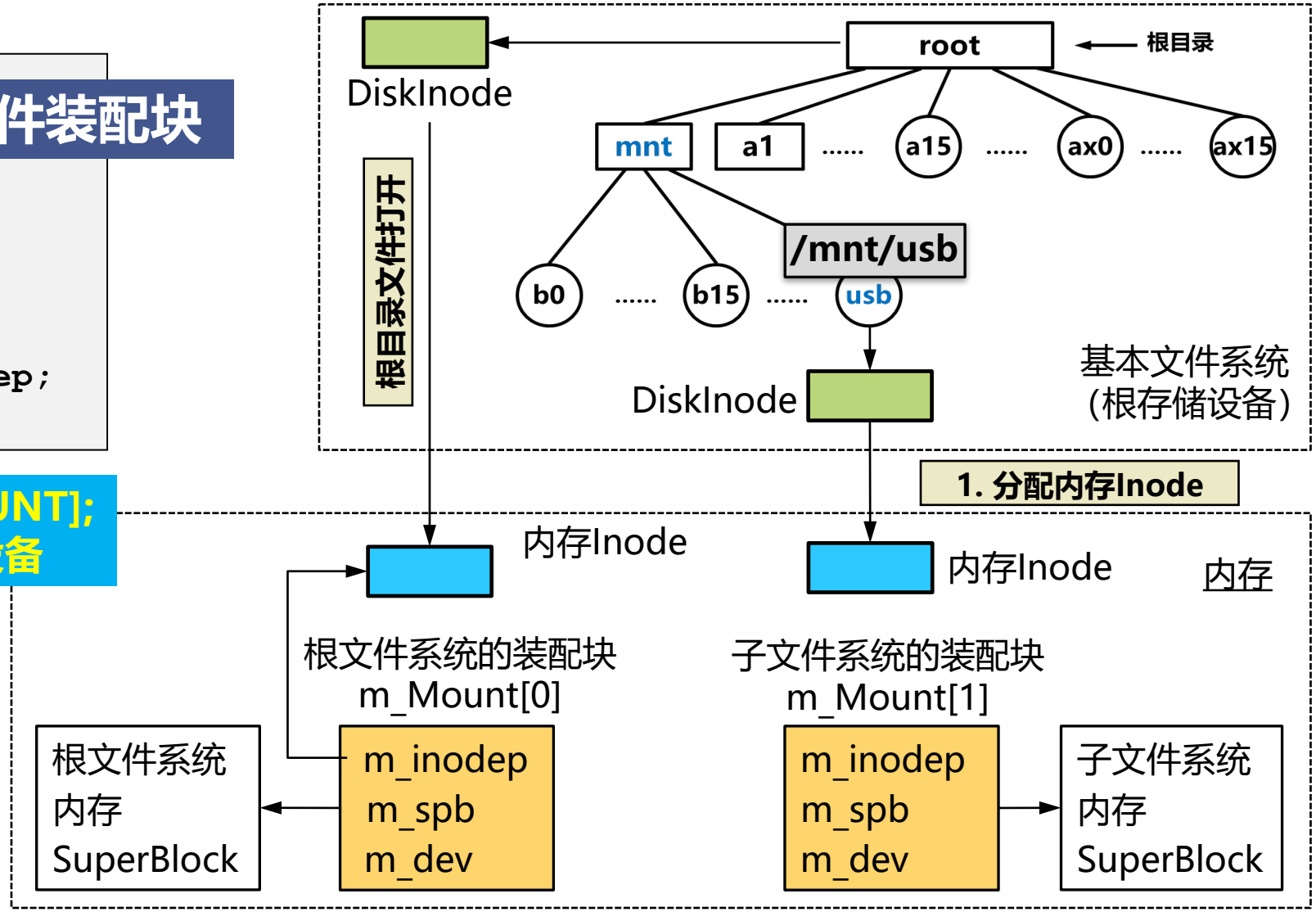
UNIX的文件目录

```
class Mount
{
    …;
/* Members */
public:
   short          m_dev;
   SuperBlock* m_spb;
   Inode*         m_inodep;
};
```

文件装配块

Mount m_Mount[NMOUNT];
m_Mount[0]用来记录根设备

将U盘挂载到
/mnt/usb



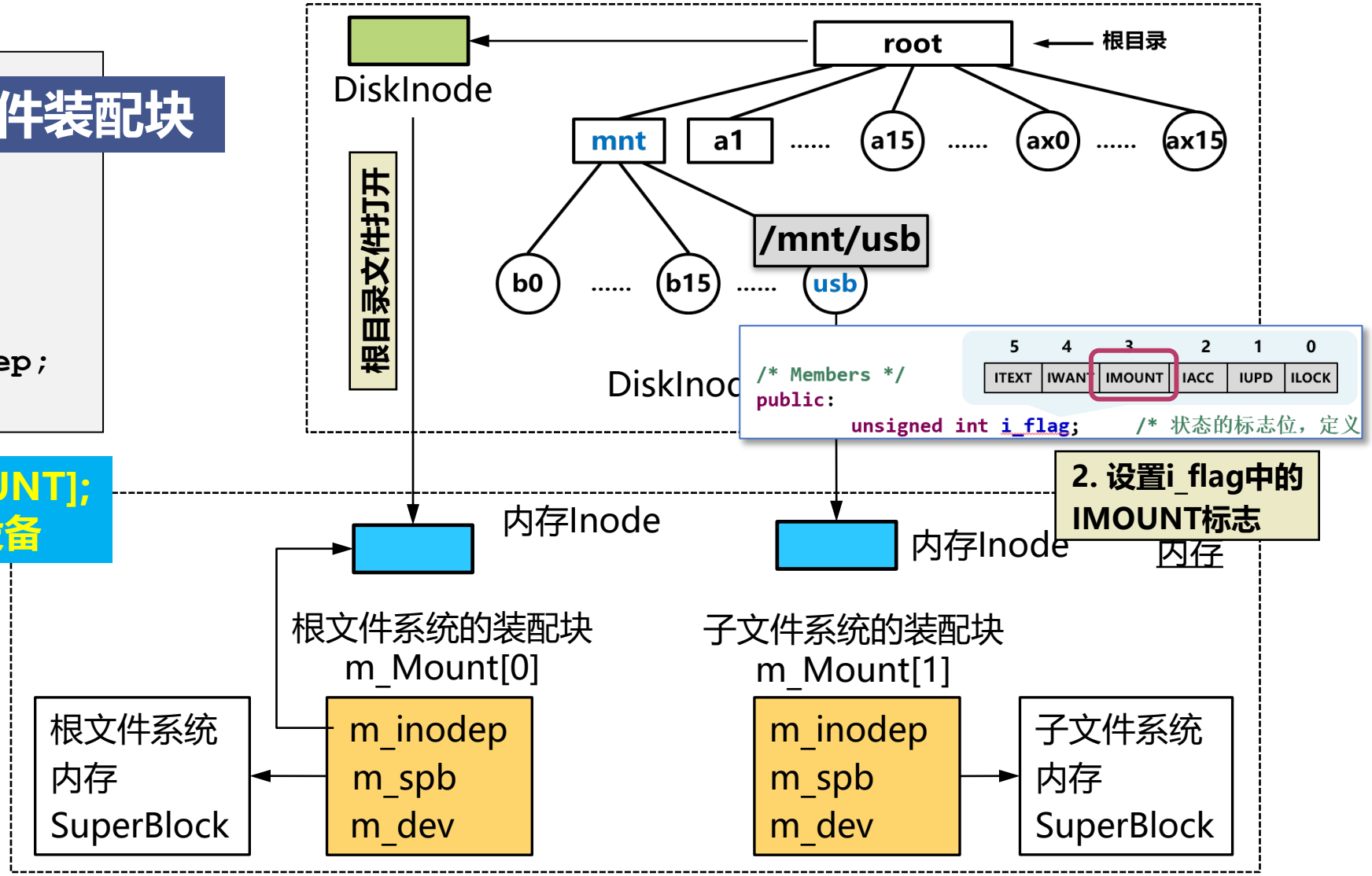
UNIX的文件目录

```
class Mount
{
    …;
/* Members */
public:
   short          m_dev;
   SuperBlock* m_spb;
   Inode*         m_inodep;
};
```

文件装配块

Mount m_Mount[NMOUNT];
m_Mount[0]用来记录根设备

将U盘挂载到
/mnt/usb



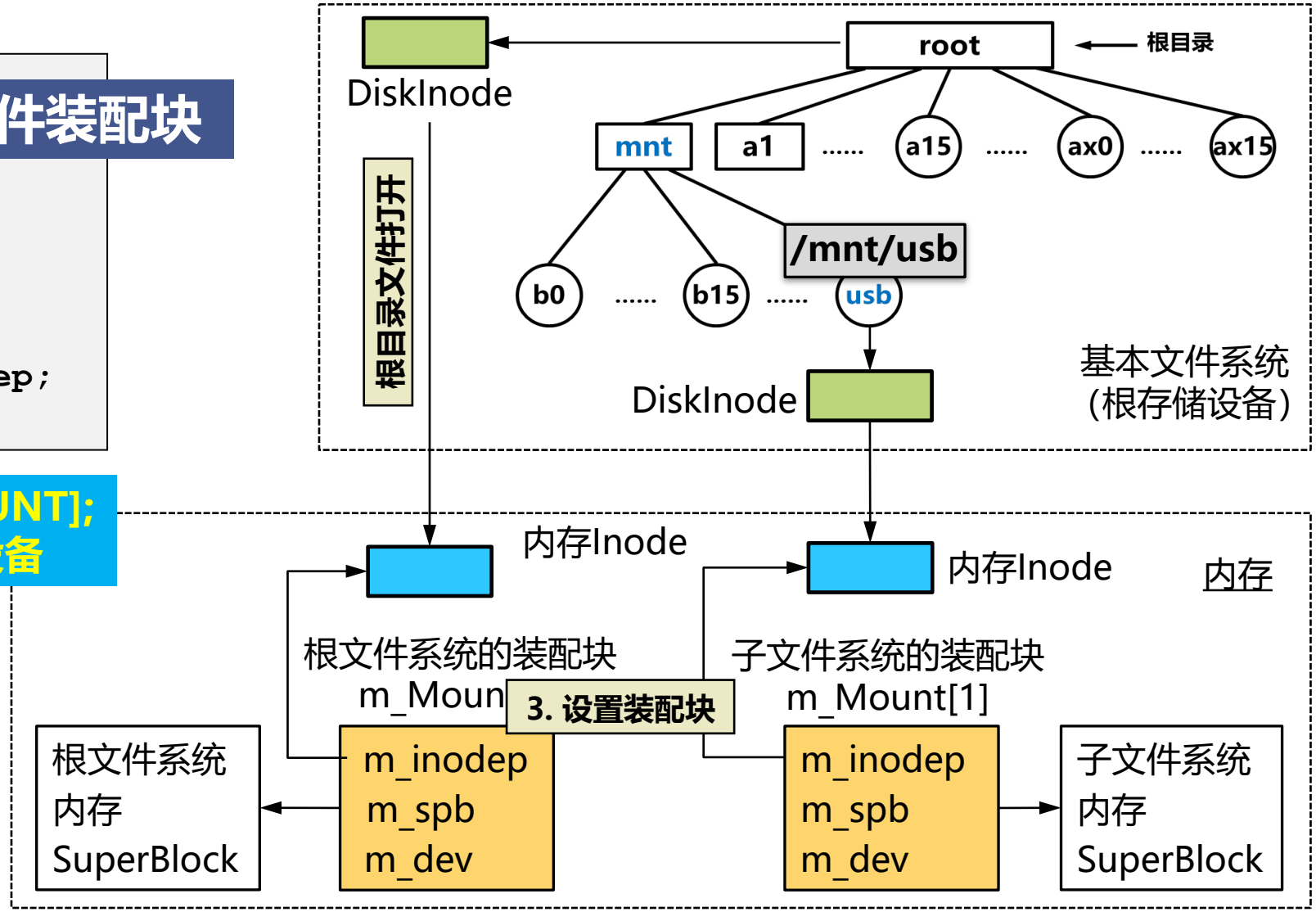
UNIX的文件目录


```
class Mount
{
    …;
/* Members */
public:
   short          m_dev;
   SuperBlock*  m_spb;
   Inode*         m_inodep;
};
```

文件装配块

Mount m_Mount[NMOUNT];
m_Mount[0]用来记录根设备

将U盘挂载到
/mnt/usb



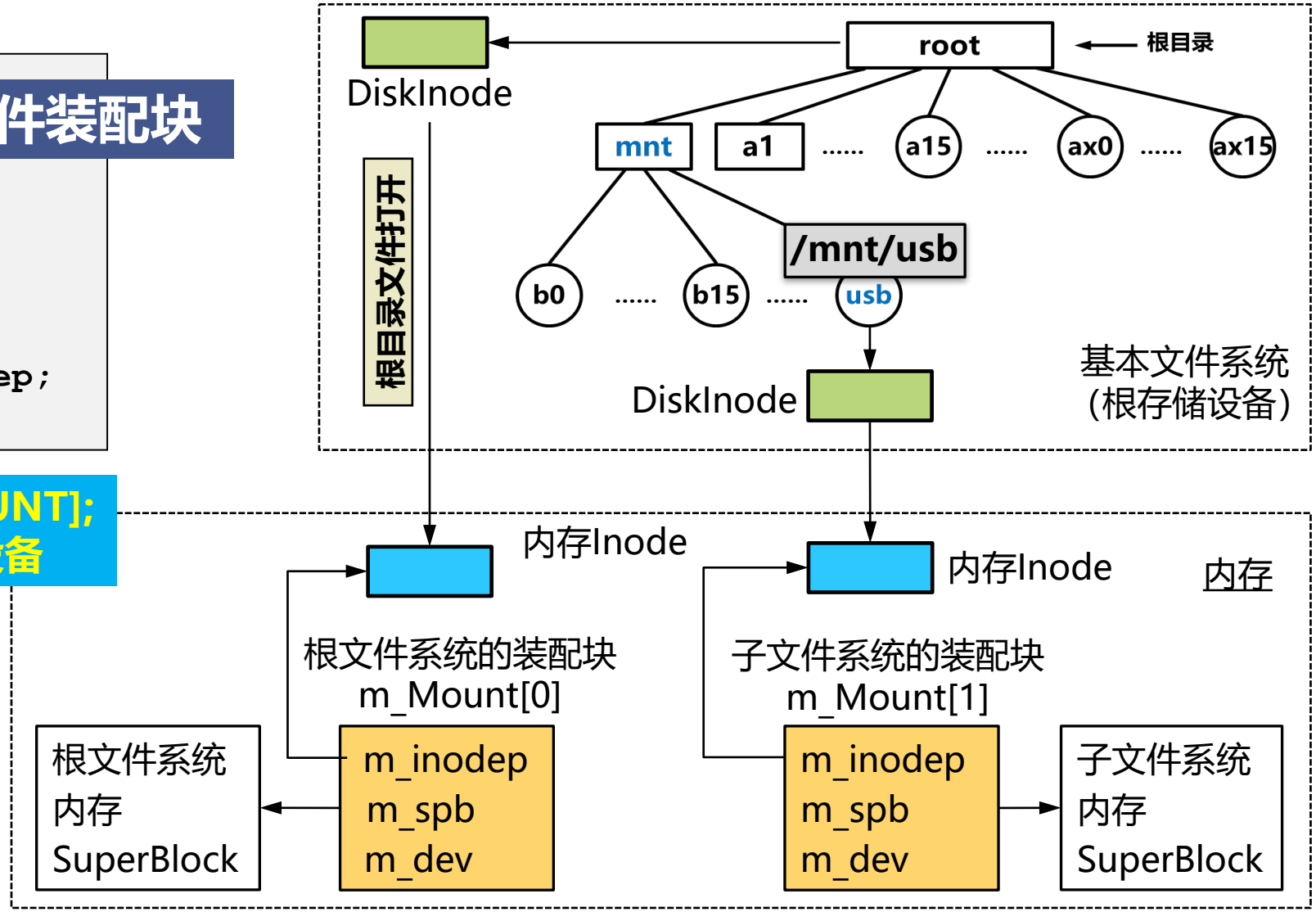
UNIX的文件目录

```
class Mount
{
    …;
/* Members */
public:
   short          m_dev;
   SuperBlock* m_spb;
   Inode*         m_inodep;
};
```

文件装配块

Mount m_Mount[NMOUNT];
m_Mount[0]用来记录根设备

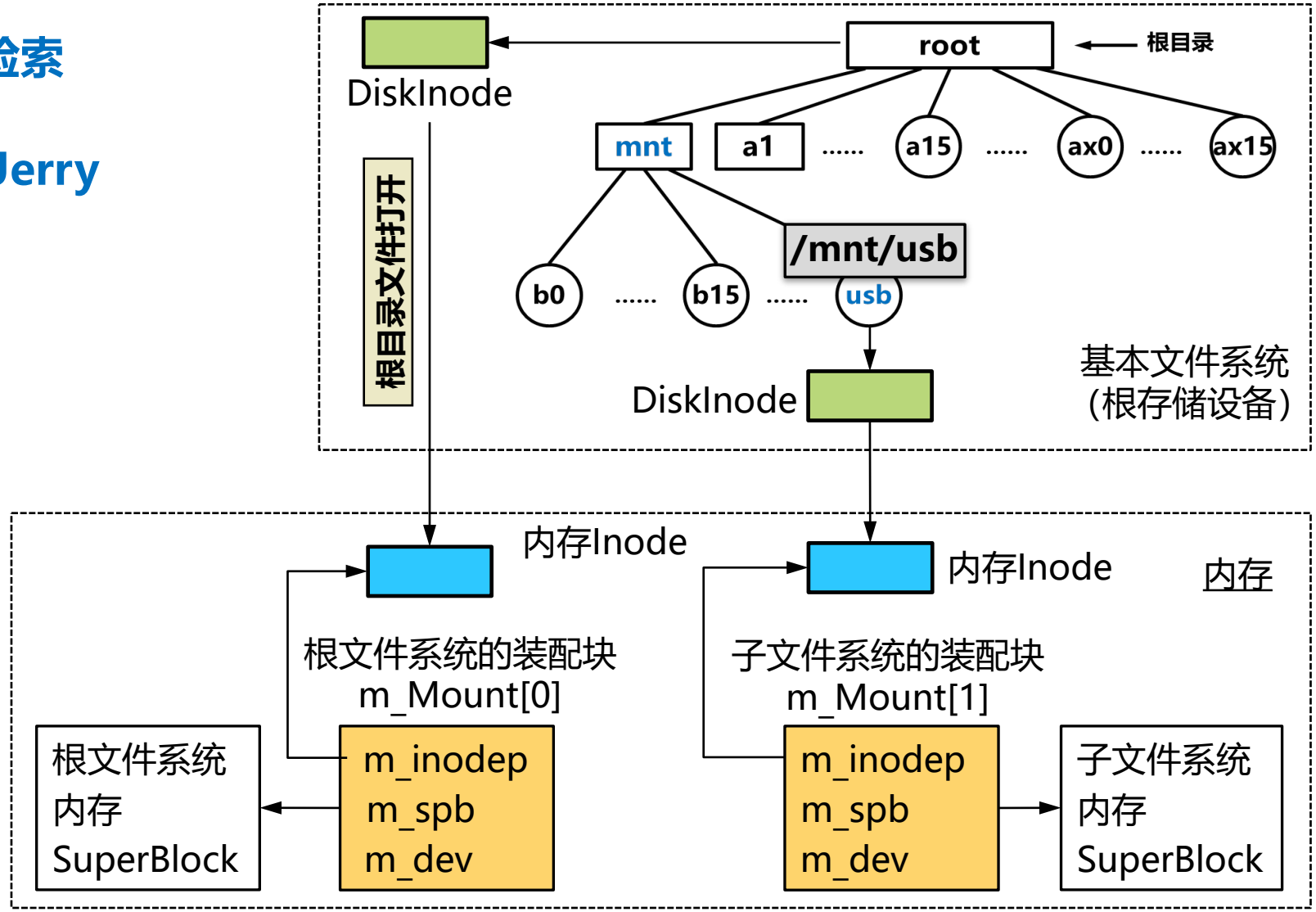
将U盘挂载到
/mnt/usb



UNIX的文件目录

子文件系统上的目录检索

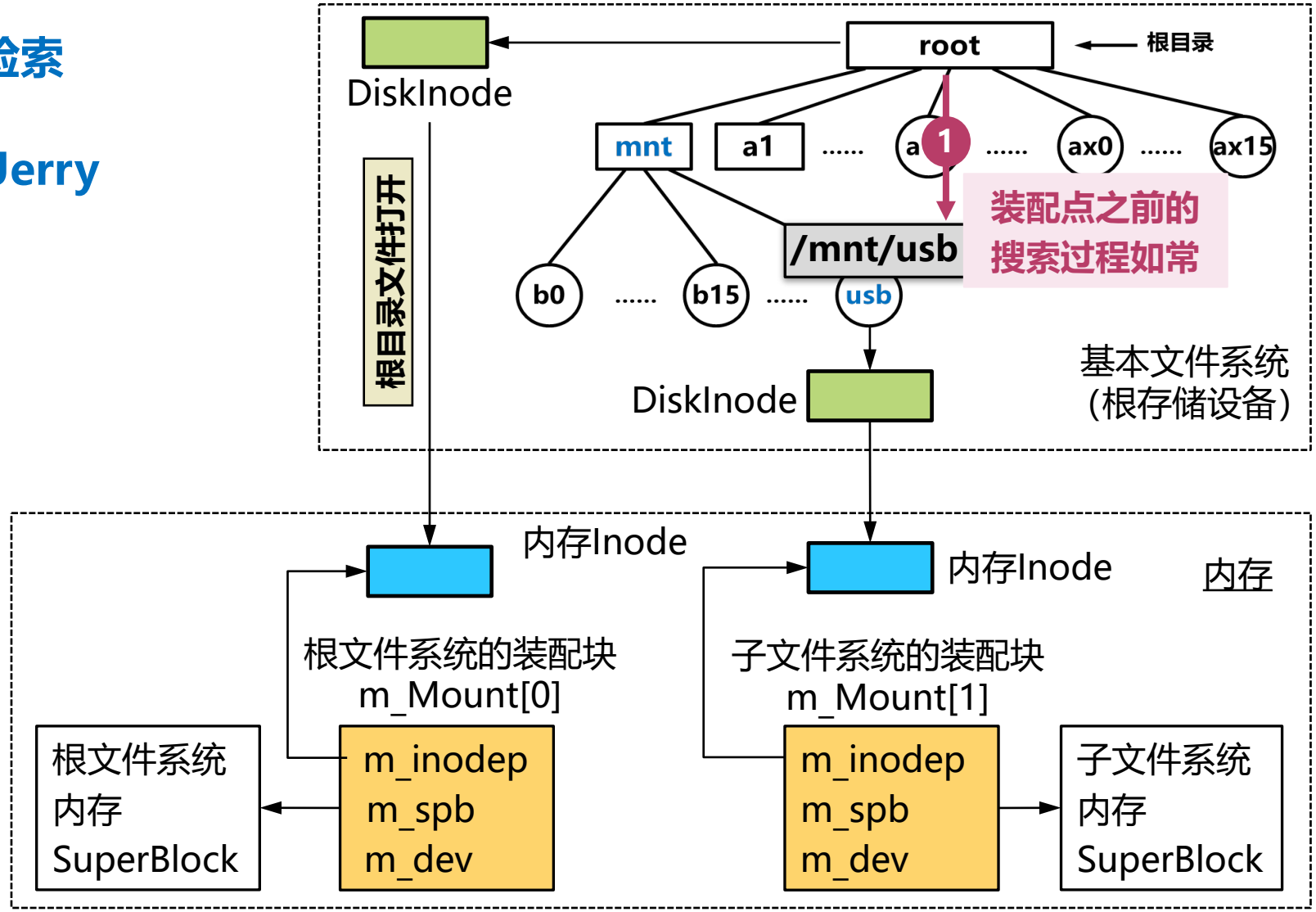
例：/mnt/usb/ast/Jerry



UNIX的文件目录

子文件系统上的目录检索

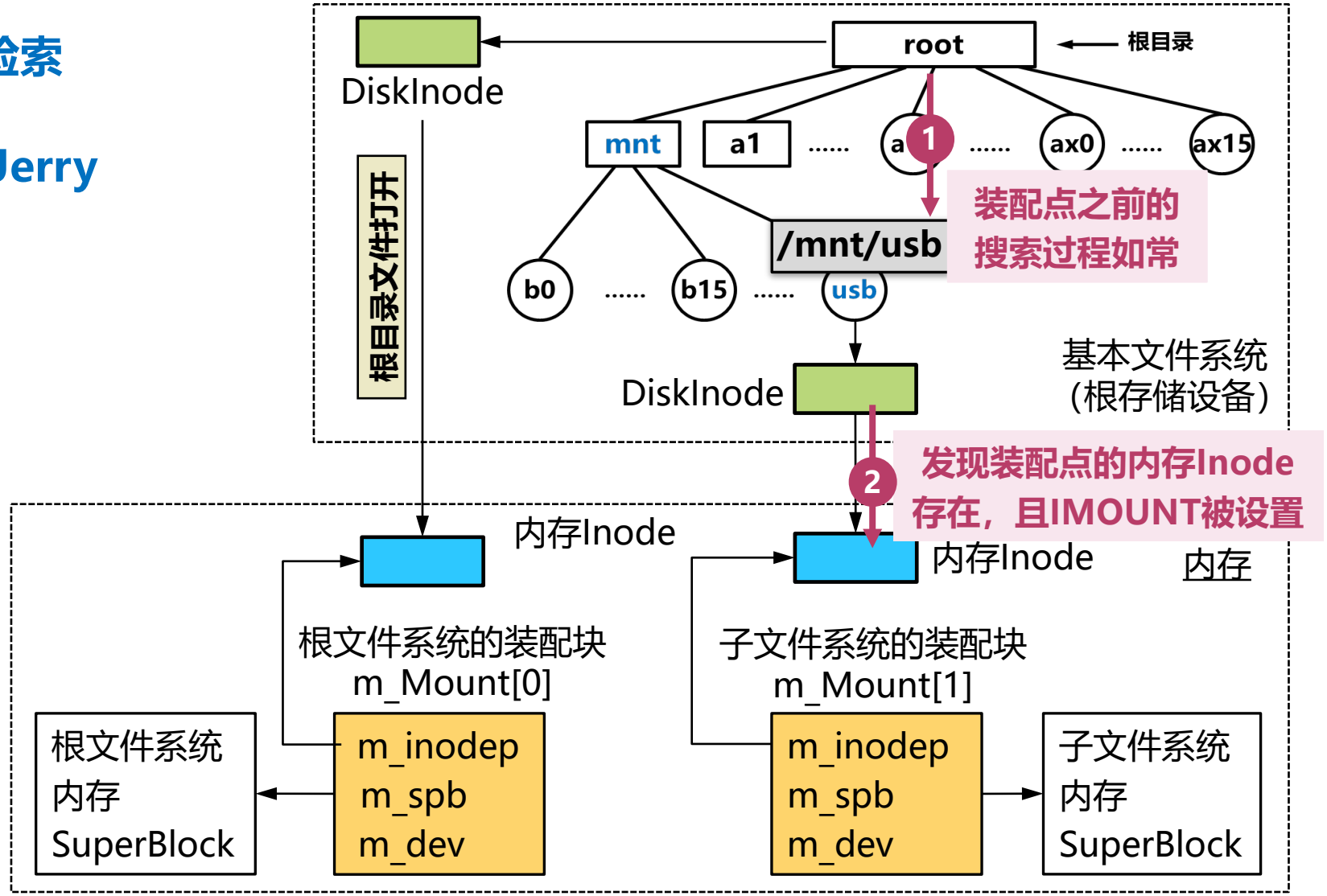
例：/mnt/usb/ast/Jerry



UNIX的文件目录

子文件系统上的目录检索

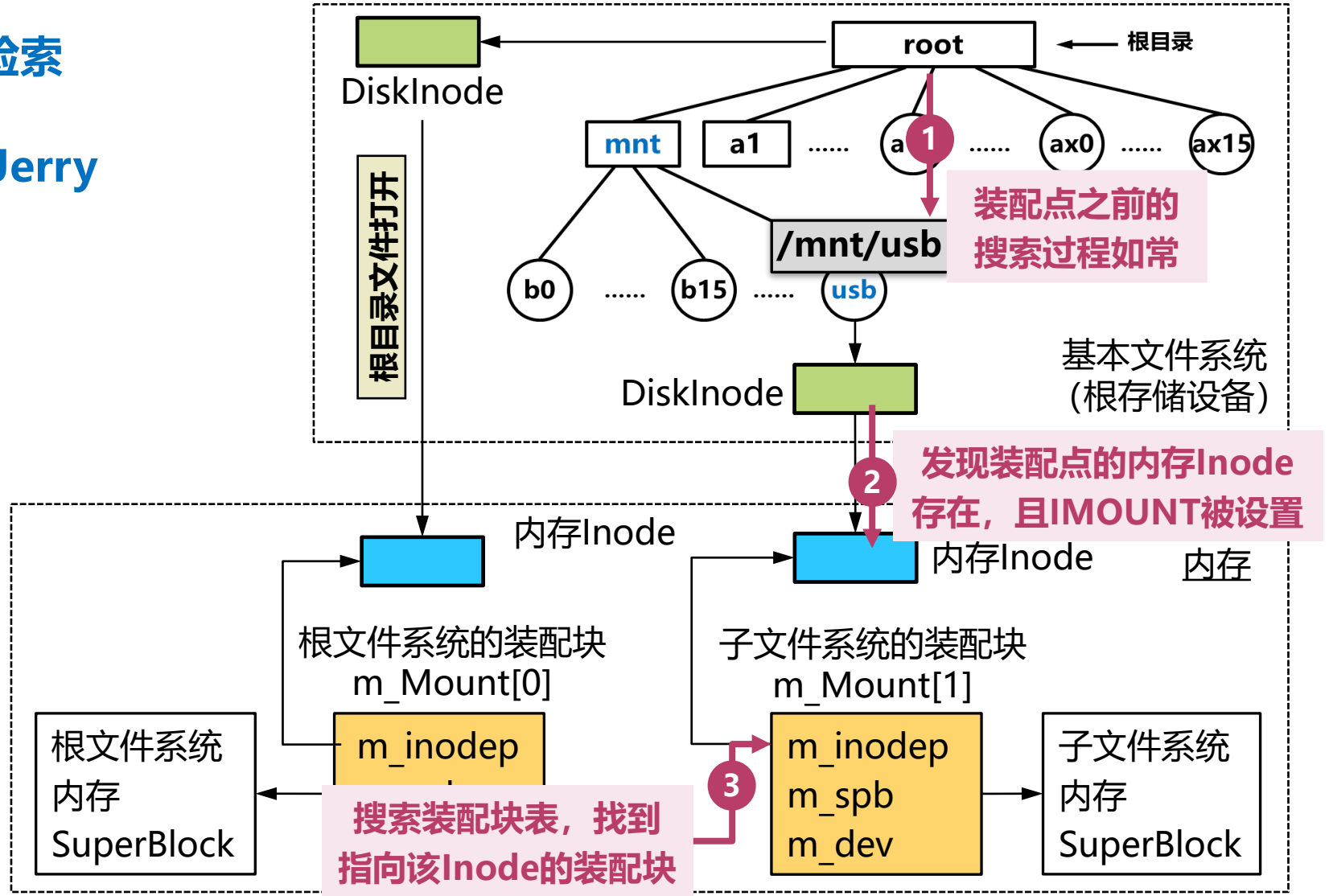
例：/mnt/usb/ast/Jerry



UNIX的文件目录

子文件系统上的目录检索

例：/mnt/usb/ast/Jerry



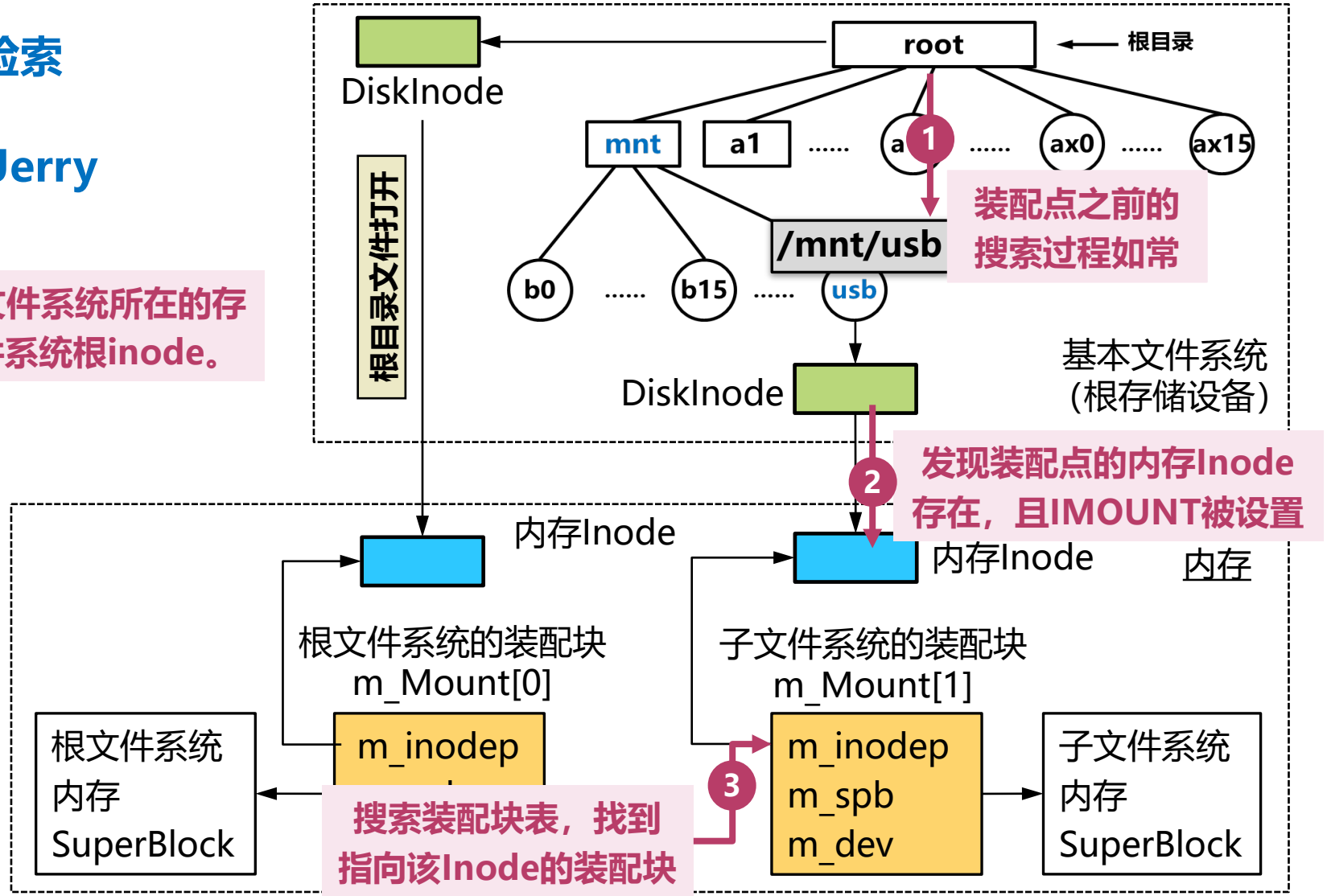
UNIX的文件目录

子文件系统上的目录检索

例：/mnt/usb/ast/Jerry

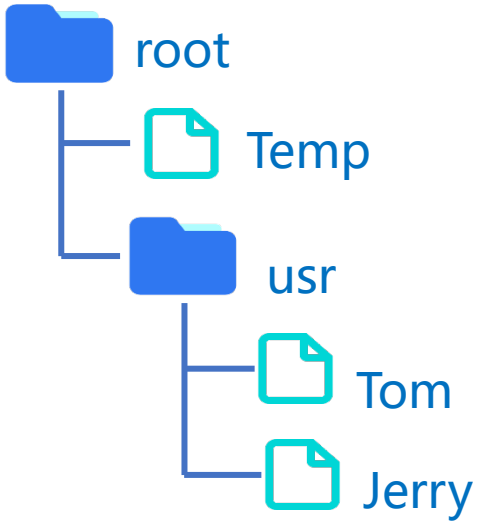
4 根据m_dev找到该子文件系统所在的存储设备，并获得子文件系统根inode。

5 由此开始搜索子文件系统目录结构，直到找到/ast/Jerry为止。



例题

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。

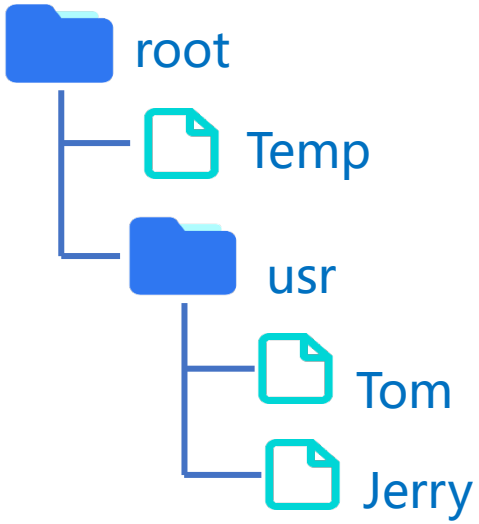


（1）请绘制出所有目录文件的内容。文件Temp占用的盘块号x = **4000**。

文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

例题

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（1）请绘制出所有目录文件的内容。文件Temp占用的盘块号x = **4000**。

root

.	1
Temp	10
usr	8

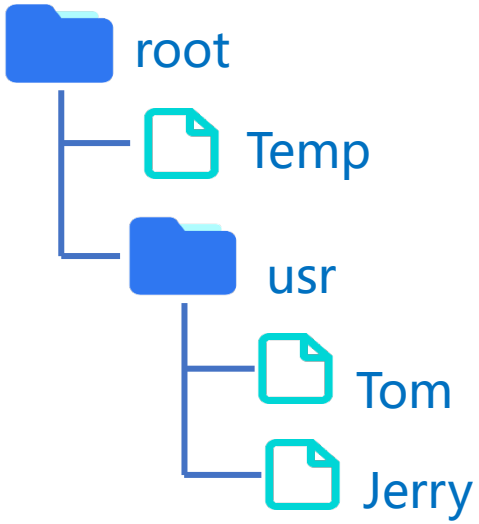
usr

.	8
..	1
Tom	10
Jerry	15

（2）进程pa执行了open(“/usr/Tom”，O_RDWR)，请简述该操作中的文件搜索过程。

例题

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（1）请绘制出所有目录文件的内容。文件Temp占用的盘块号x = **4000**。

root

.	1
Temp	10
usr	8

usr

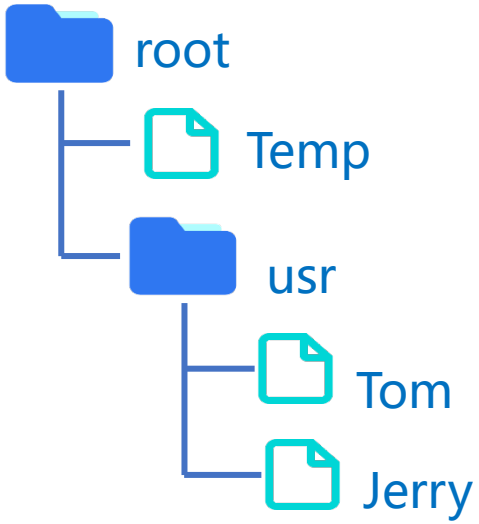
.	8
..	1
Tom	10
Jerry	15

（2）进程pa执行了open(“/usr/Tom”，O_RDWR)，请简述该操作中的文件搜索过程。

1. 根据1#Inode创建root文件的内存Inode；
2. 读入root文件，逐条记录搜索usr，找到usr目录文件的磁盘Inode号8；
3. 根据8#Inode创建usr文件的内存Inode；
4. 读入usr文件，诸条记录搜索Tom，找到Tom目录文件的磁盘Inode号10；
5. 查找成功，根据10#Inode创建Tom文件的内存Inode，返回指向该Inode的指针。

例题

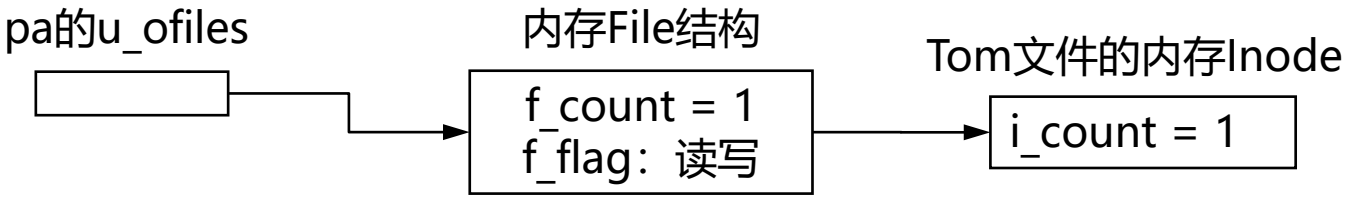
V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

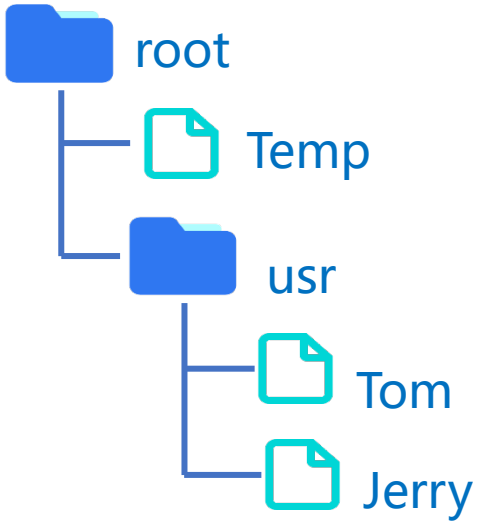
（3）如果进程pa在成功打开Tom文件后，创建了进程pb，pb上台后，以只读的方式打开了文件Temp，请绘制出此时两个文件的内存打开结构。

父进程打开Tom成功：



例题

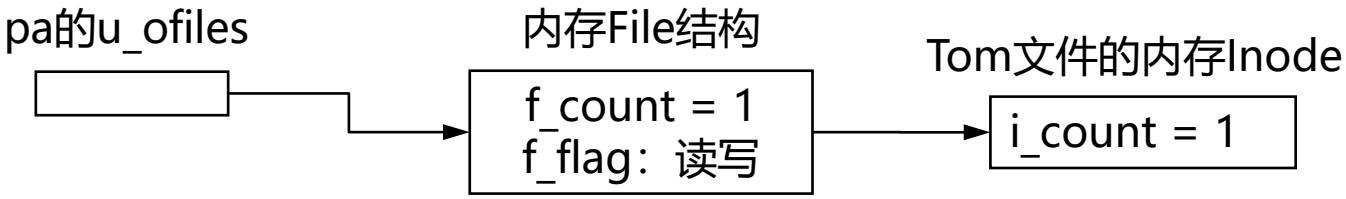
V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



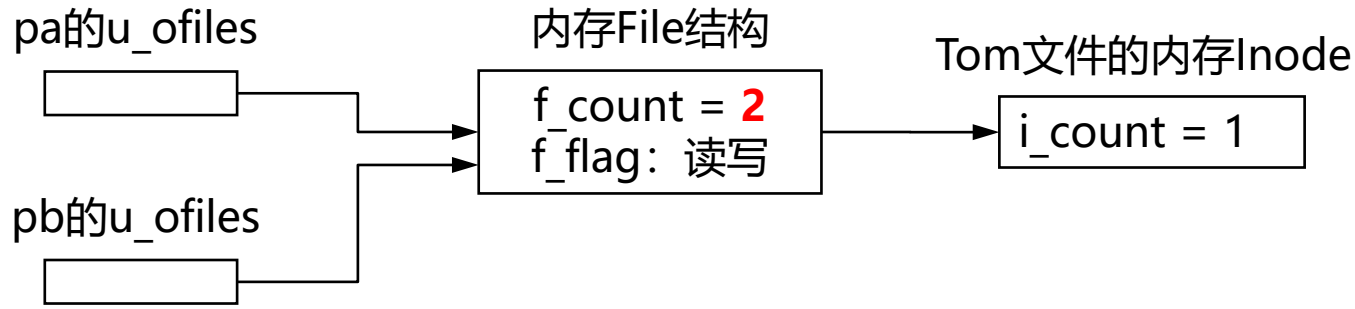
文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（3）如果进程pa在成功打开Tom文件后，创建了进程pb，pb上台后，以只读的方式打开了文件Temp，请绘制出此时两个文件的内存打开结构。

父进程打开Tom成功：

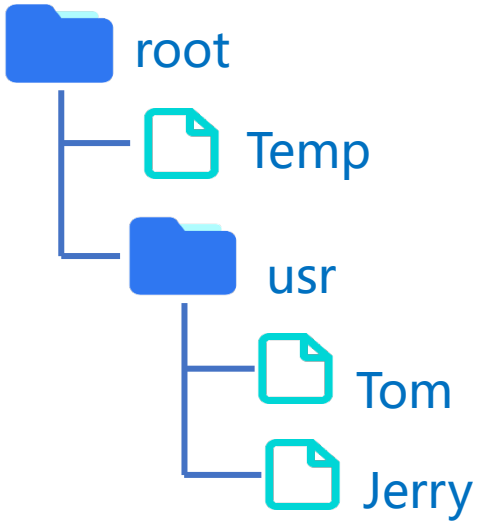


父进程创建pb成功：



例题

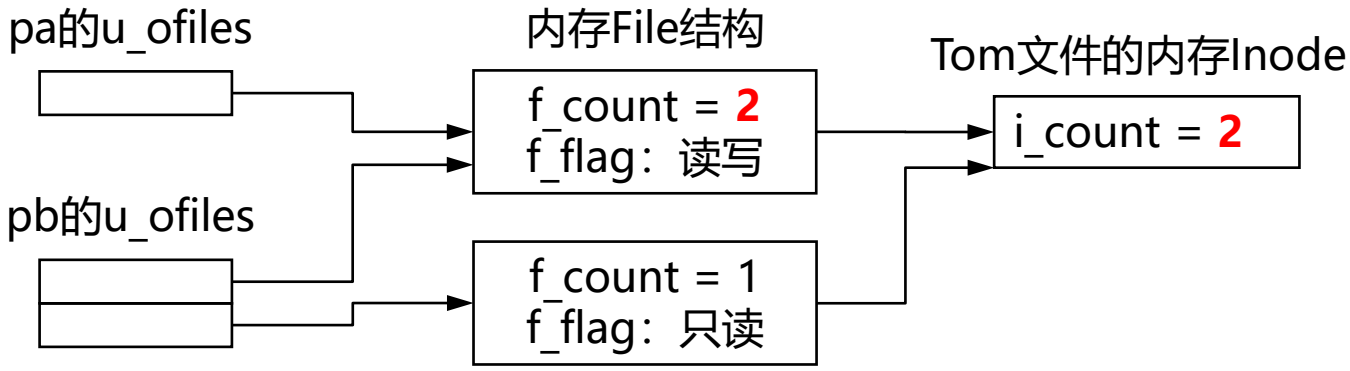
V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（3）如果进程pa在成功打开Tom文件后，创建了进程pb，pb上台后，以只读的方式打开了文件Temp，请绘制出此时两个文件的内存打开结构。

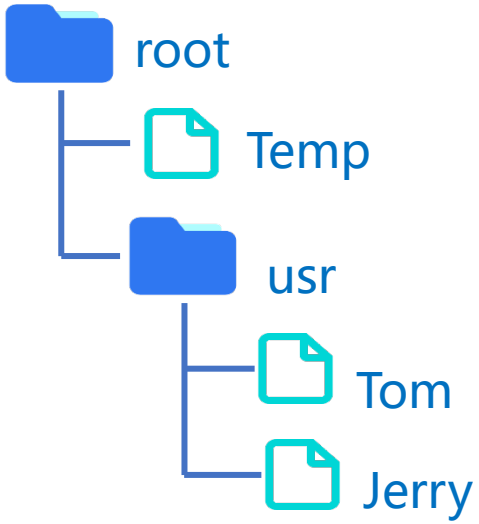
pb打开Temp成功：



pb可以使用两个不同的文件句柄以不同的权限不同的读写指针位置操作同一个文件

例题

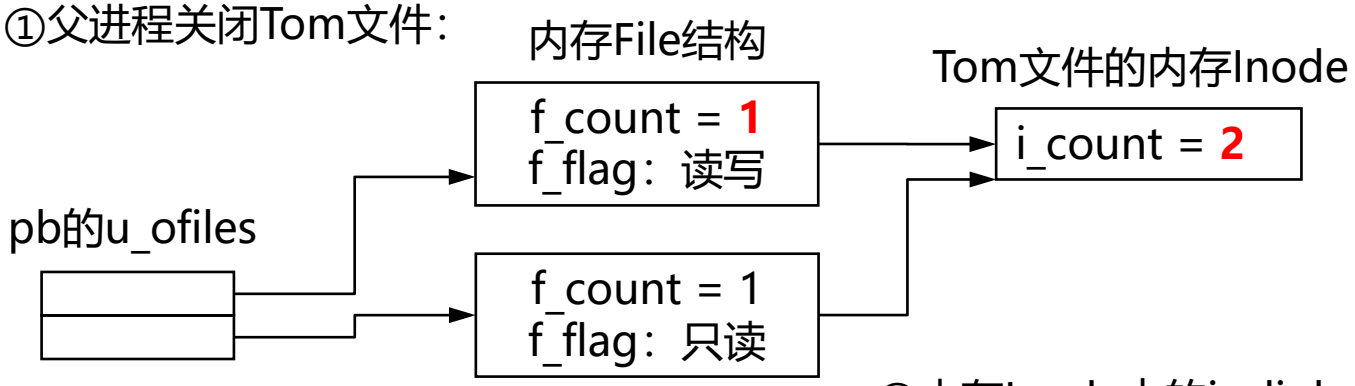
V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

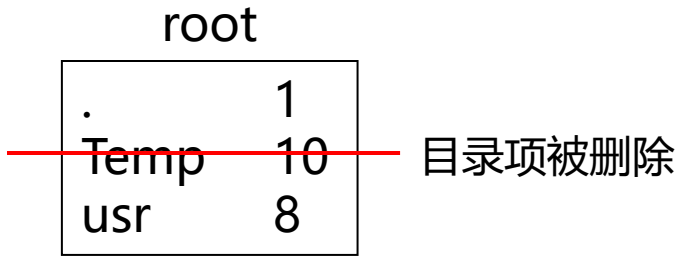
（4）此后进程pa先关闭了Tom文件，然后执行了unlink（“/Temp”），哪些数据结构会发生变化?

①父进程关闭Tom文件：



③内存Inode中的i_nlink--；待子进程关闭该文件，撤销内存Inode时，写回磁盘。

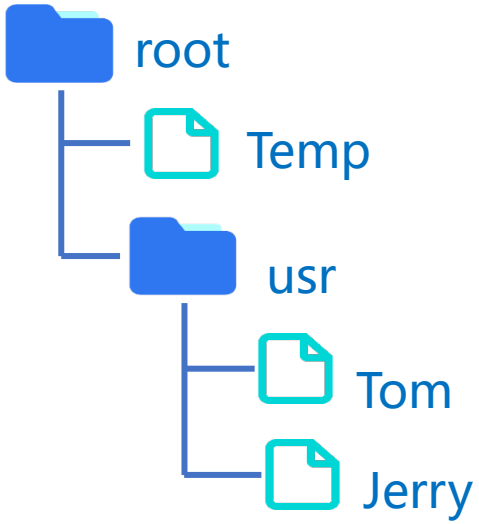
②父进程执行了unlink（“/Temp”）



！！！父进程删除了子进程之前打开文件的路径，但是不影响子进程对文件的使用（对文件的使用都是通过文件句柄）

例题

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（5）进程pa继续执行unlink（“/usr/Tom”）：

root

.	1
~~Temp~~	~~10~~
usr	8

usr

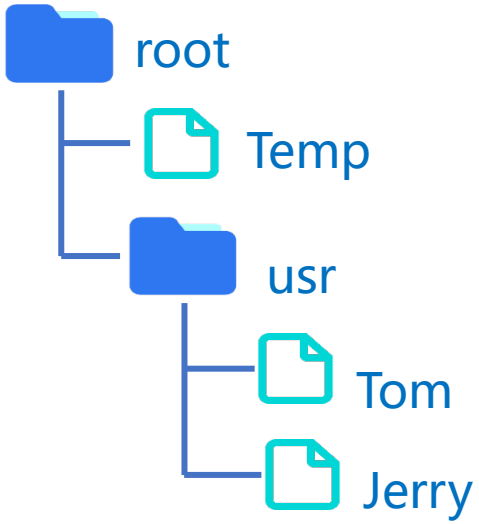
.	8
..	1
~~Tom~~	~~10~~
Jerry	15

内存Inode中的i_nlink--；
待子进程关闭该文件，撤销内存Inode时，写回磁盘。

！！！虽然i_nlink减为0，但是内存打开结构还在，文件并未物理删除。

例题

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（6）如果此时内存SuperBlock副本中磁盘Inode栈和空闲盘块栈的内容如下图所示，子进程继续关闭两个文件，哪些数据结构会发生变化？

磁盘Inode栈

```
s_ninode：85
s_inode[0]：12
……
s_inode[84]：20
```

空闲盘块栈

```
s_nfree：100
s_free[0]：1000
……
……
s_free[99]：5000
```

子进程关闭两个文件，撤销全部内存打开结构。撤销内存inode时，发现i_nlink减到0，则删除该文件。先释放4000号盘块，再释放10号Inode节点。

磁盘Inode栈

```
s_ninode：86
s_inode[0]：12
……
s_inode[84]：20
s_inode[84]：10
```

空闲盘块栈

```
s_nfree：1
s_free[0]：4000
```

4000#盘块的前101个字

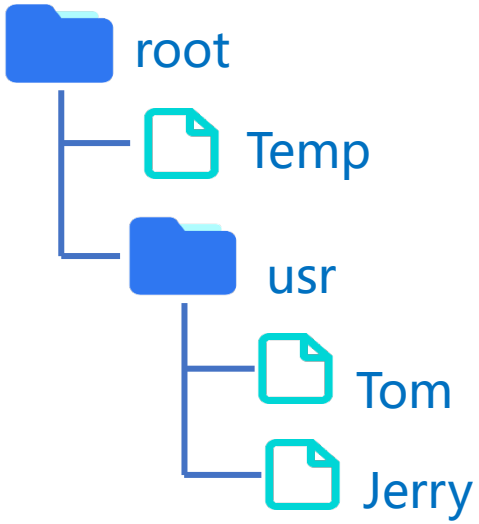
```
s_nfree：100
s_free[0]：1000
……
……
s_free[99]：5000
```

！！！更改后的内存SuperBlock将在文件系统卸载时写回磁盘。

例题

目录检索过程

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

(7) 子进程继续创建一个新的文件“/Hello”，并向文件中写入“Hello”，哪些数据结构会发生变化?

① 分配Inode节点：

磁盘Inode栈

s_ninode：85
s_inode[0]：12
……
s_inode[84]：20

10#Inode节点被重新分配

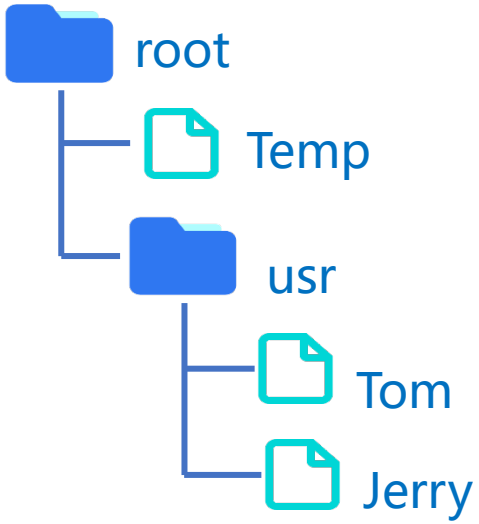
② 添加目录项：

root

.	1
Hello	10
usr	8

例题

V6++文件系统的目录结构及各文件的基本信息入下，请回答下列问题。



文件名	索引节点号	占用的盘块号
root	1	2000
usr	8	2085
Tom	10	4000
Jerry	15	3000 ~ 3010
Temp	10	x

（7）子进程继续创建一个新的文件“/Hello”，并向文件中写入“Hello”，哪些数据结构会发生变化?

③ 分配一个新的盘块：　空闲盘块栈

s_nfree：100
s_free[0]：1000
……
……
s_free[99]：5000

4000#盘块被重新分配，前101个字读入SuperBlock

④ “Hello”写完后，i_addr数组：

0	4000
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

！！！更改后的内存Inode将在文件关闭时写回磁盘。

本节小结

1　了解文件系统磁盘空间管理的一般方法

2　掌握UNIX磁盘存储空间的成组链接管理

阅读教材：262页 ~ 278页

E20：文件管理（UNIX文件系统的综合练习）