第10章 Linux文件系统

10.1 特点与文件类别

10.2 虚拟文件系统VFS

10.3 进程的文件管理

10.4 文件系统调用

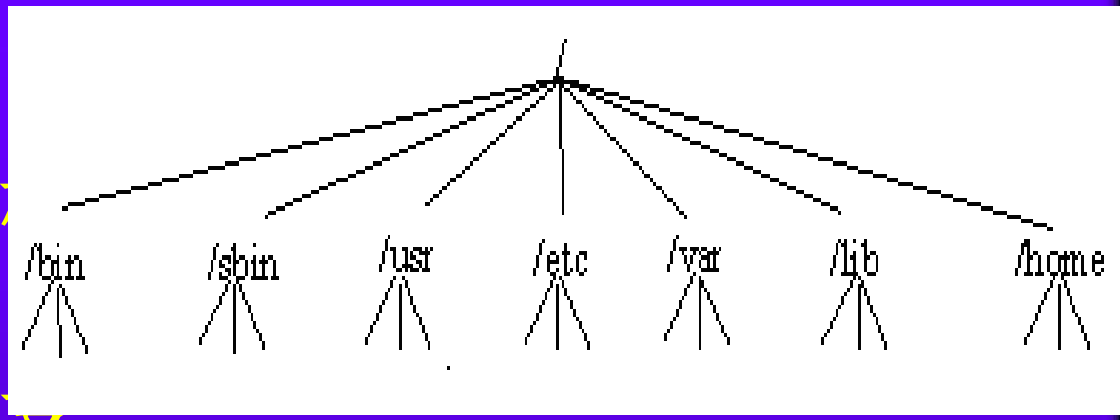
10.1 特点与文件类别

10.1.1 特点

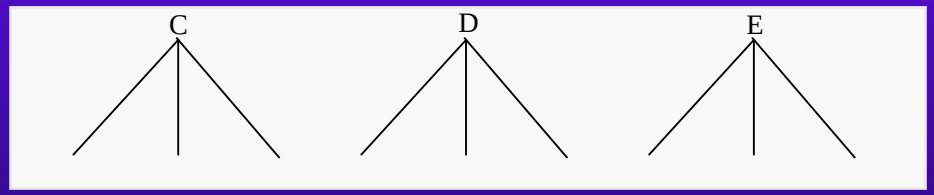
- 树型结构
- 无结构的字符流式文件
- 文件可以动态地增长或减少
- 拥有者可设置相应的访问权限保护文件
- 外部设备被看作文件

9.1.2 Linux文件系统的结构与DOS文件系统的区别

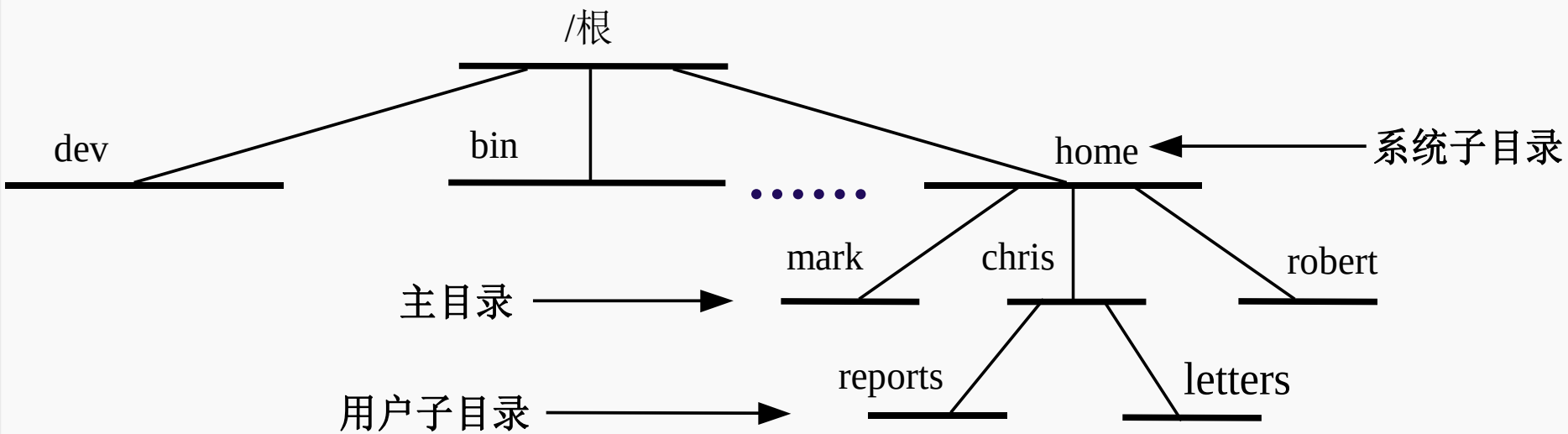
- Linux文件系统
  - 单个目录树的结
  - 根是根目录“/”
  - 往下连接各个分支



- DOS文件系统
  - 每个分区为树根
  - 多个分区，形成了多个树并列



Linux文件系统结构（1）



- 以文件的“目的”为依据，对文件进行分组，相同目的命令都放在同一子目录，完成系统的特定功能
- 系统子目录中文件的作用：保证系统的正常运行

Linux文件系统结构（2）

- /bin： 二进制可执行命令
- /dev： 设备特殊文件
- /etc： 系统管理和配置文件
- /etc/rc.d： 启动的配置文件和脚本
- /home： 用户主目录的基点，比如用户user的主目录就是/home/user
- /root： 系统管理员的主目录
- /proc： 虚拟的目录，是系统内存的映射。可直接访问，获取系统信息。

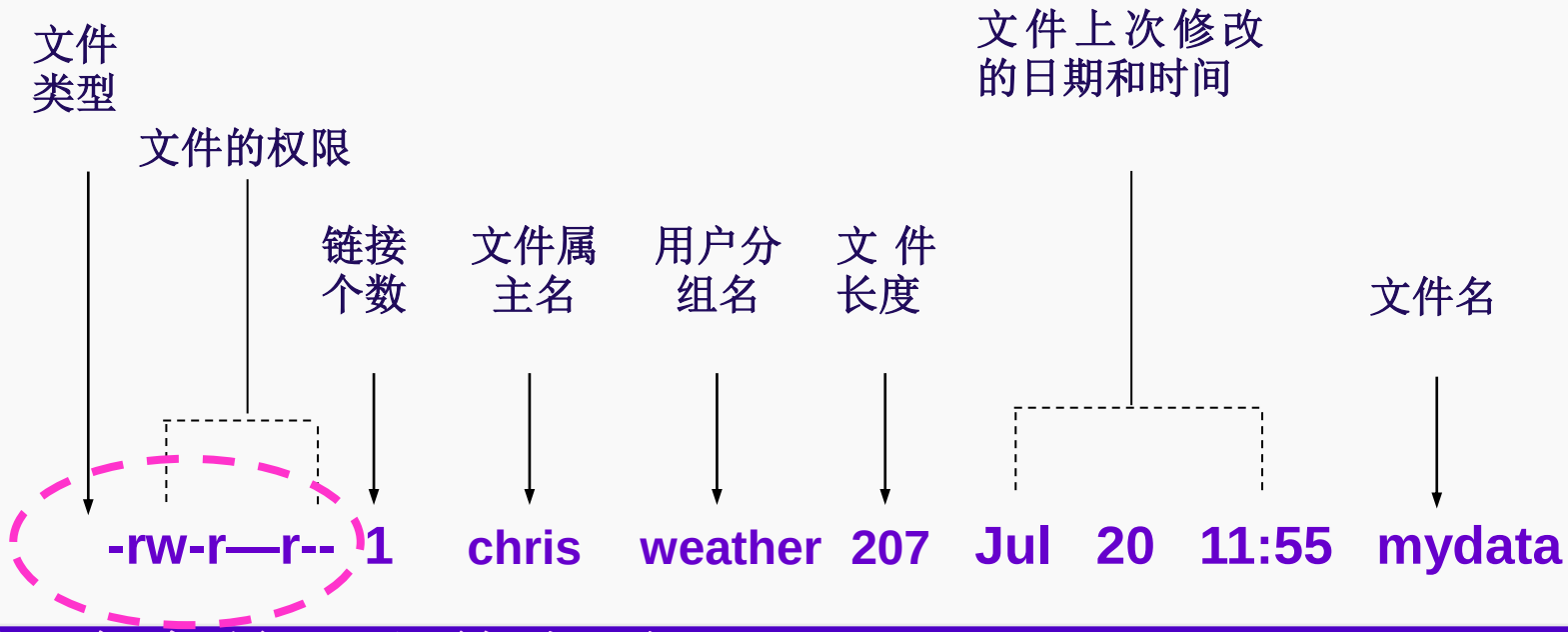
Linux文件系统结构（3）

- /usr 最庞大的目录，要用到的应用程序和文件几乎都在这个目录。
  - /usr/X11R6： 存放X window的目录
  - /usr/bin ： 众多的应用程序
  - /usr/sbin ： 超级用户的一些管理程序
  - /usr/doc ： linux文档
  - /usr/include ： linux下开发和编译应用程序所需要的头文件
  - /usr/lib ： 常用的动态链接库和软件包的配置文件
  - /usr/man： 帮助文档
  - /usr/src ： 源代码，linux内核的源代码就放在/usr/src/linux里

9.1.3 文件类型和权限

- Linux文件分为6种
  - 普通文件、目录文件、设备文件、连接、有名管道、UNIX套接字。

文件的详细信息



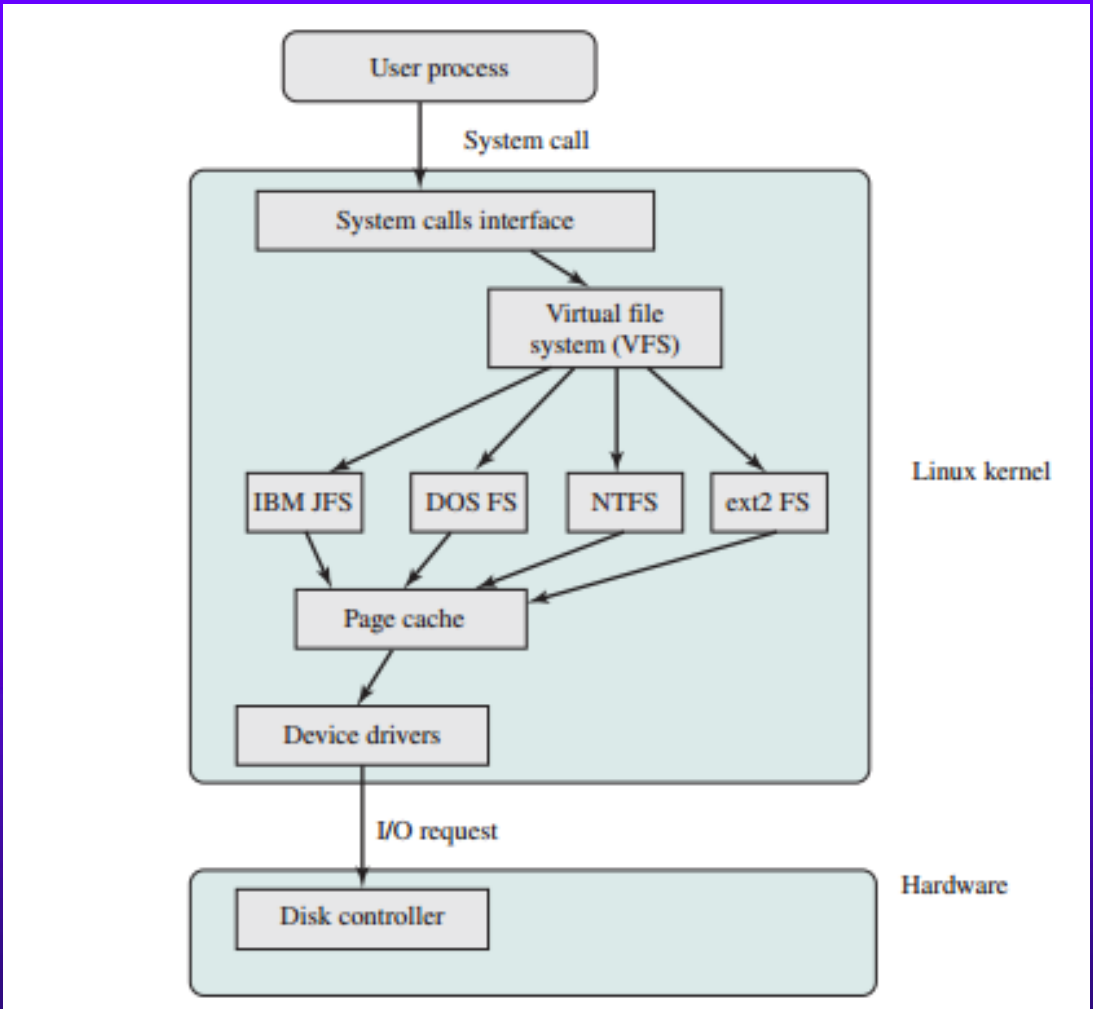
- 第1个字符：文件类型。
  - 横线：非目录文件。
  - d：目录。
- 第2-10个字符：文件权限。
  - 三个安全级别：所有者、组、其他用户
  - 三种权限：r：读　　w：写　　x：可执行。

- 所有者
  - 一般是文件的创建者。文件被创建时，自动拥有读、写和可执行权限。
- 设置权限的命令
  - **chmod**：重新设定不同的访问权限。
  - **chown**：更改所有者。仅超级用户使用 。
  - **chgrp**：更改用户组。文件的属主或超级用户使用 。

10.2 Linux的虚拟文件系统

- Linux内核中的一个软件层
  - 磁盘上物理文件系统各种结构的映像，随时建立和删除；
- 功能：支持多种不同的文件系统
  - 向Linux 内核和运行的进程提供了一个公共接口
  - 隐藏了各种硬件的细节，使得不同的物理文件系统在内核看来都是相同的。
  - Linux支持的物理文件系统
    - 基于磁盘的文件系统、基于网络的文件系统、特殊的文件系统。minix 、 ext 、 ext2、 ext3 、 msdos 、 proc 、 fat 、 vfat

10.2.1 虚拟文件系统VFS框架

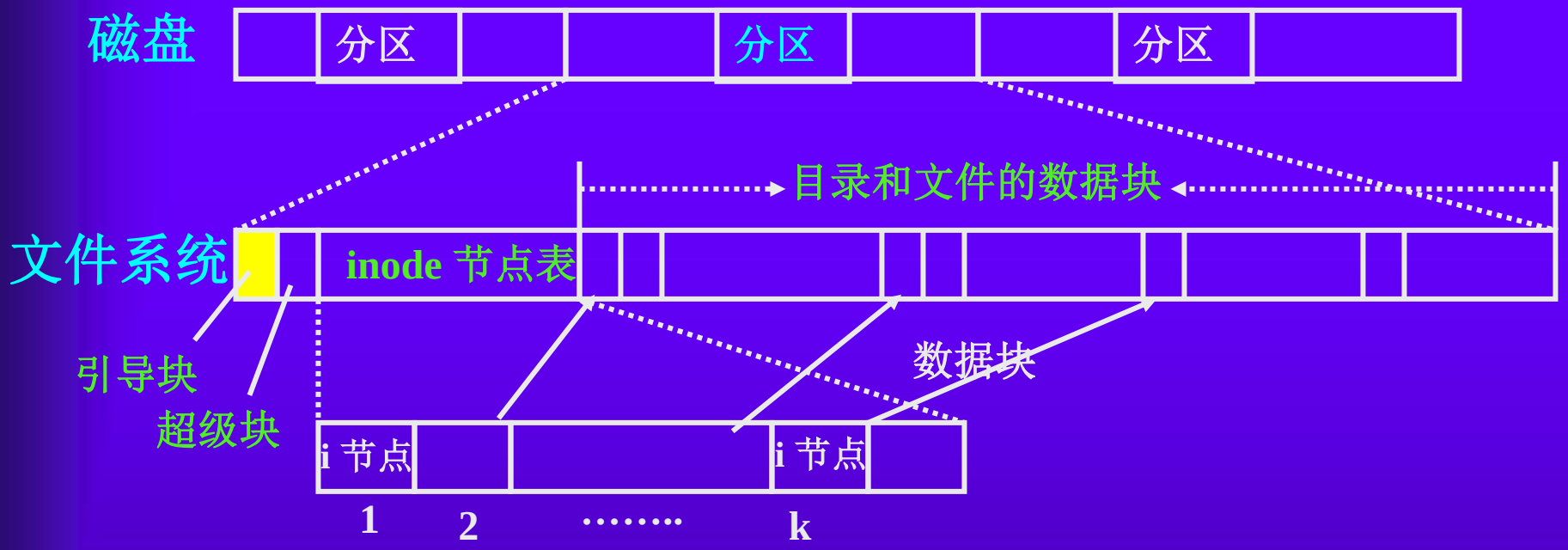


Linux Virtual File System Context

VFS的面向对象的思想

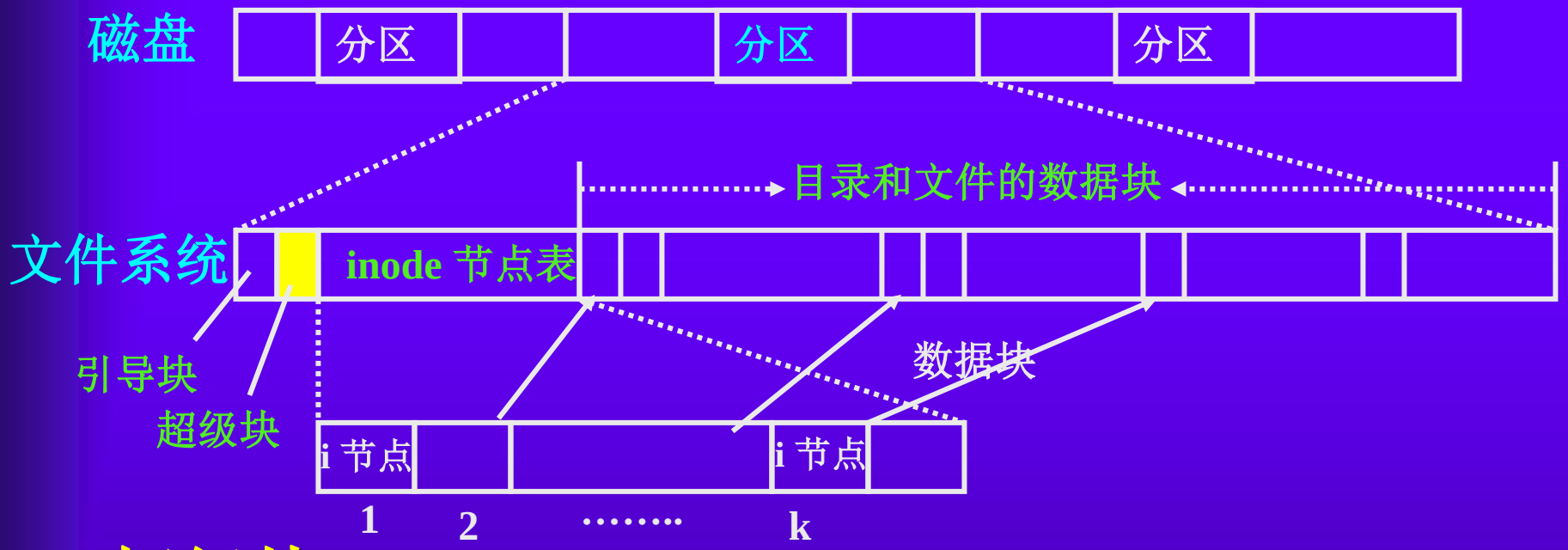
- VFS是由面向对象的思想发展起来的
  - VFS提供一个抽象基类
  - 由该基类派生出的子类支持具体的文件系统。
- Linux用C语言开发的，“对象”用数据结构实现。

磁盘分区和文件系统



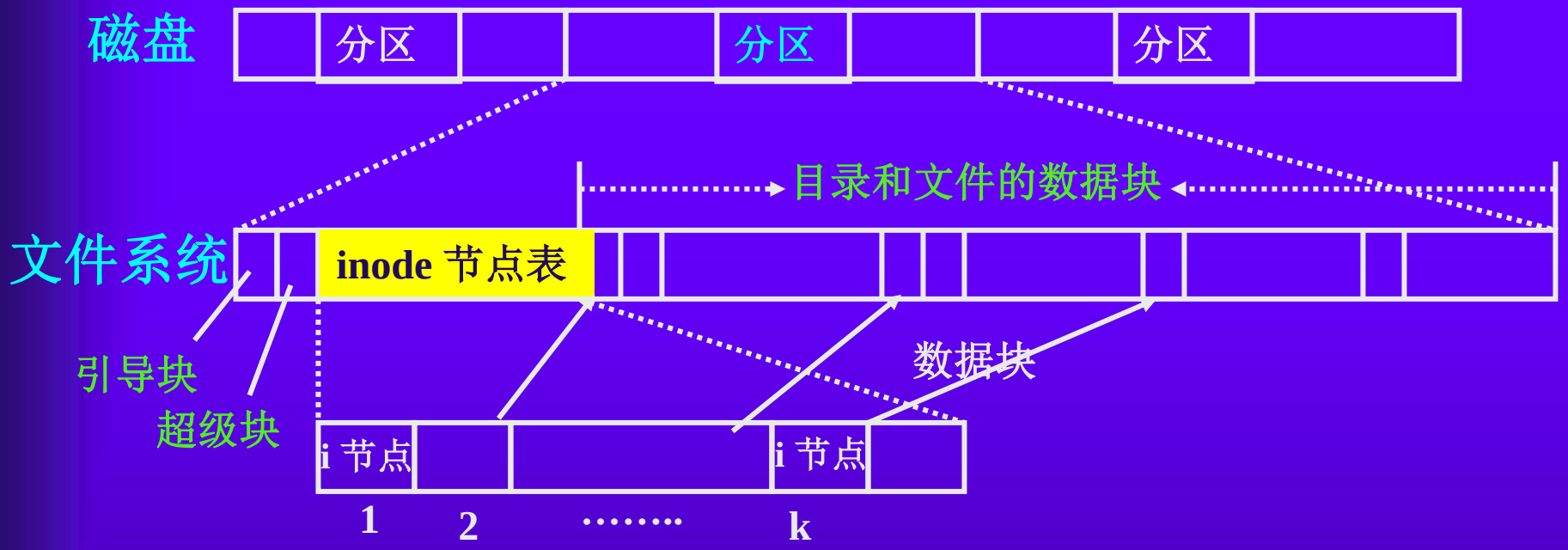
- 引导块：系统启动时所需的信息

磁盘分区和文件系统



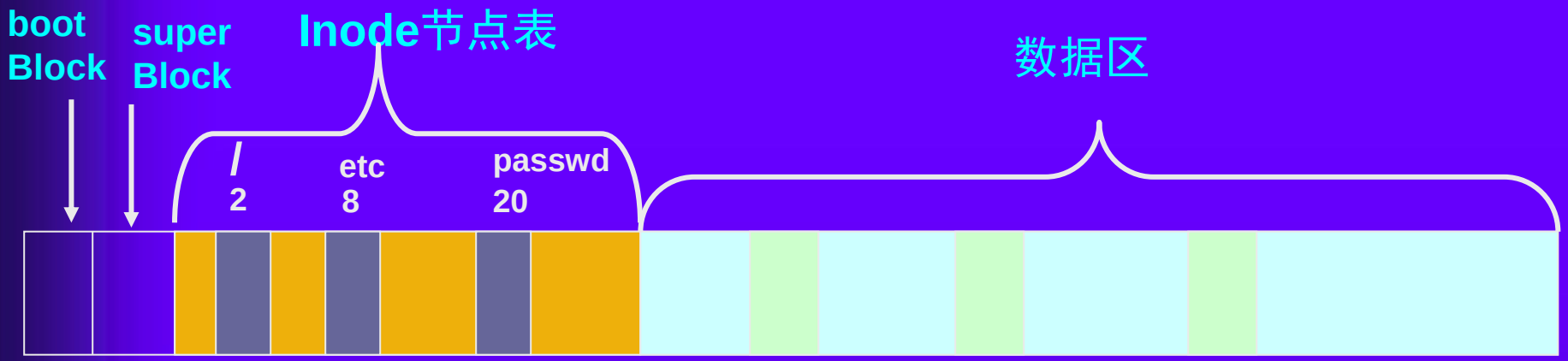
- **超级块(super-block)**
  - 文件系统的信息：状态、类型、大小、区块数、索引节点数等。
  - 系统启动后，将其复制到内存，并周期性的利用内存里的最新内容去更新硬盘上内容。

磁盘分区和文件系统



- 索引节点表
  - 所有索引节点的一个线性表，编号；0、1号一般不用，根目录“/”: 2号索引节点
  - 索引节点：存储一个文件的属性信息(除文件名)

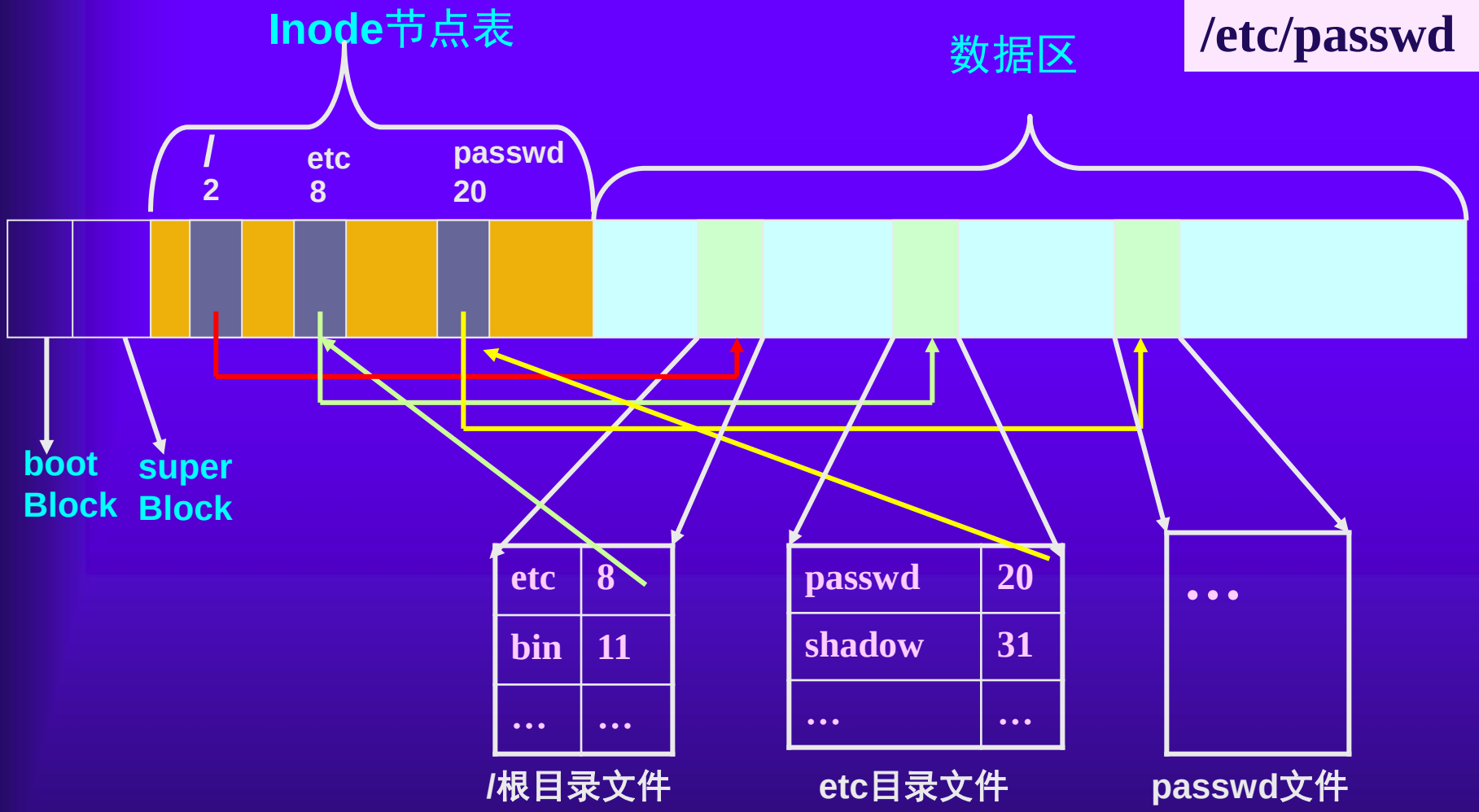
路径名的解析过程



- 目录文件
  - 文件名与其inode节点号的对应关系
- 磁盘上的文件分两部分
  - 数据块：实际存放文件数据的磁盘块
  - 索引节点inode

路径名的解析过程

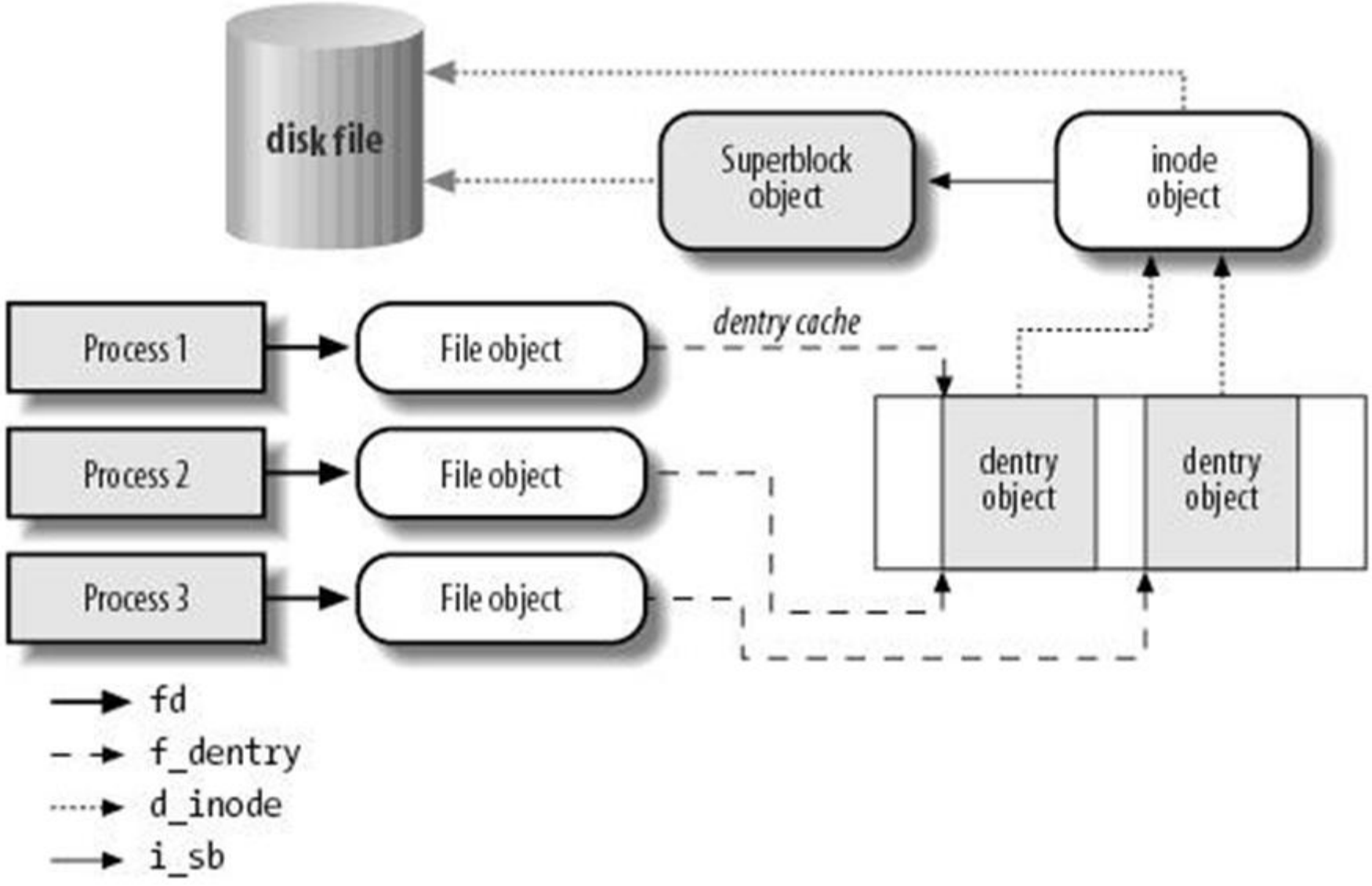
/etc/passwd



10.2.2 VFS的数据结构

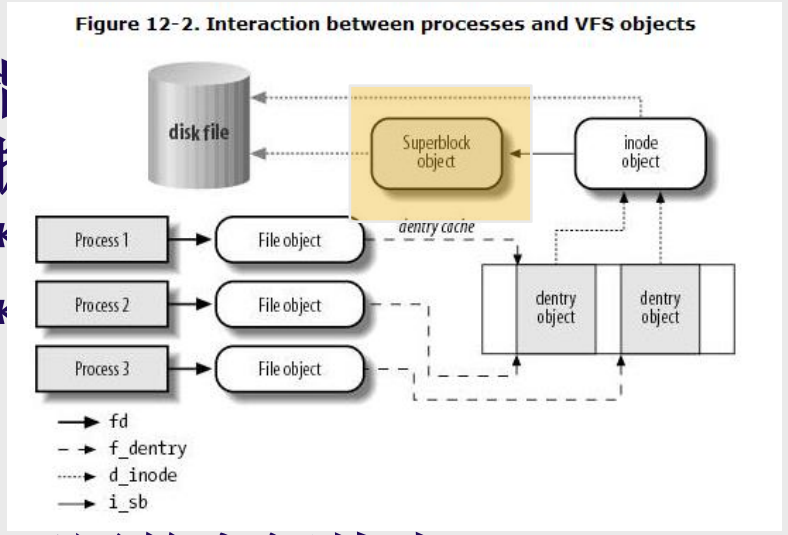
- 超级块super-block
  - 存放已挂接文件系统的有关信息
- 索引节点inode
  - 存放一个具体文件的一般信息
- 目录项dentry
  - 保存目录项与相应文件进行链接的信息
- 文件file (系统打开文件链表中的一个结点）
  - 存放打开文件与进程之间进行交互的有关信息

Interaction between processes and VFS objects



VFS超级块

```
struct super_block {
    struct list_head    s_list;          /*指
    unsigned long       s_blocksize;     /*数据
    struct list_head    s_inodes;        /*
    struct list_head    s_files;         /*
    ……
```



文件系统中总块数（文件系统大小）

柱面组的数目、 空闲块的个数、可用的空闲块表(free[])；

索引节点的数目、空闲的索引节点的个数 、空闲索引节点表

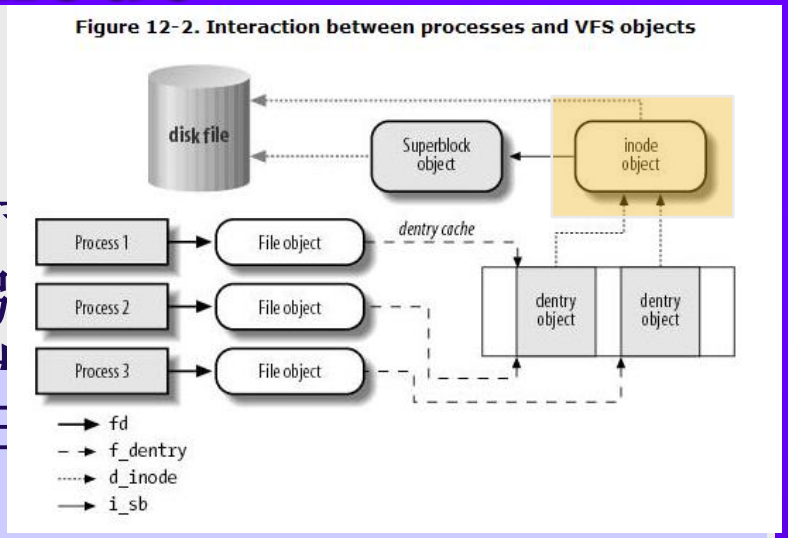
```
    ……
}
```

各种具体文件系统在安装时建立的。

在inculde/Linux/fs.h中定义

索引节点 inode

```
struct inode {
    ……
    unsigned long      i_ino;    /*索引
    atomic_t           i_count;      /*引
    ……
    文件长度 (i_size)
    ……
struct super_block    *i_sb;   /*所属的超级块*/
}
```

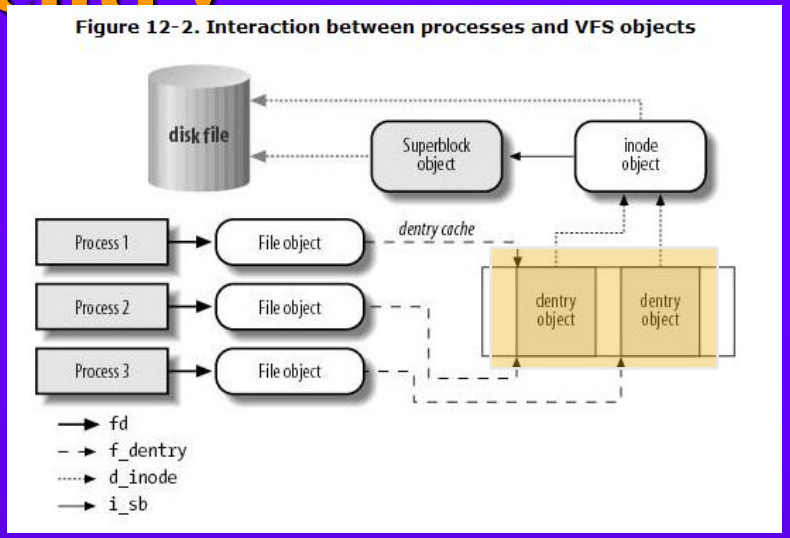


- 包含文件的长度、创建及修改时
磁盘中的位置等信息。
- 一个文件系统维护了一个索引节点数组，每个文件或目录都与数组中的唯一元素对应。
- 索引节点号：节点在数组中的索引号。
- 在/includ/fs/fs.h中定义

目录项dentry

```
struct dentry {
        unsigned int d_flags;          /*目录项标志*/
        struct inode *  d_inode;       /*与文件名关联的索引节点*/
        struct dentry *  d_parent;     /*父目录的目录项*/
        struct list_head d_hash;       /*目录项形成的哈希表*/
        struct list_head d_child;      /*父目录的子目录项所形成的链表*/
        struct list_head d_subdirs;    /*目录项的子目录所形成的链表*/
        struct list_head d_alias;      /*索引节点别名的链表*/
        struct qstr d_name;            /*目录项名（可快速查找） */
        struct super_block * d_sb; /*目录项树的根（即文件的超级块）*/
        struct  qstr   d_name          /*文件名*/
 ……
};
```

目录项 dentry



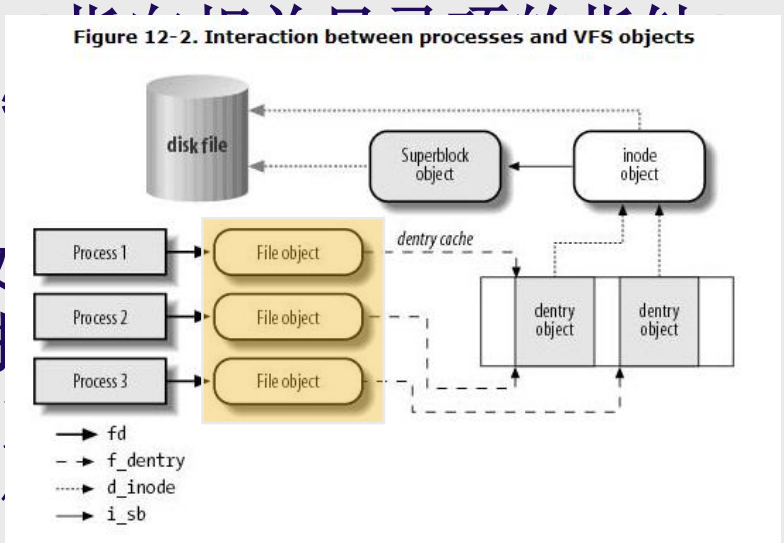
- 目的：方便查找文件。
- 一个路径的各个组成部分。
  - 不管是目录还是普通的文件，都是一个目录项对象。如，在路径/home/source/test.c中，目录 /, home, source和文件 test.c都对应一个目录项对象。
- 没有对应的磁盘数据结构，VFS在遍历路径名的过程中将它们逐个地解析成目录项对象。

系统打开文件链表

- 一个双向链表：Linux系统内核把所有进程打开的文件集中管理。
- 全局变量 first_file 指向其表头。
- 每个节点是一个file结构。
  - 存放一个已打开文件的管理控制信息
  - 进程调用open( )，建立一个file结构体，已打开的文件在内存中的表示。
  - 进程和磁盘上的文件的对应关系，用户只需与文件对象打交道。

系统打开文件链表中的结点 file

```
struct file {
    struct list_head      f_list;     /*所有打开的文件形成一个链表*/
    struct dentry         *f_dentry;
    struct vfsmount       *f_vfsmnt;/
    struct file_operations  *f_op;
    atomic_t              f_count;      /*文
    unsigned int          f_flags;      /*
    mode_t                f_mode;       /*
    loff_t                f_pos;        /*文
    struct inode *        f_inode;    /* 指向文件对应的inode */
/*预读标志、要预读的最多页面数、上次预读后的文
件指针、预读的字节数以及预读的页面数*/
……
};
```



f_mode：文件创建或打开时指定的文件属性，包括文件操作模式和访问权限。

10.3 进程的文件管理

进程打开的所有文件，由进程的私有结构管理

- **fs_struct**：记录着文件系统根目录和当前目录
- **files_struct**：包含着进程的打开文件表。

```
struct fs_struct {
 int count; /* 共享此结构的计数值 */
unsigned short umask; /* 文件掩码 */
struct inode * root, * pwd; /* 根目录和当前目录inode指针 */
};
```

root：系统根目录inode，按照绝对路径访问文件时开始。

pwd：当前目录inode，相对路径开始。

进程打开文件表 files_struct

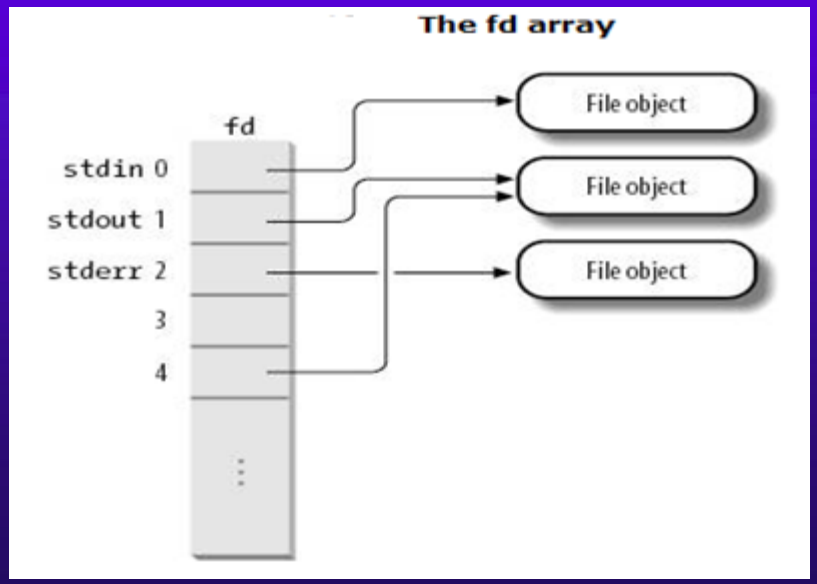
```
#define NR_OPEN 256
struct files_struct {
int count; /* 共享该结构的计数
fd_set   close_on_exec;
struct file *  fd[NR_OPEN];
};
```

fd:

元素:一个指向file结构体的指针

下标:文件标识号

- 进程打开文件时，建立一个file结构体，加入到系统打开文件链表，把首地址写入fd[ ]第一个空闲元素中



```
#define NR_OPEN 256
struct files_struct {
int count; /* 共享该结构的计数
fd_set   close_on_exec;
struct file *  fd[NR_OPEN];
};
```

fd:
元素:一个指向file结构体的指针
下标:文件标识号

- **fork( ): 子进程共享父进程的打开文件表。父子进程两者的fd下标相同的两个元素指向同一个file结构(f_count增1)。**

- **一个文件可以被某个进程多次打开，每次都分配一个file，并占用fd[ ]的一项，得到一个文件标识号。**

f_inode指向同一个inode