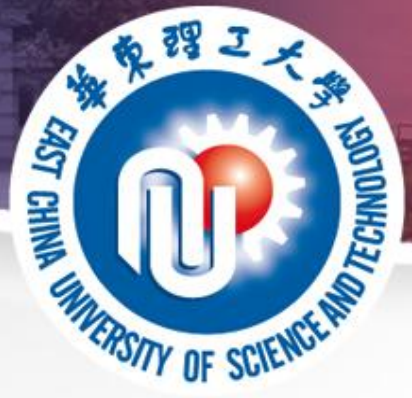


第八章　文件系统

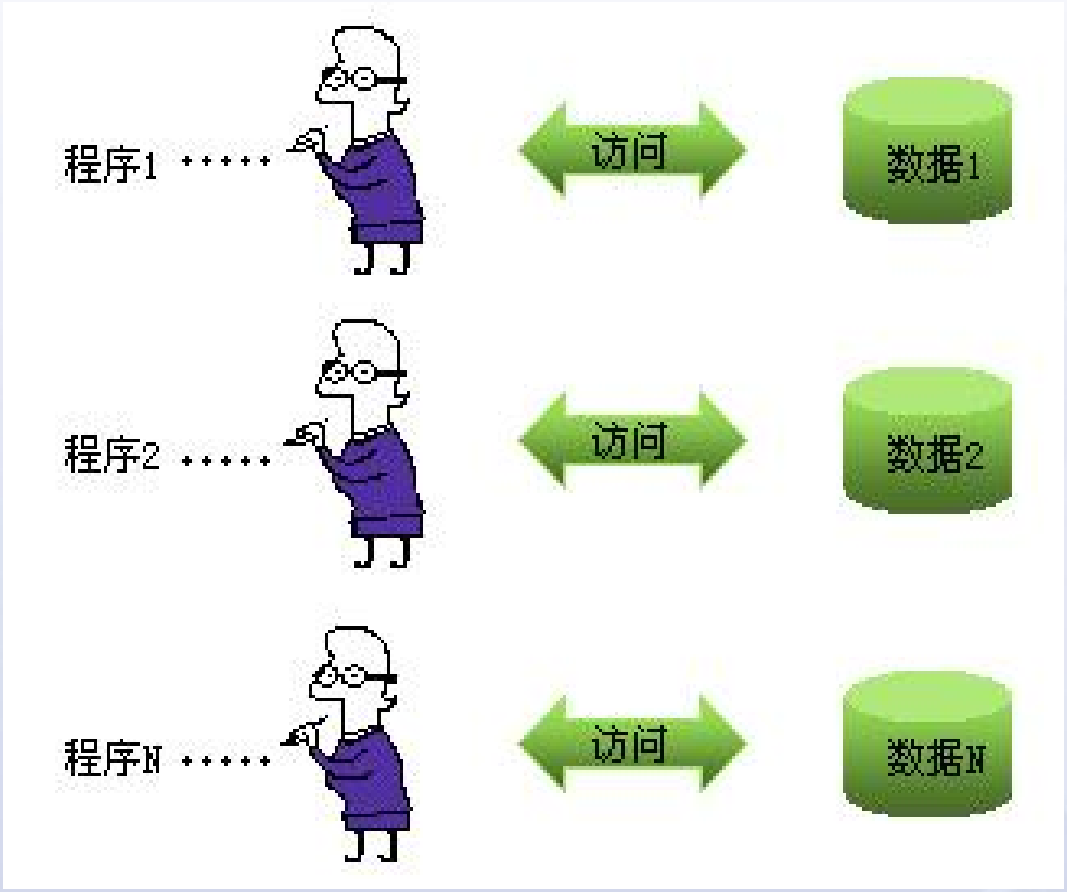
华东理工大学计算机系

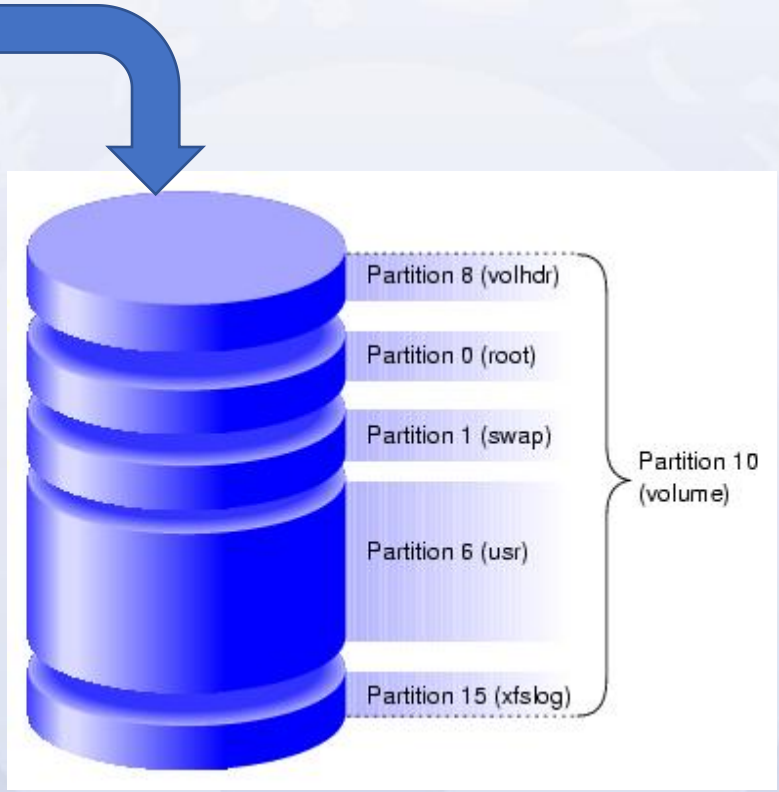
罗　飞

Content

什么是文件系统?

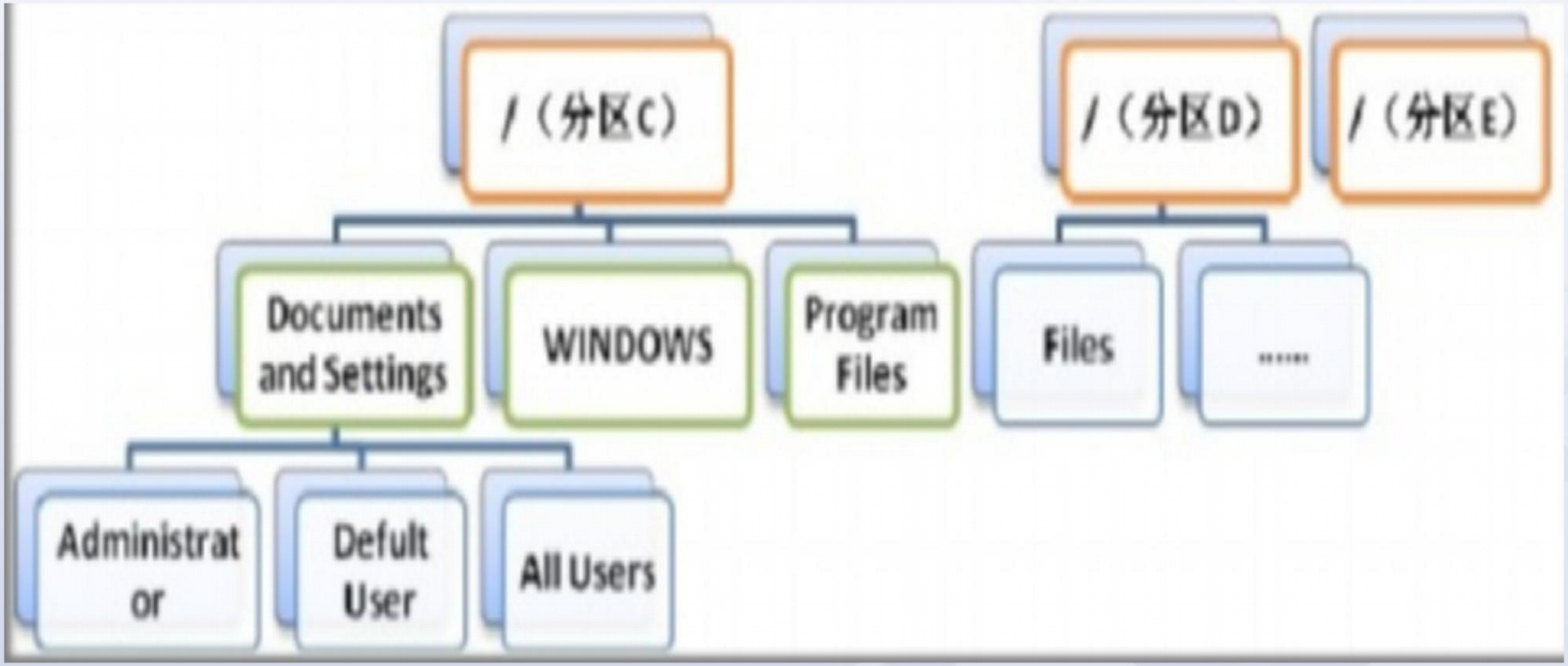
用户看到的数据在什么地方?





文件系统介绍

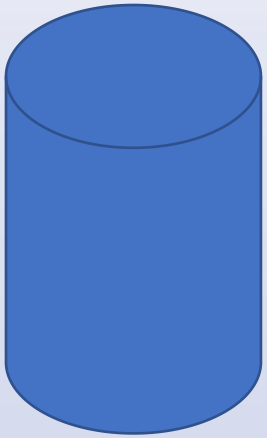
- 文件的数据结构或组织方法
- 用户通过文件直接地和操作系统交互



数据组织

系统数据

用户数据

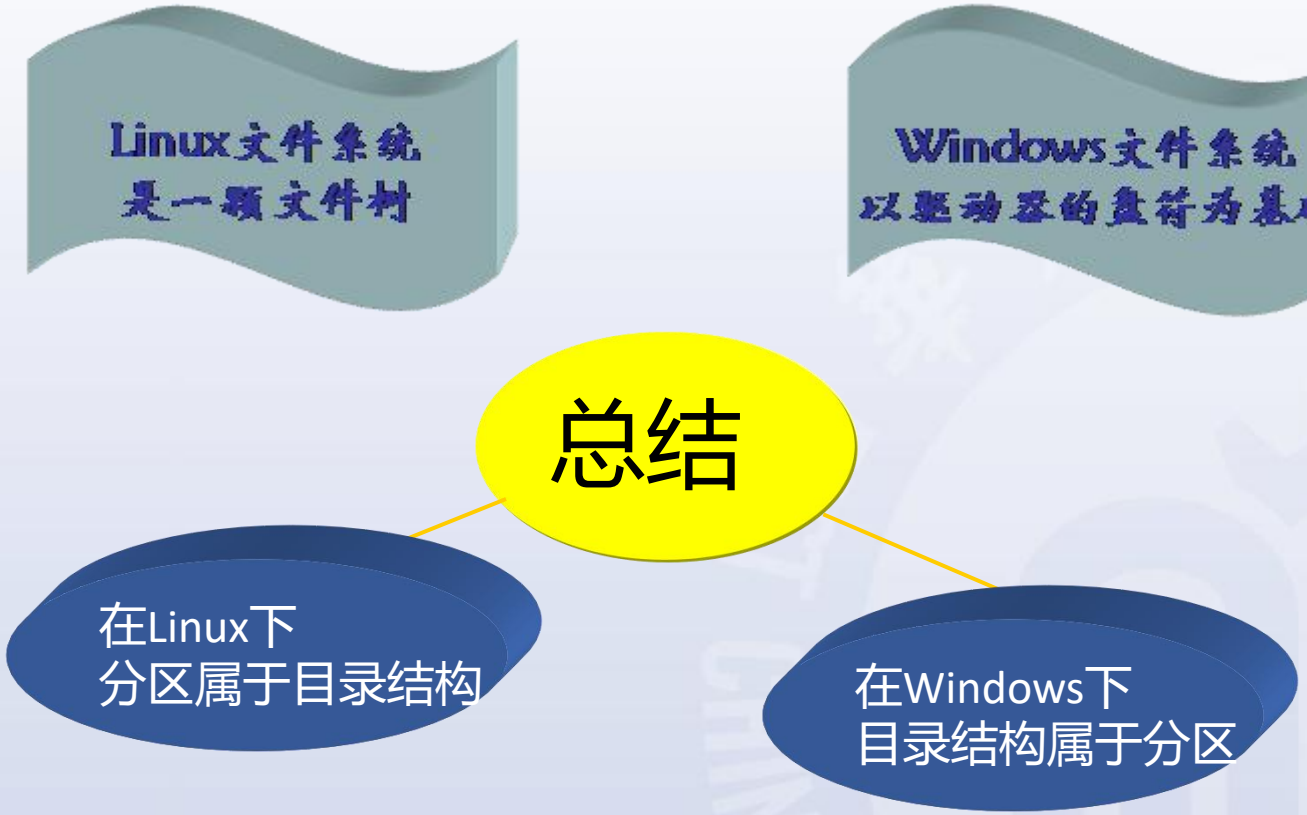


操作系统的数据

也是通过文件系统直观地存储在介质上

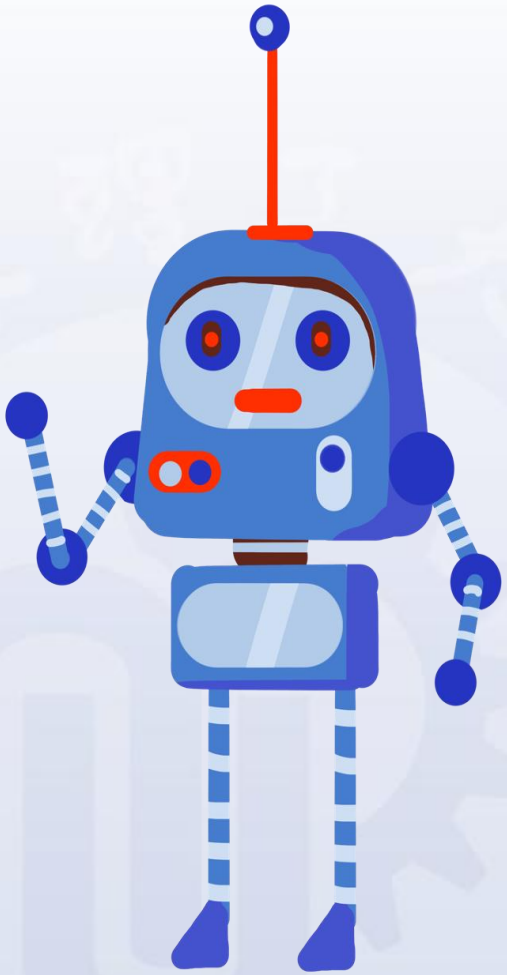
操作系统按照自己的数据格式管理

Linux与Windows文件系统的区别



嵌入式文件系统介绍

- 嵌入式文件系统
  - 在嵌入式系统中应用的文件系统
- 嵌入式文件系统影响因素
  - 嵌入式系统硬件设备
  - 嵌入式系统应用



嵌入式文件系统介绍

- **嵌入式操作系统的文件系统的设计目标**
  - （1）使用简单方便
  - （2）安全可靠
  - （3）实时响应
  - （4）接口标注的开放性和可移植性
  - （5）可伸缩性和可配置性

嵌入式文件系统介绍

- **嵌入式操作系统的文件系统的设计目标**
  - （6）开放的体系结构
  - （7）资源有效性
  - （8）功能完整性
  - （9）热插拔
  - （10）支持多种文件类型

嵌入式文件系统介绍

- **嵌入式文件系统特征**
  - 伴随嵌入式操作系统产品
  - 典型的操作系统及其文件系统
    - QNX
    - VxWorks

嵌入式文件系统介绍

- Linux文件系统
  - ext（extended file
  - Msdos, VFAT, iso96
  - smb、ncp

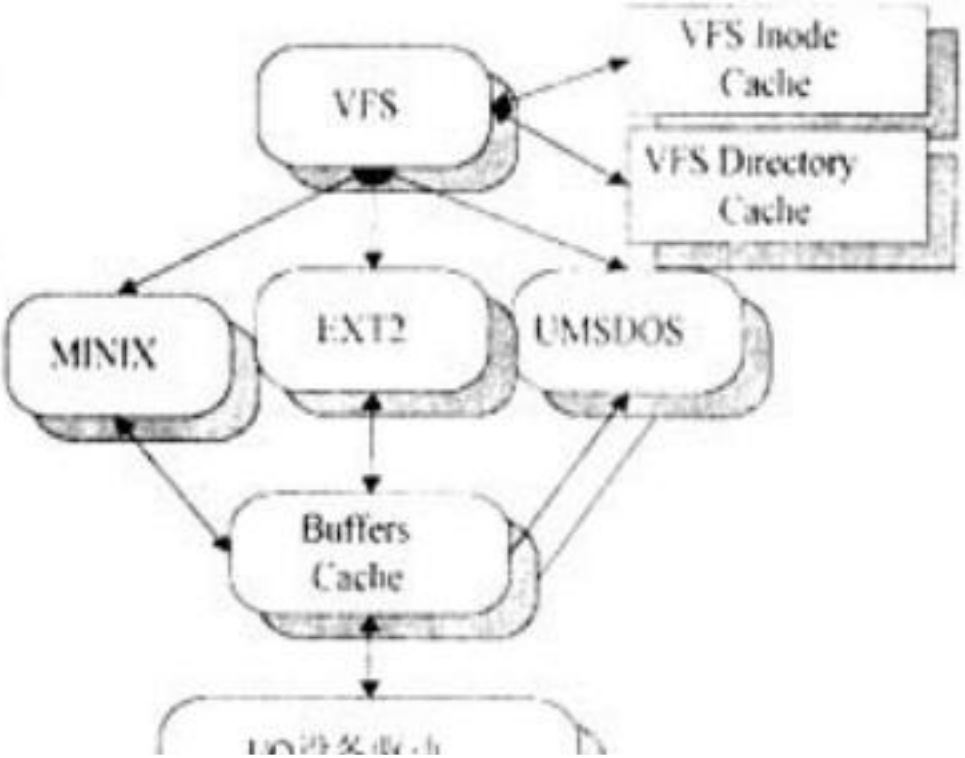
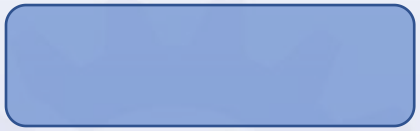


图2 VFS和文件系统层次图

嵌入式文件系统介绍

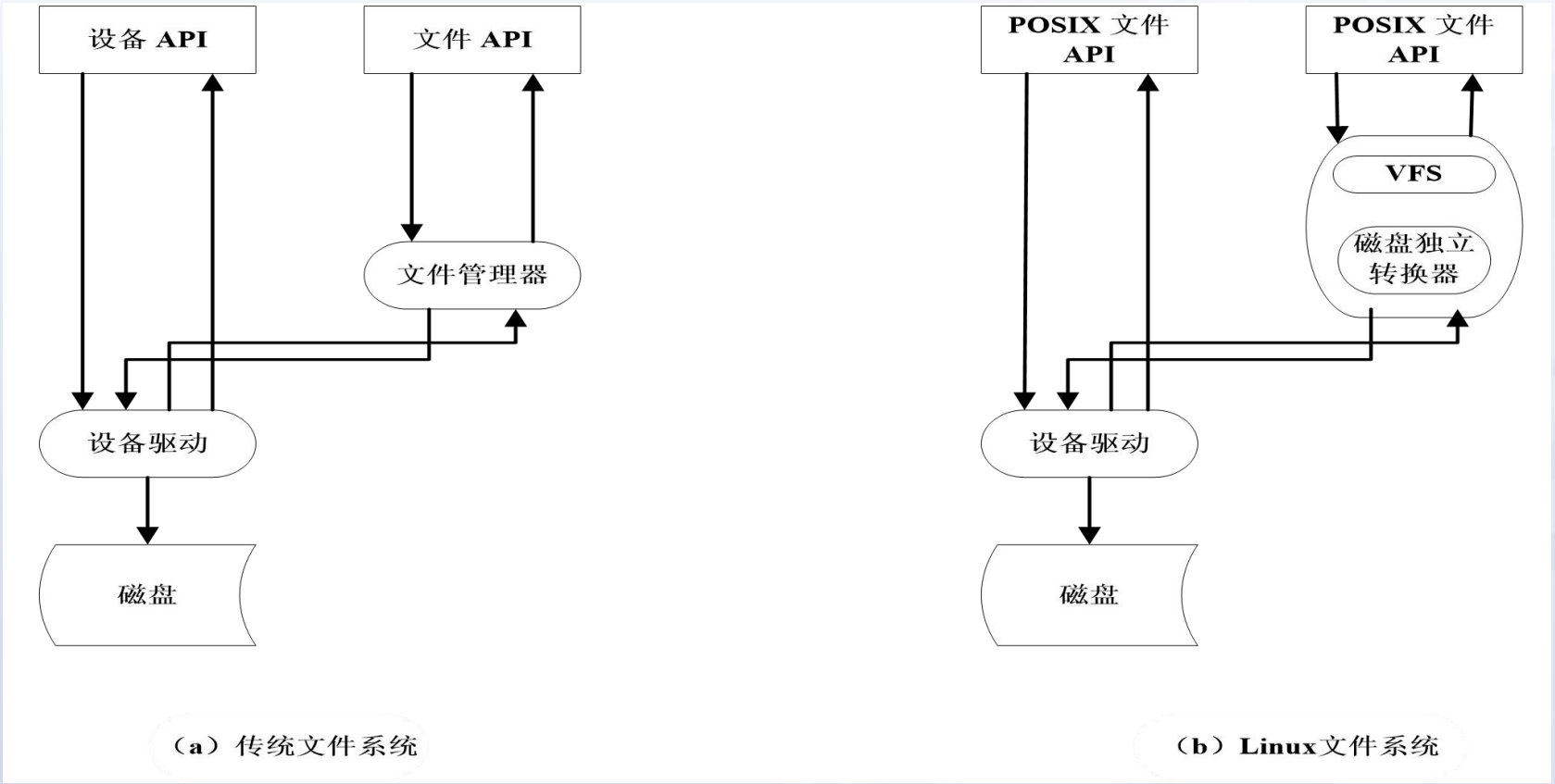
- Linux初期的基本文件系统是Minix
- 1992年开发了Linux专用的文件系统ext（Extended File System）
- 1993年增加了ext2

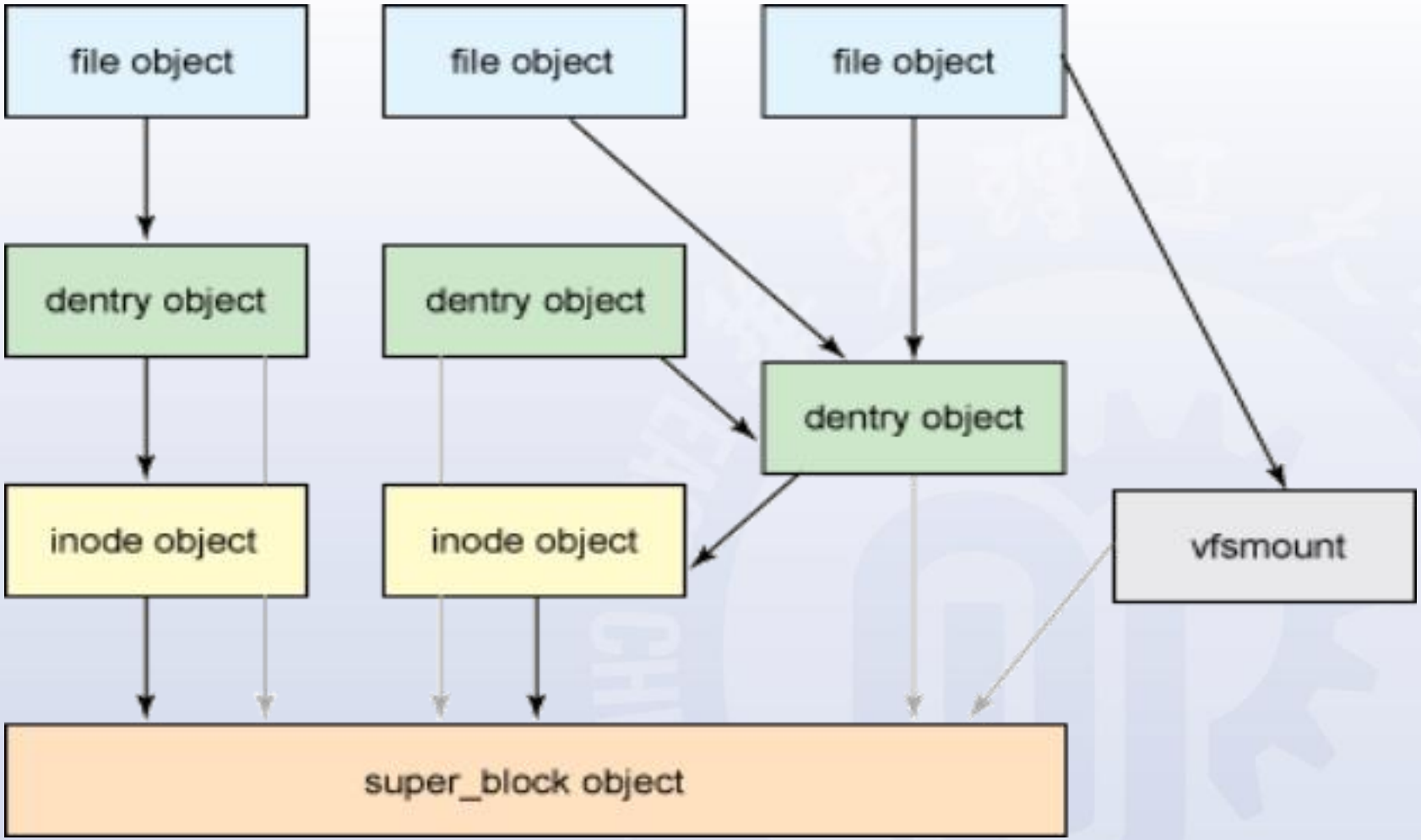


- 存储文件系统的设备为block设备（block device）

2、嵌入式Linux文件系统框架

- 现代操作系统都提供多种访问存储设备的方法




file object
file object
file object
dentry object
dentry object
dentry object
inode object
inode object
vfsmount
super_block object

嵌入式Linux文件系统框架

- **设备驱动提供用户空间设备API去直接控制硬件设备**
  - 用户的进程直接读写磁盘
  - 数据的完整性难以保证
  - 内容很有可能会被用户空间的程序覆盖
  - 使得系统的稳定性也大大地降低

嵌入式Linux文件系统框架

- 大部分操作系统都是由文件管理器来使用设备API
  - 对上层用户空间的应用程序提供文件API

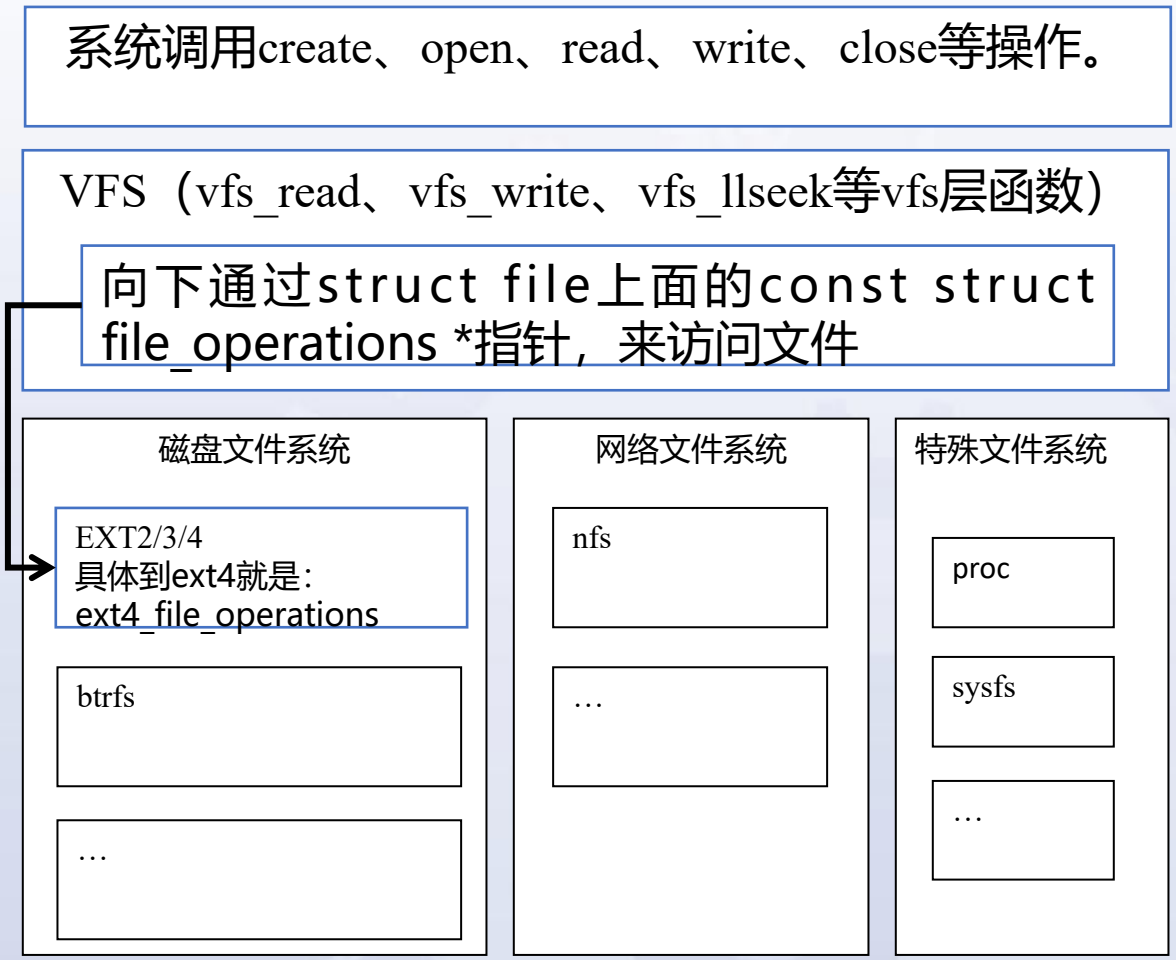
- 在特殊的环境下才允许用户通过设备API访问硬件设备

文件系统在Linux中位置

文件系统模块层次图

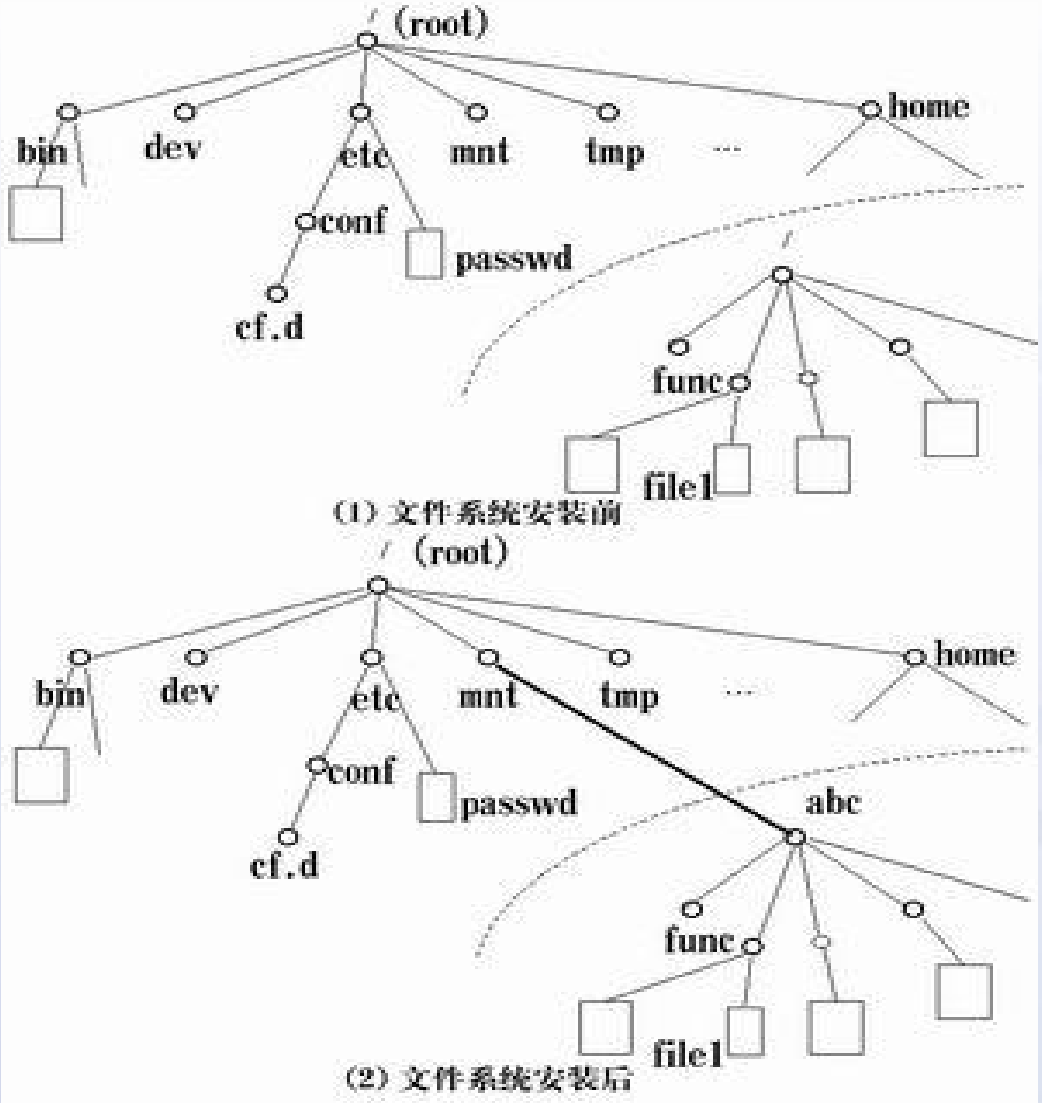
文件系统模块层次图

每个文件系统实现（比如ext2/ext3/ext4、btrfs 等等）导出一组通用接口，供 VFS 使用，和上层VFS对接。



嵌入式Linux文件系统框架

- 按树的形式组织
  - 通过连接将“树”上的“叶子”连接到其他的“叶子”或者“分支处”



嵌入式Linux文件系统框架

- 两条独立控制设备驱动的途径
  - 通过设备驱动的接口
  - 通过文件管理器接口

图 2　DiskIO 结构

- 设备驱动的接口API
  - 从文件管理器API中继承下来的
  - open()、close()、read()、write()、lseek()和ioctl()等

嵌入式Linux文件系统框架

- **嵌入式Linux中三种常用的块驱动程序**
  - （1）Blkmem驱动层
  - （2）RamDisk驱动层
  - （3）MTD驱动层

3、JFFS2嵌入式文件系统

- **克服了JFFS中的缺点**
  - （1）使用了基于哈希表的日志节点结构

    大大加快了对节点的操作速度
  - （2）支持数据压缩
  - （3）提供了“写平衡”支持
  - （4）支持多种节点类型(数据目录I节点等)
  - （5）提高了对闪存的利用率

JFFS2嵌入式文件系统

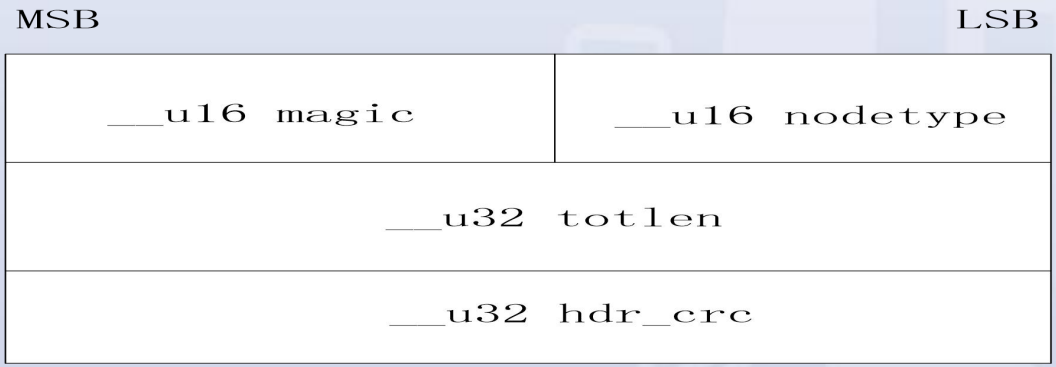
- 日志结构（log-structured）

- 多种节点
  - 数据和元数据（meta-data)的节点
  - 在闪存上顺序的存储

JFFS2嵌入式文件系统

- JFFS2数据结构及内存表示

```
struct jffs2_unknown_node
{
    __u16 magic;/*作为nodetype的补充*/
    __u16 nodetype;/*节点类型*/
    __u32 totlen;/*节点总长度*/
    __u32 hdr_crc;/*CRC校验码*/
}
```



JFFS2嵌入式文件系统

几种?

- **magic的最左边两位用来表示节点类型**
  - (1) JFFS2_FEATURE_INCOMPAT
  - (2) JFFS2_FEATURE_ROCOMPAT
  - (3) JFFS2_FEATURE-RWCOMPAT_DELETE
  - (4) JFFS2_FEATURE_RWCOMPAT_COPY:

JFFS2嵌入式文件系统

- **JFFS2 定义了三种节点类型**
  - (1) JFFS2_NODETYPE_INODE
  - (2) JFFS2_NODETYPE_DIRENT
  - (3) JFFS2_NODETYPE_CLEANMARKER

JFFS2嵌入式文件系统

- 目录节点的定义

```
    struct jffs2_raw_dirent
{
    __u16 magic;
    __u16 nodetype;/*节点类型，设置为
JFFS_NODETYPE_DIRENT*/
    __u32 totlen;
    __u32 hdr_crc;/*jffs2_unknown_node部分的CRC校验*/
    __u32 pino;/*上层目录节点的标号*/
    __u32 version;
    __u32 ino;/*节点编号，如果是0表示没有链接的节点*/
    __u32 mctime;/*创建时间*/
    __u8 nsize;/*大小*/
    __u8 unused[2];
    __u32 node_crc;/*校验码*/
    __u32 name_crc;
    __u8 name[0];/*名称*/
}
```

JFFS2嵌入式文件系统

- 数据节点

```
struct jffs2_raw_inode
{
    __u16 magic;
    __u16 nodetype;/*设置为JFFS_NODETYPE_inode*/
    __u32 totlen;/*节点的总长度（包括有效数据）*/
    __u32 hdr_crc;/*jffs2_unknown_node部分的CRC校验*/
    __u32 pino;/*上层目录节点的标号*/
    __u32 version;
    __u32 ino;
    __u32 mode;/*文件的类型*/
    __u16 uid;
    __u16 gid;
    __u32 isize;/*实际长度*/
    __u32 atime;
    __u32 mtime;
    __u32 ctime;
    __u32 offset;/*对应数据在文件中的起始位置*/
    __u32 csize;/*压缩数据的长度*/
    __u32 dsize;/*数据有效长度*/
    __u8 compr;/*当前压缩算法*/
    __u8 usercompr;/*用户指定的压缩算法*/
    __u16 flags;/*标志位*/
    __u32 data_crc;/*数据校验码*/
    __u32 node_crc;/*头节点的校验码*/
}
```

JFFS2嵌入式文件系统

- **可靠性支持**
  - 进行擦写操作的时候突然掉电
    - 可能会出现有部分数据没有被擦写干净的情况
  - **标记**
    - JFFS2对块操作的时候，如果操作成功，会在块的开始做上标记
    - 通过该标记表明块内的数据处于一致状态

JFFS2嵌入式文件系统

- 内存使用
  - JFFS2中I节点的信息并没全部存放在内存中
  - 存放内容
    - mount操作时，会为节点建立映射表
    - 映射表并不全部存放在内存里面
    - 存放的是一个缩小尺寸的jffs2_raw_inode结构体—jffs2_raw_node_ref

JFFS2嵌入式文件系统

- jffs2_raw_node_ref

```
struct jffs2_raw_node_ref
{
    struct jffs2_raw_node_ref* next_in_ino;/*链表指针*/
    struct jffs2_raw_node_ref next_phys;/*在物理上相邻的块*/
    __u32 flash_offset;/*在Flash块中的偏移，一般为0*/
    __u32 totlen;
}
```

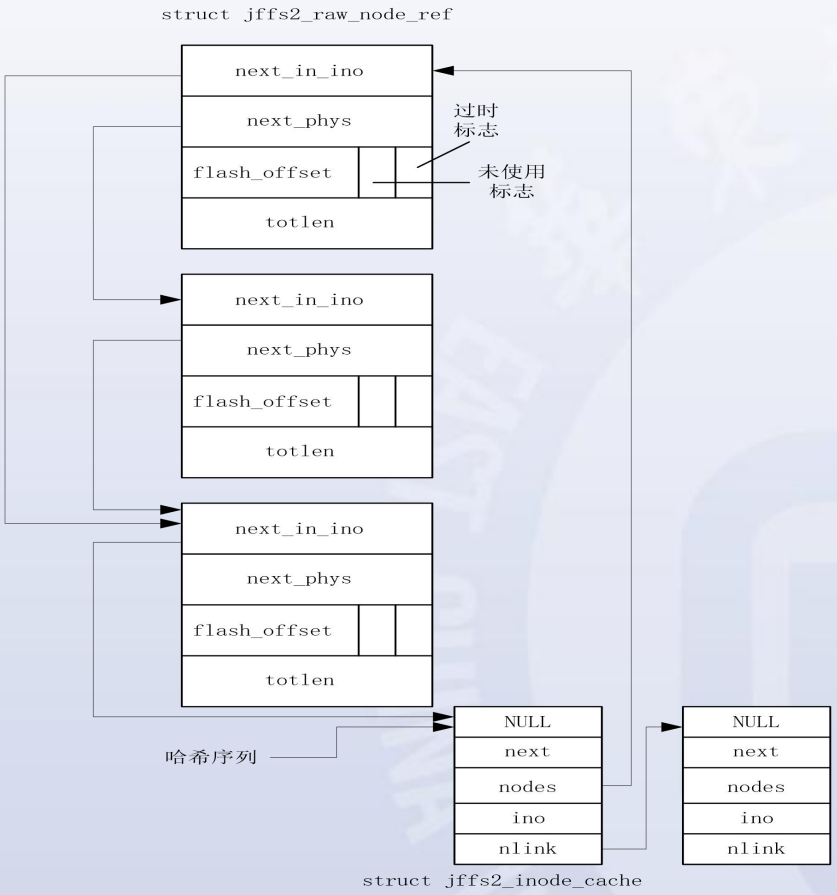
JFFS2嵌入式文件系统

- Jffs2_raw_node_ref信息在内存中通过jffs2_inode_cache结构进行管理

```
struct jffs2_inode_cache
{
    struct jffs2_scan_info *scan;/*在扫描链表的时候存放临时信息，
在扫描结束以后设置成NULL*/
    struct jffs2_inode_cache *next;
    struct jffs2_raw_node_ref * nodes;
    __u32 ino;
    int nlink; /*和当前I节点链接的节点数目*/
}
```

JFFS2嵌入式文件系统

- 内存中jffs2_inode_cache和jffs2_raw_node_ref的关系



JFFS2嵌入式文件系统

- JFFS2使用了多个级别的待回收块队列

- 垃圾收集步骤
  - （1）先看bad_used_list链表中是否有节点，如果有，先回收这个链表的节点
  - （2）做完了bad_used_list链表的回收，然后回收dirty_list链表

JFFS2嵌入式文件系统

- 写平衡
  - 平衡策略能提供较好的写平衡效果
  - 在垃圾收集中实现
    - 垃圾收集时会读取系统时间
    - 使用这个系统时间产生一个伪随机数
    - 利用这个伪随机数结合不同的待回收链表选择要进行回收的链表

JFFS2嵌入式文件系统

- **JFFS2的不足之处**
  - （1）挂载时间过长
  - （2）磨损平衡的随意性
  - （3）很差的扩展性

JFFS2嵌入式文件系统

- **JFFS3**
  - 设计目标
    - 支持大容量闪存（>1TB）的文件系统
  - JFFS3与JFFS2在设计上根本的区别
    - JFFS3将索引信息存放在闪存上
    - JFFS2将索引信息保存在内存中

4、根文件系统

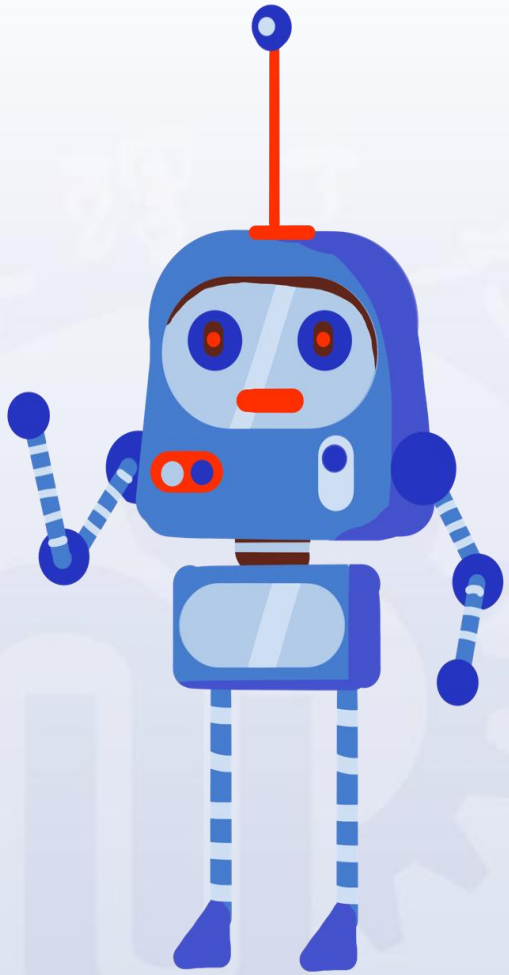
- 一个系统可同时存在不同的文件系统
  - 启动时首先挂载根文件系统
  - 若系统不能从指定设备上挂载根文件系统，则系统会出错而退出启动
  - 之后可以自动或手动挂载其他的文件系统

根文件系统

- 系统挂载的第一个文件系统
- 本质来说，根文件系统就是一种目录结构
- 和普通文件系统的区别
  - 要包括Linux启动时所必需的目录和关键性文件

什么是根文件系统

- 根文件系统（root filesystem）是存放运行、维护系统所必须的各种工具软件、库文件、脚本、配置文件和其他特殊文件的地方，也可以安装各种软件包。



根文件系统

- 根文件系统顶层目录

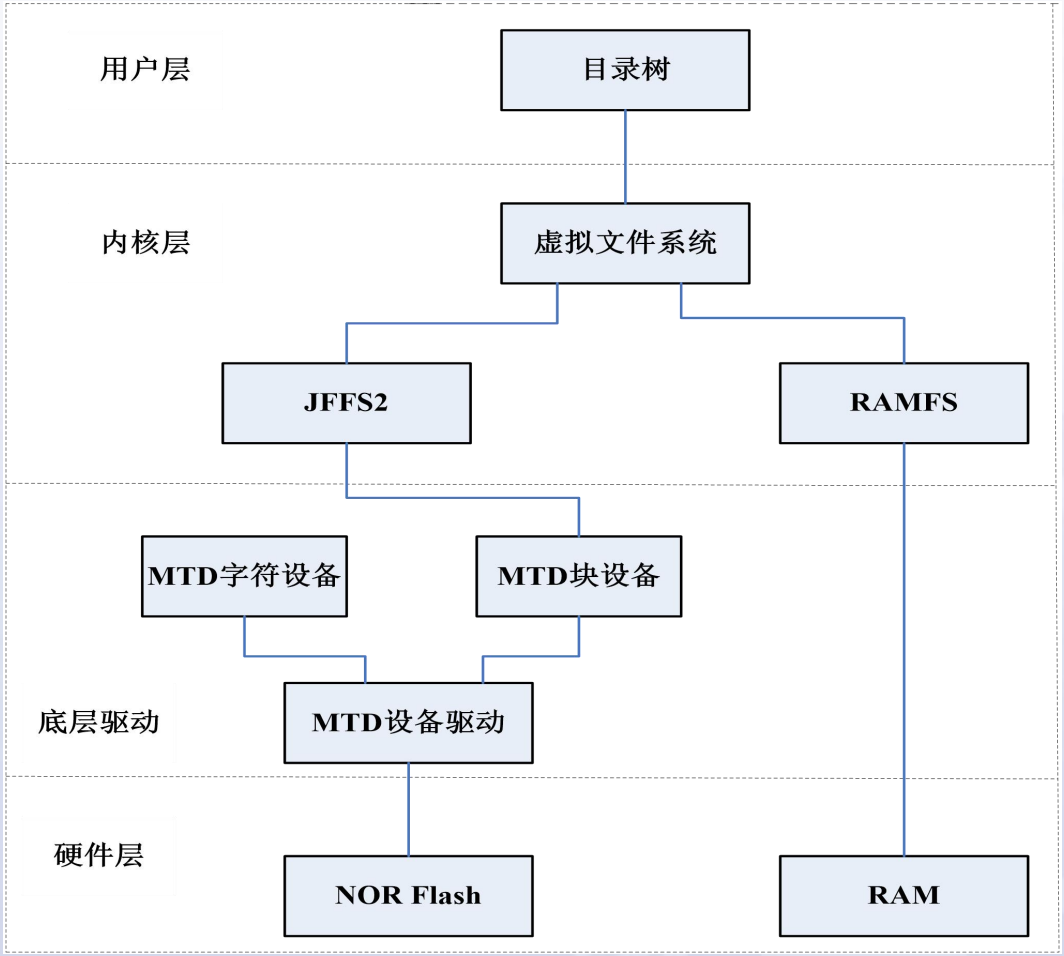
目录	内容
bin	存放所有用户都可以使用的、基本的命令。
sbin	存放的是基本的系统命令，它们用于启动系统、修复系统等。
usr	里面存放的是共享、只读的程序和数据。
proc	这是个空目录，常作为proc文件系统的挂载点。
dev	该目录存放设备文件和其它特殊文件。
etc	存放系统配置文件，包括启动文件
lib	存放共享库和可加载块(即驱动程序)，共享库用于启动系统、运行根文件系统中的可执行程序。
boot	引导加载程序使用的静态文件
h o m e	用户主目录，包括供服务账号锁使用的主目录，如FTP
mnt	用于临时挂接某个文件系统的挂接点，通常是空目录。也可以在里面创建空的子目录。
opt	给主机额外安装软件所摆放的目录
root	root用户的主目录
tmp	存放临时文件，通常是空目录。
var	存放可变的数据

根文件系统

- 建立JFFS2根文件系统
  - JFFS2
    - 作为根文件系统
    - 作为普通文件系统在系统启动后被挂载
  - 根文件指的是Romfs、var和/tmp，目录采用Ramfs，当系统断电后，该目录所有的数据都会丢失

根文件系统

- Linux下常用文件系统结构



根文件系统

- 实验：建立根文件系统

THANKS!