





嵌入式系统

An Introduction to Embedded System

第八课、嵌入式文件系统

浙江大学计算机学院人工智能研究所
航天科技－浙江大学基础软件研发中心

课程大纲

-  嵌入式文件系统概述
-  闪存存储器硬件特性简介
-  闪存存储器文件系统简介
-  嵌入式文件系统实验内容说明

课程大纲

 嵌入式文件系统概述

 闪存存储器硬件特性简介

 闪存存储器文件系统简介

 嵌入式文件系统实验内容说明

嵌入式文件系统介绍

- 嵌入式文件系统是在嵌入式系统中应用的文件系统，是嵌入式系统的一个重要组成部分。
- 随着嵌入式系统的广泛应用，对数据存储和管理提出了很高的要求，嵌入式文件系统的重要性不断突出。
- 嵌入式文件系统与桌面通用文件系统有较大差异：
 - 文件系统占用资源应尽可能小；
 - 满足可移动和便于携带的要求；
 - 满足断电后的数据完整性保护；
 - 满足抗辐射、单粒子翻转纠错；
 - 满足存储节能管理与设计需求。

嵌入式文件系统主要分类

□ ROM文件系统

- **ROMFS**是一种只读文件系统，因而可以做得很小

□ 磁盘文件系统

- **FAT16、FAT32、Ext2FS、NTFS**





□ Flash文件系统

- **TrueFFS、MFFS**
- **JFFS/JFFS2、YAFFS/YAFFS2**
- **FAT16、FAT32、NTFS、exFAT**

□ 内存文件系统



课程大纲

-  嵌入式文件系统概述
-  闪存存储器硬件特性简介
-  闪存存储器文件系统简介
-  嵌入式文件系统实验内容说明

课程大纲

 嵌入式文件系统概述

 闪存存储器硬件特性简介

 闪存存储器文件系统简介

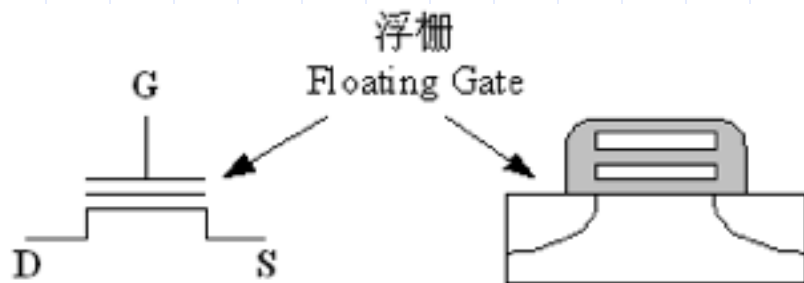
 嵌入式文件系统实验内容说明

闪存存储器（固态硬盘）概述

- 闪存（**Flash**）是一种固态非易失性存储器，主要依靠储存电荷保存数据，而不是移动电磁介质。
- **Flash**没有移动部分并且相对稳定，具有抗机械震动、轻巧、紧凑、节能等良好特性。
- 随着闪存容量的增加，价格的下降，越来越多的嵌入式系统采用闪存作为存储设备，如：电子盘（**U盘**）、手机、**PDA**、**MP3**、数码相机等，并且主板**BIOS**也已采用闪存。
- **2008年NAND**闪存市场的终端产品出货值是**2004年**的三倍多，从**50亿美元**激增到**180亿美元**。

闪存硬件基本原理

- 当前主流的非易失性存储器是：**EEPROM**、闪存存储器，两者都是电可擦写的，基于浮栅**MOS**晶体管构造而成。



- 浮栅**MOS**晶体管的特点是门限值可以根据浮栅捕获的电荷来调节，电荷捕获技术是非常可靠的，基于浮栅**MOS**的存储器可以经受**1000000**次重写操作，保存时间长达**10**年。
- **EEPROM**可以一次性擦除和重写一个单元，而闪存的源级连⁷在一起，只能以块（**block**）为单位进行擦除，闪存的擦除、

闪存发展及分类

- ❑ 闪存由**Toshiba**公司**1980**年申请专利，在**1984**年的国际半导体学术会议上发表。
- ❑ **Intel**于**1988**年首先开发出**NOR**闪存结构；**Toshiba**于**1989**年开发出**NAND**闪存结构。
- ❑ 目前主要的闪存供应商有：**Samsung**、**Intel**、**AMD**、**ATMEL**、**Fujistu**、**Hitachi**、**Hyundai**、**Micron**、**Sharp**、**Toshiba**等。
- ❑ 闪存类型主要分为：
 - **NAND**、**NOR**、**DINOR**、**AND**，其中**NOR**和**NAND**是两种主要的闪存体系结构。
 - **NOR**闪存作为**EEPROM**的替代品而设计，**NAND**型闪存专门为数据⁸存储而设计。

NOR vs NAND闪存比较（1/2）

□ NOR型闪存的特点：

- 具有独立的地址线、数据线，支持快速随机访问，容量较小；
- 具有芯片内执行（**XIP, eXecute In Place**）的功能，按照字节为单位进行随机写；
- **NOR**型闪存适合用来存储少量的可执行代码。

□ NAND型闪存的特点：

- 地址线、数据线共用，单元尺寸比**NOR**型器件小，具有更高的价格容量比，可以达到高存储密度和大容量；
- 读、写操作单位采用**512**字节的页面；
- **NAND**更适合作为高密度数据存储。

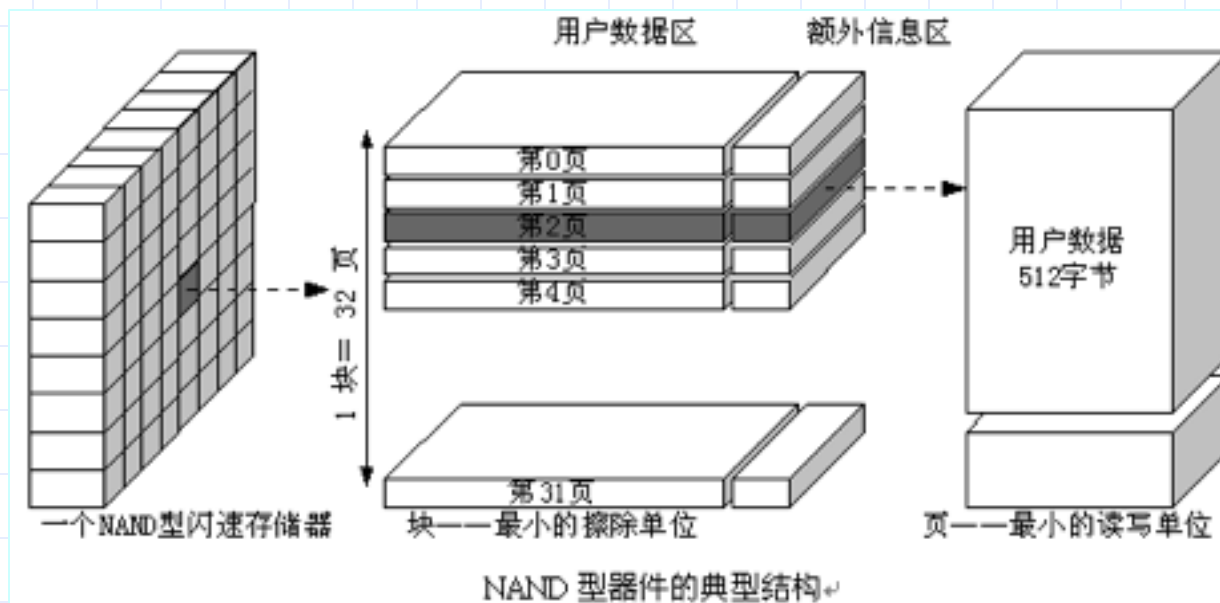
NOR vs NAND闪存比较（2/2）

性能参数	NAND型	NOR型
读操作的时间	3.5MB/sec	15MB/sec
写操作的时间	0.65MB/sec	0.15MB/sec
擦除操作的时间	2ms	1sec
擦除大小	8-32KB	64-128KB
擦除次数限制	1,000,000 次	100,000 次

- 与**NOR**型器件相比，**NAND**型器件的写入、擦除速度较快。
- **NOR**闪存带有**SRAM**接口，可以实现随机写。
- **NAND**器件使用**I/O**口串行存取数据，操作单元为**512**字节，可取代硬盘或其他块设备，需要**Memory Technology Devices(MTD)**¹⁰驱动。

闪存管理中的特殊问题（1/2）

□ 写前需先擦除，擦除单元（**Block**） > 读写单元（**Page**）：



**Nand
Flash**

- 数据更新策略：





- ✓ 本地更新（in-place update）

- ✓ 非本地更新（non-in-place update）





闪存管理中的特殊问题（2/2）

- ❑ 损耗均衡问题：闪存上的每个块都具有擦除次数的上限，被称为擦除周期计数（**erase cycle count**）。
- ❑ 为了提高闪存寿命，并减少某些块提前损坏的概率，闪存管理算法设计时应尽可能减少擦除次数，并将擦除操作均布于整个芯片。
- ❑ 位交换问题：所有闪速存储器都受到位交换现象的困扰，表现为一个**bit**位发生反转。当存储器用于敏感信息存储时，需使用错误探测/更正（**EDC/ECC**）算法提供可靠性支持。
- ❑ 坏块处理问题：**NAND**器件中的坏块是随机分布的，由于消除坏块的代价太高，因而使用**NAND**器件的初始化阶段进行扫描以发现坏块，并将坏块标记为不可用。
- ❑ 掉电保护问题：文件系统应能保证在系统突然断电时，最大限度地保护数据，使文件恢复到掉电前的一个一致性状态。

课程大纲

-  嵌入式文件系统概述
-  闪存存储器硬件特性简介
-  闪存存储器文件系统简介
-  嵌入式文件系统实验内容说明

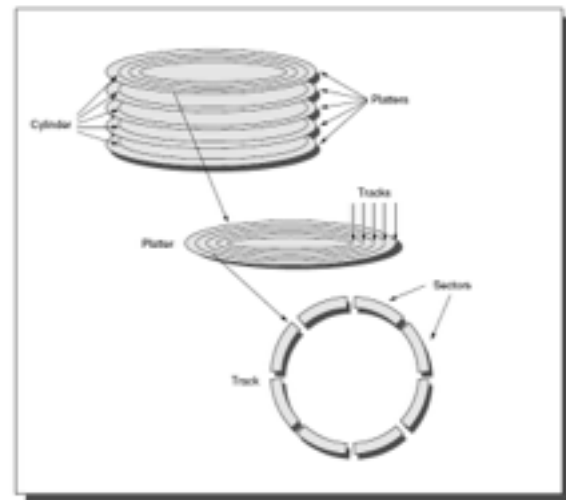
课程大纲

-  嵌入式文件系统概述
-  闪存存储器硬件特性简介
-  闪存存储器文件系统简介
-  嵌入式文件系统实验内容说明

闪存文件系统分类

□ 针对闪存设备的硬件特殊性，目前闪存文件系统主要有两种实现思路：

- 硬盘模拟法：将闪存设备模拟成具有每个扇区**512**字节的标准块设备，在此基础上使用成熟的磁盘文件系统进行管理。
- 直接实现法：直接对闪存设备进行操作，建立日志文件系统，避免模拟转化工作。

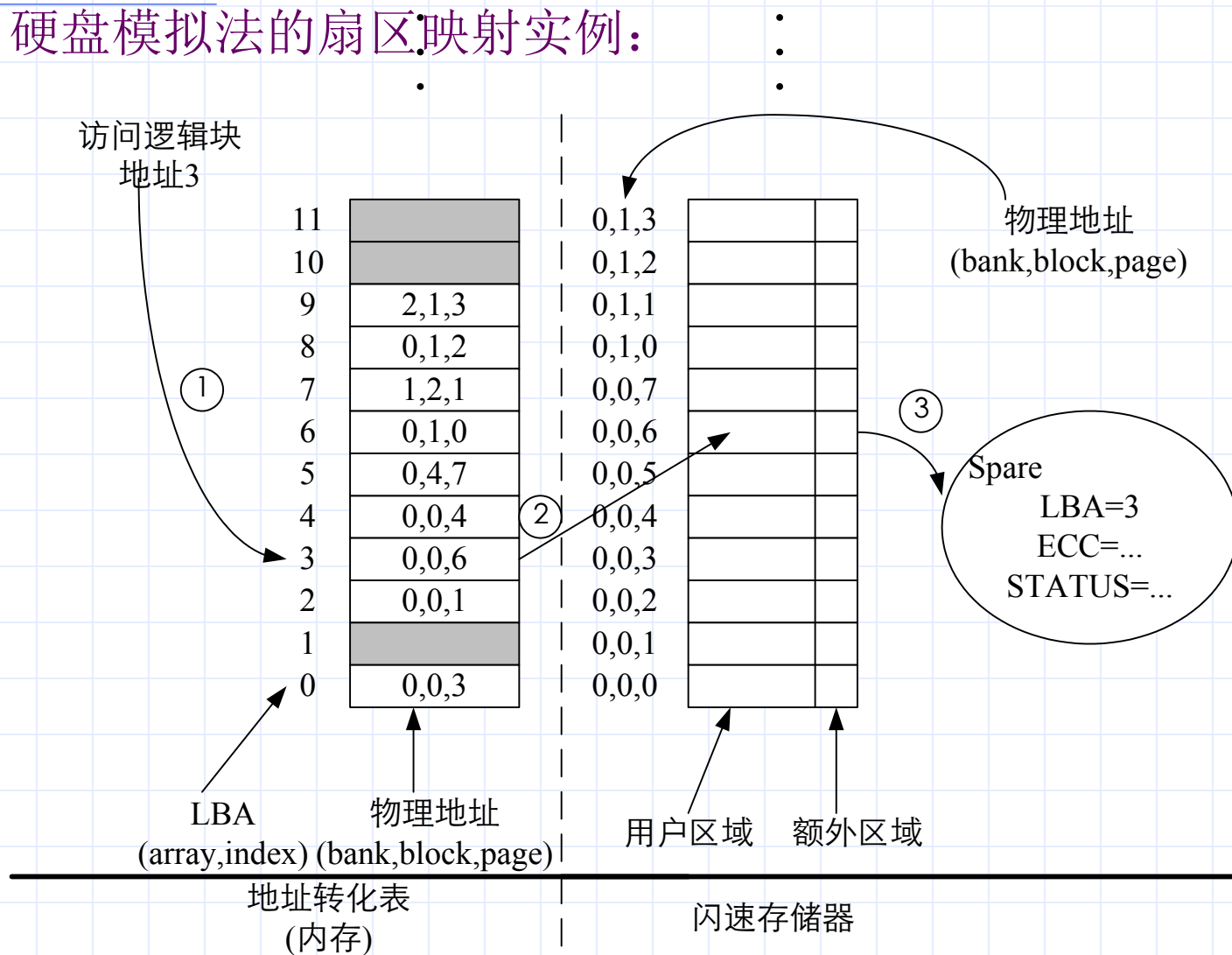


硬盘模拟法的闪存文件系统（1/4）

- ❑ 模拟方法是从模拟块设备到闪存芯片一对一的操作映射，如：模拟写请求的扇区操作时，读入整个擦除块，然后修改需要更新的部分，擦除重新写整个块。
- ❑ 硬盘模拟方法一般不考虑擦除的均衡性问题，闪存中的某些块可能会因为更新局部性迅速损坏。
- ❑ 系统一致性没有安全保证，内存随时可能断电，处于更新状态的信息会丢失，这种丢失将无法恢复。
- ❑ 为了提供均衡性和可靠的操作，模拟块设备的扇区存放在物理介质的不同位置上，地址转化层（**File Translation Layer**）用于记录当前每个扇区在模拟块设备上的位置。

硬盘模拟法的闪存文件系统（2/4）

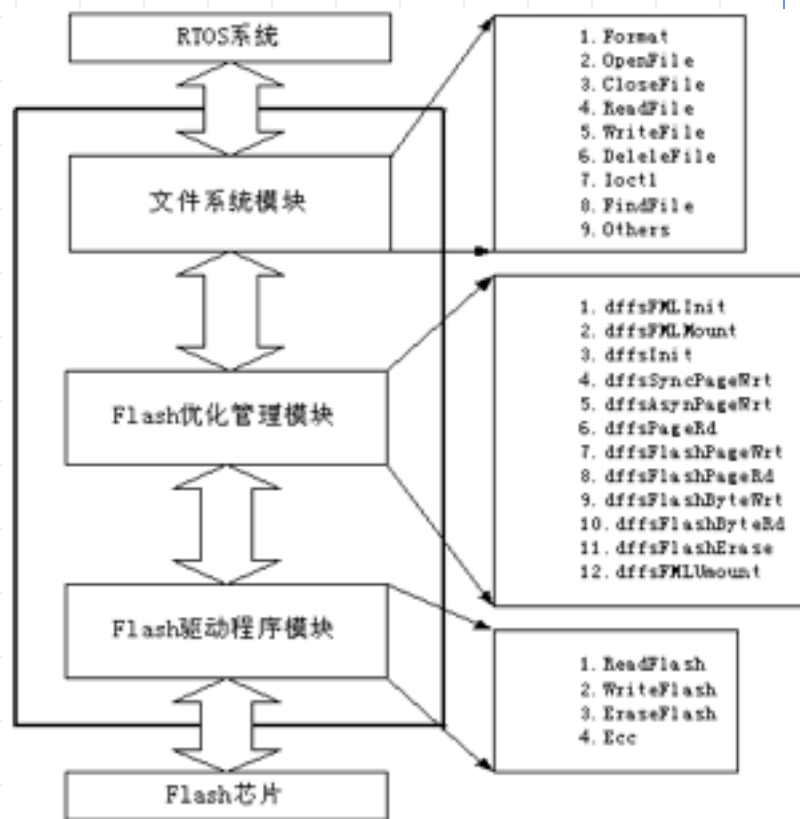
硬盘模拟法的扇区映射实例：



硬盘模拟法的闪存文件系统（3/4）

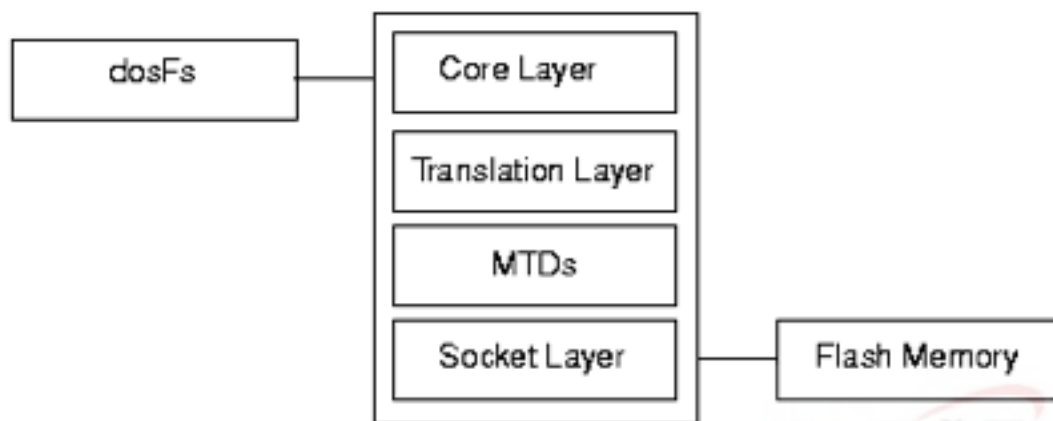
■ 硬盘模拟法的闪存文件系统实现主要分成三个层次：

- 设备驱动程序层：实现对**flash**设备最基本的操作。
- 地址转化层（**FTL**）：设备地址和逻辑地址的相互转化和数据对应关系的动态处理。
- 文件系统管理层：将文件系统操作转化成**flash**设备操作。



■ 硬盘模拟方法通用性强，移植性好，适合商业系统，便于推广，已有多种成熟的产品被广泛应用，包括**M-system**公司的 **TrueFFS**，**Intel**公司的**MFFS**等。

硬盘模拟法的闪存文件系统（4/4）



□ TrueFFS是M-system公司为支持Flash存储器专门设计的一个Flash模拟硬盘程序包。

□ TrueFFS采取多种提高Flash文件系统易用性的措施：

- 动态和静态的损耗级别判定算法；
- 安全算法保证在突然断电时数据完整性；
- 解决位交换问题的Reed-Solomon纠错算法；
- 自动的坏块管理；
- 优化处理功能，减小擦除次数的算法，优化垃圾回收操作

文件系统比较：TrueFFS vs FAT（1/4）

系统记录区	文件分配表	文件登记表	数据区
SR(System Record)	FAT(File Allocation Table)	FRT(File Register Table)	(Data Region)

■ FAT16文件系统简介

- 系统记录区（**SR**）：存储器类型、容量、版本、数据区位置
- 文件分配表（**FAT**）：存储区块占用/空闲、文件存储结构
- 文件登记表（**FRT**）：文件代号、长度、属性、时间
- 数据区（**DR**）：存放数据

分区大小 FAT16簇大小

16MB-127MB 2KB

128MB-255MB 4KB

256MB-511MB 8KB

512MB-1023MB 16KB

1024MB-2047MB 32KB

■ FAT16文件系统簇大小

文件系统比较：TrueFFS vs FAT（2/4）

□ 假设：

- 在**256M**的**U**盘上实现**FAT16**文件系统
- 簇大小为**2KB**
- 每个区块的擦除次数**10**万次

□ 向**U**盘中写入**8MB**文件

- **8M**文件共占用： $8192K/2K=4K$ 个簇
- 每写一个簇，**FAT**表更新一次，共需更新**4096**次

□ **FAT**表一直位于固定扇区中，所以**8MB**的文件最多只能更新：
如果一天备份一次，那么1个U盘的寿命只有25天！

文件系统比较：TrueFFS vs FAT (3/4)

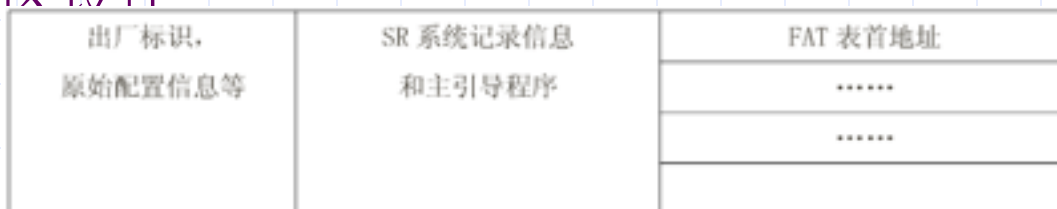
- 采用**TrueFFS**文件系统，损耗均衡算法不允许**FAT**表固定在某个扇区中，损耗平均分配给所有物理扇区。
- **U盘寿命计算**
 - **期望寿命** = (容量 × 总擦写次数 × **0.75**) / 每天写入字节，其中：
 - **0.75**：表示文件系统和**TrueFFS**管理结构的额外消耗系数
- 同样每天备份一个**8MB**文件，那么：
 - **期望寿命** = (256MB × 100000 × **0.75**) / 8MB = 2400000(天)
(约**6575**年)。
- **TrueFFS**惊人地延长了**Flash**器件的寿命。**TrueFFS**覆盖了大部分主流**Flash**芯片，考虑了各种芯片的不同擦写算法，效率较低。
- 微软的**FAT16**、**FAT32**、**NTFS**虽然实现了一定程度的损耗均衡算法，但是没有**TrueFFS**那么高效。

文件系统比较：TrueFFS vs FAT（4/4）

□ 一个种针对Flash优化的FAT16文件系统

- **FAT表、FRT表**，可以在**Flash**的一定范围内移动
- 在系统记录区（**SR**）中提供一系列**FAT表**的入口地址

□ 系统记录区设计



直接实现法的闪存文件系统（1/3）

- 使用硬盘模拟法的闪存文件系统，不能充分发挥**Flash**存储器特性，去掉**FTL**这一层转化，将对性能有很大提高，同时，磁盘文件系统的设计方法也不适合断电后数据的完整性保护。
- 在各种直接实现法的文件系统设计思路中，日志文件系统结构最符合**Flash**需要擦除的特性。

直接实现法的闪存文件系统（2/3）

- 日志文件系统结构采用了数据库系统中日志的概念，其特点是对数据的更新采用前向写入，即更新的数据部分写入空白块，而不是覆盖原先的数据，从而避免了数据块的擦除，保证了意外情况下数据存储的完整性与关联性。通过检查点、回滚技术等可以将数据恢复到某时刻的一致状态。
- 直接实现法的文件系统适用于资源少，速度要求快，以及可靠性要求高的应用场合，但缺乏通用性。

直接实现法的闪存文件系统（3/3）

□ JFFS / JFFS2

- **JFFS**文件系统主要针对**NOR FLASH**设计，是一种基于**Flash**的日志文件系统。
- **JFFS2**是**JFFS**的改进版本，改善了数据存取策略，优化了碎片整理性能，增加了数据压缩功能。
- **Linux**支持**JFFS2**文件系统，采用**MTD**（**Memory Technology Device**）驱动对**flash**的读、写、擦除等访问控制。

□ YAFFS / YAFFS2

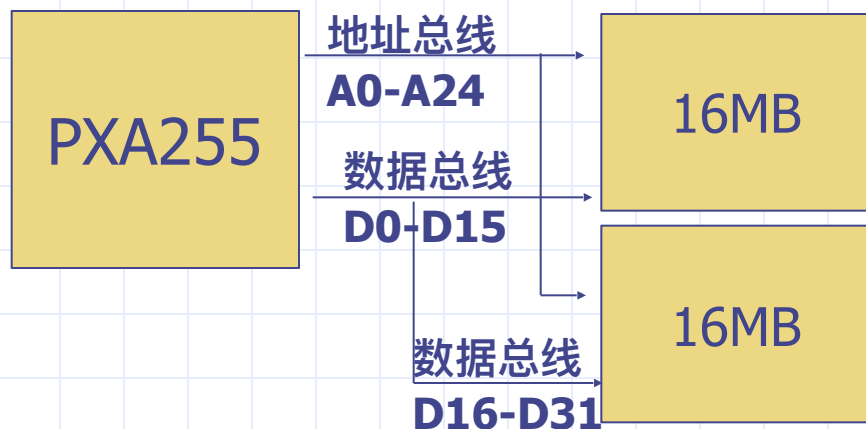
- **YAFFS**针对**NAND FLASH**设计，与**JFFS**在垃圾搜集、文件压缩支持上有所区别。
- 自带**NAND**芯片驱动，可以不采用**MTD**，直接对文件进行操作。
- **YAFFS2**是**YAFFS**的改进版本，在性能上做了优化，支持对大面

闪存驱动实例分析

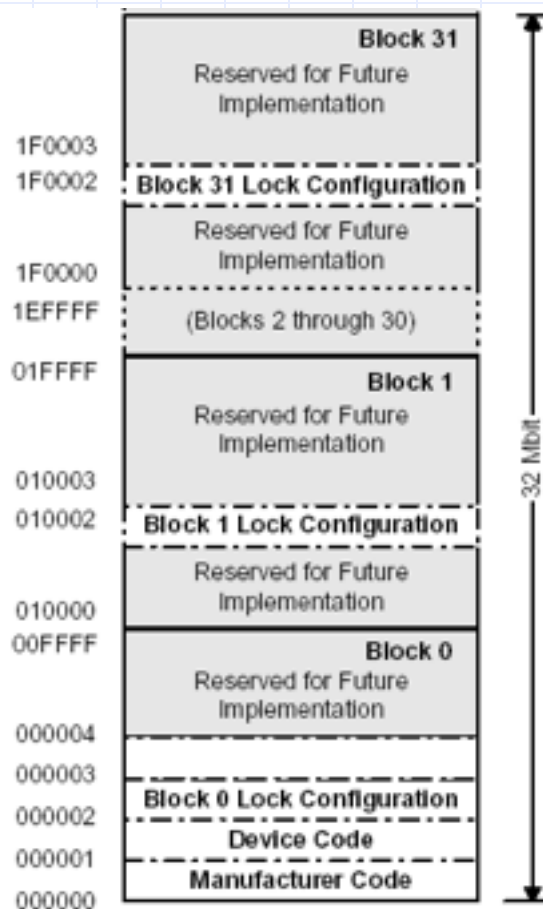
■ Intel®的StrataFlash™ (TE28F128J3C) — Nor型芯片

■ 芯片参数:

- 擦除块大小: **128KB**
- **2片16M×16位flash**并联



Flash连接图



Flash分布图

StrataFlash闪存驱动分析(1/4)

□ **Bootloader**中对闪存操作的相关指令:

- **flash** [loader/kernel/root/ramdisk]
- **erase** [loader/kernel/root/ramdisk]
- **lock** [kernel/root/ramdisk]
- **unlock**

StrataFlash闪存驱动分析(2/4)

Intel®的StrataFlash™中的指令集定义:

读状态

Flash

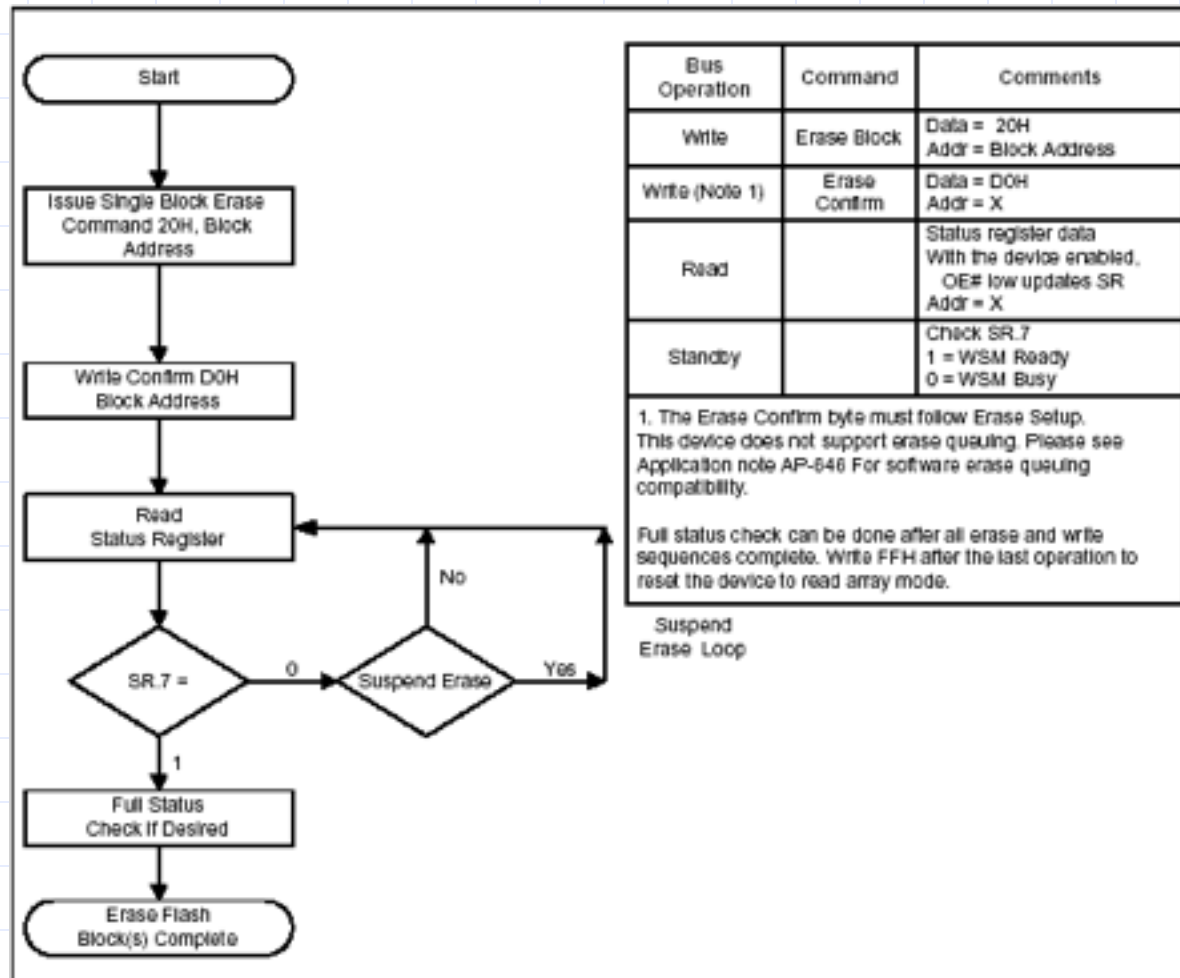
erase

lock
unlock

Command	Scaleable or Basic Command Set ⁽²⁾	Bus Cycles Req'd.	Notes	First Bus Cycle			Second Bus Cycle		
				Oper ⁽³⁾	Addr ⁽⁴⁾	Data ^(5,6)	Oper ⁽³⁾	Addr ⁽⁴⁾	Data ^(5,6)
Read Array	SCS/BCS	1		Write	X	FFH			
Read Identifier Codes	SCS/BCS	≥ 2	7	Write	X	90H	Read	IA	ID
Read Query	SCS	≥ 2		Write	X	98H	Read	QA	QD
Read Status Register	SCS/BCS	2	8	Write	X	70H	Read	X	SRD
Clear Status Register	SCS/BCS	1		Write	X	50H			
Write to Buffer	SCS/BCS	> 2	9, 10, 11	Write	BA	E8H	Write	BA	N
Word/Byte Program	SCS/BCS	2	12,13	Write	X	40H or 10H	Write	PA	PD
Block Erase	SCS/BCS	2	11,12	Write	BA	20H	Write	BA	D0H
Block Erase, Program Suspend	SCS/BCS	1	12,14	Write	X	B0H			
Block Erase, Program Resume	SCS/BCS	1	12	Write	X	D0H			
Configuration	SCS	2		Write	X	B8H	Write	X	CC
Set Read Configuration		2		Write	X	60H	Write	RCD	03H
Set Block Lock-Bit	SCS	2		Write	X	60H	Write	BA	01H
Clear Block Lock-Bits	SCS	2	15	Write	X	60H	Write	X	D0H
Protection Program		2		Write	X	C0H	Write	PA	PD

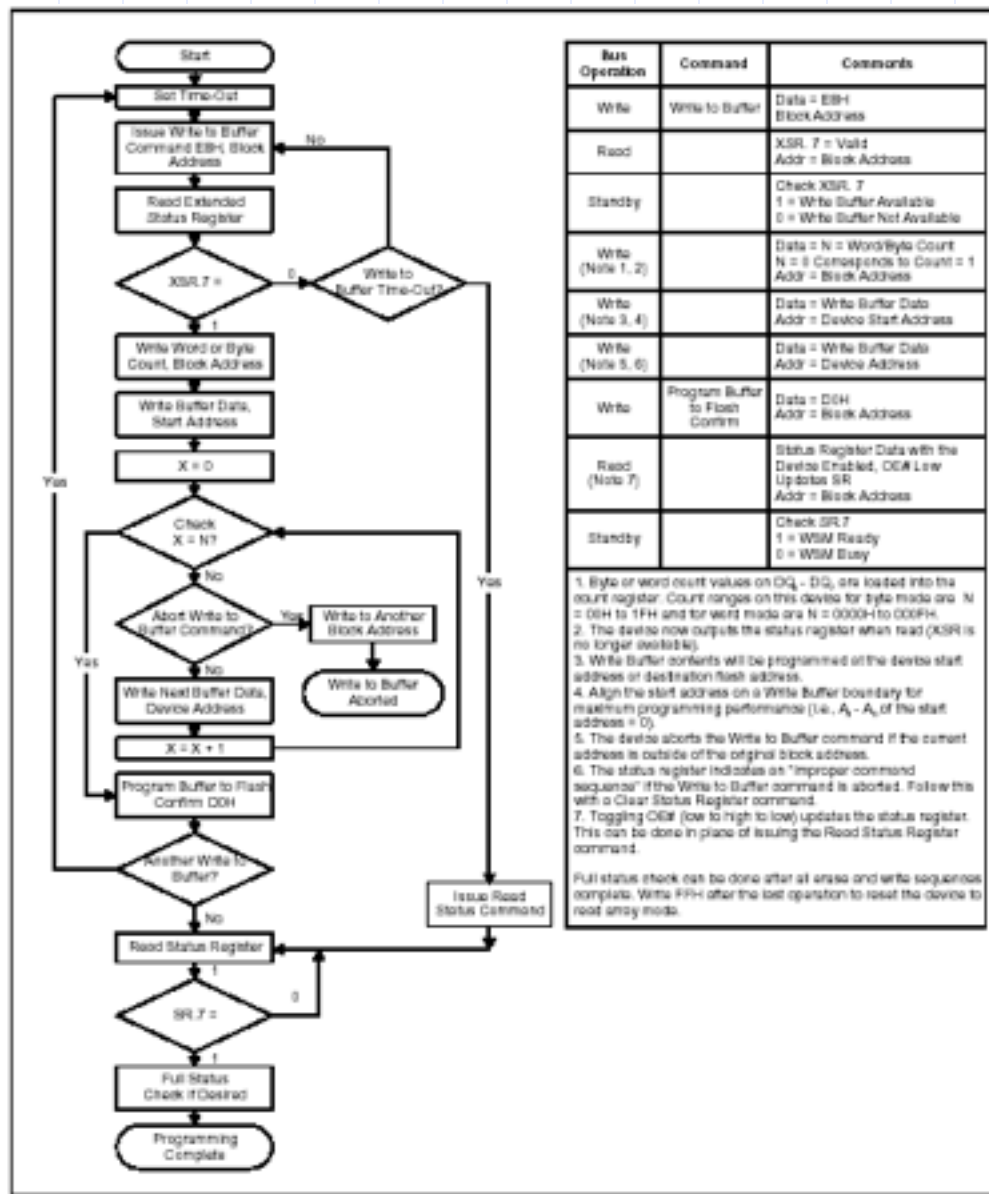
StrataFlash闪存驱动分析(3/4)

Block Erase处理流程:







StrataFlash 闪存驱动分析(4/4)





Write to Buffer 处理流程:



课程大纲

-  嵌入式文件系统概述
-  闪存存储器硬件特性简介
-  闪存存储器文件系统简介
-  嵌入式文件系统实验内容说明

课程大纲

-  嵌入式文件系统概述
-  闪存存储器硬件特性简介
-  闪存存储器文件系统简介
-  嵌入式文件系统实验内容说明

文件系统实验内容

◆ 实验一 建立Flash的JFFS2文件系统（必做）

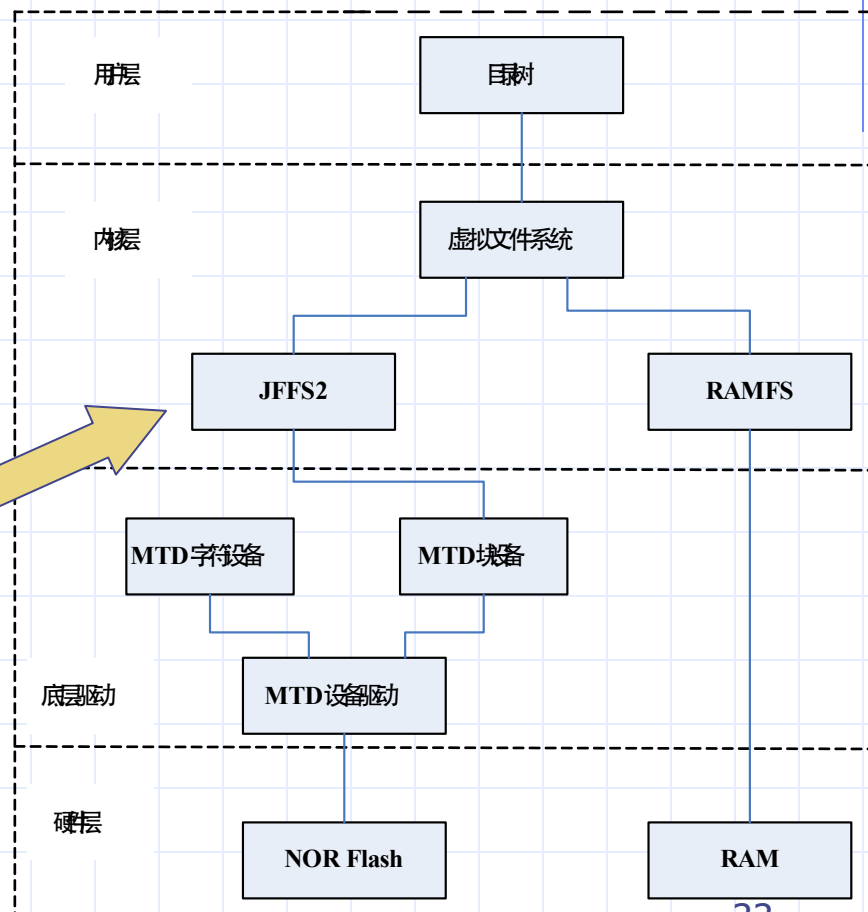
◆ 实验二 建立NFS网络文件系统（必做）

◆ 实验三 建立BusyBox文件系统（选做）

实验一 建立文件系统JFFS2 (1)

◆ 以往使用的文件系统是Ramfs，当系统断电后，文件系统的所有数据都会丢失。

◆ Linux中实现Flash的文件系统为JFFS2。

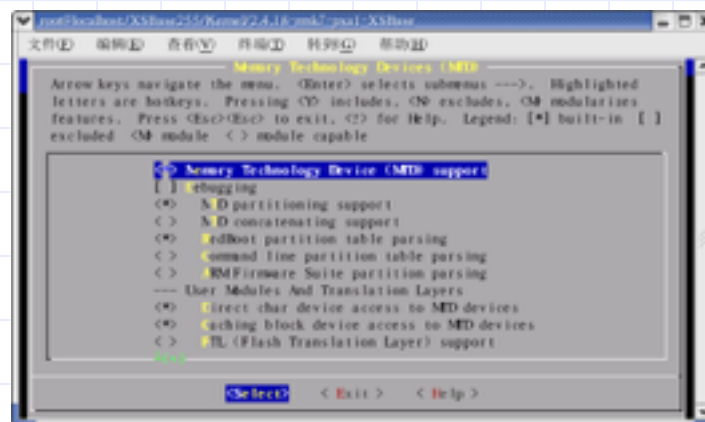
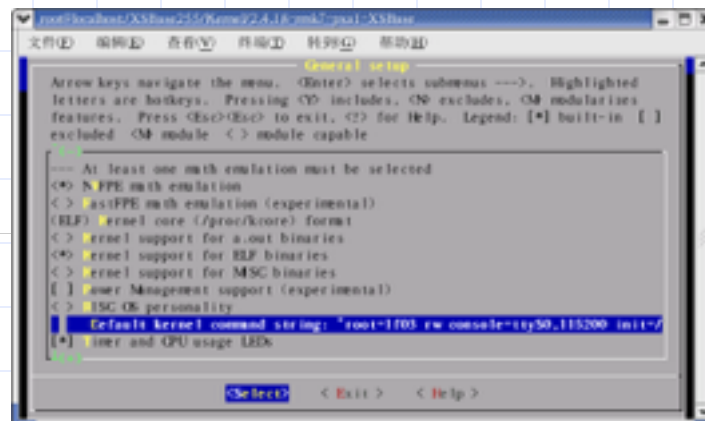


实验一 建立文件系统JFFS2 (2)



1) 内核配置

- General Setup项设成
"root=1f03 rw
console=ttyS0,115200 init=/
linuxrc"
- 通过MTD驱动在
menuconfig中调用flash
memory设备驱动
- 选择 CFI Flash device
mapped on the
XSBASE255 PXA255 board



实验一 建立文件系统JFFS2 (3)

◆ 2) JFFS2映像生成

- Jffs2 image通过 `mkfs.jffs2` 工具创建成 image
- `mkfs.jffs2` 用法: `-e` 选项确定闪存的擦除扇区大小 (通常是 64K)。 `-p` 选项用来在映像的剩余空间用零填充。 `-o` 选项用于输出文件, 这里是 `rootfs.img`

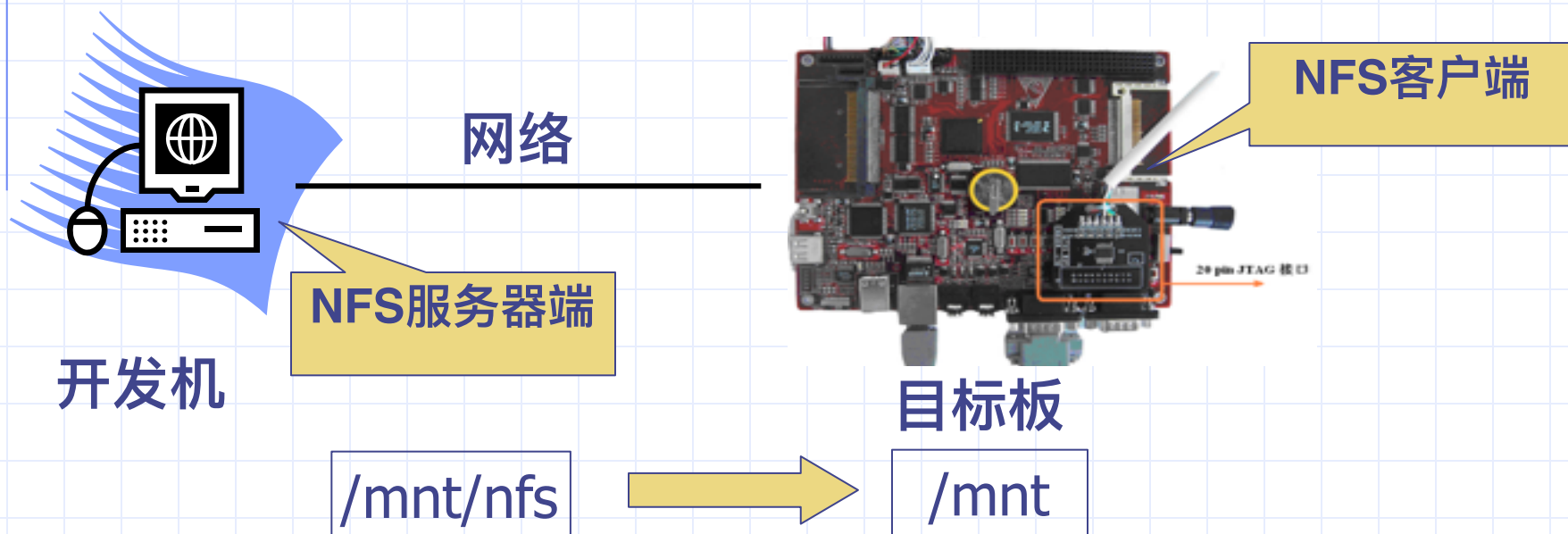
```
./mkfs.jffs2 -o rootfs -e 0x40000 -r root_XSBASE -p -l
```

◆ 3) 下载JFFS2映像

- 利用bootloader将生成的 `rootfs.img` 下载后写入flash
- 再次重起开发板, 内核就能加载JFFS2作为根文件系统

实验二 NFS文件系统实验 (1)

◆ NFS是一种用于在开发机、目标板之间，进行数据共享的网络文件系统



实验二 NFS文件系统实验 (2)

◆配置开发机端的NFS服务端

- 在开发机中打开/etc/exports文件，进行如下设置：

```
[root@XSBase home]# vi /etc/exports  
/mnt/nfs (rw,no_root_squash)
```

- 把/mnt/nfs设置完成后重新开始NFS deamon

```
[root@XSBase home]# /etc/rc.d/init.d/nfs stop  
[root@XSBase home]# /etc/rc.d/init.d/nfs start
```

实验二 NFS文件系统实验 (2)

◆配置开发机端的NFS服务端

- 在开发机中打开/etc/exports文件，进行如下设置：

```
[root@XSBase home]# vi /etc/exports  
/mnt/nfs    (rw,no_root_squash)
```

- 把/mnt/nfs设置完成后重新开始NFS deamon

```
[root@XSBase home]# /etc/rc.d/init.d/nfs stop  
[root@XSBase home]# /etc/rc.d/init.d/nfs start
```

特别提示： /etc/exports文件中的目录/
mnt/nfs，在开发机中必须手工建立

实验二 NFS文件系统实验（3）

◆ 目标板端的NFS文件系统挂载

- 目标板上的Linux内核已经支持NFS，所以不需要进行内核的再配置与生成。
- 在目标板上需进行mount操作，把开发机上的目录挂载上来：

```
[root@xsbase255]# mount -t nfs 192.168.1.1:/mnt/nfs /mnt  
[root@xsbase255]# cd /mnt
```

- 把开发机中的/mnt/nfs目录mount到目标板的/mnt目录下使用。
192.168.1.1是开发机的IP地址。
- 接下来就可以在新mount上来的目录下做各种操作。

实验二 NFS文件系统实验（3）

◆ 目标板端的NFS文件系统挂载

- 目标板上的Linux内核已经支持NFS，所以不需要进行内核的再配置与生成。
- 在目标板上需进行mount操作，把开发机上的目录挂载上来：

```
[root@xsbase255]# mount -t nfs 192.168.1.1:/mnt/nfs /mnt  
[root@xsbase255]# cd /mnt
```

- 把开发机中的/mnt/nfs目录mount到目标板的/mnt目录下使用。
192.168.1.1是开发机的IP地址。
- 接下来就可以在新mount上来的目录下做各种操作。

特别提示： 开发机的IP地址与目标板的IP地址必须在一个网段，否则无法挂载NFS文件系统

实验三 建立BusyBox文件系统（1）

◆ BusyBox——“繁忙的盒子”，是构建嵌入式Linux文件系统的常用工具。（“The Swiss Army Knife of Embedded Linux”）

◆ BusyBox最初是由Bruce Perens在1996年为Debian Linux安装盘编写的。其目标是在一张软盘上创建一个可引导的Linux 系统。一张软盘能保存1.4MB的内容，留给Linux内核及相关应用程序的空间受限。

实验三 建立BusyBox文件系统（2）

◆ 现有Linux系统存在的一些空间浪费问题

- Linux系统中的很多独立的命令程序，如grep、find、locate等命令，均具有相似的功能模块；
- 很多命令程序，均需要用到glibc库的相关调用。

◆ BusyBox构造思路

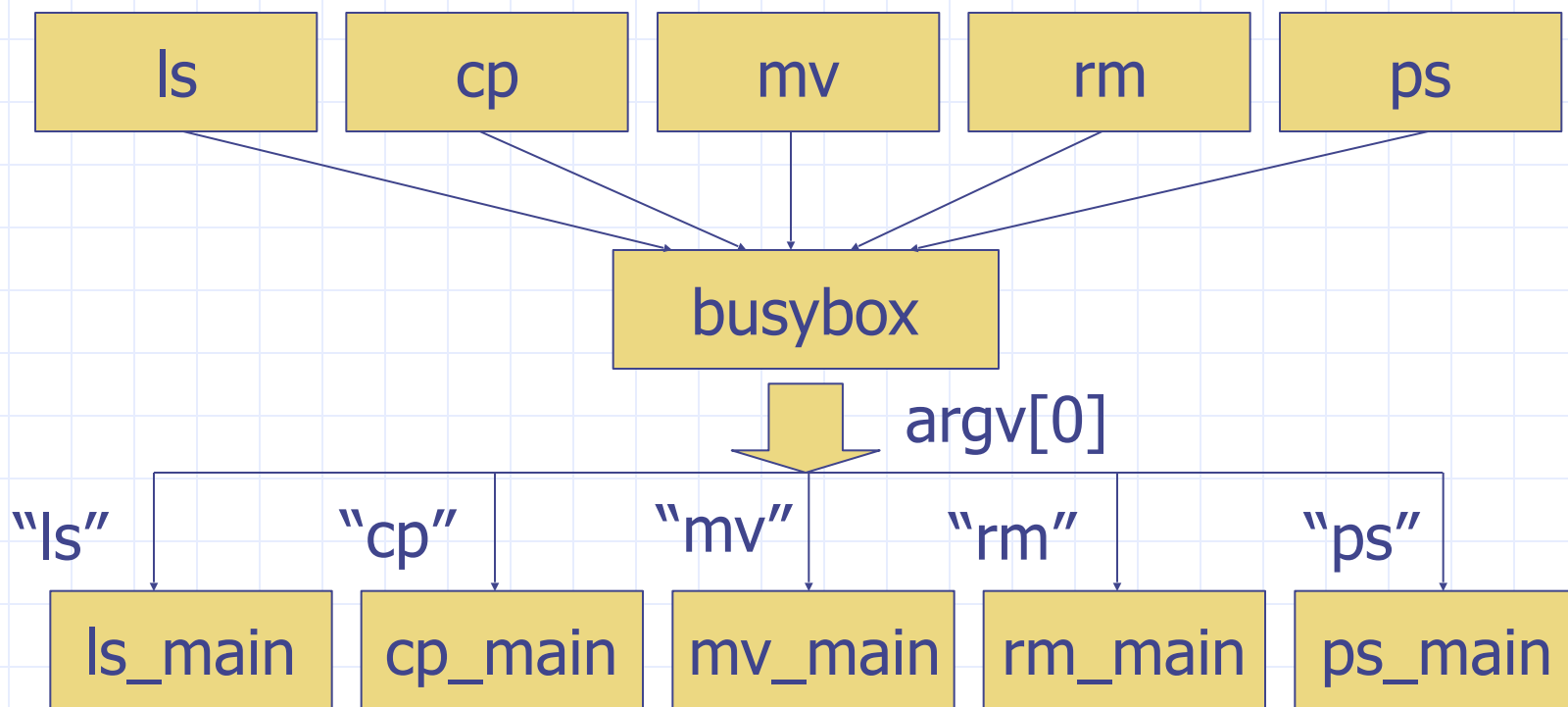
- 改造Linux系统命令程序，将所有命令合并成由一个程序（BusyBox）完成，通过符号链接的方式提供兼容，例如：`ln -s busybox ls`；`ln -s busybox cp`；
- 静态或动态链接一份glibc库，或者uclibc库。

实验三 建立BusyBox文件系统（3）

lrwxrwxrwx	1	root	root	7	May	22	23:12	ash -> busybox
-rwxr-xr-x	1	root	root	784056	May	22	23:12	busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	cat -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	chgrp -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	chmod -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	chown -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	cp -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	date -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	dd -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	df -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	dmesg -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	echo -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	egrep -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	false -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	fgrep -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	grep -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	gunzip -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	gzip -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	hostname -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	kill -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	ln -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	ls -> busybox
lrwxrwxrwx	1	root	root	7	May	22	23:12	mkdir -> busybox

所有命令，均符号链接
到一个可执行程序

实验三 建立BusyBox文件系统（4）





谢谢!

