嵌入式文件系统

提纲

- 1、嵌入式Linux文件系统简介
- 2、Linux文件系统框架和特性
- 3、嵌入式文件系统实验
  - 实验一：建立文件系统
  - 实验二：NFS文件系统实验
  - 实验三：Samba介绍与应用

1. 嵌入式Linux文件系统简介

- 嵌入式文件系统与桌面文件系统有较大区别： 嵌入式文件系统要为嵌入式系统的设计目的服务，不同用途的嵌入式操作系统下的文件系统在许多方面各不相同。
- 嵌入式Linux常用文件系统： 第二版扩展文件系统（Ext2fs）、JFFS和YAFFS

1.1嵌入式文件系统的设计目标

- 嵌入式文件系统的设计目标包括：
  - 使用简单方便
  - 安全可靠
  - 实时响应
  - 接口标注的开放性和可移植性
  - 可伸缩性和可配置性
  - 开放的体系结构
  - 资源有效性
  - 功能完整性
  - 热插拔
  - 支持多种文件类型

1.2 嵌入式Linux常用文件系统

- Flash Memory简介
- Flash Memory上的两种技术
  - NAND：串行；顺序读取；适合大容量；通常需MTD
  - NOR ： 并行；随机读取；适合数据或程序存储；XIP；
- Xsbase开发平台上所使用的闪存
  - Intel StrataFlash Memory 28F128J3A
- Ext2fs 、JFFS和YAFFS
  - ext、 ext2、xia、vfat、minix、msdos、umsdos、 proc、smb、ncp、iso9660、sysv、hpfs、affs、ufs、vfs等

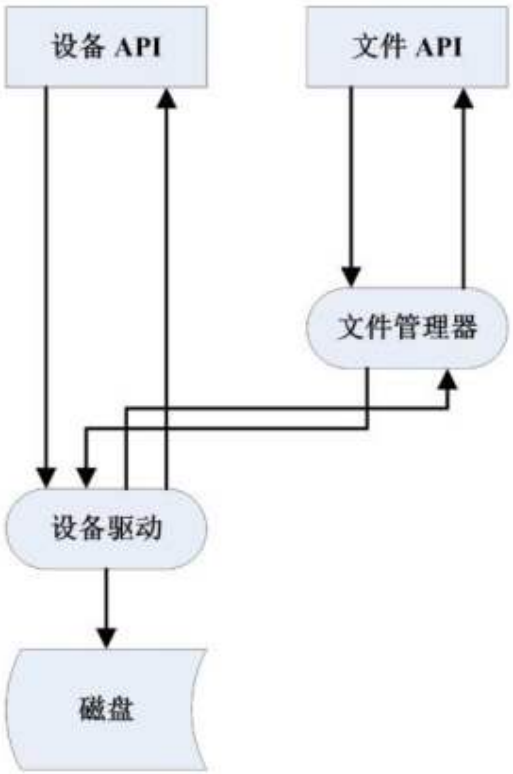
- 第二版扩展文件系统（Ext2fs）的优点
  - Ext2fs支持达4 TB的内存（Ext是2G）。
  - Ext2fs文件名称最长可以到1012个字符。
  - 当创建文件系统时，管理员可以选择逻辑块的大小（通常大小可选择 1024、2048和4096字节）。
  - Ext2fs实现快速符号链接：不需要为此目的而分配数据块，并且将目标名称直接存储在索引节点表中，这使性能有所提高，特别是在速度上。

- JFFS和YAFFS
  - 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的底层驱动主要完成文件系统对Flash芯片的访问控制，如读、写、擦除操作。
  - YAFFS主要针对NAND FLASH设计，和JFFS相比它减少了一些功能。自带NAND芯片驱动，并且为嵌入式系统提供了直接访问文件系统的API。
  - YAFFS2是YAFFS的改进版本。

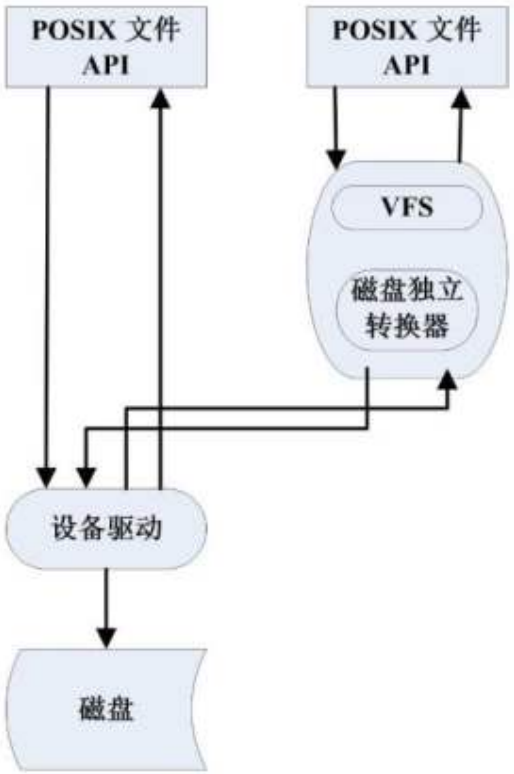
2. Linux文件系统框架和特性

- 现代操作系统都提供多种访问存储设备的方法
- Linux文件系统有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

Linux文件系统框架



（a）传统文件系统



（b）Linux文件系统

3. 文件系统实验

- 实验一　建立文件系统
  - 建立JFFS2文件系统
  - 建立RAMFS文件系统
- 实验二　NFS文件系统实验
- 实验三　Samba介绍与应用

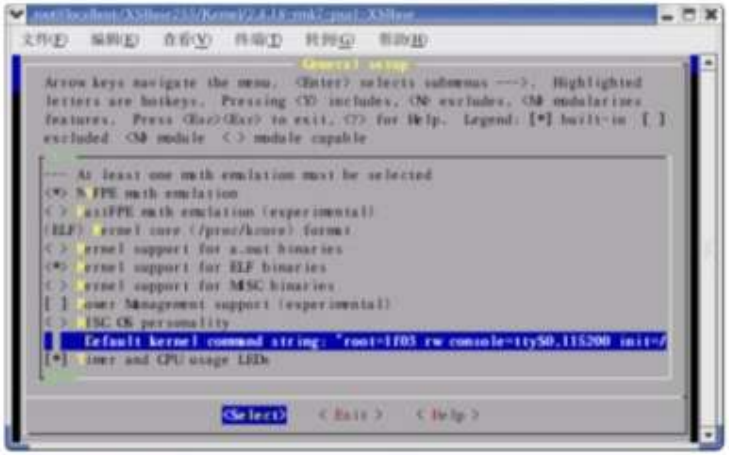
实验一：建立文件系统JFFS2（1）

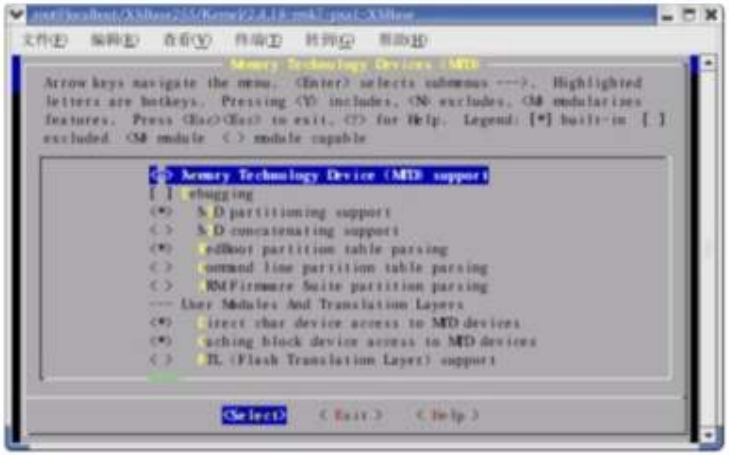
- JFFS2在Linux中有两种使用方式：
  - 作为根文件系统
  - 作为普通文件系统在系统启动后被挂载
- 目录采用Ramfs，当系统断电后，该目录所有的数据都会丢失。
- Linux下常用文件系统结构：


用户层
目录树
内核层
虚拟文件系统
JFFS2
RAMFS
MTD字符设备
MTD块设备
底层驱动
MTD设备驱动
硬件层
NOR Flash
RAM

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 0x400000 –r root_XSBASE –p -l
```

  - 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

实验一：建立文件系统RAMFS（1）

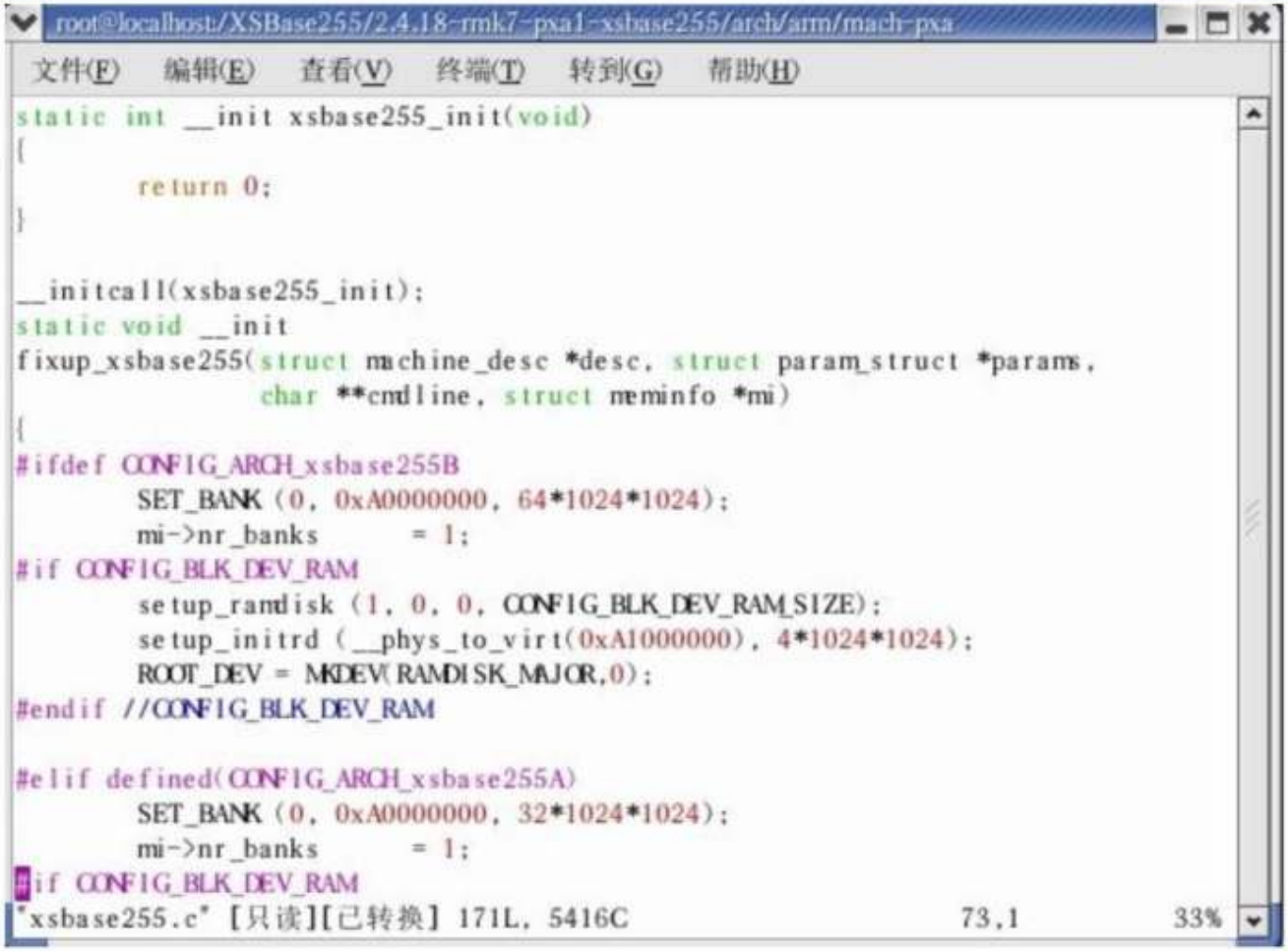
- RAMFS是内存文件系统，它工作于虚拟文件系统（VFS）层
- RAMFS是一个非常巧妙的,利用VFS自身结构而形成的内存文件系 统

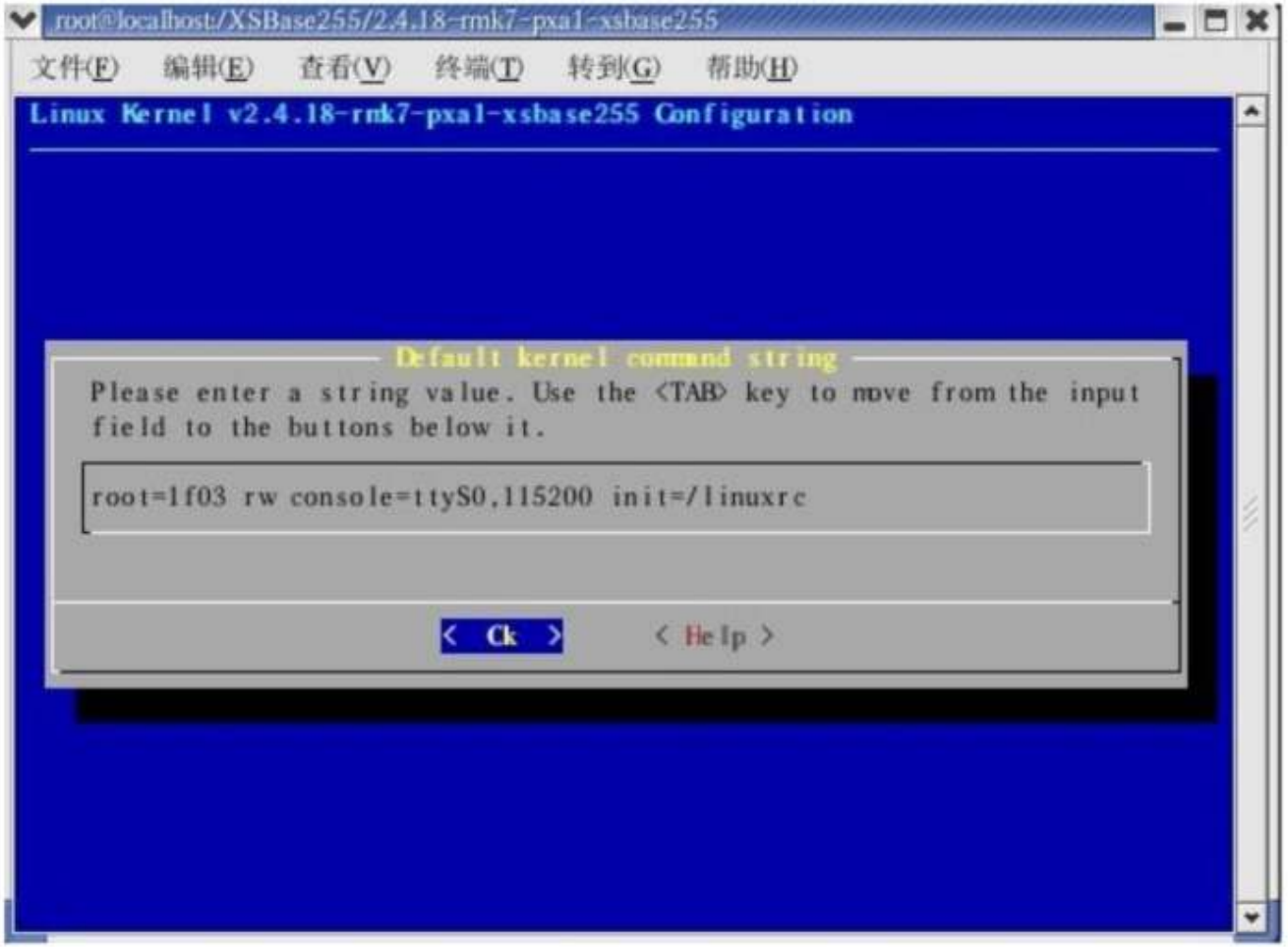
实验一：建立文件系统RAMFS（2）

- 1）使用主机的loopback设备来实现loopback文件系 统
  - 用dd if=/dev/zero of=ramdisk_img bs=1k count=8192 指令创建一个连续的8M大小的空间
  - 将此空间用 mke2fs格式化
  - 为了对格式化的空间进行mount， 创建一个目录/tmp
  - 在tmp目录里复制或创建设置文件, util, library等
  - 将工作目录进行umount，再用gzip压缩，则生成ramdisk_img.gz文件
  - 将此加载到 bootloader，则新的ramdisk可用作根文件系统。

实验一：建立文件系统RAMFS（3）

- 修改一些内核配置
  - XSBase255中 setup_initrd()的 0xA1000000 成为 SDRAM 的 ramdisk 地址，ramdisk 要下载到 Bootloader中的这个地址
  - CONFIG_BLK_DEV_RAM_SIZE是 menuconfig中设定的 “Default RAM disk size” 大小
  - 修改Default Kernel command string为：“root=/dev/ram rw console=ttyS0,115200 init=/linuxrc”。用于重新引导系统。


root@localhost:/XSBase255/2.4.18-rmk7-pxa1-xsbase255/arch/arm/mach-pxa
文件(F) 编辑(E) 查看(V) 终端(T) 转到(G) 帮助(H)
static int __init xsbase255_init(void)
{
return 0;
}
__initcall(xsbase255_init);
static void __init
fixup_xsbase255(struct machine_desc *desc, struct param_struct *params,
char **cmdline, struct meminfo *mi)
{
#ifdef CONFIG_ARCH_xsbase255B
SET_BANK (0, 0xA0000000, 64*1024*1024);
mi->nr_banks = 1;
#if CONFIG_BLK_DEV_RAM
setup_ramdisk (1, 0, 0, CONFIG_BLK_DEV_RAM_SIZE);
setup_initrd (__phys_to_virt(0xA1000000), 4*1024*1024);
ROOT_DEV = MKDEV(RAMDISK_MAJOR,0);
#endif //CONFIG_BLK_DEV_RAM
#elif defined(CONFIG_ARCH_xsbase255A)
SET_BANK (0, 0xA0000000, 32*1024*1024);
mi->nr_banks = 1;
#if CONFIG_BLK_DEV_RAM
"xsbase255.c" [只读][已转换] 171L, 5416C
73,1
33%


root@localhost:/XSBase255/2.4.18-rmk7-pxa1-xsbase255
文件(F) 编辑(E) 查看(V) 终端(T) 转到(G) 帮助(H)
Linux Kernel v2.4.18-rmk7-pxa1-xsbase255 Configuration
Default kernel command string
Please enter a string value. Use the <TAB> key to move from the input field to the buttons below it.
root=1f03 rw console=ttyS0,115200 init=/linuxrc
< Ok >
< Help >

实验二 NFS文件系统实验 （1）

- NFS是用于在不同机器，不同操作系统之间通过网络互相分享文件 的
- 建立NFS开发环境的工作分为两个方面：配置NFS服务器和配置客户端
- 基本的命令格式
  - Mount的基本命令格式如下：

```
mount -t type [-rv] -o [option] server:pathname /mount_point
```

  - Umount的基本命令格式：

```
umount [-dflnrv] dir | device [...]
```