

文章编号:1001-9081(2010)01-0118-03

## 嵌入式软件开发平台中的文件系统封装机制

何先波<sup>1,2</sup>

(1. 西华师范大学 计算机学院, 四川 南充 637002; 2. 中兴通讯股份有限公司 成都研究所, 成都 610041)

(hexianbo\_sctc@163.com)

**摘要:**针对嵌入式软件开发中操作系统多样性与应用软件领域相关性等特点,面向特定应用领域的嵌入式软件开发平台通常会对嵌入式操作系统实现功能进行封装。嵌入式操作系统提供的文件系统功能较简单,不能满足复杂软件(特别是通信领域应用软件)文件载体多样性的需求。给出了一种嵌入式文件系统封装方案,该方案把文件管理、内存设备、FTP服务器与前后台通信机制相结合,给出了相应的编程接口,并主要基于商用 VxWorks 操作系统对方案具体实现进行了探讨。

**关键词:**嵌入式操作系统;操作系统支撑层;文件系统

**中图分类号:** TP39 **文献标志码:** A

## File system encapsulation mechanism in embedded software development platform

HE Xian-bo<sup>1,2</sup>

(1. Computer School, China West Normal University, Nanchong Sichuan 637002, China;

2. Chengdu Institute, Zhongxing Telecommunication Equipment Corporation, Chengdu Sichuan 610041, China)

**Abstract:** Due to the variety of operating systems used in the embedded software development and the application correlation of embedded software, the embedded operating system is commonly encapsulated to improve its functions in the embedded software development platform. The function of file system in an embedded operating system is too simple to satisfy the function demand of complicated software such as the communication application software. In this paper a file system schema which combined file management, memory device, FTP server and foreground-background communication mechanism was presented and its implementation based on VxWorks was analyzed. In addition, the programming interface of the scheme was suggested.

**Key words:** embedded operating system; operating system support layer; file system

### 0 引言

目前,嵌入式软件在软件业中的比重正逐渐加大,所使用的操作系统一般是嵌入式实时多任务操作系统。嵌入式系统的硬件环境多样性、硬软件紧密耦合性及严格的实时性等特点决定了所使用的操作系统不可能像 Unix、Windows 等通用操作系统那样具有方方面面较强的功能。这使得针对特定领域的应用对选用的嵌入式操作系统进行再封装成为必要。面向通信领域的嵌入式操作系统支撑层(Operating System Support Layer, OSSL)是指为了能缩短通信领域嵌入式系统软件的开发周期,增强嵌入式操作系统功能和实现嵌入式程序的可移植性、可维护性和代码继承性而对所使用的特定嵌入式操作系统的再“加工”而形成的软件功能层。

针对许多商用实时操作系统核心部分源码的不开放性,支撑层一般会对文件系统、内存分配、进程调度等核心功能进行重新定义和优化,从而增强和丰富嵌入式操作系统的功能。图1展示了支撑层在统一嵌入应用软件开发平台中的位置。本文所探讨的文件系统,即为其中较重要的一个模块。

### 1 文件系统模块封装整体方案

一般而言,嵌入式操作系统的设计大多侧重于考虑实时

性与较小的内存消耗,因此其文件系统功能往往较简单。对于通信领域嵌入式设备,由于各种网络协议的实现而使得运行于其上的嵌入式软件功能越来越复杂。其文件系统部分,除了传统的硬盘文件外,还涉及到内存文件、Flash 设备文件及 FTP 服务器等相关需求。这些普适性功能模块应该集成到面向通信领域的嵌入式软件开发平台中。在面向通信领域的嵌入式应用系统中,软件可从基本功能上分为前后台模式。前台应用主要运行于嵌入式通信终端设备上,后台应用则运行于服务器或交换机上。本文提出的嵌入式软件开发平台封装层中与文件系统相关的统一框架结构如图2所示。

前台应用文件系统主要涉及到内存与 Flash 设备,后台应用文件系统则会涉及到硬盘,FTP/TFTP 服务器则用于软件版本在线更新,主备倒换等。图2框架中的文件系统可分为文件管理、内存设备和 FTP/TFTP 服务器三个子模块。

### 2 文件管理子模块

本模块为应用程序提供统一的文件操作接口,屏蔽底层操作系统和不同存储介质存取数据的具体实现细节。上层应用在存取数据时,无论对普通硬盘还是对 Flash 进行数据存取操作,都使用一致的调用接口。本模块还需提供如下功能:

1) 文件操作的写互斥、读重入机制;

收稿日期:2009-07-09;修回日期:2009-08-05。基金项目:国家863计划项目(2002AA1Z2306);四川省教育厅重点项目(08ZA015);西华师范大学校基金资助项目(08A020);西华师范大学科研启动基金资助项目(08B078)。

作者简介:何先波(1971-),男,四川苍溪人,副教授,博士,主要研究方向:嵌入式软件开发环境、通信网络。

2) 根据不同操作系统的特点,配置可打开文件数目上限;

3) 提供统计和检测手段,如当前打开的文件数目等,以利于异常的定位。

面向通信行业的嵌入式软件开发平台目前的单板中,MP板一般都配置有硬盘,而其他的单板(除去少量的接口板)一般都有Flash。为解决MP板同后台的交互、Flash访问速度过慢的问题,在前台单板上专门驻留一个文件进程,其功能主要实现接收前台应用发送的同步消息和后台发送的异步消息,以实现前台单板上文件操作。该进程可以静态配置,如果进程配置表中没有配置该文件进程,前台文件接口则直接访问前台文件。

如图3所示,借助前后台通信机制,后台与前台文件进程进行通信。后台通过向文件进程发送异步消息,实现后台对前台文件的操作与维护。

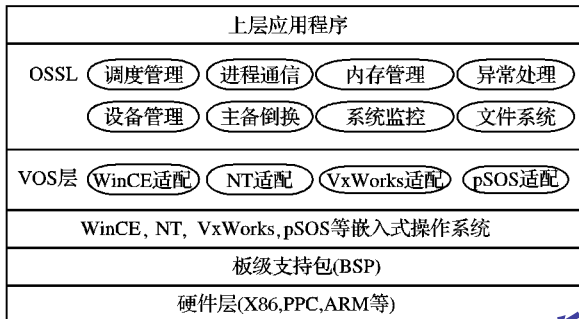


图1 嵌入式软件开发平台架构

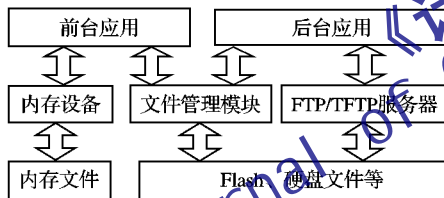


图2 文件系统模块关系框架

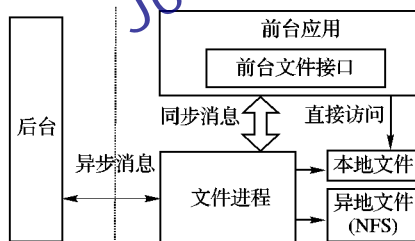


图3 文件管理示意

## 2.1 接口实现

前台文件操作接口分为文件API和同步API,两类接口使用同一套接口函数,使用方法完全相同,两种方式的变化对上层应用没有影响。文件API主要封装底层操作系统提供的非缓冲方式文件I/O系统调用(也即基本I/O),以支持及时写盘。因为用ANSI C提供的带缓存的方式进行操作,在发生重启的情况下,应用认为已经写盘的数据实际上还在缓存中,这样就有可能丢失重要的数据。如果使用硬盘,则使用文件API方式。

同步API向文件进程发送同步消息,由文件进程实现对单板上文件的操作。

前台文件接口内部实现流程图如图4所示。

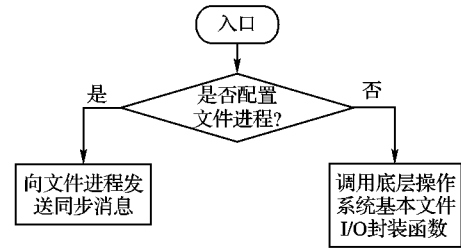


图4 接口实现流程

## 2.2 封装原理

通过提供全局文件描述表(FD表:File descriptor Table),在底层操作系统的基本文件I/O上面作一层封装,对FD表进行操作时要进行互斥保护。FD表定义见图5。

使用标志	文件名称	OS文件句柄	本次打开类型	文件所在目录	目录句柄	打开进程号

图5 文件描述表

FD表的数据结构描述如下:

```
typedef struct tagFd{
    BYTE ucUseFlag; /* 使用标志 */
    CHAR acFileName[ OSS_PathLength]; /* 文件名称 */
    WORD32 dwFileHandle; /* RTOS 的文件句柄 */
    SWORD32 sdwOpenPno; /* 打开进程 */
    WORD32 dwOpenType; /* 本次打开类型 */
    CHAR acDirName[ OSS_PathLength]; /* 文件所在目录 */
    DIR *ptDirHandle; /* 目录句柄 */
}T_Fd;
```

FD表用来保存所有文件打开的信息:文件名称、OS的文件句柄、打开进程号、本次打开类型和目录句柄等。在VxWorks中,如果把设备也看作是一个文件,缺省情况下最多可以打开50个文件。如果只考虑块设备上的文件,采用DOS文件系统时,缺省情况下可以同时打开20个文件,采用别的文件系统时,可以打开的文件个数也不一样,但都比DOS文件系统的小。由于操作系统的缺省配置可以调整,因此可配置FD表的大小,以限制同时打开的文件个数。通过FD表可以实现一些统计和检测手段,比如,当前打开的文件个数,以利于问题定位。

## 2.3 互斥机制

写互斥的判别通过文件打开类型来判断,文件打开类型规定为:OSS\_RDONLY(0),OSS\_WRONLY(1),OSS\_RDWR(2)三种类型。写互斥的判别在文件打开时进行,如果此时已经有同名文件处于读或写状态,则该次打开操作失败。对于读操作,在文件打开时,如果已经有同名文件处于读状态,仍然允许打开操作,如果已经有同名文件处于写操作,则不允许打开操作。因此在系统中要么有一个或多个任务在读文件;要么只有一个任务在写文件。

文件打开类型的互斥规则定义如表1所示(0为发生冲突;1为不发生冲突)。

表 1 互斥规则表

即将打开类型	当前已经打开类型		
	OSS_READONLY(0)	OSS_WRONLY(1)	OSS_RDWR(2)
OSS_READONLY(0)	1	0	0
OSS_WRONLY(1)	0	0	0
OSS_RDWR(2)	0	0	0

#### 2.4 文件进程主要流程

文件进程内部实现流程如图 6 所示。

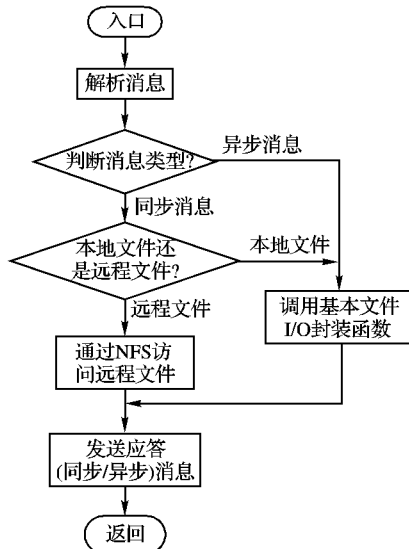


图 6 文件进程流程

### 3 内存设备子模块

内存设备为一块不受操作系统控制且大小为可配置的内存,在系统重启之间该内存中的内容不会被操作系统改写,当系统突然崩溃时,可以保存一些重要数据如发生异常时产生的现场信息和调用关系等。

基本实现方案为:在内存保留区创建 Ramdisk,并安装 DOS 文件系统。因此,可以把此 RamDisk 看成是一个小硬盘,RamDisk 提供的接口全部是内部接口。

基于 VxWorks 的建立设备文件内存功能实现算法可描述如下:

步骤 1 使用 VxWorks 的内存保留区技术,在 RAM 的顶端空出一定大小(通常 2 MB 以上)内存,该内存存在系统热重启之后,不会得到处理和清零,同时 VxWorks 不会使用该内存;

步骤 2 调用 ramDrv() 安装 Ramdisk 驱动程序;

步骤 3 调用 ramDevCreate(),从地址(sysMemTop())的返回值)开始创建 Ramdisk 设备;

步骤 4 调用 dosFsMkfs(),在该 Ramdisk 设备上安装 dosfs 文件系统。

内存设备文件产生后,其读取、写入与删除则可直接调用 VxWorks 提供的相应功能接口实现。如内存设备文件读取操作实现步骤如下:

步骤 1 根据已建立的内存设备文件名调用 VxWorks 提供的 open 函数接口打开该设备文件;

步骤 2 根据内存设备文件名调用 VxWorks 提供的 read 函数读取指定大小的内容到缓冲中;

步骤 3 根据内存设备文件名调用 VxWorks 提供的 close 函数关闭读取操作。

内存设备的写入操作实现与读取类似,只需把步骤 2 中的 read 换成 write 即可,而内存设备文件的删除可直接调用 VxWorks 提供的 unlink 接口函数实现。

### 4 FTP/TFTP 服务器子模块

通过 FTP 或 TFTP 服务器,后台可以对同一网段内前台 MP 上(包括闪存、硬盘)文件进行操作,如通信产品的软件版本在线更新便可通过 FTP 或 TFTP 实现。软件版本管理可分为两个独立的步骤:1)新版本加载,把新版本文件从 FTP 或 TFTP 服务器拷贝到需要更新的终端设备或通信单板;2)版本的分发与启动运行。

更新模块的版本后,如果新版本由于某种原因不能正确启动或出现某些严重的故障,将可能导致模块瘫痪。为此,在版本更新的同时需备份最近成功运行的旧版本,当运行新版本异常发生时,可恢复旧版本运行。版本的更新信息存放在 Flash 和引导模块中,从而既保证版本信息的安全性,又方便查询和管理。

因此,服务于软件版本更新的 FTP 或 TFTP 服务器子模块的实现主要包括两部分:

1)TFTP 或 FTP 客户端:主要由通信终端产品的板级支持包(Board Support Package, BSP)完成, BSP 可以调用 VxWorks 的系统库(如 httpLib)完成相应的功能,另外该部分程序需要识别当前的版本号,并与 TFTP 或 FTP 服务器上的版本号比较,如果不同则开始下载新版本,并保存旧版本。版本号可以写在 Flash 的固定某个地址。

2)版本管理进程。该进程运行在终端设备等需进行软件更新的嵌入式设备中。较复杂的系统存在若干相关设备的进一步更新版本分发问题,因此,多个版本管理进程需通过消息通信等方式合作完成全部更新操作。

基于 VxWorks 操作系统的功能实现主要思想为:在操作系统中加载其(如 VxWorks)本身提供的 FTP Server 组件。利用该组件,可以在指定目录上实现 FTP 服务功能。在 VxWorks 中的具体实现主要调用下面的系统调用函数:

ftpdInit(FUNCPTR pLoginRtn, int stackSize):启动 FTP 服务器任务。

ftpdDelete():终止 FTP 服务器任务。

### 5 文件系统接口规范

针对上文提出的文件系统封装方案,本文给出的面向通信领域的嵌入式软件开发台常用的编程接口如见表 2 所示。其接口实现大多是由底层操作系统再封装而成的。如内存设备文件的建立接口说明如下:

```
OSS_STATUS FS_CreateMemDev(
    CHAR *pRamDiscName,           //Ramdisk 设备名称
    CHAR *pRamAddr,               //内存地址
    SWORD32 sdwBytesPerBlk,       //块长度
    SWORD32 sdwBlksPerTrack,      //每磁道块数
    SWORD32 sdwNBlocks,           //Ramdisk 块数目
    SWORD32 sdwBlkOffset          //Ramdisk 中块的偏移量
)
```

其基于 VxWorks 的封装实现见本文 4 内存设备子模块的基于 VxWorks 的建立设备文件内存功能实现算法描述部分。

(下转第 123 页)

DGA 算法好。这说明在城市车辆网络里,节点的大部分存储空间用来缓存自己需要的数据比较好。

图 5, 图 6 展示了  $N_j\text{-access\_myself}$  权重分别为 0.5、0.6、0.7、0.8、0.9、1.0 以及 DGA 算法的模拟结果。

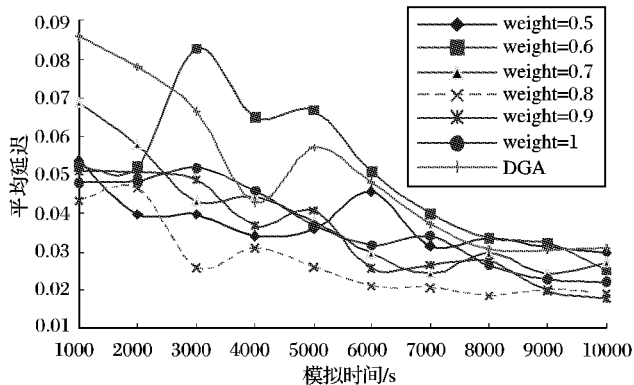


图 5 权重  $\geq 0.5$  时与 DGA 算法的平均延迟比较

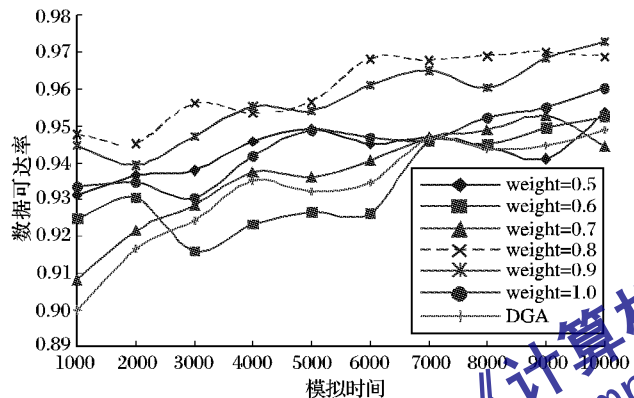


图 6 权重  $\geq 0.5$  时与 DGA 算法的数据可达率比较

从模拟结果可以看出,  $N_j\text{-access\_myself}$  的权重取 0.8 或 0.9 都是可行的, 此时的平均延迟和数据可达率都达到了较优值。

## 5 结语

本文重点研究了城市车辆网络的特点, 在此基础上, 提出了一种适用于城市车辆网络的数据缓存算法。与 DGA 算法相比, 本文提出的数据缓存算法更适合城市车辆网络, 可以达到更低的数据延迟和更高的数据可达率。最后, 通过 NS2 建立模型, 模拟验证了以上结论的正确性。

### 参考文献:

- [1] TANG BIN, GUPTA H, DAS S. Benefit-based data caching in Ad Hoc networks[C]// Proceedings of the 2006 IEEE International Conference on Network Protocols. Washington, DC: IEEE Computer Society, 2006: 208-217.
- [2] YIN LIANGZHONG, CAO GUOHONG. Supporting cooperative caching in Ad Hoc networks[J]. IEEE Transactions on Mobile Computing, 2006, 10(5): 77-89.
- [3] MUSTAFA M D, NATHRAH B. Improving data availability using hybrid replication technique in peer-to-peer environments[C]// Proceedings of the 18th International Conference on Advanced Information Networking and Applications. Washington, DC: IEEE Computer Society, 2004: 393.
- [4] ZIPF G. Human behavior and the principle of least effort[M]. Cambridge, MA: Addison-Wesley Press, 1994.
- [5] BRESLAU L, CAO P, FAN L, et al. Web caching and Zipf-like distributions: Evidence and implications[EB/OL]. [2009-05-10]. <http://www.nslj-genetics.org/wli/zipf/breslau99.pdf>.

(上接第 120 页)

表 2 常用嵌入式开发平台文件系统功能接口

接口名	功能	接口名	功能
OSS_CreateFile	产生新文件	OSS_InitFileSystem	初始化 FD 表和全局信号量
OSS_DeleteFile	删除指定文件	OSS_OpenDir	打开目录
OSS_GetFileLengt	获取指定文件长度	OSS_ReadDir	读取目录
OSS_SeekFile	文件当前读写指针定位	OSS_CloseDir	关闭目录
OSS_WriteFile	写入文件	OSS_ReadFile	读取文件内容
OSS_RenameFile	文件重命名	OSS_Create	建立文件目录
OSS_DeleteDirectory	删除指定文件目录	OSS_MoveDirectory	文件目录位置迁移
OSS_RenameDirectory	目录重命名	OSS_OpenFile	打开指定文件
OSS_ReadMemFile	内存设备文件读取	OSS_WriteMemFile	内存设备文件写入
OSS_CreateMemDev	内存设备文件建立	OSS_DeleteMemFile	删除内存设备文件
OSS_StartFtpServer	启动 FTP 服务器	OSS_StopFtpServe	停止 FTP 服务器任务

## 6 结语

本文对面向通信领域的嵌入式软件开发平台的文件系统部分给出了一种封装方案和主要基于 VxWorks 实现探讨。基于其他嵌入式操作系统的方案实现与此类似。由于这些封装方案中的子模块来源于大量通信产品软件开发的实践总结, 都是经过严格验证与测试的, 因此整个方案具有很大的普适性与较强的稳定性。

### 参考文献:

- [1] 何先波, 钟乐海, 芦冬昕. 嵌入式操作系统支撑层的设计与实现

[J]. 计算机应用, 2003, 23(5): 89-91.

- [2] 何先波, 李志蜀, 唐宁九, 等. 面向通信领域的主备倒换与数据同步技术[J]. 计算机应用, 2005, 25(10): 2312-2314.
- [3] 何先波, 李志蜀, 芦冬昕, 等. 一种面向通信领域的高效嵌入式操作系统进程队列模型[J]. 哈尔滨工程大学学报, 2006, 27(2): 263-267.
- [4] 何先波, 张芝萍, 徐立峰, 等. 一种嵌入式实时操作系统中内存分配的方法: 中国, 200410041459. 2[P]. 2004-07-13.
- [5] 何先波. 一种基于 VxWorks 的内存管理支撑层的设计与实现[J]. 西华师范大学学报: 自然科学版, 2005, 26(2): 175-179.