多操作系统平台间数据共享的设计与实现

方 正[1,2] 王 玲[1] 李 俊[1]
(中国科学院计算机网络信息中心 北京 100080)[1] (中国科学院研究生院 100039)[2]

摘 要 本文首先介绍了多种软硬件平台上实现通用数据共享的必要性,然后分析了用FTP协议和文件系统为基础的解决方案的可行性,并以两个操作系统为例,进行了方案的设计和细节实现的阐述。

关键词 数据共享,文件系统,驱动程序,FTP,Linux,WinCE

The Design and Implementation of Data Sharing among Multi-OS Platforms

FANG Zheng[1,2] WANG Ling[1] LI Jun[1]
(Computer Network Information Center, Chinese Academy of Sciences, Beijing 100080)[1]
(Graduate School of the Chinese Academy of Sciences, Beijing100039)[2]

Abstract This paper firstly introduces the necessary of data sharing among Multi-OS platforms, and then analyses the feasibility of the solution which based on FTP protocol and File System. At last, the paper makes design and implementation based on two operation systems.

Keywords Data sharing, File system, Device driver, FTP, Linux, WinCE

1 引言

随着计算机技术的发展,运算终端也越来越多样化了。这些多样性体现在硬件和软件两个方面。一方面,除了PC机之外,手机、PDA等手持设备性能越来越强,PMP、PSP等新的消费类电子设备也逐渐出现,另外IPTV、数字家电、智能家庭等新概念也离大规模应用和普及越来越近。另一方面,人们接触到的运算终端上的操作系统也呈多样性发展,服务器上UNIX和LINUX各有千秋,PC机上也有Windows、LINUX和MAC OS等并存。随着嵌入式设备性能的发展,嵌入式LINUX、WinCE和Palm等高级操作系统在手机、PDA以及新的消费类电子产品上使用也越来越广泛。

随着终端多样性的发展,终端之间数据的交互也越来越重要。因为各个平台的数据存储方式以及处理软件的差异非常大。比如,某个家庭内有一个局域网,连接了所有的终端。机顶盒录制的电视节目,想要通过电脑刻录,或者带摄像头的PDA拍摄的录像,想通过机顶盒放出来。由于各个平台不一定有兼容的存储方式或者容量,就算通过网络进行传递,那么各个平台上的软件必须能够支持相互兼容的通讯协议。这是不大可能的,厂商们不会制定这种应用层并且概念模糊的标准。所以,需要有一种在现有多样性平台上解决这一问题的通用方法。

本文首先介绍了当前进行数据交互的几种主要方式以及各自的优劣,然后提出了一种基于FTP通讯协议的解决方案,并且详细介绍了在LINUX和WinCE这两个在各种硬件平台上都应用广泛的操作系统下的设计与实现。

2 现有数据交互方式的优劣和解决方案

现有的数据交互应用最多的方式就是通过移动存储设备。服务器、PC机、PDA、手机、MP3等设备大部分都支持移动存储设备,其中USB存储器是最广泛使用的。这种方式的缺点就是需要人来操作这一数据传输过程。网络解决了这个问题,随着无线局域网等网络技术和应用的发展,以PC机为代表的个人终端,以机顶盒为代表的信息家电,和以PDA为代表的手持设备,采取使用网络来进行相互间的数据交互是发展的趋势。但是即使有了网络,信息的整合也并不是一件容易的事情。

UNIX家族的操作系统(包括LINUX)一般都支持NFS(Net File System),可以通过网络共享文件,Windows和WinCE支持CIFS(Common Internet File System)来进行文件共享。这些机制能够使远程的存储资源看起来像是在本地一样,软件可以像对待本地文件一样进行本地化操作,而不需要拷贝到本地。拷贝操作一方面需要时间,另一方面,终端不一定有足够的存储空间。但是,这些方案都和操作系统相关性强,不同的操作系统相互很难通用,特别在除这些之外的其他的操作系统上更加难以实现。

在跨平台的网络通讯协议里,HTTP或许是一个选择,随着发展,HTTP也支持丰富的在线多媒体机制,但是架设和管理一个Web Server不是一件容易的事情,特别是对于嵌入式设备来说难度和对性能的影响都很大。另外,这个方案不能兼容现有的软件模式,需要所有的软件都支持HTTP协议和在线处理。

综合起来,可行方案需要有几个特点:能够跨平台,实现和使用尽量简单,兼容现有的软件。所以,本文提出一个方案,借鉴NFS和CIFS的原理,通过FTP协议,将远程终端上的FTP服务器,映射为本地终端的一个磁盘,实现方便的数据交互。

协议选取FTP协议,主要是由于FTP协议本身和文件

方 正 硕士研究生,主要研究方向为网络信息安全、网络监控、网络协议分析;王 玲 博士研究生,主要研究方向:网络安全、网络管理;李 俊 博士、研究员,主要研究方向为下一代互联网、网络安全、多媒体通信。

操作具有相似性。并且,FTP 协议开放、简单可靠,还能够充分的进行权限控制。另外,FTP 的 Server 在很多平台都有实现,可以直接使用,管理起来安全简单,只需要再做客户端方面的开发。映射为本地的磁盘,是为了兼容本地程序,能够像操作本地文件一样操作 FTP 服务器上的文件。另外,我们应该看到,我们讨论的应用场合主要是关于多媒体的,这些应用有一定的特点,就是数据的大批量单向传输。FTP 作为文件传输协议,跟这个应用的需求有一定的相似之处。

3 文件系统与 FTP 协议映射

程序如果想操作一个文件,我们必须能够让它的 open、close、read、write、seek 等标准文件操作函数成功实现。如果这些操作能够成功地通过 FTP 协议实现到远程的终端,那么方案才能可行。本文首先分析了 LINUX 和 WinCE 两个比较有代表性的操作系统的文件系统结构,然后分析 FTP 协议与这些操作的映射,来说明将 FTP 服务器磁盘映射为文件系统的可行性。

3.1 LINUX 文件系统模型

LINUX 使用"虚拟文件系统"作为内核子系统,为用户空间提供文件系统接口。使得用户可以直接使用 open()、read()、write()等这样的系统调用来进行文件操作。LINUX 文件系统架构如图 1 所示:Kernel 通过 VFS(Virtual File System)来做存储管理。VFS 管理具体的文件系统,如 Ext2 和 Fat,然后文件系统再在具体的设备上实现相应功能,一般来说,本地文件存储介质都是块设备(Block Device),比如硬盘、存储卡等[1]。

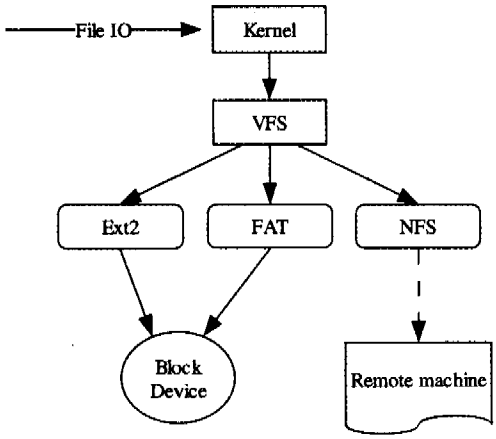

图 1 LINUX 文件系统架构

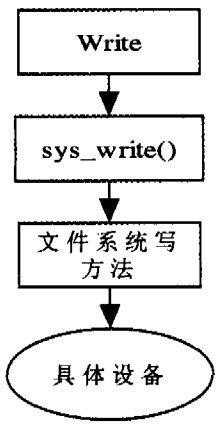

图 2 用户进程写操作

用户进程的一个写操作如图 2 所示:write 系统调用会通过 Kernel 最终调用文件系统的写方法,然后文件系统去实现到具体的物理设备上。

VFS 给出了一个高度抽象的具体文件系统实现的定义。VFS 通过 4 种对象来对某个具体的文件系统提供操作,包括:超级块对象,代表一个安装了的文件系统;索引节点对象,代表一个文件;目录项对象,代表一个目录项;文件对象,代表由进程打开的文件[1]。文件系统还需要在这 4 种对象中提供一系列的方法来让 VFS 进行操作。这些操作实现一些具体的功能,比如 mount(加载)、unmout(卸载)、mkdir(创建文件夹)、rmdir(删除文件夹)、rename(文件改名)、read(读)、write(写)、seek(移动文件指针)等等。

3.2 WinCE 存储系统模型

WinCE 的存储管理没有 LINUX 那样高度抽象,其架构也更加具体一些。WinCE 定义存储管理一共是四层,File System Filter、File System、Partition driver、Block Device。其中 File System Filter 和 Partition Driver 是可以不加载的。存储管理的架构如图 3 所示。

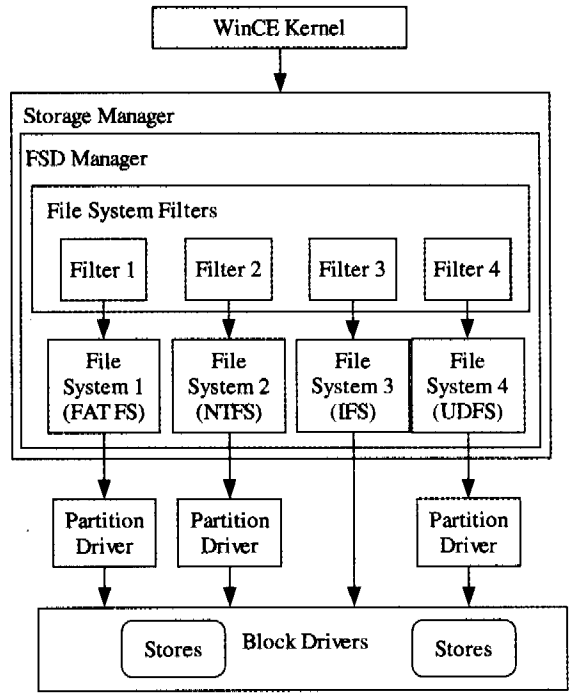

图 3 存储管理的架构

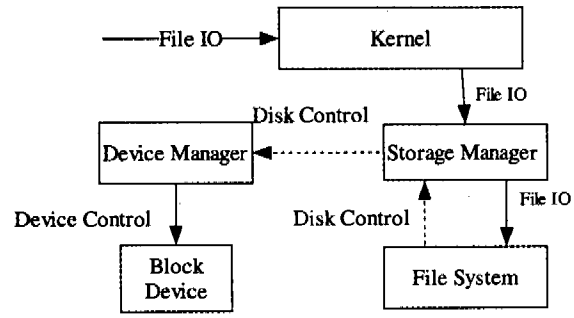

图 4 对具体 Block 设备的操作

一个文件操作的系统调用会让 Kernel 通过 FSD(File System Driver) Manager 转化为对 File System Filter 或者 File System 的调用,当 File System 想要操作具体的 Block 设

备时，FSD Manager 提供了一系列的磁盘操作接口，可以用这些接口通过 Device Manager 对具体的 Block 设备进行操作，如图 4 所示。

WinCE 的文件系统的实现是以 dll 库的形式存在的，这个库需要提供一系列的接口，主要包括：加载和卸载、目录和文件的创建和删除、文件读写以及设置指针位置、取得和设置文件属性等等。

3.3 FTP 协议和文件操作的映射

FTP 协议采用两个 TCP 的连接：一个是控制连接，用于服务器和客户端的命令和应答，另一个是传输连接，专门用于数据的传输，如图 5 所示。

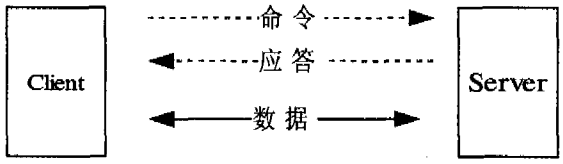

图 5 FTP 协议的连接

由于 FTP 本身就是用于文件复制的，通过分析 FTP 的命令可以发现，大多数的命令都和文件操作形成映射关系。比如 mkdir(创建目录)、mdelete(删除文件)，rest(从指定位置开始读取文件)，LIST(读取文件列表)等等。这些和上述两个文件系统所要实现的功能是比较吻合的。

综上，如果我们在 LINUX 或者 WinCE 上实现一个文件系统，然后将文件系统所需要的功能实现到远程 FTP 服务器端，就能在终端上实现将远程 FTP 服务器内容映射为本地文件夹。这将用一种简单的方式实现不同终端和平台间的数据交互，并且可以完全兼容现有软件。这个文件系统在安全性上依赖于 FTP 协议的安全性(之前 FTP 是基于明文的，新的规范则加入了 SSH 和 SSL 的支持)。因为 FTP 本身的特点，这样的文件系统会有一些自己的特征。比如，由于 FTP 可以随机的读取，但是只提供了文件上载功能，并不能实现文件的随机写操作，所以这种解决方案最好只用在读取的场合，或者在文件系统内实现本地缓存，修改完之后，再一次性上载到服务器端。

4 框架设计与实现方案

4.1 框架设计

文件系统一般都是针对块设备的，即使存储设备本身不一定物理上存在扇区，一般硬件接口也会模拟成一个块设备。WinCE 则是定义了存储管理器的对象一定是块设备。所以，为了方便起见，我们也采取文件系统加块设备驱动的架构，分别实现一个文件系统(FTP File System)和一个虚拟的块设备驱动(FTP Block Device)，操作系统专门提供的对块设备的操作，可以用来进行两个模块间的交互。设计的框架如图 6 所示。

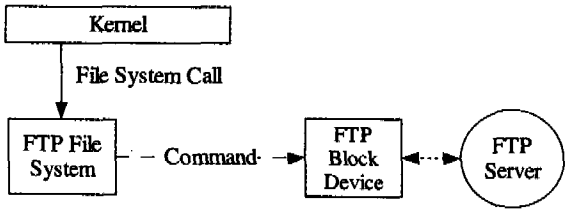

图 6 本文设计的框架

当 Kernel 对文件系统进行调用时，如果这个调用需要获取实际的信息或者数据，FTP File System 将这个解析为相关操作指令传递给 FTP Block Device，然后 FTP Block Device 与远程的 FTP 服务器进行交互，将数据返回给 FTP File System。主要过程如下(以下 FTP File System 简称为 FFS，FTP Block Device 简称为 FBD)。

(1)FFS 的加载

将远程 FTP 的 IP、端口、用户名、密码等传给 FBD，由 FBD 进行 FTP 连接。

(2)返回系统信息

FFS 直接返回预定义好的文件系统信息，如果有涉及到远程服务器的具体数据的，则要求 FBD 从 FTP 方得到。

(3)获取目录和文件信息

对目录和文件的系统查询在 LINUX 和 WinCE 下是不同的，LINUX 下要求文件系统对目录或者文件创建“索引点对象”节点，每一个节点代表一个目录或文件项，然后通过节点里的函数指针来查询目录或者文件的信息。WinCE 则是定义了若干接口函数，比如 FindNextFile(寻找下一个目录或文件)，GetFileAttribute(查询文件属性)等，要求文件系统去直接实现[3]。

虽然两个操作系统的文件系统实现会有不同，但是对于 FTP 来说都是一致的。FTP 协议可以用 List 等命令取得目录和文件的列表以及相关的信息。所以，在实现上，WinCE 下的 FFS 可以直接在接口函数中将命令发给 FBD，LINUX 下则应该在节点函数的实现里将命令发给 FBD。

(4)进行目录和文件操作

目录和文件的操作包括创建、删除等等。这些在 FTP 的命令里有直接的实现。LINUX 和 WinCE 的实现要求与上面一点的情形相似。LINUX 需要创建“目录项对象”的节点，WinCE 的文件系统需要提供 CreateDirectory 等接口函数。文件系统在节点操作函数或者接口函数中将命令传递给 FBD[4]。

(5)文件 IO

文件 IO 是文件系统最重要的部分，操作系统会传递一块内存给文件系统，要求文件系统读入或者写出指定文件指定位置的数据。在 WinCE 下，文件系统实现的 ReadFile 和 WriteFile 系列的接口会被调用；LINUX 下，“文件对象”节点中的类似 read、write、seek 等操作函数会被调用。这些调用应该将具体的读写信息以及内存传递给 FBD，然后由 FBD 映射到 FTP 的数据传递指令上实现。

(6)卸载

将命令发给 FBD，FBD 清空缓存，断开与远程 FTP 连接。

4.2 具体实现要点

由于文件系统与内核关系紧密，加之 LINUX 和 WinCE 内核架构不同，所以在实现上会有一些特殊，主要包括以下几点：

(1)文件系统的存在方式

LINUX 是“大内核”结构，而 WinCE 是“微内核”结构，如图 7 所示。在 LINUX 中，设备驱动和文件系统都是以模块的形式存在，动态或静态的“链接”在 Kernel 之中。

而 WinCE 的内核和设备管理器(device. exe)以及文件系统(fs. exe)都是独立的进程，设备驱动和文件系统都是以 dll 库的形式存在的，是动态加载到设备管理进程和文件系统进

程中。

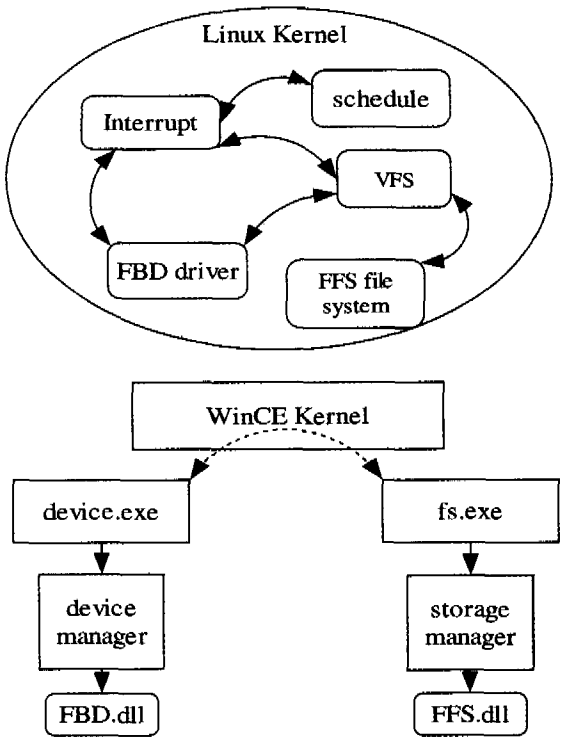

图 7 LINUX 和 WinCE 的结构

(2)模块间通讯

LINUX 各个模块是链接在一起,所以之间的通讯是通过直接调用来实现,如图 7。当然,内核的结构已经比较完善而且清晰,VFS 已经规定了 FBD 和 FFS 的函数接口。FFS 想要执行 FBD 里面的功能的时候,VFS 也提供了对标准块设备的专有操作函数,这些函数将屏蔽具体块设备的细节。

因为 WinCE 下各个模块都属于不同的进程,他们之间的通讯则是依靠进程间通讯实现的。同样,设备管理器(device manager)规定了块设备的 dll 的标准接口,存储管理器(Storage Manager)也给出了 FSDMGR ReadDisk 系列的用于磁盘操作的函数,供 FFS 与 FBD 之间通讯。

另外,模块间的通讯,特别涉及到文件读写的时候,肯定会有内存的传递,在 LINUX 下,传递的内存地址是可以直接操作的,而 WinCE 下,由于内存地址存在于各自的进程访问空间中并且受到保护,所以会不允许其他进程进行操作。WinCE 提供了 MapPtrToProcess 等系列的机制,用于将其他进程的内存地址映射到自己进程中。

(3)网络操作

网络操作的标准函数一般是应用程序来调用的,对于 LINUX,由于驱动程序存在于内核中,不能调用应用层的接口。LINUX 提供了专门用于 kernel 内进行网络操作的函数(sock_create_kern、sock_recvmsg 等)。

对于 WinCE 则没有这个限制,因为 WinCE 下的驱动程序是运行在用户空间的。可以直接像应用程序一样,使用 WinSocket 进行网络通讯。

(4)激活

实现了 FFS 和 FBD 之后,LINUX 的 mount 命令可以通过参数中在 FBD 上挂载 FFS,并且加载成指定的目录。可以通过命令行或者程序执行 mount 来进行激活;WinCE 则是在注册表中设置 FBD 的属性,指定加载的文件系统和加载后的目录。可以在启动时自动激活或者程序通过系统调用来要求激活 FBD。

结束语 本文给出的方案可以在当前软硬件平台差异都很大的情况下,实现平台间的数据共享。这个方案通用性强,既能跨各种平台,方便的实现,又能很好的实现与现有软件的兼容,实用价值相当高。本文给出的是一个通用的方案,具体实际应用中,可以在安全、性能等方面进行灵活的扩展,以满足不用平台、不同特性的场合下的需要。

参 考 文 献

1 勒伏(美). LINUX 内核设计与实现[M]. 北京:机械工业出版社, 2004
2 Microsoft Windows CE . Net 4. 2 Help. MicroSoft
3 周毓林,宁杨,陆贵强,等. WINDOWS CE. NET 内核定制及应用开发. 北京:电子工业出版社, 2005
4 Corbet J, Rubini A, Kroah-Hartman G. LINUX 应用程序开发第 2 版. 北京:电子工业出版社, 2005

(上接第 248 页)

rithm for raid data migration based on stripe unit heat. In: International Conference on Machine Learning and Cybernetics (ICMLC'2005). 2005. 3421~3427
3 Li Huai-Yang, Xie Chang-Sheng, Liu Yan, et al. The approximate parameters analysis of mS RAID. In:2005 International Conference on Machine Learning and Cybernetics (ICMLC'2005): 2005. 2753~2759
4 Scheideler C, Brinkmann A, Salzwedel K. Efficient, distributed data placement strategies for storage area networks. In: Proceedings of the 12th annual ACM symposium on Parallel algorithms and architectures
5 Ethan L, Miller R, Honicky J. A fast algorithm for online placement and reorganization of replicated data. In: IPDPS 2003 17th International Parallel and Distributed Symposium, 2003
6 Cidi S, Yao R, Zimmermann A, et al. An efficient randomized technique to reorganize continuous media blocks. In: IEEE 18th International Conference on Data Engineering (ICDE'02), 2002. 473~482
7 Ghande H, Shahabi C. Management of physical replicas in parallel multimedia information systems. In: Proceedings of the Foundations of Data Organization and Algorithm (FODO) conference, 1993. 51~58
8 Baird R, Tobagi F, Pang J, et al. Streaming raid: A disk array management system for vedio files. In First ACM Conference on Multimedia, 1993. 393~400
9 Wilkes J, Golding R, Staelin C. The HP AutoRAID hierarchical Storage System. In:Proceedings of the 15th ACM Symposium on Operating Systems Principles, Copper Mountain, 1995. 96~108
10 Santos J R, Muntz R. Performance and analysis of the rio multimedis storage system with heterogeneous disk configurations. In: ACM Multimedia, Bristol, UK, 1998. 303~308
11 Zimmermann R. Continuous media placement and scheduling in heterogeneous disk storage systems: [Ph D Thesis]. University of Southern California, 1998
12 Dan A, Sitaram D. An online video placement policy based on bandwidth to space ratio (bsr). In:Proceedings of the SIGMOD, San Jose, CA, 1995. 376~385
13 Wilkes J. Personal communication. September 1999
14 Hu Y, Yang Q. A new hierarchical disk architecture. IEEE Micro, 1998. 64~75
15 Cortes T, Labarta J. Taking advantage of heterogeneity in disk arrays. Journal of Parallel and Distributed Computing, 2003. 448~464
16 Gonzalez J L, Cortes T. Increasing the capacity of RAID5 by online gradual assimilation. In: The 13th International Conference on Parallel Architectures and Compilation Techniques (PACT), 2004
17 Cortes T, Labarta J. HRaid: a Flexible Storage-system Simulator. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, CSREA Press, 1999. 772~778
18 Seagate Corporation. Cheetah 73 Family: ST173404LW/LWV/LC/LCV Product Manual, http://www.seagate.com/support/disk/mannuals/scsi/83329478f.pdf