第27卷 第1期 电子科技大学学报 Vol.27 No.1
1998年2月 Journal of UEST of China Feb. 1998

UNIX设备驱动程序的剖析与实例*

彭寿全** 宋 杰 严海锦
（电子科技大学计算机系 成都 610054）

【摘要】 为国产操作系统COSIX1.1加入多媒体声音机制是一个新课题。文中阐明了UNIX的声卡驱动程序调用机制；重新构造了能支持声卡的UNIX内核；论述了实现声卡驱动程序的双缓冲等关键技术。

关 键 词 多媒体； 声卡； 进程； 设备文件
中图分类号 TP316

UNIX操作系统自70年代问世以来，在计算机科学界和工业界得到了广泛应用。目前，它不仅是高档微机服务器、工作站系统和小型机上的主流操作系统，而且也是大中型计算机事实上的标准操作系统。UNIX虽然具有无可比拟的优越性，但它的多媒体功能却不足。如果能把DOS，WINDOWS平台上实现的影音多媒体功能加到UNIX中去，让UNIX也实现影音图文并茂，那么，凭借UNIX强大的网络功能，并结合丰富的多媒体功能，必将推动信息产业进入当今的网络多媒体时代。这不仅对普通用户是一喜讯，同时对丰富和完善UNIX系统也将起到积极作用。因此，国家将国产操作系统COSIX配置多媒体声音功能也提上了议事日程，本文正是对研究、开发和实现此功能的小结。

1 声卡简介

为COSIX研制开发多媒体声音功能重点是编写声卡驱动程序和重配系统核心。目前，大多数工作站都支持多媒体声音功能，如SUN工作站和SG1工作站，但这些工作站使用的声卡都是专用的，其驱动程序也未公开。本研究要求使用市面上流行的CREATIVE真16位声卡。它由版本为4的DSP、CT1745混音器、MPU-401 UART方式的MIDI接口和OPT13调频立体声合成器组成，其原理如图1所示。

外部音源通过采样和16位的A/D转换后，变为数字量存储起来。数字声音数据采用脉冲编码调制PCM格式，播放时再通过D/A转换为模拟信号，放大后送入扬声器播放。采样和播放的数据传送都采用DMA方式。当设置了DMA控制器，并用采样率、传输时间和传输块大小设置DSP后，启动DMA，声卡即可开始工作。当DMA传输计数完毕，将产生EOP信号给声卡，声卡则产生中断信号，系统接受中断后进入声卡驱动程序，把数据移入或移出缓冲区，以便开始下一次处理。

对声卡的控制是借助对DSP进行读写来实现。DSP有若干可选择的I/O地址口，分别为复位口、读数据口、写命令/数据口、写缓冲状态和读缓冲状态口。向复位口写入1并等待3 ms再写入0，则复位此DSP，一般DSP用100 ms来初始化。从DSP读时先检查读状态口是否就绪，是则读；同理，发命令时，首先判断写状态口是否就绪，是则写入DSP。

* 1997年5月19日收稿，1997年6月27日修改定稿
* 电子部预研基金资助项目
** 男 58岁 大学 教授

对混音器的操作主要涉及音源选择、音量大小和输入/输出混音路径逻辑等功能。混音器使用两个连续的 I/O 口:2x4 h 和 2x5 h(x 为 I/O 基址设置)。由于混音器内部有多个寄存器,因此编程时可将要操作的寄存器的索引值写入 2x4 h 口,把要设的值写入 2x5 h 口,或从 2x5 h 口读寄存器的值。关于寄存器的索引功能等请参阅声卡资料。

对混音器中地址为 80 h 的"中断设置寄存器"和 81 h 的"DMA 设置寄存器"的操作,可实现对 DMA 通道的中断申请进行软设置。一般 16 bit 声音数据用一个 16 bit 的 DMA 通道。8 bit 声音数据用一个 8 bit DMA 通道。

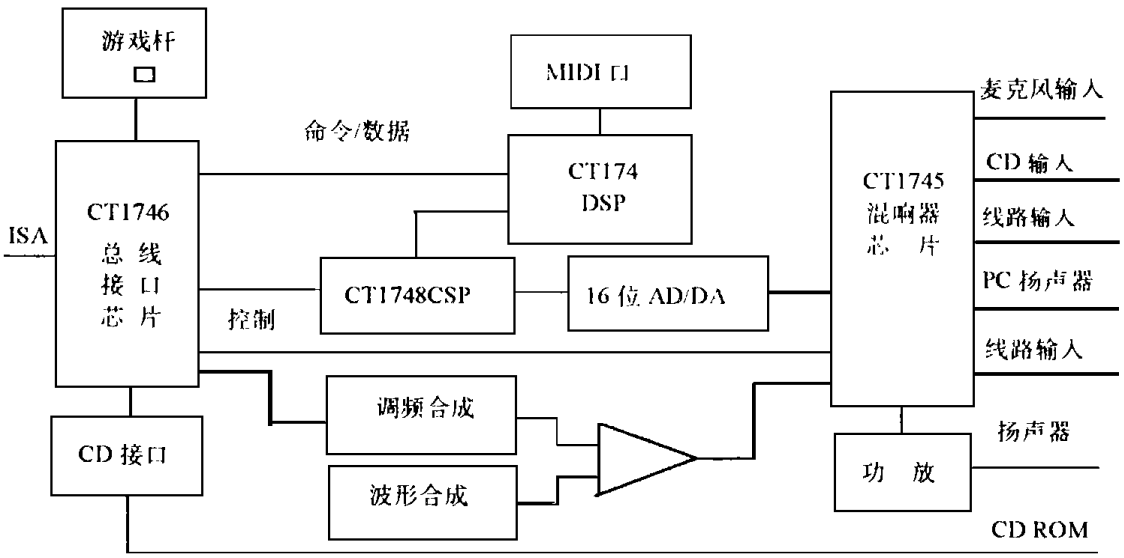

图 1　声卡逻辑框图

2　UNIX 系统的设备驱动程序及其调用

输入/输出子系统使进程能与外围设备进行通信,而控制设备的核心模块常称为设备驱动程序。通常,设备驱动程序与设备类型是一一对应的:系统可以只含有一个硬盘驱动程序来控制所有的硬盘,一个磁带驱动程序来控制所有的磁带机。系统通过主设备号来识别不同类型的设备,同类型的多个设备通过子设备号加以区分。

UNIX 系统中有两类设备:字符设备和块设备。字符设备上的数据被组织为一个字节流,进程只能以顺序方式访问它们。而块设备上的数据则是可寻址的,每个字节都有一个"块号:块内偏移量"格式的地址,进程可以按这个地址来访问数据。块设备也可以有一个字符设备接口。用户通过文件系统与设备接口,在文件系统中,设备由一个设备特殊文件来表示。每个设备特殊文件有一个索引节点,有一个象文件名那样的名字,并在文件系统目录树中占据一个节点。设备特殊文件以存储在它的索引节点中的文件类型与其他文件相区别,其类型或是"块"的,或是"字符"的,相应于它所代表的设备。

核心与驱动程序的接口是由块设备开关表和字符设备开关表来描述。每种设备在表中有相应的表项,这些表项在系统调用时引导核心转向适当的驱动程序接口。当进程进行系统调用时,核心通过文件描述符由系统打开文件表找到设备特殊文件的相应表项,然后从该表项出发,找到设备索引节点在内存索引节点表中的位置,并从索引节点中抽取出主、次设备号。有了主设备号以后,核

心就用它作为索引值进入适当的开关表,再根据用户所发的系统调用来调用驱动程序中的相应函数,子设备号此时作为函数的一个参数用于区分不同的子设备,如图2所示。

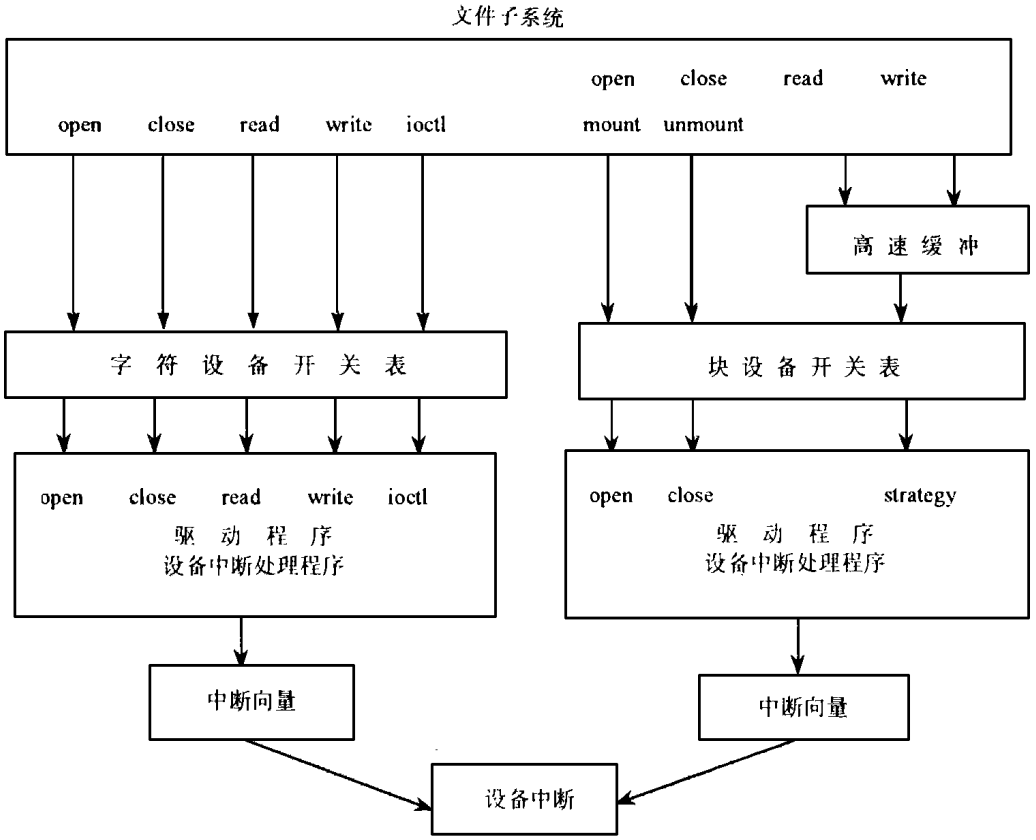

图2 系统调用

以声卡驱动程序为例,当进程对声卡设备特殊文件"/dev/aud"进行read或write系统调用时,核心就以主设备号为索引查找字符设备开关表,找到声卡的驱动程序表项后,再根据所发系统调用来调用声卡驱动程序中相应的audread或audwrite函数。此时驱动程序函数就可以对DMA控制器编程,对声卡的有关寄存器进行设置,发出命令,这样声卡就能录音和放音了。

3 内核支持声卡应采取的措施

编写了驱动程序以后,为了做到对声卡的支持,还必须重建内核。在UNIX中,有多种重建内核的方法,目前应用较为广泛的是可安装驱动程序包(IDP)。在整个创建过程中,前缀aud用来唯一标识声卡驱动程序,其创建过程如下:

1)编译驱动程序的源程序,生成名为Driver.o的目标文件。然后可用IDP命令idinstall-a-o aud将它加入到/etc/conf/pack.d/aud目录中。

2)用户必须创建一个描述声卡需要哪些硬件资源的系统设备文件System,格式如下:

aud Y 17 2 5 220 233 0 0

其中,前三个域分别表示驱动程序名、驱动程序是否将被连接到内核以及被驱动程序支持的子设备

的数目。第四，第五和第六个域分别指明 IDP 声卡的中断优先级、是否需要中断和中断向量号。第七和第八个域标明声卡占据的 I/O 地址空间。最后两个域则分别代表声卡控制器使用的内存的起始地址和结束地址。在创建过程中，这个文件可用 IDP 命令 idinstall-a-s aud 拷贝到/etc/conf/Sdevice.d/aud 文件中，也可由用户用 cp 命令拷贝。

3）建立一个描述设备接口的主设备文件 Master，格式如下：

aud　ocrwil　icDH　aud　0　0　0　1　5

第一个域说明了这个文件用于声卡驱动程序。ocrwil 表示这个驱动程序支持 open，close，read，write，ioctl 和 init 接口。第三个域表明该驱动程序是可安装的，它支持字符设备，并且要控制硬件，要使用 DMA 通道。第四个域是用于辨别接口进入点的处理程序前缀。第五和第六个域分别为用于块方式和字符方式的该设备的主设备号，将这两个数置成 0 使 IDP 能根据系统的实际情况给它们自动赋以适合的值。第七和第八个域分别标明驱动程序支持的最小和最大子设备数。最后一个域用于指示设备使用的 DMA 通道号，如果设备不使用 DMA 这个域，应置为 0。在创建过程中，Master 文件中的内容可由用户或 IDP 加到/etc/conf/cf.d/mdevice 中，用 IDP 进行安装的命令是 idinstall-a-m aud。

Master 和 System 文件说明了设备所需的硬件资源和驱动程序与内核的接口，但用户进程是通过设备的文件系统中相应的设备特殊文件来使用设备的。所以，还必须为设备在文件系统中建立相应的设备特殊文件，这项工作是由文件 Node 来完成的。格式如下：

aud　aud　c　0

第一个域说明这个文件用于描述声卡驱动程序。第二和第三个域指明作为字符设备的声卡可通过/dev/aud 来引用。第四个域是声卡的子设备号。有了 Node 文件，可用 IDP 命令 idinstall-a-n aud 将其加入/etc/conf/Node.d 目录中。

至此，声卡所有的支持文件已齐全，内核可通过 idbuild 命令进行重建。如一切顺利，在重新启动系统后，用户就能通过/dev/aud 文件来驱动声卡工作。

4　实现声卡驱动程序的关键措施

为了使声音播放效果连续，我们采用双缓冲的方式。申请两块不跨越 64 k 物理页边界且固定连续的 DMA 缓存，对 DMA 控制器和声卡编程使之对一缓存进行操作。当声卡对此缓存进行 DMA 操作时，另一缓存进行数据准备。这样，下一次就能及时转入另一块就绪的缓存，进行不间断的处理，达到连续播录的效果。

为了分配物理上连续且没有跨越 64 k 段边的 DMA 缓存，根据 Init()模块是在系统初启内存尚未打碎时运行的缘故，在 audinit()模块中分配 DMA 缓存。一般设一块 DMA 缓存为 16 k，根据鸽巢原理，一次分配连续 64 k 缓存，用滑动窗口机制，在 64 k 段边界以内，肯定能找到 32 k 在段边界以内物理连续的内存，对于其余不用的内存则释放，以免占用系统资源。

对声卡的设置是通过标准的 ioctl()接口实现的。进程通过 ioctl()调用来传递命令和参数给核心的驱动程序，驱动程序根据接收的命令用参数直接设置声卡的寄存器，从而给进程提供控制声卡的手段。

当进程在用户空间分配缓存并作预处理后，通过系统调用 Read()或 Write()陷入操作系统核心，核心通过进程 U 区指针得到用户空间缓存的地址和要处理的字节数，核心空间和用户空间的数据传输是用 DDI 函数 Copyin()和 Copyout()来完成的。核心空间和声卡上的数据传输是用 DMA 方式实现的。

对双缓冲的同步是采用UNIX中进程睡眠和唤醒的机制来进行。当声卡用DMA方式对一块缓存区操作时，进程在核心空间和用户空间移动数据，准备另一块缓存数据。准备好后如果声卡尚未结束工作，则进程调用sleep()函数使自己在内存中挂起，放弃CPU处理权，以便让UNIX系统其他的进程得到CPU，这样才符合一个多用户、多任务操作系统的要求。当声卡产生中断时，就可唤醒睡眠的进程，使它挂到就绪队列中，就能够接着运行，进行下一次处理，直到完成数字声音的处理。

声卡中断服务程序主要功能就是清除声卡中断设置位，以便接收下一次中断，然后唤醒在内存中睡眠的进程。

本驱动程序加入核心后，提供与文件系统的标准接口，用户可像对普通文件一样对声卡设备文件进行读写。

4 结束语

本文对开发和实现COSIX多媒体声音机制进行了研究。目前，COSIX1.1多媒体声音机制已经实现，鉴定证明本文对分时系统下多媒体声音机制采用的若干技术和声卡驱动程序的设计是正确可行的。

感谢卢显良教授对本文工作的支持。

参 考 文 献

1 吴元清译. 声霸开发指南. 北京：电子工业出版社，1996

2 [美] 贝奇莫里斯. UNIX操作系统设计. 北京：北京大学出版社，1992

UNIX Device Driver: Analysis and Realization

Peng Shouquan Song Jie Yan Hanjin

(Dept. of Computer Scicence, UEST of China Chengdu 610054)

Abstract Adding multi-media function for chinese open operation system COSIX1.1 is a new project. This paper describes many key techniques for its realization, including the soundblaster device driver, the kernel rebuild, and the format of device file.

Key words multi-media; soundblaster; process; device file

编辑 黄 辛