嵌入式通信设备驱动程序设计标准化

李 盛，张 扬

(电子科技大学电子工程学院 成都 610054)

【摘要】通过对Windows和Linux环境下设备驱动程序设计模型的比较，结合通信领域嵌入式系统的特点，提出了嵌入式通信系统设备驱动程序设计标准化的构想；通过参考常用的设备驱动程序的设计思想和设计模型，制定了嵌入式通信系统设备驱动程序的分层结构，统一了底层驱动程序对上层应用或管理程序的接口，屏蔽底层的硬件特性，实现了驱动程序的规范化、标准化；在VxWorks开发环境下，对设计标准进行了详细解析，并阐述了该标准制定的原因和意义。

关 键 词 嵌入式通信系统； 设备驱动程序； 标准化； 分层结构

中图分类号 TP302.1 文献标识码 A

Specification of the Embedded Communication System Device Driver

LI Sheng，ZHANG Yang

(School of Electronic Engineering, UEST of China Chengdu 610054)

Abstract This paper introduces the design standard of the embedded communication system device driver and the reason for proposing it. According to characteristics of the embedded communication system and compared with the design model of device driver in Windows and Linux, the architecture of embedded communication system device driver has been proposed. It can make the driver design much specifically and effectively. The driver design standard has been expatiated in VxWorks referring to specification of general device driver and development experience.

Key words embedded communication system; device driver; standardization; layered archit-ecture

嵌入式系统一般由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及应用程序四个部分组成。嵌入式系统已广泛应用于信息家电、移动通信、工业控制、军事电子等领域。嵌入式系统的发展和广泛应用对嵌入式软件系统的开发提出了更高的要求。

嵌入式软件系统的结构可分为操作系统、设备驱动、应用中间件和应用系统四个层次。设备驱动是整个系统的重要组成部分，是连接软硬件平台的桥梁。有了设备驱动程序，系统才能有效利用相关的硬件资源。因此，系统的效率很大程度上取决于设备驱动程序的效率。

嵌入式通信系统涵盖了大量不同种类的硬件设备，同时系统又有实时性和灵活性的要求，本文针对嵌入式通信系统的特点，将提出其设备驱动程序设计标准化的构想并对设计规范进行较详细地解析。

1 嵌入式通信系统的特点需要系统的设计标准

1.1 Windows环境下的设备驱动程序

Microsoft公司的Win2000系列和WinXP操作系统采用了WDM设备驱动程序模型[1]。从Windows3.0到

收稿日期：2004 – 11 – 19

作者简介：李 盛(1974 –)，男，硕士生，主要从事嵌入式通信系统方面的研究.

Windows2000和WindowsXP，都为其环境下的设备驱动程序制定了标准的体系结构，同时提供了丰富的系统函数封装和系统数据结构供设备驱动程序开发调用。

如图1所示，每个硬件设备在WDM模型中至少有两个驱动程序：功能驱动程序(Function Driver)和总线驱动程序(Bus Driver)。一个设备还可能通过过滤驱动程(Filter Driver)来变更标准设备驱动程序的行为。服务于同一个设备的驱动程序组成了一个链表，称为设备栈。

图2的左边描述了一个DEVICE_OBJECT数据结构栈。在数据结构栈中，最底层是物理设备对象PDO(Physical Device Object)，用于描述设备与物理总线的关系。在PDO上是功能设备对象FDO(Function Device Object)，用来描述设备的逻辑功能。在FDO的附近，有许多过滤设备对象FiDO(Filter Device Object)。数据结构栈中的每一个对象都属于一个特定的驱动程序，PDO属于功能驱动程序，FiDO属于过滤驱动程序。

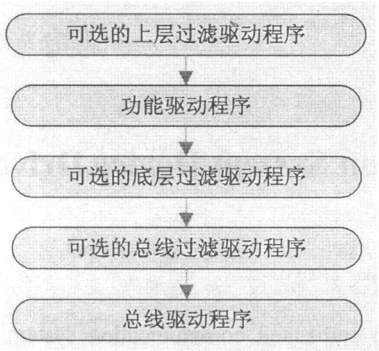

图1 WDM的分层结构

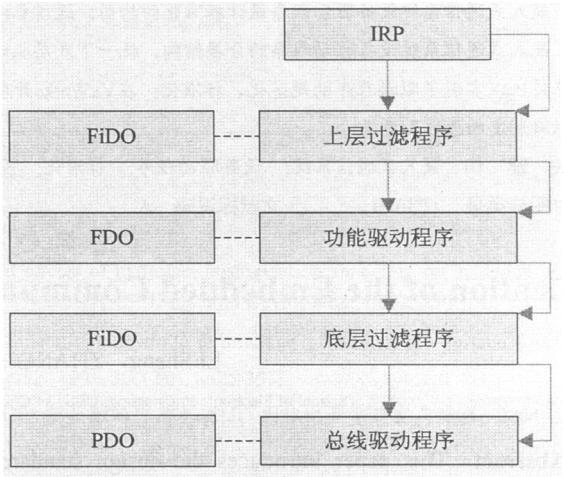

图2 WDM的分层数据结构

1.2 Linux环境下的设备驱动程序[2]

Linux的设备驱动程序要求对不同设备提供一致接口，一般是把一个设备映射为一个设备文件。如图3所示，Linux对硬件设备支持两个标准接口：块特别设备文件(Block Devices)和字符特别设备文件(Character Devices)。块设备接口仅支持面向块的I/O操作，它可以支持几乎任意长度和位置上的I/O请求，即提供随机存取的功能。字符设备接口支持面向字符的I/O操作，规定I/O请求的长度必须是设备要求的基本块长的倍数。Linux设备驱动程序可以分为三个主要组成部分：自动配置和初始化子程序；服务于I/O请求的子程序；中断服务子程序。在系统内部，I/O口的存取通过一组固定的入口来进行，这组入口是由每个设备的设备驱动程序提供的。

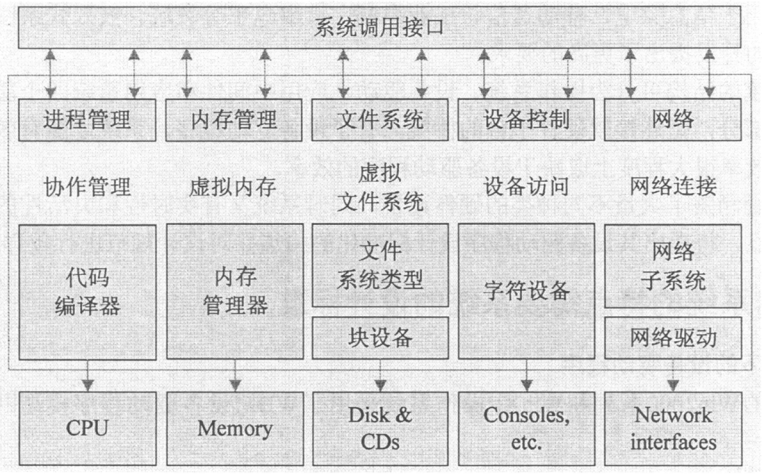

图3 Linux内核分层结构

1.3　嵌入式通信系统的设备驱动程序

嵌入式通信系统的设备驱动程序和上述环境下的设备驱动程序有很大的不同。

首先，通信领域的嵌入式系统具有实时性。要求系统能够以微秒级的速度响应外部突发事件，这要求嵌入式操作系统在事件响应过程中要尽量避免过于频繁的堆栈操作和复杂的上下文切换，所以通信领域的嵌入式系统都没有很复杂的分层结构。

其次，通信领域的嵌入式系统具有可配置性、可裁剪性和高可靠性。这要求整个系统要具有微内核结构，并且模块间有较高的相互独立性。这些特点决定了没有系统地定义设备驱动程序模型，也没有系统级的函数封装可供设备驱动程序调用。虽然VxWorks提出了标准化I/O设备的概念[3]，将I/O设备分为字符设备和块设备，并给出了如图4所示标准的ioLib。但是其标准化I/O设备有局限性，通信系统中大量使用的硬件芯片如TSI，HDLC，PCIBridge等，大多是控制类芯片，并不属于字符设备和块设备，针对这类器件的设备驱动无法纳入VxWorks标准I/O体系结构中。如何设计其驱动程序取决于设计工程师的设计思路。

最后，设备驱动程序是直接面向硬件的，任何对硬件的不正确操作，或者对中断、异常事件的不恰当处理都可能会导致系统崩溃。因此，嵌入式系统的特性需要制定一套系统的设备驱动程序规范来指导我们的开发工作。

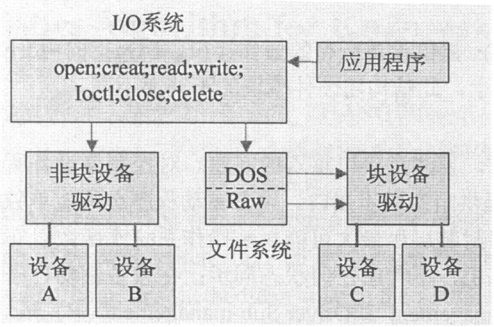

图4　VxWorks标准I/O设备

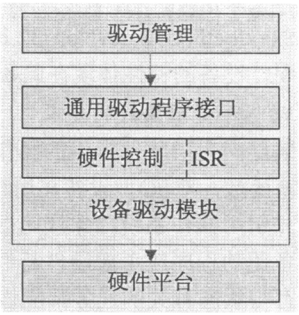

图5　嵌入式通信设备驱动程序分层结构

2　嵌入式通信系统设备驱动程序设计的标准化

2.1　嵌入式通信系统设备驱动程序设计标准的分层结构

嵌入式通信系统设备驱动程序设计标准化的目标是制定清晰的设备驱动程序的分层结构，统一上层应用或管理程序接口；提高设备驱动程序的可移植性和健壮性，减少重复开发。如图5所示，设备驱动程序在功能上分为两层[4]：硬件控制层和接口封装层。硬件控制层负责对硬件模块按功能提供各种控制接口；接口封装层负责将硬件控制层封装成标准应用接口。

2.1.1　硬件控制层

硬件控制层(Hardware Control)是设备驱动程序结构中的最底层，完成所有对硬件设备的配置和控制操作。硬件控制层直接面向硬件设备，与硬件设备的相关性最大，也是最灵活的一层。硬件控制层将硬件设备的功能实现为控制接口，由接口封装层调用。考虑到硬件设备的多样性，控制接口没有明确定义接口规范，但控制接口功能大体上可分为4类：

1) 硬件设备初始化：完成所有与硬件初始化相关的功能。

2) 硬件设备属性控制：对硬件设备属性的设置和查询。该类接口实际就是实现与硬件设备正常运行相关的寄存器的设置和查询，如串口波特率的设置和查询。对与硬件设备运行状态无关但有配置需求的寄存器，可以在设备初始化接口中完成设置。

3) 流设备的I/O操作：控制硬件发送和接收数据。控制类的硬件设备没有此类接口。

4) ISR：ISR功能上比较独立。ISR的处理要尽可能的短，应该只做一些判断处理，如错误判断等；对于接收数据及错误处理可以通过管道或消息队列，由任务来完成；ISR中不要使用导致系统阻塞的函数如

semTake()、taskDelay()等；不要有任何操作任务TCB的操作；不要使用导致系统I/O调用的函数，如printf()；ISR允许用户挂接自己的回调函数。

2.1.2 接口封装层

接口封装层(Common Driver Interface)将硬件控制层的控制接口封装成为标准API，对上层管理软件如资源管理提供一致的接口。在接口封装层，硬件设备的差异绝大部分将会被屏蔽，仅体现在对API函数数据结构的解析上。下面以资源管理模块为例对接口封装层标准API做详细描述。

STATUSxxxDrvInstall (void): 完成整个管理模块硬件的初始化工作，确保其他接口的正常工作。返回值是成功或失败。

U32xxxOpen (void *ctrlBlock): ctrl是对硬件的控制描述，包含硬件基地址，工作模式等参数。返回值是硬件设备号，用以区分模块中的多个相同硬件设备。

STATUSxxxStart/Stop (U32 devNo): 激活/停止设备工作。

STATUSxxxRecv (U32 devNo,U32 unitNo,void*sndBuf,U32 maxByte,U32 flag): 接收数据，对控制型设备，此函数为空操作，flag允许用户设置执行方式。

STATUSxxxSend (U32 devNo,U32 unitNo,void*sndBuf,U32 maxByte,U32 flag): 发送数据，对控制型设备，此函数为空操作。

STATUSxxxIoctl (U32 devNo,U32 iCmd,void*iArg): 提供设备所有的操作接口。IArg指向由driver定义的数据结构。硬件设备的特性体现在此数据结构的定义上，此结构随硬件模块的不同而不同。

2.2 设备驱动程序标准设计的要求

接口封装层向上层软件提供的操作均为元语操作，要注意接口重入的问题。对控制型设备所有接口均为同步非阻塞函数，对I/O型设备在用户允许下可以提供阻塞同步接口。设备驱动程序必须能单独编译，设计过程中优先引用操作系统抽象层函数；其次是接口封装层尽最大可能封装操作系统库函数以实现ANSIC库函数；设备驱动程序应考虑硬件特性，不包含状态机，不考虑任何状态同步；在设备驱动程序中不能包含任务，所有与状态有关的任务都应该放在Driver Management 或Driver Sub-management 中完成。

3 结 束 语

从嵌入式通信系统的开发需要统一的设备驱动程序编写标准，提出了嵌入式通信设备驱动程序设计标准化的构想，定义了较清晰的分层结构，增强了代码的可移植性、可重用性，减少了代码维护的困难，使得嵌入式通信系统设备驱动程序的设计更加规范化、标准化。

参 考 文 献

[1] Ye Kola. 底层设备驱动程序编写规范[M]. 北京: O’ Reilly, 2002

[2] Wind River. VxWorks programmer’s guide[S]. Alameda: Wind River, 2001: 187-345

[3] Alessandro R, Jonathan C. Linux device driver 2nd edition[M]. Sebastopol: O’ Reilly, 2000

[4] Grehan R. Real-time programming-a guide to 32-Bit embedded development[M]. Alameda: Wind River, 2001

编 辑 徐安玉