UNIX 系统设备驱动程序开发

丁书耕　（山东电力科学研究院计算机室　250021）

摘要：本文讨论了 UNIX 系统下设备驱动程序的概念，设备管理方式和设备驱动程序的开发方法。

关键词：UNIX　设备　驱动程序　开发

在多任务系统下，对设备的所有访问都要通过设备驱动程序进行。对于标准设备，如显示卡、网络卡、多用户卡等，生产厂家一般提供相应的驱动程序；对于非标准设备，如在控制领域中用到的各种 I/O 卡，厂家一般不提供驱动程序，需要用户自己开发；在有些情况下，标准设备的驱动程序可能不能满足一些特殊需要，也需用户自行开发。本文以 X86 系列微机上的 SCO UNIX 操作系统为例，讨论设备驱动程序的开发问题。

1. 设备驱动程序的概念

设备驱动程序是操作系统核心的一部分，其中包括一组与硬件设备进行通信并向操作系统核心提供统一接口的过程，它对用户程序和外部设备间的数据传输和控制进行管理，并使外部设备在系统内以设备文件的形式存在。操作系统核心本身就是一个特殊的可执行程序，而设备驱动程序只是其中的一部分，因此，驱动程序只相当于一个目标模块，要与操作系统组合为一体后才能起作用，与一般的应用程序完全不同。其区别主要有以下几个方面：(1)驱动程序被编译为目标文件，应用程序被编译连接为可执行程序；(2)应用程序由 main 过程和子过程组成，驱动程序没有 main 过程，其过程是系统核心的子过程；(3)应用程序一般“主动”完成某任务后就结束本次执行，而驱动程序随系统的启动而启动，并在被应用进程调用后起作用；(4)系统对应用程序的时间性没有要求，而对于驱动程序要求尽量少占用 CPU 时间，并避免死循环和长时间的等待操作；(5)应用程序的每个进程在系统内都有一个内存映像副本，而驱动程序在系统内只有一个拷贝，并且可能要同时接受多个进程的请求，因此要考虑“重入”问题。

2. 硬件设备的分类及设备文件

硬件设备大体上分为块设备和字符设备两类。在块设备和操作系统之间有输入输出缓冲区，数据传输以一定大小的数据块进行，如硬盘可能以每次 512 字节的数据块进行读写。对于块设备的操作，系统参与较多，用户程序一般只是对系统缓冲区进行读写操作，实际的操作由系统在缓冲区和设备之间进行。在字符设备和操作系统之间可能有也可能没有输入输出缓冲区，数据传输以不定数量的字节流进行，如串口每次只能读或写 1 字节的数据。对于字符设备的操作，系统参与较少，一般是用户程序提供数据传输参数，如用户数据区的地址，传输的数据个数等，然后由驱动程序完成用户数据区和硬件设备间数据传输。

UNIX 系统下，在硬盘或软盘等介质上保存的数据文件称为普通文件；在/dev 目录下存在的、代表一个设备的特殊文件称为设备文件。对用户来说，设备文件并非真正的文件，只是访问相应设备的入口点。每个设备文件都有一个系统内独一无二的设备号来标识相应的设备。设备号由主设备号和次设备号两部分组成，核心利用主设备号区分不同的驱动程序，利用次设备号传给驱动程序一些额外信息。比如系统内的两个串行口具有相同的主设备号，串口 1 和串口 2 靠次设备号 0 和 1 区分。设备文件看起来与普通文件有些相象，在/dev 目录下运行 ls -l 命令，与串口有关的输出可能如下所示：

```
crw-rw-rw- 1 bin bin 5, 0 Aug 7 18:20 /dev/ttyla
crw-rw-rw- 1 bin bin 5, 1 Aug 10 17:33 /dev/ttylb
```

其中 c 表示该设备为字符设备(b 表示块设备)，5 是主设备号，0 和 1 分别是串口 1 和串口 2 的次设备号，ttyla 和 ttylb 分别是串口 1 和串口 2 的设备文件名。若几个设备文件的主设备号相同，则对应的设备由同一个驱动程序管理。对设备的访问要通过设备文件进行，访问方式与普通文件基本相同，首先要用 open 调用打开设备文件，然后可用 read 和 write 调用读写数据，最后要用 close 调用关闭设备文件。设备文件的特殊性主要在于对设备的控制，如对串口操作时要调用 ioctl 设定通信波特率，数据位数和停止位等参数。

3. 进程运行模式及 u－area

进程运行时有两种模式，若执行用户程序中的指令，称为用户模式；若执行核心指令则称为核心模式。用户进程一般在用户模式下运行，不能访问硬件设备，当执行系统调用时便转换为核心模式，可以访问硬件设备。系统中断也可以引起用户模式和核心模式的切换。每个进程都有一个特殊的内存区域，称为 u－area，该区域包含系统核心管理该进程所需要的有关信息和一个核心模式堆栈，u－area 不在用户数据空间，用户进程无法访问。当进程执行系统调用时，其寄存器值被保存到核心堆栈，堆栈指针指向核心堆栈，当系统调用完成后，寄存器值被恢复，堆栈指针指向进程堆栈。每个进程都有一个固定大小的核心堆栈，在 UNIX SYSTEM V 中，核心堆栈为 4096 个字节。当中断发生时，中断进程使用当前进程的核心堆栈，若当前进程处于用户模式，则中断进程使用整个核心堆栈，若当前进程处于核心模式，则中断进程与当前进程共用核心堆栈。因此，中断进程要尽可能少的占用核心堆栈，以免堆栈溢出。进程执行系统调用时的参数放到 u－area 中，进入核心模式后，驱动程序从 u－area 中取得这些参数。u－area 在系统中定义为一个结构，其实例名为 u，是系统提供的一个外部变量，驱动程序常用的几个参数如下：

(1)u.u-base 读写操作时用户数据区的首地址；

(2)u.u-count 读写数据的字节数；

(3)u.u-ofile 设备文件描述符；

(4)u.u-offset 设备文件内数据传输指针；

(5)u.u-error 错误标志。除了上述参数，核心还向驱动程序提供设备的主设备号和次设备号。

从上面的讨论可知，用户访问硬件设备的入口点是设备文件，对硬件的访问通过系统调用进行，调用参数通过 u－area 传给驱动程序，那么剩下的就是驱动程序与硬件设备通信，完成数据的传输工作。

4.驱动程序组成

设备驱动程序由核心过程和设备驱动程序过程组成。核心过程由系统提供，是操作系统的固有代码，驱动程序调用这些过程完成对硬件设备的读写和其他操作，如从口地址读数据和向口地址写数据等。驱动程序过程是由驱动程序编写者提供的过程，包括一般过程和响应硬件中断的中断过程。这些过程的编写要遵循一定的规则，一方面驱动程序作为系统核心的一部分，要尽可能的精练、高效，并且不能与系统冲突；另一方面要屏蔽掉硬件设备的具体细节，提供统一的软件接口。用户过程的名字由两部分组成，一部分是二到四个字节的前缀，一部分是标准的后缀。如读过程名 xxread，其中 xx 就是前缀，read 是标准的后缀。下面列出字符设备驱动程序的几个用户过程：

xxopen 开始对字符设备访问，与打开文件的 open 调用对应

xxclose 结束对字符设备访问，与关闭文件的 close 调用对应

xxread 把内部缓冲区数据传到用户数据区，与 read 调用对应

xxwrite 把用户数据传到内部缓冲区，与 write 调用对应

xxioctl 执行 I/O 控制命令，与 ioctl 调用对应

xxinit 系统启动时执行对设备的初始化

xxintr 中断响应过程

核心过程由系统内核提供，下面列出对 I/O 口操作的几个过程：

inb 读一个字节

inw 读一个字(16bits)

outb 写一个字节

outw 写一个字(16bits)

核心过程很多，包括内存分配，DMA 操作，数据拷贝等，可参见有关手册。

5.驱动程序开发

开发设备驱动程序需要有 UNIX 操作系统，UNIX 开发系统(提供 C 或汇编语言编译器)，驱动程序开发指南和核心过程的参考手册。下面以一假想的串口为例，讨论具体的开发步骤。

(1)选择前缀。前缀为二到四个字符，不能与已有的前缀相同，一般应有特定含义，如串口可为 seri。

(2)编写程序。系统规定需要编写的用户过程，但并非每个用户过程都是必需的，可根据具体需要确定。如只需从串口读数据，就可以不编写 seriwrite 过程。有些过程可能非常简单，没有实质性的内容，如关闭串口的 sericlose 不需执行任何操作。

(3)编译源程序。生成目标文件，用 C 语言编译程序(cc)时，需下列命令选项："－c －K －Zp4 －DINKERNEL"，编译完成后要把目标文件改名为 Driver.o，并拷贝到"/etc/conf/pack.d/设备名"目录下，此处"设备名"即是 seri。

(4)生成设备文件。用 mknod 命令在/dev 目录下生成设备文件，利用该文件就可对设备进行访问，设备文件和驱动程序通过设备号相关联。设备文件的命名与一般文件相同，也可以采用过程名的前缀，此处为 seri。

(5)系统配置。系统提供 configure 命令，利用该命令可以取得未用的设备号，并可确定设备的中断是否与其他设备的中断冲突。在列出设备的中断号、设备号、口地址范围、用户过程等信息后，就可以用 configure 命令把设备配置到系统中。

(6)连接核心。即把驱动程序的目标文件(Driver.o)和系统连接成为一个新的核心。

运行/etc/conf/cf.d 目录下的 link-unix 命令就可生成新的系统核心。

完成上述步骤后，重新启动系统，就可以利用新加入的驱动程序接口访问相应的硬件设备。一般情况下一个非常简单的设备驱动程序调试起来也很麻烦，如果有问题，可以修改驱动程序，重新编译并用 link-unix 连接，生成设备文件和系统配置等步骤一般不需要重新进行。

以上是手工操作步骤，如果是用 shell 程序写出驱动程序安装脚本(厂家的驱动程序一般都提供安装脚本)，则可自动安装。如果购买了驱动程序开发工具，对于驱动程序的开发来讲就会更加方便。

参考文献

[1] 陈华瑛等，UNIX 操作系统设计与实现，电子工业出版社，1992 年 3 月。

[2] 樊建平等，UNIX SYSTEM V 操作系统内核代码剖析，海洋出版社，1992 年 5 月。

(来稿时间：1998 年 6 月)