基于嵌入式 Linux 系统键盘驱动的原理及实现

Theory of keyboard driver and its implementation based on embedded Linux system

郑志国　（中国电子科技集团公司第十研究所　成都 610036）
赵万春　（总装重庆军代局驻成都地区军代室　成都 610036）

摘　要：本文基于 Linux 内核中键盘驱动程序的整体框架结构，介绍 Linux 键盘驱动的原理，并以 MPC5200 开发平台为例具体介绍嵌入式 Linux 系统中键盘驱动的实现。

关键词：Linux　嵌入式　键盘驱动

1　引言

由于 Linux 具有内核功能强大、稳定，易于扩展和裁减，丰富的硬件支持等优点，在嵌入式系统中得到了广泛的应用。目前很多基于嵌入式 Linux 系统的应用都须进行人机交互，这就需要配备一个用户自定义的键盘，同时需为自定义的键盘编写驱动程序。

2　linux 键盘驱动的原理

键盘是进行电脑操作时最常用的标准输入设备，其驱动原理如图 1 所示。击键时，键盘会发送相应的扫描码（scancodes）给键盘驱动；键盘驱动中的 handle_scancode( ) 函数解析 scancodes 流并通过 kdb_translate( ) 函数里的转换表（translation – table）将击键事件（key press events）和键释放事件（key release events）转换成连续的 keycode；比如，“a”的 keycode 是 30，击“a”键时产生 keycode 30，释放“a”键时产生 keycode 158（128 +30）。

keycode 通过对 keymap 的查询被转换成相应 key 符号；key 符号被送入 raw tty 队列 tty_flip_buffer，receive_buf( ) 函数周期性的从 tty_flip_buffer 中获得 key 符号，然后把这些 key 符号送入 tty read 队列；

当用户进程需要得到输入时，它会在进程的标准输入（stdin）调用 read( ) 函数，sys_read( ) 函数调用定义在相应的 tty 设备（如/dev/tty0）的 file_operations 结构中指向 tty_read 的 read( ) 函数来读取字符并返回给用户进程。

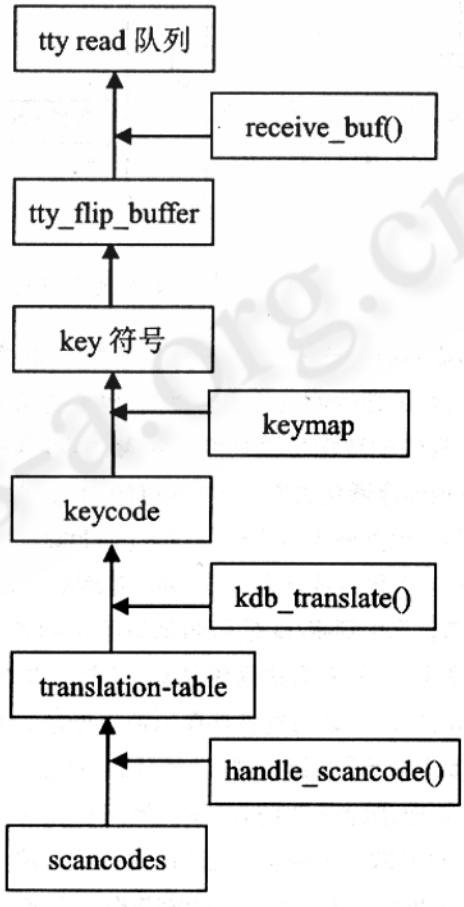

图 1　Linux 键盘驱动原理示意图

3　linux 驱动程序简介

系统调用是操作系统内核和应用程序之间的接口，设备驱动程序是操作系统内核和硬件之间的接口。设备驱动程序为应用程序屏蔽了硬件的细节，在应用程序看来，硬件设备只是一个设备文件，应用程序可以象操作普通文件一样对硬件设备进行操作。设备驱动程序是内核的一部分，它具有对设备初始化和释放、传输、读取数据等功能。

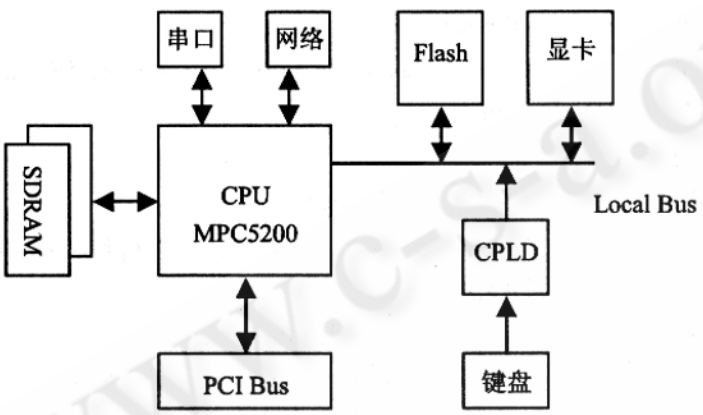

图 2　MPC5200 开发平台框图

在 Linux 操作系统中有两类主要的设备文件类型：字符设备和块设备。字符设备主要针对操作进行优化，块设备主要针对管理进行优化。

用户进程通过设备文件与硬件进行通信。每个设备文件都有其文件属性(c/b)，表示其属字符设备或块设备。每个文件有两个设备号，第一个是主设备号，标识驱动程序；第二个是从设备号，标识同一个设备驱动程序的不同的硬件设备，比如有两个软盘，就可以用从设备号来区分他们。

4　linux 键盘驱动的实现

4.1　硬件实现

在本设计中，采用 Motorola 公司的 MPC5200 CPU 构建一个嵌入式系统，采用 Intel 8042 芯片设计标准的鼠标键盘控制器，并使用 CPLD 扩展一个自定义小键盘，系统硬件框图如图 2 所示。

CPLD 连接到 CPU 本地总线上，自定义键盘有十个键，分别对应十条 CPLD I/O 管脚，使用电阻拉高，按下其中任一个键时，该 I/O 管脚输入为低。另外 CPU 的 LP 总线接入 CPLD，只需要 8 位数据线和 1 位地址线以及少量控制信号线，1 位地址线定义两个地址，分别用来读取键盘 ID 值和键盘输入的键值。在 CPLD 中使用 case 语句对输入值进行译码，然后通过按键产生的中断通知 CPU 来读取键值。

4.2　linux 键盘驱动程序的编写

在 Linux 系统中，键盘驱动程序采用层次型体系结构，分两层来实现。其中，上层属通用的键盘抽象层，完成键盘驱动中不依赖于底层具体硬件实现的一些功能，同时负责为底层提供服务；下层为硬件处理层，负责对硬件进行直接操作。

在这种体系结构下，开发者只需为其编写驱动程序中的底层硬件处理函数，就可以将该键盘驱动起来。一般说来，编写底层硬件处理函数最重要的工作就是在键盘中断处理中获取按键的扫描码，并以它为参数调用 handle_scancode( )，该扫描码可以自己定义，但它必须唯一地标识出按键在键盘上的位置。此外，开发者还需要提供对应的从自定义扫描码到键码的转换函数 kbd_translate。

以下是 Linux 键盘驱动程序编写的具体实现。

(1) 定义部分。定义部分除了定义键盘 ID 地址、键盘码值地址以及中断号等常量外，最主要的是定义键盘设备的数据结构和初始化 tasklet 队列：

```
...
#define EKB_DATA_ADDR 0xAD000010
#define EKB_ID_ADDR 0xAD00001F
......
/* for IRQ */
#define EKB_IRQ 0x7
......
typedef struct EKB_DEV_t
{
  unsigned char *data_base_addr;
  unsigned char *id_base_addr;
  unsigned char data;
```

```
    devfs_handle_t handle;
    int state;/ * wake when state =1 * /
  } EKB_DEV;
  ......
  / * 定义 tasklet * /
  DECLARE_TASKLET( ekb_read_tasklet, ekb_do_
tasklet, (unsigned long)0);
```

(2) 中断部分。Linux 将中断处理分为两个部分:顶半部(top half)为实际响应中断的程序,即 request_irq 注册的中断程序;底半部(bottom half)是一个被顶半部调度,并在稍后更安全的时间内执行的程序。典型的情况:顶半部保存设备的数据到一个设备特定的缓冲区并调度它的底半部,然后退出,这些处理非常快;然后,底半部执行其他必要的工作,这种方式使得底半部工作期间,顶半部还可以继续为新的中断服务。

在键盘驱动程序中,顶半部只是简单的调度了 tasklet 队列:

```
  void ekb_irq_handler(int irq, void * dev_id, struct
pt_regs * regs)
  {
    tasklet_schedule(&ekb_read_tasklet);
  }
```

而底半部完成读取键盘码值,并执行键盘驱动与系统接口的函数:

```
  void ekb_do_tasklet(unsigned long unused)
  {
    / * 读取键盘键值,在地址偏移 0x10 处,值为
0x1 ~ 0xA * /
    dev_ekb.data = * (dev_ekb.data_base_ad-
dr);
    keyb_event(dev_ekb.data, 1/ * down * /);
  }
```

(3) 驱动程序接口。由于键盘驱动不需要为应用程序提供接口,因此不需要定义 read()、write()等接口函数的具体实现,只需在中断服务底半部读取到键盘码值后,直接调用函数 handle_scancode()将码值转换为 scancodes,然后调度操作系统定义的键盘 tasklet 队列,系统就能够响应键盘操作了。

```
  static void keyb_event(unsigned int code, int
down)
  {
    if (emulate_raw(code, down)) / * 该函数调
用 handle_scancode() 对码值进行处理。 * /
      tasklet_schedule(&keyboard_tasklet);
  }
```

(4) 驱动程序的装载与卸载。模块的装载主要是向内核注册设备驱动,并完成驱动程序必要的初始化,以及注册中断等。

```
  int __init ekb_init_module(void)
  {
    int result;
    result = register_chrdev(ekb_major, DEVICE_
NAME, &ekb_ops);
    ......
    / * register the devfs * /
    dev_ekb.handle = devfs_register(NULL/ * dir
* /, DEVFS_NAME,
      DEVFS_FL_AUTO_DEVNUM,
      0, 0, S_IFCHR | S_IRUGO | S_IWUGO,
      &ekb_ops, &dev_ekb);
    ekb_init();
    return 0;
  }
```

驱动程序的卸载刚好相反,它将驱动程序所占用的系统资源、中断号进行释放,这样内核又可以利用这些资源完成其它任务。(代码略)

5 结束语

开发者以 MPC5200 开发板平台为例,重点描述了在嵌入式 Linux 系统下,为特殊键盘编写驱动程序时需要完成的工作,为类似的开发提供了一种思路和参考。

参考文献

1 魏永明等,Linux 设备驱动程序 第二版,2002.
2 Phrack Inc,如何编写一个获取所有敲键盘的模块.
3 Linux Kernel Module Programming.
4 Andries Brouwer,The Linux keyboard driver.
5 Roy G,如何编写 Linux 的设备驱动程序,2002.