Linux 0.01 键盘驱动程序架构分析与解读

Weball 于 12。14

这两天看了一些键盘的程序，所以把我的一些心得和体会写下来，希望对大家有一个参考作用，Linux0。01 内核的键盘驱动程序对应文件为 kernel 目录下的 keyboard.s，虽然它是一个汇编文件，但是看懂还是比较容易的，因为它主要实现的还是一些键盘的基本功能和原理，这里只对其大致的结构和重点的方面进行分析。

Intel 体系的键盘基本知识

在阅读具体的代码之前，我想先介绍一下 Intel 体系的键盘的一些基本知识，我们都知道，在 intel 体系结构中，键盘是和 8259 的 ir1 脚相连的，所以键盘的一切操作都是基于中断来处理的，所以在键盘的程序中是通过 keyboard_interrupt 这个中断服务例程来统领全部函数的，明白了这点，对键盘的结构就有了一个框架性的认识。

在我们按每一个键的是否会发生两次中断，一个是按下的时候产生的一个中断，另一个是键释放的时候产生的，我们一般只需要处理按下的时候的中断，而对释放的时候系统的中断不理会，但是在某些时候，我们也可以对释放进行处理。一般一个键盘内部有一个控制芯片处理键盘的所有键，我们按下键时，键盘产生一个中断，然后主机就从键盘中读取产生的信息并保存在内部的 ram 中，所以我们的中断处理程序的主要功能就是通过从端口 60 读取 ram 的数据信息和从端口 64 读取键盘的状态信息，然后根据读取的扫描码进行各个功能特殊键的判断（比如判断是否 ctrl、alt 和 shift 的按下，是否为 f1 到 f12 这些功能键，以及是否是 ctrl+alt+del 系统重启热键等）来对各种情况进行处理，然后把对应的扫描码转换为 ASCII 码，最后放入键盘队列中，所以实际上我们读取输入信息实际就是从键盘的输入队列中读取数据，如果键盘队列处于空状态，我们就会一直等待直到有数据输入，所以键盘函数提供对其他部分有用的结构就是一个键盘的输入队列。

Linux0.01 中 keyboard.s 文件的流程分析

根据上面分析的，键盘程序的主要入口就是_keyboard_interrupt 这个中断处理函数，首先保存各个寄存器的值，然后从 60 端口读入键盘的扫描码，同时对 61 这个控制寄存器端口进行一些处理保证读取成功，然后判断是否按下的是特殊键，因为如果按下特殊键，需要进行其他处理，然后根据得到的扫描码来在 key_table（各个键对应的处理函数表）中查找各个键的处理函数并进行调用，最后调用完毕之后复位中断控制器 8259。

对所有键的处理过程都在其对应的处理函数之中，总的来说，主要是一个 do_self 处理函数，它是普通键的处理函数，它根据是否按下各种特殊键的状态来在几个映射表中进行查找（key_map、alt_map、shift_map）得到对应的 ascii 码并最后放入键盘队列中，还有一些其他键的处理函数，但是都比较简单和直观。下面只对其中的函数做一个简单的注解：

mode：表明是否按下 alt、ctrl 和 shift 的模式

leds：记录几个控制键盘灯的参数

e0：表明是否为特殊键按下

keyboard_interrupt：键盘中断函数

put_quene：将 ascii 码放入键盘队列函数
ctrl、alt、unctrl、unalt、unshift、lshift、rshift 等函数判断是否为 alt、shift 和 ctrl 按下
caps、scroll、num 这些控制键盘灯开关的键中需要处理 set_leds 等函数
cursor 处理小键盘的键，同时对特殊键进行处理
func 对功能键进行处理
key_map、alt_map 和 shift_map 定义了三个映射表
do_self 对普通键处理的函数，通过查表转换到 ascii 码
minus 是 linus 自身的处理函数
key_table 包含了各个键的映射函数
kb_wait 等待键盘取走数据函数
reboot 是当按下 alt+ctrl+del 的热键时处理函数