中断。在发生每个这样的键盘中断时，键盘驱动程序都要从与键盘相关联的I/O端口提取信息，以了解发生了什么事情。其他的一切事情都是在软件中发生的，在相当大的程度上独立于硬件。

当想象往shell窗口（命令行界面）键入命令时，可以更好地理解本小节余下的大部分内容。这是程序员通常的工作方式。我们将在下面讨论图形界面。

1. 键盘软件

I/O端口中的数字是键编号，称为*扫描码*（scan code），而不是ASCII码。键盘所拥有的键不超过128个，所以只需7个位表示键编号。当键按下时，第8位设置为0，当键释放时，第8位设置为1。跟踪每个键的状态（按下或弹起）是驱动程序的任务。

例如，当A键被按下时，扫描码（30）被写入一个I/O寄存器。驱动程序应该负责确定键入的是小写字母、大写字母、CTRL-A、ALT-A、CTRL-ALT-A还是某些其他组合。由于驱动程序可以断定哪些键已经按下但是还没有被释放（例如SHIFT），所以它拥有足够多的信息来做这一工作。

例如，击键序列

按下SHIFT，按下A，释放A，释放SHIFT

指示的是大写字母A。然而击键序列

按下SHIFT，按下A，释放SHIFT，释放A

指示的也是大写字母A。尽管该键盘接口将所有的负担都加在软件上，但是却极其灵活。例如，用户程序可能对刚刚键入的一个数字是来自顶端的一排键还是来自边上的数字键盘感兴趣。原则上，驱动程序能够提供这一信息。

键盘驱动程序可以采纳两种可能的处理方法。在第一种处理方法中，驱动程序的工作只是接收输入并且不加修改地向上层传送。这样，从键盘读数据的程序得到的是ASCII码的原始序列。（向用户程序提供扫描码过于原始，并且高度地依赖于机器。）

这种处理方法非常适合于像emacs那样的复杂屏幕编辑器的需要，它允许用户对任意字符或字符序列施加任意的动作。然而，这意味着如果用户键入的是dste而不是date，为了修改错误而键入三个退格键和ate，然后是一个回车键，那么提供给用户程序的是键入的全部11个ASCII码，如下所示：

dste← ← ←ate CR

并非所有的程序都想要这么多的细节，它们常常只想要校正后的输入，而不是如何产生它的准确的序列。这一认识导致了第二种处理方法：键盘驱动程序处理全部行内编辑，并且只将校正后的行传送给用户程序。第一种处理方法是面向字符的，第二种处理方法是面向行的。最初它们分别被称为*原始模式*（raw mode）和*加工模式*（cooked mode）。POSIX标准使用稍欠生动的术语*规范模式*（canonical mode）来描述面向行的模式。*非规范模式*（noncanonical mode）与原始模式是等价的，尽管终端行为的许多细节可能被修改了。POSIX兼容的系统提供了若干库函数，支持选择这两种模式中的一种并且修改许多参数。

如果键盘处于规范（加工）模式，则字符必须存储起来直到积累完整的一行，因为用户随后可能决定删除一行中的一部分。即使键盘处于原始模式，程序也可能尚未请求输入，所以字符也必须缓冲起来以便允许用户提前键入。可以使用专用的缓冲区，或者缓冲区也可以从池中分配。前者对提前键入提出了固定的限制，后者则没有。当用户在shell窗口（Windows的命令行窗口）中击键并且刚刚发出一条尚未完成的命令（例如编译）时，将引起尖锐的问题。后继键入的字符必须被缓冲，因为shell还没有准备好读它们。那些不允许用户提前键入的系统设计者应该被涂柏油、粘羽毛[⊖]，或者更加严重的惩罚是，强迫他们使用他们自己设计的系统。

虽然键盘与监视器在逻辑上是两个独立的设备，但是很多用户已经习惯于看到他们刚刚键入的字符出现在屏幕上。这个过程叫做*回显*（echoing）。

当用户正在击键的时候程序可能正在写屏幕，这一事实使回显变得错综复杂（请再一次想象在shell

⊖ 原文为be tarred and feathered，是英国古代的一种酷刑。受刑人全身涂上灼热的柏油（tarred），然后将其身上粘满羽毛（feathered）。这样，羽毛当然很难脱下，要脱下也难免皮肉之伤。be tarred and feathered现用于比喻受到严厉惩罚。——译者注