结合阻塞与非阻塞访问、poll函数可以较好地解决设备的读写，但是如果有了异步通知就更方便了。异步通知的意思是：一旦设备就绪，则主动通知应用程序，这样应用程序根本就不需要查询设备状态，这一点非常类似于硬件上“中断”地概念，比较准确的称谓是“信 号驱动(SIGIO)的异步I/O”。

我们先来看一个使用信号驱动的例子，它通过signal(SIGIO, input\_handler)对STDIN\_FILENO启动信号机制，输入可获得input\_handler被调用，其源代码如下：

#include <sys/types.h>

#include <sys/stat.h>

#include <stdio.h>

#include <fcntl.h>

#include <signal.h>

#include <unistd.h>

#define MAX\_LEN 100

void input\_handler(int num)

{

char data[MAX\_LEN];

int len;

//读取并输出STDIN\_FILENO上的输入

len = read(STDIN\_FILENO, &data, MAX\_LEN);

data[len] = 0;

printf("input available:%s\n", data);

}

main()

{

int oflags;

//启动信号驱动机制

signal(SIGIO, input\_handler);

fcntl(STDIN\_FILENO, F\_SETOWN, getpid());

oflags = fcntl(STDIN\_FILENO, F\_GETFL);

fcntl(STDIN\_FILENO, F\_SETFL, oflags | FASYNC);

//最后进入一个死循环，程序什么都不干了，只有信号能激发 input\_handler的运行

//如果程序中没有这个死循环，会立即执行完毕

while (1);

}

为了使设备支持该机制，我们需要在驱动程序中实现fasync()函数，并在write()函数中 当数据被写入时，调用kill\_fasync()函数激发一个信号，此部分工作留给读者来完成。