字符显示的控制

难度系数：★★☆☆☆

实验目的

- 加深对操作系统设备管理基本原理的认识，实践键盘中断、扫描码等概念；
- 通过实践掌握Linux 0.11对键盘终端和显示器终端的处理过程。

实验内容

本实验的基本内容是修改Linux 0.11的终端设备处理代码，对键盘输入和字符显示进行非常规的控制。

在初始状态，一切如常。用户按一次F12后，把应用程序向终端输出所有字母都替换为“*”。用户再按一次F12，又恢复正常。第三次按F12，再进行输出替换。依此类推。

以ls命令为例：

正常情况：

```
# ls
hello.c hello.o hello
```

第一次按F12，然后输入ls：

```
# **
*****.* *****.* *****
```

第二次按F12，然后输入ls：

```
# ls
hello.c hello.o hello
```

第三次按F12，然后输入ls：

```
# **
*****.* *****.* *****
```

实验报告

完成实验后，在实验报告中回答如下问题：

1. 在原始代码中，按下F12，中断响应后，中断服务程序会调用func？它实现的是什么功能？
2. 在你的实现中，是否把向文件输出的字符也过滤了？如果是，那么怎么能只过滤向终端输出的字符？如果不是，那么怎么能把向文件输出的字符也一并进行过滤？

评分标准

- F12切换，40%
- 输出字符隐藏，40%
- 实验报告，20%

实验提示

键盘输入处理过程

键盘I/O是典型的中断驱动，在kernel/chr_drv/console.c文件中：

```
void con_init(void)  //控制台的初始化
{
    set_trap_gate(0x21,&keyboard_interrupt);  //键盘中断响应函数设为keyboard_interrupt
}
```

所以每次按键有动作，keyboard_interrupt函数就会被调用，它在文件kernel/chr_drv/keyboard.S（注意，扩展名是大写的S）中实现。所有与键盘输入相关的功能都是在此文件中实现的，所以本实验的部分功能也可以在此文件中实现。详读《注释》一书中对此文件的注解会大有裨益。

简单说，keyboard_interrupt被调用后，会将键盘扫描码做为下标，调用数组key_table保存的与该按键对应的响应函数。

输出字符的控制

printf()等输出函数最终都是调用write()系统调用，所以控制好write()，就能控制好输出字符。