终端设备的控制

一、实验目的

1、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；

2、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。

3、能够通过自主学习，解决在内核编译运行中的问题。

二、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：

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

三、实验提示

可参考 Bilibili 上的 操作系统哈尔滨工业大学李治军老师的 I/O 与显示器(L26)和键盘(L27)部分视频。

本实验需要修改 Linux 0.11 的终端设备处理代码（`kernel/chr_drv/console.c` 文件），对键盘输入和字符显示进行非常规的控制。

1、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{
    // 键盘中断响应函数设为 keyboard_interrupt
    set_trap_gate(0x21, &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。

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。

2、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。

四、实验思考

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？