北京郵電大學

实验报告

题目：键盘驱动程序的分析与修改

班　　级：2020211301

学　　号：2020211221

姓　　名：翁岳川

学　　院：计算机学院(国家示范性软件学院)

2021 年 12 月 11 日

一、实验目的

1、理解 I/O 系统调用函数和 C 标准 I/O 函数的概念和区别；
2、建立内核空间 I/O 软件层次结构概念，即与设备无关的操作系统软件、设备驱动程序和中断服务程序；
3、了解 Linux-0.11 字符设备驱动程序及功能，初步理解控制台终端程序的工作原理；
4、通过阅读源代码，进一步提高 C 语言和汇编程序的编程技巧以及源代码分析能力；
5、锻炼和提高对复杂工程问题进行分析的能力，并根据需求进行设计和实现的能力。

报告邮寄（最迟时间：2021 年 12 月 22 日晚 23：59）：

大一班（1-4 班）：clavicle@bupt.edu.cn

二、实验环境

1、硬件：学生个人电脑（x86-64）
2、软件：Windows 10，VMware Workstation 15 Player，32 位 Linux-Ubuntu 16.04.1
3、gcc-3.4 编译环境
4、GDB 调试工具

三、实验内容

解压 lab4.tar.gz 文件，解压后进入 lab4 目录得到如下文件和目录：
****
安装 gcc 编译器：
****
实验常用执行命令如下：

执行./run ，可启动 bochs 模拟器，进而加载执行 Linux-0.11 目录下的 Image 文件启动 linux-0.11 操作系统

进入 lab4/linux-0.11 目录，执行 make 编译生成 Image 文件，每次重新编译(make)前需先执行 make clean

如果对 linux-0.11 目录下的某些源文件进行了修改，执行./run init 可把修改文件回复初始状态

本实验包含 2 关，要求如下：

Phase 1

键入 F12，激活*功能，键入学生本人的姓名拼音，首尾字母等显示*

比如：zhangsan，显示为：*ha*gsa*

Phase 2

键入“学生本人的学号”：激活*功能，键入学生本人的姓名拼音，首尾字母等显示*

比如：zhangsan，显示为：*ha*gsa*，

键入“学生本人的学号-”：取消显示*功能

提示：完成本实验需要对 lab4/linux-0.11/kernel/chr_drv/目录下的 keyboard.s、console.c 和 tty_io.c 源文件进行分析，理解按下按键到回显到显示频上程序的执行过程，然后对涉及到的数据结构进行分析，完成对前两个源程序的修改。修改方案有两种：

在 C 语言源程序层面进行修改

在汇编语言源程序层面进行修改

其他说明见 实验四.ppt 。linux 内核完全注释(高清版).pdf 一书中对源代码有详细的说明和注释。

四、实验步骤及实验分析

建议按照：准备工作、阶段 1、阶段 2、...等来组织内容

各阶段需要有操作步骤、运行截图、分析过程的内容

1. 准备工作

搭建虚拟机并在虚拟机上安装 Ubuntu16.04.1，在虚拟机中安装 gcc3.4.6，将文件 lab4.tar.gz 利用解压到文件夹 lab4，截图如下


```
wyc2002@ubuntu:~/Desktop$ ls
lab4  lab4.tar.gz
wyc2002@ubuntu:~/Desktop$ cd lab4
wyc2002@ubuntu:~/Desktop/lab4$ ls
a.out         dbg-asm  gdb          hdc-0.11.img       mount-hdc
bochs         dbg-c    gdb-cmd.txt  linux-0.11         run
bochsout.txt  files    hdc          linux-0.11.tar.gz  rungdb
```

将 linux-0.11.tar.gz 解压到 linux-0.11 文件夹下，此时有如下

```
wyc2002@ubuntu:~/Desktop/lab4$ tar -xvf linux-0.11.tar.gz -C linux-0.11
```

```
wyc2002@ubuntu:~/Desktop/lab4$ cd linux-0.11
wyc2002@ubuntu:~/Desktop/lab4/linux-0.11$ ls
boot  Image    init    lib       mm          tags
fs    include  kernel  Makefile  System.map  tools
```

进入 kernel 文件夹下的 chr_drv 文件夹，并对其中的内容进行查看

```
wyc2002@ubuntu:~/Desktop/lab4/linux-0.11/kernel$ cd chr_drv
wyc2002@ubuntu:~/Desktop/lab4/linux-0.11/kernel/chr_drv$ ls
chr_drv.a  keyboard.o  Makefile  serial.c  tty_ioctl.c
console.c  keyboard.s  rs_io.o   serial.o  tty_ioctl.o
console.o  keyboard.S  rs_io.s   tty_io.c  tty_io.o
```

2.解决 phase_1

keyboard.S 键盘中断驱动程序，而 console.c 为控制台显示驱动程序，则此时对这两者进行更改，则思路明确，在 keyboard.S 中修改 f12 所调用的函数，然后在 console.c 中写入函数，并利用 state 变量对字符串进行合理操作，则有如下步骤

①由于 f12 调用的时 keyboard.S 中的 func 函数，则对此位置进行更改，让其调用我们所需要的函数，此处命名为 change_f12flag，有如下截图

```
func:
        pushl %eax
        pushl %ecx
        pushl %edx
        call show_stat
        popl %edx
        popl %ecx
        popl %eax
        subb $0x38,%al
        jb end_func
        cmpb $9,%al
        jbe ok_func
        subb $18,%al
        cmpb $10,%al
        jb end_func
        cmpb $11,%al
        ja end_func
        call change_f12flag
```

此时在函数的最后一个语句处调用了我们需要的函数，此时对 keyboard.S 的修改完成，接下来修改 console.c 文件

②上一步修改到调用函数 change_f12flag，则我们此时在 console.c 中需要设置一个全局变量作为 f12 的开启和关闭的状态变量，这里选取 f12flag，当 f12flag=0 时功能关闭，当 f12flag=1 时功能开启，同时要写出函数 change_f12flag，修改其 f12flag 的状态 则有如下截图

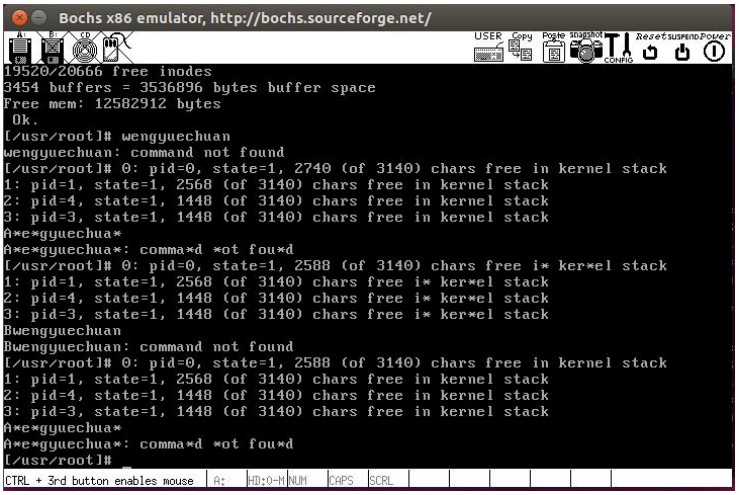
```
#define NPAR 16
int f12flag=0;

void change_f12flag(void){
        switch(f12flag){
        case 0: f12flag=1; break;
        case 1: f12flag=0; break;
}

}
```

③而在 console.c 中有 con_write 部分对字符串进行操作，而此时要控制输入的为字母，则对应的 ascii 值为(c>64 && c<91)||(c>96 && c<123)，则在 con_write 适当位置插入判断语句，修改字符串的值，而我的姓名全拼为 wengyuechuan，则将'w'和'n'转换成'*'，则有如下插入现象，截图如下

```
    }
    if(f12flag==1 && ((c>64 && c<91)||(c>96 && c<123)) && (c=='w' || c=='n'))
            c='*';
    __asm__("movb attr,%%ah\n\t"
            "movw %%ax,%1\n\t"
            ::"a" (c),"m" (*(short *)pos)
            );
    pos += 2;
    x++;
```

④保存，然后在 linux-0.11 目录下利用 make clean 和 make 语句对内核进行重新编译，再利用./run 进行尝试，按照输入顺序如下输入

wengyuechuan f12 wengyuechuan f12 wengyuechuan f12，有如下结果



则证明 f12 作为操作的开关，可以完成所需要的操作，成功解决。

2.解决 phase_2

总体思路与第一个相同，不同的在于此时不需要改变按键本身的作用，只需要在 console.c 中改变 con_write 中判断条件，而没有必要对 keyboard.S 进行操作，则有如下步骤

①设置变量 total 用于计数，表示输入的字符串的长度，同时利用 flag 表示'*'操作开启与否，0 代表未开启，1 代表开启，有截图如下

```
#define NPAR 16                         int nr;
int flag=0;                             char c;
                                        int total=1;
extern void keyboard_interrupt(void);

static unsigned char    video type;     nr = CHARS(tty->write_q);
```

②我的学号为 2020211221，则第一次输入 2020211221 时，开启'*'功能，再次输入"2020211221-"则关闭功能，则有几个需要注意的地方：

1°需要保证连续输入，若有输入如 2020xxxx211221 输入需要排除。

2°当前面有干扰时计数器不进行计数，只有读取到 2020211221 时才算作一次，则例如输入 xxxx2020211221 时代表开启'*'功能，则此时可利用计数器进行判断，若出现 xxxx22020211221 时，从第二个 2 开始计数和

操作。
3°当出现序列 202020211221 时此时仍然是可以进行，则在 total==6 时，同时此时 c 为 0 时，将 total 修改为 5
则程序如下：

```
}
if(c=='2' &&(total==1||total==3||total==5||total==8||total==9))
        total++; /*2020211221*/
else if(c=='0' &&(total==2||total==4))
        total++;
else if (c=='0' && total==6){
        total=5;
}
else if(c=='1'&&(total==6||total==7))
        total++;
else if(c=='1'&& total==10){
        total++;
        flag=1;
        }
else if(c=='-' && total==11){
        total++;
        flag=0;
}
else {
        total=1;
        if(c=='2'){
                total=2;
        }
}
if(flag==1){
        if(c=='w' || c=='n')
                c='*';
        }
```

③回到 lab4 目录下，利用./run 进行尝试，输入顺序如下
2020211221
wengyuechuan
2020211221-
wengyuechuan
xxxx2020211221
wengyuechuan
2020211221-
wengyuechuan
2222020211221
wengyuechuan
2020211221-
wengyuechuan
则有如下结果

```
[/usr/root]# 2020211221
2020211221: comma*d *ot fou*d
[/usr/root]# *e*gyuechua*
*e*gyuechua*: comma*d *ot fou*d
[/usr/root]# 2020211221-
2020211221-: command not found
[/usr/root]# wengyuechuan
wengyuechuan: command not found
```

```
[/usr/root]# xxxxxx2020211221
xxxxxx2020211221: comma*d *ot fou*d
[/usr/root]# *e*gyuechua*
*e*gyuechua*: comma*d *ot fou*d
[/usr/root]# 2020211221-
2020211221-: command not found
[/usr/root]# wengyuechuan
wengyuechuan: command not found
```

```
[/usr/root]# 2222020211221
2222020211221: comma*d *ot fou*d
[/usr/root]# *e*gyuechua*
*e*gyuechua*: comma*d *ot fou*d
[/usr/root]# 2020211221-
2020211221-: command not found
[/usr/root]# wengyuechuan
wengyuechuan: command not found
```

```
[/usr/root]# 202020211221
202020211221: comma*d *ot fou*d
[/usr/root]# *e*gyuechua*
*e*gyuechua*: comma*d *ot fou*d
[/usr/root]# 2020211221-
2020211221-: command not found
[/usr/root]# wengyuechuan
wengyuechuan: command not found
```

则结果按照预期进行，成功解决问题

五、总结体会

总结心得（包括实验过程中遇到的问题、如何解决的、过关或挫败的感受、实验投入的时间和精力、

意见和建议等）

实验过程中遇到的问题：

1. 没有很好的理解 Linux 内核，先读 Linux 内核完全注释，然后在实验中逐步理解了实验，然后慢慢上手，解决了 Linux 的内核问题

2. 在过程中，由于对 Linux 中文件操作不够熟练，导致很多时候出现了错误，这个时候需要反复解压，所以，更好的方法是在本地上留一个副本，然后每次使用的时候进行复制，这样更快的解决了问题

3. 本实验考察了 Linux-0.11 内核键盘驱动中输入输出的基本原理，一开始比较迷茫无从下手，仔细阅读书本资料之后有了一定的体会，解答起来也相对容易一些。

实验投入的时间和精力：

用了 3 个小时去熟悉知识理解内容，解答利用了 5 个小时，而大部分时间的浪费出现在不够仔细上，对目标进行反复操作消耗了很多时间。

意见和建议

1. 实验时保留副本，方便复制

2. 可以在主机的编译器上先对 console.c 等文件进行更改再导入到虚拟机中，这样也节省了很多时间

六、诚信声明（不签扣 10 分）

需要填写如下声明，并在底部给出手写签名的电子版。

我参考了以下资料：

1、Linux 内核完全注释

在我提交的程序中，还在对应的位置以注释形式记录了具体的参考内容。

我独立完成了本次实验除以上方面之外的所有工作，包括分析、设计、编码、调试与测试。

我清楚地知道，从以上方面获得的信息在一定程度上降低了实验的难度，可能影响起评分。

我从未使用他人代码，不管是原封不动地复制，还是经过某些等价转换。

我未曾也不会向同一课程（包括此后各届）的同学复制或公开我这份程序的代码，我有义务妥善保管好它们。

我编写这个程序无意于破坏或妨碍任何计算机系统的正常运行。

我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按照 0 分计。

（签名）