

北京邮电大学

实验报告



题目： 键盘驱动程序的分析与修改

班 级： 2020211306

学 号： 2020211376

姓 名： 马天成

学 院： 计算机学院

2021 年 12 月 10 日

目录

一、实验目的	3
二、实验环境	3
三、实验内容	3
四、实验步骤及实验分析	4
准备工作	4
phase1	6
初步理解	6
修改代码	6
运行结果	7
phase2	8
初步理解	8
修改代码	8
运行结果	10
五、总结体会	11
六、诚信声明	11

版本修正：

有限状态自动机的 state 转移修改了一处：见 **P8, P9**

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）:

大二班（5-8 班）: yangyyj98@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

准备工作

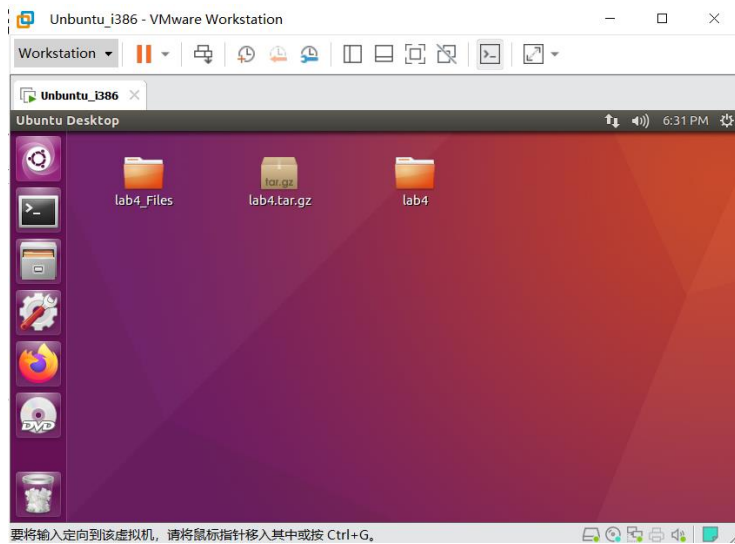
①. 安装虚拟机

 ubuntu-16.04.1-desktop-i386.iso	2021/12/7 20:14	光盘映像文件	1,495,552...
 ubuntu-20.04.3-desktop-amd64.iso	2021/12/7 12:41	光盘映像文件	2,999,936...

选择第一个 32 位光盘映像文件

 CentOS	2021/8/2 13:05	文件夹
 CentOS_x86_64	2021/12/7 20:29	文件夹
 CnetOS_Share	2021/12/7 17:24	文件夹
 Ubuntu	2021/12/7 20:15	文件夹
 Ubuntu_amd64	2021/12/7 20:29	文件夹
 Ubuntu_i386	2021/12/11 10:30	文件夹
 Ubuntu_Share	2021/12/8 22:50	文件夹
 VMpackage	2021/12/7 20:15	文件夹
 VMware WorkStation Pro	2021/12/7 16:43	文件夹

我的 Visual Machine Folder



Ubuntu_i386

②. 安装 C (C++) 环境

```
username@ubuntu: ~  
username@ubuntu:~$ cd Desktop  
username@ubuntu:~/Desktop$ ls  
lab4 lab4_Files lab4.tar.gz  
username@ubuntu:~/Desktop$ cd  
username@ubuntu:~$ ls  
cpp-3.4_3.4.6-6ubuntu3_i386.deb libstdc++6-dev_3.4.6-6ubuntu3_i386.deb  
Desktop  
Documents Music  
Downloads Pictures  
g++-3.4_3.4.6-6ubuntu3_i386.deb Public  
gcc-3.4_3.4.6-6ubuntu3_i386.deb Templates  
gcc-3.4-base_3.4.6-6ubuntu3_i386.deb Videos  
username@ubuntu:~$
```

C(C++)语言运行环境的下载和安装

此外，还有两个东西需要自行解决：

- Vim 版本不够导致上下左右键输出 ABCD
- 不能进行 make 指令

这个需要在网上搜指令解决。

phase1

初步理解

第一题主要是跟着 PPT 走思路，将 io 的思路了解一下。

功能是：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*

- MaTianCheng -> *aTianChen*
- matiancheng -> *atianchen*
- MATIANCHENG -> *ATIANCHEN*

主要是进行两个文件的修改：

- keyboard.S
- console.c

修改代码

①. 修改 keyboard.S

```
username@ubuntu: ~/Desktop/lab4/linux-0.11/kernel/chr_drv
/*
 * this routine handles function keys
 */
func:
    pushl %eax
    pushl %ecx
    pushl %edx
    call show_stat
    popl %edx
    popl %ecx
    popl %eax
    subb $0x3B,%al
    jb end_func
    cmpl $9,%al
    jbe ok_func
    subb $18,%al
    cmpl $10,%al
    jb end_func
    cmpl $11,%al
    ja end_func
ok_func:
    cmpl $4,%ecx          /* check that there is enou
    jl end_func
    movl func_table(,%eax,4),%eax
    xorl %ebx,%ebx
    jmp put_queue
end_func:
    ret
-- INSERT --
```

原来的 keyboard.S

```
username@ubuntu: ~/Desktop/lab4/linux-0.11/kernel/chr_drv
/*
 * this routine handles function keys
 */
func:
    pushl %eax
    pushl %ecx
    pushl %edx
    call show_stat
    popl %edx
    popl %ecx
    popl %eax
    subb $0x3B,%al
    jb end_func
    cmpl $9,%al
    jbe ok_func
    subb $18,%al
    cmpl $10,%al
    jb end_func
    cmpl $11,%al
    ja end_func
    call change_f12Flag
ok_func:
    cmpl $4,%ecx          /* check that there is enou
    jl end_func
    movl func_table(,%eax,4),%eax
    xorl %ebx,%ebx
    jmp put_queue
end_func:
    ret
-- INSERT --
```

现在的 keyboard.S

并且需要加上全局变量

```
.text
.globl keyboard_interrupt, f12Flag
```

②. 修改 console.c

```
int f12Flag = 0 ;
void change_f12Flag( void ) {
    f12Flag = ! f12Flag ;
}
```

void change_f12Flag(void)

```
if ( f12Flag == 1 && ((c=='M')||(c=='m')||(c=='G')||(c=='g')) )
    c = '*' ;

__asm__( "movb attr,%%ah\n\t"
        "movw %%ax,%1\n\t"
        :: "a" (c), "m" (*(short *)pos)
        );
pos += 2;
```

修改条件

运行结果

```
Bochs x86 emulator, http://bochs.sourceforge.net/
Options: apmbios pcibios eltorito rombios32
ata0 master: Generic 1234 ATA-6 Hard-Disk ( 60 MBytes)
Booting from Floppy...
Loading system ...
Partition table ok.
39044/62000 free blocks
19520/20666 free inodes
3454 buffers = 3536896 bytes buffer space
Free mem: 12582912 bytes
Ok.
[usr/root]# MaTianCheng_matiancheng MATIANCHENG    0: pid=0, state=1, 2740 (of
3140) chars free in kernel stack
1: pid=1, state=1, 2568 (of 3140) chars free in kernel stack
2: pid=4, state=1, 1448 (of 3140) chars free in kernel stack
3: pid=3, state=1, 1448 (of 3140) chars free in kernel stack
B *aTianChen* *atianchen* *
aTianChen* 0: pid=0, state=1, 2588 (of 3140) chars free in kernel stack
1: pid=1, state=1, 2568 (of 3140) chars free in kernel stack
2: pid=4, state=1, 1448 (of 3140) chars free in kernel stack
3: pid=3, state=1, 1448 (of 3140) chars free in kernel stack
aMaTianCheng
CTRL + 3rd button enables mouse | A: | HD:0-M | NUM | CAPS | SCRL |
```

phase2

初步理解

显然，这是一个有限状态自动机的问题。那么只需要把转移条件写出来就行。

state	state++	state=1	state=4
0	✓		
1	0	✓	
2	2		
3	0	✓	
4	2		
5	1	✓	0
6	1	✓	
7	3	✓	
8	7	✓	
9	6	✓	
10	state=0. flag 修改		

state : 0 -> 10

表示输入后已经正确了几个数字。

其中，状态 5 极为特殊：

在数字为 0 时是返回 state = 4

Flag : 0 1

1 表示屏蔽，0 表示正常。

修改代码

① ./run init 清空第一题

②. 修改 console.c

```
// -----  
int myState = 0 ;  
int Flag376 = 0 ;  
// -----
```

变量


```

// -----
switch( myState ) {
    case( 0 ): {
        if ( c == '2' )
            myState++ ;
        } break ;

    case( 1 ): {
        if ( c == '0' )
            myState++ ;
        else if ( c == '2' )
            myState = 1 ;
        else
            myState = 0 ;
        } break ;

    case( 2 ): {
        if ( c == '2' )
            myState++ ;
        else
            myState = 0 ;
        } break ;

    case( 3 ): {
        if ( c == '0' )
            myState++ ;
        else if ( c == '2' )
            myState = 1 ;
        else
            myState = 0 ;
        } break ;

    case( 4 ): {
        if ( c == '2' )
            myState++ ;
        else
            myState = 0 ;
        } break ;

    case( 5 ): {
        if ( c == '1' )
            myState++ ;
        else if ( c == '2' )
            myState = 1 ;
        else if ( c == '0' )
            myState = 2 ;
        else
            myState = 4 ;
        } break ;
}

```

code_1

```

        case( 6 ): {
            if ( c == '1' )
                myState++ ;
            else if ( c == '2' )
                myState = 1 ;
            else
                myState = 0 ;
        } break ;

        case( 7 ): {
            if ( c == '3' )
                myState++ ;
            else if ( c == '2' )
                myState = 1 ;
            else
                myState = 0 ;
        } break ;

        case( 8 ): {
            if ( c == '7' )
                myState++ ;
            else if ( c == '2' )
                myState = 1 ;
            else
                myState = 0 ;
        } break ;

        case( 9 ): {
            if ( c == '6' )
                myState++ ;
            else if ( c == '2' )
                myState = 1 ;
            else
                myState = 0 ;
        } break ;
    }

    if ( myState == 10 ) {
        myState = 0 ;
        Flag376 = ! Flag376 ;
    }

    if ( Flag376 == 1 && ( (c=='M')||(c=='m')||(c=='G')||(c=='g') ) )
        c = '*' ;
    // -----

```

code_2

运行结果

```

Bochs x86 emulator, http://bochs.sourceforge.net/
. http://www.nongnu.org/vgabios
Bochs VBE Display Adapter enabled
Bochs BIOS - build: 02/13/08
$Revision: 1.194 $ $Date: 2007/12/23 19:46:27 $
Options: apmbios pcibios eltorito rombios32
ata0 master: Generic 1234 ATA-6 Hard-Disk ( 60 MBytes)
Booting from Floppy...
Loading system ...
Partition table ok.
39044/62000 free blocks
19520/20666 free inodes
3454 buffers = 3536896 bytes buffer space
Free mem: 12582912 bytes
Ok.
[/usr/root]# MaTianCheng matiancheng MATIANCHENG 2020211376
*aTianChen*: co**and not found
[/usr/root]# *aTianChen* *atianchen* *ATIANCHEN* 2020211376
MaTianCheng: command not found
[/usr/root]# MaTianCheng
CTRL + 3rd button enables mouse  A:  HD:0-H NUM CAPS SCRL

```

五、总结体会

本次实验较之前比较简单,难点主要在 ubuntu 环境下软件的配置问题以及理解各个层面的相互调用上,在动手完成 ppt 上的实例后已经对整个层面有了初步的了解,再加上阅读一定的文献,对 linux 的 io 内核有了更深层的理解,所以实验做起来也算得心应手。因为之前大一上 oj 上练过好几次有限状态自动机的题,所以看到第二阶段的意思也没有很慌张,而是很快画出了状态图并转换成了 c 代码,完成了阶段 2。

在下载安装 as86 的过程中,出现了很多问题,我也在不断寻找解决问题的途径和手段,在这个过程中对于 sudo 等终端内的指令更加熟练的操作,并通过在 ubuntu 系统下更加明显的感受到 linux 系统相比于 windows 在开源程度等方面的差异和区别,在 linux 环境下,有效帮助我更加了解了计算机 io 等深层计算机功能的认识。

本次实验很令人愉悦,增强了对 linux 系统的操作熟练度,让我真正体会到了命令行控制电脑的魅力。

六、诚信声明

需要填写如下声明,并在底部给出手写签名的电子版。

此外,我还参考了以下资料:

linux 指令:

- 重装新版本 `Vim sudo apt-get install vim`
- 重装 `yum -y install make`

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)