

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： 2022211320

学 号： 2022211683

姓 名： 张晨阳

学 院： 计算机学院（国家示范性软件学院）

2022 年 12 月 16 日

一、实验目的

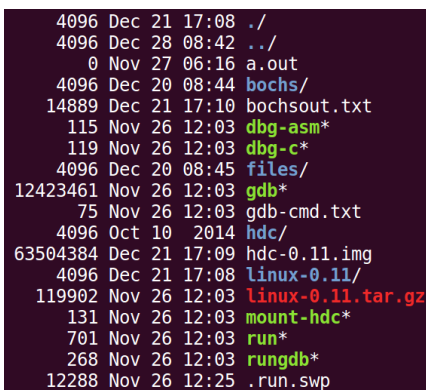
1. 理解 I/O 系统调用函数和 C 标准 I/O 函数的概念和区别；
2. 建立内核空间 I/O 软件层次结构概念，即与设备无关的操作系统软件、设备驱动程序和中断服务程序；
3. 了解 Linux-0.11 字符设备驱动程序及功能，初步理解控制台终端程序的工作原理；
4. 通过阅读源代码，进一步提高 C 语言和汇编程序的编程技巧以及源代码分析能力；
5. 锻炼和提高对复杂工程问题进行分析的能力，并根据需求进行设计和实现的能力。

二、实验环境

1. 硬件：学生个人电脑（x86-64）
2. 软件：Windows 11, VMware Workstation 16 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

从网盘下载 lab4.tar.gz 文件，解压后进入 lab4 目录得到如下文件和目录：



```
4096 Dec 21 17:08 ./
4096 Dec 28 08:42 ../
0 Nov 27 06:16 a.out
4096 Dec 20 08:44 bochs/
14889 Dec 21 17:10 bochsout.txt
115 Nov 26 12:03 dbg-asm*
119 Nov 26 12:03 dbg-c*
4096 Dec 20 08:45 files/
12423461 Nov 26 12:03 gdb*
75 Nov 26 12:03 gdb-cmd.txt
4096 Oct 10 2014 hdc/
63504384 Dec 21 17:09 hdc-0.11.img
4096 Dec 21 17:08 linux-0.11/
119902 Nov 26 12:03 linux-0.11.tar.gz
131 Nov 26 12:03 mount-hdc*
701 Nov 26 12:03 run*
268 Nov 26 12:03 rungdb*
12288 Nov 26 12:25 .run.swp
```

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“学生本人学号”：激活*功能，键入学生本人姓名拼音，首尾字母等显示*
比如：zhangsan，显示为：*ha*gsa*，
再次键入“学生本人学号-”：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

实验 4 的其他说明见 lab4.pdf 课件和爱课堂中虚拟机环境搭建相关内容。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、源代码的分析及修改

准备阶段

此阶段完成实验环境的配置。按照实验内容上的要求一步一步配置环境如下：

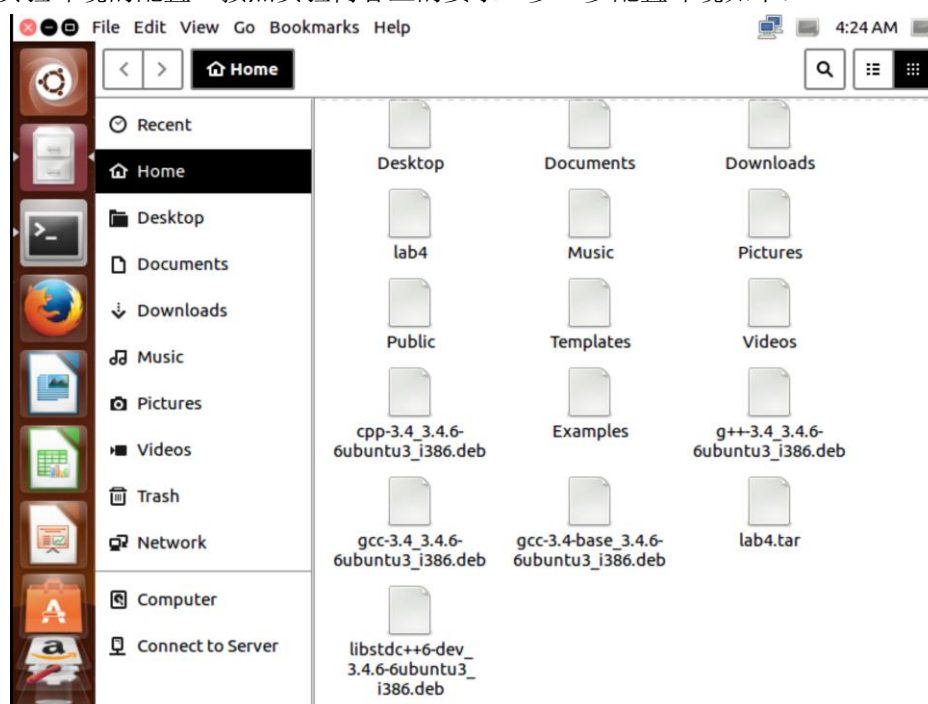


图 1-环境配置

解压 lab4.tar 文件：

```
csapp123@ubuntu:~$ cd lab4
csapp123@ubuntu:~/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  run gdb
csapp123@ubuntu:~/lab4$
```

图 2-文件解压

至此完成准备阶段。

第一阶段

第一阶段需要实现这样一个功能：键入 F12，激活*功能，键入学生本人姓名拼音，首尾字母等显示*，而且该过程是可逆的，即：再次键入 F12 键的时候将恢复原先状态。

根据《linux 内核完全注释》第 426 页“10.1.5.1 控制台驱动程序”的部分内容以及查阅相关资料可知该阶段流程大致为：首先在 tty_io.c 文件中判断字符是否回显，若回显，再判断是否为 F12，直接通过 write_q 等输出；若不回显，则进入 secondary，之后再通过 write_q 等输出。

则实现该功能具体流程图如下：

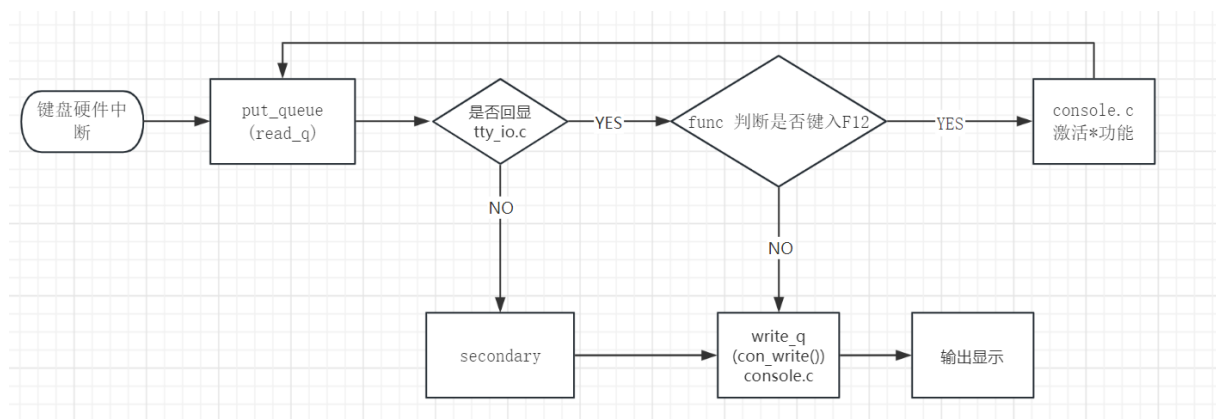


图 3-阶段一流程图

故考虑在 keyboard.S 函数中对键入字符进行检测，如果键入的字符是 F12 键，那么就将表示 F12 的 F12flag 的值进行翻转。根据参考书第 438 页关于 keyboard.S 程序的描述来看，其中的 func 函数将处理键入字符为 F12 的情况，具体代码如下：

```
console.c keyboard.S
#endef
cur_table:
    .ascii "HAS DGC YB623"

/*
 * 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
    cmpb $9,%al
    jbe ok_func
    subb $18,%al
    cmpb $10,%al
    jb end_func
    cmpb $11,%al
    ja end_func
ok_func:
    cmpl $4,%ecx          /* check that there is enough room */
    jl end_func
    movl func_table(,%eax,4),%eax
C Tab Width: 8 Ln 226, Col 20 INS
```

图 4-func 函数代码

查找资料可知 F12 和 F11 的扫描码如下：

按 键	单 键		SHIFT		CTRL		ALT	
按 键	扫描码	ASCII码	扫描码	ASCII码	扫描码	ASCII码	扫描码	ASCII码
F1	3B	00	54	00	5E	00	68	00
F2	3C	00	55	00	5F	00	69	00
F3	3D	00	56	00	60	00	6A	00
F4	3E	00	57	00	61	00	6B	00
F5	3F	00	58	00	62	00	6C	00
F6	40	00	59	00	63	00	6D	00
F7	41	00	5A	00	64	00	6E	00
F8	42	00	5B	00	65	00	6F	00
F9	43	00	5C	00	66	00	70	00
F10	44	00	5D	00	67	00	71	00
F11	85	00	87	00	89	00	8B	00
F12	86	00	88	00	8A	00	8C	00

图 5-F12、F11 扫描码

分析 func 函数代码段：

键入字符的扫描码存储在寄存器 %al 中。第 8 行和第 12 行减去 59 和 18 是为了后面根据功能键的索引号来直观地判断键入的字符。第 13 行判断键入的字符是否是 F11；第 15 行判断键入的字符是否是 F12，如果不是则不进行任何处理。

在该汇编代码的后面增添语句 call change_F12flag，该语句将在键入字符为 F12 的时候被调用，从而翻转 F12flag 的值：

```
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
```

图 6-keyboard.S 内容修改

在汇编代码更改的基础上，还需要更改相应的 C 代码：

change_F12flag 函数应存储在 console.c 程序中，打开 console.c 程序并在其中新增全局变量 F12flag 和函数 change_F12flag 如下：

```
int F12flag = 0;

void change_F12flag(void)
{
    switch(F12flag){
        case 0:
            F12flag = 1;
            break;
        case 1:
            F12flag = 0;
            break;
    }
}
```

图 7-console.c 代码修改

接下来需要实现当 F12flag==1 时，将姓名首尾字母显示为 * 的功能。

首先确定需要显示为 * 的字符：本人姓名拼音为 zhangchenyang，故需要将 z 和 g 显示为 *；

根据参考书第 462 页关于 con_write 的描述，可以在此函数中实现输出功能的修改，只需添加如下语句：

```
if (c>31 && c<127) {
    if (x>=video_num_columns) {
        x -= video_num_columns;
        pos -= video_size_row;
        lf();
    }
    if (F12flag && (c == 'z' || c == 'g'))
        c = '*';
    __asm__("movb attr,%%ah\n\t"
            "movw %%ax,%i\n\t"
            ::"a" (c), "m" (*(short *)pos)
            );
    pos += 2;
    x++;
}
```

图 8-con_write 函数修改

保存上述修改，在路径 linux-0.11 下使用 make clean 命令和 make 命令生成相应的 image 文件：

```
csapp123@ubuntu:~/lab4/linux-0.11$ ll
total 292
drwxrwxr-x 10 csapp123 csapp123 4096 Dec 16 06:11 ./
drwxrwxr-x 6 csapp123 csapp123 4096 Dec 4 2019 ../
drwxr-xr-x 2 csapp123 csapp123 4096 Dec 16 06:11 boot/
drwxr-xr-x 2 csapp123 csapp123 4096 Dec 16 06:11 fs/
-rw-rw-r-- 1 csapp123 csapp123 124065 Dec 16 06:11 Image
drwxr-xr-x 5 csapp123 csapp123 4096 Sep 4 2010 include/
drwxr-xr-x 2 csapp123 csapp123 4096 Dec 16 06:11 init/
drwxr-xr-x 5 csapp123 csapp123 4096 Dec 16 06:11 kernel/
drwxr-xr-x 2 csapp123 csapp123 4096 Dec 16 06:11 lib/
-rw-rw-r-- 1 csapp123 csapp123 3442 Aug 23 2011 Makefile
drwxr-xr-x 2 csapp123 csapp123 4096 Dec 16 06:11 mm/
-rw-rw-r-- 1 csapp123 csapp123 10774 Dec 16 06:11 System.map
-rw-rw-r-- 1 csapp123 csapp123 110724 Oct 9 2008 tags
drwxr-xr-x 2 csapp123 csapp123 4096 Dec 16 06:11 tools/
```

图 9-生成 image

然后在路径 lab4 下使用指令 ./run 可以启动 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，实验截图如下：

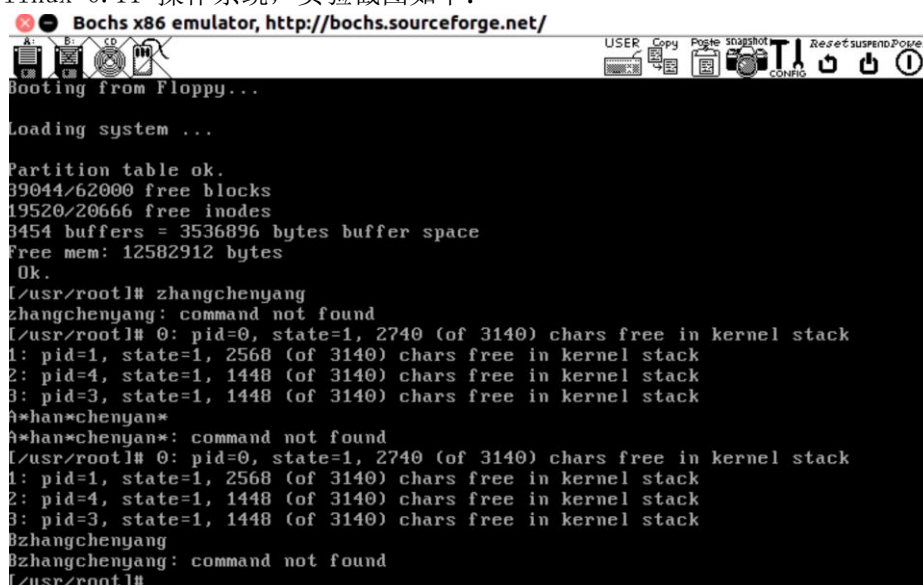


图 10-阶段一成功

如图 10，一开始正常输入“zhangchenyang”，键入 F12 之后，输入“zhangchenyang”，显示出“*han*chenyan*”，再次键入 F12 后，又正常显示出“zhangchenyang”。

第一阶段实验完成。

第二阶段

本阶段需要实现功能：键入学号“2022211683”之后，激活*功能，键入学生本人姓名拼音，首尾字母等显示*，且该过程可逆：键入“2022211683-”时恢复正常状态。

本阶段实现思路与阶段一相似：在第一阶段中我们设置了变量 F12flag 来判断 * 功能是否应该被触发，在第二阶段中，因为是通过键入一个字符串来判断是否实现功能，故引入一个全局变量 count，来标记字符串相对应的位置，具体流程如下：

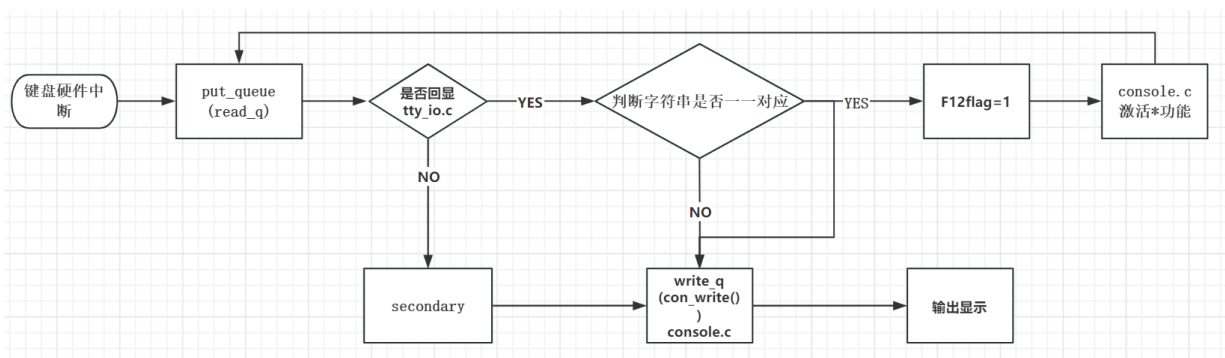


图 11-阶段二流程图

接下来考虑如何具体实现 count 的标记：因为输入是一个字符一个字符的进行的，所以使用数组存储并不现实。考虑到输入的字符串数字顺序是固定的，写出如下代码：

```
void con_write(struct tty_struct * tty)
{
    int nr;
    char c;

    nr = CHARS(tty->write_q);
    while (nr--) {
        GETCH(tty->write_q, c);
        switch(state) {
            case 0:
                if (c > 31 && c < 127) {
                    if (x >= video_num_columns) {
                        x -= video_num_columns;
                        pos -= video_size_row;
                        lf();
                    }
                }

                if (c == '2' && (count == 1 || count == 3 || count == 4 || count == 5))
                    count++;
                else if (c == '0' && (count == 2))
                    count++;
                else if (c == '1' && (count == 6 || count == 7))
                    count++;
                else if (c == '6' && (count == 8))
                    count++;
                else if (c == '8' && (count == 9))
                    count++;
                else if (c == '3' && (count == 10)) {
                    count++;
                    F12flag = 1;
                }
                else if (c == '-' && (count == 11)) {
                    count = 1;
                    F12flag = 0;
                }
                else
                    count = 1;

                if (F12flag && (c == 'z' || c == 'g'))
                    c = '*';
                __asm__ ("movb attr, %%ah\n\t"
                        "movw %%ax, %i\n\t"
                        :: "a" (c), "m" (*(short *)pos)
                        );
                pos += 2;
                ....
            }
        }
    }
}
```

图 12-判断学号代码部分

因为学号是需要显示的，所以修改部分在 con_write()中即可。

在路径 linux-0.11 使用 make clean 命令和 make 命令生成相应的 image 文件，并在路径 lab4 下使用指令 ./run 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，实验截图如下：

```
[/usr/root]# zhangchenyang
zhangchenyang: command not found
[/usr/root]# 2022211683
2022211683: command not found
[/usr/root]# *han*chenyan*
*han*chenyan*: command not found
[/usr/root]# 2022211683-
2022211683-: command not found
[/usr/root]# zhangchenyang
zhangchenyang: command not found
[/usr/root]#
```

图 13-阶段二成功

如图 13，一开始正常输入“zhangchenyang”，键入“2022211683”后，输入“zhangchenyang”，显示出“*han*chenyan*”，再键入“2022211683-”后，又正常显示出“zhangchenyang”。

第二阶段实验完成。

五、总结体会

所遇问题及解决:

1. 准备阶段

(1) sudo apt install bin86 后出现报错: unable to locate package bin86

```
csapp123@ubuntu:~$ sudo apt install bin86
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package bin86
```

图 14-配置环境报错

解决办法: 执行命令 sudo apt-get update 后再执行 sudo apt install bin86 即可:

```
csapp123@ubuntu:~$ sudo apt-get update
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-backports/universe i386 DEP-11 Metadata [6,612 B]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial-backports/universe DEP-11 64x64 Icons [5,608 B]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial-backports/multiverse i386 DEP-11 Metadata [216 B]
Get:4 http://us.archive.ubuntu.com/ubuntu xenial-backports/multiverse DEP-11 64x64 Icons [29 B]
Fetched 34.6 MB in 19s (1,764 kB/s)
Reading package lists... Done
csapp123@ubuntu:~$ sudo apt install bin86
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bin86
0 upgraded, 1 newly installed, 0 to remove and 730 not upgraded.
Need to get 85.7 kB of archives.
After this operation, 217 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe i386 bin86 i386 0.16.17-3.1ubuntu3 [85.7 kB]
Fetched 85.7 kB in 1s (55.6 kB/s)
Selecting previously unselected package bin86.
(Reading database ... 173152 files and directories currently installed.)
Preparing to unpack .../bin86_0.16.17-3.1ubuntu3_i386.deb ...
Unpacking bin86 (0.16.17-3.1ubuntu3) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up bin86 (0.16.17-3.1ubuntu3) ...
csapp123@ubuntu:~$
```


图 15-报错解决方案

(2) 无法从 windows 系统直接拖动 lab4. tar 文件至虚拟机。分析原因: 使用的 VMware Workstation 是大一时安装的, 没有安装 VMware Tools for Linux。

解决办法：安装 VMware Tools。

2. 阶段一

(1) 无法生成 image 文件：



```
csapp123@ubuntu: ~/lab4/linux-0.11
sync
make[1]: Leaving directory '/home/csapp123/lab4/linux-0.11/kernel/blk_drv'
(cd kernel/chr_drv; make)
make[1]: Entering directory '/home/csapp123/lab4/linux-0.11/kernel/chr_drv'
gcc-3.4 -march=i386 -m32 -g -Wall -O -fstrength-reduce -fomit-frame-pointer -f
inline-functions -nostdinc -I../include \
-c -o tty_io.o tty_io.c
tty_io.c: In function 'copy_to_cooked':
tty_io.c:160: warning: subscript has type 'char'
tty_io.c: In function 'tty_write':
tty_io.c:316: warning: subscript has type 'char'
gcc-3.4 -march=i386 -m32 -g -Wall -O -fstrength-reduce -fomit-frame-pointer -f
inline-functions -nostdinc -I../include \
-c -o console.o console.c
console.c: In function 'con_write':
console.c:461: error: 'F12flag' undeclared (first use in this function)
console.c:461: error: (Each undeclared identifier is reported only once
console.c:461: error: for each function it appears in.)
Makefile:24: recipe for target 'console.o' failed
make[1]: *** [console.o] Error 1
make[1]: Leaving directory '/home/csapp123/lab4/linux-0.11/kernel/chr_drv'
Makefile:82: recipe for target 'kernel/chr_drv/chr_drv.a' failed
make: *** [kernel/chr_drv/chr_drv.a] Error 2
```

图 16-生成 image 文件报错

分析其报错：

```
console.c: In function 'con_write':
console.c:461: error: 'F12flag' undeclared (first use in this function)
```

图 17-报错原因分析

可知：F12flag 应该定义在 con_write 前面，修改代码将 F12flag 及其函数定义至 con_write 函数前即可。

心得体会：

本次实验难度中等，用时约 5 小时。

在配置环境部分，因为大一自己装过虚拟机使用，这次安装时提示没有办法在同一台电脑上同时装两个 workstation，而卸载又比较麻烦，索性直接使用大一装的虚拟机。因为已经不记得当时的所作所为，耗费了一部分时间来研究虚拟机的重新配置，幸运的是成功了，不需要重新卸载安装。

在第一阶段，我一开始没有参照《linux 内核完全注释》来研究，因此寻找处理键入 F12 的函数花费了不少的时间，后来学习率相关内容就很快找到了函数 func。一开始我的全局变量设置位置不对，导致无法生成 image 文件，后来分析了报错内容解决了这个问题。

作为计算机系统基础课的最后一个实验，我重温了虚拟机的配置操作，学习了 linux 的相关知识，并且动手在虚拟机上对 linux 的源代码进行修改，既检测了对汇编代码的掌握程度，又增强了对结合不同文件最终生成 image 文件的认识。

建议和意见：

个人觉得可以借着这个实验，给大家提供一份虚拟机的使用指南/操作手册，便于大家学习虚拟机。因为虚拟机并不是只用来做实验而已，之后也会和它继续的打交道，可以接着大家的兴趣和新鲜感让同学们更深入的自发的学习、使用、拓展虚拟机。