东莞理工学院

操作系统课程设计报告

院　　系：计算机学院

班　　级：14 软卓

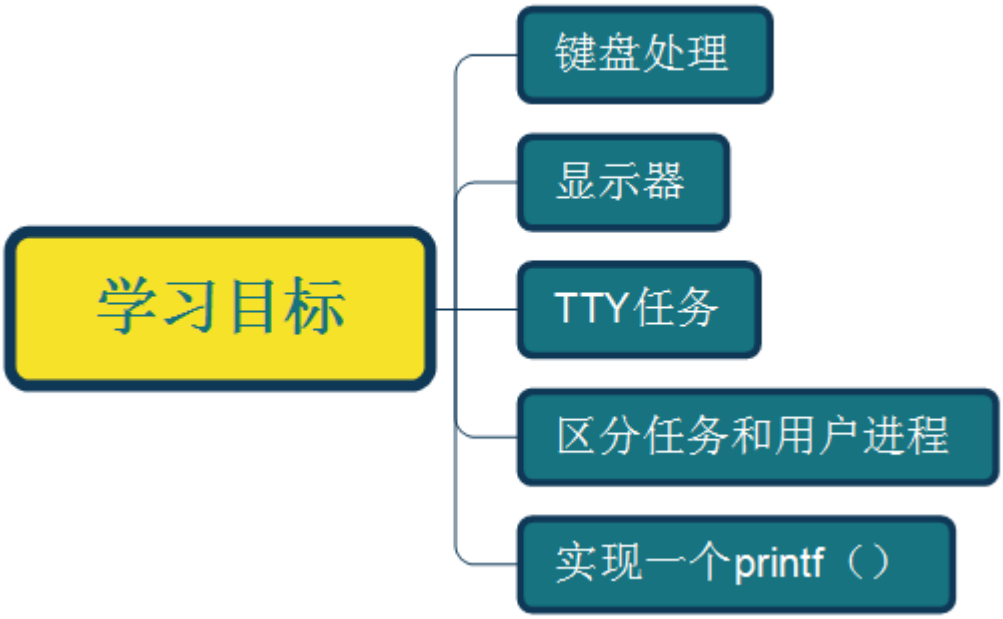
姓　　名：赖键锋

学　　号：201441402130

指导老师：李伟

日　　期：2016.6 - 2016.7

一、 相关说明



二、 相关知识的记录和说明

1．打开键盘中断并关联键盘中断处理函数

```
18  PUBLIC void keyboard_handler(int irq)
19  {
20      disp_str("*");
21  }

27  PUBLIC void init_keyboard()
28  {
29      put_irq_handler(KEYBOARD_IRQ, keyboard_handler);/*设定键盘中断处理程序*/
30      enable_irq(KEYBOARD_IRQ);                       /*开键盘中断*/
31  }
```

2. 键盘把扫描码送入缓冲区后，只有把缓冲区中的扫描码读出来后，8042 才会继续响应新的按键。通过对 8042 的输入和输出缓冲区寄存器端口进行 in，out 指令来进行读取缓冲区。

3. 按下组合键如“左 shit”+ “a”:得到的输出为：左 shit 的 Make

Code，a 的 Make Code，a 的 Break Code，左 Shit 的 Break Code。

4. 按下任何的键，不管是单键还是组合键，想让屏幕输出什么，或者产生什么反应，都是由软件来控制的，这种机制比较灵活。

5. Break Code 是 Make Code 与 0x80 进行 OR 或运算的结果。其中，0xE0 和 0xE1 开头的扫描码要区别对待。

5. 扫描码在计算机中用

数组 keymap[NR_SCAN_CODES * MAP_COLS] 来存储：

没按shit时	按shit时	E0 XX
0	0	0
ESC	ESC	0
‘1’	‘!’	0
‘2’	‘@’	0
‘3’	‘#’	0
‘4’	‘$’	0
…		0
…		0
‘q’	‘Q’	0
‘w’	‘W’	0
‘e’	‘E’	0
…	…	

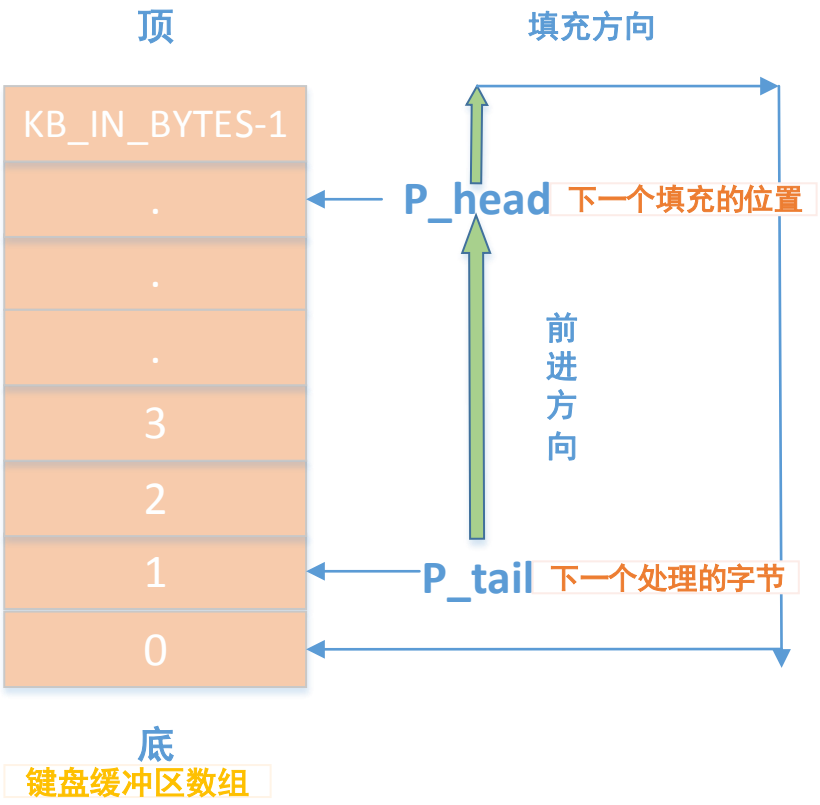
在解码时，现根据扫描码判断是数组的哪一行，然后在根据是否有按 shit 键判断列，最终获得字符。

由于键盘操作可以是很频繁和复杂的，所有我们用键盘中断处理函数

keyboard_handler（）将扫描码先存进缓冲区中。

6. 键盘缓冲区

```
122  /* Keyboard structure, 1 per console. */
123  typedef struct s_kb {
124      char*   p_head;          //指向缓冲区中下一个空闲位置
125      char*   p_tail;          //指向键盘任务应处理的字节
126      int count;           //缓冲区中共有多少字节
127      char    buf[KB_IN_BYTES];    //缓冲区
128  }KB_INPUT;
```

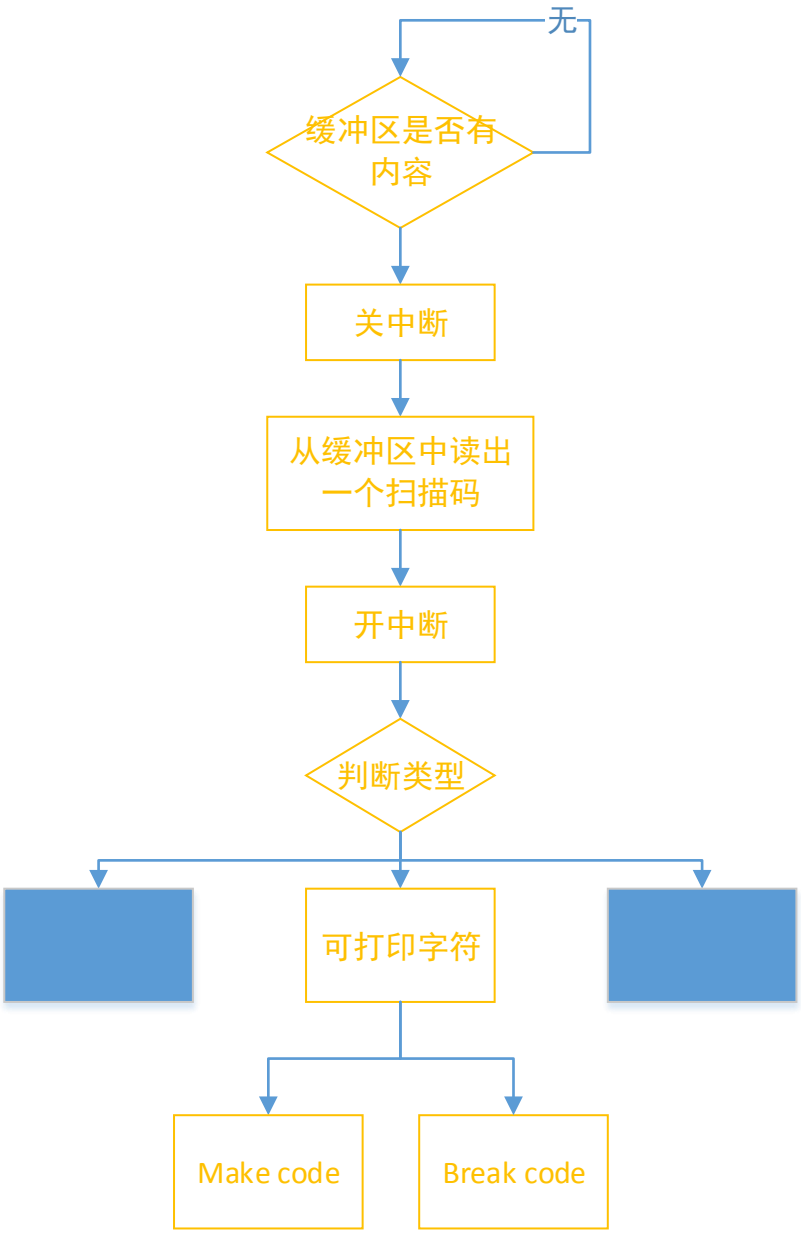


7. 键盘中断的处理：用一个任务 task_tty()，不停地调用函数 keyboard_reak(),从缓冲区中读出扫描码并处理。

```
21  PUBLIC void task_tty()
22  {
23      while (1) {
24          keyboard_read();
25      }
26  }
```

8. 扫描码的解析函数 keyboard_read():

解析扫描码的一般流程：



9. 扫描码的处理函数 keyboard_read():

```
71  PUBLIC void keyboard_read()
72  {
73      u8  scan_code;
74      char    output[2];
75      int make;   /* 1: make;  0: break. */
76      u32 key = 0;/* 用一个整型来表示一个键。比如，如果 Home 被按下，
77                   * 则 key 值将为定义在 keyboard.h 中的 'HOME'。
78                   */
79      u32*    keyrow; /* 指向 keymap[] 的某一行 */
80
81      if(kb_in.count > 0){
82          code_with_E0 = 0;
83
84          scan_code = get_byte_from_kbuf();
85
```

从缓冲区中读取一个扫描码

读出一个扫描码 scan_code 后就要解析各种可能的情况：

A. 是 0xE1 类型的：单独处理

B. 是 0xE0 类型的：单独处理

C. 其他普通情况：

- 先判断是 make code 还是 break code：利用 make code 和 break code 的区别，break code 是 make code 与 0x80 进行 OR 运算的结果。
- 获取扫描码在数组中的行数：利用规律
- 再根据此扫描码之前的按键是否为 shit，判断所在列数
- 最后调用 in_process(key)上层处理函数来处理。(区分 ctrl,shit,alt)

10. 80 * 25 显示器：每个字符的显示需要两个字节：

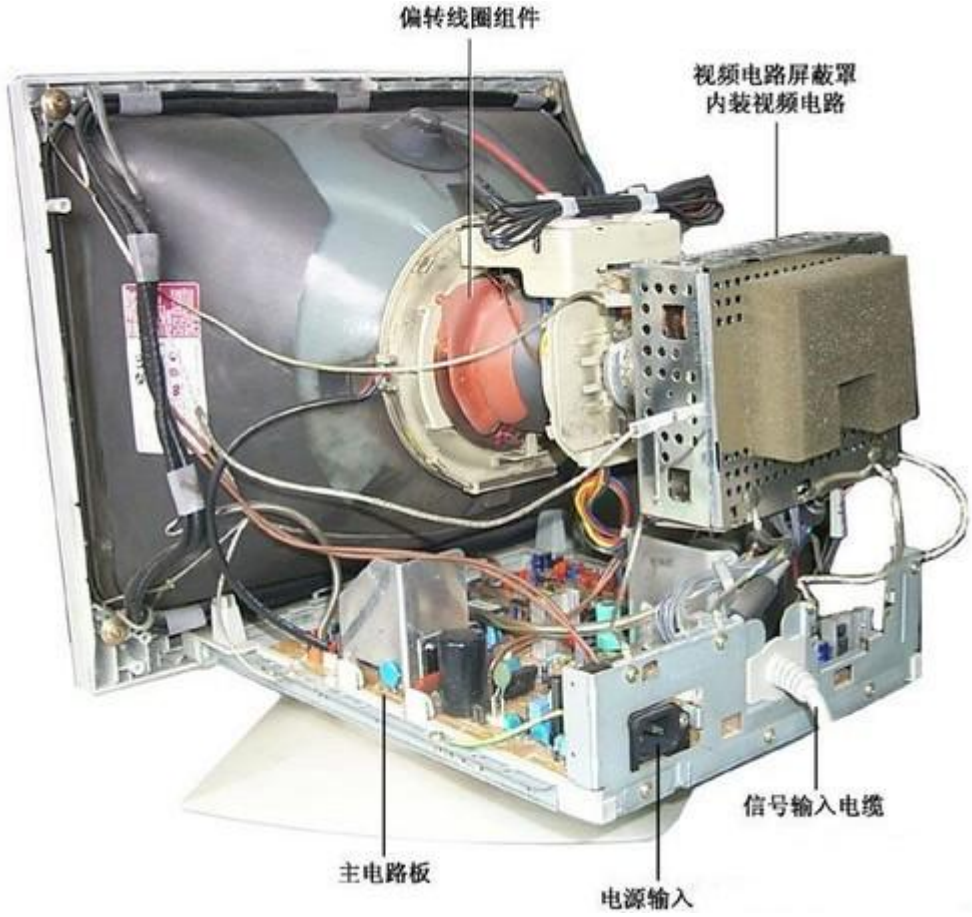
背景				前景			
闪烁	R	G	B	高亮	R	G	B
7	6	5	4	3	2	1	0

颜色设置

- 颜色设置在高字节，字符在低字节；颜色符合 RGB 配色方案。
- 一行最多可以显示 80 个字节，一行共占 160 个字节；
- 一个屏幕有 25 行，一个屏幕占用 160 * 25 = 4000 字节；
- 显存有 32K，可以存放 8 个屏幕的数据。

11．显示器的显示原理，通过对寄存器的操作，控制输出。

CRT 显示器学名为“阴极射线显像管”，是一种使用阴极射线管（Cathode Ray Tube）的显示器。



常见的显示接口：

VGA 接口

显卡所处理的信息最终都要输出到显示器上，显卡的输出接口就是电脑与显示器之间的桥梁，它负责向显示器输出相应的图像信号。CRT显示器因为设计制造上的原因，只能接受模拟信号输入，这就需要显卡能输入模拟信号。VGA接口就是显卡上输出模拟信号的输出接口，VGA（Video Graphics Array）接口，也叫D-Sub接口。

VGA接口是一种D型接口，上面共有15针空，分成三排，每排五个。

HDMI 接口

幻灯播放

高清晰度多媒体接口（英文：High Definition Multimedia Interface，HDMI）是一种数字化视频/音频接口技术，是适合影像传输的专用型数字化接口，其可同时传送音频和影音信号，最高数据传输速度为5Gbps。同时无需在信号传送前进行数/模或者模/数转换。HDMI可搭配宽带数字内容保护（HDCP），以防止具有著作权的影音内容遭到未经授权的复制。

HDMI的规格书中规定了四种HDMI接头：Type A(应用于HDMI 1.0版本)、Type B(应用于HDMI 1.0版本)、Type C(应用于HDMI 1.3版本)和Type D(应用于HDMI 1.4版本)。

DVI 接口

幻灯播放

的英文全名为Digital Visual Interface，中文称为“数字视频接口”。是一种视频接口标准，设计的目标是通过数字化的传送来强化个人电脑显示器的画面品质。目前广泛应用于LCD、数字投影机等显示设备上。此标准由显示业界数家领导厂商所组成的论坛：“数字显示工作小组”（Digital Display Working Group，DDWG）制订。DVI接口可以传送未压缩的数字视频数据到显示设备。

DVI-I(Dual Link)接口

DP 接口

VGA 寄存器分为六组：即外部寄存器、CRT 控制器寄存器、定序器寄存器、图形控制器寄存器、属性控制器寄存器和数模转换器寄存器。设置显示效果就是操作端口。

如设置光标位置和滚屏：

```
31  PUBLIC void in_process(u32 key)
32  {
33          char output[2] = {'\0', '\0'};
34
35          if (!(key & FLAG_EXT)) {
36                  output[0] = key & 0xFF;
37                  disp_str(output);
38
39          disable_int();
40          out_byte(CRTC_ADDR_REG, CURSOR_H);
41          out_byte(CRTC_DATA_REG, ((disp_pos/2)>>8)&0xFF);
42          out_byte(CRTC_ADDR_REG, CURSOR_L);
43          out_byte(CRTC_DATA_REG, (disp_pos/2)&0xFF);
44          enable_int();
45          }
46          else {
47                  int raw_code = key & MASK_RAW;
48                  switch(raw_code) {
49                  case UP:
50                          if ((key & FLAG_SHIFT_L) || (key & FLAG_SHIFT_R)) {
51                                  disable_int();
52                                  out_byte(CRTC_ADDR_REG, START_ADDR_H);
53                                  out_byte(CRTC_DATA_REG, ((80*15) >> 8) & 0xFF);
54                                  out_byte(CRTC_ADDR_REG, START_ADDR_L);
55                                  out_byte(CRTC_DATA_REG, (80*15) & 0xFF);
56                                  enable_int();
57                          }
58                          break;
59                  case DOWN:
```

设置光标位置跟随字符

向上滚屏

12. TTY

系统有一个固定的显存内存位置，显示的最基本原理就是 N 个字节里面保存字符和颜色信息，最常见的是的 565 色。多终端机制的一般规则为多个终端对应一个屏幕，用户在使用一个屏幕的时候，可以随时切换到其他屏幕而好像是切换了用户一样，只有当前聚焦的 TTY 才会从键盘获得输入。实现这样机制的一个进程叫做 TTY 进程，他作为第一个微内核以为的系统进程而存在，已区别于一般的用户进程。

大概实现思路：

```
30  PUBLIC void task_tty()
31  {
32      TTY*    p_tty;
33
34      init_keyboard();
35
36      for (p_tty=TTY_FIRST;p_tty<TTY_END;p_tty++) {
37          init_tty(p_tty);
38      }
39      select_console(0);
40      while (1) {
41          for (p_tty=TTY_FIRST;p_tty<TTY_END;p_tty++) {
42              tty_do_read(p_tty);
43              tty_do_write(p_tty);
44          }
45      }
46  }
```

初始化键盘

遍历初始化TTY

当前TTY才会从键盘获得输入

TTY将自己缓冲区的内容读出来

13. vsprintf() 这个函数的作用。它是 printf() 的一个处理函数。

它接受 3 个变量：

- buf 是用来存放已经处理完的字符串，就是解析后的字符串，初始字符串中的%s,%X 等已经被后面的参数替代了；
- fmt 很明显，就是 printf() 的第一个参数，即初始的字符串，它包含未解析的%s，%x 等。

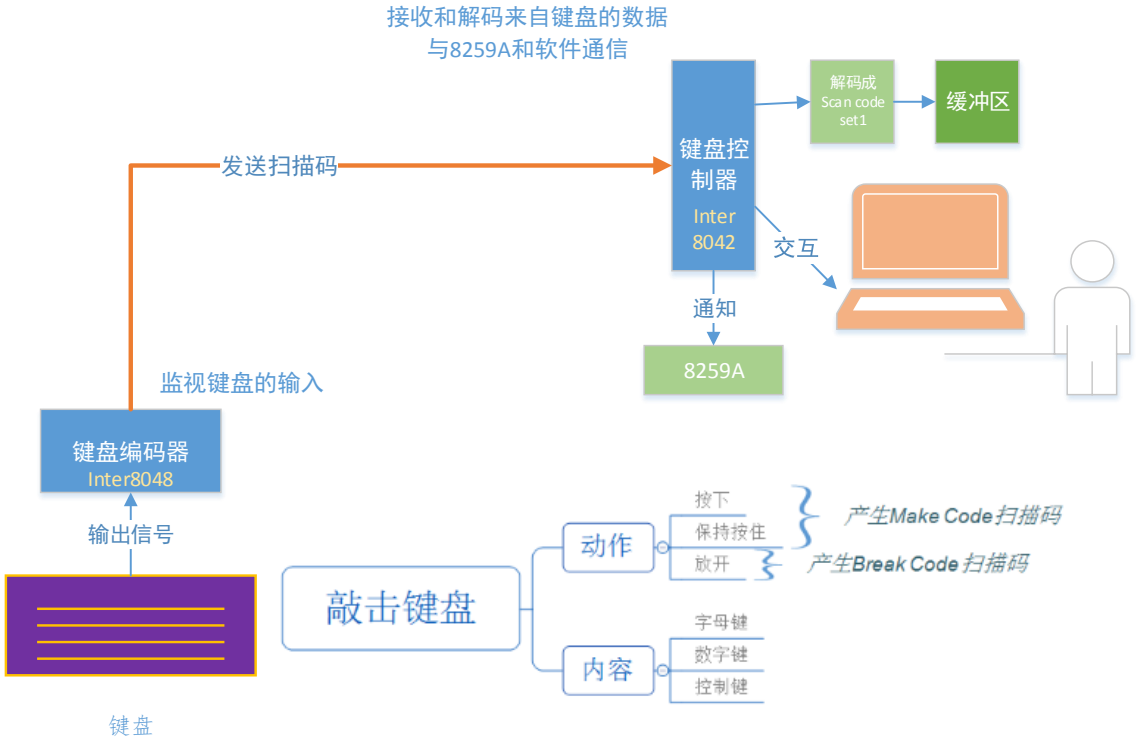
- 第三个参数就是 arg 就是 printf 后面的参数，可变参数（多个参数），其中的每个参数会分别替换 fmt 中的%x 等。

处理过程：

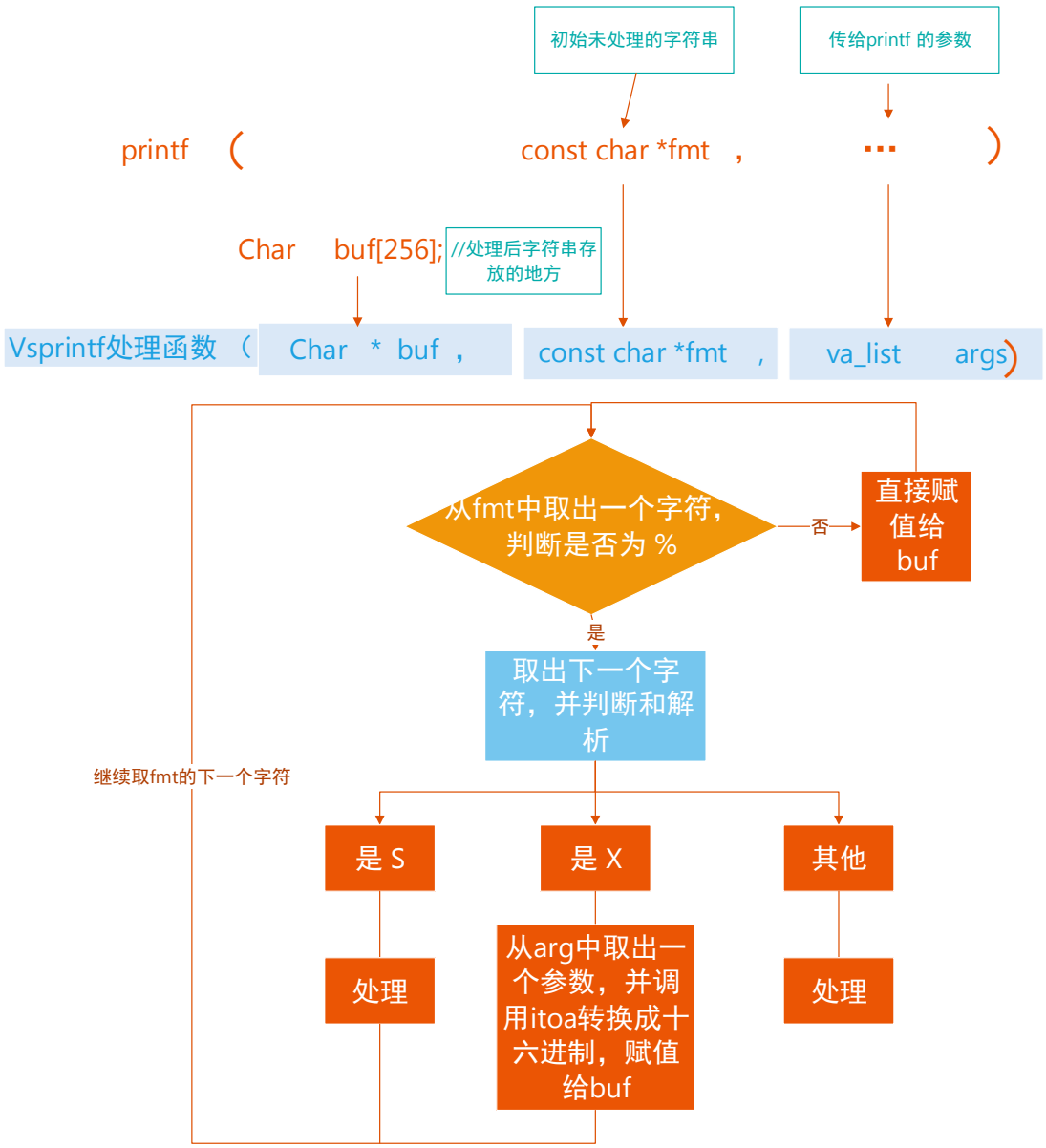
不断地扫描 fmt ，如果不是 % ，则直接赋给 buf ；如果是 % 的话，那么就看他的下一位是什么，是 % 的话，说明只是 % 的一个转义符，直接赋给 buf 。否知，如果是非零的数字，那么说明是一个 宽度说明 ，就必须计算宽度的长度。之后肯定就是控制格式符号 c，s，x 等了。在这些控制格式符中，d 和 x 由于存在进制的转换，所以要有相应的函数来处理进制转换，itoa() 就实现了这一个功能。

三、 程序关系图或流程图

键盘的构造及工作流程：



Printf 处理过程：

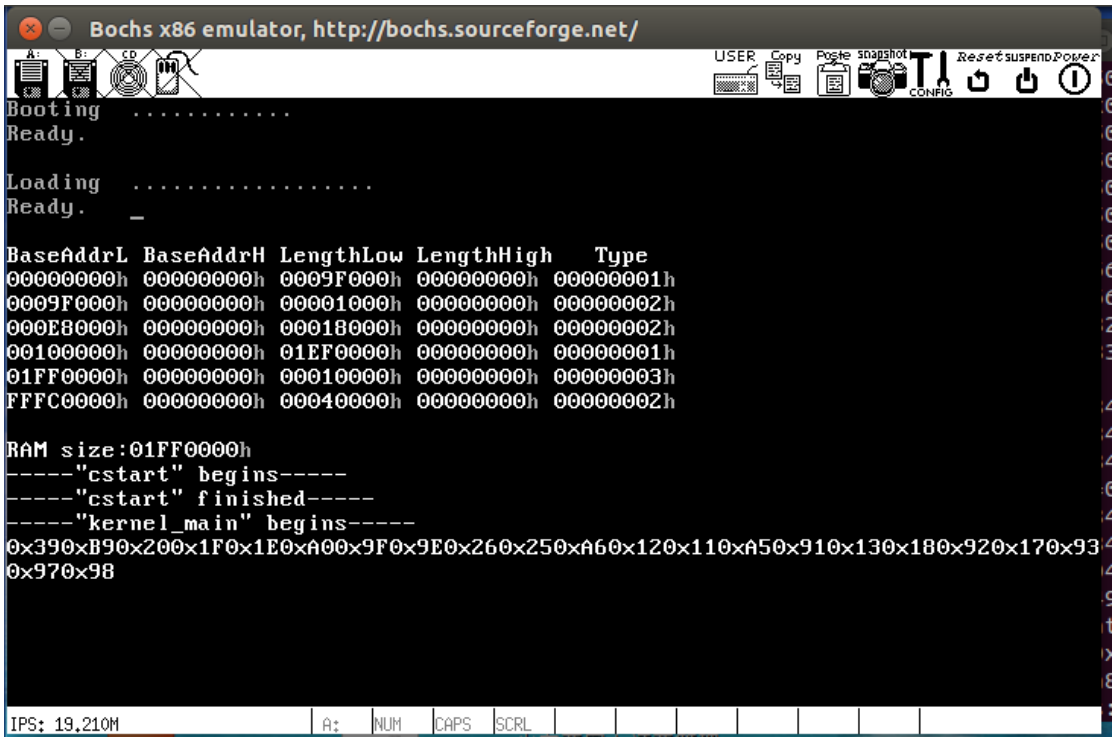


Return buf的字节长度

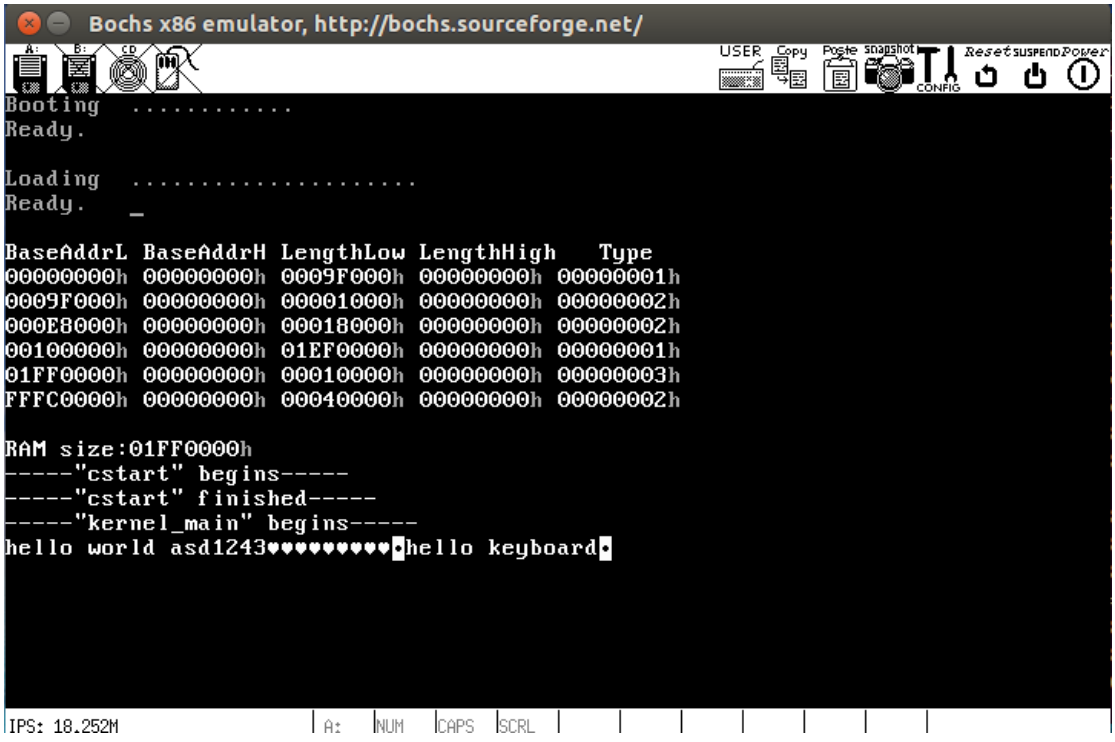
返回printf 函数， 将buf写进TTY

四、 运行过程及理解

7.c 按键输出扫描码

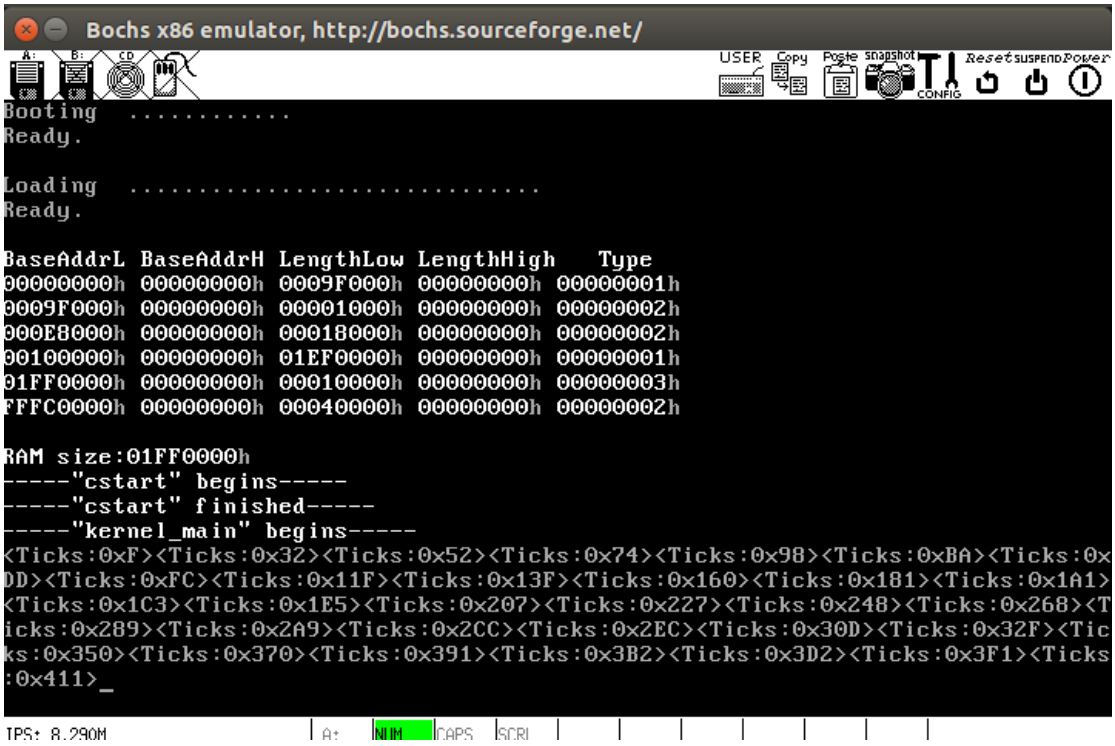


7.d 第一次处理扫描码，显示按键



7.h 翻页且光标跟随字符移动

7.o 系统调用 printf



五、 重点知识总结