# 第三部分 设备管理（任务十七—十八）

## 任务十七 字符显示的控制

### 1.实验目的

加深对操作系统设备管理基本原理的认识，掌握键盘中断、扫描码等概念；

掌握Linux 对键盘设备和显示器终端的处理过程。

### 2.实验内容

字符显示的控制

### 3.实验步骤

**3.1 准备实验**

使用浏览器登录平台领取本次实验对应的任务，从而在平台上创建个人项目（Linux 0.11 内核项目），然后使用VSCode将个人项目克隆到本地磁盘中并打开。

**3.2 字符显示的控制**

①将kernel/chr\_drv/keyboard.s文件中的第 274行注释掉。

②在include/asm/system.h文件的第三行前添加一个全局变量的声明：  
extern int judge;  
此变量是用作判断标志。初始值为0，当F12键被按下时judge被设置为1，F12键再次被按下时，judge又被设置为0。如此循环。

③在kernel/chr\_drv/tty\_io.c文件的第61行添加judge的定义：  
int judge = 0;

④并修改copy\_to\_cooked函数，在第196行GETCH(tty->read\_q,c);语句的后面添加如下代码：

if(c=='L')

{

if(judge)judge=0;

else judge=1;

break;

}

这样，在取到字符后，如果字符为“L”，表示F12被按下（按键F12的接通码会转换为ASCII码76，即字母“L”）。

⑤修改kernel/chr\_drv/console.c文件中的con\_write函数,在第615行后面添加如下代码, 如果字符是英文字母或者阿拉伯数字，就将之替换为“\*”号：

if(judge)

{

if((c>='a'&&c<='z')||(c>='A'&&c<='Z')||(c>='0'&&c<='9'))

c=42;

}

⑥生成项目，确保没有语法错误。

⑦然后按F5启动调试。

⑧执行ls命令，可以正常看到当前目录下的所有文件。

⑨按F12，此时再执行ls命令，可以看到，所有的英文字符和阿拉伯数字全都变成了“\*”。再按F12，再执行ls命令，可以看到，一切又回归到正常。

⑩修改步骤3中的源代码,将字母“L”换成其他字母（F1-F12分别对应‘A’-‘L’），重新调试，试试效果。

**3.3 提交作业**

实验结束后先使用VSCode左侧的“源代码版本控制窗口”查看文件变更详情，确认无误后再将本地项目提交到平台的个人项目中。

### 4.运行结果

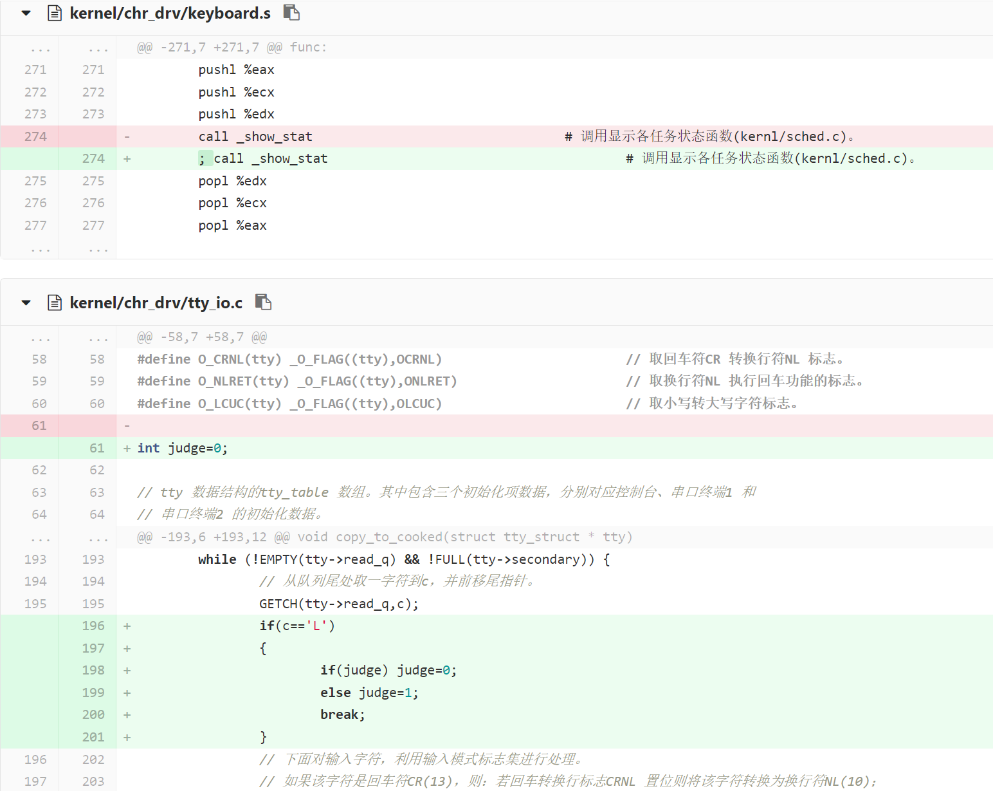
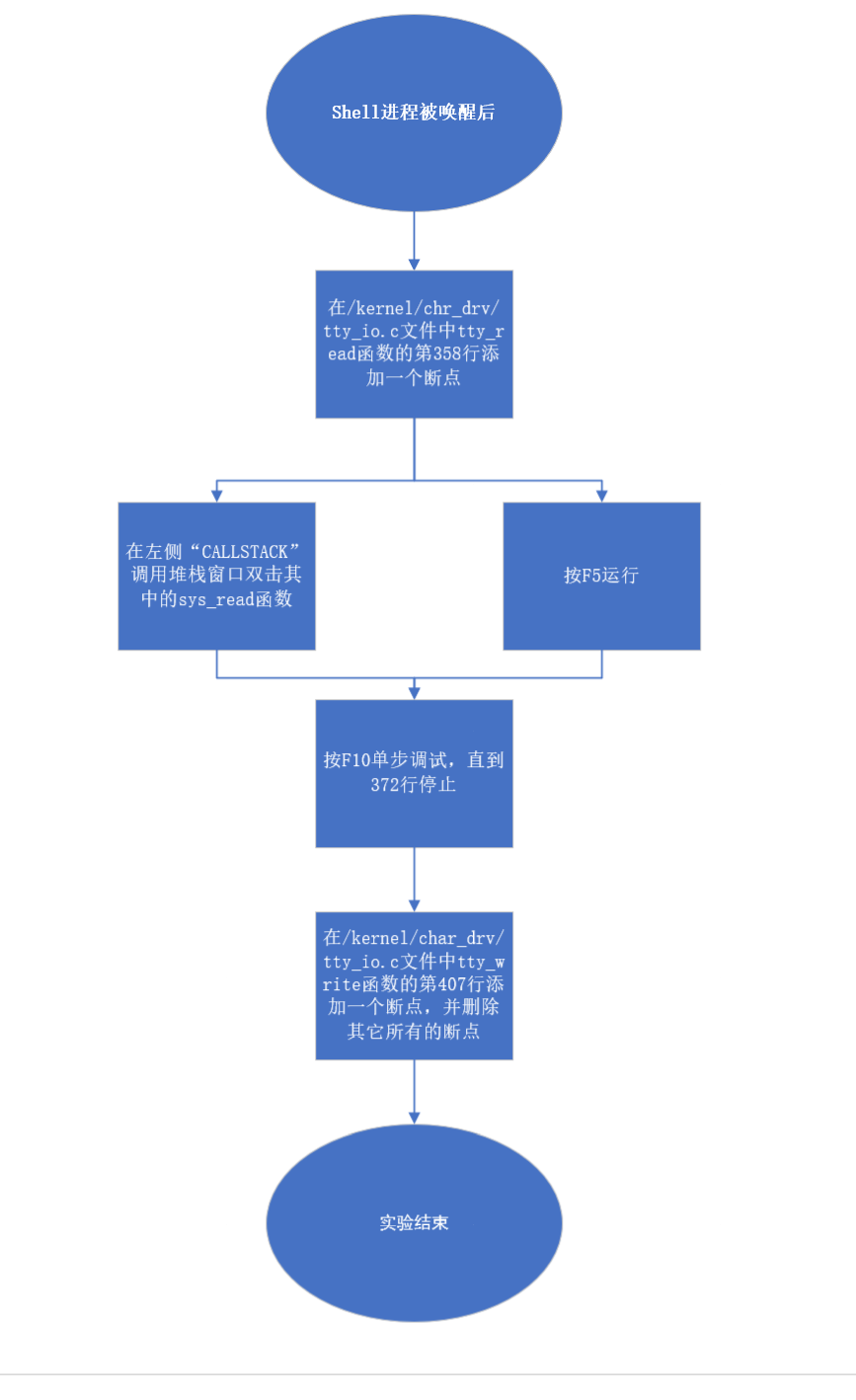
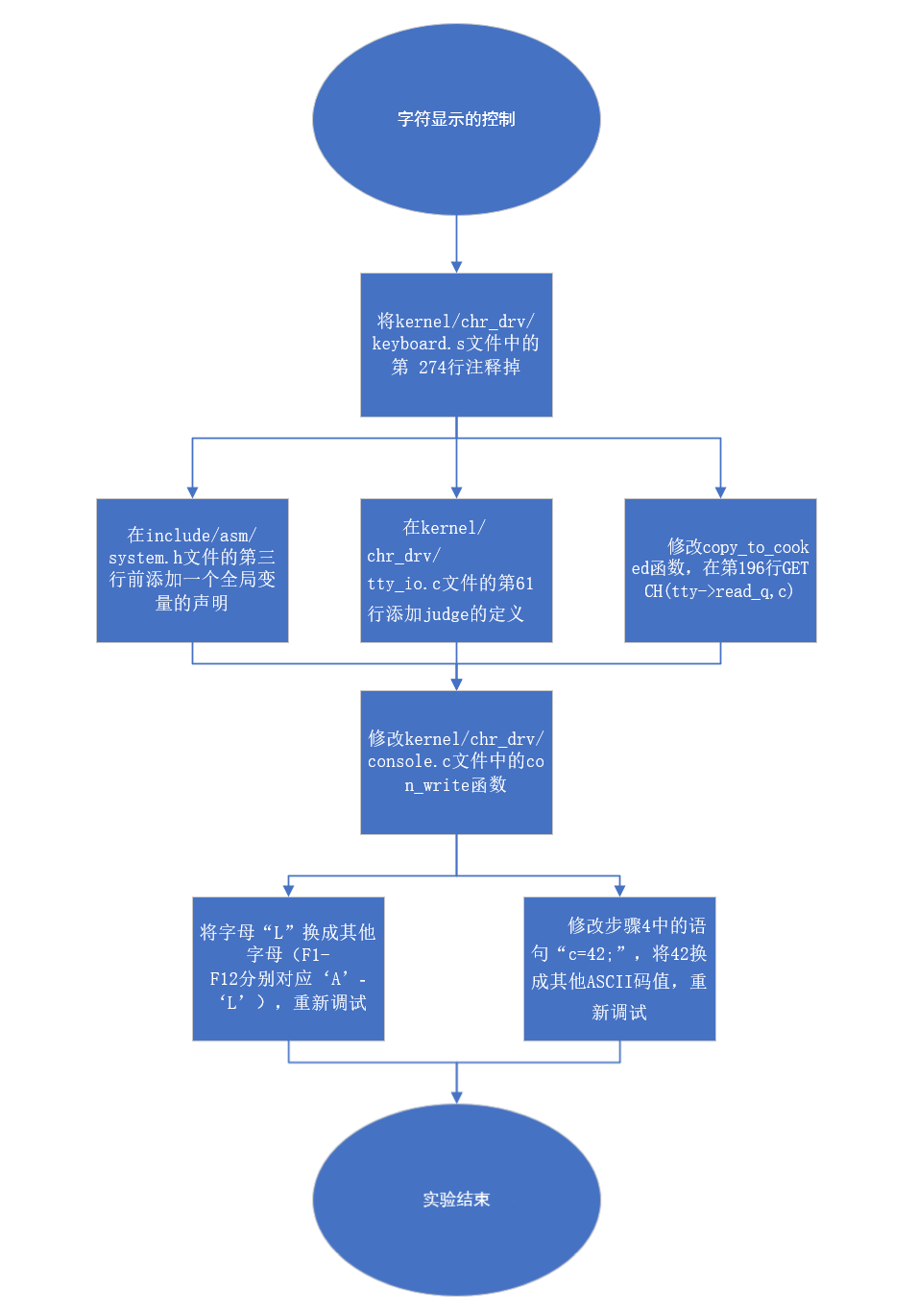
共修改4个文件

图54

### 5.流程图



图55

### 6.实验体会

通过本次实验，我深刻认识到设备管理是操作系统的一个重要功能之一。设备管理的主要任务是对计算机硬件设备进行管理，包括对输入设备（如键盘、鼠标等）和输出设备（如显示器、打印机等）的管理。

### 7.思考与练习

无

## 任务十八 实现贪吃蛇游戏

### 1.实验目的

加深对操作系统设备管理基本原理的认识，掌握键盘中断、扫描码等概念；

掌握Linux 对键盘设备和显示器终端的处理过程。

### 2.实验内容

实现贪吃蛇游戏

### 3.实验步骤

**3.1 准备实验**

使用浏览器登录平台领取本次实验对应的任务，从而在平台上创建个人项目（Linux 0.11 内核项目），然后使用VSCode将个人项目克隆到本地磁盘中并打开。

**3.2按照下面的步骤完成实验**

①在文件include/asm/system.h的第三行添加一个全局变量的声明：  
extern int fflag;  
此变量是用作判断标志。初始值为0，表示贪吃蛇游戏未启动，当“q”键被按下时设置为1，表示贪吃蛇游戏启动。

②在文件kernel/chr\_drv/tty\_io.c中的第62行添加全局变量的定义：  
int fflag = 0;

③并在第195行的后面(在copy\_to\_cooked函数中“GETCH(tty->read\_q,c);”语句的后面)添加如下代码。其作用是当q键被按下时，将标志设置为1，启动贪吃蛇游戏。“w、s、a、d”键分别代表上下左右四个方向,分别将标志设置为2、3、4、5。

if(c=='q') //q启动游戏

{

fflag = 1;

PUTCH(c,tty->secondary);

break;

}

if(c == 'w') //w上

{

fflag = 2;

PUTCH(c,tty->secondary);

break;

}

if(c == 's')//s下

{

fflag = 3;

PUTCH(c,tty->secondary);

break;

}

if(c == 'a')//a左

{

fflag = 4;

PUTCH(c,tty->secondary);

break;

}

if(c == 'd')//d右

{

fflag = 5;

PUTCH(c,tty->secondary);

break;

}

④在kernel/chr\_drv/console.c文件的第598行后面(con\_write函数的前面)添加如下代码。函数snake\_move将蛇头字符设置在显示内存中由 pos 指定的位置，从而显示在屏幕上。snake\_stop函数中使用全局变量jiffies完成一个计时循环(jiffies可参考实验六中的说明)，从而让贪吃蛇停止一定的时间。

//将蛇头字符设置在显示内存中由 pos 指定的位置，从而显示在屏幕上

void snake\_move()

{

\_\_asm\_\_("movb \_attr,%%ah\n\t"

"movw %%ax,%1\n\t"

::"a" ('+'),"m" (\*(short \*)pos)

);

}

//让贪吃蛇停留一定的时间

void snake\_stop()

{

int i;

for(i = jiffies + 50; jiffies <= i;)

;

}

⑤在kernel/chr\_drv/console.c文件的第634行后面(con\_write函数中“GETCH(tty->write\_q,c);”语句的后面，第一个switch语句的前面)添加如下代码，根据标志位来启动游戏，并且让贪吃蛇向指定的方向移动。需要说明的是，向右移动是通过一个while循环来实现不停地向右移动的。

switch(fflag)

{

case 0:break;

case 1: //初始化

csi\_J(2); //清屏

x = 0; y = 0;

gotoxy(x,y); //将当前光标设置为屏幕左上角的坐标

set\_cursor(); //将光标移动到屏幕左上角

case 5: //向右移动

while(fflag == 5 || fflag == 1)

{

delete\_line(); //删除光标所在的行

x < video\_num\_columns-1 ? x++ : x; //光标坐标向右移动一位

gotoxy(x,y); //更新光标位置

set\_cursor(); //设置显示器光标的位置

snake\_move(); //蛇向右移动

snake\_stop(); //停留

}

break;

case 3://向下移动

while(fflag == 3)

{

delete\_line(); //删除光标所在的行

y < video\_num\_lines-1 ? y++ : y; //光标坐标向下移动一位

gotoxy(x,y); //更新光标位置

set\_cursor(); //设置显示器光标的位置

snake\_move(); //蛇向下移动

snake\_stop(); //停留

}

break;

}

if(fflag != 0)

{

continue;

}

⑥生成项目，确保没有语法错误。

⑦按F5启动调试。

⑧按“q”启动贪吃蛇游戏。游戏启动后，贪吃蛇从屏幕的左上角出现并自动向右移动。此时可以按“s”将贪吃蛇的移动方向改为向下。再按“d”即可将贪吃蛇的移动方向改为向右。

**3.3 提交作业**

实验结束后先使用VSCode左侧的“源代码版本控制窗口”查看文件变更详情，确认无误后再将本地项目提交到平台的个人项目中，方便教师通过平台查看读者提交的作业。

### 4.运行结果

### 5.流程图

图56

### 6.实验体会

在深化我的学习体验过程中，我系统地研究了操作系统设备管理的基础机制，并通过广泛的文献阅读和实际操作，我深化了对键盘中断和扫描码等核心概念的理解。同时，我也掌握了Linux系统对键盘设备和显示器终端的高效管理方式。

在字符控制输入的学习上，我通过一系列练习熟练掌握了输入输出字符的基本流程，这让我更加明确了字符控制在Linux系统中的核心地位。这种实践性的学习让我深刻感受到，理论学习与实践操作的结合对于深化理解至关重要。

特别是在最后的项目实践中，我致力于实现贪吃蛇游戏的部分功能，通过不断的尝试和调整，我体验到了边学边做的乐趣。这一过程不仅增强了我的实践能力，也让我更加坚信，在计算机学习中，动手实践是理解和掌握知识的最佳途径。通过实际操作，我能够更直接地发现问题、解决问题，从而加深对知识点的理解和记忆。

### 7.思考与练习

目前只实现了将贪吃蛇向右移动和向下移动的基础功能，请读者在充分理解之前添加的代码的基础上，为贪吃蛇游戏添加下面的功能，使其具有一定的可玩性：

a) 添加贪吃蛇向上和向左移动的功能。

b) 当蛇头“+”移动到屏幕的边缘时，就会在与之相反的边缘出现，继续同方向移动。

c) 在屏幕上的某些位置出现“#”字符，当蛇头“+”与“#”相遇后，“#”消失，并且在另外一个位置再出现一个“#”。同时，蛇的尾部就多出一个“\*”，作为蛇身，吃的“#”越多，蛇身就越长。

d) 蛇身越长，贪吃蛇移动的速度越快。

e) 当蛇头“+”撞到蛇身“\*”后，结束游戏。

f) 同时出现2个贪吃蛇，实现双人对战，甚至多人对战。

（a）实现向上和向左移动的功能，仿照向下和向右移动的代码，将光标坐标和蛇的移动方向改为向上或向左。

（b）贪吃蛇从边缘出现的功能，在移动蛇头的时候判断是否到达了屏幕的边缘，如果是，则将蛇头的坐标设置为相反边缘的坐标。

（c）出现“#”字符的功能，在屏幕的某些位置随机生成“#”字符，并在蛇头移动的时候判断是否与“#”字符相遇，如果是，则将“#”字符删除，并在另外一个位置生成一个新的“#”字符，并在蛇的尾部添加一个“\*”字符，表示蛇身变长。

（d）蛇身越长速度越快的功能，在移动蛇头的时候，根据蛇身的长度来调整移动的速度，蛇身越长，速度越快。

（e）蛇头撞到蛇身结束游戏的功能，在移动蛇头的时候判断是否与蛇身相遇，如果是，则游戏结束。

（f）双人对战或多人对战的功能，在程序中添加多个贪吃蛇，并在移动蛇头的时候分别判断每个贪吃蛇的移动方向和坐标。