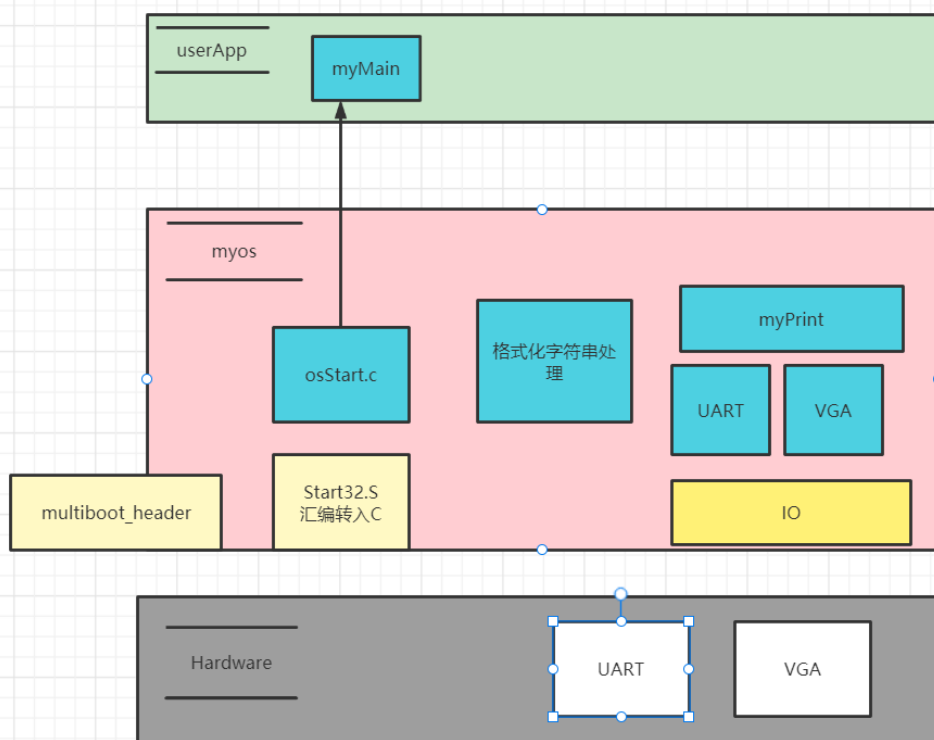


lab2实验报告

一、软件框图



二、主流程

1.流程图



2.文字说明

计算机开机之后，经过一些初始化步骤后，首先读取multibootheader文件，在校验对接完成之后，结尾处将借助myOS提供的_start 入口跳入myOS中。

进入myOS中首先执行由汇编语言构成的start.S文件，这个文件会进行设置栈和将BSS 段(存放已定义但没有复制的全局变量)清0两个操作，为下面的C语言执行提供必要的环境。执行完这两个操作后已经具备运行C语言编写的程序，在文件末尾通过osStart.c提供的入口跳入该.c文件中。

进入osStart.c中调用库函数进行vga清屏和将开始信息输出到vga，之后便跳入userApp执行main.c函数。

mian.c函数通过调用os提供的函数接口进行字符串输出。运行完毕，回到OS打印结束信息。

三、主要功能模块

1.vga.c的实现

先设置一个结构体

```
struct VGA{
    int16* head;
    int16* tail;
    int16 raw;
    int16 column;
    int16* cursor;
}vga = {(int16*)VGA_BASE, (int16*)(VGA_BASE + 2 * VGA_SCREEN_WIDTH *
VGA_SCREEN_HEIGHT), (int16)VGA_SCREEN_WIDTH, (int16)VGA_SCREEN_HEIGHT,
(int16*)VGA_BASE};
```

head, tail分别指向显存起始地址和末尾地址，raw与column分别表示vga的长和宽。cursor指向光标当前所在位置。之所以设置为int16也就是short int 类型是为了读写方便（一个字符占两个字节），cursor的起始地址是0。

当append2screen收到一个字符串后会调用put_char2pos对字符进行逐个输出。分为三种情况，屏满时调用scroll_screen()函数，将整体数据向上迁移一行，原本第一行丢弃，最后一行清0。不屏满遇到"\n"，通过计算将光标置到下一行开头。代码如下：

```
vga.cursor = vga.head + ((vga.cursor - vga.head)/vga.raw + 1) * vga.raw;
```

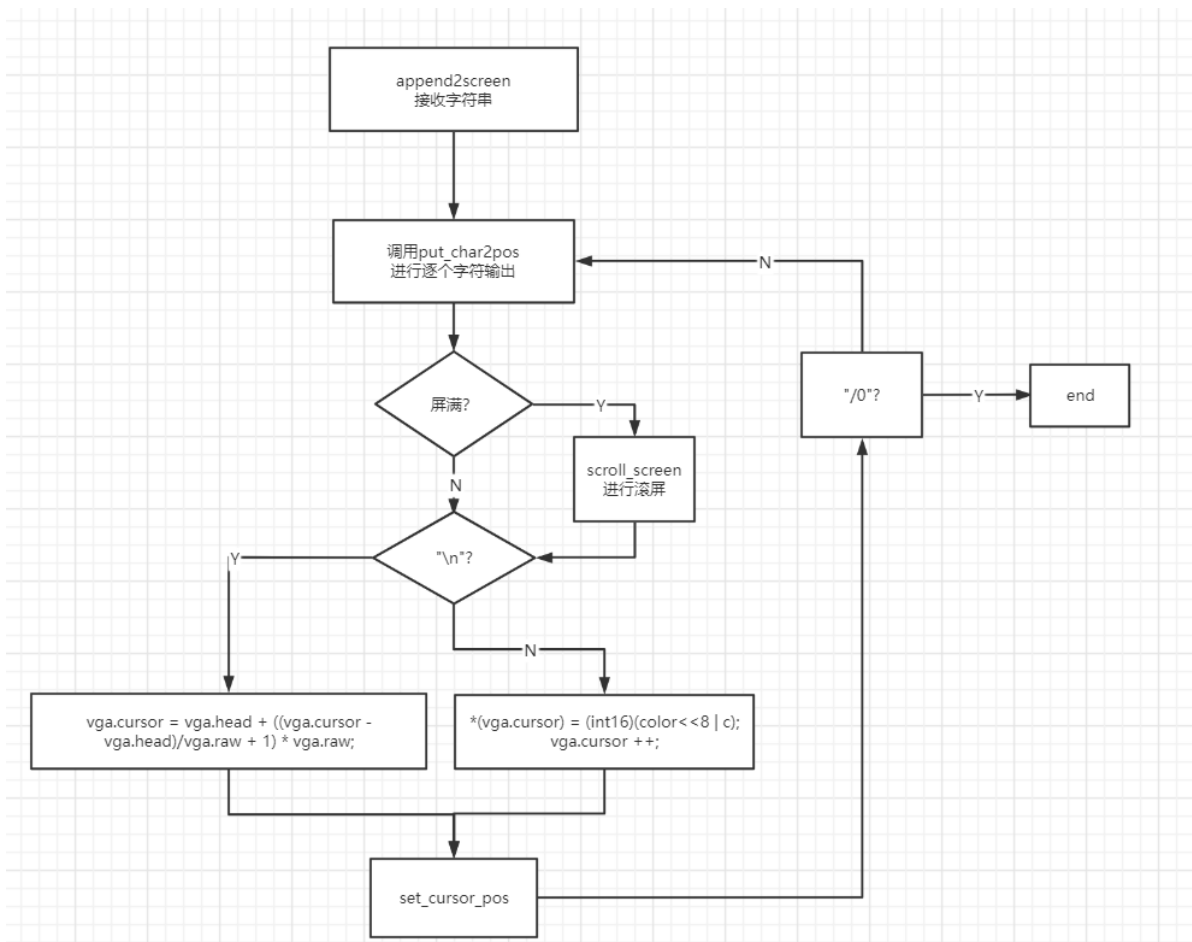
若不是"\n"，则在当前光标位置输出字符串，光标移动到下一位。

```
*(vga.cursor) = (int16)(color<<8 | c);
vga.cursor ++;
```

每次输出完后要调用set_cursor_pos 将光标真正设置到对应地址。

清屏函数clear_screen的实现就是让光标从头开始每次都输出ASCII的00，即空操作。

```
for(p = vga.head; p < vga.tail; p++){
    *p = 0x0700;
}
```



2. uart.c的实现

调用outb(), inb()即可简单实现。

```

void uart_put_chars(char *str) {
    int i;
    for(i = 0; str[i] != '\0'; i++){
        if(str[i] == '\n'){
            outb(UART_PORT, '\n');
        }
        else{
            outb(UART_PORT, str[i]);
        }
    }
}

```

3. vsprintf.c的实现

在C库中找到相应的vsprintf.c, 查看其要调用的外部函数, 一并移植。涉及到的函数大多属于string.c, 还有一些输出要调用math.c的函数。

4. myPrintk.c的实现

将上述的模块完成后, 该文件调用它们的函数即可实现。

```

int myPrintk(int color, const char *format, ...) {
    va_list args;

    va_start(args, format);
    int cnt = vsprintf(kBuf, format, args);
    va_end(args);

    /*利用串口进行输出*/
    uart_put_chars(kBuf);
    /*利用VGA进行输出*/
    append2screen(kBuf,color);
    return cnt;
}

```

四、源代码说明

1.目录组织如下

```

$ tree
.
|-- Makefile
|-- multibootheader
|   `-- multibootHeader.S
|-- myOS
|   |-- Makefile
|   |-- dev
|   |   |-- Makefile
|   |   |-- uart.c
|   |   `-- vga.c
|   |-- i386
|   |   |-- Makefile
|   |   `-- io.c
|   |-- include
|   |   |-- io.h
|   |   |-- myPrintk.h
|   |   |-- uart.h
|   |   |-- vga.h
|   |   `-- vsprintf.h
|   |-- lib
|   |   |-- Makefile
|   |   `-- vsprintf.c
|   |-- myOS.ld
|   |-- osStart.c
|   |-- printk
|   |   |-- Makefile
|   |   `-- myPrintk.c
|   |-- start32.S
|   `-- userInterface.h
|-- source2img.sh
`-- userApp
    |-- Makefile
    `-- main.c

```

分为三个主要部分，multibootheader，myOS，userApp。与启动头，OS，用户程序——对应。

myOS中有与设备相关的dev文件夹，库文件夹lib，给上层用户提供的接口userInterface.h。

2.Makefile组织

```
.
|-- MULTI_BOOT_HEADER
|   |-- output/multibootheader/multibootHeader.o
|
|-- OS_OBJS
|   |-- MYOS_OBJS
|       |-- output/myOS/start32.o
|       |-- output/myOS/osStart.o
|       |-- DEV_OBJS
|           |-- output/myOS/dev/uart.o
|           |-- output/myOS/dev/vga.o
|       |-- I386_OBJS
|           |-- output/myOS/i386/io.o
|       |-- PRINTK_OBJS
|           |-- output/myOS/printk/myPrintk.o
|       |-- LIB_OBJS
|           |-- output/myOS/lib/vsprintf.o
|
|-- USER_APP_OBJS
|   |-- output/userApp/main.o
```

a) myOS部分

在myOS下的每个子文件夹均含有一个Makefile文件，负责描述要生成的目标文件。

描述语言格式如下：

```
xxx_OBJS = 要输出的目录/xxx.o\
           下一个要生成的目标文件
```

在myOS本层还有一个总的Makefile文件，来链接子文件夹的Makefile，统一描述要生成目标文件。

链接语言如下：

```
include $(SRC_RT)/myOS/子文件名称/Makefile
```

b) userAPP

userApp的Makefile布局与myOS原理一致

c) 最外层

到最外层有一个总Makefile文件，用于统一链接每个Makefile，统一生成目标文件。

五、代码布局说明

地址空间排布如下：

1. 从1M的内存地址开始，首先放multiboot_header，写入12个字节。
2. 要求8字节对齐，.text代码从16字节开始。

3. 放完.text后, 进行16字节对齐。接着放入OS代码, 即.data数据。
4. 再次进行16字节对齐后为全局变量分配空间。
5. 16字节对齐后, 最后为栈分配空间。进行512字节对齐。

对VGA显存部分, 每两个字节间放一个字符, 第一个放到0xB8000,第二个从0xB8002开始。

六、编译过程说明

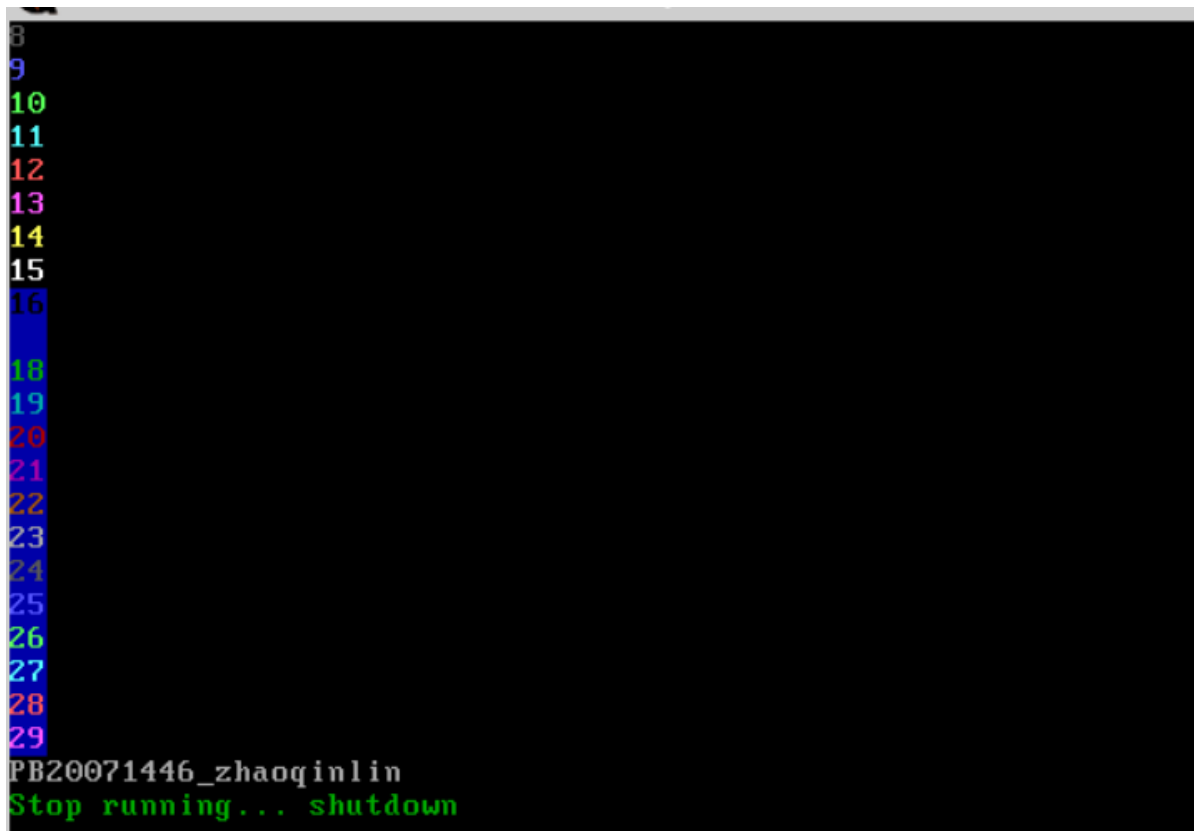
将所有文件完成后, 在linux环境中打开命令行终端, 输入./source2img.sh 一键编译运行。

而背后的大致流程为: 编译各个文件生成相应的.o文件, 再根据链接描述文件将各个.o文件进行链接, 生成myOS.elf文件。

七、运行结果

在linux环境中打开命令行终端, 输入./source2img.sh

1.VGA



```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
PB20071446_zhaoqinlin
Stop running... shutdown
```

2.UART

```
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
PB20071446_zhaoqinlin
Stop running... shutdown
```

八、遇到的问题与解决方法

问题：第一次利用vga显存输出时只能输出半个屏幕

解决方法：末尾地址计算错误，将其乘2后即可正常运行。