



中国科学技术大学

University of Science and Technology of China

011174.01: Operating System 操作系统原理与设计

Project 2: Multiboot2myMain

陈香兰(xlanchen@ustc.edu.cn)

高效能智能计算实验室, CS, USTC @ 合肥

嵌入式系统实验室, CS, USTC @ 苏州

实验2基础



中国科学技术大学
University of Science and Technology of China

- 本实验在实验1的基础上进行
- 在实验一提交的截止时间过后，同学们可以就实验一的内容互通有无
- 实验二可以在其他同学实验一的基础上进行
 - 无论你使用哪一个（包括自己的），请在实验报告中标注，实验二的基础来自哪个同学（可以是自己）
 - 给你使用的实验一 打分

- **【必须】** 在源代码的语言层面，完成从汇编语言到C语言的衔接
- **【必须】** 在功能上，实现清屏、格式化输入输出，设备包括VGA和串口，接口符合要求
- **【必须】** 在软件层次和结构上，完成multiboot_header、myOS和userApp的划分，体现在文件目录组织和Makefile组织上
- **【必须】** 采用自定义测试用例和用户（助教）测试用例相结合的方式验收
- **【必须】** 提供脚本完成编译和执行

提纲



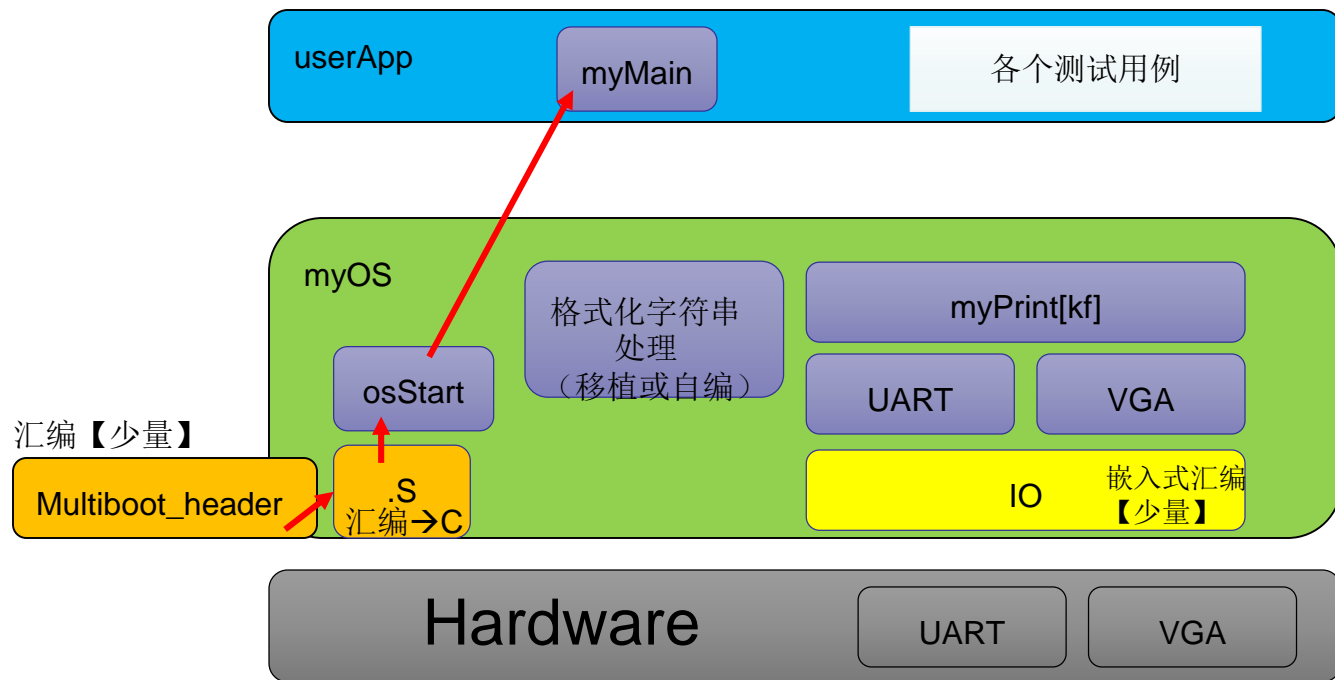
中国科学技术大学
University of Science and Technology of China

1. 软件架构和功能说明
2. 代码组织及其实现
3. 主流程实现
4. 主要功能模块实现
5. 验收标准

1 软件架构和功能



中国科学技术大学
University of Science and Technology of China



流程:

Multiboot_header

为进入C程序准备好上下文

初始化操作系统

调用userApp入口
myMain (自测)

multiboot_header → myOS → userApp

主要功能模块:

IO: 端口输入输出【嵌入式汇编】

UART: 串口输入

VGA: 清屏、屏幕彩色输出 (带滚屏)

格式化字符串处理, 支持常见格式化输出
彩色格式化输入输出 (内核版 VS 用户版)

同时从VGA和串口输出

测试:

被测功能: 彩色格式化输入输出 (带滚屏、常见格式化输出如%d)

测试方法: 自测+他测, 基于约定接口

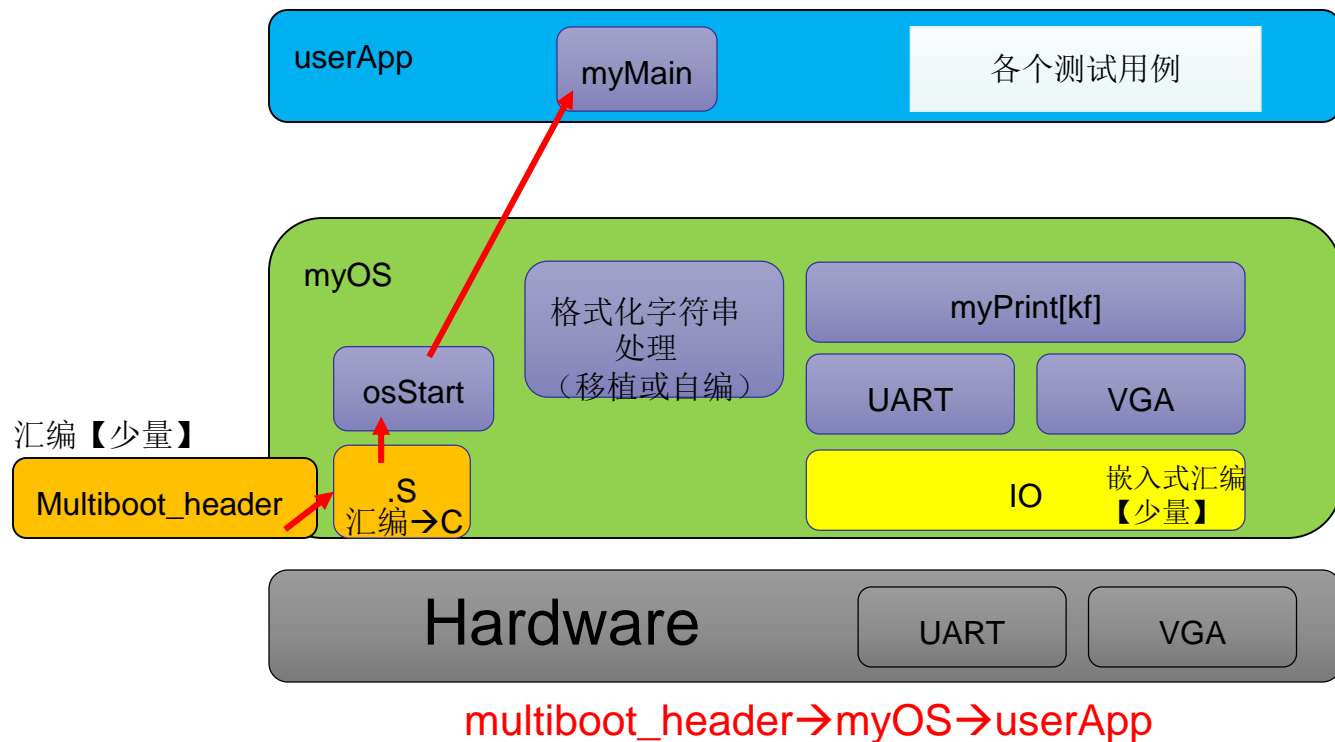
自测: **userApp**

他测: 替换**userApp**

2 代码组织及其实现



中国科学技术大学
University of Science and Technology of China



对源代码进行合适的源代码结构和Makefile组织建设

目录划分线索：层次、功能模块、内核代码和用户代码分开管理

基本上每个目录下有一个Makefile

目录结构示意



中国科学技术大学
University of Science and Technology of China

- 为Project2建立一个子目录，该目录下
 - **Makefile** 提供，可修改
 - Multibootheader子目录
 - 内核子目录 子目录下可以进一步按功能划分子目录
 - userApp子目录
 - main.c
 - output子目录（所有编译链接生成的文件在此）
 - **source2run.sh** 脚本文件，提供，不要修改

Makefile系列



中国科学技术大学
University of Science and Technology of China

```
SRC_RT=$(shell pwd)
```

```
CROSS_COMPILE=
```

```
ASM_FLAGS= -m32 --pipe -Wall -fasm -g -O1 -fno-stack-protector
```

```
C_FLAGS = -m32 -fno-stack-protector -g
```

```
.PHONY: all
```

```
all: output/myOS.elf
```

```
MULTI_BOOT_HEADER=output/multibootheader/multibootHeader.o
```

```
include $(SRC_RT)/myOS/Makefile
```

```
include $(SRC_RT)/userApp/Makefile
```

```
OS_OBJS    = ${MYOS_OBJS} ${USER_APP_OBJS}
```

```
output/myOS.elf: ${OS_OBJS} ${MULTI_BOOT_HEADER}
```

```
    ${CROSS_COMPILE}ld -n -T myOS/myOS.ld ${MULTI_BOOT_HEADER} ${OS_OBJS} -o output/myOS.elf
```

```
output/%.o : %.S
```

```
    @mkdir -p $(dir $@)
```

```
    @${CROSS_COMPILE}gcc ${ASM_FLAGS} -c -o $@ $<
```

```
output/%.o : %.c
```

```
    @mkdir -p $(dir $@)
```

```
    @${CROSS_COMPILE}gcc ${C_FLAGS} -c -o $@ $<
```

```
clean:
```

```
    rm -rf output
```

```
include $(SRC_RT)/myOS/dev/Makefile
include $(SRC_RT)/myOS/i386/Makefile
include $(SRC_RT)/myOS/printk/Makefile
```

```
MYOS_OBJS =  output/myOS/start32.o \
             ${DEV_OBJS} \
             ${I386_OBJS} \
             ${PRINTK_OBJS}
```

```
USER_APP_OBJS = output/userApp/main.o
```

参考GNU/make的文档:

<https://www.gnu.org/software/make/manual/make.html>

链接描述文件



中国科学技术大学
University of Science and Technology of China

```
SECTIONS {  
    . = 1M;  
    .text : {  
        *(.multiboot_header)  
        . = ALIGN(8);  
        *(.text)  
    }  
  
    . = ALIGN(16);  
    .data      : { *(.data*) }  
  
    . = ALIGN(16);  
    .bss      :  
    {  
        __bss_start = .;  
        _bss_start = .;  
        *(.bss)  
        __bss_end = .;  
    }  
    . = ALIGN(16);  
    __end = .;  
    . = ALIGN(512);  
}
```

参考GNU linker的手册:

<https://sourceware.org/binutils/docs-2.42/ld.html>

版本可能会发生变化, 可以从源头找:

GNU binutils的网页:

<https://www.gnu.org/software/binutils/>

What is BSS segment?

脚本source2run.sh



中国科学技术大学
University of Science and Technology of China

```
#!/bin/sh

make clean

make

if [ $? -ne 0 ]; then
    echo "make failed"
else
    echo "make succeed"
    qemu-system-i386 -kernel output/myOS.elf -serial stdio
fi
```

3 主流程实现



中国科学技术大学
University of Science and Technology of China

Multiboot_header

MyOS的汇编入口

MyOS的C入口
osStart

userApp的main入口
myMain

1. Multiboot_header 【略】
2. 从multiboot header到myOS
 - 分离Multiboot header和myOS
 - myOS提供_start入口
 - Multiboot header中调用_start入口
3. 第一次调用C语言入口前的准备
 - 思考：在执行C语言程序前要做哪些准备？
 - 提供参考代码
4. myOS→userApp

```
.text  
.code32  
start:  
    call _start  
    hlt
```

从汇编到C



中国科学技术大学
University of Science and Technology of China

Multiboot_header

MyOS的汇编入口

MyOS的C入口
osStart

userApp的main入口
myMain

- myOS从汇编开始，入口为_start

```
.text
.code32
_start:
    jmp establish_stack

dead: jmp dead                # Never here

# Set up the stack
establish_stack:
    movl $???????, %eax

    movl %eax, %esp          # set stack pointer
    movl %eax, %ebp          # set base pointer

# Zero out the BSS segment
zero_bss:
    cld                     # make direction flag count up
    movl $_end, %ecx         # find end of .bss
    movl $_bss_start, %edi   # edi = beginning of .bss
    subl %edi, %ecx          # ecx = size of .bss in bytes
    shr %ecx                 # size of .bss in longs
    shr %ecx
```

```
xorl %eax, %eax            # value to clear out memory
repne stosl                 # while ecx != 0
                             # clear a long in the bss
```

Transfer control to osStart (in C)

to_osStart:

```
    call osStart
```

shut_down:

```
    jmp shut_down          # Never here
```

设置初始栈在物理内存中的位置
确保栈的使用不会覆盖任何myOS相关的代码

操作系统第一个C入口



中国科学技术大学
University of Science and Technology of China

Multiboot_header

MyOS的汇编入口

MyOS的C入口
osStart

userApp的main入口
myMain

- 本实验以osStart为myOS的第一个C入口
 - 功能：先进行OS初始化；初始化完毕后，执行userApp
 - 可以调用myOS内部接口
 - 与userApp之间的接口是myMain

```
extern int myPrintk(int color, const char *format, ...);
extern void clear_screen(void);
extern void myMain(void);

void osStart(void){
    clear_screen();

    //操作系统各个功能模块的初始化

    myPrintk(0x2, "START RUNNING userApp.....\n");
    myMain();
    myPrintk(0x2, "STOP RUNNING userApp.....ShutDown\n");
    while(1);
}
```

用户程序入口



中国科学技术大学
University of Science and Technology of China

Multiboot_header

MyOS的汇编入口

MyOS的C入口
osStart

userApp的main入口
myMain

- 进入用户程序的接口
 - void myMain(void);
 - 在操作系统初始化完毕后，调用myMain，转入用户程序运行
- 用户程序只能调用操作系统定义的接口和用户自定义的函数
 - 例如：myPrintf
- 本实验中的功能：调用myPrintf输出某些内容
 - 【必须】输出内容：main, 学号_姓名拼音

```
extern int myPrintf(int color,const char *format, ...);
```

```
void myMain(void){  
    myPrintf(0x7,"main\n");  
    //.....  
    return;  
}
```

4 主要功能实现



中国科学技术大学
University of Science and Technology of China

1. IO：端口输出，采用嵌入式汇编【提供】

```
void outb (unsigned short int port_to, unsigned char value){  
    __asm__ __volatile__ ("outb %b0,%w1"::"a" (value),"Nd" (port_to));  
}
```

2. 串口uart输出（仍不要求初始化）

- 单个字符输出：void uart_put_char(unsigned char c)
- 字符串输出：void uart_put_chars(char *str)
 - 注意回车换行的处理：遇到 '\n' 实际输出 '\r'

3. VGA输出，管理好字符界面输出

- 清屏功能：void clear_screen(void)
- 屏幕输出功能：void append2screen(char *str, int color)
 - 能够显示光标，光标位置要正确
 - 提供合理的字符界面输出功能，屏满时能滚屏继续输出

4. 实现myPrint[kf]

关于光标



中国科学技术大学
University of Science and Technology of China

- 光标的位置可以读取，也可设置，由显卡相关端口上的读写操作完成
 - 首先通过索引端口指定要读写的寄存器，然后通过数据端口进行读写
 - 索引端口地址是：0x3D4 数据端口地址是：0x3D5
- 光标位置使用行号和列号来指明，行号和列号分别使用两个8位的寄存器来记录
 - 行号寄存器的索引号是14，即0xE；列号寄存器的索引号是15，即0xF
- 编程实现读取当前位置，或者设置新位置
 - 首先将上述某个索引号写入索引端口0x3D4，即可指定对应的寄存器
 - 然后读 数据端口0x3D5，就可以读取光标的行值/列值
 - 或者写 数据端口0x3D5，就可以设置光标的行值/列值

```
unsigned char inb(unsigned short int port_from){
    unsigned char _in_value;

    /* something for __asm__
       %w1: w=word
       read a byte from port %w1 to %0
       "=a": read-only, get value from eax/ax/al
       "d": set value to edx/dx/dl
    */
    __asm__ __volatile__ ("inb %w1,%0":"=a"(_in_value):"Nd"(port_from));

    return _in_value;
}
```


实现myPrint[kf]



中国科学技术大学
University of Science and Technology of China

```
int myPrintk(int color,const char *format, ...);  
int myPrintf(int color,const char *format, ...);
```

- 能够识别基本的格式化字符串 【移植/自编】
 - 满足当前【必须】和后续【非必须，可以后面改进】实验的输出要求
 - %d
- ```
int vsprintf(char *buf, const char *fmt, va_list args)
int sprintf(char *buf, const char *fmt, ...)
```
- 调用VGA和串口输出接口

- 提交：源代码打包 + 实验报告；验收标准如下：
  - 完成源代码编写和调试，能够编译正确运行
    - 实现主流程，提供规定接口
    - 实现主要功能，提供规定接口
    - 将源代码进行合理的组织、提供相应的Makefile，能够生成myOS
    - 提供编译和运行脚本
  - 提交实验报告，实验报告中包括
    - 给出软件的框图，并加以概述
    - 详细说明主流程及其实现，画出流程图
    - 详细说明主要功能模块及其实现，画出流程图
    - 源代码说明（目录组织、Makefile组织）
    - 代码布局说明（地址空间）
    - 编译过程说明
    - 运行和运行结果说明
    - 遇到的问题和解决方案说明

# 演示



中国科学技术大学  
University of Science and Technology of China

```
xlanchen@DESKTOP-9IQF7F8: ~/workspace/OSlab/lab2
xlanchen@DESKTOP-9IQF7F8:~$ cd workspace/OSlab/lab2
xlanchen@DESKTOP-9IQF7F8:~/workspace/OSlab/lab2$./source2run.sh
./source2run.sh: 2: shell: not found

rm -rf output
ld -n -T myOS/myOS.ld output/multibootheader/multibootHeader.o output/myOS/start32.o output/myOS/osStart.o output/myOS/dev/uart.o output/myOS/dev/vga.o output/myOS/i386/io.o output/myOS/printk/myPrintk.o output/userApp/main.o -o output/myOS.elf
make succeed
osStart(): START RUNNING.....
myMain is running:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Back to osStart(): STOP RUNNING.....ShutDown
```

```
QEMU
Machine View
osStart(): START RUNNING.....
myMain is running:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Back to osStart(): STOP RUNNING.....ShutDown
```