**中国科学技术大学计算机学院**

**《操作系统原理与设计》实验报告**



实验题目：lab2\_Multiboot2myMain

学生姓名：胡毅翔

学生学号：PB18000290

完成日期：2020年3月17日

计算机实验教学中心制

2019年09月

实验目的

1. 在源代码层面，实现从汇编语言到C语言的衔接。
2. 在功能上，实现清屏、格式化输入输出，I/O设备包括VGA和串口。
3. 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。

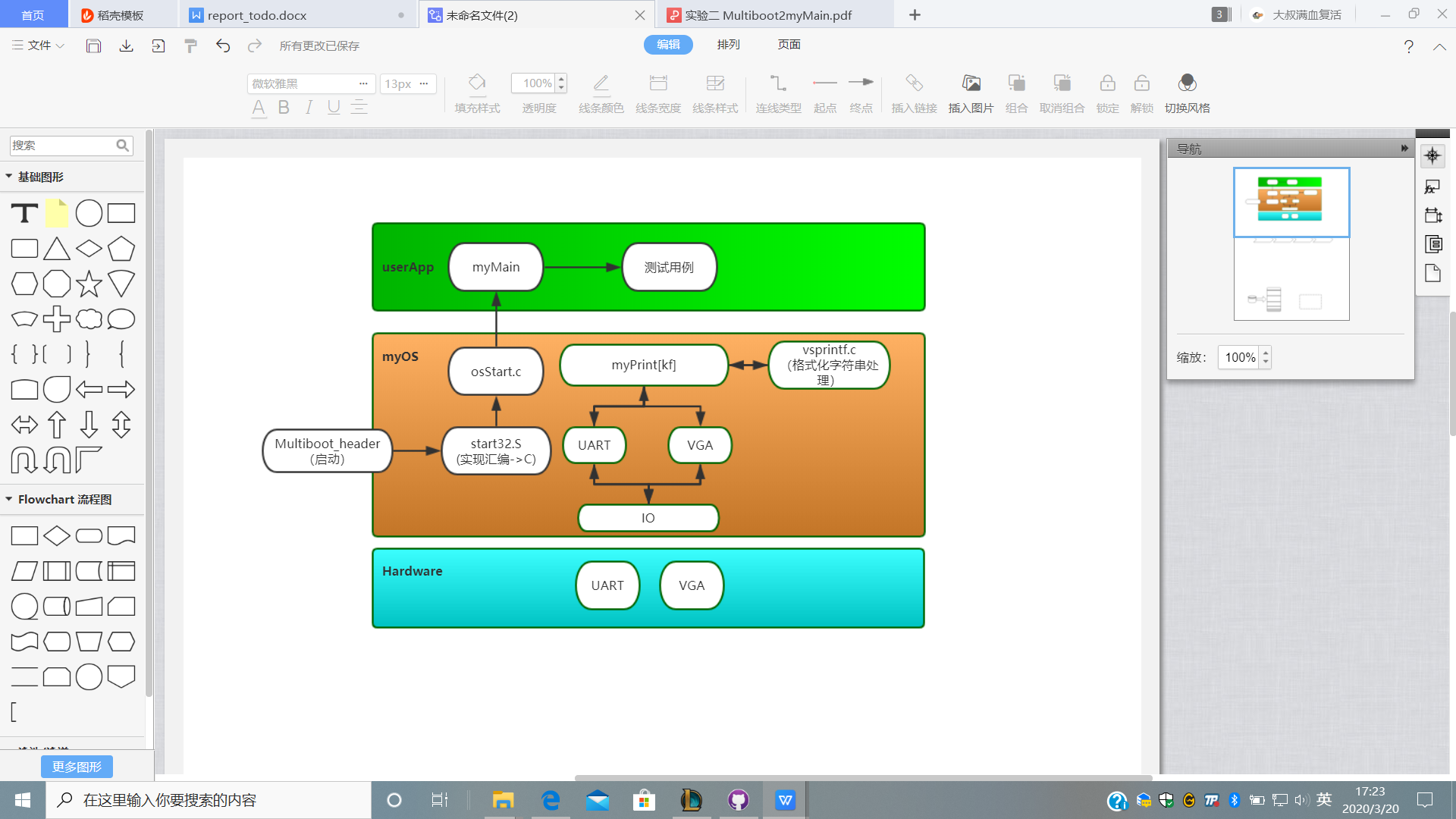
实验环境

1. PC一台
2. Windows系统
3. Ubuntu
4. QEMU
5. Xserver

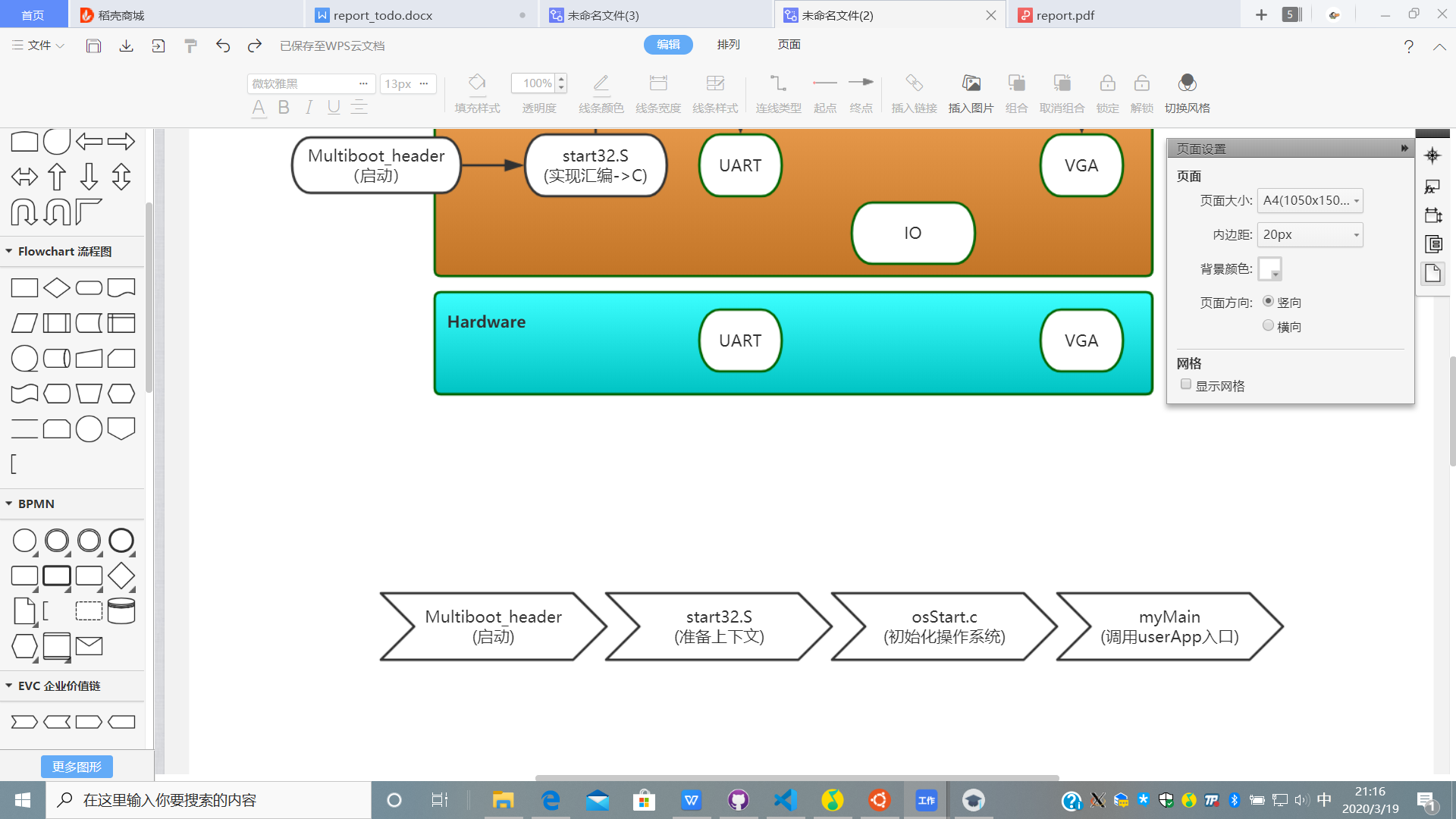
软件框图

本实验的软件框图如图1所示。软件层次分为multiboot\_header、myOS和userApp三部分。

Multiboot\_header为系统启动部分，系统启动后进入myOS，在osStart.c中调用myMain.c进入userApp部分。若userApp部分，myMain及其调用的程序需要输出字符串时，通过myPrintk/f，将经vsprintf处理后的字符串通过IO接口，在VGA和UART上完成输出。



主流程



本实验的主流程如上图所示：

1. 在multiboot\_header中完成系统的启动。
2. 在start32.S中准备好上下文，最后调用osStart.c把进入c程序。
3. 在osStart.c中完成清屏等初始化操作，调用myMain，进入userApp部分。
4. 运行myMain中的代码。

主要功能模块及其实现

IO

串口uart输出

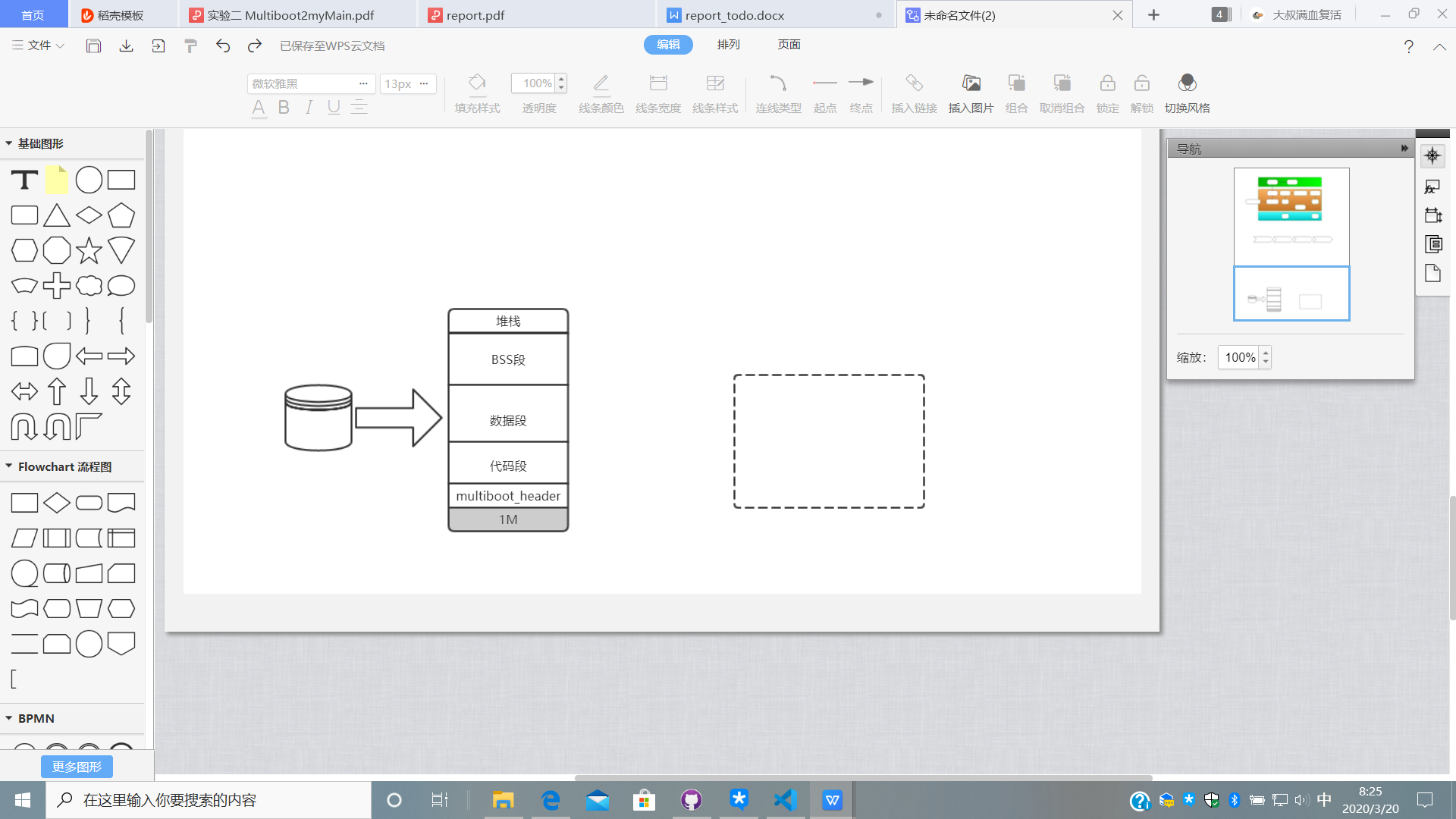
VGA输出

myPrint[kf]的实现

源代码说明

代码布局说明（地址空间）

从物理内存1M的位置开始放代码和数据，前面12个字节为multiboot\_header,向后对齐8个字节，放代码。再向后对齐16个字节，用于放初始化的数据（数据段）。在数据段之后，再向后对齐16个字节。之后为BSS（Block Started by Symbol）段,用于存放程序中未初始化的全局变量和静态变量。并在BSS段后，再向后对齐16个字节。剩余部分为堆栈段。



编译过程说明

ASM\_FLAGS= -m32 --pipe -Wall -fasm -g -O1 -fno-stack-protector

-m32用32位机器的编译器来编译这个文件

--pipe使用管道代替编译中临时文档

-Wall 打开警告选项

-fasm 识别asm关键字

-g 使用调试器GDB

-O1 优化生成代码

-fno-stack-protector停止使用stack-protector功能

C\_FLAGS =  -m32 -fno-stack-protector -fno-pic -fno-builtin -g

****-fpic**** 如果支持这种目标机,编译器就生成位置无关目标码.适用于共享库(shared library)

生成multiHeader.o、osStart.o、start32.o、uart.o、vga.o、io.o、myPrintk.o、vsprintf.o及main.o多个目标文件，链接生成myOS.elf文件

运行和运行结果说明

输入./source2run.sh指令后，编译，链接，生成myOS.elf文件并运行之。程序通过串口和VGA按mymain.c中的要求输出相应内容。运行结果如下图

遇到的问题和解决方案说明

1. 编译时出现报错“对‘\_GLOBAL\_OFFSET\_TABLE\_’未定义的引用”。

解决方案：在src目录下的Makefile文件的CFLAGS变量中添加-fno-pic.

1. 编译出现fatal error: bits/libc-header-start.h: No such file or directory

解决方案:在Ubuntu中输入apt-get install gcc-multilib，完善编译环境

1. 编译时出现warning: assignment makes pointer from integer without a cast

解决方案：在给指针赋值前进行强制格式转化(unsigned short int\*)

1. 编译时出现warning: conflicting types for built-in function

解决方案：在src目录下的Makefile文件的CFLAGS变量中添加-fno-builtin