

# 操作系统 实验报告一

21级软件工程一班 宁智伟 202131603131

实验题目：启用C语言

实验时间：2023. 11

实验内容：

在 start.bin 的基础上增加操作系统的基本功能，编写操作系统的高级语言首选 C 语言。解决 C 语言编程的基本问题，内容主要包括 gcc 简介，头文件和库文件，基本工具集以及 x86 汇编与 C 语言的混合编程问题。

## 1. gcc环境

在命令行测试gcc环境

```
C:\Users\32644>gcc --version
gcc (GCC) 12.2.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\Users\32644>|
```

gcc环境正常，版本为gcc（GCC）12.2.0

## 2. 实验步骤

1. 复制之前的myos3文件夹内容至myos4文件夹

名称	修改日期	类型	大小
A	2023/11/5 13:37	光盘映像文件	1 KB
mbr.asm	2023/11/5 13:37	ASM 文件	1 KB
mbr.bin	2023/11/5 13:37	BIN 文件	1 KB
start.asm	2023/11/5 13:33	ASM 文件	1 KB
start.bin	2023/11/5 13:36	BIN 文件	1 KB
writeA	2023/11/5 13:35	C 源文件	1 KB
writeA	2023/11/5 13:35	应用程序	82 KB
ex	2023/11/12 13:23	C 源文件	1 KB
ex	2023/11/12 13:23	应用程序	80 KB

2. 新建ex.c文件，写入c源代码

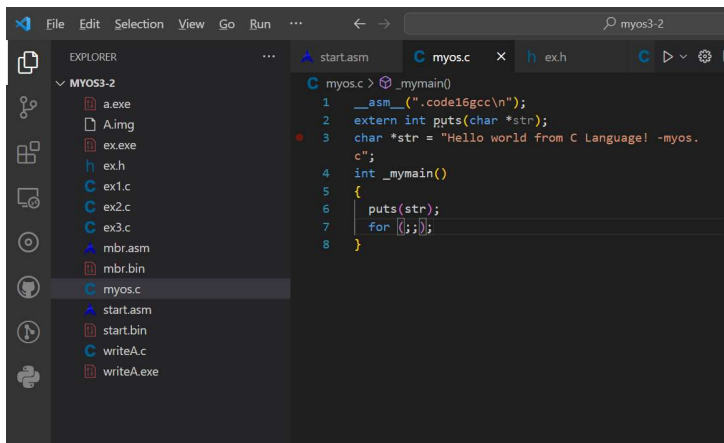
3. 编译ex.c，得到下图内容

4. 把函数声明放到头文件ex.h中

5. 把 myfunc1 函数的定义放到 C 语言源程序 ex1.c 中，如下所示。

6. 把 myfunc2 函数的定义放到 C 语言源程序 ex2.c 中，如下所示：





## 12. 汇编 start.asm

```
D:\exe\myos\myos3-2>a.exe
First function!
Second function!

D:\exe\myos\myos3-2>nasm -f elf start.asm -o start.o
start.asm:20: error: comma, colon, decorator or end of line expected after operand

D:\exe\myos\myos3-2>nasm -f elf start.asm -o start.o

D:\exe\myos\myos3-2>|
```

(这里第一次出现了汇编错误，原因是第一次修改后 `JMP __mymain` 语法错误)

```
D:\exe\myos\myos3-2>nasm -f elf start.asm -o start.o
start.asm:20: error: comma, colon, decorator or end of line expected after operand

D:\exe\myos\myos3-2>nasm -f elf start.asm -o start.o

D:\exe\myos\myos3-2>objdump -a start.o

start.o:      file format elf32-i386
start.o
```

## 13. 编译 myos.c。myos.c 只能用“-c”选项编译成目标文件。

```
D:\exe\myos\myos3-2>gcc myos.c -o myos.o
C:\Users\32644\AppData\Local\Temp\cckdy9CR.o:myos.c:(.text+0x39): undefined reference to `WinMain@16'
collect2.exe: error: ld returned 1 exit status

D:\exe\myos\myos3-2>gcc -c myos.c -o myos.o

D:\exe\myos\myos3-2>
```

## 14. 最后，把它们链接成一个可执行文件。

```
D:\exe\myos\myos3-2>gcc -c myos.c -o myos.o

D:\exe\myos\myos3-2>ld -s --entry=start -Ttext=0x0 start.o myos.o -o myos.exe
ld: myos.exe:.text: section below image base
ld: myos.exe:.data: section below image base
ld: myos.exe:.rdata: section below image base
ld: myos.exe:.idata: section below image base
ld: myos.exe:.reloc: section below image base

D:\exe\myos\myos3-2>objdump -a myos.exe

myos.exe:      file format pei-i386
myos.exe
```

(其中“-s”选项要求忽略输出文件中所有的符号信息，这可以有效减小可执行文件的大小，“--entry=start”选项及参数指出程序的入口点是 start.asm 中的 start 符号处，而“-Ttext=0x0”选项及参数则告诉链接器 text 节的绝对地址为 0，“-o myos.exe”选项及参数指出输出的可执行文件名为 myos.exe。)

## 15. 用 objcopy 工具将 myos.exe 拷贝成纯二进制格式的可执行文件 [myos.com](#)。

16. 用工具 writeA.exe 把 myos.com 优雅的写入 A.img 中从第 2 个扇区开始的位置。

```
D:\exe\myos\myos3-2>objdump -a myos.exe

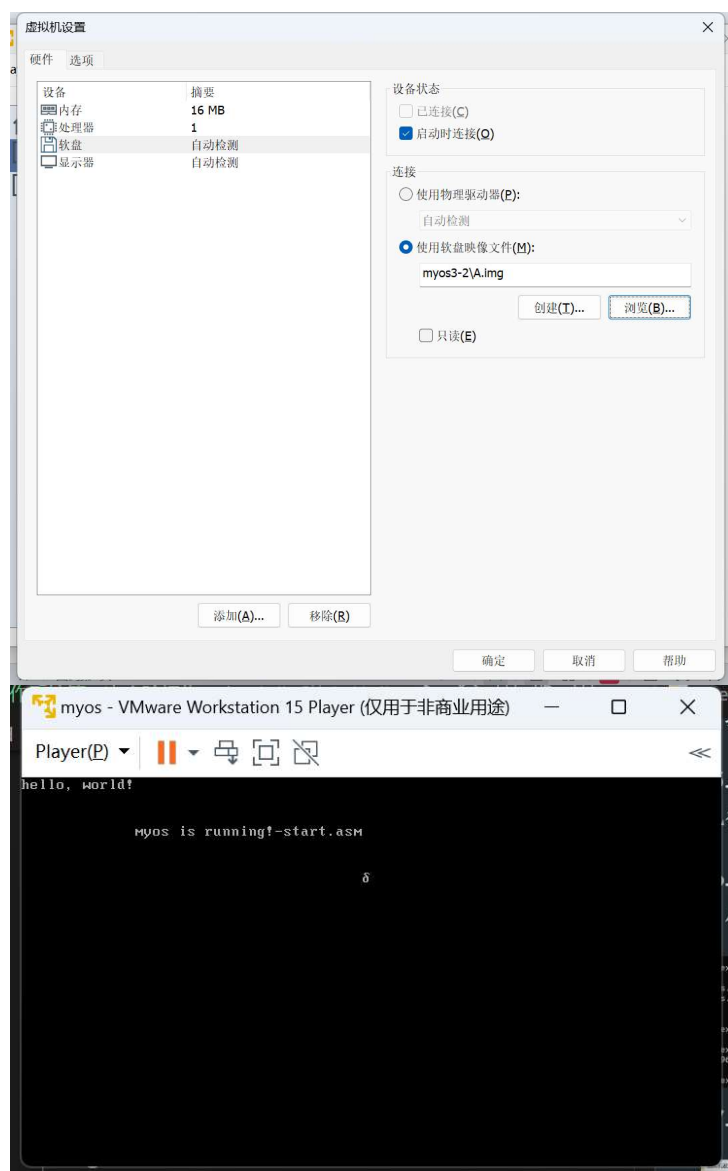
myos.exe:      file format pei-i386
myos.exe

D:\exe\myos\myos3-2>objcopy -O binary myos.exe myos.com

D:\exe\myos\myos3-2>writeA.exe myos.com A.img 2
16396 bytes copied from myos.com to A.img

D:\exe\myos\myos3-2>|
```

17. 测试操作系统



## 实验心得

通过本次实验，我知道了怎么将C语言写入我们的操作系统，以及汇编代码和c语言代码之间的相互链接，收益颇丰！