# 操作系统实验零

学号：171491219

姓名：崔凯慧

##### 实验题目

lab0，操作系统的编程基础

##### 二、实验环境

<http://www.shiyanlou.com/courses/221>

##### 三、实验要求

（一）安装linux环境，并安装gcc和gdb。

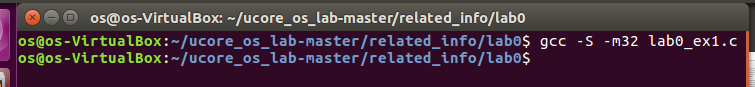
（二）了解汇编

1.尝试理解下面的命令

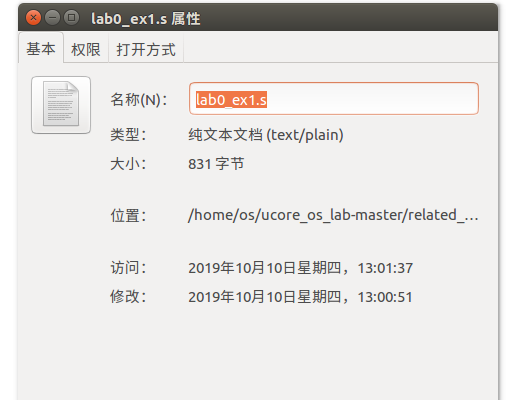
$gcc -S -m32 lab0\_ex1.c

接着我们将得到lab0\_ex1.s文件，请写出汇编代码与c代码之间的关系。

**实验结果：**







汇编代码与C代码关系：

程序在计算机中的运行是通过栈的方式不断的调用函数最会回归至主函数。

汇编语言是采用助记符号来编写程序的，用辅助符号代替[机器语言](https://www.baidu.com/s?wd=%E6%9C%BA%E5%99%A8%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的二进制码，就把机器语言变成了汇编语言。可以直接同计算机的底层软件甚至硬件进行交互。编写的代码非常难懂，不好维护，所以出现一些[面向过程](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E8%BF%87%E7%A8%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的语言，C语言就是其中之一。C是可以说是中级语言。它把高级语言的基本结构和语句与[低级语言](https://www.baidu.com/s?wd=%E4%BD%8E%E7%BA%A7%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的实用性结合起来。C 语言可以像汇编语言一样对位、字节和地址进行操作。

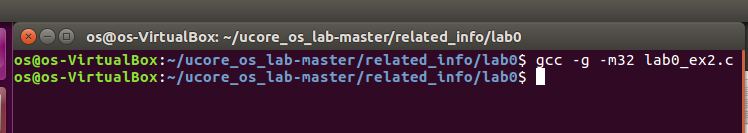
2.用gdb调试

尝试下面的命令，

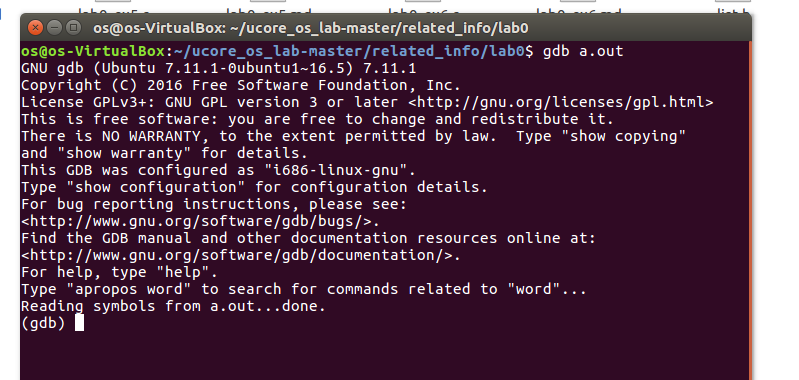
$gcc -g -m32 lab0\_ex2.c

接着我们会得到a.out文件，请用gdb调试，并写出设置断点、单步执行及查看变量的过程。

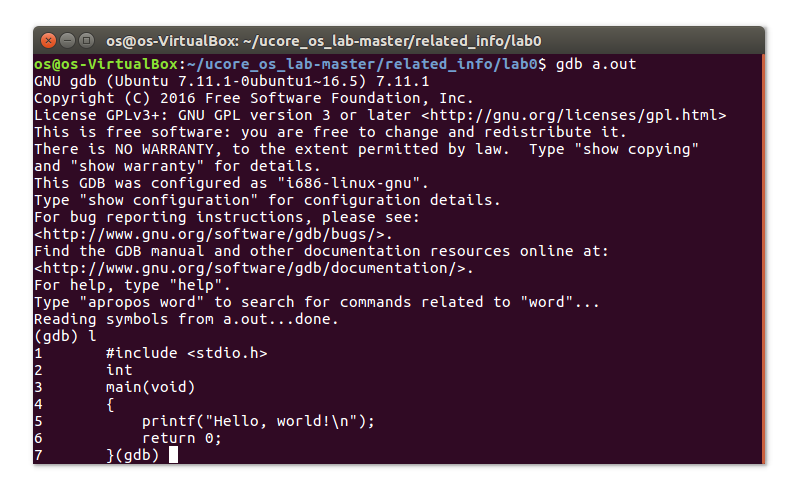
**实验结果：**



输入gdb a.out得到



然后输入 l 得到lab0\_ex2.c的源代码内容

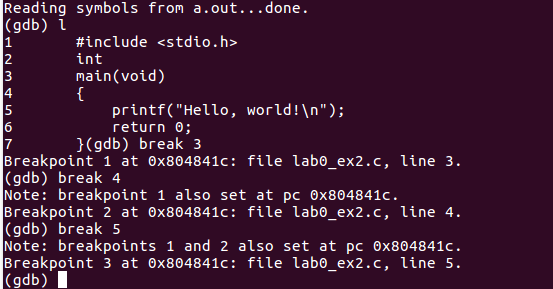


接下来开始调试：

（1）设置断点

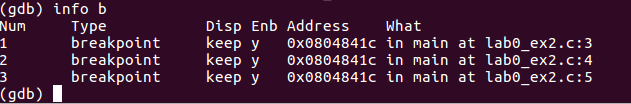
如下图所示，设置了三个断点，分别在第三行、第四行、第五行。

输入break 3回车 break 4回车 break 5



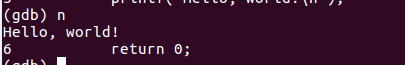
（2）查看设置完的断点信息

输入info b显示断点信息



（3）单步执行

单步执行输入n



（4）查看变量

查看变量的值可用命令p+变量名

追踪变量的值可用命令display+变量名

由于hello world程序没有变量，故无法显示。

3.掌握指针和类型转换相关的Ｃ编程

分析如下代码段，

#include <stdio.h>

#define STS\_IG32 0xE // 32-bit Interrupt Gate

#define STS\_TG32 0xF // 32-bit Trap Gate

typedef unsigned uint32\_t;

#define SETGATE(gate, istrap, sel, off, dpl) { \

(gate).gd\_off\_15\_0 = (uint32\_t)(off) & 0xffff; \

(gate).gd\_ss = (sel); \

(gate).gd\_args = 0; \

(gate).gd\_rsv1 = 0; \

(gate).gd\_type = (istrap) ? STS\_TG32 : STS\_IG32; \

(gate).gd\_s = 0; \

(gate).gd\_dpl = (dpl); \

(gate).gd\_p = 1; \

(gate).gd\_off\_31\_16 = (uint32\_t)(off) >> 16; \

}

/\* Gate descriptors for interrupts and traps \*/

struct gatedesc {

unsigned gd\_off\_15\_0 : 16; // low 16 bits of offset in segment

unsigned gd\_ss : 16; // segment selector

unsigned gd\_args : 5; // # args, 0 for interrupt/trap gates

unsigned gd\_rsv1 : 3; // reserved(should be zero I guess)

unsigned gd\_type : 4; // type(STS\_{TG,IG32,TG32})

unsigned gd\_s : 1; // must be 0 (system)

unsigned gd\_dpl : 2; // descriptor(meaning new) privilege level

unsigned gd\_p : 1; // Present

unsigned gd\_off\_31\_16 : 16; // high bits of offset in segment

};

int

main(void)

{

unsigned before;

unsigned intr;

unsigned after;

struct gatedesc gintr;

intr=8;

before=after=0;

gintr=\*((struct gatedesc \*)&intr);

SETGATE(gintr, 0,1,2,3);

intr=\*(unsigned \*)&(gintr);

printf("intr is 0x%x\n",intr);

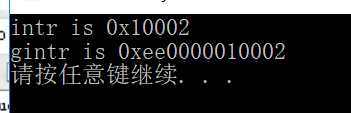
printf("gintr is 0x%llx\n", gintr);

return 0;

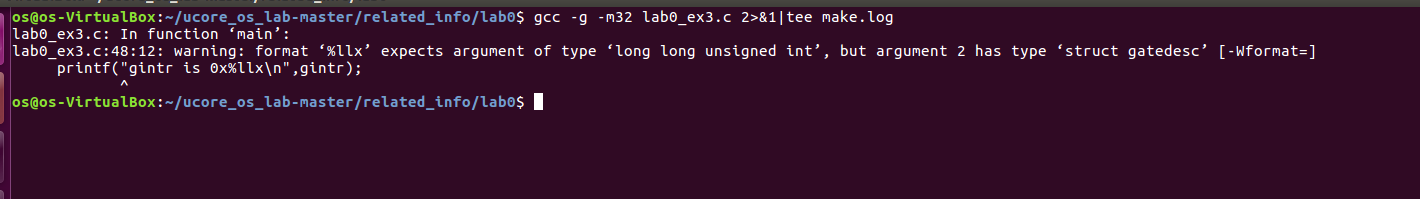
}

写出gintr和intr的结果，试着编译这段代码，如果遇到错误进行改正，并分析错误原因。

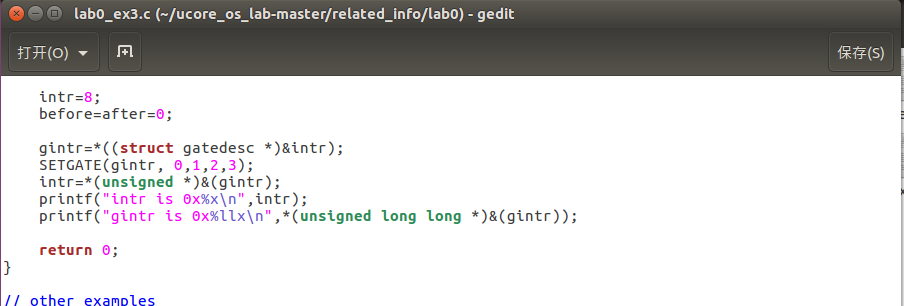
**运行结果：**



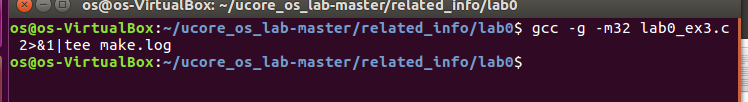
错误原因如下：



改正：



改正后再编译：

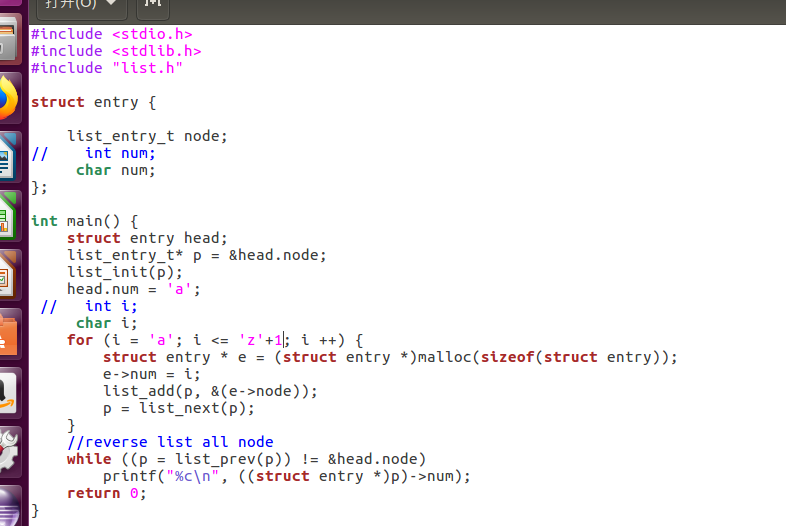


最终生成一个 a.out文件

4. 掌握通用链表结构相关的Ｃ编程

查看list.h和lab0\_ex4.c，编写一个程序，利用list.h中的链表结构，将26个英文字母存入链表中，并逆序打印出来。

Lab0\_ex4.c代码如下：



实验运行结果如下：

