实验题目：lab0，操作系统的编程基础

实验环境

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

安装linux环境，并安装gcc和gdb。

1. 了解汇编

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。

这个命令的意思是：GCC 编译 lab0\_ex2.c 编译32位应用程序（大概是吧）

汇编代码与c代码之间的关系：C语言是从汇编中提炼而来 C语言最终会被编译为汇编 c能做的事汇编能做，反过来则不一定。

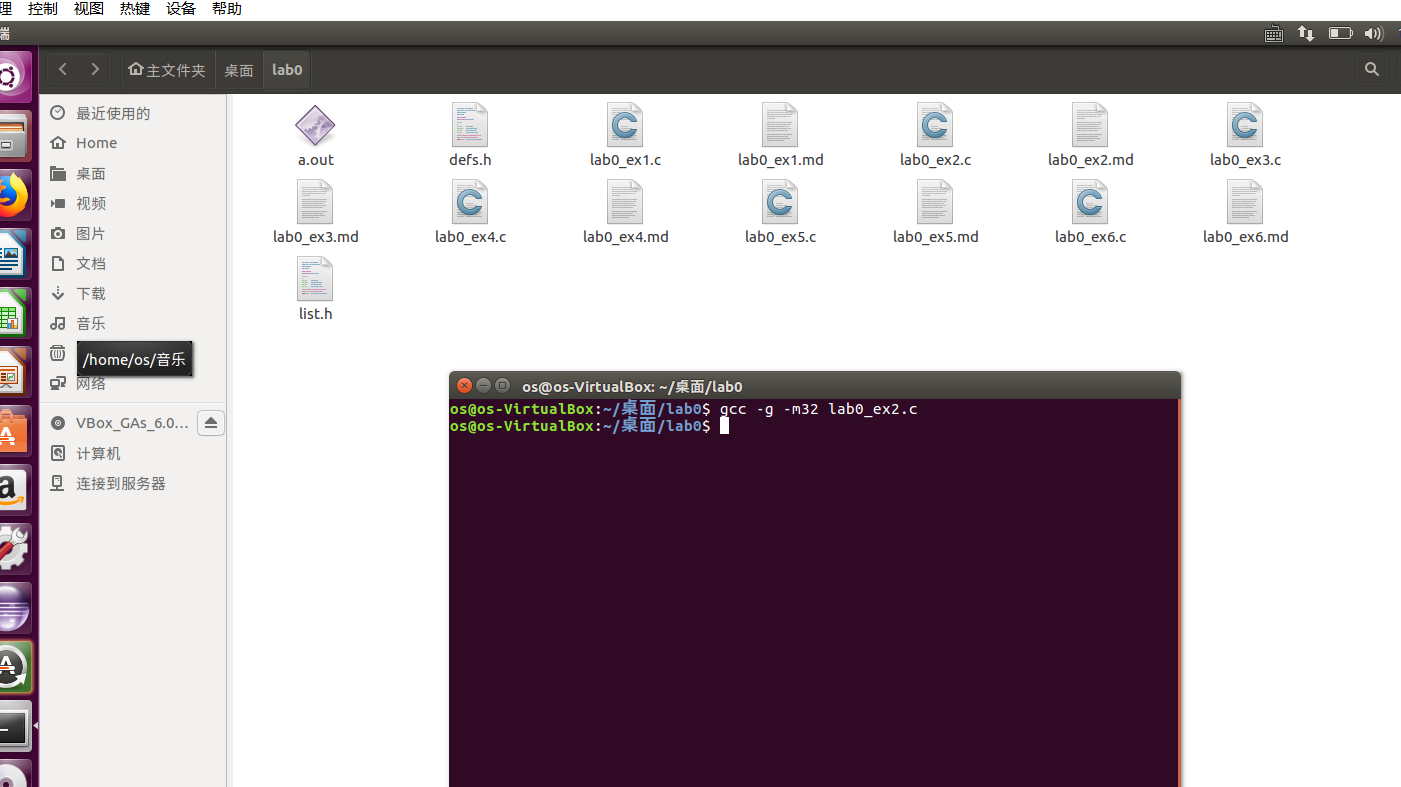
1. 用gdb调试

尝试下面的命令，

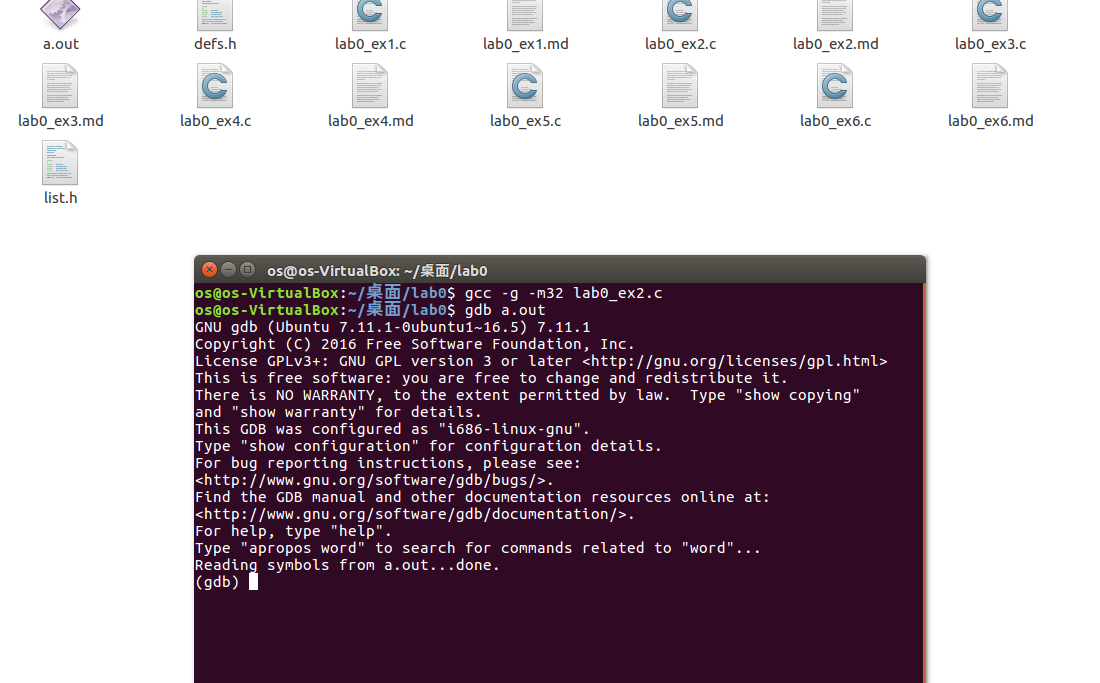
$gcc -g -m32 lab0\_ex2.c

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

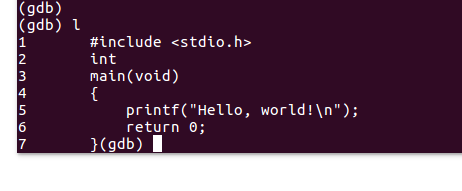
得到a.out :



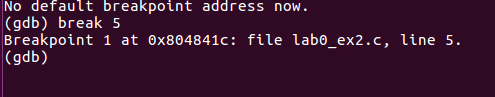
启动gdb：



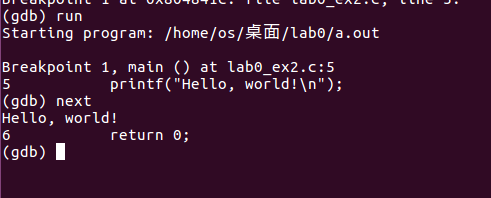
查看代码：



设置断点：



单步执行：



查看变量：



1. 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);

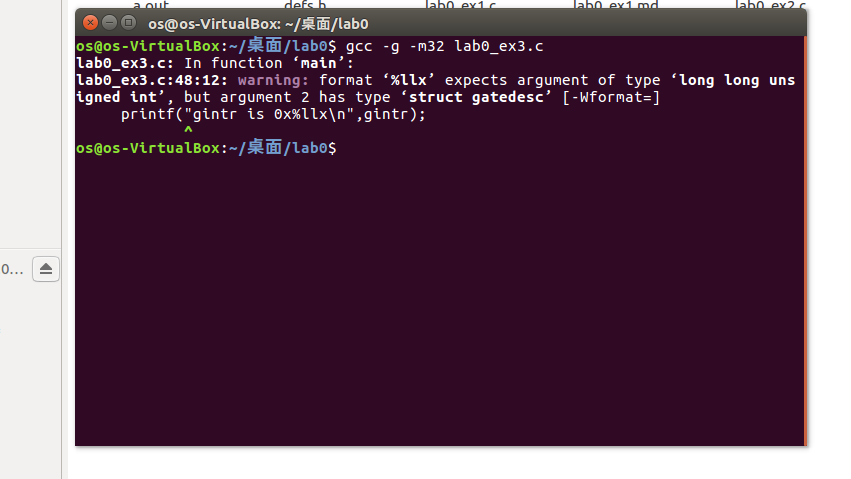
printf("intr is 0x%llx\n", gintr);

return 0;

}

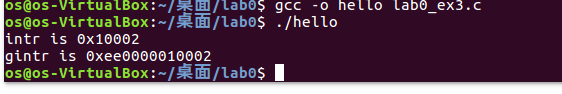
写出gintr和intr的结果，试着编译这段代码，如果遇到错误进行改正，并分析错误原因。

结果：



错误原因：

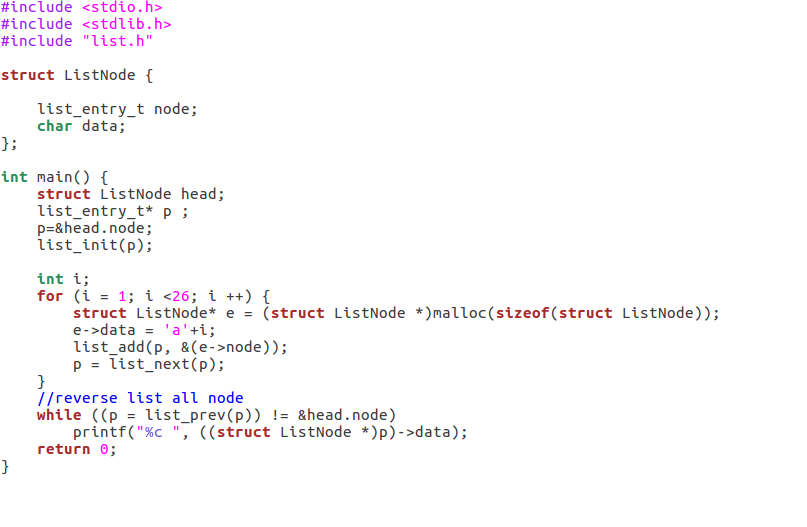
最终结果：



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。

改写lab0\_ex4.c:



运行结果：

