**实验题目：操作系统的编程基础**

安装linux环境，并安装gcc和gdb。

1. 了解汇编

$gcc -S -m32 lab0\_ex1.c

int count=1;

Int value=1;

Int buf[10];

Void main()

{

Asm(

“cld\n\t”//将标志寄存器Flag的方向标志位DF清零

“rep\n\t”//重复前缀指令

“stosl”//将EAX中的值保存在ES：EDI指向的地址中

:

:”c”(count),”a”(value),”D”(buf[0])

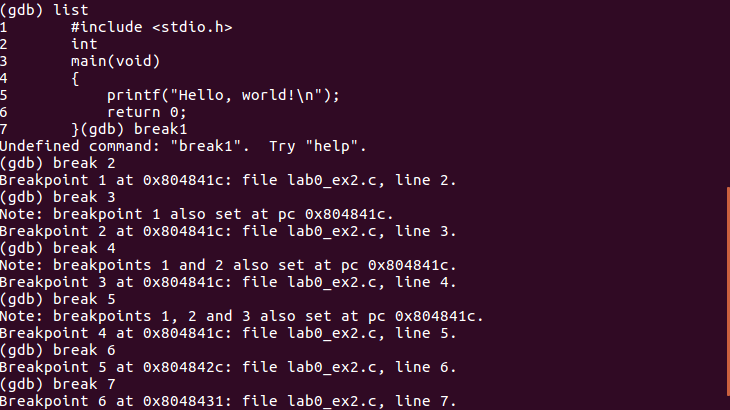
:

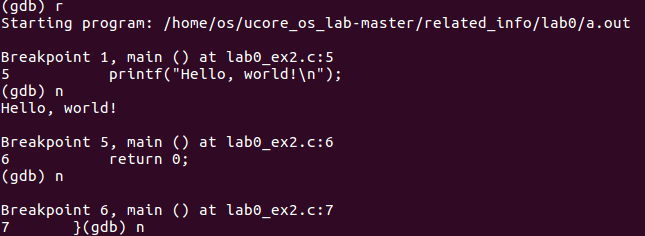
);

}

1. 用gdb调试

$gcc -g -m32 lab0\_ex2.c





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);

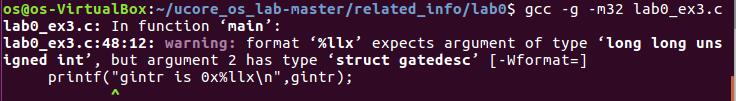
printf("intr is 0x%x\n",intr);

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。



改正

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",\*(unsigned long long\*)&( gintr));

分析：取gintr的地址，强制转化为指向llu型变量的指针，再引用这一地址得到llu型的变量



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。

#include <stdio.h>

#include <stdlib.h>

struct list\_entry {

struct list\_entry \*prev, \*next;

};

typedef struct list\_entry list\_entry\_t;

struct entry {

list\_entry\_t node;

int num;

};

int main() {

struct entry head;

list\_entry\_t\* p = &head.node;

list\_init(p);

head.num = 0;

int i;

for (i = 1; i != 26; i ++) {

struct entry \* e = (struct entry \*)malloc(sizeof(struct entry));

e->num = i;

list\_add(p, &(e->node));

p = list\_next(p);

}

printf("%d\n",((struct entry \*)p)->num);

while ((p = list\_prev(p)) != &head.node)

printf("%d\n", ((struct entry \*)p)->num);

return 0;

}