# 实验题目：lab0，操作系统的编程基础

## 了解汇编

1. $gcc -S -m32 lab0\_ex1.c
2. Gcc是编译命令，
3. -s是指生成汇编文件
4. -m32”选项将int、long和指针类型设置为32位，并生成在任何i386系统上运行的代码
5. 目标文件是lab0\_ex1.c

写出汇编代码与c代码之间的关系

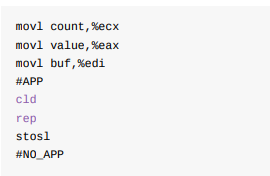
这是一个c程序



这儿asm（””）是gcc的基本内联汇编语句，引号选取的内容才是汇编语言

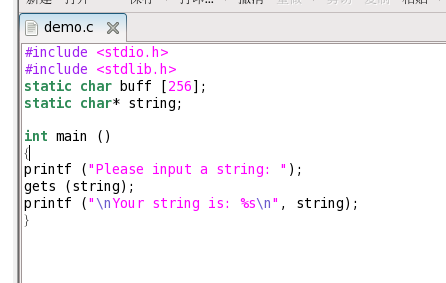
每一行都要加上 "\n\t"，是为了让 gcc 把内联汇编 代码翻译成一般的汇编代码时能够保证换行和留有一定的空格

生成的汇编代码如下

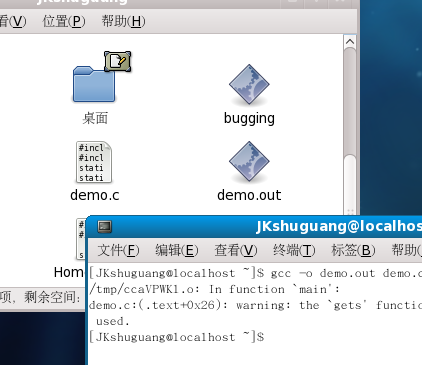


## 2用gdb调试

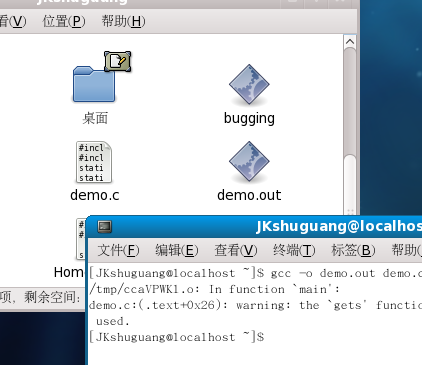
我先建立了一个c程序



在终端进行编译运行之后（输入jinxiaokun）



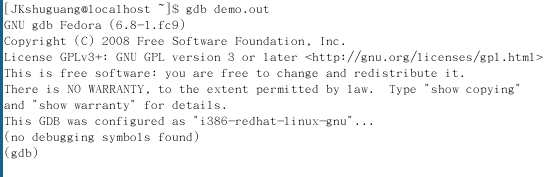
得到了一个警告信息，是编译器不建议使用gets()函数，认为这是有危险的，但是并不妨碍我们继续执行。



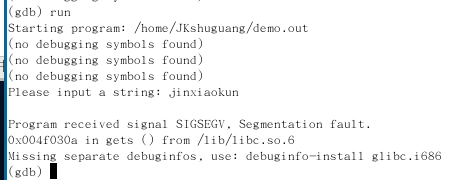
得到了一个段错误提示。

**于是我们的调试就开始了（！！！）**

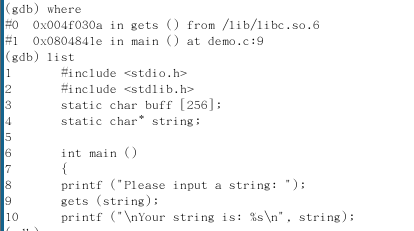
先使用gdb将我们生成的.out文件运行起来



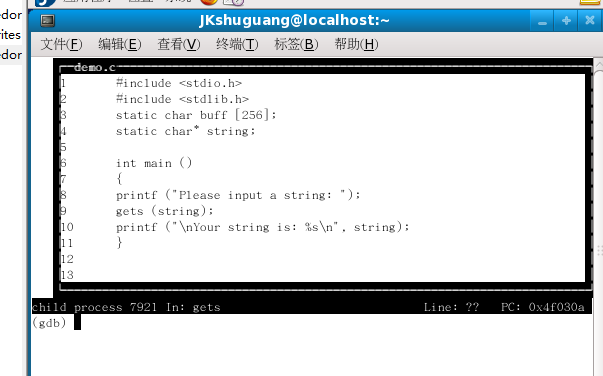
然后run跑动程序，输入字符串jinxiaokun



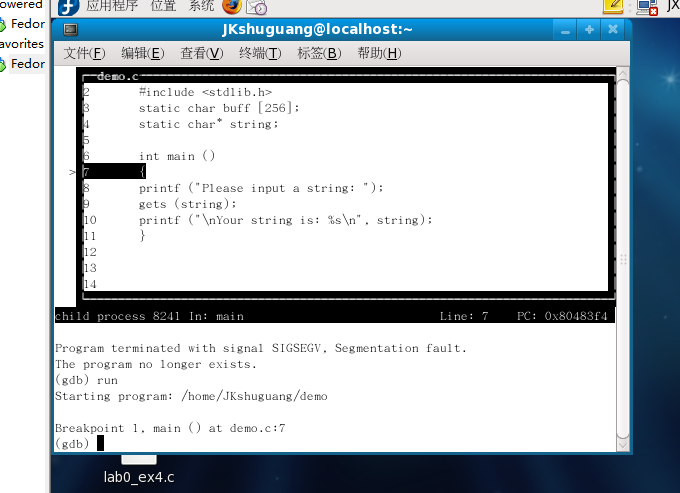
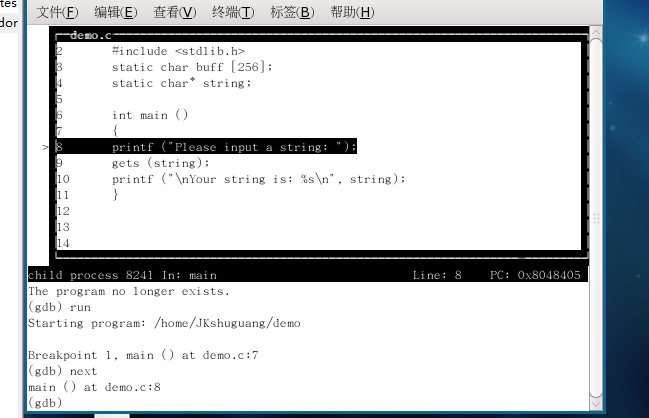
在输入where就可以得到程序错误的位置，输入list列出程序

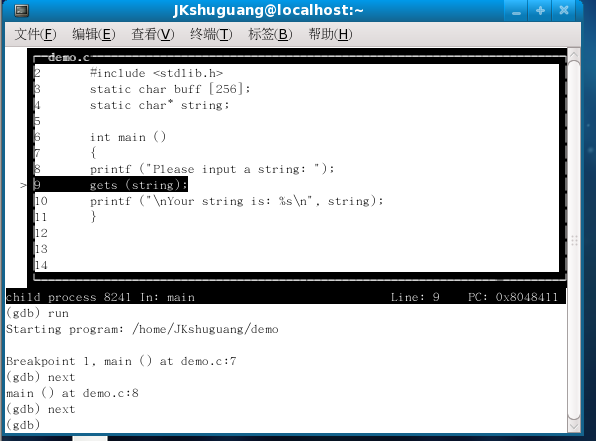
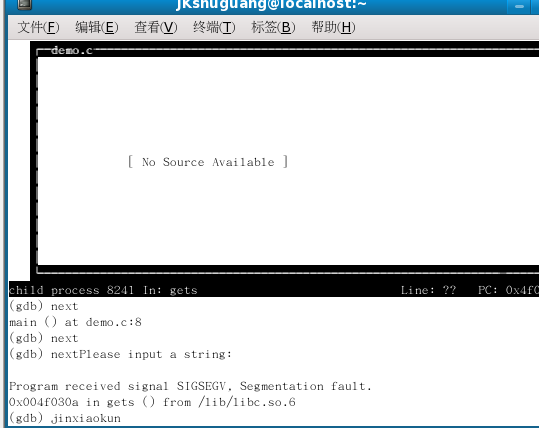


为了让调试变得可视化，我找到了layout src 命令



于是,开始设置断点为1



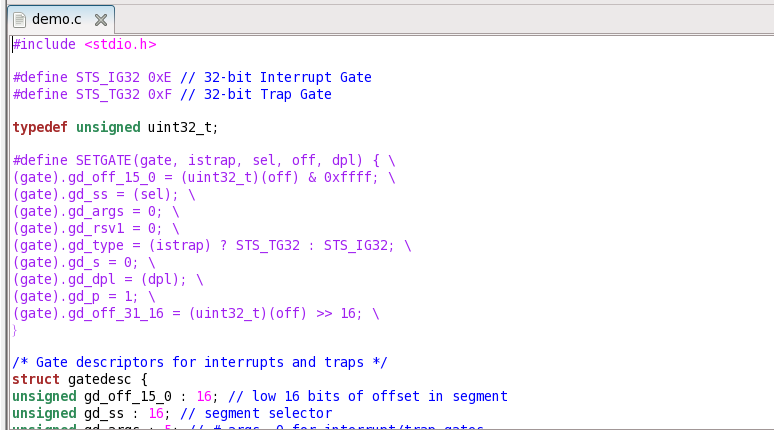


得到段错误，找到问题所在，就可以修改了。

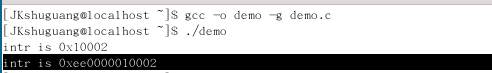
我们把buff赋值给string后得到解决

# 掌握指针和类型转换相关的Ｃ编程

编译程序，写出gintr和intr的结果（将代码拷贝到demo.c中）



运行后并没有发现问题，emmmm（除非下面输出少了一个g算的话）



intr is 0x10002

intr is 0xee0000010002

## 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;

};

//初始化指针

void list\_init(list\_entry\_t \*elm) {

elm->prev = elm->next = elm;

}

//判断指针是否微孔

bool list\_empty(list\_entry\_t \*list) {

return list->next == list;

}

void \_\_list\_add(list\_entry\_t \*elm, list\_entry\_t \*prev, list\_entry\_t \*next) {

prev->next = next->prev = elm;

elm->next = next;

elm->prev = prev;

}

void list\_add\_after(list\_entry\_t \*listelm, list\_entry\_t \*elm) {

\_\_list\_add(elm, listelm, listelm->next);

}

//添加元素

void list\_add(list\_entry\_t \*listelm, list\_entry\_t \*elm) {

list\_add\_after(listelm, elm);

}

void list\_add\_before(list\_entry\_t \*listelm, list\_entry\_t \*elm) {

\_\_list\_add(elm, listelm->prev, listelm);

}

list\_entry\_t \* list\_next(list\_entry\_t \*listelm) {

return listelm->next;

}

int main(){

struct entry head;

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

list\_init(p);

head.num = 0 ;

for (int i = 26; i != 0; i --) {

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

e->num = i;

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

p = list\_next(p);

printf("%c",i+'A'-1);

}

P = head.node;

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

struct entry \* e = p.node;

printf("%c",e->num+'A'-1);

p = list\_next(p);

}

return 0;

}

打印得到输出。