南昌大学实验报告

姓名：陈洪

学号：6103115105

邮箱地址：863159411@qq.com

专业班级：计科153班

实验日期：2018.04.16

课程名称：Linux程序设计实验

实验项目名称

C Programming in Linux

实验目的

掌握在Linux环境下编写C语言的方法，掌握gcc等相关工具的使用

实验基础

C语言语法，Linux系统的基本操作

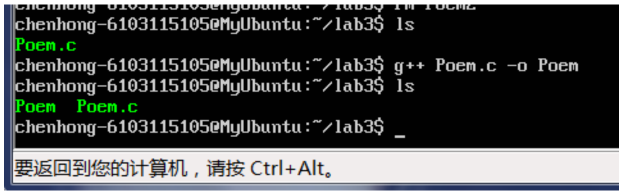
实验步骤

1.Linux Poem：The American C

代码诗歌的源代码如下：

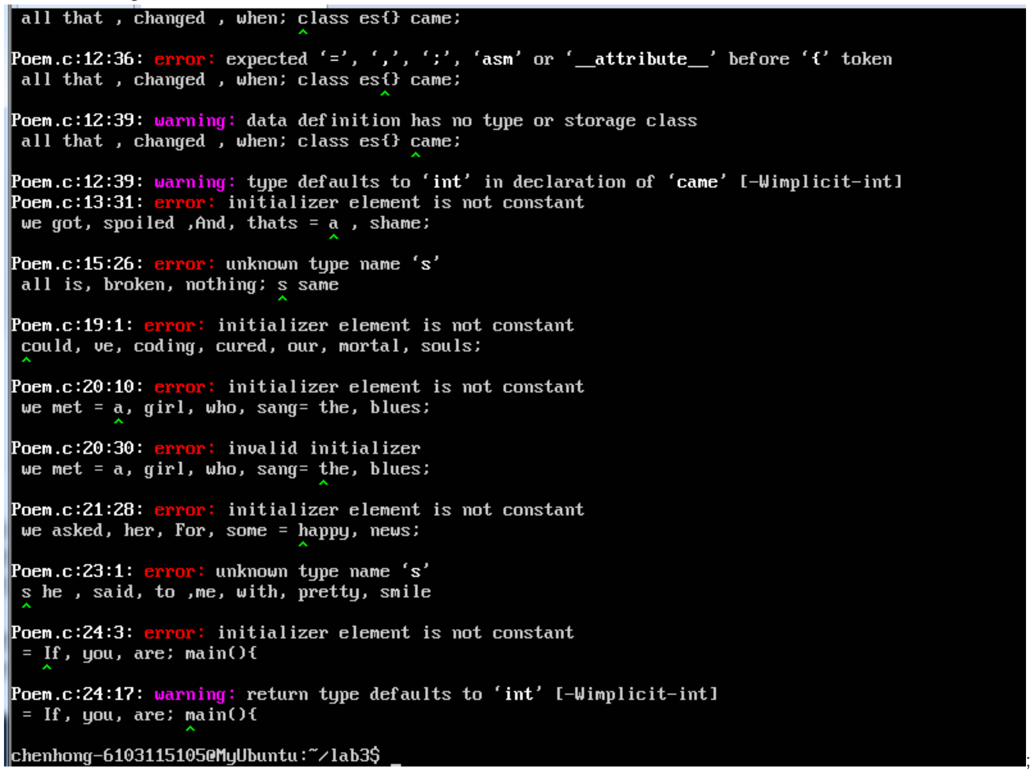
```
long long time,ago,
i,can,still,remember,how;
typedef struct s{} was,
all,we,had;

s o,  I,knew,If=I; had my, chance =
I,could, code, a, perfect ,prance;

s ee; was ruling;
wc were,happy
,good,olc,times;

all that,changed,when; class es{} came;
we got,spoiled,And,thats= a,shame;

all is,broken,nothing;s same
,c_plus_plus,has,killed,the,flame;

had We,believed,in,rocknroll=
could,ve,coding,cured,our,mortal,souls;

we met=a,girl,who,sang= the,blues;
we asked,her,For,some=happy,news;

s he,said,to,me,with,pretty,smile
=If,you,are; main(){
  return
  the,time
    }
```

1. 将这首代码诗歌正确编译
   不敢想象这么一首诗歌，居然是一段符合C语言规则的一段代码，但是————
   注意到代码的第四段的一个关键字"class",看来上面的"c语言"一词要改为"c++"了，因为显然，C语言里面是没有类个概念了，否则C语言就该是面向对象而不是面向过程的语言了。
   显然，老师上课的时候强调的选择正确的编译器，大概就是在暗示我们一些什么吧。既然是编译c++的代码，那么我们就不应该用GCC来编译，而应该用G++来编译。实验截图如下：

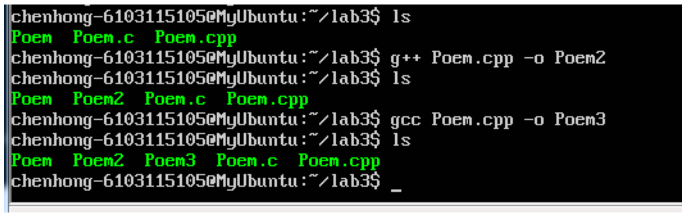

```
chenhong-6103115105@MyUbuntu:~/lab3$ ls
Poem.c
chenhong-6103115105@MyUbuntu:~/lab3$ g++ Poem.c -o Poem
chenhong-6103115105@MyUbuntu:~/lab3$ ls
Poem  Poem.c
chenhong-6103115105@MyUbuntu:~/lab3$ _
```
要返回到您的计算机，请按 Ctrl+Alt。

显然，编译成功了。

我们看看如果用gcc来编译，会出现什么问题：


```
 all that , changed , when; class es{} came;
                              ^
Poem.c:12:36: error: expected '=', ',', ';', 'asm' or '__attribute__' before '{' token
 all that , changed , when; class es{} came;
                                    ^
Poem.c:12:39: warning: data definition has no type or storage class
 all that , changed , when; class es{} came;
                                       ^
Poem.c:12:39: warning: type defaults to 'int' in declaration of 'came' [-Wimplicit-int]
Poem.c:13:31: error: initializer element is not constant
 we got, spoiled ,And, thats = a , shame;
                               ^
Poem.c:15:26: error: unknown type name 's'
 all is, broken, nothing; s same
                          ^
Poem.c:19:1: error: initializer element is not constant
 could, ve, coding, cured, our, mortal, souls;
 ^
Poem.c:20:10: error: initializer element is not constant
 we met = a, girl, who, sang= the, blues;
          ^
Poem.c:20:30: error: invalid initializer
 we met = a, girl, who, sang= the, blues;
                              ^
Poem.c:21:28: error: initializer element is not constant
 we asked, her, For, some = happy, news;
                            ^
Poem.c:23:1: error: unknown type name 's'
 s he , said, to ,me, with, pretty, smile
 ^
Poem.c:24:3: error: initializer element is not constant
 = If, you, are; main(){
   ^
Poem.c:24:17: warning: return type defaults to 'int' [-Wimplicit-int]
 = If, you, are; main(){
                 ^
chenhong-6103115105@MyUbuntu:~/lab3$ _
```
;

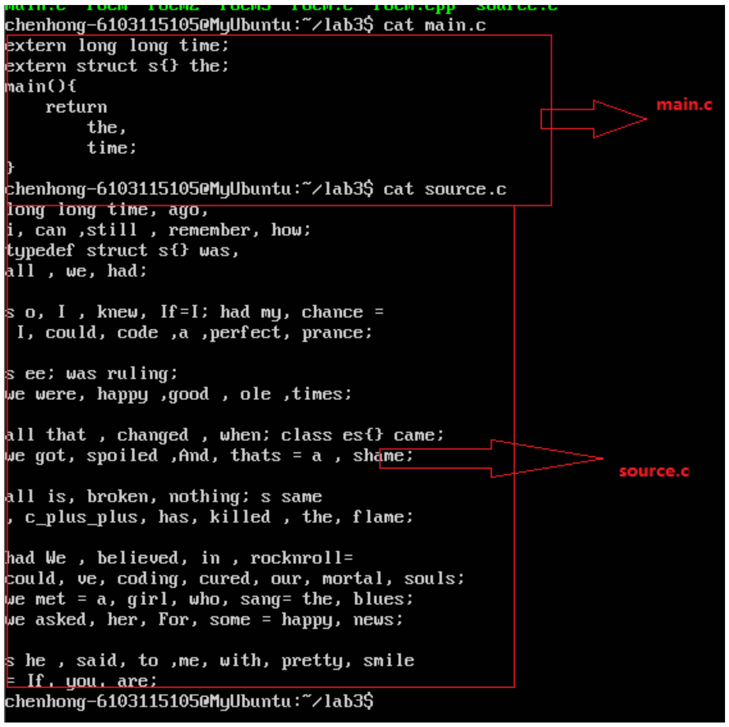
现在来讨论一个问题，我在当前目录先把代码"Poem.c"复制一份，名字为"Poem.cpp",然后分别用gcc和g++编译一下，结果是：两个都通过了！！！这不是和上面的言辞矛盾了吗？先看看实验结果：


```
chenhong-6103115105@MyUbuntu:~/lab3$ ls
Poem  Poem.c  Poem.cpp
chenhong-6103115105@MyUbuntu:~/lab3$ g++ Poem.cpp -o Poem2
chenhong-6103115105@MyUbuntu:~/lab3$ ls
Poem  Poem2  Poem.c  Poem.cpp
chenhong-6103115105@MyUbuntu:~/lab3$ gcc Poem.cpp -o Poem3
chenhong-6103115105@MyUbuntu:~/lab3$ ls
Poem  Poem2  Poem3  Poem.c  Poem.cpp
chenhong-6103115105@MyUbuntu:~/lab3$ _
```

查的资料才知道：

> 对于gcc和g++来说，文件名后缀为.c的，gcc把它当作是C程序，而g++当作是c++程序；后缀为.cpp的，两者都会认为是c++程序。
> 这么已解释，上面的结果也就讲得通了。

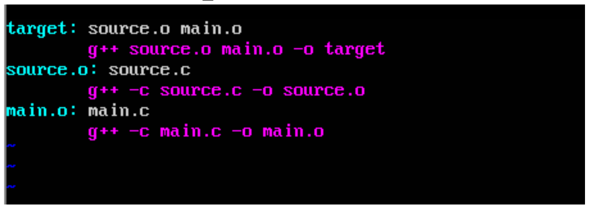
2. 将源代码拆成两个文件：`source.c`和`main.c`
   由代码诗歌可以看出来，在main()函数里面也就使用了两个变量，而剩下的大部分变量都只是声明了一下，因此我们做如下的分割：



```
main.c  Poem  Poem2  Poem3  Poem.c  Poem.cpp  source.c
chenhong-6103115105@MyUbuntu:~/lab3$ cat main.c
extern long long time;
extern struct s{} the;
main(){
    return
        the,
        time;
}
chenhong-6103115105@MyUbuntu:~/lab3$ cat source.c
long long time, ago,
i, can ,still , remember, how;
typedef struct s{} was,
all , we, had;

s o, I , knew, If=I; had my, chance =
 I, could, code ,a ,perfect, prance;

s ee; was ruling;
we were, happy ,good , ole ,times;

all that , changed , when; class es{} came;
we got, spoiled ,And, thats = a , shame;

all is, broken, nothing; s same
, c_plus_plus, has, killed , the, flame;

had We , believed, in , rocknroll=
could, ve, coding, cured, our, mortal, souls;
we met = a, girl, who, sang= the, blues;
we asked, her, For, some = happy, news;

s he , said, to ,me, with, pretty, smile
= If, you, are;
chenhong-6103115105@MyUbuntu:~/lab3$
```

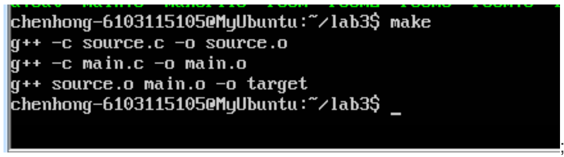
3. 使用Makefile重新编译上面的代码

   先创建一个名为"makefile_demo"的文件，其中的内容如下所示：



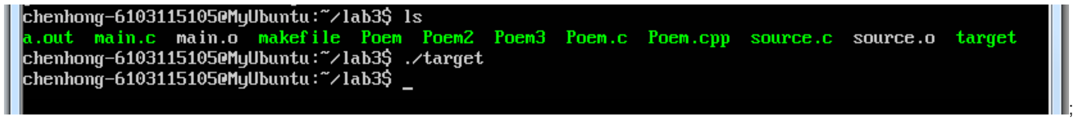
```
target: source.o main.o
        g++ source.o main.o -o target
source.o: source.c
        g++ -c source.c -o source.o
main.o: main.c
        g++ -c main.c -o main.o
~
~
~
```

   编译的结果如下：



```
chenhong-6103115105@MyUbuntu:~/lab3$ make
g++ -c source.c -o source.o
g++ -c main.c -o main.o
g++ source.o main.o -o target
chenhong-6103115105@MyUbuntu:~/lab3$ _
```

;



```
chenhong-6103115105@MyUbuntu:~/lab3$ ls
a.out  main.c  main.o  makefile  Poem  Poem2  Poem3  Poem.c  Poem.cpp  source.c  source.o  target
chenhong-6103115105@MyUbuntu:~/lab3$ ./target
chenhong-6103115105@MyUbuntu:~/lab3$ _
```

;

   由上图可知，Makefile编译上述的两个文件成功了。

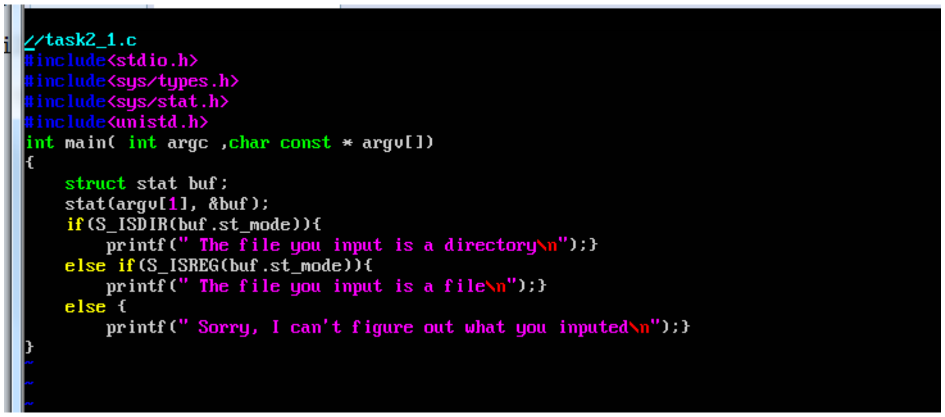
2.Use C and Makefile for Coding(使用c语言程序处理文件问题)

Tasks

1. You are asked to write a C code to check whether an input string is a file, o directory, or else.
2. Print the mode of the file, if it is a file. If you are the owner of the file, chmod it into 777, using C code.
3. If this is not a file or a folder/directory, provide a mechanism to handle the error.
4. Write a makefile for the above three codes and make a successful compilation

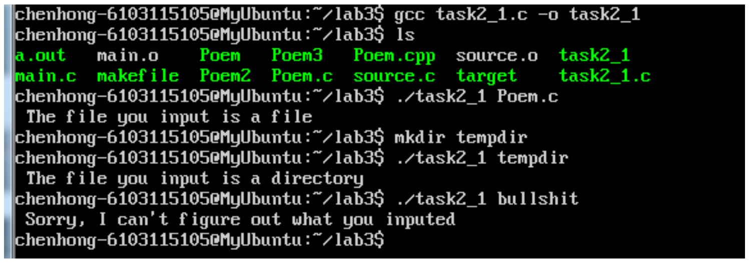
在完成下面任务之前，首先要去大致了解一下关于结构体stat的一些基本知识。我在以下链接初步了解了一下知识
https://www.cnblogs.com/CSU-PL/archive/2013/06/06/3120757.html

Q1:代码如下图所示：



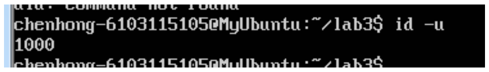
```
//task2_1.c
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
int main( int argc ,char const * argv[])
{
    struct stat buf;
    stat(argv[1], &buf);
    if(S_ISDIR(buf.st_mode)){
        printf(" The file you input is a directory\n");}
    else if(S_ISREG(buf.st_mode)){
        printf(" The file you input is a file\n");}
    else {
        printf(" Sorry, I can't figure out what you inputed\n");}
}
```

实验结果如下：



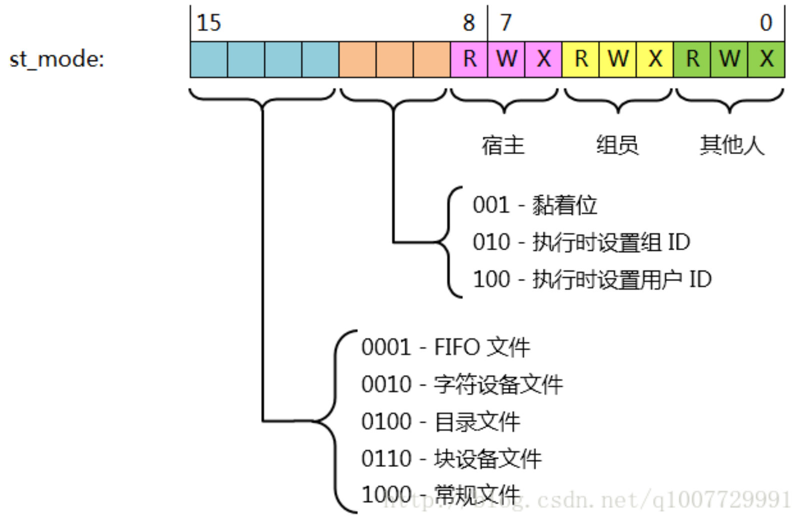
```
chenhong-6103115105@MyUbuntu:~/lab3$ gcc task2_1.c -o task2_1
chenhong-6103115105@MyUbuntu:~/lab3$ ls
a.out    main.o     Poem    Poem3   Poem.cpp  source.o  task2_1
main.c   makefile   Poem2   Poem.c  source.c  target    task2_1.c
chenhong-6103115105@MyUbuntu:~/lab3$ ./task2_1 Poem.c
 The file you input is a file
chenhong-6103115105@MyUbuntu:~/lab3$ mkdir tempdir
chenhong-6103115105@MyUbuntu:~/lab3$ ./task2_1 tempdir
 The file you input is a directory
chenhong-6103115105@MyUbuntu:~/lab3$ ./task2_1 bullshit
 Sorry, I can't figure out what you inputed
chenhong-6103115105@MyUbuntu:~/lab3$
```

Q2：这题首先得知道当前用户下的uid号是多少，可以用命令`id -u`来查找,我的结果为1000，结果如下：

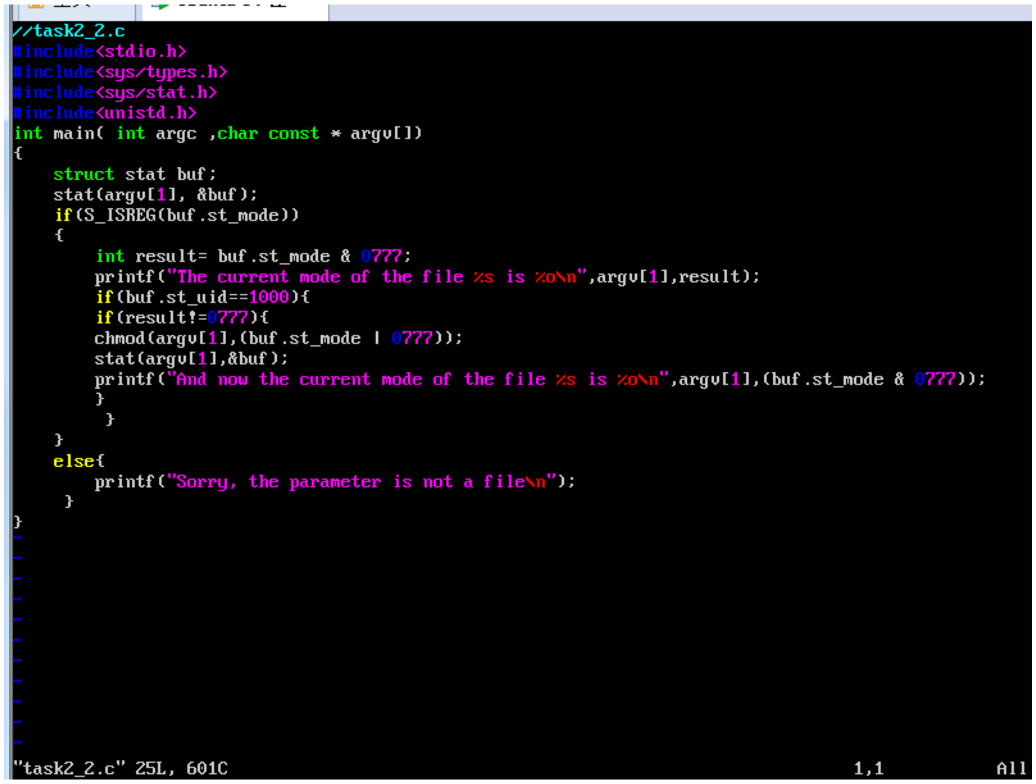


```
uid: command not found
chenhong-6103115105@MyUbuntu:~/lab3$ id -u
1000
chenhong-6103115105@MyUbuntu:~/lab3$
```

其次，我们来看一看st_mode的结构：

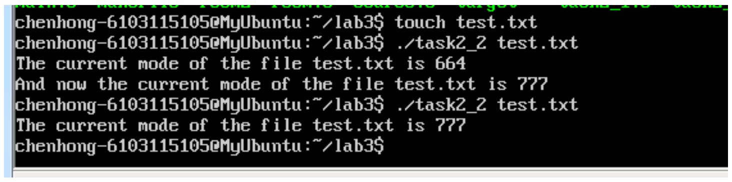


为了清晰地知道当前文件的权限情况，我们需要将st_mode的值与八进制的值“0777”用逻辑符号"&"相与一下，再用八进制的格式输出。而要把文件的权限改为777，那么我们只要把当前的st_mode的值与八进制值“0777”用逻辑符号"|"相或一下，这样就可以改写当前文件的权限为777。
那么这题的代码就如下图所示了：



```
//task2_2.c
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
int main( int argc ,char const * argv[])
{
    struct stat buf;
    stat(argv[1], &buf);
    if(S_ISREG(buf.st_mode))
    {
        int result= buf.st_mode & 0777;
        printf("The current mode of the file %s is %o\n",argv[1],result);
        if(buf.st_uid==1000){
        if(result!=0777){
        chmod(argv[1],(buf.st_mode | 0777));
        stat(argv[1],&buf);
        printf("And now the current mode of the file %s is %o\n",argv[1],(buf.st_mode & 0777));
        }
         }
    }
    else{
        printf("Sorry, the parameter is not a file\n");
     }
}
~
~
~
~
~
~
~
~
~
~
~
"task2_2.c" 25L, 601C                                                1,1           All
```

实验结果如下：



```
chenhong-6103115105@MyUbuntu:~/lab3$ touch test.txt
chenhong-6103115105@MyUbuntu:~/lab3$ ./task2_2 test.txt
The current mode of the file test.txt is 664
And now the current mode of the file test.txt is 777
chenhong-6103115105@MyUbuntu:~/lab3$ ./task2_2 test.txt
The current mode of the file test.txt is 777
chenhong-6103115105@MyUbuntu:~/lab3$
```

Q3.错误处理

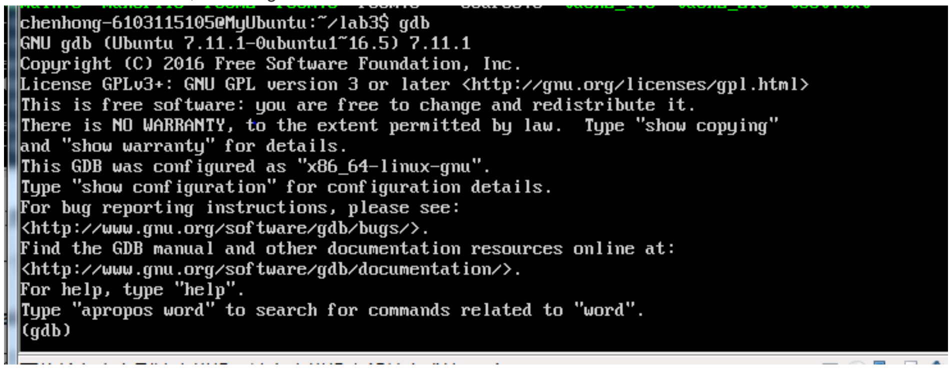
这一步在第一步已经体现出来了，在此不再叙述。

Q4.使用Makefile重新编译

sorry，这题不会。。。上面两个文件是两个相对独立的文件，他们之间没有什么相互调用的关系，这样应该生成两个独立的target，但是老师要求的是把上面的文件用makefile重新编译。。。那就是生成一个可执行文件。。。。。。

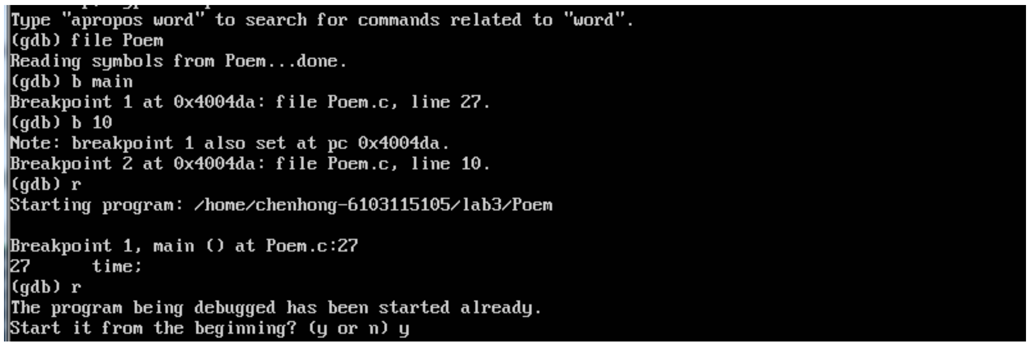
3.使用GDB

- 重新编译源文件Poem.c,然后进入gdb调试阶段



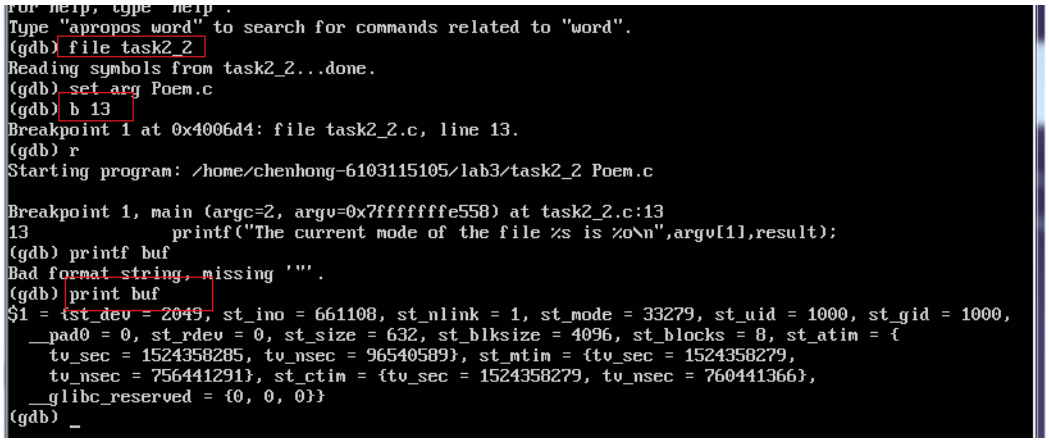
```
chenhong-6103115105@MyUbuntu:~/lab3$ gdb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)
```

设置断点，以及调试



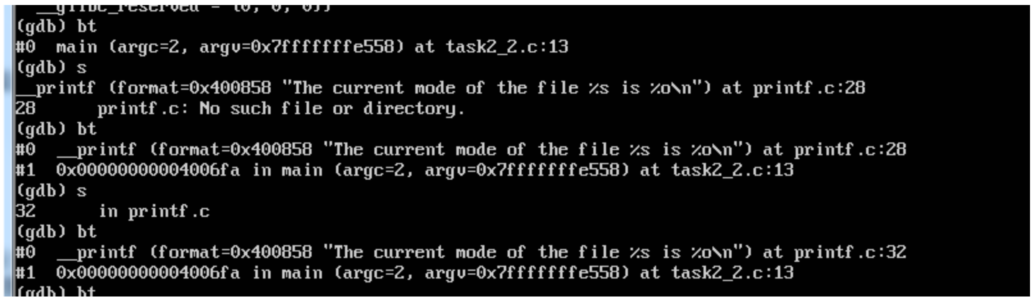
```
Type "apropos word" to search for commands related to "word".
(gdb) file Poem
Reading symbols from Poem...done.
(gdb) b main
Breakpoint 1 at 0x4004da: file Poem.c, line 27.
(gdb) b 10
Note: breakpoint 1 also set at pc 0x4004da.
Breakpoint 2 at 0x4004da: file Poem.c, line 10.
(gdb) r
Starting program: /home/chenhong-6103115105/lab3/Poem

Breakpoint 1, main () at Poem.c:27
27        time;
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
```

设置断点，以及调试



```
Type "apropos word" to search for commands related to "word".
(gdb) file task2_2
Reading symbols from task2_2...done.
(gdb) set arg Poem.c
(gdb) b 13
Breakpoint 1 at 0x4006d4: file task2_2.c, line 13.
(gdb) r
Starting program: /home/chenhong-6103115105/lab3/task2_2 Poem.c

Breakpoint 1, main (argc=2, argv=0x7fffffffe558) at task2_2.c:13
13                printf("The current mode of the file %s is %o\n",argv[1],result);
(gdb) printf buf
Bad format string, missing '"'.
(gdb) print buf
$1 = {st_dev = 2049, st_ino = 661108, st_nlink = 1, st_mode = 33279, st_uid = 1000, st_gid = 1000,
  __pad0 = 0, st_rdev = 0, st_size = 632, st_blksize = 4096, st_blocks = 8, st_atim = {
    tv_sec = 1524358285, tv_nsec = 96540589}, st_mtim = {tv_sec = 1524358279,
    tv_nsec = 756441291}, st_ctim = {tv_sec = 1524358279, tv_nsec = 760441366},
  __glibc_reserved = {0, 0, 0}}
(gdb) _
```

堆栈情况：



```
(gdb) bt
#0  main (argc=2, argv=0x7fffffffe558) at task2_2.c:13
(gdb) s
__printf (format=0x400858 "The current mode of the file %s is %o\n") at printf.c:28
28      printf.c: No such file or directory.
(gdb) bt
#0  __printf (format=0x400858 "The current mode of the file %s is %o\n") at printf.c:28
#1  0x00000000004006fa in main (argc=2, argv=0x7fffffffe558) at task2_2.c:13
(gdb) s
32      in printf.c
(gdb) bt
#0  __printf (format=0x400858 "The current mode of the file %s is %o\n") at printf.c:32
#1  0x00000000004006fa in main (argc=2, argv=0x7fffffffe558) at task2_2.c:13
```

从堆栈的情况来看，在程序的单步调试过程中，堆栈也在变化。

实验数据或结果

如上面所述的各个步骤

实验思考

问：为什么写个实验报告用了将近一天？
答：我太菜了。

这次实验了解了关于makefile的一些基本的用法，以及如何在C语言中一些处理文件的方法。然后是初步了解了jdb的一些设置断点，传参，查值，单步调试等命令。感慨前人为我们搭路的时候，都经历了写那些难以想象的痛楚。但是，他们学到的底层的东西，才是真正的知识。

参考资料

百度