南昌大学实验报告

姓名：胡正阳

学号：8000115350

邮箱地址：zxcv123411@yeah.net

专业班级：计科154班

实验日期：2018/4/21

课程名称：Linux程序设计实验

实验项目名称

Lab 3: C Programming in Linux

实验目的

- the C programming language
- the development toolchain (pre-processor, compiler, assembler, linker)
- the automating the compilation process using Makefiles

实验基础

- 安装有Ubuntu 16.04.4 x64系统的电脑

实验步骤

1. 编写并运行第3张PPT的C程序
2. 将C代码分成两个文件：main.c和source.c
3. 尝试用Makefile重新编译一遍
4. 编写C代码检查输入字符串为文件还是目录亦或是别的什么东西
5. 若为文件则打印文件权限，若为文件所有者，则修改权限为777
6. 若既不是文件也不是目录，则进行错误处理
7. 用makefile成功编译上述三段代码
8. 将前面的代码用gdb friendly的方式编译

9. 设置断点来观察在第一个文件中的time变量值的变化以及第二个文件中的输入参数
10. 检查代码在哪并打印目标代码目前的栈信息

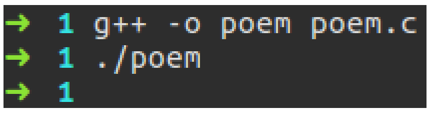
实验数据或结果

1. 编写并运行第3张PPT的C程序

   编写程序

```
long long time, age,
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; main(){
    return
        the,
        time;
}
```

   编译并运行程序



```
➜  1 g++ -o poem poem.c
➜  1 ./poem
➜  1
```

2. 将C代码分成两个文件：main.c和source.c

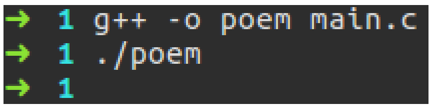
   main.c

```
#include "source.c"
main(){
    return
        the,
        time;
}
```

   source.c

```
long long time, age,
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;
```

再次编译并运行程序



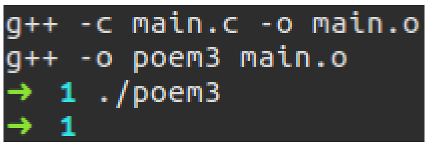
```
➜  1 g++ -o poem main.c
➜  1 ./poem
➜  1
```

3. 尝试用Makefile重新编译一遍

mk文件

```
#main/Makefile

all: poem3

poem3: main.o
    g++ -o poem3 main.o

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

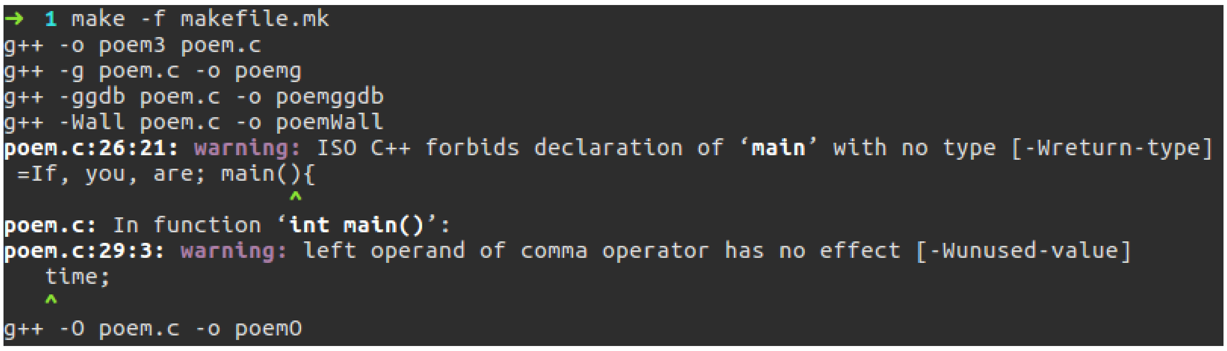
编译并运行



```
g++ -c main.c -o main.o
g++ -o poem3 main.o
➜  1 ./poem3
➜  1
```

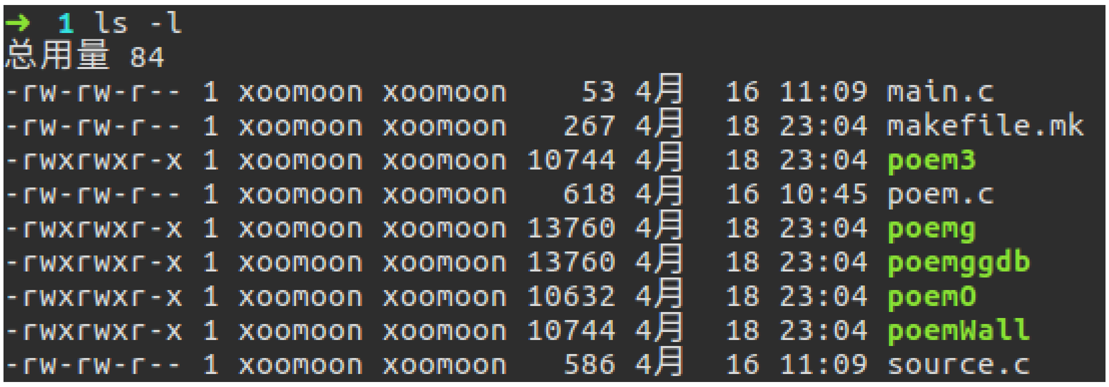
使用不同的命令编译

```makefile
#main/Makefile

all: poem3 poemg poemggdb poemWall poemO

poem3: poem.c
	g++ -o poem3 poem.c

poemg: poem.c
	g++ -g poem.c -o poemg

poemggdb: poem.c
	g++ -ggdb poem.c -o poemggdb

poemWall: poem.c
	g++ -Wall poem.c -o poemWall

poemO: poem.c
	g++ -O poem.c -o poemO
```



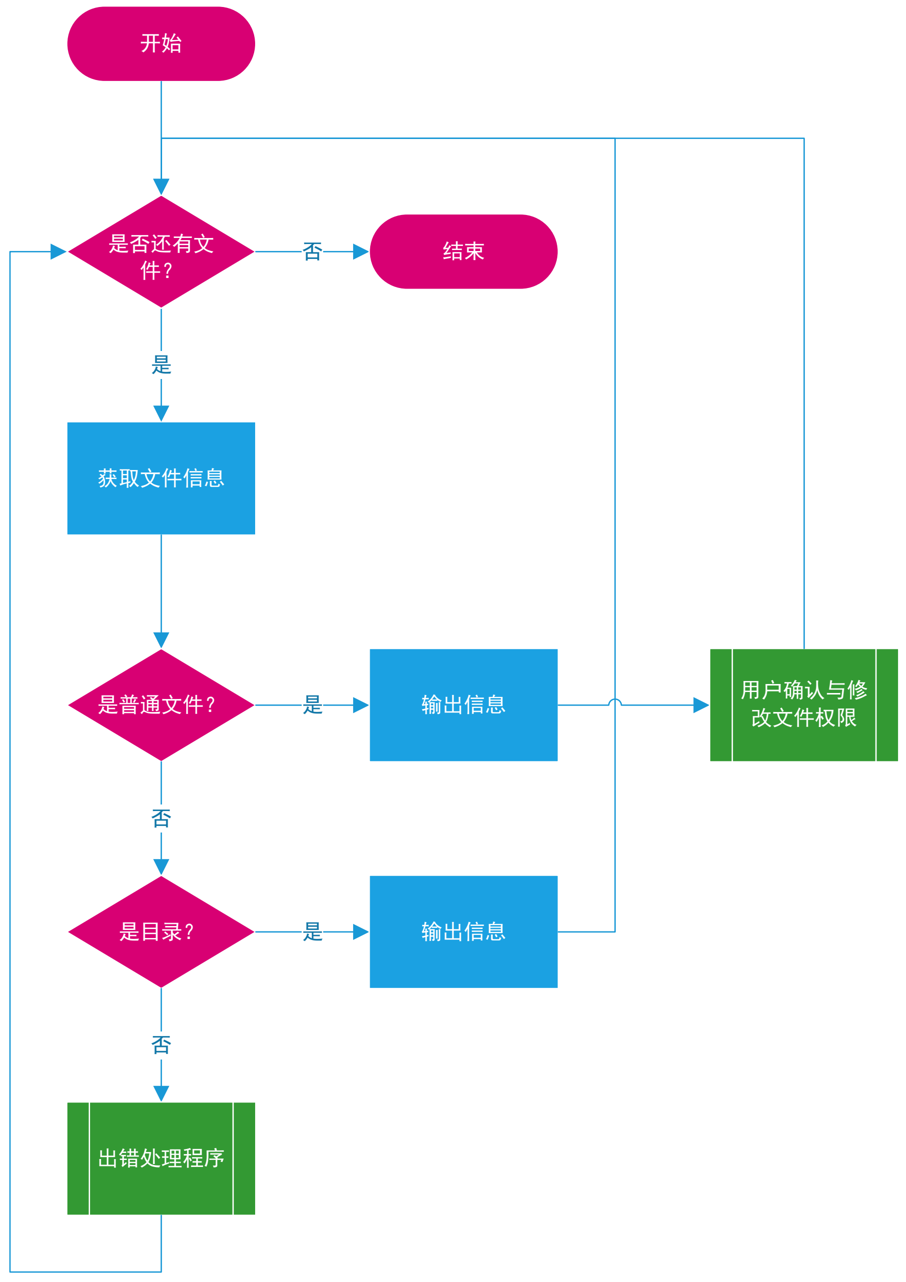
```
➜  1 make -f makefile.mk
g++ -o poem3 poem.c
g++ -g poem.c -o poemg
g++ -ggdb poem.c -o poemggdb
g++ -Wall poem.c -o poemWall
poem.c:26:21: warning: ISO C++ forbids declaration of ‘main’ with no type [-Wreturn-type]
 =If, you, are; main(){
                     ^
poem.c: In function ‘int main()’:
poem.c:29:3: warning: left operand of comma operator has no effect [-Wunused-value]
   time;
   ^
g++ -O poem.c -o poemO
```

查看区别



```
➜  1 ls -l
总用量 84
-rw-rw-r-- 1 xoomoon xoomoon    53 4月  16 11:09 main.c
-rw-rw-r-- 1 xoomoon xoomoon   267 4月  18 23:04 makefile.mk
-rwxrwxr-x 1 xoomoon xoomoon 10744 4月  18 23:04 poem3
-rw-rw-r-- 1 xoomoon xoomoon   618 4月  16 10:45 poem.c
-rwxrwxr-x 1 xoomoon xoomoon 13760 4月  18 23:04 poemg
-rwxrwxr-x 1 xoomoon xoomoon 13760 4月  18 23:04 poemggdb
-rwxrwxr-x 1 xoomoon xoomoon 10632 4月  18 23:04 poemO
-rwxrwxr-x 1 xoomoon xoomoon 10744 4月  18 23:04 poemWall
-rw-rw-r-- 1 xoomoon xoomoon   586 4月  16 11:09 source.c
```

4. 编写C代码检查输入字符串为文件还是目录亦或是别的什么东西
5. 若为文件则打印文件权限，若为文件所有者，则修改权限为777
6. 若既不是文件也不是目录，则进行错误处理
   (1)构建流程图



开始
是否还有文件？
否
结束
是
获取文件信息
是普通文件？
是
输出信息
用户确认与修改文件权限
否
是目录？
是
输出信息
否
出错处理程序

(2)源代码

- file_check.c

```c
#include "file_check.h"
#include <sys/stat.h>
#include <stdio.h>

/*
 *  函数名：main
 *  参数：int 文件路径数
 *        char*[] 文件路径数组
 *  返回值：无
 *  用法：确认每个文件路径所指文件类型并做相应处理
 */
int main(int argv, char* argc[])
{
    for (int i = 1; i < argv; ++i)
    {
        struct stat s;
        if (stat(argc[i], &s))
            error_handle(OPEN_ERROR, argc[i], "", ".\n");
        else if (S_ISREG(s.st_mode))
        {
            printf("%s is a file, ", argc[i]);
            file_mode(argc[i]);
            printf(".\n");
        }
        else if (S_ISDIR(s.st_mode))
            printf("%s is a folder/directory.\n", argc[i]);
        else
            error_handle(OTHER_ERROR, argc[i], "", ".\n");
    }
}
```

- file_mode.c

```c
#include "file_check.h"
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>

/*
 *  函数名: file_mode
 *  参数: const char* 文件路径
 *  返回值: 无
 *  用法: 根据路径信息获取文件并根据文件是否为用户所有来修改权限
 */
void file_mode(const char *pathname)
{
    struct stat s;
    if (stat(pathname, &s))
        error_handle(OPEN_ERROR, pathname, "", ".\n");
    else if (S_ISREG(s.st_mode))
    {
        printf("mode: %o", s.st_mode & 0777);
        if (getuid()==s.st_uid && (s.st_mode & 0777) != 0777)
        {
            chmod(pathname, 0777);
            stat(pathname, &s);
            printf(" => %o", s.st_mode & 0777);
        }
    }
    else
        error_handle(OTHER_ERROR, pathname, "", ".\n");
}
```

- error_handle.c

```c
#include "file_check.h"
#include <stdio.h>

/*  函数名：file_type
 *  参数：int 错误代码
 *        const char *info 错误信息
 *        const char *prefix 信息前缀
 *        const char *suffix 信息后缀
 *  返回值：无
 *  用法：根据错误代码输出错误信息
 */
void error_handle(const int error_code, const char *info,
        const char * prefix, const char * suffix)
{
    printf("%s", prefix);
    switch (error_code)
    {
        case OPEN_ERROR:
            printf("error, %s is a bad pathname.", info);
            break;
        case OTHER_ERROR:
            printf("error, %s is not a file or folder/directory", info);
            break;
    }
    printf("%s", suffix);
}
```

- file_check.h

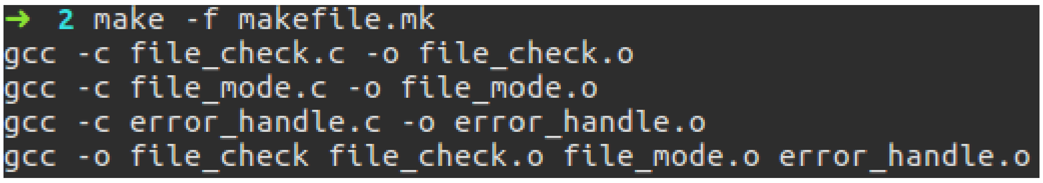
```c
#ifndef _FILE_CHECK_
#define _FILE_CHECK_

#define OPEN_ERROR 1     //路径信息错误
#define OTHER_ERROR 2    //路径为其他类型文件

void file_mode(const char*);
void error_handle(const int, const char*, const char*, const char*);

#endif
```

7. 用makefile成功编译上述三段代码

- makefile文件

```makefile
all: file_check

file_check : file_check.o file_mode.o error_handle.o
    gcc -o file_check file_check.o file_mode.o error_handle.o

file_check.o : file_check.c file_check.h
    gcc -c file_check.c -o file_check.o

file_mode.o : file_mode.c file_check.h
    gcc -c file_mode.c -o file_mode.o

error_handle.o : error_handle.c file_check.h
    gcc -c error_handle.c -o error_handle.o
```

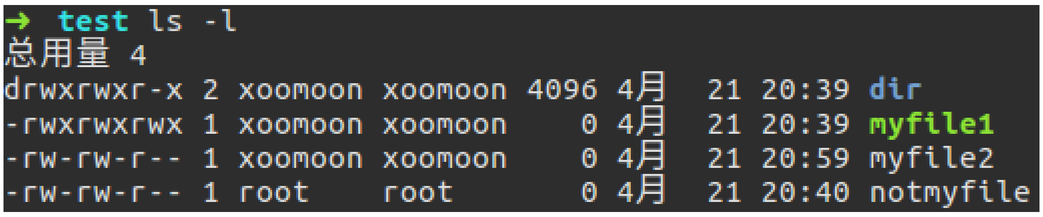
运行结果



```
➜  2 make -f makefile.mk
gcc -c file_check.c -o file_check.o
gcc -c file_mode.c -o file_mode.o
gcc -c error_handle.c -o error_handle.o
gcc -o file_check file_check.o file_mode.o error_handle.o
```

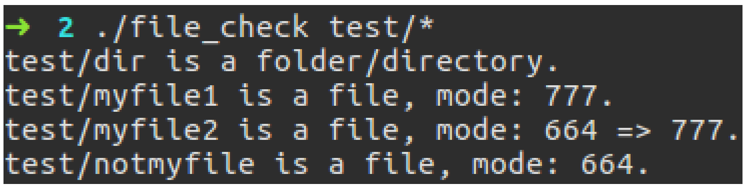
- 测试

  测试文件



```
➜  test ls -l
总用量 4
drwxrwxr-x 2 xoomoon xoomoon 4096 4月  21 20:39 dir
-rwxrwxrwx 1 xoomoon xoomoon    0 4月  21 20:39 myfile1
-rw-rw-r-- 1 xoomoon xoomoon    0 4月  21 20:59 myfile2
-rw-rw-r-- 1 root    root       0 4月  21 20:40 notmyfile
```

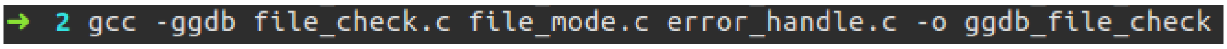
  测试结果



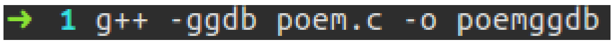
```
➜  2 ./file_check test/*
test/dir is a folder/directory.
test/myfile1 is a file, mode: 777.
test/myfile2 is a file, mode: 664 => 777.
test/notmyfile is a file, mode: 664.
```

8. 将前面的代码用gdb friendly的方式编译
9. 设置断点来观察在第一个文件中的time变量值的变化以及第二个文件中的输入参数
10. 检查代码在哪并打印目标代码目前的栈信息

    (1)生成文件



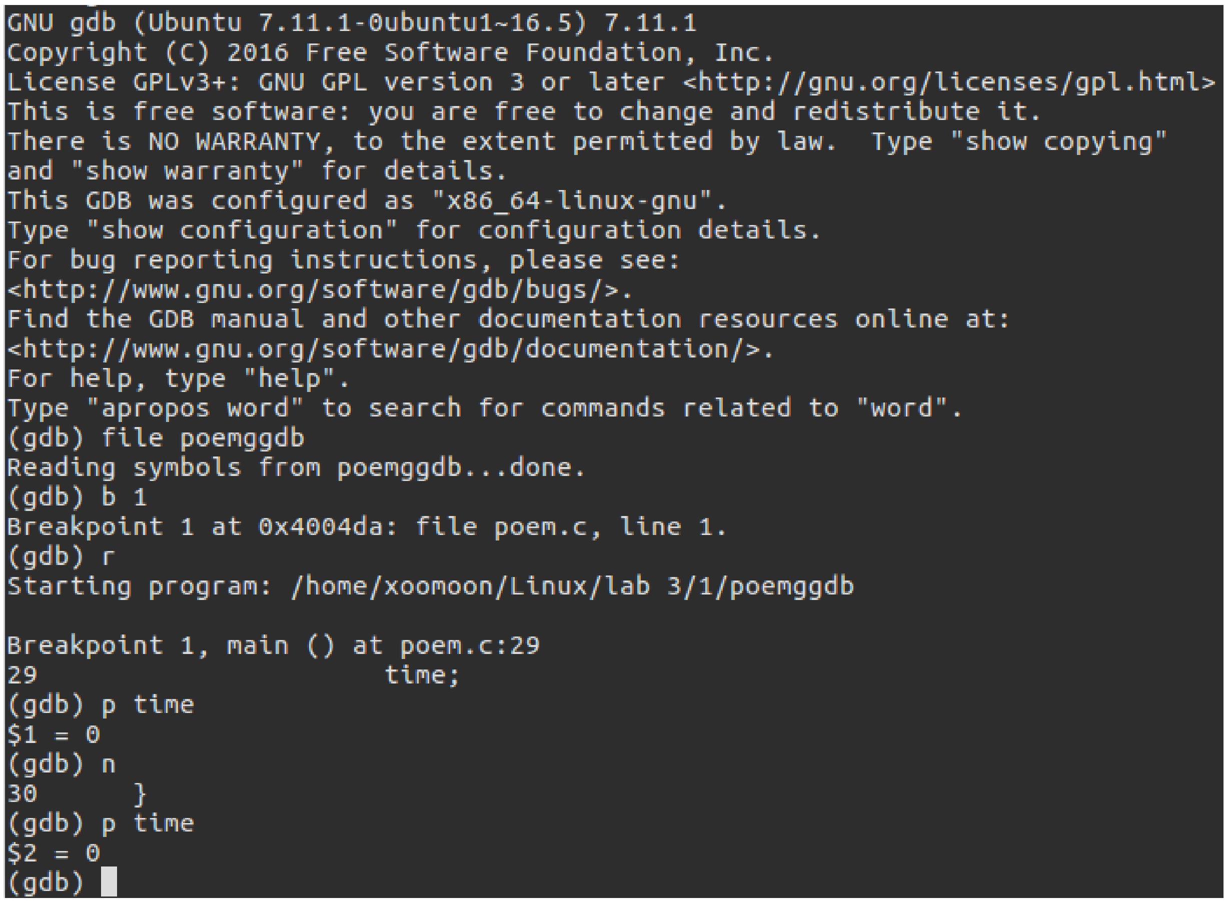
```
➜  2 gcc -ggdb file_check.c file_mode.c error_handle.c -o ggdb_file_check
```



```
➜  1 g++ -ggdb poem.c -o poemggdb
```

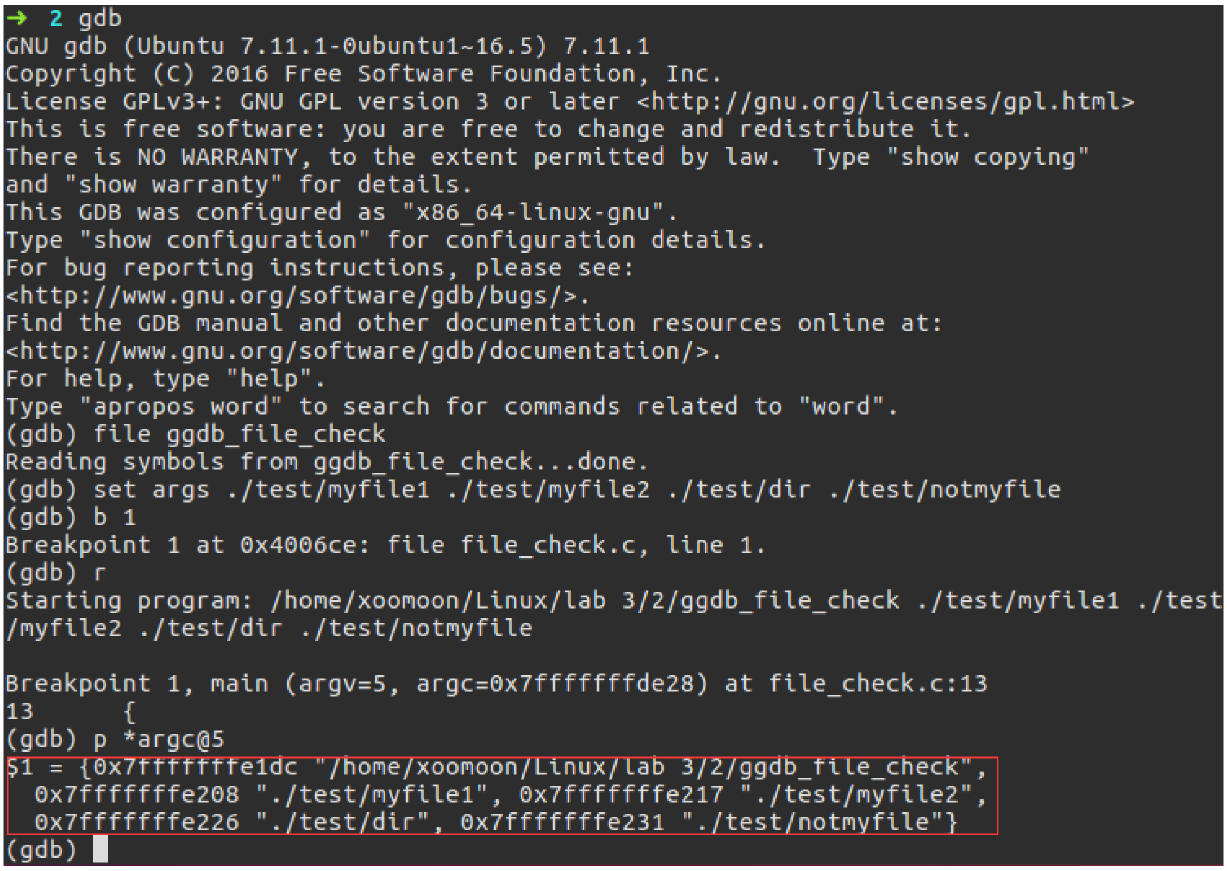
    (2)观察变量

    poemggdb:



```
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) file poemggdb
Reading symbols from poemggdb...done.
(gdb) b 1
Breakpoint 1 at 0x4004da: file poem.c, line 1.
(gdb) r
Starting program: /home/xoomoon/Linux/lab 3/1/poemggdb

Breakpoint 1, main () at poem.c:29
29                      time;
(gdb) p time
$1 = 0
(gdb) n
30      }
(gdb) p time
$2 = 0
(gdb) 
```

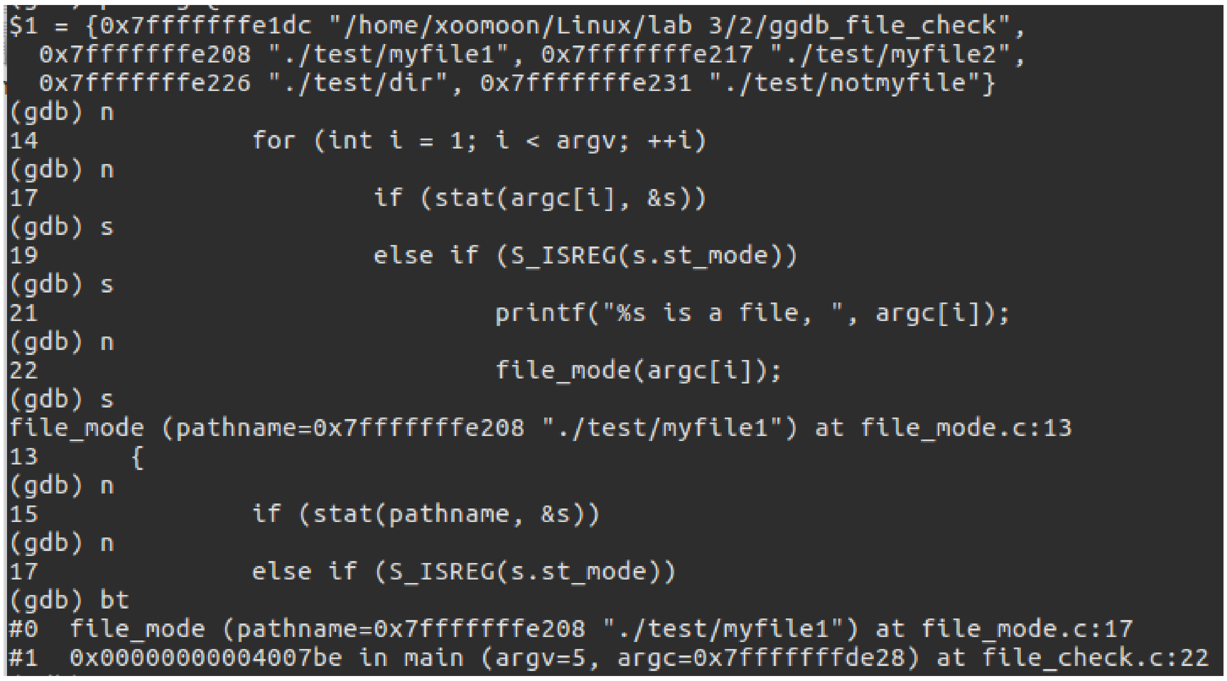
gdb_file_check:



```
➜  2 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) file ggdb_file_check
Reading symbols from ggdb_file_check...done.
(gdb) set args ./test/myfile1 ./test/myfile2 ./test/dir ./test/notmyfile
(gdb) b 1
Breakpoint 1 at 0x4006ce: file file_check.c, line 1.
(gdb) r
Starting program: /home/xoomoon/Linux/lab 3/2/ggdb_file_check ./test/myfile1 ./test
/myfile2 ./test/dir ./test/notmyfile

Breakpoint 1, main (argv=5, argc=0x7fffffffde28) at file_check.c:13
13      {
(gdb) p *argc@5
$1 = {0x7fffffffe1dc "/home/xoomoon/Linux/lab 3/2/ggdb_file_check",
  0x7fffffffe208 "./test/myfile1", 0x7fffffffe217 "./test/myfile2",
  0x7fffffffe226 "./test/dir", 0x7fffffffe231 "./test/notmyfile"}
(gdb) 
```

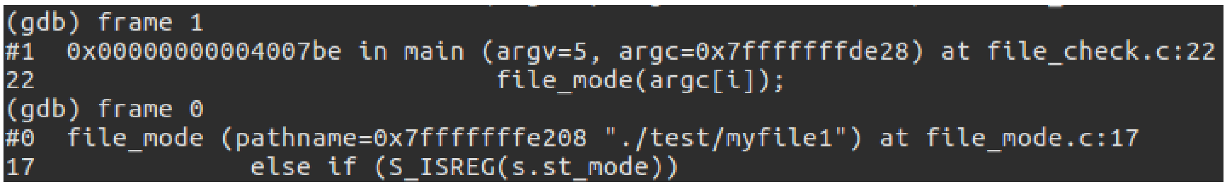
(3)输出栈信息

进入函数：



```
$1 = {0x7fffffffe1dc "/home/xoomoon/Linux/lab 3/2/ggdb_file_check",
  0x7fffffffe208 "./test/myfile1", 0x7fffffffe217 "./test/myfile2",
  0x7fffffffe226 "./test/dir", 0x7fffffffe231 "./test/notmyfile"}
(gdb) n
14              for (int i = 1; i < argv; ++i)
(gdb) n
17                      if (stat(argc[i], &s))
(gdb) s
19                      else if (S_ISREG(s.st_mode))
(gdb) s
21                              printf("%s is a file, ", argc[i]);
(gdb) n
22                              file_mode(argc[i]);
(gdb) s
file_mode (pathname=0x7fffffffe208 "./test/myfile1") at file_mode.c:13
13      {
(gdb) n
15              if (stat(pathname, &s))
(gdb) n
17              else if (S_ISREG(s.st_mode))
(gdb) bt
#0  file_mode (pathname=0x7fffffffe208 "./test/myfile1") at file_mode.c:17
#1  0x00000000004007be in main (argv=5, argc=0x7fffffffde28) at file_check.c:22
```

查看函数信息：



```
(gdb) frame 1
#1  0x00000000004007be in main (argv=5, argc=0x7fffffffde28) at file_check.c:22
22                              file_mode(argc[i]);
(gdb) frame 0
#0  file_mode (pathname=0x7fffffffe208 "./test/myfile1") at file_mode.c:17
17              else if (S_ISREG(s.st_mode))
```

实验思考

实验难度不算大，但比较考验知识的灵活运用。比如第一题，通过观察诗的代码结构，发现了class这个C里头不可能出现的东西，因此推断出这是用C++写的，因此使用g++成功通过了编译。而第二小问则是对于Linux系统的理解，明白了后缀名对于Linux的意义就可以很快解决了。
第二大题基本上是按老师的流程走一遍就可以了。估计是我C比较熟练的缘故，基本上没遇到什么难题，程序很快就编译成功了。
第三题差不多是送分题，只要掌握了基本的gdb技巧就可以搞定。

参考资料

[1] 金国庆，刘加海，季江民，谢井．Linux程序设计（第三版）[M]．杭州：浙江大学出版社，2017.9
[2] Neil Matthew, Richard Stones．Linux程序设计（第4版）[M]．陈健，宋健建，译．北京：人民邮电出版社，2010.6