

江南大学物联网工程学院实验报告

课程名称 《Linux 环境程序设计》 实验名称 实验 5 常用开发工具 实验日期 2017.11.14
班级 计科 1404 姓名 阎覃 学号 1030414414
实验报告要求 1. 实验目的 2. 实验内容 3. 实验步骤 4. 运行情况 5. 实验体会

1 实验目的

- 掌握 C 语言编译的基本用法;
- 掌握 gdb 调试工具的基本用法;
- 理解 make 工具的功能, 学会编制 makefile 的方法。

2 实验内容

- 利用 gcc 命令编译 C 语言程序, 使用不同选项, 观察并分析显示结果
- 用 gdb 命令调试一个编译后的 C 语言程序
- 编写一个由多个文件构成的 C 语言程序, 编制 makefile, 运行 make 工具进行维护

3 实验步骤及运行情况

Problem 1 改写例 6.1, 使用下列选项对它进行编译: -I, -D, -E, -c, -o, -l

Answer

下面分别是 tmp/testI.h 和 hello.c 的源代码:

tmp/testI.h

```
1 #define fatal "abc"
```

hello.c

```
1 #include "testI.h"
2 #define fatal "please call Lawrence for help"
3
4 main()
5 {
6     /*resting CPP options */
7     printf("display -D variable %s \n",DOPTION);
8     printf("display overwirte fatal=%s\n",fatal);
9     printf("Hello,everybody!!\n");
10 }
```

-I 指定头文件的目录

```
1 gcc -I ./tmp hello.c
```

指定 ./tmp 目录后找不到头文件的报错就消失了。

-D 定义一个宏, 若不指定值则为 1

```
1 gcc -I ./tmp -D DOPTION='"testing -D"' -D fatal hello.c
```

这里存在两次重复定义警告, 第一次将 1 重定义为 abc, 第二次为 please call Lawrence for help

-E 只对文件进行预处理，不进行编译、链接，生成的结果送标准输出，即只运行预编译器

```
1 gcc -I ./tmp -D DOPTION='"testing -D"' -E hello.c
```

可以发现原来没定义的 DOPTION 等宏都被替换了。

```
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 330 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "hello.c" 2
# 1 "./tmp/testI.h" 1
# 2 "hello.c" 2
hello.c:2:9: warning: 'fatal' macro redefined [-Wmacro-redefined]
#define fatal "please call Lawrence for help"
      ^
./tmp/testI.h:1:9: note: previous definition is here
#define fatal "abc"
      ^
main()
{
    printf("display -D variable %s \n", "testing -D");
    printf("display overwrite fatal=%s\n", "please call Lawrence for help");
    printf("Hello, everybody!!\n");
}
1 warning generated.
```

图 1: 预编译后的结果

-c 只把源文件编译成目标代码.o

```
1 gcc -I ./tmp -D DOPTION='"testing -D"' -c hello.c
```

```
MacBook-Pro:ex5 ethan$ gcc -I ./tmp -D DOPTION='"testing -D"' -c hello.c
hello.c:2:9: warning: 'fatal' macro redefined [-Wmacro-redefined]
#define fatal "please call Lawrence for help"
      ^
./tmp/testI.h:1:9: note: previous definition is here
#define fatal "abc"
      ^
hello.c:4:1: warning: type specifier missing, defaults to 'int' [-Wimplicit-int]
main()
^
hello.c:7:5: warning: implicitly declaring library function 'printf' with type
'int (const char *, ...)' [-Wimplicit-function-declaration]
    printf("display -D variable %s \n", DOPTION);
    ^
hello.c:7:5: note: include the header <stdio.h> or explicitly provide a
declaration for 'printf'
3 warnings generated.
MacBook-Pro:ex5 ethan$ ls
doc    hello.c hello.o ref    tmp
```

图 2: 编译成目标文件

-o file1 编译成可执行文件 file1。

```
1 gcc -I ./tmp -D DOPTION='"testing -D"' -o abc hello.c
```

-llib 指定程序要链接的库为 lib

```
1 gcc -I ./tmp -D DOPTION='"testing -D"' -llib hello.c
```

```

MacBook-Pro:ex5 ethan$ gcc -I ./tmp -D DOPTION="testing -D" -o abc hello.c
hello.c:2:9: warning: 'fatal' macro redefined [-Wmacro-redefined]
#define fatal "please call Lawrence for help"
      ^
./tmp/testI.h:1:9: note: previous definition is here
#define fatal "abc"
      ^
hello.c:4:1: warning: type specifier missing, defaults to 'int' [-Wimplicit-int]
main()
^
hello.c:7:5: warning: implicitly declaring library function 'printf' with type
'int (const char *, ...)' [-Wimplicit-function-declaration]
printf("display -D variable %s \n",DOPTION);
^
hello.c:7:5: note: include the header <stdio.h> or explicitly provide a
declaration for 'printf'
3 warnings generated.
MacBook-Pro:ex5 ethan$ ls
abc      doc      hello.c  hello.o  ref      tmp
MacBook-Pro:ex5 ethan$ abc
-bash: abc: command not found
MacBook-Pro:ex5 ethan$ ./abc
display -D variable testing -D
display overwrite fatal=please call Lawrence for help
Hello,everybody!!
MacBook-Pro:ex5 ethan$

```

图 3: 编译成可执行文件

Problem 2 完成对思考题 6.5 的调试**Answer**

由于 macOS 没有预装 gdb, 所以要通过 HomeBrew 进行安装。

```
1 brew install gdb
```

程序如图所示

badprog.c

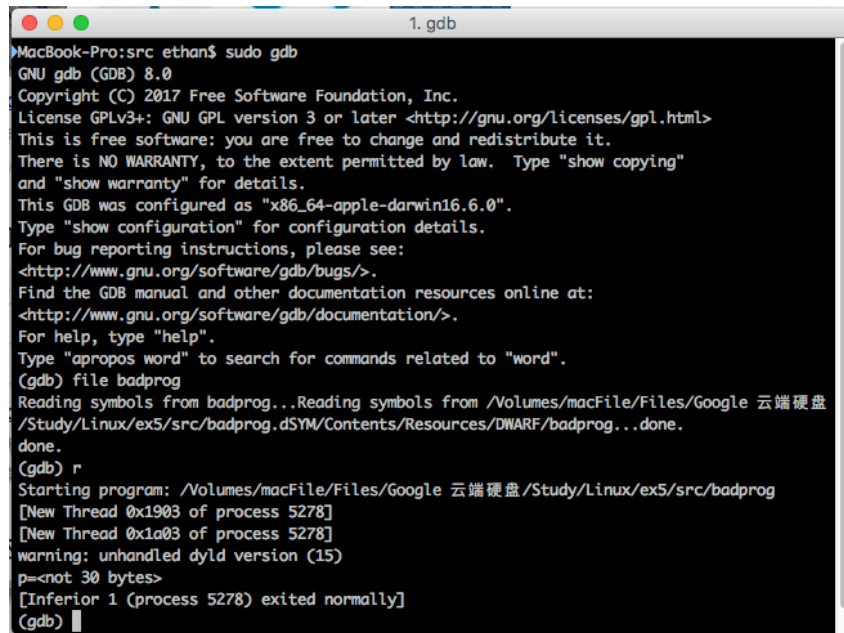
```

1 /*badprog.c 错误地访问内存*/
2 #include <stdio.h>
3 #include <stdlib.h>
4 int main(int argc, char **argv) {
5     char *p;
6     int i;
7     p = malloc(30);
8     strcpy(p, "not 30 bytes");
9
10    if (argc == 2) {
11        if (strcmp(argv[1], "-b") == 0)
12            p[50] = 'a';
13        else if (strcmp(argv[1], "-f") == 0) {
14            free(p);
15            p[0] = 'b';
16        }
17    }
18    printf("p=<%s\n", p);
19    /*free(p)*/
20    return 0;
21 }

```

程序有一个参数, -b 访问越界内存, -f 访问清空后的内存。

运行截图



```

MacBook-Pro:src ethan$ sudo gdb
GNU gdb (GDB) 8.0
Copyright (C) 2017 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-apple-darwin16.6.0".
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 badprog
Reading symbols from badprog...Reading symbols from /Volumes/macFile/Files/Google 云端硬盘/Study/Linux/ex5/src/badprog.dSYM/Contents/Resources/DWARF/badprog...done.
(gdb) r
Starting program: /Volumes/macFile/Files/Google 云端硬盘/Study/Linux/ex5/src/badprog
[New Thread 0x1903 of process 5278]
[New Thread 0x1a03 of process 5278]
warning: unhandled dyld version (15)
p=<not 30 bytes>
[Inferior 1 (process 5278) exited normally]
(gdb)

```

Problem 3 完成对思考题 6.6 的调试

Answer

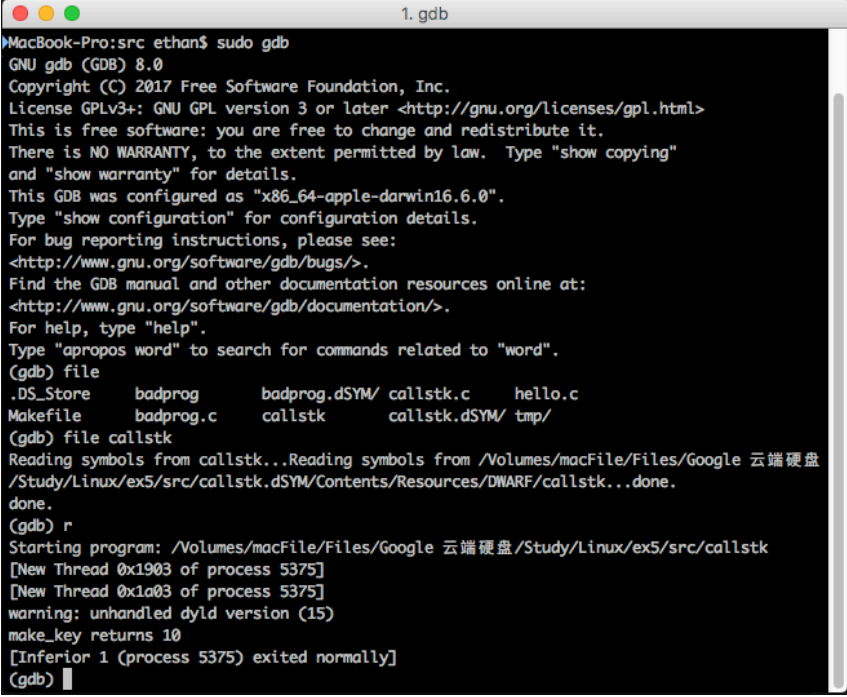
程序如图所示

```

callstk.c
1  /*callstk.c 有3个函数调用深度的调用栈*/
2  #include <stdio.h>
3  #include <stdlib.h>
4  int make_key(void);
5  int get_key_num(void);
6  int number(void);
7
8  int main(void) {
9      int ret = make_key();
10     printf("make_key returns %d\n", ret);
11     exit(EXIT_SUCCESS);
12 }
13
14 int make_key(void){
15     int ret = get_key_num();
16     return ret;
17 }
18
19 int get_key_num(void)
20 {
21     int ret = number();
22     return ret;
23 }
24
25 int number(void){
26     return 10;
27 }

```

运行截图



```

1. gdb
MacBook-Pro:src ethan$ sudo gdb
GNU gdb (GDB) 8.0
Copyright (C) 2017 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-apple-darwin16.6.0".
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
.DS_Store      badprog      badprog.dSYM/ callstk.c      hello.c
Makefile       badprog.c    callstk      callstk.dSYM/ tmp/
(gdb) file callstk
Reading symbols from callstk...Reading symbols from /Volumes/macFile/Files/Google 云端硬盘
/Study/Linux/ex5/src/callstk.dSYM/Contents/Resources/DWARF/callstk...done.
(gdb) r
Starting program: /Volumes/macFile/Files/Google 云端硬盘/Study/Linux/ex5/src/callstk
[New Thread 0x1903 of process 5375]
[New Thread 0x1a03 of process 5375]
warning: unhandled dyld version (15)
make_key returns 10
[Inferior 1 (process 5375) exited normally]
(gdb)

```

Problem 4 完成对思考题 6.9 的编制，并使用 make 命令进行维护

Answer

Makefile 如图所示

Makefile

```

1 prog: a.o b.o c.o d.o assmb.o
2     gcc a.o b.o c.o d.o assmb.o -L/home/user/lib -lm -o prog
3 a.o: a.c
4     gcc -c a.c
5 b.o: b.c defs.h
6     gcc -c b.c
7 c.o: c.c
8     gcc -c c.c
9 d.o: d.c defs.h
10    gcc -c d.c
11 assmb.o: assmb.s
12    as -o assmb.o assmb.s
13 clean:
14    rm prog *.o

```

4 实验体会

虽然很早就会用 `gcc` 编译 `c` 程序,但是从没有过如此详细的了解 `gcc` 的各种参数。过去都是通过 IDE 来开发程序,通过这次学习我了解到 `gcc` 编译器其实支持许多功能,也认识到 IDE 工作的原理,以往在 IDE 中设定的头文件等工作目录是怎么加载的。通过 `gdb` 可以调试程序, `gdb` 的功能十分强大,这次试验也仅仅是一次体验。`make` 是一个很常用的工具,通过编写 `Makefile`,我们只需要 `make` 就可以编译整个工程。

实验报告采用 \LaTeX 编写,代码托管至 GitHub:
<https://github.com/Ethan0w0/JNU-Linux-exp>

教师评价	优		良		中		及格		不及格		教师签名		日期	
------	---	--	---	--	---	--	----	--	-----	--	------	--	----	--