

实验 1

李子龙 518070910095

2021 年 3 月 2 日

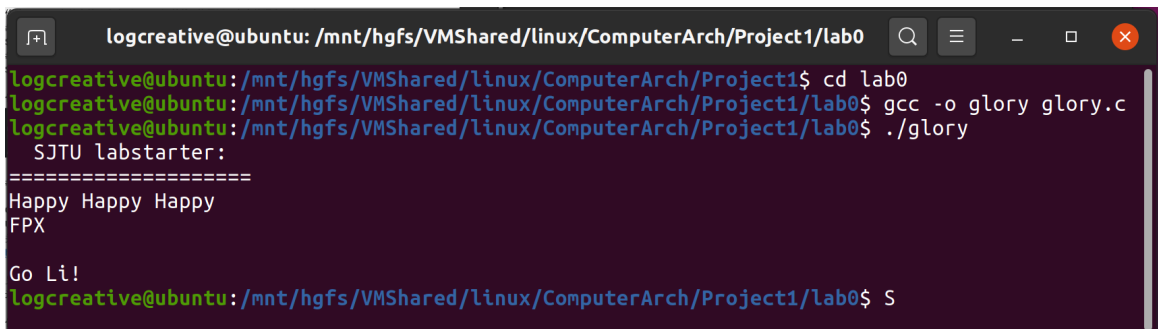
0. 准备操作 解压缩操作

```
1 tar --get -f lab0.tar.gz
```

1. gcc 修改 `glory.c` 宏参数为

```
1 /* Only change any of these 4 values */
2 #define V0 3
3 #define V1 3
4 #define V2 1
5 #define V3 3
```

运行结果如下:



```
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1$ cd lab0
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ gcc -o glory glory.c
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ ./glory
SJTU labstarter:
=====
Happy Happy Happy
FPX

Go Li!
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ S
```

2. gdb

(1) How do you pass command line arguments to a program when using gdb?

答: 在编译时添加 `-g` 的指令, 以期调试。

```
1 gcc -g -o hello hello.c
2 gdb hello
```

- (2) How do you set a breakpoint which only occurs when a set of conditions is true (e.g. when certain variables are a certain value)?

答: 进入 gdb 环境后, 采用如下命令对变量 `ch` 进行监视, 当其变为 `e` 的时候触发第 12 行的断点。这样就设置了条件断点。

```
1 (gdb) break 12 if ch=='e'
```

- (3) How do you execute the next line of C code in the program after stopping at a breakpoint?

答: 使用单步调试命令:

```
1 (gdb) next
```

并且持续回车可以持续单步调试。

- (4) If the next line of code is a function call, you'll execute the whole function call at once if you use your answer to (3). How do you tell GDB that you want to debug the code inside the function instead?

答: 直接传入函数参数作为断点:

```
1 (gdb) break main
```

- (5) How do you resume the program after stopping at a breakpoint?

答: 使用继续命令:

```
1 (gdb) continue
```

并且持续回车可以连续继续。

- (6) How can you see the value of a variable (or even an expression like `1+2`) in gdb?

答: 在设置完断点后, 运行, 中断时使用 `print` 打印 `ch` 或 `ch+1` 的值:

```
1 (gdb) break 12
2 (gdb) run
3 (gdb) print ch
4 $1 = 108 '1'
5 (gdb) print ch+1
6 $2 = 109
7 (gdb) continue
```

- (7) How do you configure gdb so it prints the value of a variable after every step?

答: 在设置完断点后, 运行, 中断时使用 `awatch` 指令监视 `ch` 或 `ch+1` 的值:

```
1 (gdb) run
2 (gdb) awatch ch
3 (gdb) awatch ch+1
4 (gdb) continue
```

就在每次中断后收到如下的信息:

A screenshot of a terminal window with a dark background. The window title is 'logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0'. The GDB prompt is '(gdb)'. The output shows two hardware watchpoint triggers. The first is for watchpoint 2 on variable 'ch', showing an old value of 101 ('e') and a new value of 108 ('l'). The second is for watchpoint 3 on variable 'ch+1', showing an old value of 102 and a new value of 109. Below this, the current execution state is shown: 'main (argc=1, argv=0x7fffffffdea8) at hello.c:11' and the code line '11 for (i = 0; i < strlen(str); i++)'.

(8) How do you print a list of all variables and their values in the current function?

答: 在运行中断后, 使用下述命令打印所有的局部变量:

```
1 (gdb) info locals
2 i = 0
3 str = 0x555555556004 "hello, world!"
4 ch = 0 '\000'
```

(9) How do you exit out of gdb?

答: 使用退出指令:

```
1 (gdb) quit
```

3. 调试

答: 编译完成后进入 `gdb` 调试, 在第 10 行处打下断点, 并监视变量 `a` 和 `b` 的值。

```
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0
(gdb) list
6     } node;
7
8     /* FIXME: this function is buggy. */
9     int ll_equal(const node* a, const node* b) {
10         while (a != NULL) {
11             if (a->val != b->val)
12                 return 0;
13             a = a->next;
14             b = b->next;
15         }
(gdb) awatch a
No symbol "a" in current context.
(gdb) break 10
Breakpoint 1 at 0x1179: file ll_equal.c, line 10.
(gdb) run
Starting program: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0/ll_equal
Breakpoint 1, ll_equal (a=0x7fffffffdd00, b=0x7fffffffdd00) at ll_equal.c:10
10     while (a != NULL) {
(gdb) awatch a
Hardware access (read/write) watchpoint 2: a
(gdb) awatch b
Hardware access (read/write) watchpoint 3: b
(gdb) continue
Continuing.
```

在进入第二轮测试时，出现错误。

```
logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0
11             if (a->val != b->val)
(gdb)
Program received signal SIGSEGV, Segmentation fault.
0x000055555555185 in ll_equal (a=0x7fffffffdd20, b=0x0) at ll_equal.c:11
11             if (a->val != b->val)
(gdb)
Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb)
The program is not being run.
(gdb)
```

错误原因是 `b == NULL`，空指针没有值。所以退出 `gdb`，并在 `Vim` 修改

```
1 vim ll_equal.c
```

更正后的函数如下：

```

1  /* FIXME: this function is buggy. */
2  int ll_equal(const node* a, const node* b) {
3      while (a != NULL && b != NULL) {
4          if (a->val != b->val)
5              return 0;
6          a = a->next;
7          b = b->next;
8      }
9      /* lists are equal if a and b are both null */
10     return a == b;
11 }

```

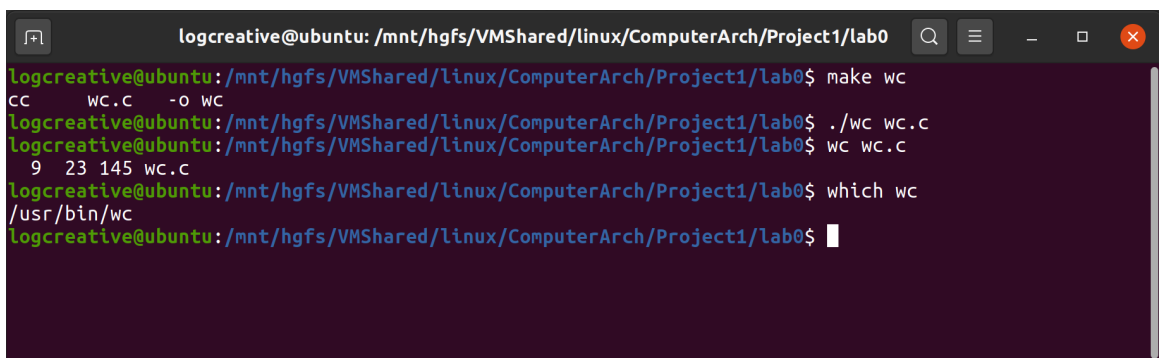
之后就正常运行了：

```

equal test 1 result = 1
equal test 2 result = 0

```

4. make 运行几条指令：



```

logcreative@ubuntu: /mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ make wc
cc      WC.C      -o WC
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ ./WC WC.C
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ wc WC.C
 9 23 145 WC.C
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$ which wc
/usr/bin/wc
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/ComputerArch/Project1/lab0$

```

make 指令编译 `WC.C` 文件，使用 `./WC` 将会运行该文件，而只使用 `wc` 会调用系统指令，输入帮助指令

```
1 wc --help
```

会得到帮助：

Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified.

会打印文件的行数、单词数、文件大小。