

# 《操作系统》实验报告 (OS-Lab1)

中国人民大学 信息学院 崔冠宇 2018202147

## 一、实验题目: Shell 编程.

## 二、实验目的:

- 掌握编写 UNIX/Linux Shell 脚本程序的基本方法;
- 熟悉 Shell 脚本程序的调试方法.

## 三、实验方法:

在Linux 兼容环境 (Mac 环境) 下, 编写 Shell 脚本(见下“程序清单”部分).

脚本的执行思路是: 先检查参数个数, 对用户传入的每一个合法目录参数调用 `enterCompAndExec` 函数. 这个函数进入以参数形式传入的目录, 将目录下——

- 所有以 `.c` 结尾的文件调用 `gcc` 进行编译, 输出程序运行结果或可能发生的编译错误;
- 所有子目录递归调用自身.

为了测试程序的健壮性, 我们在 `TestCode` 文件夹下按下图放置了一些文件和目录(蓝底).

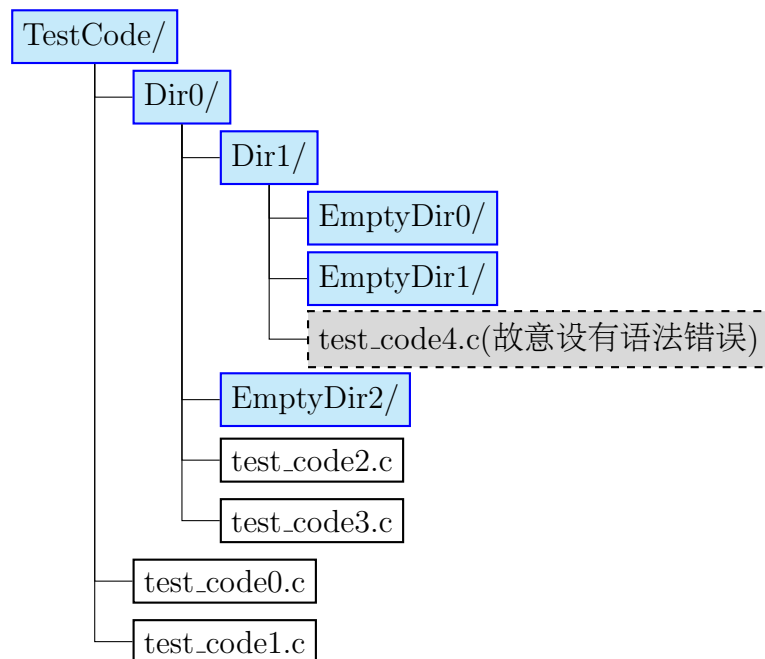


Figure 1: 测试目录下的文件结构

## 四、程序清单

### 1. 本次实验的主角, Shell 脚本 —— **Lab1.sh:**

```
1  #! /bin/bash
2
3  function enterCompAndExec(){
4      # Change dir
5      cd "$1"
6      # Iterate each file
7      for FILE in *
8      do
9          # If it's a .c file
10         if [ "${FILE##*.}" = "c" ]; then
11             # Drop .c, create execName
12             execName="${FILE%*.c*}"
13             instr="gcc $FILE -o $execName"
14             # Compile and get result
15             '$instr'
16             compResult=$?
17             # Error occurs
18             if [ $compResult -ne 0 ]; then
19                 echo "Compile error of file: 'pwd'/$FILE"
20                 echo
21             # Run and get result
22             else
23                 echo "Running $execName in 'pwd':"
24                 instr="./$execName"
25                 runResult='$instr'
26                 echo "$runResult"
27                 echo
28             fi
29         fi
30         # Iterate all directories
31         # and recursively enter-compile-and-execute
32         if [ -d "$FILE" ]; then
33             enterCompAndExec "$FILE"
```

```

34         cd ..
35     fi
36 done
37 }
38
39 # Check parameter count
40 if [ "$#" -eq 0 ]; then
41     echo "Lack parameter!"
42     echo "You need to pass at least one parameter."
43     exit 1
44 fi
45 # Iterate all the parameters and call enterCompAndExec
46 for DIR in "$*"
47 do
48     # Validate the directory's existence
49     if [ -d "$DIR" ]; then
50         enterCompAndExec "$DIR"
51     else
52         echo "Directory $DIR doesn't exist!"
53         echo
54     fi
55 done

```

## 2. 一个C语言测试文件 —— **test\_code0.c**:

```

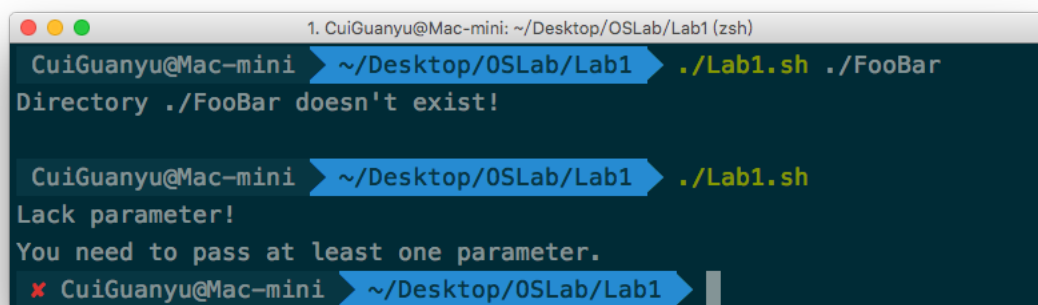
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     printf("Hello,world!\n");
6     printf("This is test_code0.c\n");
7     return 0;
8 }

```

3. 其它 C 测试文件与 test\_code0.c 相似, 仅 printf 内容略有区别, 以及 test\_code4.c 第五行行末故意缺失分号. 此处不再列出.

## 五、实验结果

我们先用不合法的参数(不传参/传不存在的路径)来测试脚本主过程的正确性.



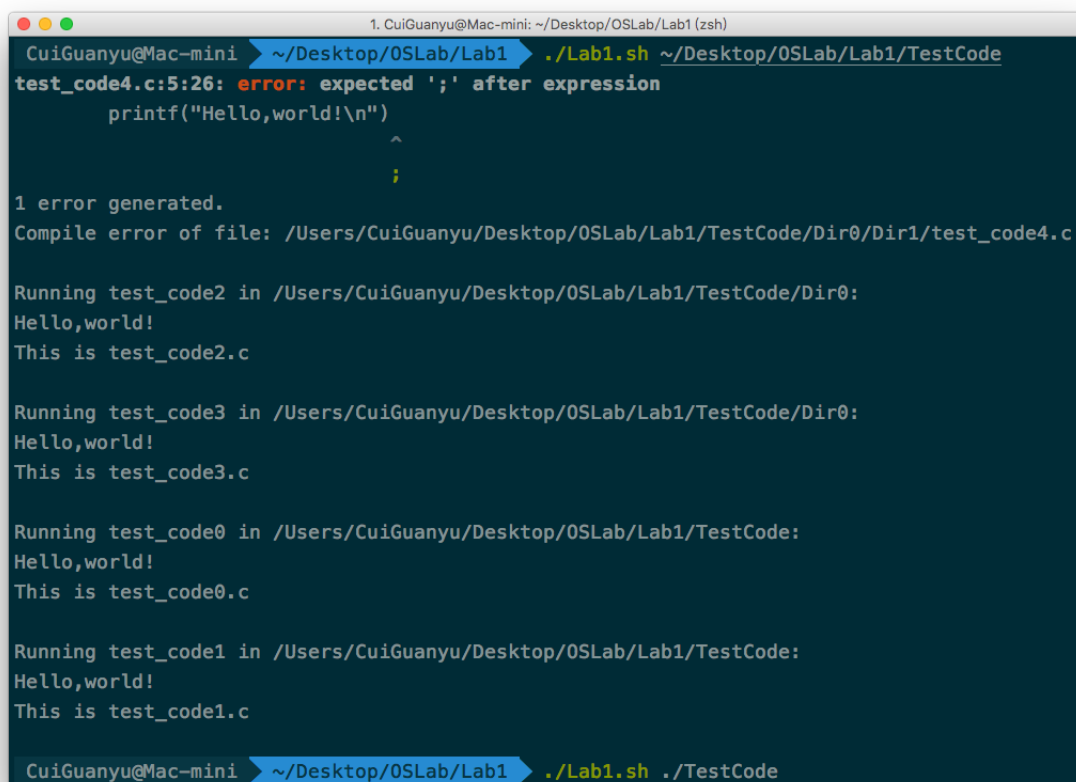
```
1. CuiGuanyu@Mac-mini: ~/Desktop/OSLab/Lab1 (zsh)
CuiGuanyu@Mac-mini ~/Desktop/OSLab/Lab1 ➤ ./Lab1.sh ./FooBar
Directory ./FooBar doesn't exist!

CuiGuanyu@Mac-mini ~/Desktop/OSLab/Lab1 ➤ ./Lab1.sh
Lack parameter!
You need to pass at least one parameter.
✗ CuiGuanyu@Mac-mini ~/Desktop/OSLab/Lab1 ➤
```

Figure 2: 错误传参运行结果.

可见脚本正确识别了用户传入的不存在的目录以及缺少参数的情况.

为保证程序某种程度上的“幂等性”(即多次运行结果相同), 我选择将脚本运行两次, 观察两次结果是否正确且一致. 下图是第一次将 ./TestCode 传入脚本的运行结果:



```
1. CuiGuanyu@Mac-mini: ~/Desktop/OSLab/Lab1 (zsh)
CuiGuanyu@Mac-mini ~/Desktop/OSLab/Lab1 ➤ ./Lab1.sh ~/Desktop/OSLab/Lab1/TestCode
test_code4.c:5:26: error: expected ';' after expression
    printf("Hello,world!\n")
                        ^
                        ;
1 error generated.
Compile error of file: /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode/Dir0/Dir1/test_code4.c

Running test_code2 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode/Dir0:
Hello,world!
This is test_code2.c

Running test_code3 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode/Dir0:
Hello,world!
This is test_code3.c

Running test_code0 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode:
Hello,world!
This is test_code0.c

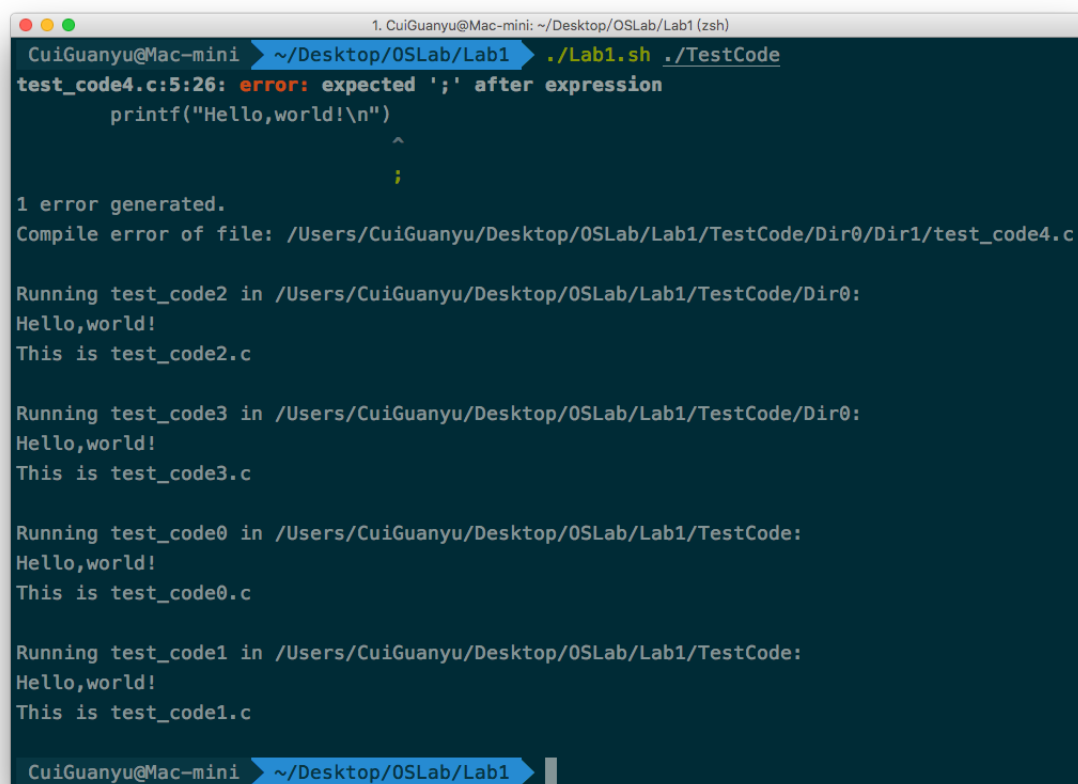
Running test_code1 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode:
Hello,world!
This is test_code1.c

CuiGuanyu@Mac-mini ~/Desktop/OSLab/Lab1 ➤ ./Lab1.sh ./TestCode
```

Figure 3: 第一次运行结果.

我们可以看出, 脚本实现了我们的想法, 能够正确的识别用户给定的目录(包含子目录)中的所有 C 语言文件并将它们编译执行, 而且脚本针对编译错误的文件也给出了相应提示.

再次运行脚本(此时目录下已经存在可执行文件), 如下图所示. 可以看出执行结果与第一次相同, 与我们的预期相同.



```
1. CuiGuanyu@Mac-mini: ~/Desktop/OSLab/Lab1 (zsh)
CuiGuanyu@Mac-mini > ~/Desktop/OSLab/Lab1 > ./Lab1.sh ./TestCode
test_code4.c:5:26: error: expected ';' after expression
    printf("Hello,world!\n")
                        ^
                        ;
1 error generated.
Compile error of file: /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode/Dir0/Dir1/test_code4.c

Running test_code2 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode/Dir0:
Hello,world!
This is test_code2.c

Running test_code3 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode/Dir0:
Hello,world!
This is test_code3.c

Running test_code0 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode:
Hello,world!
This is test_code0.c

Running test_code1 in /Users/CuiGuanyu/Desktop/OSLab/Lab1/TestCode:
Hello,world!
This is test_code1.c

CuiGuanyu@Mac-mini > ~/Desktop/OSLab/Lab1 >
```

Figure 4: 第二次运行结果.

## 六、问题分析

### 1. 针对“可能出现的问题”的处理思路

(a) Q: 如果指定的目录下存在子目录, 如何处理?

A: 对用户传入的目录下出现的子目录进行递归操作.

(b) Q: 编译有错, 如何处理?

A: 将gcc的错误信息以及文件所在目录输出, 但不停止脚本.

(c) Q: 如果有非 .c 文件存在, 如何处理?

A: 如果不是编译出的可执行文件, 则忽略这个文件.

### 2. 实验中遇到的问题及解决方法

- (a) 由于是第一次编写 Shell 脚本, 我对语法还不是很熟悉, 所以写起脚本来不如 C/C++ 顺畅, 经常有不知道如何表达的情况出现. 为了解决这个问题, 我一边写脚本一边在网上查不会的语法, 算是勉强完成了脚本的编写.
- (b) 我的脚本初期也有许多的问题, 比如在目录跳转、递归、空格等地方踩了一些坑, 最终还是靠一步步调试慢慢改掉了这些bug. 这里多亏我使用的环境可以比较方便的单步调试和显示变量(如下图), 帮了我一些忙.

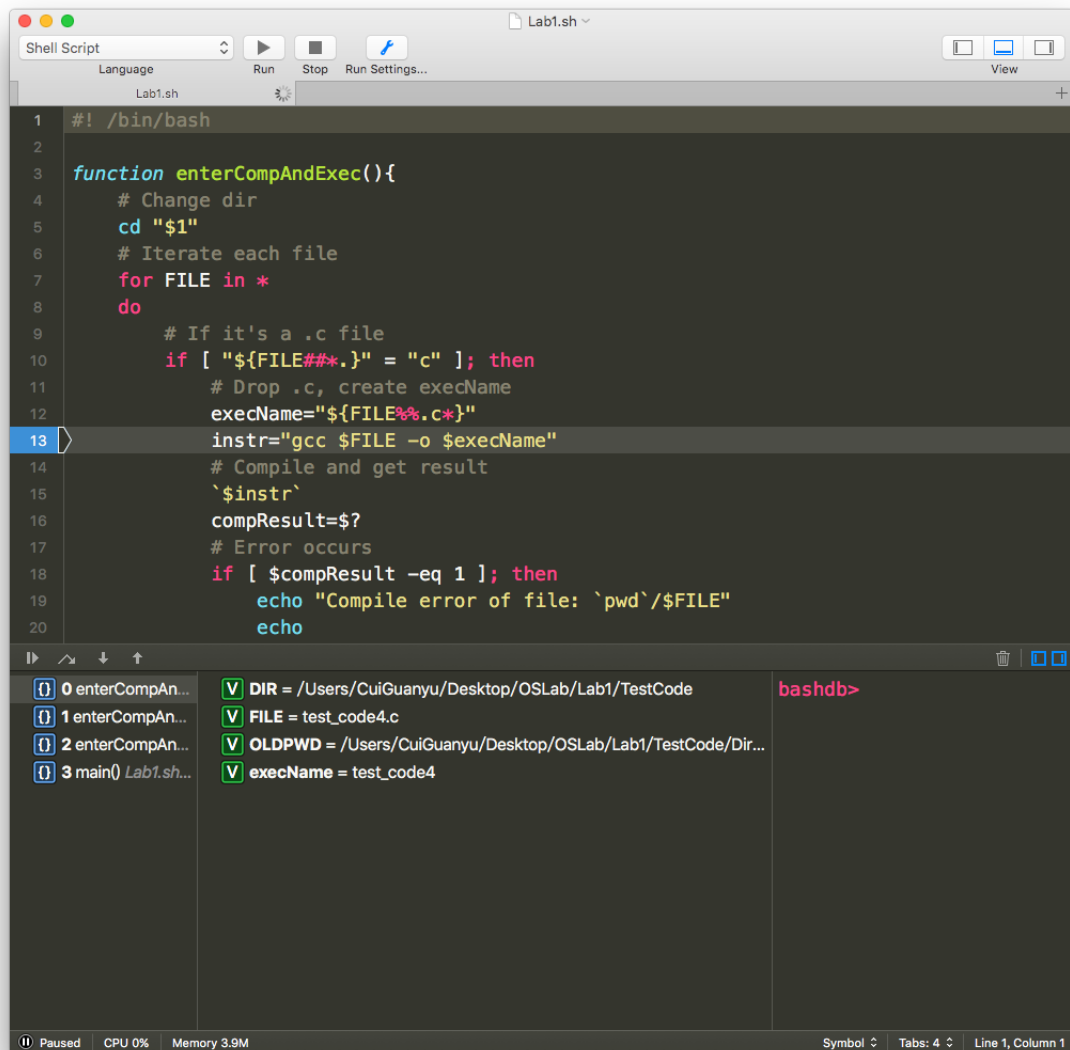


Figure 5: 编写、调试环境

- (c) 目前来看, 脚本可以基本完成实验要求, 但还是有不足的地方. 比如说用户不能选择是否要对给定目录下的子目录进行递归操作. 对此我也想了一些解决方案, 比如模仿Linux下的其它命令, 加一个 -R 开关来控制对每个目录是否要进行递归操作等. 这里因为我对语法还不熟练, 就没有具体实现.