实验四 Shell 程序设计

学号：515030910067 姓名：杨超琪 日期：2018.6.12

一、实验题目

1. 编写一个 Shell 程序 findit，该程序搜索参数 1 指定的目录树，查找所有的以.c 和.h 结尾的文件，如文件行中含有参数指定的字符串，显示该行和相应的文件名。如目录参数 1 缺省，则从当前目录中搜索。如：findit /home/wang/work searchstring

搜索以/home/wang/work 为根的目录树中的 c 程序和头文件，查找含有 searchstring 字符串的行，显示文件名。

findit searchstring

从当前目录开始搜索。

2. 编一 Shell 程序，以类似书本的目录结构的形式，按层次输出当前目录树中的所有目录和文件，要求每一层缩进 4 个空格。

二、算法思想

1. 搜索文件

首先按照题目要求，有两种参数规则：（1）只有匹配的字符串，则默认把当前目录作为搜索目录；（2）有目录与匹配的字符串。这两种情况可以通过$#来判断参数数目，并利用 if 与 test 语法将其归一，统一处理。

然后从根目录出发，遍历访问文件夹中的文件，如果不是目录文件，则不处理，如果是目录文件，则直接打印文件夹名称，然后使用 grep 加 *.c *.h 命令匹配文件夹下所有的符合要求的文件。并使用子进程递归访问子文件夹，做相同的操作。

同时，如果用户采用了非法的格式输入，程序将会提示，给出正确的输入格式。

2. 打印目录树

打印目录树的程序可以在上述的程序的基础上进行修改。原理是一致的，递归访问文件夹到子文件夹，然后逐一使用 echo 打印出目录名与文件名即可。

这的要求是按照每一级目录进行 4 位空格缩进。应对这个问题，我们的策略是定义一个全局的参数，用于输出前缀的空格，每进入一级目录则增加四个空格，在使用 echo 命令的时候在前面先输出空格即可完成要求。

三、运行说明

在这里，我们写的是.sh 脚本，将题目要求的 shell 函数写在脚本内，之后在脚本最后调用该函数，运行脚本，如：bash soluton1.sh xxx 即可运行产生相同的效果。

因为我发现在直接写成函数的时候，没有等到}出现函数即被中断，不是很方便，因此直接使用 shell 脚本来完成该实验。

四、测试方法与分析

1. 搜索文件

我们搜索文件的 shell 脚本叫做 solution1。

（1）当前目录搜索

首先我们搜索了当前目录下的，带有关键词“if”的所有行，使用 bash solution1 “if”的命令。省略了第一个路径参数，默认为当前路径。

```
linuxvirtualpc1@ubuntu:~/test$ bash solution1 "if"
.
./1.c:2:if you are me
./client.c:11:   if(FileOpen){
./server.c:8:    if((qid=msgget(MSGKEY,IPC_CREAT|0666))==-1)    // get the message ID
./server.c:12:       if(FileOpen){
./share_memory.c:19:    if(semop(sid,&sb,1) == -1) {
./share_memory.c:33:            if((sid=semget(key,1,0666|IPC_CREAT))==-1){ // 关键词为1的信号灯组，权限0666
./share_memory.c:37:            if(semctl(sid,0,SETVAL,arg)==-1){ // 如果失败将会返回-1
./share_memory.c:59:    if((segid=shmget(SHMKEY,SIZE, IPC_CREAT|0666))==-1) { // 共享内存申请失败
./share_memory.c:66:    if(!fork()){
./share_memory.c:69:                    if(FileOpen1){
./share_memory.c:85:                    if(FileOpen2){
./test1.c:13:   if (pid=fork()) {// the father process
./test1.c:26:           printf("excel error.\n");  // if some error happens.
test_test
```

（2）其他路径搜索

接着我们搜索了../test_1 目录下的，带有关键词“if”的所有行，使用 bash solution1 ../test_1 “if”的命令。所参数都是全的。

```
linuxvirtualpc1@ubuntu:~/test$ bash solution1 ../test_1 "if"
../test_1
../test_1/client.c:11:   if(FileOpen){
../test_1/server.c:8:    if((qid=msgget(MSGKEY,IPC_CREAT|0666))==-1)    // get the message ID
../test_1/server.c:12:       if(FileOpen){
../test_1/share_memory.c:19:    if(semop(sid,&sb,1) == -1) {
../test_1/share_memory.c:33:            if((sid=semget(key,1,0666|IPC_CREAT))==-1){ // 关键词为1的信号灯组，权限0666
../test_1/share_memory.c:37:            if(semctl(sid,0,SETVAL,arg)==-1){ // 如果失败将会返回-1
../test_1/share_memory.c:59:    if((segid=shmget(SHMKEY,SIZE, IPC_CREAT|0666))==-1) { // 共享内存申请失败
../test_1/share_memory.c:66:    if(!fork()){
../test_1/share_memory.c:69:                    if(FileOpen1){
../test_1/share_memory.c:85:                    if(FileOpen2){
../test_1/test1.c:13:   if (pid=fork()) {// the father process
../test_1/test1.c:26:           printf("excel error.\n");  // if some error happens.
test_1_q
test_1_q/server.c:8:    if((qid=msgget(MSGKEY,IPC_CREAT|0666))==-1)    // get the message ID
test_1_q/server.c:12:       if(FileOpen){
test_1_q/share_memory.c:19:     if(semop(sid,&sb,1) == -1) {
test_1_q/share_memory.c:33:             if((sid=semget(key,1,0666|IPC_CREAT))==-1){ // 关键词为1的信号灯组，权限0666
test_1_q/share_memory.c:37:             if(semctl(sid,0,SETVAL,arg)==-1){ // 如果失败将会返回-1
test_1_q/share_memory.c:59:     if((segid=shmget(SHMKEY,SIZE, IPC_CREAT|0666))==-1) { // 共享内存申请失败
test_1_q/share_memory.c:66:     if(!fork()){
test_1_q/share_memory.c:69:                     if(FileOpen1){
test_1_q/share_memory.c:85:                     if(FileOpen2){
```

可以看到，在两个测试下，我们的程序都完成了要求，测试通过。

2. 打印目录树

我们又测试了打印目录树的程序，在这程序中，我们的函数没有任何的参数，直接打印当前目录下的所有文件，并按照缩进进行分级。

输入命令：bash soluton2

```
linuxvirtualpc1@ubuntu:~/test$ bash solution2
2.h
a
a.out
client
input2.txt
input.txt
msgcom.h
output2.txt
output.txt
server
share_memory
solution1
solution2
s_test
----1.c
----client.c
----server.c
----share_memory.c
----s_s_tes
--------1.c
--------client.c
--------server.c
--------share_memory.c
--------test1.c
--------test4.c
----test1
----test1.c
----test2.sh
----test3
----test4.c
test1
test2.sh
test3
test_test
----client.c
----file1.c
----msgcom.h
----server.c
```

可见，按照了要求列出了所有的文件，并且按照格式每一级缩进 4 格，测试成功。要说明的一点是，在这里我们使用“-”符号代替了空格完成了任务，因为在 shell 中，如果用空格\制表符等空串作为变量的话，会被偶人忽略，因此在这里使用“-”进行了替代，我会在实验思考中详细说明。

五、程序及测试的改进与体会

1、改进与不足：

1）在这里为了方便，我直接使用了 grep 命令进行文件内的字符串匹配查找，比较快。由于时间问题，如果时间充足的话，我可能会使用其他更加底层的语言，比如用流文件命令打开每一个文件，用 read 函数一行行地读来进行匹配，这样的话，可能会让我对 shell 命令有个更全面，更深刻的掌握与理解。

2）程序中之前遇到一个问题，在使用 gerp string –n source 命令进行匹配的时候，如果程序中找不到 source 文件的话，会报错。比如在程序中我使用了 grep $2 –n $i/*.c $i/*.h 进行 c 程序与头文件的匹配，但是如果文件夹中没有 c 与 h 文件的时候，程序就会出错 no such file。之后是按照网上博客上所说，加了-s 的命令以后就不再报错了。但是我认为肯定会有更好的处理机制。不用先尝试 grep，而是先选对文件类型，再进行匹配，效率应该会高很多。但是在经过几次尝试之后，始终无法判断出文件的类型。

```
file=”thisfile.txt”
echo “filename: ${file%.*}”
echo “extension: ${file##*.}”
输出：
filename: thisfile
extension: txt
附：
```

比如这是[1]中博客上提到的获取文件后缀的方式，但是在我的 shell 命令行中运行不起来，不知道为什么。如果可以做到先找到 c 或者 h 的文件，再匹配应该效率会好很多。

3）在第二个实验中，我使用了“-”替代了空格，因为尝试了使用空格，使用制表符，均被 shell 忽略，未能起到效果。这可能就是我的程序存在的不足之处了。问了其他同学，了解到可以用 echo 来直接打印 4 个空格，而不应该存在 shell 变量中，由于时间原因，就不修改了。

2、体会

这次实验本来觉得会比较简单，因为自己比较熟悉 linux 的操作命令，但是发现实验要求的是写 shell 指令，从未在 linux 命令行中跑过循环，更没有使用（）调用过子进程。这次实验进度是相当的缓慢的，一开始有点眼高手低，想通过各种博客的东平西凑来完成实验，最终发现无从下手，有点棘手。

因此还是采取了老老实实的方式。先仔细地复习了一遍书上 shell 的内容，并用虚拟机完整地敲了一边代码，有些感觉了，之后才一条一条命令拼凑起来，完成了整个实验的代码，虽然代码量很小，但是指令的参数确实比较麻烦，需要从各种博客及网络资源上去现场学习。

这次实验让我体会到了第一次写 shell 的感觉，第一次与 linux 的底层打交道，在程序运行成功之后，感觉还是满惊喜的。

六、源代码及其注释

Solution1:

```
findit()
{
    if [ $# -eq 2 ]                                  # 参数完全
    then
        for i in $*                                  # 循环该目录下的文件
        do
            if [ -d $i ]                             # 如果文件是目录文件
            then
                echo $i                              # 打印目录的名称
                grep $2 -s -n $i/*.c $i/*.h          # grep 函数进行字符串匹配
                (cd $i                               # 调用子进程递归进行操作
                for j in *; do
                    findit $j $2
                done)
            fi
        done
    elif [ $# -eq 1 ]                                # 如果只带一个参数
    then
        findit . $2                                  # 补全成两个参数的形式
    else
    then
        echo "please use the format: 'findit dir string'"
```

```
    fi
}
findit $1 $2
```

Solution2:

```
space=""                                    # 保存需要输出的“-”
category()
{
    for i in *; do                          # 循环遍历当前目录所有文件
        if [ -d $i ]; then                  # 如果当前文件是个目录
            echo $space$i                   # 打印该目录的名称
            (cd $i                          # 调用子进程递归操作
             space="----$space"             # 进入一层前置“-”加四个
             category)
        else
            echo -e $space$i                # 返回文件名
        fi
    done
}
category
```

七、参考文献

[1] Shell 字符串处理、获取文件名和后缀名
https://blog.csdn.net/guojin08/article/details/38704823
[2] Shell 编程：Bash 空格的那点事
https://www.jb51.net/article/60328.htm
[3] shell 脚本有关空格语法注意事项
https://blog.csdn.net/xin_9412/article/details/52755772