
实验五 Shell 程序设计实验报告

1 程序设计一

1.1 设计要求

编写一个 Shell 程序 `findit`，该程序搜索参数 1 指定的目录树，查找所有以 `.c` 和 `.h` 为结尾的文件，如果该文件行中含有参数 2 指定的字符串，则显示该行和相应的文件名。如果目录参数 1 缺省，则从当前目录中搜索。

1.2 示例

- 命令 `findit /home/wang/work searchstring`，将搜索以 `/home/wang/work` 为根的目录树中所有的 `c` 程序文件和头文件，并查找含有 `searchstring` 字符串的行、显示文件名。
- 命令 `findit searchstring`，将从当前目录开始搜索所有 `c` 程序文件和头文件，查找含有 `searchstring` 字符串的行并显示文件名。

1.3 设计思路

首先，通过查阅资料得知，在 Linux 系统中可以使用 `find` 和 `grep` 命令进行查找。其中，`find` 命令主要对文件进行查找，可以使用 `-name` 参数指定需要查找的文件名，并且可以使用通配符；而 `grep` 命令主要对字符串进行查找，可以通过 `-H` 参数显示对应的文件名称，通过 `-n` 参数显示对应的行号。显然，可以结合这两个命令完成题目要求的功能。

此外，由于查询时需要至少输入一个参数作为查找的字符串，因此应该考虑用户输入时可能出现的多种情况：输入参数过少、输入参数过多、仅输入一个字符串参数和完整地输入两个参数。每一种情况需要有其对应的不同处理和响应方法。值得注意的是，后两种虽然都需要用到用户传入的参数，但是在程序中 `$1` 指代的是不同作用的参数，且仅有一个字符串参数时程序中不能出现 `$2` 参数。

1.4 设计代码

最终的程序设计代码如下所示：

```

1  #!/bin/bash
2
3  if [ $# -lt 1 ]
4  then
5      echo "Please enter at least one string \
6          as a search parameter"
7      exit
8  fi
9
10 if [ $# -gt 2 ]
11 then
12     echo "Enter up to two strings"
13     exit
14 fi
15
16 if [ $# -eq 1 ]
17 then
18     path=./
19     searching=$1
20 else
21     path=$1
22     searching=$2
23 fi
24
25 # for testing
26 # echo $path
27 # echo $searching
28
29 find $path -name "*.ch" -exec grep -Hn $searching {} \;

```

1.5 运行结果

首先我们在 `/home/zyr/os_homework` 文件夹下创建一个 `a.h` 文件，一个 `b.h` 文件。除此以外，在 `test1` 文件夹下新建一个 `b.h` 文件，在 `test2` 文件夹下新建一个 `a.c` 文件和 `hello.txt` 文件，其中的内容分别如图1、2、3、4所示。

```

zyr@ubuntu:~/os_homework$ cat a.h
#include <stdio.h>

```

图 1: a.h 文件内容

```
zyr@ubuntu:~/os_homework$ cat ./test2/a.c
#include <stdlib.h>

int main()
{
    return 0;
}
```

图 2: a.c 文件内容

```
zyr@ubuntu:~/os_homework$ cat ./test1/b.h
#include <stdlib.h>
#include <stdio.h>

void nothing(int a, int b);
```

图 3: b.h 文件内容

```
zyr@ubuntu:~/os_homework$ cat b.c
#include <stdlib.h>
```

图 4: b.c 文件内容

然后我们运行命令 `./findit.sh /home/zyr/os_homework/test1 include`，程序运行结果如图5所示：

```
zyr@ubuntu:~/os_homework$ ./findit.sh /home/zyr/os_homework/test1 include
/home/zyr/os_homework/test1/b.h:1:#include <stdlib.h>
/home/zyr/os_homework/test1/b.h:2:#include <stdio.h>
```

图 5: 两个输入参数时 findit.sh 的输出结果

运行命令 `./findit.sh include`，程序运行结果如图6所示：

```
zyr@ubuntu:~/os_homework$ sudo ./findit.sh include
./b.c:1:#include <stdlib.h>
./test1/b.h:1:#include <stdlib.h>
./test1/b.h:2:#include <stdio.h>
./a.h:1:#include <stdio.h>
./test2/a.c:1:#include <stdlib.h>
```

图 6: 一个输入参数时 findit.sh 的输出结果

之后进行非法输入，运行命令 `./findit.sh`，其结果如图7所示：

```
zyr@ubuntu:~/os_homework$ ./findit.sh
Please enter at least one string as a serch parameter!
```

图 7: 输入参数过少时 findit.sh 的输出结果

运行命令 `./findit.sh /home/zyr include xxx`, 最终结果如图8所示:

```
zyr@ubuntu:~/os_homework$ ./findit.sh /home/zyr include xxx
Enter up to two strings!
```

图 8: 输入参数过多时 findit.sh 的输出结果

可见, 该程序能够完成指定的任务, 且对于异常输入有较好的处理。

在进行异常输入处理后, 本想检测当输入为两个参数时, 参数顺序是否正确等, 但是考虑这两个参数的差异并不大, 实现起来较为困难; 而且实际应用中许多 shell 程序在接收错误输入后, 最终也仅仅是返回错误的运行结果, 因此没有对程序做进一步的改进。

2 程序设计二

2.1 设计要求

编写一个 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, 要求每一层缩进四个空格。

2.2 设计思路

由于需要按层次进行输出, 因此自然想到使用递归进行设计。经过查询资料发现, 可以通过 `-d` 和 `-f` 参数来判断当前的文件是一个文件还是一个目录。因此只需要在文件是目录时, 先缩进 4 个空格, 然后将此文件夹路径作为参数传递给函数进行递归调用; 而在文件是一个文件时, 直接使用 `echo` 将文件名在屏幕上打印出来即可。除此以外, 在第一次调用函数时, 可以使用 `pwd` 来获取当前的文件夹路径。

2.3 设计代码

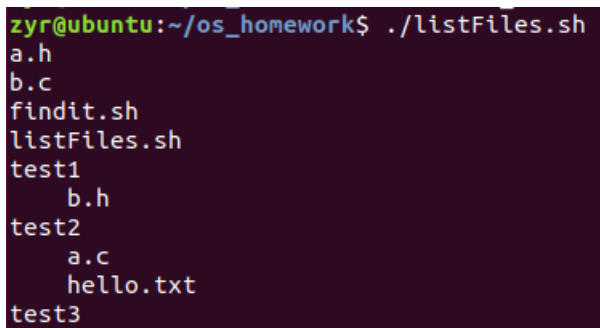
最终的程序设计代码如下所示:

```
1 function listFiles()
2 {
3     for file in ls $1
4     do
5         if [ -d "$1/$file" ]; then
6             echo "$2$file"
7             listFiles "$1/$file" "$2"
8         else
```

```
9         echo "$2$file"
10     fi
11 done
12 }
13 path=pwd
14 listFiles $path ""
```

2.4 运行结果

在 `/home/zyr/os_homework` 文件夹下运行结果如图9所示：



```
zyr@ubuntu:~/os_homework$ ./listFiles.sh
a.h
b.c
findit.sh
listFiles.sh
test1
    b.h
test2
    a.c
    hello.txt
test3
```

图 9: listFiles 输出结果

3 心得体会

通过本次程序设计，我对于 Linux 系统中的 shell 指令有了更为深入的了解。它的语法具有自己的特点，十分有趣。在实验过程中，不可避免遇到一些问题，但是通过查阅资料以及和同学们进行讨论，很好地解决了这些疑问，进一步培养了自己的交流沟通能力。且在考虑异常输入处理时，对问题进行了细致的思考、类比，锻炼了自己解决问题的能力。总之通过本次实验，我收获颇丰。