中国矿业大学计算机学院

2019-2020(2)本科生 Linux 操作系统课程报告

内容范围 shell 编程

指 标 点 1.2 占 比 20%

学生姓名 袁孝健 学 号 06172151

专业班级 信息安全 2017-01 班

任课教师 杨东平

课程基础理论掌握程度	熟练 □	较熟练 □	一般 □	不熟练 □
综合知识应用能力	强 □	较强 □	一般 □	差 □
报告内容	完整 □	较完整 □	一般 □	不完整 □
报告格式	规范 □	较规范 □	一般 □	不规范 □
实验完成状况	好 □	较好 □	一般 □	差 □
工作量	饱满 □	适中 □	一般 □	欠缺 □
学习、工作态度	好 □	较好 □	一般 □	差 □
抄袭现象	无 □	有 □ 姓名:		
存在问题				
总体评价				

综合成绩: 任课教师签字:

年 月

摘　要

Shell 是一个用 C 语言编写的程序，它是用户使用 Linux 的桥梁，实际上 Shell 既是一种命令语言，又是一种程序设计语言。对于 Linux 的学习，仅仅是掌握命令的实用是远远不够的，使用 Shell 的熟练程度反映了用户对 Linux 的掌握程度。为了加深对 Shell 编程的学习，本文中实现了冒泡排序、猜数游戏、进制转换、判断主机存活四个具有一定实际功能的程序。实验中主要用到了 shell 编程中的基本知识以及循环控制语句，最后对本次实验及学习 shell 编程的收获进行总结。

关键词：Linux 操作系统；shell 脚本；程序设计

Abstract

Shell is a program written in C language. It is a bridge for users to use Linux. In fact, Shell is both a command language and a programming language. For the study of Linux, it is not enough to just grasp the practicality of commands. The proficiency of using Shell reflects the user's mastery of Linux. In order to deepen the learning of Shell programming, in this article, four programs with certain practical functions, such as bubble sorting, guessing game, base conversion, and judging the survival of the host computer, are implemented. The basic knowledge of shell programming and loop control statements are mainly used in the experiment. Finally, the results of this experiment and learning of shell programming are summarized.

Keywords: Linux system; shell script; programming

目　　录

2 shell 编程

要求：独立完成 3 个以上 shell 程序，每个 shell 程序都要阐明功能和作用，要有必要的注释，并将运行过程和结果截图附上，最后阐明体会与收获。

在 shell 脚本中要体现条件控制(如 if 结构和条件分支),循环(for,while 和 until 循环）等结构，要掌握 shell 程序的调试过程。

根据所完成的 shell 程序的难度与阐述的详细和完整程度给出分值，若不够 3 个，每少 1 个扣 7 分，若无运行截图，该程序扣 3 分。

说明：如果撰写规范不符合《计算机学院考查类课程报告撰写规范》要求的，整体上酌情扣除 1-10 分。

2.1 冒泡排序法

2.1.1 功能与作用

冒泡排序法是计算机科学领域一种非常常见的排序算法，所以我尝试用 shell 编程实现了冒泡排序算法，主要用到了 for 循环和 if 语句，功能为对用户输入的一组数字按照从小到大进行排序并输出。

基本功能及实现如下：

（1）比较相邻的元素。如果第一个比第二个大，就交换他们两个。

（2）对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。此时，最后的元素应该会是最大的数。

（3）针对所有的元素重复以上的步骤，除了最后一个。

（4）持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较，即排序完成。

2.1.2 代码实现

```
#!/bin/bash
echo "Please input numbsers: "
read -a nums    # 读取用户输入(-a 表示接收为数组)
len=${#nums[@]} # 输入数组的长度
for ((i=0;i<$len;i++))    # 外循环为排序趟数,len 个数进行 len-1 趟
do
    for ((j=0;j<`expr $len - $i - 1`;j++)) # 内循环为每趟比较次数,第 i 趟比较 len-i 次
    do
        if [ ${nums[`expr $j + 1`]} -lt ${nums[$j]} ]   # 相邻元素比较,若逆序则交换
```

```
        then
            tmp=${nums[$j]}
            nums[$j]=${nums[`expr $j + 1`]}
            nums[`expr $j + 1`]=$tmp
        fi
    done
done
# 输出排序结果
echo -e "After sort:"
echo ${nums[*]}
```

2.1.3 运行结果

```
ubuntu@VM-0-14-ubuntu:~/learn$ ./sort.sh
Please input numbsers:
2 8 3 9 7 1 4 6 5
After sort:
1 2 3 4 5 6 7 8 9
```

2.2 猜数游戏

2.2.1 功能与作用

用 shell 脚本实现一个猜数游戏，主要用到了 while 循环、if...elif..else 语句、break 和 continue 语句。

基本功能及实现如下：

（1）首先，生成一个 10 以内的随机数 ans。

（2）然后用 while 语句循环接收用户的输入。

（3）接下来用 if...elif...else 语句判断用户输入的结果。

① 若输入为 q 或 Q，则执行 break 语句，退出游戏。

② 若输入为非数字，则输出“Not a number!”，并执行 continue 语句。

③ 若输入等于随机数，则输出“You are right!”，并执行 break 语句，游戏结束。

④ 若输入大于随机数，则输出“too big!”，循环继续。

⑤ 若输入小于随机数，则输出“too small!”，循环继续。

2.2.2 代码实现

```
#!/bin/bash
ans=$[RANDOM%10+1]      # 生成 1-10 之间的随机数
while true
do
    read -p "Guess a number from 1-10: " num
    if [ $num == q -o $num == Q ]   # 输入 q 或 Q 退出游戏
    then
            echo "Quit!"
            break
    elif [[ ! $num =~ ^[0-9]+$ ]]   # 正则匹配输入为非数字则 continue
    then
            echo "Not a number!"
            continue
    elif [ $num == $ans ]   # 猜测正确
    then
            echo "You are right!"
            break
    elif [ $num -gt $ans ]  # 猜测过大
    then
            echo "too big!"
    else                    # 猜测过小
            echo "too small!"
    fi
done
```

2.2.3 运行结果

```
ubuntu@VM-0-14-ubuntu:~/learn$ ./guess.sh
Guess a number from 1-10: cumt
Not a number!
Guess a number from 1-10: q
Quit!
ubuntu@VM-0-14-ubuntu:~/learn$ ./guess.sh
Guess a number from 1-10: 5
too small!
Guess a number from 1-10: 10
too big!
Guess a number from 1-10: 7
You are right!
```

2.3 进制转换

2.3.1 功能与作用

在 Linux 下，可以通过((num=2#1111))的方式，转换为 10 进制，#前为进制，#后为要转的数，而这里我用 shell 脚本来实现任意进制数向 10 进制的转换。

基本功能及实现如下：

（1）首先通过 read 命令读取要转换的进制和要转换的数。

（2）获取数的长度 len 后，用 while 循环 len 次。

（3）由于任意进制中可能出现字母，于是考虑利用 ascii 进行运算，实用`printf "%d" 'a`可以将 a 转换为 a 的 ascii 码。

（4）每次循环用${n:$i:1}截取 n 的第 i 为转换，然后根据 ascii 码进行运算并加到结果上。

（5）最后输出结果。

2.3.2 代码实现

```
#!/bin/bash
read -p "Please input the radix: " r   # 要转换的进制
read -p "Please input the num: " n     # 要转换的数
len=${#n}  # 数的长度
i=0
ans=0  # 保存结果
while [ $len != $i ]       # len 次循环
do
    ans=`expr $ans \* $r`  # ans 循环除以进制 r
    tmp=`printf "%d" \'${n:$i:1}`        # 截取 n 的第 i 位,并转为 Ascii 码
    if [ $tmp -ge 48 -a $tmp -le 57 ]    # 若是 0-9,则减 48 再加 ans
    then
        ans=`expr $ans + $tmp - 48`
    elif [ $tmp -ge 65 -a $tmp -le 85 ] # 若是 A-Z,则 55 再加 ans
    then
        ans=`expr $ans + $tmp - 55`
    else                                # 若是 a-z,则减 87 再加 ans
        ans=`expr $ans + $tmp - 87`
    fi
    i=`expr $i + 1`
done
echo $ans
```

2.3.3 运行结果

（1）2 进制转 10 进制：

```
ubuntu@VM-0-14-ubuntu:~/learn$ ./transform.sh
Please input the radix: 2
Please input the num: 1111
15
```

验证：

```
ubuntu@VM-0-14-ubuntu:~/learn$ ((num=2#1111))
ubuntu@VM-0-14-ubuntu:~/learn$ echo $num
15
```

（2）8 进制转 10 进制

```
ubuntu@VM-0-14-ubuntu:~/learn$ ./transform.sh
Please input the radix: 8
Please input the num: 173
123
```

验证：

```
ubuntu@VM-0-14-ubuntu:~/learn$ ((num=8#173))
ubuntu@VM-0-14-ubuntu:~/learn$ echo $num
123
```

（3）16 进制转 10 进制

```
ubuntu@VM-0-14-ubuntu:~/learn$ ./transform.sh
Please input the radix: 16
Please input the num: 2CF
719
```

验证：

```
ubuntu@VM-0-14-ubuntu:~/learn$ ((num=16#2CF))
ubuntu@VM-0-14-ubuntu:~/learn$ echo $num
719
```

2.4 判断主机存活

2.4.1 功能与作用

Linux 系统的 ping 命令是常用的网络命令，它通常用来测试与目标主机的连通性，因此我尝试编写一个 shell 脚本，通过接收用户输入的 ip 在内部实用 ping 命令进行判断，

最后返回该 ip 地址对应的主机是否存活。

基本功能及实现如下：

（1）首先通过 read 命令读取要判断的 ip 地址列表，用 ip_list 数组来存储。

（2）然后遍历每一个 ip 地址，对每个 ip 地址我们最多尝试 3 次 ping 命令。

（3）每次 ping 命令发送一个数据包，只要有一次成功 ping 通，则判断目标主机存活并 break。

（4）每次 ping 命令失败，则用 fail_cnt 变量记录失败次数。

（5）若失败次数到达 3 次，则判断目标主机不存活。

2.4.2 代码实现

```
#!/bin/bash
read -p "Input IP: " -a ip_list  # 输入一个 ip 列表
for ip in ${ip_list[@]}  # 遍历 ip 列表里的每个 ip
do
    fail_cnt=0 # 记录失败次数
    for ((i=0;i<3;i++))       # 每个 ip 尝试 3 次 ping 命令
    do
        if ping -c 1 $ip >/dev/null  # 每次发送一个包
        then
            echo "$ip is alive."  # 若 ping 通了则主机存活,然后 break
            break
        else
            let fail_cnt++     # ping 不通则失败次数自增 1
        fi
    done
    if [ $fail_cnt -eq 3 ]    # 失败 3 次,返回主机不存活
    then
        echo "$ip is not alive."
    fi
done
```

2.4.3 运行结果

```
ubuntu@VM-0-14-ubuntu:~/learn$ ./judgeAlive.sh
Input IP: 39.156.69.79 66.102.251.24 47.107.21.49
39.156.69.79 is alive.
66.102.251.24 is not alive.
47.107.21.49 is alive.
```

2.5 收获与体会

通过此次实验，对 shell 脚本的编写有了更深的了解，这也意味着对 Linux 操作系统的学习又上了一个台阶。选取了四个具有一定实际意义的程序，除了基本的 shell 编程知识外，主要用到了循环控制语句来实现相应功能。

而由于平时习惯用 C++、Python 的高级语言编程，在刚开始时对 shell 编程的一些语法规定有些不适应，如：

（1）赋值号两端不能有空格；

（2）if 语句后要加 then，且 if 块必须以 fi 结束；

（3）for、while 等循环语句块不适用大括号，而是以 done 结束；

（4）小括号、中括号、大括号及括号数量的不同，其表示的作用也不同；

（5）单引号与双引号在某些地方也具有不同的功能；

总而言之，通过此次实验还是学习了很多 Linux 相关知识，但所编写的 shell 脚本程序还并不是特别复杂,要想再不断提高自己的 Linux 水平,还必须要在以后的学习过程中多看、多写、多练才能达到。

参考文献

[1] 鸟哥. 鸟哥的 Linux 私房菜: 基础学习篇[M]. 人民邮电出版社, 2010.

[2] Richard Blum. Linux 命令行与 shell 脚本编程大全. 人民邮电出版社, 2012