**实验报告四 shell编程**

姓名：叶倩琳

班级：软工1706班

学号：201706061330

提交日期：2019年12月27日

[实验环境：Ubuntu]

1. **实验目的**

1.了解shell的特点和主要种类。

2.掌握shell脚本的建立和执行方式。

3.掌握bash的基本语法。

4.学会编写shell脚本。

1. **实验内容**

1.shell脚本的建立和执行。

2.历史命令和别名定义。

3. shell 变量和位置参数、环境变量。

4. bash 的特殊字符。

5.一般控制结构。

6.算术运算及bash函数。

**三、主要实验步骤**

1.利用vi建立一个脚本文件，其中包括date、 cal、 pwd、Is等常用命令:然后以不同方式执行该脚本。

yql@DESKTOP-TRMDDFQ:~$ vi command.sh

i

#!/bin/bash

echo "date:":`date`;

echo "cal":`cal`;

echo "pwd":`pwd`;

echo "ls":`ls`.

Enter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash command.sh

date::Sat Dec 28 21:36:44 DST 2019

cal: December 2019 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

pwd:/home/yql

ls:SAMPLE a.txt a\_copy.txt b.txt command.sh cp1.sh database f.sh lab3.sh lowerToUpper.sh mydir myfile print.sh sfile.txt showdate.sh test1 test2 testCopy.sh testFile.sh testSort.sh yourfile.

yql@DESKTOP-TRMDDFQ:~$ bash<command.sh

date::Sat Dec 28 21:36:44 DST 2019

cal: December 2019 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

pwd:/home/yql

ls:SAMPLE a.txt a\_copy.txt b.txt command.sh cp1.sh database f.sh lab3.sh lowerToUpper.sh mydir myfile print.sh sfile.txt showdate.sh test1 test2 testCopy.sh testFile.sh testSort.sh yourfile.

yql@DESKTOP-TRMDDFQ:~$ chmod a+x command.sh

yql@DESKTOP-TRMDDFQ:~$ ./command.sh

date::Sat Dec 28 21:36:44 DST 2019

cal: December 2019 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

pwd:/home/yql

ls:SAMPLE a.txt a\_copy.txt b.txt command.sh cp1.sh database f.sh lab3.sh lowerToUpper.sh mydir myfile print.sh sfile.txt showdate.sh test1 test2 testCopy.sh testFile.sh testSort.sh yourfile.

2.运行history命令，配置历史命令环境。

yql@DESKTOP-TRMDDFQ:~$ history

1 123456

2 who

3 whoami

4 password

5 lt

6 ls

.....

635 ./command.sh

636 history

3.体会bash的命令补齐功能。

4.用alias定义别名，然后执行。

yql@DESKTOP-TRMDDFQ:~$ alias new='date'

yql@DESKTOP-TRMDDFQ:~$ new

Fri Dec 27 15:23:47 DST 2019

yql@DESKTOP-TRMDDFQ:~$ unalias new

yql@DESKTOP-TRMDDFQ:~$ new

Command 'new' not found, but can be installed with:

sudo apt install mmh

sudo apt install nmh

5.对习题4.8中的shell脚本进行编辑，然后执行。

6.按习题4.14要求编写脚本，然后执行。

7.按习题4.18要求编写脚本，然后执行。

8.运行例4.20的程序。若取消其中的“eval", 则会出现什么情况?

答：若不用eval()函数，则所有结果都为字符串。即 eval $1=’$temp’是将temp的值赋值给名为$1的变量，而 $1=’$temp’会直接把字符串 $temp 赋值给名为$1的变量。

eval的原理：首先将引号剥去，若得到的数不可以进行计算的，它就开始查找是否是一个变量的名字，输出了其中的内容；否则将其计算输出。

**四、实验过程**

4.1常用的shell 有哪儿种？Linux系统中默认的shell什么?

答：1inux几种常见的she1l: sh、bash、csh、tcsh、 ash

Linux下默认的She11是 bash（Bourne Again She11）

4.2简述shell的主要功能。bash有什么特点?

答：she11是系统的用户界面，提供了用户与内核进行交互操作的一种接口。

它接收用户输入的命令并把它送入内核去执行。为用户提供shell高级程序设计语言，方便了管理人员对系统的维护和普通用户的应用开发，提高了编程效率。

bash特点：

1. 对已有的命令进行适当的组合，构成新的命令，如建立shell脚本。
2. 提供文件名扩展字符（通配符，如\*，?，[]），使得单一字符串可以匹配多个文件名。
3. 可以使用shell内置命令（如cd，echo，exit，pwd，kill），而无需创建新的进程。
4. 允许灵活地使用数据流。
5. 结构化的程序模块，提供顺序、条件、循环等控制流程。
6. 提供在后台（&）执行命令的能力。
7. 提供可配置的环境，允许用户创建和修改命令和其他系统行为。
8. 提供一个高级命令语言，允许用户创建从简单到复杂的程序。
9. 命令补全： $PATH；路径补全：TAB命令别名：alias, unalias；命令历史：history；bash 快捷键。

4.3执行shell脚本的方式是哪些?

答: 执行shell脚本的方式有三种：

（1）输入定向到she11脚本。该方式用输入重定向方式让shel1从给定

文件中读入命令行，并进行相应处理。其-般形式是:

$bash<脚本名，例如：

$ bash <ex1

（2）以脚本名作为bash参数。其一般形式是:

$bash脚本名[参数]，例如：

$ bash ex2 /user/meng /user/zhang

（3）将she11脚本的权限设置为可执行，然后在提示符下直接进行。例如：

$ chmod a+x ex3

$ ./ex3

4.4将主提示符改为用户的主目录名，并予以输出

答:

yql@DESKTOP-TRMDDFQ:~$ export PS1= $HOME

/home/yql

/home/yqlecho $HOME

/home/yql

export 命令用于设置或显示环境变量。

在 shell 中执行程序时，shell 会提供一组环境变量。export 可新增，修改或删除环境变量，供后续执行的程序使用。export 的效力仅限于该次登陆操作。

语法：

export [-fnp][变量名称]=[变量设置值]

参数：

-f 　代表[变量名称]中为函数名称。

-n 　删除指定的变量。变量实际上并未删除，只是不会输出到后续指令的执行环境中。

-p 　列出所有的shell赋予程序的环境变量。

4.5说明三种引号的作用有什么区别。

答: Linux She11中有三种引号，分别为双引号(“”)、单引号(‘’)以及

反引号(`)。

（1）双引号(“”)：部分解析。在双引号中，除了$,倒引号(`)，和反斜杠(\)以外所有的字符都作为普通字符对待。

（2）单引号(‘’)：不解析。在单引号中所有的字符包括特殊字符($,’，和\)都将解释成字符本身而成为普通字符。

（3）倒引号(`)：全解析。在反引号中的字符串将解释成she11命令来执行。

4.6利用变量赋值方式，将字符串DOS file c:>\$student\\*显示出来。

答：

yql@DESKTOP-TRMDDFQ:~$ a=DOS

yql@DESKTOP-TRMDDFQ:~$ b=file

yql@DESKTOP-TRMDDFQ:~$ c="c:>\$student\\*"

yql@DESKTOP-TRMDDFQ:~$ echo $a $b $c

DOS file c:>$student\\*

4.7显环境变量的设置情况,说明各自的意义。

答:

（1）只对当前的she11起作用的环境变量

控制台中设置，只对当前的she11起作用，换一个she11设置就无效了：

直接控制台中输入: $PATH="$PATH":/NEW\_ PATH (关闭 she11 Path会还

原为原来的path)

yql@DESKTOP-TRMDDFQ:~$ PATH=$PATH:/bin"

PATH=$PATH:$HOME/bin这个代码是设置PATH环境变量，设置环境变量用等号，:冒号是分割符。

（2）对所有的用户的都起作用的环境变量

修改/etc/profile 文件， 如果你的计算机仅仅作为开发使用时推存

使用这种方法，因为所有用户的she11都有权使用这个环境变量，可能会给系统

带来安全性问题。这里是针对所有的用户的，所有的she11：

yql@DESKTOP-TRMDDFQ:~$ vi /etc/profile

i

export PATH="$PATH:/bin"

Enter esc

:wq

（3）针对当前特定的用户起作用的环境变量

修改bashrc文件，这种方法更为安全,它可以把使用这些环境变量的权

限控制到用户级别，这里是针对某一特定的用户，如果你需要给某个用户权限使

用这些环境变量，只需修改其个人用户主目录下的 .bashrc文件：

yql@DESKTOP-TRMDDFQ:~$ vi /.bashrc

i

export PATH="$PATH:/bin"

Enter esc

:wq

4.8

答：该脚本的功能类似于执行堆栈操作。

yql@DESKTOP-TRMDDFQ:~$ vi stack.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

count=$# #获取脚本参数个数

cmd=echo #初始化cmd变量，赋值为“echo”

while[$count -gt 0] #如果count>0，执行下面的循环

do

cmd="$cmd \$$count" #给cmd重新赋值为“echo ${count的值}”

count=`expr $count-1` #count减1

done  #继续返回while判断

eval $cmd #执行cmd命令(eval命令用于重新运算求出参数的内容)

enter esc

:wq

bash stack.sh 2

2

1

4.9编写一个shell脚本，它把第二个位置参数及其以后的各个参数指定的文件复制到第一个位置参数指定的目录中。

yql@DESKTOP-TRMDDFQ:~$ vi cp1.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

dir=$1 #初始化第一个参数

shift #参数左移

while [ $1 ] #循环

do

file=$1 #赋值现在位置的参数1

cp $dir $dir #将现参数1中的值复制到目录中

shift #参数左移

done

ls $dir #显示指定目录中的列表

enter esc

:wq

bash cp1.sh mydir a.txt b.txt

a.txt b.txt

4.10编写一个shell脚本，显示当天日期，查找给定的某用户是否在系统中工作。如果在系统中，就发一个问候给他。

yql@DESKTOP-TRMDDFQ:~$ vi showdate.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

date #显示日期

echo "请输入访问用户:"

read username

if who|grep ${username} #寻找当前用户

then

write ${username}<<! #用write给对用户发消息

!

echo "hello!"

!

fienter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash showdate.sh yql

Fri Dec 27 11:40:40 DST 2019

请输入访问用户:

yql

4.11打印给定目录的某些文件，由第一个参数指出文件所在的目录，其余参数是要打印的文件名。

yql@DESKTOP-TRMDDFQ:~$ vi print.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

dir=$1 #第一参数为目录名  
cd $dir #进入该目录  
shift  
for f in $@ #$@表示全部参数  
do  
 cat $f  
done

enter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash print.sh mydir a.txt b.txt

aaa

bbb

4.12利用for循环将当前目录下的（\*.c）文件移到指定的目录下，并按文件大小排序，显示移动后指定目录的内容。

yql@DESKTOP-TRMDDFQ:~$ vi move.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

for file in `ls -l /home/yql/a|grep ".\*.c"`

do

mv /home/yql/a/$file /home/yql/b

done

ls -ls /home/yql/b #-s表示排序，默认是降序排列;-lrs升序

enter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash move.sh

4.13利用数组形式存放10个城市的名字，然后利用for循环把他们打印出来。

yql@DESKTOP-TRMDDFQ:~$ vi printcity.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

for China in Beijing Shanghai Guangzhou Shenzhen Chongqing Tianjin Wuhan Taiwan Xianggang Aomen

do

echo $China

done

enter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash printcity.sh

Beijing

Shanghai

Guangzhou

Shenzhen

Chongqing

Tianjin

Wuhan

Taiwan

Xianggang

Aomen

4.14编写一个shell脚本，求斐波那契数列的前10项及总和。

yql@DESKTOP-TRMDDFQ:~$ vi f.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

a=1

b=1

echo -n -e "$a\t$b"

#echo -n 不换行输出;echo -e 处理特殊字符(/t插入tab)

let n=a+b

count=4

while [[ $count -gt 0 ]]

do

let a=a+b #let+要执行的表达式，若含空格则要加引号

let b=b+a

echo -n -e "\t$a\t$b"

let n+=a+b

let count=count-1

done

echo

echo "The sum is $n"

enter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash f.sh

1 1 2 3 5 8 13 21 34 55

The sum is 143

在运行时出现了“ [: =: unary operator expected” 的错误，就一直找不到原因，尝试了删除等号两侧的空格和括号里的空格都不管用。最后在条件表达式外再添加一层“[]”，就不会出错了，如下：

While [[ $count -gt 0 ]]

其原因是因为如果变量count值为空，那么条件表达式就成了 [ -gt 0 ] ，显然 [ 和 "0" 不可比并且缺少了 [ 符号，所以报了这样的错误。

4.15下述表达式的作用是什么?  
${name[\*]} ${name[@]} ${name #\*/} ${ name%%.\* } ${#name[\*]} ${name:-Hello}  
答：

${name[\*]}：数组所有元素  
${name[@]}：数组每一个元素  
${name#\*/}：删掉第一个/及其左边的字符串  
${name%%.\*}：删掉第一个.及其右边的字符串，即提取第一个.前面的内容  
${#name[\*]}：数组元素的个数  
${name:-Hello}：用 Hello 代替 name进行替换，name的值不变

4.16显示前面所执行的40个命令的清单，重新执行倒数第5条命令。编写其中一条命令，然后执行。  
答；运行：

yql@DESKTOP-TRMDDFQ:~$ history 40

yql@DESKTOP-TRMDDFQ:~$ !-5

vi f.sh

使用上下方向键找到要编辑的命令，然后用左右键选择要编辑的地方删除要去掉的内容，然后输入要更改的内容。

#### 方法1：!!

会重复执行最后执行的那条命令。如果需要 root 权限，同样只需在前面加 sudo 即可。

$ sudo !!

#### 方法2：!un

un 不是固定的，它是代表你想要重复输入的命令的前几个字符。如想重复执行 uname -a 这条命令：

$ !un  
uname -a

#### 方法3：!-1

!-n 表示重复执行倒数第 n 条指令。

$ !-1  
uname -a  
Linux DESKTOP-TRMDDFQ 4.4.0-17763-Microsoft #864-Microsoft Thu Nov 07 15:22:00 PST 2019 x86\_64 x86\_64 x86\_64 GNU/Linux

#### 方法4：ctrl+r

ctrl+r 这个快捷键可以搜索你之前输入过的所有的命令。

如要重复执行 uname -a ，先按一下 ctrl+r 这个组合键，然后再依次敲入 ‘u' 'n' 'a' ，如果之前输入的命令比较少的话，基本只需敲一个 u 或 n 就可以找到你要输入的命令了。

(reverse-i-search)`u': uname -a

找到之后，再敲一下回车，即可重复执行输入的命令了。

#### 方法5：ctrl+p 或 ctrl+n 配合 ctrl+o

通过 ctrl+p/n 的组合键找到想要的命令后，按 ctrl+o 来执行这条命令。这个组合键与回车不太相同，因为它除了回车之外，还另外跳转到最后一条命令。通过不断地 ctrl+p/n ，然后 ctrl+o ，可以高效的重复执行你要执行的命令。

4.17定义一个别名命令，它等价的功能是：显示当前日期和工作目录，并列出有多少用户在系统上工作。

答：

yql@DESKTOP-TRMDDFQ:~$ alias new='date && pwd && who|wc -w'

yql@DESKTOP-TRMDDFQ:~$ alias

alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s\*[0-9]\+\s\*//;s/[;&|]\s\*alert$//'\'')"'

alias egrep='egrep --color=auto'

alias fgrep='fgrep --color=auto'

alias grep='grep --color=auto'

alias l='ls -CF'

alias la='ls -A'

alias ll='ls -alF'

alias ls='ls --color=auto'

alias new='date && pwd && who|wc -w'

yql@DESKTOP-TRMDDFQ:~$ new

Fri Dec 27 14:49:54 DST 2019

/home/yql

0

4.18设计一个程序cuts，它由标准输入读取数据，获取由第一个参数n和第二个参数m所限定范围的数据，n和m都是整数，即从输入的字符串中抽取第n个字符至第m个字符之间的所有字符（包括这两个字符）。  
答：

yql@DESKTOP-TRMDDFQ:~$ vi cuts.sh

i

#!/bin/bash #调用bash程序对其解释执行，相当与于头文件

read line;

expr substr "$line" $1 $2;

enter esc

:wq

yql@DESKTOP-TRMDDFQ:~$ bash cuts.sh 1 4

abcdefg

abcd

**五、实验体会**

通过本次实验，我对liunx的shell编程有了基本的掌握，了解并能够实际应用bash编程实现一个小功能，对liunx的控制语句（for语句、while语句）、函数、别名定义取消有了更进一步的学习。在实验中的许多小错误都是由三种引号引起的，因此，我会继续加强对三种引号不同作用的学习，强化编程能力。