HELL脚本编程基础

一.编程基础

1.程序基本概念

- Linux内核的编写者Linus Tovarls曾经说过：Talk is cheap, show me the code.(用代码说话);所以学shell编程的唯一办法就是写shell程序，不断的写写写。
- shell 编程基础主要涉及变量、基本算术运算、条件测试和条件判断等编程基础知识。在正式学习shell编程之前先了解一些程序相关的基本概念。

```
程序的概念：
程序：程序本质上是一系列的数据结构加上处理这些数据的算法形成的，程序的核心就是数据结构。
算法：使用程序描述的处理数据的方式
数据结构：数据在计算机中的类型和组织方式(数据在内存中的存储和组织方式)

编写程序时有多种程序编程风格：

过程式：以指令为中心，数据服务于指令
对象式：以数据为中心，指令服务于数据
```

2.不同的程序的执行方式

- 计算机只能识别二进制指令；
- 低级编程语言：

```
机器语言：由二进制的0和1的序列组成，称为机器指令;与自然语言差异太大，难懂、难写
汇编语言：用一些助记符号替代机器指令，称为汇编语言
        如：ADD A,B 将寄存器A的数与寄存器B的数相加得到的数放到寄存器A中
汇编语言写好的程序需要汇编程序转换成机器指令
汇编语言稍微好理解，即机器指令对应的助记符，助记符更接近自然语言
```

- 高级编程语言

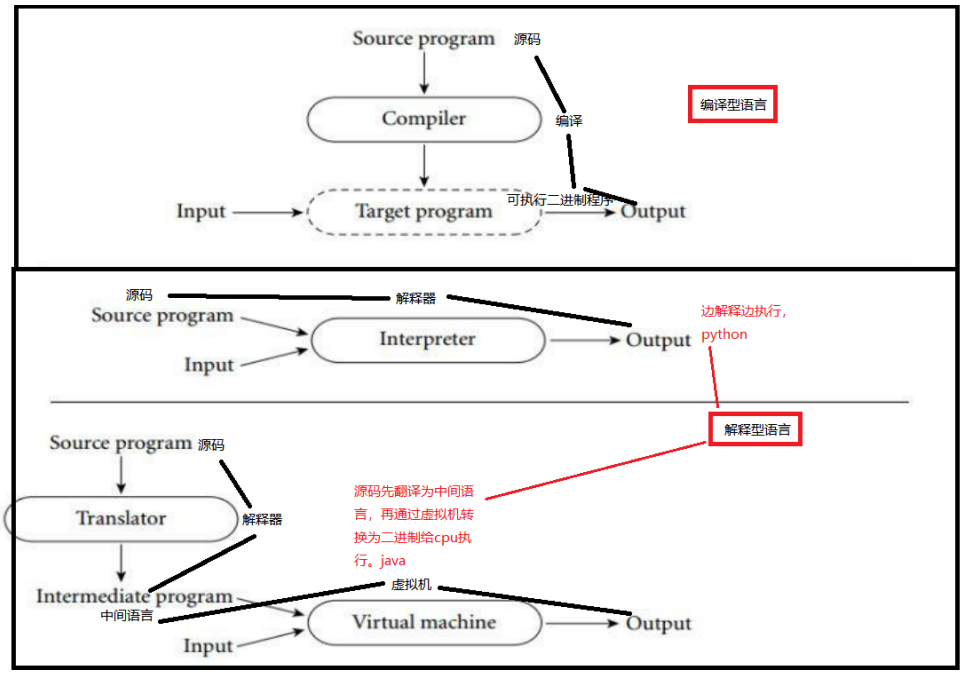
```
高级语言又有解释型和编译型语言；
编译型语言：高级语言-->编译器-->机器代码-->执行
          如：C，C++
解释语言：高级语言-->执行-->解释器-->机器代码
          如：  shell,python,php,JavaScript,perl,Ruby
```

3.编译型语言和解释型语言

- 编译型语言将源码编译为二进制可执行文件后由计算机运行，运行时无需源码

- 解释型语言是在程序运行时一条一条的将代码解释为计算机可识别的指令，运行时需要源码
- 对比

编译型语言	解释型语言
c,c++,Delphi,Rust	shell,python,php,JavaScript,perl,Ruby
一次性将源码编译为机器指令	在程序执行时才一条一条的将源代码转换为机器指令
高性能，运行速度快	支持夸平台，运行速度慢
在语言编译阶段可以做充分的优化	不需要编译阶段，直接执行代码



二.shell脚本基础

1.shell脚本的基本结构

- 脚本的基本结构

```
SHEBANG                         # SHEBANG机制，指明解释该脚本的shell程序
CONFIGURATION_VARIABLES           # 在写明解释器后，接着的行一般用来声明一些变量
FUNCTION_DEFINITIONS              # 接着定义函
MAIN_CODE                         # 接着是脚本内容
```

2.hell脚本格式要求：首行shebang机制

```
#!/bin/bash
#!/usr/bin/python
#!/usr/bin/perl
```

- shell脚本的用途有：

```
自动化常用命令
执行系统管理和故障排除
创建简单的应用程序
处理文本或文件
```

- 创建shell脚本

```
第一步：使用文本编辑器来创建文本文件
        第一行必须包括shell声明序列：#!
        示例：#!/bin/bash
        添加注释
            注释以#开头
第二步：运行脚本
        给予执行权限，在命令行上指定脚本的绝对或相对路径
        直接运行解释器，将脚本作为解释器程序的参数运行
```

- 脚本代码开头约定

```
1、第一行一般为调用使用的语言
2、程序名，避免更改文件名为无法找到正确的文件
3、版本号
4、更改后的时间
5、作者相关信息
6、该程序的作用，及注意事项
7、最后是各版本的更新简要说明
```

- shell脚本示例

```
#!/bin/bash
# ----------------------------------------
# Filename:             hello.sh
# Revision:             0.99
# Date:                 2019/09/18
# Author:               steve
# Email:                steve@gmail.com
# Website:              suosuoli.cn
# Description:          This is my first script.
# Copyright:            2019 steve
# License:              GPL
# ----------------------------------------
 echo -e “Hello world! I'm a programmer now! Wow!”
```

3.脚本调试

```
检测脚本中的语法错误
bash -n /path/to/some_script
调试执行
bash -x /path/to/some_script
```

三.变量

1.bash中变量的种类

- 在shell中根据变量的生效范围等标准划分下面变量类型

```
局部变量：生效范围为当前shell进程；对当前shell之外的其它shell进程，包括当前shell的子shell进程均无效

环境变量：生效范围为当前shell进程及其子进程

本地变量：生效范围为当前shell进程中某代码片断，通常指函数

位置变量：$1, $2, ...来表示，用于让脚本在脚本代码中调用通过命令行传递给它的参数

特殊变量：$?, $0, $*, $@, $#,$$
    $? 存储脚本、命令或者函数的退出状态值；脚本的退出状态为脚本中最后一条命令的退出状态；函数退出状态也为最后一条命令的退出状态;一般成功执行退出状态为0；命令执行失败退出状态为1-255之间的整数.
    $0 执行脚本时脚本的路径名
    $* 将所有位置参数视为单个字符串
    $@ 每个位置参数存储为单独引用的字符串，分开对待每个位置参数
    $# 脚本后所跟的参数的个数
    $$ 为PID变量，存储其出现在的脚本所属的进程的进程号
```bash
[root@centos7 ~]$ pstree -p |grep sshd.*bash
|-sshd(1164)---sshd(1842)---bash(1848)---bash(6007)-+-grep(6043)
[root@centos7 ~]$ echo $$                          # 当前所在bash进程为6007
6007
[root@centos7 ~]$exit                             # 退出6007号bash进程
exit
[root@centos7 ~]$echo $$                          # 此时$$记录1848
1848
```

- 局部变量赋值和引用

```
变量赋值：name='value'
(1) 可以是直接字串：name='root'
(2) 变量引用：name="$USER"
(3) 命令引用：name=`COMMAND`
              name=$(COMMAND)
变量引用：${name}  或者  $name
```
```

- 显示已定义的所有变量使用set命令
- 删除变量：unset name

2.练习

- 1、编写脚本 systeminfo.sh，显示当前主机系统信息，包括主机名，IPv4地址，操作系统版本，内核版本，CPU型号，内存大小，硬盘大小

```
#!/bin/bash
#
#********************************************************************************
#Author:                    steveli
#QQ:                        1049103823
#Data:                      2019-10-06
#FileName:                  systemInfoForCentos678.sh
#URL:                       https://blog.csdn.net/YouOops
#Description:               Show some sys info.
#Copyright (C):             2019 All rights reserved
#********************************************************************************

R_SEED=$(( $RANDOM%7+31 ))
COLOR_START="\e[1;${R_SEED}m"
COLOR_END="\e[0m"
echo -e "The hostname is:              ${COLOR_START}`hostname`. ${COLOR_END}"
echo -e "IPV4 addr is:                 ${COLOR_START}`ifconfig | egrep -o '([0-9]
{1,3}\.){3}[0-9]{,3}' |head -n1` ${COLOR_END}"
echo -e "The OS version is:            ${COLOR_START}`cat /etc/redhat-release |
egrep -o '[0-9]+\.[0-9]+(\.[0-9]+)?'` ${COLOR_END}"
echo -e "The kernel version is:        ${COLOR_START}` uname -r `. ${COLOR_END}"
free -h |grep "Mem" | tr -s " " |cut -d" "  -f2,4 > /tmp/total_free
TOTAL=`cat /tmp/total_free | cut -d" "  -f1`
FREE=`cat /tmp/total_free | cut -d" "  -f2`
echo -e "The total online memory is:   ${COLOR_START}$TOTAL(${FREE}
free)${COLOR_END}"
echo -e  "\nHarddisk usage:             ${COLOR_START} \n\n`df -h | egrep -e
/dev/sd -e /dev/nvme` ${COLOR_END}"
unset R_SEED COLOR_START COLOR_END
unset TOTAL FREE
```

2、编写脚本 backup.sh，可实现每日将/etc/目录备份到/backup/etcYYYY- mm-dd中

```
#!/bin/bash
#
#********************************************************************************
#Author:            steveli
#QQ:                1049103823
#Data:              2019-10-07
```

```
 #FileName:          backup.sh
 #URL:               https://blog.csdn.net/YouOops
 #Description:       Test scripting.
 #Copyright (C):     2019 All rights reserved
 #********************************************************************************
DIR="/data/backup/"
echo "Backup starting ..."
sleep 1
cp -av /etc/ ${DIR}etc`date +%F`
echo -e "\e[1;32mBackup finished."
echo -e "The backup file located at : ${DIR} \a\e[0m"
```

3、编写脚本 disk.sh，显示当前硬盘分区中空间利用率最大的值

```
 #!/bin/bash
 #
 #********************************************************************************
 #Author:            steveli
 #QQ:                1049103823
 #Data:              2019-10-07
 #FileName:          disk.sh
 #URL:               https://blog.csdn.net/YouOops
 #Description:       Test scripting.
 #Copyright (C):     2019 All rights reserved
 #********************************************************************************
 MAX=`df -h | grep -E /dev/\(sd\|nvme\) | egrep -o "[0-9]{,3}%" | cut -d% -f1 |
sort -nr | head -n1`
 echo "The max usage of space is $MAX(%)."
```

4、编写脚本 links.sh，显示正连接本主机的每个远程主机的IPv4地址和连接数，并按连接数从大到小排序

```
 #!/bin/bash
 #
 #********************************************************************************
 #Author:            steveli
 #QQ:                1049103823
 #Data:              2019-10-07
 #FileName:          backup.sh
 #URL:               https://blog.csdn.net/YouOops
 #Description:       Test scripting.
 #Copyright (C):     2019 All rights reserved
 #********************************************************************************
if [[ $# -gt 0 ]]; then
    FILE=$1
    cat ${FILE} |grep ESTAB | tr -s " " | cut -d" " -f5 | cut -d: -f1 | sort |
uniq -c | sort -nr
elif [[ $# -eq 0 ]]; then
    ss -tun |grep ESTAB | tr -s " " | cut -d" " -f6 | cut -d: -f1 | sort | uniq -c
```

```
| sort -nr
fi
```

3.环境变量

- 环境变量的声明、赋值

```
export name=VALUE
declare -x name=VALUE
```

- 变量引用

```
$name, ${name}
```

- 显示所有环境变量

```
env
printenv
export
declare -x
```

- 删除变量

```
unset name
```

4.bash内建的环境变量

```
PATH
SHELL
USER
UID
HOME
PWD
SHLVL
LANG
MAIL
HOSTNAME
HISTSIZE
_ :下划线
```

5.只读和位置变量

- 只读变量：只能声明，但不能修改和删除
- 声明只读变量：
- - readonly name
- - declare -r name
- 查看只读变量：
- - readonly -p
- 位置变量：在脚本代码中调用通过命令行传递给脚本的参数

```
$1, $2, ...
对应第1、第2等参数，shift [n]换位置
$0
    命令本身
$*
    传递给脚本的所有参数，全部参数合为一个字符串
$@
    传递给脚本的所有参数，每个参数为独立字符串
$#
    传递给脚本的参数的个数
注意：$@ $* 只在被双引号包起来的时候才会有差异
set --  清空所有位置变量
```

- 在linux中，进程使用退出状态来报告成功或失败;存储退出状态的环境变量使用$?；$?存储的值范围为0-255。

```
0  代表成功
1－255代表失败
示例：
    ping -c1 -W1 hostdown &> /dev/null
    echo $?
```

- bash允许自定义退出状态码

```
bash自定义退出状态码
    exit [n]：自定义退出状态码
    注意：脚本中一旦遇到exit命令，脚本会立即终止；终止退出状态取决于exit命
令后面的数字
    注意：如果未给脚本指定退出状态码，整个脚本的退出状态码取决于脚本中执
行的最后一条命令的状态码
```

四.bash-shell脚本的基本算术运算

- bash中的算术运算使用let命令

```
基本的运算符包括：
+, -, *, /, %取模（取余）, **（乘方），乘法符号有些场景中需要转义
```

- 实现算术运算

```
(1) let var=算术表达式
(2) var=$[算术表达式]
(3) var=$((算术表达式))
(4) var=$(expr arg1 arg2 arg3 ...)
(5) declare -i var = 数值
(6) echo ‘算术表达式’ | bc
```

- bash有内建的随机数生成器变量：$RANDOM（0-32767）

```
示例：生成 0 - 49 之间随机数
echo $[$RANDOM%50]
```

- bash也支持增强型赋值

```
增强型赋值：
+=, -=, *=, /=, %=

let varOPERvalue
例如:let count+=3
     自加3后自赋值

自增，自减：
let var+=1
let var++
let var-=1
let var--
```

- 练习 1、编写脚本 sumid.sh，计算/etc/passwd文件中的第10个用户和第20用户的UID之和

```
#!/bin/bash
#
#********************************************************************************
#Author:                    steveli
#QQ:                        1049103823
#Data:                  2019-10-06
#FileName:              sumid.sh
#URL:                https://blog.csdn.net/YouOops
```

```
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************

id1=`cat /etc/passwd | head -n10 | tail -n1 | cut -d: -f3`
id2=`cat /etc/passwd | head -n20 | tail -n1 | cut -d: -f3`
let sum=$id1+$id2
echo $sum

unset $id1
unset $id2
unset $sum
```

2、编写脚本 sumspace.sh，传递两个文件路径作为参数给脚本，计算这两个文件中所有空白行之和

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-06
#FileName:                  sumspace.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************

if [[ $# -eq 2 ]]; then
    F1=`cat $1 | grep "^$" | wc -l`
    F2=`cat $2 | grep "^$" | wc -l`
    echo $(( $F1 + $F2 ))
else
echo -e "Please specify two files.\nUsage: `basename $0` file1 file2"
fi
```

3、编写脚本 sumfile.sh，统计/etc, /var, /usr 目录中共有多少个一级子目录和文件

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-09
#FileName:                  sumfile.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************
```

```
DIR=`find -L /var/ /etc/ /usr/ -maxdepth 1 -type d | wc -l`
FILE=`find -L /var/ /etc/ /usr/ -maxdepth 1 -type f | wc -l`
echo "The total is : $(( $DIR+$FILE-3 ))"
echo "The dirnum is : $(( $DIR-3 ))"
echo "The filenum is : $(( $FILE ))"
unset $DIR
unset $FILE
```

五.bash中的各种测试

1.条件测试

```
- 判断某需求是否满足，需要由测试机制来实现
    专用的测试表达式需要由测试命令辅助完成测试过程
- 评估布尔声明，以便用在条件性执行中
    •若真，则返回0
    •若假，则返回1
测试命令：
    •test EXPRESSION
    •[ EXPRESSION ]
    •[[ EXPRESSION ]]
注意：EXPRESSION前后必须有空白字符
```

2.bash的数值测试

```
-v VAR
    测试变量VAR是否设置
    示例：判断 NAME 变量是否定义
        [ -v NAME ]
```

- 数值测试

```
-gt 是否大于
-ge 是否大于等于
-eq 是否等于
-ne 是否不等于
-lt 是否小于
-le 是否小于等于
```

3.bash的字符串测试

```
-z "STRING" 字符串是否为空，空为真，不空为假
-n "STRING" 字符串是否不空，不空为真，空为假
```

```
=  是否等于
>  ascii码是否大于ascii码
<  是否小于
!= 是否不等于
== 左侧字符串是否和右侧的PATTERN相同
=~ 左侧字符串是否能够被右侧的PATTERN所匹配
    注意:此表达式用于[[  ]]中，PATTERN为通配符
```

4.Bash的文件测试

```
存在性测试
    -a FILE: 同 -e
    -e FILE: 文件存在性测试，存在为真，否则为假
存在性及类别测试
    -b FILE: 是否存在且为块设备文件
    -c FILE: 是否存在且为字符设备文件
    -d FILE: 是否存在且为目录文件
    -f FILE: 是否存在且为普通文件
    -h FILE 或 -L FILE: 存在且为符号链接文件
    -p FILE: 是否存在且为命名管道文件
    -S FILE: 是否存在且为套接字文件
```

5.Bash的文件权限测试

```
文件权限测试:
    -r FILE: 是否存在且可读
    -w FILE: 是否存在且可写
    -x FILE: 是否存在且可执行
文件特殊权限测试:
    -u FILE: 是否存在且拥有suid权限
    -g FILE: 是否存在且拥有sgid权限
    -k FILE: 是否存在且拥有sticky权限
```

6.Bash的文件属性测试

```
文件大小测试:
    -s FILE: 是否存在且非空
文件是否打开:
    -t fd: fd  文件描述符是否在某终端已经打开
    -N FILE: 文件自从上一次被读取之后是否被修改过
    -O FILE: 当前有效用户是否为文件属主
    -G FILE: 当前有效用户是否为文件属组
```

7.双目测试:

```
FILE1 -ef FILE2: FILE1是否是FILE2的硬链接
FILE1 -nt FILE2: FILE1是否新于FILE2（mtime）
FILE1 -ot FILE2: FILE1是否旧于FILE2
```

8.Bash的组合测试条件

```
第一种方式：
    [ EXPRESSION1 -a EXPRESSION2 ] 并且
    [ EXPRESSION1 -o EXPRESSION2 ] 或者
    [ ! EXPRESSION ]
取反
   -a 和 -o 需要使用测试命令进行，[[ ]] 不支持
第二种方式：
    COMMAND1 && COMMAND2  并且，短路与，代表条件性的AND  THEN
    COMMAND1 || COMMAND2  或者，短路或，代表条件性的OR  ELSE
! COMMAND   非
示例：
 [ -f “$FILE” ] && [[  “$FILE”=~ .*\.sh$  ]]
示例：
  test  "$A" =  "$B" && echo "Strings are equal"
  test  "$A"-eq  "$B"  && echo "Integers are equal"
[  "$A" =  "$B"  ] && echo "Strings are equal"
[  "$A" -eq "$B" ] && echo "Integers are equal"
[  -f /bin/cat -a -x /bin/cat ] && cat /etc/fstab
[  -z  "$HOSTNAME" -o   $HOSTNAME = "localhost.localdomain"  ] && hostname
suosuoli.cn
```

9.练习

1、编写脚本 argsnum.sh，接受一个文件路径作为参数；如果参数个数小于1， 则提示用户“至少应该给一个参数”，并立即退出；如果参数个数不小于1，则显示第一个参数所指向的文件中的空白行数

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-06
#FileName:                  argsum.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************
[[ $# -lt 1 ]] && (echo "Usage:`basename $0` /path/to/somefile" && exit) || grep
"^$" $1 | wc -l
```

2、编写脚本 hostping.sh，接受一个主机的IPv4地址做为参数，测试是否可连 通。如果能ping通，则提示用户“该IP地址可访问”；如果不可ping通，则提 示用户“该IP地址不可访问”

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-06
#FileName:                  hostping.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************

IP="\b(([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([1-9]?[0-9]|1[0-9]
{2}|2[0-4][0-9]|25[0-5])\b"
if [[ $1 =~ $IP ]]; then
    ping -W1 -c1 $1 &> /etc/null
    if [[ $? -eq 0 ]]; then
        echo -e "\e[1;32mThe host is up.\e[0m"
    else
        echo -e "\e[1;35mThe host is down!\e[0m"
    fi
else
    echo "It is not a legal ip, please check your input."

fi

unset IP
```

3、编写脚本 checkdisk.sh，检查磁盘分区空间和inode使用率，如果超过80%，就发广播警告空间将满

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-10
#FileName:                  checkdisk.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           checkdisk.sh
#Copyright (C):     2019 All rights reserved
#********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#********************************************************************************
#

D_MAX=`df -h | grep -E /dev/\(sd\|nvme\) | egrep -o "[0-9]{,3}%" | cut -d% -f1 |
```

```
sort -nr | head -n1`
 I_MAX=`df -hi | grep -E /dev/\(sd\|nvme\) | egrep -o "[0-9]{,3}%" | cut -d% -f1 |
sort -nr | head -n1`
 THREAD_D=85
 THREAD_I=85

 if [[ $D_MAX -gt $THREAD_D ]]; then
     echo -e "\e[1;31mWarning!\nThe disk space is almost full.\e[0m"
 elif [[ $I_MAX -gt $THREAD_I ]]; then
     echo -e "\e[1;31mWarning!\nThe inode is almost full.\e[0m"

 else
     echo -e "\e[1;32mNice.\e[0m"
     exit 0
 fi

 unset D_MAX
 unset I_MAX
 unset THREAD_D
 unset THREAD_I
```

4、编写脚本 per.sh，判断当前用户对指定参数文件，是否不可读并且不可写

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-10
#FileName:                  per.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           per.sh
#Copyright (C):     2019 All rights reserved
#********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#********************************************************************************
#

if [ ! -r $1 ]; then
    if [ ! -w $1 ]; then
        echo "Not readable and not writable."
    else
        echo "Not readable but writable."
    fi
elif [ -r $1 ]; then
    if [ ! -w $1 ]; then
        echo "Readable but not writable."
    else
        echo "Readable and writable."
    fi
```

```
else
    echo "ERROR!"
fi
```

5、编写脚本 excute.sh ，判断参数文件是否为sh后缀的普通文件，如果是，添加所有人可执行权限，否则提示用户非脚本文件

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-10
#FileName:                  executable.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           executable.sh
#Copyright (C):     2019 All rights reserved
#********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#********************************************************************************
#

if [[ $1 =~ \.sh$ ]]; then
    chmod +x $1
    echo  -e "Adding executing permission to $1...\nAdded."
else
    echo "$1 is not executable."
fi
```

6、编写脚本 nologin.sh和 login.sh，实现禁止和允许普通用户登录系统

```
#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-09
#FileName:                  nologin.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************

if [ -f /etc/nologin ]; then
    echo "File exists,nobody but root now can login."
else
    echo System maintenance,deny login! > /etc/nologin
```

```
fi

#!/bin/bash
#
#********************************************************************************
#Author:                        steveli
#QQ:                            1049103823
#Data:                      2019-10-09
#FileName:                  allowLogin.sh
#URL:                   https://blog.csdn.net/YouOops
#Description:           Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************
if [ -f /etc/nologin ]; then
    /usr/bin/rm /etc/nologin
else
    echo "Everyone can login."
fi
```

10.read命令

- 使用read命令来接受输入，把输入值分配给一个或多个shell变量 -p 指定要显示的提示 -s 静默输入，一般用于密码 -n N 指定输入的字符长度N -d ‘字符’ 输入结束符 -t N TIMEOUT为N秒 read 从标准输入中读取值，给每个单词分配一个变量 所有剩余单词都被分配给最后一个变量 read -p "Enter a filename: " FILE echo $FILE

条件判断if

1.单分支结构

```
if 判断条件;then
    条件为真的分支代码
fi
```

2.双分支结构

```
f 判断条件; then
    条件为真的分支代码
else
    条件为假的分支代码
fi
```

3.多分支结构

- 逐条件进行判断，第一次遇为“真”条件时，执行其分支;而后结束整个if语句

```
if 判断条件1; then
    条件1为真的分支代码
elif 判断条件2; then
    条件2为真的分支代码
elif 判断条件3; then
    条件3为真的分支代码
else
    以上条件都为假的分支代码
fi
```

条件判断case

- case语句支持glob风格的匹配

```
    *: 任意长度任意字符
    ?: 任意单个字符
    []: 指定范围内的任意单个字符
    a|b: a或b
```

```
case 变量引用 in
PAT1)
    分支1
;;
PAT2)
    分支2
;;
    ...
*)
    默认分支
;;
esac
```

练习

1、编写脚本 createuser.sh，实现如下功能：使用一个用户名做为参数，如果 指定参数的用户存在，就显示其存在，否则添加之；显示添加的用户的id号等 信息

```
#!/bin/bash
#
#********************************************************************************
```

```
#Author:            steveli
#QQ:                1049103823
#Data:              2019-10-10
#FileName:          createUser.sh
#URL:               https://blog.csdn.net/YouOops
#Description:       createUser.sh
#Copyright (C):     2019 All rights reserved
#********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#********************************************************************************
#
USER_EXIST=2
if [ $# -eq 0 ]; then
    echo "Usage:`basename $0` user"
else
    if id $1 &> /dev/null; then
        echo "User $1 exists.Exiting..."
        exit $USER_EXIST
    else
        useradd $1
        [ $? -eq 0 ] && echo -e "User $1 added.\nThe info of $1 as folloes:\n"
        id $1
    fi
fi
unset USER_EXIST
```

2、编写脚本 yesorno.sh，提示用户输入yes或no,并判断用户输入的是yes还是 no,或是其它信息

```
#!/bin/bash
#
#********************************************************************************
#Author:            steveli
#QQ:                1049103823
#Data:              2019-10-10
#FileName:          yesorno.sh
#URL:               https://blog.csdn.net/YouOops
#Description:       yesorno.sh
#Copyright (C):     2019 All rights reserved
#********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#********************************************************************************
#
read -p "Input yes or no: " ANS
case $ANS in
[Yy][Ee][Ss]|[Yy])
    echo "You'v input yes."
;;
[Nn][Oo]|[Nn])
    echo "You'v input no."
;;
```

```
*)
    echo "Ni hao tiao a ."
esac
```

3、编写脚本 filetype.sh，判断用户输入文件路径，显示其文件类型（普通，目 录，链接，其它文件类型）

```
#!/bin/bash
#
#********************************************************************************
#Author:            steveli
#QQ:                1049103823
#Data:              2019-10-10
#FileName:          exe.sh
#URL:               https://blog.csdn.net/YouOops
#Description:       Indicate file type.
#Copyright (C):     2019 All rights reserved
#********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#********************************************************************************
#

NUM=$#
declare -i J=0

if [ -f /tmp/mid ]; then
    : > /tmp/mid
fi

[ $# -eq 0 ] && echo "Usage:`basename $0` FILE1 [FILE2 ...]"

echo "$*" | tr -s " " "\n" >> /tmp/mid

exec 6< /tmp/mid
for ((j=0 ; j<=$NUM ; j++)); do
while read -u 6 I; do
        let J+=1
        if [[ -d $I ]]; then
            echo "File$J is a directory."
        elif [[ ! -e $I ]]; then
            echo "File$J not exists."
        else
            C=$(echo $(ls -l $I 2> /dev/null) | cut -c1)
            if [[ "$C" = "-" ]]; then
                echo "File$J is a normal file."
            #elif [[ "$C" = "d" ]]; then
            #    echo "File$J is a directory."
            elif [[ "$C" = "b" ]]; then
                echo "File$J is a block device file."
            elif [[ "$C" = "c" ]]; then
                echo "File$J is a character device file."
            elif [[ "$C" = "l" ]]; then
```

```
                echo "File$J is a symbolic link."
            elif [[ "$C" = "p" ]]; then
                echo "File$J is a pipe."
            elif [[ "$C" = "s" ]]; then
                echo "File$J is a socket."
            fi
        fi
done
done
```

4、编写脚本 checkint.sh，判断用户输入的参数是否为正整数

```
#!/bin/bash
#
#********************************************************************************
#Author:            steveli
#QQ:                1049103823
#Data:              2019-10-06
#FileName:          checkint.sh
#URL:               https://blog.csdn.net/YouOops
#Description:       Test scrpting.
#Copyright (C):     2019 All rights reserved
#********************************************************************************
PATTERN="^([^-]|[0-9]+)?[0-9]+$"
if [[ $1 =~ $PATTERN ]]; then
    echo "$1 is a unsigned int."
else
    echo "$1 is not a unsigned int."
fi
unset PATTERN
```

配置用户环境

1.bash展开命令行的优先次序(优先级)

```
graph TD
A(把命令行分成单个命令词) -->B(展开别名)
B-->C(展开大括号的声明)
C-->D(展开波浪符声明)
D-->E(命令替换)
E-->F(再次把命令行分成命令词)
F-->G(展开文件通配)
G-->H(准备I/0重定向)
H-->I(运行命令)
```

2.防止bash扩展某些特殊字符

```
反斜线（\）会使随后的字符按原意解释
    echo Your cost: \$5.00
     Your cost: $5.00
加引号来防止扩展
•单引号（’’）防止所有扩展
•双引号（”“）也可防止扩展，但是以下情况例外：
    $（美元符号） 变量扩展
    ` ` (反引号）    命令替换
    \（反斜线）    禁止单个字符扩展
    !（叹号）      历史命令替换
```

3.bash的配置文件

- 按生效范围划分，存在两类：

```
全局配置：
    /etc/profile
    /etc/profile.d/*.sh
    /etc/bashrc
个人配置：
    ~/.bash_profile
    ~/.bashrc
```

- 按功能划分，存在两类：

```
Profile类
profile类：为交互式登录的shell提供配置
    全局：/etc/profile, /etc/profile.d/*.sh
    个人：~/.bash_profile
功用：
    (1) 用于定义环境变量
    (2) 运行命令或脚本
```

```
Bashrc类
bashrc类：为非交互式和交互式登录的shell提供配置
    全局：/etc/bashrc
    个人：~/.bashrc
功用：
    (1) 定义命令别名和函数
    (2) 定义本地变量
```

4.shell登录两种方式

- 交互式登录：

```
(1)直接通过终端输入账号密码登录
(2)使用“su - UserName” 切换的用户
执行顺序：/etc/profile --> /etc/profile.d/*.sh --> ~/.bash_profile -->
~/.bashrc --> /etc/bashrc
```

- 非交互式登录：

```
(1)su UserName
(2)图形界面下打开的终端
(3)执行脚本
(4)任何其它的bash实例
执行顺序： /etc/profile.d/*.sh --> /etc/bashrc -->~/.bashrc
```

5.编辑配置文件后使其生效的方式

```
修改profile和bashrc文件后需生效
两种方法:
1重新启动shell进程
2使用.命令或source 命令
例:
    .   ~/.bashrc
```

- 退出登录时让Bash执行退出任务

```
任务保存在~/.bash_logout文件中（用户）
在退出登录shell时运行
作用
    •创建自动备份
    •清除临时文件
```

6.set命令和$-变量

- $- 变量保存shell的一些功能开关

```
h：hashall，打开这个选项后，Shell 会将命令所在的路径hash下来，避免每
次都要查询。通过set +h将h选项关闭
i：interactive-comments，包含这个选项说明当前的 shell 是一个交互式的
shell。所谓的交互式shell,在脚本中，i选项是关闭的
m：monitor，打开监控模式，就可以通过Job control来控制进程的停止、继
续，后台或者前台执行等
B：braceexpand，大括号扩展
H：history，H选项打开，可以展开历史列表中的命令，可以通过!感叹号来
完成，例如“!!”返回上最近的一个历史命令，“!n”返回第 n 个历史命令
```

7.脚本安全相关

- 使用set命令可以避免一些灾难性的错误

```
-u      在扩展一个没有设置的变量时，显示错误信息
        等同set -o nounset
-e     如果一个命令返回一个非0退出状态值(失败)就退出
        等同set -o errexit
```

8.练习

1、让所有用户的PATH环境变量的值多出一个路径，例如： /usr/local/apache/bin

```
echo "PATH=/usr/local/apache/bin:$PATH" >> /etc/profile.d/env.sh
```

2、用户 root 登录时，将命令指示符变成红色，并自动启用如下别名：

```
echo "PATH=/usr/local/apache/bin:$PATH" >> /etc/profile.d/env.sh
cat <<EOF
rm='rm -i'
cdnet='cd /etc/sysconfig/network-scripts/'
editnet='vim /etc/sysconfig/network-scripts/ifcfg-eth0'
######
V=`cat /etc/redhat-release | grep -o "[0-9]\+" | cut -d. -f1 | head -n1`
NET0=iifconfig | grep -o "ens[0-9]\+" | cut -ds -f2 | head -n1
NET1=iifconfig | grep -o "ens[0-9]\+" | cut -ds -f2 | head -n2 | tail -n1
if [ $V = 7 ]; then
alias    editnet='vim /etc/sysconfig/network-scripts/ifcfg-ens33'
elif [ $V = 8 ]; then
alias    editnet0='vim /etc/sysconfig/network-scripts/ifcfg-ens160'
alias    editnet1='vim /etc/sysconfig/network-scripts/ifcfg-ens192'
fi
EOF
```

3、任意用户登录系统时，显示红色字体的警示提醒信息“Hi,dangerous！”

```
echo 'echo -e "\e[1;31mHi,dangerous!\e[0m"' > /etc/profile.d/Warning.sh
```

4、编写生成脚本基本格式的脚本，包括作者，联系方式，版本，时间，描述等

```
 set number
 set tabstop=4
```

```
 set softtabstop=4
 set shiftwidth=4
 set expandtab
 set ignorecase
 set cursorline
 set autoindent
 autocmd BufNewFile *.sh exec ":call SetTitle()"
 func SetTitle()
     if expand("%:e") == 'sh'
         call setline(1,"#!/bin/bash")
         call setline(2,"#")
         call
setline(3,"#**************************************************************************
*********")
         call setline(4,"#Author:            steveli")
         call setline(5,"#QQ:                1049103823")
         call setline(6,"#Data:              ".strftime("%Y-%m-%d"))
         call setline(7,"#FileName:          ".expand("%"))
         call setline(8,"#URL:               https://blog.csdn.net/YouOops")
         call setline(9,"#Description:       ".expand("%"))
         call setline(10,"#Copyright (C):        ".strftime("%Y")." All rights
reserved")
         call
setline(11,"#**************************************************************************
**********")
         call
setline(12,"#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):w
hite(37)")
         call
setline(13,"#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):w
hite(47)")
         call
setline(14,"#**************************************************************************
**********")
         call setline(15,"#")
         call setline(16,"")
         endif
 endfunc
 autocmd BufNewFile * normal G
```

5、编写用户的环境初始化脚本reset.sh，包括别名，登录提示符，vim的设置， 环境变量等

```
#!/bin/bash
#
#**************************************************************************
#Author:            steveli
#QQ:                1049103823
#Data:              2019-08-29
#FileName:          reset6.sh
#URL:               https://blog.csdn.net/YouOops
#Description:       This script doing some init configurations on centos7&8.
#Copyright (C):     2019 All rights reserved
```

```
#*********************************************************************************
#*********************************************************************************
#Fontcolor#red(31):green(32):yellow(33):blue(34):purple(35):cyan(36):white(37)
#Backcolor#red(41):green(42):yellow(43):blue(44):purple(45):cyan(46):white(47)
#*********************************************************************************
VIMRC=~/.vimrc
BASHRC=~/.bashrc
PROMPT_PATH=/etc/profile.d/
HOSTNAME=stevenux
HOSTNAME_PATH=/etc/hostname
HOSTS_PATH=/etc/hosts

#1.Prompt configuration
echo "1.Entering to /etc/profile.d/ ..."

sleep 1
cd $PROMPT_PATH
echo "Create env.sh..."
sleep 1

if [ ! -e ${PROMPT_PATH}/env.sh ]; then
touch env.sh
cat >> env.sh <<EOF
#!/bin/bash
PS1="\[\e[37;40m\][\[\e[32;40m\]\u\[\e[37;40m\]@\h \[\e[36;40m\]\w\[\e[0m\]]\$"
EOF
fi

echo "Change prompt style..."
. /etc/profile.d/env.sh
echo "1.done"
echo


#2.VIM style configuration
echo "2.VIM style configuration..."
sleep 1

if [ ! -f $VIMRC ];then
touch $VIMRC
cat >> $VIMRC <<EOF
set number
set tabstop=4
set softtabstop=4
set shiftwidth=4
set expandtab
set ignorecase
set cursorline
set autoindent
autocmd BufNewFile *.sh exec ":call SetTitle()"
func SetTitle()
    if expand("%:e") == 'sh'
        call setline(1,"#!/bin/bash")
        call setline(2,"#")
```

```
 call setline(1,"#!/bin/bash")
         call setline(2,"#")
         call
setline(3,"#**************************************************************************
*********")
         call setline(4,"#Author:            steveli")
         call setline(5,"#QQ:                1049103823")
         call setline(6,"#Data:              ".strftime("%Y-%m-%d"))
         call setline(7,"#FileName:          ".expand("%"))
         call setline(8,"#URL:               https://blog.csdn.net/YouOops")
         call setline(9,"#Description:       Test scrpting.")
         call setline(10,"#Copyright (C):        ".strftime("%Y")." All rights
reserved")
         call
setline(11,"#**************************************************************************
*********")
         call setline(12,"")
         endif
 endfunc
 autocmd BufNewFile * normal G
 EOF
 fi
 . $VIMRC &> /dev/null
 echo "2.done"
 echo


 #3.BASH style configuration.
 echo "3.BASH style changing..."
 sleep 1

 if [ -e $BASHRC ]; then
 cat >> $BASHRC <<EOF
 #my customized command
 alias rm='rm -i'
 alias cp='cp -i'
 alias mv='mv -i'
 alias ll='ls -l'
 alias la='ls -a'
 alias cls='clear'
 alias scandisk='echo - - - > /sys/clss/scsi_host/host2/scan'
 alias cdnet='cd /etc/sysconfig/network-scripts'
 alias netstat='netstat -ntupa'
 alias route='route -n'
 alias arp='arp -n'
 alias ipr='ip route'
 alias ipl='ip link'
 alias cdnet='cd /etc/sysconfig/network-scripts'
 alias bashn='bash -n'
 alias bashx='bash -x'
 alias freeh='free -h'
 alias yy='yum install -y'
 alias ee='echo $?'
 EOF
```

```
    else
        touch $BASHRC
    cat >> $BASHRC <<EOF
    #my customized command
    alias rm='rm -i'
    alias cp='cp -i'
    alias mv='mv -i'
    alias ll='ls -l'
    alias la='ls -a'
    alias cls='clear'
    alias scandisk='echo - - - > /sys/clss/scsi_host/host2/scan'
    alias cdnet='cd /etc/sysconfig/network-scripts'
    alias netstat='netstat -ntupa'
    alias route='route -n'
    alias arp='arp -n'
    alias ipr='ip route'
    alias arp='arp -n'
    alias ipr='ip route'
    alias ipl='ip link'
    alias cdnet='cd /etc/sysconfig/network-scripts'
    alias bashn='bash -n'
    alias bashx='bash -x'
    alias freeh='free -h'
    alias yy='yum install -y'
    alias ee='echo $?'
    EOF
    . $BASHRC &> /dev/null
    fi

    echo "3.done"
    echo

    #4,5,6 From Jibill Chen:Contact him:http://www.jibiao.work
    #4.Configure cd_rom for yum.
    #CD_NAME=$(lsblk | grep 'sr[0-9]'|head -1 | cut -c 1,2,3)
    #[ $(df | grep 'sr' | wc -l) -ge 1 ] && CD_MOUNT_POINT=$(df |sed -nr
'/dev\/sr/s@^/dev.*% @@p') || { mount /dev/$CD_NAME /mnt &> /dev/null ;
CD_MOUNT_POINT='/mnt';}
    mkdir -p /mnt/cdrom &> /dev/null
    CD_MOUNT_POINT="/mnt/cdrom"
    umount /dev/sr0 &> /dev/null
    mount /dev/sr0 $CD_MOUNT_POINT &> /dev/null
    rm -f /etc/yum.repos.d/*.repo &> /dev/null
    cat > /etc/yum.repos.d/cdrom.repo << EOF
    [cdrom]
    name=cdrom
    baseurl=file://${CD_MOUNT_POINT}
            https://mirrors.aliyun.com/epel/7/x86_64/
    enabled=1
    gpgcheck=0
    EOF
    [ $? -eq 0 ] && echo -e "${CO_GREEN}Configure yum of cdrom successful!${LOR}" ||
echo -e "${CO_RED}Configure yum of cdrom fail!${LOR}"
    sleep 1
```

```
 #5.Install tree screen vim bzip2
 yum clean all &> /dev/null
 yum repolist &> /dev/null
 yum -y install chrony  &> /dev/null
 [ $? -eq 0 ] && echo -e "${CO_GREEN}install chrony successful!${LOR}" || echo -e
"${CO_RED}install tree fail!${LOR}"
 yum -y install wget tree &> /dev/null
 [ $? -eq 0 ] && echo -e "${CO_GREEN}install wget tree successful!${LOR}" || echo
-e "${CO_RED}install tree fail!${LOR}"
 yum -y install screen net-tools &> /dev/null
 [ $? -eq 0 ] && echo -e "${CO_GREEN}install screen net-tools successful!${LOR}"
|| echo -e "${CO_RED}install tree fail!${LOR}"
 yum -y install  vim bzip2 &> /dev/null
 [ $? -eq 0 ] && echo -e "${CO_GREEN}install vim bzip2 successful!${LOR}" || echo
-e "${CO_RED}install tree fail!${LOR}"

 #6.Disable iptables and firewalld
 #
 systemctl stop firewalld &> /dev/null
 [ $? -ne 0 ] && echo "${CO_RED}stop firewalld fail${LOR}"
 systemctl disable firewalld &> /dev/null
 [ $? -ne 0 ] && echo "${CO_RED}disable firewalld fail${LOR}"
 iptables -F &> /dev/null
 [ $? -ne 0 ] && echo "${CO_RED}shutdown iptables fail${LOR}"
 sed -r -i 's#(^SELINUX=)(.*$)#\1disabled#' /etc/selinux/config &> /dev/null
 [ $? -ne 0 ] && echo "${CO_RED}modify selinux/config fail${LOR}"
```