shell 001

=============================================

**一，shell基本介绍**

**1，什么是shell**

在Linux内核与用户之间的解释器程序

通常指 /bin/bash

负责向内核翻译及传达用户/程序指令

相当于操作系统的“外壳”

**shell使用方式：**

**交互式 —— 命令行**

人工干预、智能化程度高

逐条解释执行、效率低

**非交互式 —— 脚本**

需要提前设计、智能化难度大

批量执行、效率高

方便在后台静悄悄地运行

[root@svr7 ~]# cat /etc/shells //查看所有解释器

[root@svr7 ~]# sh //切换成sh解释器

sh-4.2# ls //利用sh解释器输入命令

sh-4.2#exit //退出sh解释器

[root@svr7 ~]#yum -y install ksh //安装新解释器

[root@svr7 ~]#ksh //进入新解释器

**bash优点：tab键、快捷键、历史命令、支持别名、管道、重定向**

**shell脚本：提前写好可执行的语句，可以完成特定任务的文件**

**按顺序、批量化执行**

**shell脚本编写规范**

**1，声明解释器**

**#!/bin/bash**

**2，编写注释**

**#可以描述脚本功能、变量作用等信息**

**3，执行指令**

**ls**

**cd**

。。。

。。。

**脚本的执行方式**

**1，为脚本添加x权限，使用相对或者绝对路径执行**

chmod u+x test01.sh

./test01.sh 或者 /opt/test01.sh

用户（root）---bash---bash---echo

**2，使用解释器程序执行脚本，无需x权限**

bash test01.sh

用户（root）---bash ---bash---echo 会开启新的子进程

**3，使用source命令执行脚本，无需x权限**

source test01.sh

用户（root）---bash---echo 不会开启新的子进程

**编写脚本，搭建yum**

#!/bin/bash

#搭建yum软件仓库

mkdir /abc

mount /dev/cdrom /abc

rm -rf /etc/yum.repos.d/\*.repo

echo "[abc]

name=abcabc

baseurl=file:///abc

enabled=1

gpgcheck=0" > /etc/yum.repos.d/abc.repo

**编写脚本，安装网站服务并开启，浏览网站时显示"web-test~~"**

#!/bin/bash

#搭建网站

yum -y install httpd

echo "web-test~~" > /var/www/html/index.html

systemctl restart httpd

**编写脚本，安装ftp服务并开启且设置为开机自启**

> 重定向标准输出

2> 重定向错误输出

&> 重定向所有输出

#!/bin/bash

yum -y install vsftpd &> /dev/null //将不需要的信息扔黑洞

systemctl restart vsftpd

systemctl enable vsftpd

----------------------------------------------------------------------

**变量 使用固定的名称存放可能发生变化的值，可以提高脚本的**

**灵活度、适应力**

**常量 固定不变的内容**

**变量的种类**

1. **自定义变量，名称可以用数字、字母、下划线，不能以数字**

开头，不能使用特殊符号，等号两边不能有空格

变量名称=值

a=10 //创建变量（对变量赋值），名字是a，值是10

a=30 //再次赋值，之前的会被覆盖

echo $a //调用变量时使用$符号

unset a //取消变量的定义

a= //将变量a赋值为空，效果同上

echo ${a}RMB //变量名容易与后续字符发生混淆时使用大括号隔开

**2，环境变量，由系统提前定义好，使用时直接调用**

USER 当前用户名 UID当前用户的id号 HOME当前用户家目录

HOSTNAME 主机名 SHELL当前用户的解释器 PWD当前位置

PATH 存储命令的路径 PS1 一级提示符 PS2 二级提示符

**3，位置变量与预定义变量**

$1 执行脚本时后面第1个位置参数

$2 执行脚本时后面第2个位置参数

$3 执行脚本时后面第3个位置参数

$\* 执行脚本时后面所有位置参数

$# 执行脚本时后面位置参数的个数

$$ 随机进程号

$? 判断上一条指令是否执行成功，0是成功，非0是失败

**编写脚本，创建用户tom，配置密码123456**

#!/bin/bash

useradd tom

echo "123456" | passwd --stdin tom

**改良版**

#!/bin/bash

useradd $1

echo "$2" | passwd --stdin $1

env 查看所有环境变量

set 查看所有变量

-------------------------------------------------

**变量的扩展知识**

**1，引号和反撇号**

" " 双引号 界定范围

a=10

echo "$a" //正常调用变量

' ' 单引号 界定范围 屏蔽特殊符号的功能

echo '$a' //无法调用变量，$被屏蔽

` ` 反撇号 获取命令的执行结果，还可以使用$( )实现相同效果

a=date //仅仅将四个字母赋值给a

a=`date` //将date执行结果赋值给a

a=$(date) //效果同上

**2，使用read指令**

#!/bin/bash

read -p "请输入用户名" u //-p是可以定义提示信息，u相当于自

定义变量名称，可以存储用户看到提示信息后输入的字符

useradd $u

read -p "请输入密码" n

echo "$n" | passwd --stdin $u

stty -echo 屏蔽回显

stty echo 恢复回显

再次改良脚本：

#!/bin/bash

read -p "请输入用户名" u

useradd $u

stty -echo

read -p "请输入密码" n

stty echo

echo "$n" | passwd --stdin $u

**3，使用export发布全局变量**

局部变量 仅仅在当前解释器进程中使用的变量

全局变量 解释器产生的子进程中也可以使用的变量

a=10 //创建变量

export a //将已有变量发布为全局效果

export b=20 //创建变量的同时发布为全局效果

export -n b //将全局变量恢复为局部变量

注意，测试时要在父进程创建变量，然后可以执行bash进入

子进程测试效果，如果要返回父进程要执行exit指令

------------------------------------------------

**运算**

**加、减、乘、除、求模(取余数)**

**使用expr命令，可以运算并输出结果**

expr 1 + 1 //加法，运算符号两边要有空格

expr 2 – 1 //减法

expr 2 '\*' 2 //乘法，单引号可以屏蔽特殊符号\*的功能，本身

是通配符，而不是乘号，屏蔽之后就可以识别为乘号

expr 2 \\* 2 // \是转义符号，可以屏蔽身后一个特殊符号的功能

expr 4 / 2 //除法

expr 5 % 2 //取余数

a=10

b=20

expr $a + $b

expr $a + 100

expr $a + $a

**2，$[ ] 运算 使用$(( )) 可以实现相同效果**

echo $[2\*2]

echo $[2-2]

echo $[2/2]

echo $[2%2]

a=10

b=20

expr $a + $b

echo $[a+b]

-------------------------------------------------------------------------------------

练习题：

1，如何执行Shell脚本？

2 自定义Shell变量时，有哪些注意事项？

3 编写一个Shell脚本程序，用来报告当前用户的环境信息。

4 简述预定义变量$$、$?、$#、$\*的作用。

5 编写一个Shell脚本程序，能够部署ftp服务，并开启ftp默认共享的目录权限。

6 简述单引号、双引号、反撇号在变量赋值操作中的特点。

1，如何执行Shell脚本？

方法一：添加x权限，然后使用路径运行

方法二：bash 脚本文件路径

方法三：source 脚本文件路径

2 自定义Shell变量时，有哪些注意事项？

可以包括数字、字母、下划线，不能以数字开头

赋值时等号两边不要有空格

尽量不要使用关键字和特殊字符

给同一个变量多次赋值时，最后一次的赋值生效

3 编写一个Shell脚本程序，用来报告当前用户的环境信息。

新建脚本文件report.sh，执行后能够输出当前的主机名、登录用户名、所在的文件夹路径。

#!/bin/bash

echo "当前的主机名是：$HOSTNAME"

echo "登录用户是：$USER"

echo "当前位于 $PWD 文件夹下"

4 简述预定义变量$$、$?、$#、$\*的作用。

$$ 保存当前运行进程号

$? 保存命令执行结果的（返回状态）0是成功，非0是失败

$# 保存位置变量的（个数）

$\* 保存所有位置变量的（值）

5 编写一个Shell脚本程序，能够部署ftp服务，并开启ftp默认共享的目录权限。

#!/bin/bash

yum -y install vsftpd &> /dev/null

systemctl restart vsftpd

systemctl enable vsftpd

chmod 777 /var/ftp/pub //pub目录是默认的共享目录，777可以开放所有权限

6 简述单引号、双引号、反撇号在变量赋值操作中的特点。

双引号 " "：可以界定范围

单引号 ' '：可以界定范围，还可以屏蔽特殊符号，即便 $ 也视为普通字符

反撇号 ` `：可以获取命令的执行结果