**shell day 01**

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

**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的使用方式**

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

**什么是Shell脚本？**

提前写好可执行语句，能够完成特定任务的文件

顺序、批量化处理

[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**

**脚本的编写规范：**

1. 声明解释器

#!/bin/bash

1. 注释

#内容可以脚本功能的描述、作者介绍、脚本中各种参数的含义等

1. 执行指令

ls

echo

cd

**脚本的执行方式**

1. 添加x权限，使用相对或者绝对路径

./test01.sh /opt/test01.sh

2，使用解释器执行脚本，无需x权限，会开启解释器子进程

bash test01.sh

3，使用source命令执行脚本，无需x权限，不开启解释器子进程

source test01.sh

可以在另外一个命令行终端使用pstree命令查看

父进程---子进程

bash---bash---test01.sh //使用解释器执行脚本

bash---test01.sh //使用source执行脚本

#!/bin/bash

#这是个测试脚本

echo abc

mkdir xyz

cd xyz

touch 666

//使用bash执行看不到进入xyz目录的效果，因为bash开启了子进程

执行完毕后退出了，父进程并没有进入xyz目录，而使用source执行脚

本可以看到进入xyz目录，因为没有开启子进程，而是父进程亲自进入

该目录

**编写部署yum的脚本**

#!/bin/bash

#搭建yum仓库

mkdir /yum

mount /dev/cdrom /yum

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

echo "[abc]

name=abcabc

baseurl=file:///yum

gpgcheck=0

enabled=1" > /etc/yum.repos.d/abc.repo

**编写脚本，部署httpd，定义默认网站页面是”web\_test~~~~~~”**

#!/bin/bash

#部署网站服务

yum -y install httpd

echo “web\_test~~~~~~” > /var/www/html/index.html

systemctl start httpd

**测试时需要临时关闭防火墙**

systemctl stop firewalld

运行脚本后，可以利用真机的浏览器打开虚拟机的网页

http://192.168.4.7

**编写部署ftp服务的脚本，要求可以实现开机自启**

#!/bin/bash

yum -y install vsftpd &> /dev/null

systemctl start vsftpd

systemctl enable vsftpd

systemctl status vsftpd //查看

>

echo abc > 123.txt

2>

常量 固定不变的值（内容）

变量 以固定名称存放可能发生变化的值，可以提高脚本对任务的需求、运行环境变化的适应能力，方便在脚本中重复使用

变量的种类：

1. 自定义变量，定义名称可以用数字、字母、下划线，不能用数字开头，不能使用特殊符号

变量名称=变量的值

[root@svr7 opt]# a=10 //定义变量（赋值）

[root@svr7 opt]# echo $a //调用变量，此时$a相当于10

**使用自定义变量要注意：**

1. a =10 等号两边不能有空格
2. a=10 然后 a=20 多次赋值之后，仅最后一次生效

3，如果变量不需要了可以使用unset a 取消，或者a=

4，echo ${a}RMB 当变量名与后续常量发生混淆时要用括号隔开

2，环境变量，是由系统提前定义好，使用时直接调用

USER当前用户名 HOME当前用户的家目录

UID 当前用户的id号 PWD 当前位置

SHELL 当前用户的解释器 HOSTNAME 完整主机名

PS1 一级提示符 PS2 二级提示符

PATH 存储了执行指令的路径

1. 预定义变量与位置变量

$\* $# $$ $? $1 $2 $3 ….

#!/bin/bash

echo $1 执行脚本后面跟的第1个位置参数

echo $2 执行脚本后面跟的第2个位置参数

echo $3 执行脚本后面跟的第3个位置参数

echo $\* 所有的位置参数

echo $# 所有的位置参数的个数

echo $$ 随机的进程号

echo $? 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，可以创建用户abc01 并配置密码123456

#!/bin/bash

useradd abc01

echo 123456 |passwd --stdin abc01

改良版：可以根据需要创建任意名称的用户以及随意的密码

#!/bin/bash

useradd $1 //创建用户时调用第1个位置变量

echo $2 |passwd --stdin $1 //配置密码时调用第2个位置变量

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

env //查看所有的环境变量

env | grep HOSTNAME //在所有的环境变量中查询HOSTNAME

set //查看所有变量

set | grep abcd //过滤查询，可以先提前定义变量abcd=8888

变量的扩展知识

1. **引号与反撇号**

**"" 双引号 界定范围**

touch "a b" //可以创建a空格b的文件

a=10

echo "$a" //可以调用变量

''  **单引号 界定范围 屏蔽特殊符号的功能**

touch 'a b' //也可以创建a空格b的文件

echo '$a' //无法调用变量，$是调用变量的特殊符号，被单引号屏蔽

``  **反撇号 调用命令的执行结果，或者使用$( )**

a=date //定义变量时如果使用命令则不会直接识别

a=`date` //使用反撇号可以将命令的执行结果赋值给变量

a=$(date) //效果同上

2，使用read指令，以交互式使用脚本

read -p "请输入用户名：" u

useradd $u

read -p "请输入密码：" p

echo $p |passwd --stdin $u

stty -echo //屏蔽回显

stty echo //恢复回显

改良版

#!/bin/bash

read -p "请输入用户名：" u

useradd $u

stty -echo

read -p "请输入密码:" p

stty echo

echo $p |passwd --stdin $u