Shell 编程

一．shell 的作用和历史

二．shell 的常用功能

三．shell 变量种类

四．shell 脚本的运行

五．编写简单的 shell 脚本

一. shell 的作用和历史

Shell 的作用 -- 命令解释器，“翻译官”

vim /etc/shells　　shell 文件

二. shell 常用功能

自动补全　Tab

命令历史　history

history -w　同步历史命令/写入隐藏文件

history -c　清除历史记录

!n：执行历史记录中的第 n 条命令

!str：执行历史记录中以“str”开头的命令

vim /etc/profile

48 HISTSIZE=1000

为使用频率较高的复杂命令行设置简短的调用名称

存放位置：~/.bashrc

vim /root/.bashrc

查看命令别名

格式：alias [别名]

设置命令别名

执行：alias 别名='实际执行的命令'

alias ls='ls -lha'

ls

取消已设置的命令别名

格式：unalias 别名

unalias ls

输入重定向

man wc

wc < install.log

输出重定向

ls > bak.list

ls -l >> bak.list

标准错误输出

lss 2> bak.list

lss 2>> bak.list

重定向标准输出和标准错误

ls &> /dev/null 黑洞设备文件（空设备文件）

lss >> a.txt 2>&1 重定向标准错误

管道符 |

```
netstat -an | grep ESTABLISHED | wc -l  统计与服务器的连接数量
```

三 .Shell 变量的应用

Shell 变量的种类

用户自定义变量：由用户自己定义、修改和使用

环境变量：由系统维护，用于设置用户的 Shell 工作环境，只有极少数的变量用户可以修改

预定义变量：Bash 预定义的特殊变量，不能直接修改

位置变量：通过命令行给程序传递执行参数

变量的赋值与引用

定义新的变量

变量名要以英文字母或下划线开头，区分大小写

格式：变量名=变量值

查看变量的值

格式：echo $变量名

查看所有变量：set

清除变量

unset 变量名

定义变量

Var=lamp

echo ${var}3.0

""   ''   ``   单引号 双引号 反引号 对比

```
DAY=xingqier
echo $DAY
echo  "$DAY"
echo '$DAY'
echo `$DAY`

unset DAY
DAY=ls
echo `$DAY`
```

环境变量赋值

设置变量的作用范围

格式：export   变量名...

        export 变量名=变量值 [...变量名 n=变量值 n]

查看环境变量

env 或 export

清除用户定义的变量

格式：unset 变量名

```
命令执行时查找顺序
1、以相对/绝对路径执行
2、由 alias 找到的执行
3、bash 内部命令执行
4、按$PATH 路径执行

环境变量 PS1
echo $PS1
          \d  日期  \t  时间（24）\T 时间（12）
          \H  完整主机名      \h  简写主机名
          \u  用户名                \v  bash 版本
          \w  完整目录         \W  最后一个目录
          \   执行了第几个命令     \$  提示符
PS1= ‘[\u@\h  \W  \t  \]\$’
```

```
预定义变量

表示形式如下
$：命令行中位置参数的个数
$*：所有位置参数的内容
$?：上一条命令执行后返回的状态，当返回状态值为 0 时表示执行正常，非 0 值表示执行异常或
出错
$$：当前所在进程的进程号
$!：后台运行的最后一个进程号
$0：当前执行的进程/程序名

[root@localhost ~] bash
[root@localhost ~] echo  $0  $$
bash 5887
[root@localhost ~] exxit
bash: exxit: command not found
[root@localhost ~] echo $?
127
[root@localhost ~] exit
exit
[root@localhost ~] echo $?
0
```

```
;   命令顺序执行。
&& 前后命令的执行存在逻辑与关系，只有&&前面的命令执行成功后，它后面的命令才被执行。
||   前后命令的执行存在逻辑或关系，只有||前面的命令执行失败后，它后面的命令才被执行。
```

```
通配符与特殊符号
通配符
        *    任意多个
        ?    任意一个
        []   括号内任一个   [^0-9]非数字

rm   -rf  *
rm   -rf  ?.??
touch   1.txt  2.txt  3.txt
rm -rf  [1-9].*

特殊符号
        \    转义符
        &    后台
        !    非
```

四. Shell 脚本的概念

```
Shell 脚本
1.用途：完成特定的、较复杂的系统管理任务
2.格式：集中保存多条 Linux 命令，普通文本文件
3.执行方式：按照预设的顺序依次解释执行
```

```
[root@localhost ~] vi repboot.sh
!/bin/bash
 To show usage of /boot directory and mode of kernel file.
echo "Useage of /boot: "
du -sh /boot
echo "The mode of kernel file:"
ls -lh /boot/vmlinuz-*
[root@localhost ~] chmod a+x repboot.sh
```

```
运行 Shell 脚本程序
```

1.直接执行具有“x”权限的脚本文件
 例如：./repboot.sh
2.使用指定的解释器程序执行脚本内容
 例如：bash repboot.shss
3.通过 source 命令（或 . ）读取脚本内容执行
 例如：souce repboot.sh 或 . hello.sh
```

 五. Shell 脚本应用示例

```
示例 1：
每周五 17:30 清理 FTP 服务器的公共共享目录
 检查 /var/ftp/pub/ 目录，将其中所有子目录及文件的详细列表、当时的时间信息追加保存到 /var/log/pubdir.log 日志文件中，然后清空该目录
```

```
[root@localhost ~] vi /opt/ftpclean.sh
!/bin/bash
date  >>  /var/log/pubdir.log
ls  -lhR  /var/ftp/pub  >>  /var/log/pubdir.log
rm  -rf  /var/ftp/pub/*

[root@localhost ~] crontab -e
30 17 * * 5  /opt/ftpclean.sh

chmod  +x   /opt/ftpclean.sh
```

```
示例 2:
每隔 3 天时间 3：30 对数据库目录做一次完整备份
 统计 /usr/local/mysql/var 目录占用的空间大小、查看当前的日期，并记录到临时文件 /tmp/dbinfo.txt 中， 将 /tmp/dbinfo.txt 文件、/usr/local/mysql/var 目录进行压缩归档，备份到/opt/dbbak/目录中， 备份后的包文件名中要包含当天的日期信息， 最后删除临时文件 /tmp/dbinfo.txt
```

```
[root@localhost ~] vi /opt/dbbak.sh
!/bin/bash
DAY=`date +%Y%m%d`
SIZE=`du -sh /usr/local/mysql/var`
echo "Date: $DAY" >> /tmp/dbinfo.txt
echo "Data Size: $SIZE" >> /tmp/dbinfo.txt
mkdir  /opt/dbbak
```
```

```
cd /opt/dbbak
tar -zcPf mysqlbak-${DAY}.tar.gz /usr/local/mysql/var /tmp/dbinfo.txt
rm -f /tmp/dbinfo.txt

[root@localhost ~] crontab -e
30 3 */3 * *  /opt/dbbak.sh

chmod  +x  /opt/dbbak.sh
```

作业

课堂笔记 写一遍

代码至少三遍