shell

编程基础

- vi/vim 编辑器的命令 vimrc设置
- 命令基础 100多个命令
- 基础 高端的网络服务 nfs rsyno inotify lanmp sersync ssh key批量分发管理

shell脚本简介

shell是一个命令解释器，负责直接与用户对话，把用户的输入解释给操作系统，并处理各种各样的操作系统的输出结果，并在屏幕展示给用户

清空日志的三种方法(主要用于保留文件，清空内容)：

- echo "">test.log
- >test.log
- cat /dev/null >test.log

shell脚本很擅长处理纯文本类型的数据，而linux中几乎所有的配置文件，日志文件都是纯文本类型的文件。

脚本语言的种类：

- bourne shell(sh,ksh,bash)
- C shell(csh,tcsh)

shell脚本语言是弱类型语言。

centos 默认的shell是**bash**.

规范的shell脚本

- 脚本开头

  > 一个规范的shell脚本的第一行会指出哪个程序来执行脚本中的内容。如果脚本的开头不指定解释器，那么就要用对应的解释器来执行脚本。

  ```
  #! /bin/bash 或者 #! /bin/sh
  #! 又称为幻数，内核会根据它来确定用哪个程序来解释脚本中的内容。
  ```

- 注释

shell脚本的执行

当shell脚本以非交互的方式运行时，它会先查找环境变量ENV，该变量指定了一个环境文件（通常是.bashrc），然后从该环境变量文件开始执行，当读取了ENV文件后，SHELL才开始执行shell脚本中的内容。

shell脚本的执行通常可以采用以下三种方式：

- bash script-name或sh script-name
- path/script-name或 ./script-name
- source script-name 或 . script-name 点号后面有空格的

执行说明：

第一种说明：当脚本文件本身没有可执行权限（即文件x位为-号）时常使用的方法，这里推荐使用bash执行，或者文件开头没有指定解释器。

第二种方法需要先将脚本权限设置为可执行，然后通过脚本路径就可以直接执行脚本了。

第三种方法通常是使用source或者“.”点号读入或者加载指定的shell脚本文件（san,sh），然后，依次执行指定的shell脚本文件中的所有语句。这些语句将作为当前父shell脚本father.sh进程的一部分运行，因此，使用source或者“.”点号可以将san.sh自身脚本中的变量的值或函数等的返回值传递到当前的父shell脚本father.sh中使用。这是第三种方法和前两种方法的最大区别。

shell脚本的九点规范

1. 开头指定脚本解释器
2. 开头加版本版权等信息
3. 脚本中不用中文注释
4. 脚本以.sh为扩展名
5. 代码书写优秀习惯

> 1. 成对的内容一次写出来，反之遗漏
> 2. []中括号两端要有空格，书写时即可留出空格[ ],然后再退格书写内容。
> 3. 流程控制语句一次书写完，再添加内容
> 4. 通过缩进使代码易读。

shell变量

分为两类 环境变量（全局变量）和局部变量

环境变量称为全局变量，可以在创建他们的shell及其派生出来的任意子进程shell中使用，局部变量只能在创建他们的shell函数或脚本中使用，还有一些变量是用户创建的，其他的则是专用shell变量。

环境变量

环境变狼可以在命令行中设置，但用户推出时这些变量值也会丢失，因此最好在用户家目录下的.bash_profile文件中或全局配置/etc/bashrc,/etc/profile文件或者/etc/profile.d/中定义。将环境变量放入profile中，每次用户登陆时这些变量值都会初始化。

环境变量最好全为大写，应用于用户进程前，必须用export命令导出。

全局变量赋值方法

1. export变量名=value
2. 变量名=value export 变量名
3. declare -x 变量名=value

用env或set显示默认的环境变量。

用unset来取笑环境变量。

局部变量

本地变量，在用户当前的shell生存期的脚本中使用。如果在shell中启动另一个进程或退出，本地变量的值也会取消。

shell变量名的要求：一般是字母，数字，下划线组成，字母开头。

定义变量的三种方法：

1. 直接定义变量内容，内容一般为简单连续的数字，字符串，路径名等。
2. 通过单引号定义变量，这种方法的特点是：输出变量时引号里是什么就输出什么，即使内容中有变量也会把变量名原样输出，此法比较适合于定义显示纯字符串。
3. 通过双引号定义变量，这种方式的特点是:输出变量时引号里的变量会经过解析输出该变量内容，而不是把引号中变量名原样输出，适合于字符串中附带有变量的内容的定义。

习惯：数字不加引号，字符加双引号。

shell特殊变量

位置变量

- $0 获取当前执行的脚本文件名 包括路径

- $n 获取当前执行的脚本的第n个参数值，n为0...9，加入大于9 则为${10} 叫大括号
- $# 获取命令行参数的个数

进程状态变量

- $? 获取执行上一个指令的返回值（0为成功 非零为失败）

> 在装软件的时候，make后面使用这个来判断是否成功。
>
> $?返回值很多，可以适当收集一下。

0	**表示运行成功**
2	权限拒绝
1~125	表示运行失败，脚本命令，系统命令错误或参数传递错误
126	找到该命令了，但是无法执行
127	未找到要运行的命令
>128	命令被系统强制结束

- $$ 获取当前shell的进程号

$* 和 $@ 的区别

- $* 将所有的命令所有参数视为单个字符串，等同于"$1$2$3“
- $@ 将命令行每个参数视为单独的字符串，等同于"$1“"$2""$3"，这是将参数传递给其他程序的最佳方式，因为他会保留所有内嵌在每个参数里的任何空白

bash内部变量

shift

按如下方式重新命名所有的位置参数变量，$2变$1 后面的参数向前变。

shell变量子串

- ${#string} 返回字符串的长度
- ${string:position} 在$string中，从位置$position之后开始提取子串
- ${string:position:length} 提取长度为length的子串
- ${string#substring} 从开头开始删 最短匹配
- ${string##substring} 从开头开始删 最长匹配
- ${string%substring} 从结尾开始删 最短匹配
- ${string%%substring} 从开头开始删 最长匹配