Shell 读书笔记

首先，shell是一个命令解释器。它不仅仅是操作系统内核和用户之间的隔离层，它还是一种相当强大的编程语言。shell程序称为脚本，是一种易于使用的工具，可以通过“粘在一起”系统调用、工具、实用程序和编译二进制文件来构建应用程序。实际上，UNIX命令、实用程序和工具的全部曲目可供shell脚本调用。如果这还不够的话，内部shell命令，比如测试和循环结构，会给脚本带来更多的力量和灵活性。shell脚本特别适合于管理系统任务和其他常规重复任务，而不需要完全结构化的编程语言。

在Linux机器启动时，它会在/etc/rc.d中执行shell脚本 来恢复系统配置并设置服务

1.何时不使用shell脚本

资源密集型任务，特别是速度是一个因素（排序，哈希，递归 [2] ...）

涉及重型数学运算的程序，特别是浮点运算，任意精度计算或复数（使用 C ++或FORTRAN 代替）

需要跨平台的可移植性（使用 C或Java 代替）

复杂的应用程序，其中结构化编程是必要的（变量类型检查，函数原型等）

您投注公司未来的关键任务应用程序

安全性很重要的情况下，您需要保证系统的完整性，并防止入侵，破解和破坏行为

项目由具有互锁依赖关系的子组件组成

需要大量的文件操作（Bash仅限于串行文件访问，并且只能以特别笨拙和低效的逐行方式进行。）

需要对多维数组的本机支持

需要数据结构，如链接列表或树

需要生成/操作图形或GUI

需要直接访问系统硬件或外部外围设备

需要端口或插座 I / O

需要使用库或与传统代码的接口

专有的闭源应用程序（Shell脚本将源代码直接公开给全世界看。）

1. 基础知识

在案件结束选项[双分号]

“点”作为文件名的一部分。使用文件名时，前导点是 “隐藏”文件的前缀，即 ls通常不会显示的文件。

考虑目录名称时，单个点表示当前工作目录，两个点表示父目录

部分引用 [双引号]

全文引用 [单引号]

通配符。 这个？字符作为单字符“通配符” 用于文件名扩展的文件名替换，以及代表一个字符 在一个延长的正则表

空命令[冒号]。

设定

vim /etc/vimrc

:set nu|nonu ####行号的显示

:set mouse=a ####设定鼠标可以操作光标位置

:help ####查看vim的用法，:q退出帮助

ctrl + v ####高亮选择模式，可以选者一片区域操作

u ####恢复到未操作之前

ctrl + r ####恢复到操作之后

批量添加字符

ctrl + v 选中要加字符所在的位，按I键，在加入想要加入的字符，按esc

####分屏功能

ctrl+w s ####上下分屏

ctrl+w v ####左右分屏

ctrl+w c ####撤销光标所在屏幕

ctrl+w 上|下|左|右 #####移动光标到指定的屏幕中

######替换字符###########

在命令模式中

:%s/原字符/替换后字符/g

##########搜索关键字#########

/关键字

n向下匹配

N向上匹配

####################字符的管理

dw 删除光标所在单词

dl 删除光所在字母

dd 删除一整行

d数字d 删除数字指定的行数

d数字 上键 向上删除指定行数

yw 复制光标所在单词

yl 复制光所在字母

yy 复制一整行

p 粘贴

y数字y 复制数字指定的行数

cw 剪切光标所在单词

cl 剪切光所在字母

cc 剪切一整行

c数字c 剪切数字指定的行数

剪切完成后会进入插入模式，要粘贴先退出插入模式再按P键

################移动光标到指定行############

：行号

G ##移动光标到文件末行

2.vim的插入模式

i 光标坐在位置插入字符

I 光标所在行行首插入

a 光标所在字符的下一位插入

A 光标所在行行尾插入

o 光标所在行的下一行

O 光标所在行的上一行

s 删除光标所在字符插入

S 删除光标所在行插入

3.vim的退出模式

在命令模式下输入

:q ###退出，在未对文件作任何修改的时候可以用

:q! ###强制退出

:wq ###退出保存

:wq! ###强制退出保存

###############vimtutor##########

vimtutor是vim的说明工具

用：q退出此工具

一个未初始化的变量有一个 “null”值 - 根本没有赋值（不是零！）

从技术上讲，变量的 名称称为 左值，意味着它出现在赋值语句的左侧，如VARIABLE = 23。变量的值是一个右值，意味着它出现在 赋值语句的右侧，如 VAR2 = $ VARIABLE。

事实上，一个变量的 名字是一个引用，一个指向与该变量相关的实际数据所在的内存位置的 指针。

与许多其他编程语言不同，Bash不按“类型”分隔变量。基本上，Bash变量是字符串，但是，根据上下文，Bash允许对变量进行算术运算和比较。决定因素是变量的值是否只包含数字。

在更一般的情况下，每个过程都有一个 “环境”，即过程可能引用的一组变量。在这个意义上，shell的行为就像其他进程一样。

每次shell启动时，它都会创建与自己的环境变量相对应的shell变量。更新或添加新的环境变量会导致shell更新其环境，并且所有shell的 子进程（它所执行的命令）都将继承此环境。

$ 0是脚本本身的名称， $ 1是第一个参数， $ 2是第二个，$ 3 是第三个，等等。 [2] 后$ 9，该参数必须在括号中，例如，$ {10} ， $ {11} ，$ {12} 。

特殊变量$ \*和$ @ 表示所有的位置参数

与echo和 sed一起使用

\ n

表示换行符

\ r

意味着返回

\ t

手段标签

符\ v

意味着垂直标签

\ b

意味着退格

\一个

意味着警报（哔声或闪光）

\ 0XX

转换为 等同于0nn的八进制ASCII，其中 nn是一串数字

每个命令都会返回一个 退出状态 （有时称为 返回状态 或退出代码）。 一个成功的命令返回一个0，而一个不成功的命令返回一个 通常可以被解释为错误代码的非零值

$？读取最后执行的命令的退出状态。函数返回后， $？给出函数中执行的最后一个命令的退出状态。这是Bash给予函数“返回值” 的方式

算术表达式的 退出状态不是错误值。

当 条件测试中的if和then在同一行时，分号必须终止if语句。无论 是否与随后 的关键字。关键字（或命令）开始语句，并且在同一行上的新语句开始之前，旧语句必须终止。

-e

文件已存在

-一个

文件已存在

这与-e效果相同。它已被“弃用” [1] ，其使用被劝阻。

-F

文件是一个普通 文件（不是目录或设备文件）

-s

文件不是零大小

-d

文件是一个目录

-b

文件是块设备

-C

文件是一个字符设备

-p

文件是一个管道

-H

文件是一个符号链接

-L

文件是一个符号链接

-S

文件是一个套接字

-t

文件（描述符）与终端设备相关联

此测试选项可用于检查 给定脚本中的stdin [-t 0]或 stdout [-t 1]是否为终端。

-r

文件具有读取权限（对于运行测试的用户）

-w

文件具有写入权限（对于运行测试的用户）

-X

文件具有执行权限（对于运行测试的用户）

-G

在文件或目录中设置set-group-id（sgid）标志

如果某个目录 设置了sgid标志，则在该目录内创建的文件属于拥有该目录的组，而不一定是创建该文件的用户组。这对于工作组共享的目录可能很有用。

-u

在文件上设置set-user-id（suid）标志

即使在普通用户调用它的情况下， 具有set-user-id标志集的root拥有的二进制文件也会以root权限运行。 [2] 这对于需要访问系统硬件的可执行文件（例如pppd和cdrecord）非常有用 。缺少 suid标志，这些二进制文件不能被非root用户调用。

-rwsr-xr-t 1 root 178236 2000年10月2日/ usr / sbin / pppd

设置了suid标志的文件在其权限中显示s。

-k

粘性位设置

俗称粘滞位， 在保存文本模式的标志是一种特殊类型的文件权限。如果文件设置了该标志，该文件将被保存在高速缓冲存储器中，以便快速访问。 [3] 如果设置在目录中，则会限制写入权限。设置sticky位会 在文件或目录列表的权限上添加一个t。这限制了将该目录中的特定文件更改或删除到这些文件的所有者。

drwxrwxrwt 7 root 1024 5月19日21:26 tmp /

如果用户不拥有设置了粘滞位但在该目录中具有写入权限的目录，则她只能删除其拥有的文件。这可以防止用户无意中覆盖或删除可公开访问的目录中的对方文件，例如/ tmp。（当然，目录或 root的所有者可以在那里删除或重命名文件。）

-O

你是文件的所有者

-G

文件的group-id与你的相同

-N

文件自上次读取后被修改

f1 -nt f2

文件f1比f2更新

f1 -ot f2

文件f1比f2旧

f1 -ef f2

文件f1和 f2是到同一文件的硬链接

！

“不” - 反转上述测试的意义（如果条件不存在，则返回true）。

7.3。其他比较运算符

一个二进制比较运算符比较两个变量或数量。请注意，整数和字符串比较使用一组不同的运算符。

整数比较

当量

等于

如果[“$ a”-eq“$ b”]

-ne

不等于

如果[“$ a”-ne“$ b”]

-gt

大于

如果[“$ a”-gt“$ b”]

-ge

大于或等于

如果[“$ a”-ge“$ b”]

-lt

小于

如果[“$ a”-lt“$ b”]

-le

小于或等于

如果[“$ a”-le“$ b”]

<

小于（在双括号内）

（（“$ a”<“$ b”））

<=

小于或等于（在双括号内）

（（“$ a”<=“$ b”））

>

大于（在双括号内）

（（“$ a”>“$ b”））

> =

大于或等于（在双括号内）

（（“$ a”> =“$ b”））

字符串比较

=

等于

如果[“$ a”=“$ b”]

警告

请注意 构成=的空格。

如果[“$一个” =“$ B”]是 不等同于以上所述。

==

等于

if [“$ a”==“$ b”]

这是=的同义词。

注意

所述==比较运算符行为内的不同双括号 比单括号内测试。

[[$ a == z \*]]＃如果$ a以“z”（模式匹配）开头，则为真。

[[$ a ==“z \*”]]＃如果$ a等于z \*（文字匹配），则为真。

[$ a == z \*]＃发生文件通配和分词。

[“$ a”==“z \*”]＃如果$ a等于z \*（文字匹配），则为真。

＃谢谢，StéphaneChazelas

！=

不等于

如果[“$ a”！=“$ b”]

该运算符使用[[...]]结构内的模式匹配。

<

小于，按ASCII字母顺序

如果[[“$ a”<“$ b”]]

如果[“$ a”\ <“$ b”]

请注意，“<”需要 在[]构造内 转义。

>

大于，按ASCII字母顺序排列

如果[[“$ a”>“$ b”]]

如果[“$ a”\>“$ b”]

请注意，“>”需要在[]结构中转义。

有关此比较运算符的应用，请参见示例27-11。

-z

字符串为空，即长度为零

String =''＃零长度（“null”）字符串变量。

如果[-z“$ String”]

然后

回声“\ $字符串为空”。

其他

回声“\ $字符串不是空。”

fi＃$ String为空。

-n

字符串不为空。

警告

该-n测试要求的字符串测试括号内引用。使用带引号的字符串 ！-z，或者甚至仅仅是测试括号内没有引号的字符串,通常可行，但这是一种不安全的做法。总是引用测试过的字符串。

Bash不理解浮点运算。它将包含小数点的数字视为字符串。

shell脚本将数字解释为十进制数（以10为底），除非该数字具有特殊的前缀或符号。前面加上 0的数字是八进制数 （基数8）。前面加上0x的数字 是十六进制（基数为16）。

类似于let命令，（（...））结构允许算术扩展和评估。以最简单的形式，a = $（（5 + 3））将a设置 为5 + 3或 8。但是，这个双括号结构也是一种允许在Bash中对C变量进行操作的机制，例如 （（var ++））。

**运算符优先级**

| **操作者** | **含义** | **注释** |
| --- | --- | --- |
|  |  | 最高的优势 |
| var ++ var-- | 后增量，后减量 | [C风格的](http://tldp.org/LDP/abs/html/assortedtips.html" \l "CSTYLE)运营商 |
| ++ var --var | 预先递增，预先递减 |  |
|  |  |  |
| ！〜 | [否定](http://tldp.org/LDP/abs/html/special-chars.html" \l "NOTREF) | 逻辑/位，反转后面的操作符的意义 |
|  |  |  |
| \*\* | [幂](http://tldp.org/LDP/abs/html/ops.html" \l "EXPONENTIATIONREF) | [算术运算](http://tldp.org/LDP/abs/html/ops.html" \l "AROPS1) |
| \* /％ | 乘法，除法，模 | 算术运算 |
| + - | 加法，减法 | 算术运算 |
|  |  |  |
| << >> | 左转，右转 | [按位](http://tldp.org/LDP/abs/html/ops.html" \l "BITWSOPS1) |
|  |  |  |
| -z -n | 一元比较 | 字符串是/不是[空的](http://tldp.org/LDP/abs/html/comparison-ops.html" \l "STRINGNULL) |
| -e -f -t -x等 | 一元比较 | [文件测试](http://tldp.org/LDP/abs/html/fto.html) |
| <-lt> -gt <= -le> = -ge | 复合比较 | 字符串和整数 |
| -nt -ef -ef | 复合比较 | 文件测试 |
| == -eq [！=](http://tldp.org/LDP/abs/html/comparison-ops.html" \l "NOTEQUAL) -ne | 平等/不平等 | 测试运算符，字符串和整数 |
|  |  |  |
| ＆ | 和 | 按位 |
| ^ | XOR | 独占 OR，按位 |
| | | 要么 | 按位 |
|  |  |  |
| ＆＆ -一个 | 和 | [逻辑](http://tldp.org/LDP/abs/html/ops.html" \l "LOGOPS1)， 化合物 比较 |
| || -o | 要么 | 合乎逻辑的复合 比较 |
|  |  |  |
| ？ | [三元运算符](http://tldp.org/LDP/abs/html/special-chars.html" \l "CSTRINARY) | C-风格 |
| = | [分配](http://tldp.org/LDP/abs/html/varassignment.html" \l "EQREF) | （不要混淆平等 测试） |
| \* = / =％= + = - = << = >> =＆= | [组合分配](http://tldp.org/LDP/abs/html/ops.html" \l "ARITHOPSCOMB) | 时间相等，分等，模相等等。 |
|  |  |  |
| ， | [逗号](http://tldp.org/LDP/abs/html/ops.html" \l "COMMAOP) | 链接一系列操作 |
|  |  | 最低的优势 |

第3部分。超越基础

$ RANDOM是一个内部Bash 函数（不是常量），它返回 范围为0 - 32767 的伪随机 [1]整数。它 不应该用于生成加密密钥

串提取

$ {字符串：位置}

从提取子$字符串在 $位置。

如果$ string参数是 “ \* ” 或“ @ ”，那么这将提取 位置参数， [1] 从$ position开始。

$ {字符串：位置：长度}

从$位置的$ string中 提取子字符串的$ length字符

子字符串替换

$ {字符串/子/换货}

更换第一比赛的 $子与 $更换。 [2]

$ {//串子/换货}

替换的所有比赛 $子与 $更换。

**子字符串替换**

$ {字符串/子/换货}

更换第一比赛的 *$子*与 *$更换*。

$ {//串子/换货}

替换的所有比赛 *$子*与 *$更换*。

可变扩展/子串替换

这些构造已经从ksh中采用 。

$ {VAR：POS}

变量VAR扩大，从偏移开始POS。

$ {VAR：POS：LEN}

扩张的最大len个 变量的字符变种，从偏移 POS。有关 此运算符的创造性使用的示例，请参见示例A-13。

$ {VAR /模式/换货}

Pattern中的第一个匹配项，在var中替换为 Replacement。

如果替换被省略，则Pattern的第一个匹配 被替换为 无，即被删除。

$ {VAR //模式/换货}

全球更换。 Pattern中的 所有匹配项，在var中替换为 Replacement。

如上所述，如果替换 被省略，则所有出现的 模式被替换为 无，即被删除。

甲嵌套循环是一个循环内的循环，一个外的主体内的内循环。这是如何工作的，外部循环的第一遍触发内部循环，执行完成。然后外部循环的第二遍再次触发内部循环。这重复直到外部循环结束。当然，断裂或者内部或外部循环中会中断此过程

##########################用户操作命令################

id 查看用户信息

id -u ###用户uid

-g ###用户gid

-a ###用户所有信息

-G ###用户所在所有的id

-un ###用户名字

-gn ###用户初始组名称

-Gn ###用户所在所有组名称

su - username ###切换用户用户所使用的环境

echo $PATH ###查看当前用户的用户环境

echo $USER ###显示当前用户的信息

useradd ###新建用户

-u ###指定用户的uid

-g ###指定用户初始组，这个组一定要存在

-G ###指定附加组，组同样要存在

-c ###用户说明

-d ###指定用户加目录

-s ###指定用户使用的shell，shell种类参看/etc/shells

userdel -r ####删除用户

groupadd -g ###新建组，-g表示指定组的id

groupdel ###删除组

passwd ####更改用户密码

###删除用户

passwd ####更改用户密码

查看用户建立或管理过程使用的命令

watch -n 1 'tail -3 /etc/passwd;echo @@@@@@@@@@@@@@@@@@@@@@@@@@;tail -3 /etc/group'

usermod -u ##改变用户uid

-g ##改变用户初始组

-G ##改变用户附加组

-aG ##增加用户附加组

-d ##改变用户加目录信息

-md ##改变用户加目录信息和加目录名称

-s ##改变用户的shell

-c ##改变用户的说明

-L ##冻结用户

-U ##解锁用户

###################用户权限下放（sudo）###################

权限下放动作的配置文件

/etc/sudoers

这个文件用visudo命令编辑

文件内容写法：

授权目标用户 主机名称=（授权用户得到的新用户身份） 授权用户执行命令

示例：

###test用户能在localhost主机执行useradd（要test密码）

test localhost=(root) /usr/sbin/useradd

###test用户能在localhost主机执行useradd（不要要test密码）

test localhost=(root) NOPASSWD: /usr/sbin/useradd

###测试结果：

su - test

sudo useradd username

##########################密码期限更改###################

chage -d 0 ##秘密使用了多久，如何设定为0表示用户登陆系统前必须更改密码

-m ##最短有效期

-M ##最长有效期

-W ##警告期

-I ##非活跃期

-E ##到期时间格式yyyy-mm-dd

######################################################

1.文件属性查看

ls -l filename

- rw-rw-r-- 1 kiosk kiosk 0 Oct 2 17:05 file

文件类型 文件权限 文件副本个数 文件所有人 文件所有组 大小 最后一次被修改的时间 名字

2.查看目录属性

ls -ld directoy

d rwxr-xr-x 3 kiosk kiosk 33 Aug 27 09:47 directory

类型 权限 子目录个数 目录所有人 目录所有组 子文件属性大小 目录中内容变化 目录名字

3.文件用户组的更改

chown 用户名称 文件 ##更改文件所有人

chgrp 组名称 文件 ##更改文件所有组

chown -R 用户 目录 ##更改目录本身以及目录中的子文件的所有人

chgrp -R 组名 目录 ##更改目录本身以及目录中的子文件的所有组

1.文件属性查看

ls -l filename

- rw-rw-r-- 1 kiosk kiosk 0 Oct 2 17:05 file

文件类型 文件权限 文件副本个数 文件所有人 文件所有组 大小 最后一次被修改的时间 名字

2.查看目录属性

ls -ld directoy

d rwxr-xr-x 3 kiosk kiosk 33 Aug 27 09:47 directory

类型 权限 子目录个数 目录所有人 目录所有组 子文件属性大小 目录中内容变化 目录名字

3.文件用户组的更改

chown 用户名称 文件 ##更改文件所有人

chgrp 组名称 文件 ##更改文件所有组

chown -R 用户 目录 ##更改目录本身以及目录中的子文件的所有人

chgrp -R 组名 目录 ##更改目录本身以及目录中的子文件的所有组

4.权限的识别

ls -l 中2-10字符为文件权限

rwx r-x r-x

用户权限 组成员权限 其他用户权限

权限种类

r

r权限针对文件，表示可以查看文件内容

r权限针对目录，表示可以ls 查看目录中存在的文件名称

w

w权限针对文件，表示可以更改文件的内容

w权限针对目录，表示是否可以删除目录中的子文件或者子目录

x

x权限对于文件，表示是否可以开启文件当中记录的程序

x权限对于目录，表示是否可以进入目录中

chmod ugo+-=rwx ##改权限

u=rwx=7 |g=rwx=7|o=rwx=7

421

u=rw-=6 g=r--=4 o=r--=4 644

r=4

w=2

x=1

-=0

7=rwx，6=rw-，5=r-x，4=r--，3=-wx，2=-w-，1=--x，0=---

5.文件的默认权限

umask ##显示系统预留权限值

vim /etc/bashrc 71行是普通用户的更改，73是超级用户的更改

vim /etc/profile 60行是普通用户的更改，62是超级用户的更改

source /etc/bashrc ##刷新bash配置

source /etc/profile ##刷新系统配置

6.特殊权限

stickyid ###强制位

o+t ###之针对目录，当一个目录上有t权限时，这个目录中的文件只能被文件拥有者删除

t=1

chmod o+t directroy

chmod 1777 directory

sgid ###粘制位

g+s ##针对目录，在目录中创建的文件都自动归属到目录所在组，针对二进制文件，文件内记录的程序在执行时和执行者的组身份没有关系，而是以二进制文件的所有组的身份执行的

chmod g+s file|directory

chmod 2777 file|directory

suid ###冒险位

u+s ###针对文件，文件记录动作在执行时是一文件所有人身份执行的，与是谁发起的无关

chmod u+s file

chmod 4777 fileg+s

4.如何用命令查看进程

ps 进程查看工具

-a shell中运行的进程

-x 不在shell重运行的进程

-u 查看进程用户

-l 显示进程的详细信息

-f 完全信息full的缩写

-e 显示系统中的所有进程

ps -aux |grep 进程关键字

ps ax -o %cpu,%mem,user,pid,comm ##查看进程某项信息

ps aux --sort +|-%cpu|%mem... ##进程按指定方式排序

5.如何控制shell重进程的前后台调用

命令& ##进程在shell后台运行

ctrl +z ##把占用shell的进程打入后台停止

jobs ##查看被打入后台的进程信息

fg job号 ##把指定进程调回前台

bg job号 ##运行后台停止进程

ctrl +c ##关闭占用shell进程

6.如何用命令控制进程

1)信号

信号 1 ##reload进程

信号 2 ##ctrl +c 删除进程在内存中的信息

信号 3 ##内存中的鼠标信息清除

信号 9 ##强行关闭进程

信号 15 ##正常关闭进程

信号 18 ##开启停止的进程

信号 19 ##停止正在运行的进程

信号 20 ##ctrl +z

2)信号发起

kill 信号 pid

killall 信号 进程名称

pkill -u username ##结束某个用户的所有进程

7.用户登陆监控

w ##查看当前使用系统的用户

w -f ##查看当前使用系统的用户并显示使用地点

last ##查看登陆过系统的用户

lastb ##查看试图登陆但没有成功的用户

8.top命令

u username

k 信号

h

s

m

C

这些文件包含 作为用户shell运行的Bash提供的别名和环境变量，以及系统初始化后调用的所有Bash脚本。

/ etc / profile文件

全系统的默认设置，主要是设置环境（所有Bourne类型的shell，而不仅仅是Bash [1]）

在/ etc / bashrc中

全系统功能和Bash 别名

$ HOME /.bash\_profile

用户特定的Bash环境默认设置，可在每个用户的主目录（与/ etc / profile的本地对应目录）中找到

$ HOME /.bashrc

用户特定的Bash初始化文件，可在每个用户的主目录（与/ etc / bashrc的本地对应目录 ）中找到。只有交互式shell和用户脚本读取此文件。有关示例 .bashrc文件，请参阅 附录M.

注销文件

$ HOME /.bash\_logout

用户特定的指令文件，可在每个用户的主目录中找到。从登录（Bash）shell退出后，此文件中的命令将执行。

数据文件

/ etc / passwd文件

系统中所有用户帐户的列表，它们的标识，主目录，它们所属的组以及它们的默认外壳。请注意，用户密码不 存储在此文件中[2]， 而是以加密形式存储 在/ etc / shadow中。

系统配置文件

的/ etc / SYSCONFIG / hwconf

列出并描述附加的硬件设备。这些信息是文本形式的，可以被提取和解析。

/bin ##二进制可执行文件也就是系统命令

/sbin ##系统管理命令存放位置

/boot ##启动分区，负责系统启动

/dev ##设备管理文件

/etc ##大多数系统管理文件

/home ##普通用户的家目录

/lib ##32位系统库文件存放位置

/lib64 ##64位系统库文件存放位置

/media ##系统临时设备挂载点

/mnt ##系统临时设备挂载点

/run ##系统临时设备挂载点

/opt ##第三方软件安装位置

/proc ##系统信息

/root ##超级用户家目录

/srv，/var ##系统数据

/sys ##系统管理，主要是关于内核的

/tmp ##系统临时文件存放位置

/usr ##系统用户相关信息数据及用户自定义软件存放位置

fg，BG

该FG命令切换在后台到前台运行的作业。的 BG命令重新开始暂停的作业，并运行在后台。如果未指定作业编号，则fg或bg 命令将作用于当前正在运行的作业。

等待

挂起脚本执行，直到在后台运行的所有作业都已终止，或者直到作为选项指定的作业编号或进程标识终止。返回waited-for命令的退出状态。

您可以使用wait命令来防止脚本在后台作业完成执行前退出（这会创建一个可怕的 孤立进程）。

Jobs

列出在后台运行的作业，并提供作业编号。不如ps有用。

注意

混淆工作和 流程很容易 。某些内置函数，例如 kill，disown和 wait会接受作业号或进程号作为参数。该FG， BG和工作 命令只接受工号。

bash $ sleep 100＆

[1] 1384

bash $ jobs

[1] + Running sleep 100＆

“1”是作业编号（作业由当前shell维护）。“1384” 是PID或过程ID号（过程由系统维护）。杀死这个工作/进程，杀死％1 或杀死1384

Suspend

这与Control - Z具有类似的效果 ，但它会挂起shell（shell的父进程应该在适当的时候恢复它）

times

提供执行命令时所用系统时间的统计信息

kill通过发送一个适当的终止信号强制终止一个进程

builtin 调用builtin BUILTIN\_COMMAND将命令 *BUILTIN\_COMMAND*作为shell [内建运行](http://tldp.org/LDP/abs/html/internal.html" \l "BUILTINREF)，临时禁用具有相同名称的函数和外部系统命令。

4.如何在系统中查找文件

find 命令

find 路径 条件 条件值

find /mnt -name file

find /mnt -name "file\*"

find /etc -name "\*.conf"

-user ##用户

-group ##组

条件1 -o 条件2 ##或者

-type ##类型

-mmin +|-数字 ##时间

xargs的

用于将参数提供给命令的过滤器，以及用于汇编命令本身的工具。它将数据流分解成足够小的块，以便过滤和处理命令。把它看作是反引号的强大替代品。在命令替换失败并且参数错误太多的情况下，替换xargs通常是有效的。 [1] 通常，xargs从标准输入或管道读取 ，但也可以输出文件。

xargs的默认命令是 echo。这意味着输入到xargs的输入可能会有换行符和其他空白字符被删除

ls | xargs -p -l gzip gzip在当前目录中的每个文件，每次一个，在每次操作之前提示。

注意

请注意，xargs按顺序处理传递给它的参数，一次一个。

bash $ find / usr / bin | xargs文件

/ usr / bin：目录

/ usr / bin / foomatic-ppd-options：perl脚本文本可执行文件

。。。

小费

一个有趣的xargs 选项是-n NN，它将传递参数的数量限制为NN。

ls | xargs -n 8 echo在 8列中列出当前目录中的文件。

小费

另一个有用的选项是 -0，结合find -print0或grep -lZ。这允许处理包含空格或引号的参数。

找到/ -type f -print0 | xargs -0 grep -liwZ GUI | xargs -0 rm -f

grep -rliwZ GUI / | xargs -0 rm -f

上述任何一项都会删除任何包含“GUI”的文件。 （谢谢，SC）

要么：

cat / proc /“$ pid”/“$ OPTION”| xargs -0回声

＃格式输出：^^^^^^^^^^^^^^^

＃来自Han Holl对“get-commandline.sh”的修正

＃/脚本在“/ dev和/ proc”一章。

小费

xargs的-P选项 允许并行运行进程。这加快了使用多核CPU的机器的执行速度。

＃！/斌/庆典

ls \* gif | xargs -t -n1 -P2 gif2png

＃将当前目录中的所有gif图像转换为png。

＃选项：

＃=======

＃-t将命令打印到stderr。

＃-n1每个命令行最多1个参数。

＃-P2同时运行最多2个进程。

find 命令

find 路径 条件 条件值

find /mnt -name file

find /mnt -name "file\*"

find /etc -name "\*.conf"

-user ##用户

-group ##组

条件1 -o 条件2 ##或者

-type ##类型

-mmin +|-数字 ##时间

dd if=/dev/zero of=/mnt/file bs=1024 count=10

dd ##数据截取

if ##数据模板

of ##数据存放文件

bs ##数据快大小

count ##数据快个数

--size 数字|-数字|+数字 ###查找大小符合条件的文件

--perm 444 ###文件全权必须时rrr

--perm -444 ###文件每一位都要含有r权限

--perm /444 ###文件任意一位含有r权限

-maxdepth 数字 ###最深目录层

-mindepth 数字 ###最小目录层

-exec 命令 {} \; ###对查找出的结果做相应处理

########locate#########

locate +条件 ##在数据库中查找符合条件的文加

updatedb ##更新数据库

5.链接

#######硬链接#########

硬链接是文件的副本，为了节省设备节点号

ln 文件 链接

######软链接############

软链接时文件的快捷方式，为了节省设备存储快

ln -s 文件 链接

ls -i ##查看一个文件节点号

1.系统延迟任务发起命令

[root@aclhost mnt]# at 11:03

at> touch /mnt/file{1..10} ##要执行的命令

at> <EOT><按ctrl+D> ##任务发起

job 2 at Sat Nov 7 11:03:00 2015

[root@aclhost mnt]# at -l ##查看任务列表

2 Sat Nov 7 11:03:00 2015 a root

[root@aclhost mnt]# atrm 2 ##删除编号为2的人物

[root@aclhost mnt]# at now+1min ##任务延迟一分钟执行

3.at命令的使用控制

/etc/at.deny ##默认存在系统中，在此列表中出现的用户被禁止

test ##禁止test执行at

/etc/at.allow ##默认在系统中不存在，

##但是当此文件存在时，

###deny文件失效，

#并且所有普通用户能执行at的权力被禁止，只有在此文件中的用户可以执行at

student ##系统中的普通用户只有student可以使用at

4.计划定期任务

发起方式1

crontab

-e ##编辑定期任务

-l ##显示定期任务

-r ##删除定期任务

crontab格式

分钟 小时 天 月 周

分钟/数字 ##每隔几分钟，以此类推，小时，天，月，周都适用

发起方式2

vim /etc/cron.d/ps ##ps文件名称任意

分钟 小时 天 月 周 用户 命令

4.crontab使用限制

/etc/cron.deny ##默认存在系统中，在此列表中出现的用户被禁止

test ##禁止test执行crontab

/etc/cron.allow ##默认在系统中不存在，

##但是当此文件存在时，

###deny文件失效，

#并且所有普通用户能执行crontab的权力被禁止，只有在此文件中的用户可以执行crontab

student ##系统中的普通用户只有student可以使用crontab

5.系统临时文件

vim /usr/lib/tmpfiles.d/xxx.conf

类型 名称 权限 所有人 所有组 存在时间（s m d）

systemd-tmpfiles --create /usr/lib/tmpfiles.d/xxx.conf ##创建

systemd-tmpfiles --clean /usr/lib/tmpfiles.d/xxx.conf ##清理

1.diff比较两个文件的不同

diff -u file file.new >file.path ##生成补丁文件

yum install patch -y ##安装打补丁工具

patch -b file file.path ##给老软件打补丁，清且对老软件进行备份

2.grep ##过滤字符

-i ##或略大小写

-n ##显示结果所在行的行号

-c ##结果个数

-v ##反向过滤

-r 关键字 目录 ##在目录中过滤还有关键字的文件

-E "关键字1|关键字2|....." ##过滤多个关键字

3.sort ##排序

-n ##纯数字

-r ##倒序

-u ##去掉重复行

-t ##指定分隔符

-k ##指定排序的列

4.uniq ##处理重复行

-c ##统计重复行的个数

-u ##显示唯一的行

-d ##显示重复行

5.tr ##转换字符大小写

tr 'a-z' 'A-Z' <file ##把file中所有小写字符转换成大写

6.sed ##控制流输出

sed 's/nologin/westos/g' passwd ##替换输出中的nologin为westos

sed '1,5s/nologin/westos/g' passwd ##替换输出中1-5行的nologin为wesots

sed '/games/,/nobody/s/nologin/westos/g' passwd ##替换输出中的games字符到nobody字符之间的nologin为westos

sed -e 's/nologin/westos/g' -e 's/sbin/lee/g' passwd ##多条替换策略用-e连接

sed 3d passwd ##不显示文件中的第三行

sed 3,7d file ##不显示文件中的3-7行

sed 3,7p file ##重复显示文件中的3-7行

sed -n 3,7p file ##只显示文件中的3-7行

[root@westos mnt]# cat rule ##规则文件内容

s/nologin/westos/g

s/sbin/lee/g

[root@westos mnt]# sed -f rule file ##使用规则文件更改输出

1 root:x:0:0:root:/root:/bin/bash

2 bin:x:1:1:bin:/bin:/lee/westos

3 daemon:x:2:2:daemon:/lee:/lee/westos

4 adm:x:3:4:adm:/var/adm:/lee/westos

打包表示把一堆文件变成一个

tar ####打包工具

-f ####指定生成包的名字

-c ####创建包

-v ####显示创建过程

-t ####查看包中内容

-x ####解包

-r ####添加文件到包中

--delete filename ##删除包中指定文件

--get filename ##取出包中指定文件

#################压缩##################

gzip

gzip xxxx.tar =====> xxxx.tar.gz ##压缩

gunzip xxxx.tar.gz =====> xxxx.tar ##解压

tar zcvf xxxx.tar.gz 目标文件 ##打包压缩文件

tar zxvf xxxx.tar.gz ###xxxx.tar.gz===>xxxx

bz2

bzip2 xxxx.tar =====> xxxx.tar.bz2 ##压缩

bunzip2 xxxx.tar.bz2 =====> xxxx.tar ##解压

tar jcvf xxxx.tar.bz2 目标文件 ##打包压缩文件

tar jxvf xxxx.tar.bz2 ###xxxx.tar.bz2===>xxxx

xz

xz xxxx.tar =====> xxxx.tar.xz ##压缩

unxz xxxx.tar.xz =====> xxxx.tar ##解压

tar Jcvf xxxx.tar.xz 目标文件 ##打包压缩文件

tar Jxvf xxxx.tar.xz ###xxxx.tar.xz===>xxxx

zip

zip -r xxx.tar.zip xxx.tar ###压缩

unzip xxx.tar.zip ###解压

######################远程文件传输#####################

scp ##远程复制

scp file user@ip:/directory ##上传文件

scp -r dir user@ip:/directory ##长传目录

scp user@ip:/filename /direcotry ##下载文件

scp -r user@ip:/directory /direcotry ##下载目录

rsync ###远程同步

rsync file|direcotry user@ip:/directory

rsync user@ip:/directory /directory

-r ##同步目录

-l ##不忽略链接

-p ##不忽略权限

-t ##不忽略时间戳

-g ##不忽略组信息

-o ##不忽略用户信息

-D ##不忽略设备文件

Sftp

rpm ###底层软件管理工具

-i ###install

-v ###显示信息

-h ###指定包的解密方式为hash

-q ###查询

-p ###指定软件包本身

-l ###列出软件中文件的名字

--scripts ###查看软件安装和卸载过程中所执行的动作

-f ###查询某个文件属于那个安装包

-a ###所有

-e ###卸载

--force ###强制

--nodeps ###忽略依赖性

第5部分。高级主题

正则表达式

正则表达式包含以下一项或多项：

一个字符集。这些是保留其字面含义的字符。最简单的正则表达式 只包含一个字符集，没有元字符。

锚点。这些指定（锚定）RE要匹配的文本行中的位置。例如，^和$是锚点。

修饰符。这些扩展或缩小（修改）RE要匹配的文本范围。修饰符包括星号，括号和反斜杠。

正则表达式（RE）的主要用途是文本搜索和字符串操作。RE 匹配单个字符或一组字符 - 字符串或字符串的一部分。

星号 - \* - 匹配字符串或RE之前的任意数量的重复，包括零实例。

“1133 \*”匹配11 +一个或者多个3： 113，1133， 1133333，等等。

该点 - 。 - 匹配任何一个字符，除了换行符。 [2]

“13.” 匹配的任何字符（包括空间）13 +的至少一种：1133， 11333，但不是 13（附加的字符丢失）。

有关点单字符 匹配的演示，请参见示例16-18。

脱字符 - ^ - 匹配一行的开头，但有时根据上下文否定RE中一组字符的含义。

RE结尾的美元符号 - $ - 匹配一行的结尾。

“XXX $”匹配XXX在一行的末尾。

“^ $”匹配空白行。

括号 - [...] - 附上一组字符以在单个RE中匹配。

“[xyz]”匹配字符x，y或z中的任何一个 。

“[cn]”匹配c 到n范围内的任何一个字符。

“[B-Pk-y]”与范围B 到P中的任何一个字符匹配 ，k到 y。

“[a-z0-9]”匹配任何单个小写字母或任何数字。

“[^ bd]”匹配除b到 d范围以外的任何字符 。这是^否定或反转后面RE的意义（在不同的上下文中扮演类似于！的角色 ）的实例 。

组合的括号字符序列匹配常见的单词模式。“[YY] [EE] [SS]”匹配 是的，是的， YES，YES，等等。 “[0-9] [0-9] [0-9] - [0-9] [0-9] - [0-9] [0-9] [0-9] [0-9]” 匹配任何社会安全号码。

反斜线 - \ - 转义特殊字符，这意味着字符被逐字解释（因此不再特殊）。

一个“\ $”恢复到它的字面含义“$”，而不是它的RE结尾的含义。同样，“\\” 的字面意思是“\”。

转义 “尖括号” - \ <... \> - 标记单词边界。

尖括号必须转义，否则它们只有它们的文字字符含义。

“\ <the \>”匹配单词“the”，但不匹配 “them”， “there”， “other”等。

这是指定要匹配的字符范围的替代方法。

[：alnum：]匹配字母或数字字符。这相当于 A-Za-z0-9。

[：alpha：]匹配字母字符。这相当于 A-Za-z。

[：blank：]匹配空格或制表符。

[：cntrl：]匹配控制字符。

[：digit：]匹配（十进制）数字。这相当于 0-9。

[：graph：]（图形可打印的字符）。匹配 ASCII 33 - 126范围内的字符。这与下面的 [：print：]相同，但不包括空格字符。

[：lower：]匹配小写字母字符。这相当于 az。

[：print：]（可打印字符）。匹配ASCII 32 - 126范围内的字符。这与上面的 [：graph：]相同，但添加了空格字符。

[：space：] 匹配空格字符（空格和水平制表符）。

[：upper：]匹配大写字母字符。这相当于 AZ。

[：xdigit：]匹配十六进制数字。这相当于 0-9A-Fa-f。

一个exec <filename命令将stdin重定向 到一个文件。从那时起，所有的 stdin都来自该文件，而不是它的正常源代码（通常是键盘输入）。这提供了一种逐行读取文件的方法，并可能使用sed 和/或awk解析每行输入

同样，exec> filename 命令将stdout重定向到指定的文件。这会将通常会转到stdout的所有命令输出发送到该文件。

代码块，例如while，until和for循环，即使/然后测试块也可以包含stdin的重定向。即使是一个函数也可以使用这种形式的重定向。代码块末尾的<运算符完成此操作。

在限制模式下运行脚本或脚本的一部分 会禁用某些可用的命令。这是一种安全措施，旨在限制脚本用户的权限并尽量减少运行脚本可能造成的损害。

以下命令和操作被禁用：

使用cd更改工作目录。

更改 $ PATH， $ SHELL， $ BASH\_ENV或$ ENV 环境变量的值。

阅读或更改$ SHELLOPTS，shell环境选项。

输出重定向。

调用包含一个或多个/ s的命令 。

调用exec来替换shell的不同进程。

各种其他的命令，可以使一个意想不到的目的变成或企图颠覆脚本。

在脚本中退出受限模式。

像“真实”的编程语言一样，Bash具有功能，尽管其实施有限。一个函数是一个子程序，一个代码块，实现一组操作，一个执行指定任务的“黑匣子”。无论哪里有重复的代码，当一个任务重复只有程序的轻微变化时，请考虑使用一个函数。

较新版本的Bash支持一维数组。 数组元素可以用变量[xx]表示法进行初始化 。或者，脚本可以通过显式declare -a变量语句引入整个数组 。要取消引用（检索数组元素的内容），请使用 大括号表示法，即 $ {element [xx]}。

所述的/ dev目录包含的条目的物理设备可以或可以不存在于硬件中。 [1] 恰当地，这些被称为设备文件。例如，包含已安装文件系统的硬盘驱动器分区在/ dev中具有条

在/ proc中的目录实际上是一个伪文件系统。/ proc镜像中的文件当前正在运行系统和内核进程， 并包含有关它们的信息和统计信息

把/ dev / null想象成一个黑洞。它本质上相当于一个只写文件。写入它的一切都消失了。尝试读取或输出它不会导致任何结果。同样，/ dev / null在命令行和脚本中都是非常有用的

像/ dev / null一样， / dev / zero是一个伪设备文件，但它实际上产生了一个空值流（二进制零，而不是ASCII类型）。写入/ dev / zero的输出消失，实际读取发射的空值相当困难，尽管可以使用od 或十六进制编辑器来完成。/ dev / zero的主要用途 是创建一个预定义长度的初始化虚拟文件，用作临时交换文件

以下是一些有用选项的列表。它们可以以缩写形式（前面是单个破折号）或完整名称（前面是双 破折号或-o）来指定。

**表33-1。Bash选项**

| **缩写** | **名称** | **影响** |
| --- | --- | --- |
| -B | 大括号扩展 | 启用 [大括号扩展](http://tldp.org/LDP/abs/html/special-chars.html" \l "BRACEEXPREF)（默认设置= 打开） |
| + B | 大括号扩展 | 禁用 大括号扩展 |
| -C | noclobber选项 | 防止通过重定向覆盖文件（可能会被> |覆盖） |
| -D | （没有） | 列出以$为前缀的双引号字符串，但不要在脚本中执行命令 |
| -一个 | allexport | 导出所有定义的变量 |
| -b | 通知 | 通知在后台运行的作业何时终止（在脚本中没有太多用处） |
| -C ... | （没有） | 从...读取命令 |
| checkjobs |  | 在shell退出时通知用户任何打开的[作业](http://tldp.org/LDP/abs/html/x9644.html" \l "JOBSREF)。在 Bash的[第4版中](http://tldp.org/LDP/abs/html/bashver4.html" \l "BASH4REF)引入，仍然是“实验”。 用法： shopt -s checkjobs（注意：可能会挂起！） |
| -e | errexit | 当第一个错误中止脚本时，命令以非零状态退出（除非在[until](http://tldp.org/LDP/abs/html/loops1.html" \l "UNTILLOOPREF)或[while循环中](http://tldp.org/LDP/abs/html/loops1.html" \l "WHILELOOPREF)，[if-tests](http://tldp.org/LDP/abs/html/testconstructs.html" \l "TESTCONSTRUCTS1)，[list构造](http://tldp.org/LDP/abs/html/list-cons.html" \l "LCONS1)） |
| -F | noglob | 文件名扩展（通配符）已禁用 |
| globstar | [globbing star-match](http://tldp.org/LDP/abs/html/bashver4.html" \l "GLOBSTARREF) | 启用\*\* [globbing](http://tldp.org/LDP/abs/html/globbingref.html)操作符（Bash的[4+版本](http://tldp.org/LDP/abs/html/bashver4.html" \l "BASH4REF)）。 用法： shopt -s globstar |
| -一世 | 互动 | 脚本以交互模式运行 |
| -n | NOEXEC | 在脚本中读取命令，但不要执行它们（语法检查） |
| -o选项 - 名称 | （没有） | 调用选项名称 选项 |
| -o posix | POSIX | 更改Bash或调用的脚本的行为以符合[POSIX](http://tldp.org/LDP/abs/html/sha-bang.html" \l "POSIX2REF) 标准。 |
| -o pipefail | 管道故障 | 导致管道返回管道中返回非零返回值的最后一个命令的[退出状态](http://tldp.org/LDP/abs/html/exit-status.html" \l "EXITSTATUSREF)。 |
| -p | 特权 | 脚本以“suid”运行（小心！） |
| -r | 限制 | 脚本以受限 模式运行（参见[第22章](http://tldp.org/LDP/abs/html/restricted-sh.html)）。 |
| -s | 标准输入 | 从标准输入读取命令 |
| -t | （没有） | 第一条命令后退出 |
| -u | nounset | 尝试使用未定义的变量输出错误消息，并强制退出 |
| -v | 详细 | 在执行之前将每个命令打印到stdout |
| -X | X跟踪 | 与-v类似，但扩展了命令 |
| - | （没有） | 选项标志结束。所有其他参数都是[位置参数](http://tldp.org/LDP/abs/html/internalvariables.html" \l "POSPARAMREF)。 |
| - | （没有） | 取消设置位置参数。如果给出的参数（*- arg1 arg2*），位置参数设置为参数。 |

通过对shell这本书的阅读，其他很多基础命令以及语法是以前用过的，但是由于练习的少，只知道用但是并不熟悉。还有就是这本里面

有很多的思想，技巧以及细节的东西是以前没有用到或者注意到的，

经过阅读学习，让我更清楚的了解到了以前的不足，让我知道了，怎么去更好去学习写shell编程，这些可能通过几天学习是达不到的，但是有了这样的方向，以及知道怎么不足，以后学习练习的当中，我会更加注重这方面的培养。

非常感谢分享！