shell编程

笔记本： Linux
创建时间： 2015/12/15 16:56
标签： Linux

shell编程
2015年12月15日
16:56

一 命令行基础

1，常用命令回顾

Useraddd/adduser [option] [username]：添加用户账户，用户账户信息和密码信息在/etc/passwd和/etc/shadow中

Usermod [option] [username]：修改用户账户，不同选项代表不同含义，usermod修改密码会变成明文，不安全。**Chpasswd** 可以从文件读取userid:passwd对批量修改密码。

Userdel –r [username]：删除账号，-r同时删除用户目录

Passwd [option] [username]：账户密码管理，修改/锁定密码等选项可选择

Groupadd [option] [groupname]：创建新用户组，组信息在/etc/group中

Groupmod [option] [groupname]：修改组名和组号，其他无法修改

Groupdel [groupname]：删除用户组，组内有用户时要先删除用户

Cp/mv [option] [source] [destination]：复制/移动文件或目录

Chown [option] [owner] [filename]：更改文件的所有者

SUID和SGID是可执行权限的升级，SUID拥有文件所有者的一切权力，SIGD拥有文件所在组的权利，可以使用权力内的一切权限和资源。使用时使用chmod u/g+s file。

Find [path] [option] [操作]：查找文件，操作为print，exec和ok打印或执行其他命令，{}表示查找的内容最为exec和ok执行命令的参数

Grep [option] [匹配模式] [文件]：在文件中查找内容，匹配模式用到正则表达式。和grep有关的还有egrep和fgrep命令。

Stat 文件名：查看某个文件所有状态信息。**File** 文件名：查看文件类型。

Ps [option] ：查看系统进程信息; **top** [option]。实时显示进程的各种信息。

Df -h ：查看所有已挂载磁盘的使用情况。**Du** -h：显示当前目录下文件，目录的磁盘使用情况。

Sort [option] 文件名：排序，默认按照字符排序，可以使用选项指定按照什么排序。

Fdisk /dev/设备名：对存储设备进行分区操作。主要使用其中的p，n，w等操作。

Lsof -a -p **$$** -d 0,1,2：列出当前进程打开的文件描述符。

Tee filename：T型管道，可以将输出同时发送给标准输出和指定的文件中。

Mail：查看本人用户的e-mail信息

Vmstat ：显示自上次重启以来的平均负载值。**Uptime**：显示系统统计信息。

2，正则表达式

符号	含义
*	匹配任意多个字符
.	匹配以个字符
^	行首

$	行尾
[]	字符集合，加上^为反
\< \>	精确匹配某个字符串
A\{n\}	前面字符出现n次，至少n次为\{n,\}，n-m次为\{n,m\}
？	前面字符出现0或1次
+	前面字符出现1或多次
（ ）	一个字符集或用在语句中
|	或

3，sed文本处理

Sed有三种启动方式：

1. 在shell中：**sed [选项] ‘sed命令’ 输入文件**
2. 将sed命令写入脚本中，再调用脚本：**sed [选项] –f sed脚本 输入文件**
3. 将sed命令写入脚本，直接执行脚本：**./sed脚本 输入文件**

下面表列出sed的各种参数和含义：

Sed命令选项							
选项	含义		选项		含义		
-n	不打印所有未编辑的行		-e		后跟sed编辑命令，用于多个编辑命令存在时		
-f	在sed脚本中使用，否则出错						
Sed定位文本方法							
选项		含义		选项			含义
X		指定行好		X,y			从x到y的行号范围
/模式/		查询包含匹配模式的行		/模/模/			查询包含两个模式的行
/模/,x		从匹配行到x行的范围		X,/模/			从x到匹配的范围
X,y!		X到y之外的范围					
Sed编辑命令							
选项		含义		选项		含义	

P		打印匹配行		=		打印文件行号	
a\		定位行号后追加文本		i\		定位行号前插入文本	
D		删除定位行		c\		用新文本替换定位文本	
S		S/被替换字符串/新字符串/选项		R		从另一个文件读文本	
W		将文本写入到一个文件		Y		y/被变换字符/新等长字符	
Q		第一个模式匹配完成后退出		L		显示控制字符	
{}		在定位行执行的命令组		N		读取下一个输入行，用下一个命令处理新行，常与{}合用	
H		将模式缓冲区文本追加到保持缓冲区		h		将模式缓冲区文本复制到保持缓冲区	
X		互换模式缓冲区和保持缓冲内容		g		保持缓冲区复制到模式缓冲区	
G		保持缓冲区追加到模式缓冲区					

4，awk脚本编程（参考C语言）

awk的三种启动方式：

(1) shell中：awk [-F 域分隔符] ‘awk程序段 ‘ 输入文件

(2) 写入脚本调用：awk -f awk脚本 输入文件

(3) 直接执行脚本：./awk脚本 输入文件

4.1 awk模式匹配

awk将每一行看作记录，行中每个字符串看作域，$+索引可以指定该索引处的字符串，如此命令：{print $2,$1,$4,$3}，$0代表全部域。

awk各类运算符和含义					
选项		含义	选项	含义	
<,>,<=,>=,==,!=		关系运算符	~，！~	匹配正则表达式(不)	
||,&&,!		逻辑运算符	+,-,*,/,%,^或**	算术运算符	
awk中的系统变量和含义					
变量	含义		变量	含义	
ARGC	命令行参数数量		ARGIND	当前文件的位置	

ARGV	命令行参数数组		CONVFMT	数字转换格式	
ENVIRON	环境变量关联数组		ERRNO	最后一个错误描述	
FIELDWIDTHS	字段宽度列表		FILENAME	当前文件名	
FNR	浏览文件记录数		FS	域分隔符	
IGNORECASE	真忽略大小写		NF	当前行的域数	
NR	当前行数		OFMT	数字输出格式	
OFS	输出域分隔符		ORS	输出记录分隔符	
RLENGTH	match匹配长度		RS	行分隔符	
RSTART	match匹配位置		SUBSEP	数组下标分隔符	

4.2 字符串函数

Awk中print和printf和C语言中一样，字符串函数不一样，下面为主要字符串函数：

函数名	含义
Gsub(r,s)	用s替换t
Gsub(r,s,t)	在域t中用s替换r
Index(s,t)	返回域s中字符串第一个t的位置
Length(s)	返回s的长度
Match(s,t)	测试s中是否包括匹配他的字符串
Split(r,s,t)	在域t上将r分成序列s
Sub(r,s,t)	将t中第一次出现的r替换成s
Substr(r,s)	返回字符串r中从s开始的后缀部分
Substr(r,s,t)	返回字符串r从s开始长度为t的后缀部分

向脚本传递参数：awk脚本 参数=值 输入 文件，但是参数不能在BEGIN中使用。

4.3 awk关联数组

Awk数组不定义数组类型和大小，可以直接使用，数组的索引是字符串，即9和09不是同一个索引。

For(index in array) {操作数组元素}----------for循环访问关联数组

index in array也可以用于条件表达式判断元素是否在数组中

环境变量的索引为环境变量名，值为环境变量的值。

5，环境变量设置

全局变量不仅对shell会话，对所有shell创建的子进程也可见；局部变量只对创建它们的shell可见。**Printenv** ：查看所有全局变量。**Echo $变量名**：查看某个变量的值。**Set**：查看所有的变量，包括全局变量和当前进程的局部变量。

局部变量设置： **变量名=值**。注：值中有空格时需要用单引号界定；变量名，等号和值之间没有空格。

全局变量设置：先创建局部变量，然后导出到全局变量 **export 局部变量名**。注：导出时不能用$符号。

删除环境变量：unset 变量名。注：处理全局变量时，如果是在子进程中删除全局变量，在父进程中仍然有效。

环境变量还可以作为数字使用，如下面的示例：

test=(one two three foor)------创建可变数组

echo ${test[2]} / echo ${test[*]}----显示某个索引处值/显示所有元素

Alias -p：查看命令别名列表。**Alias 命令名=别名**：创建命令别名，但是只在该shell中有效。要创建全局的别名要在.bashrc中配置别名。

6，linux软件安装

6.1 基于Debian的系统

在debian系linux中dpkg是一个软件包管理工具，aptitude则是一个完整的软件包管理系统。直接输入**aptitude**可以进入管理系统的命令行界面，可以查看系统软件情况。

Aptitude show 包名：查看某个软件包的详细信息。

Aptitude search 包名：在软件源查找软件包，若显示则表示已经安装。

Aptitude install 包名：安装软件包。

Aptitude safe-upgrade：将所以软件包更新到最新版本。没有包参数

Aptitude purge 包名：卸载软件并删除相关数据和配置文件，只卸载软件可用remove。

Dpkg -L 包名：查看跟特定软件包关联的所有文件列表。

Dpkg --search 绝对路径文件名：查看某个文件属于哪个软件包。

Aptitude的默认软件源存放在/etc/apt/source.list文件中，可在该文件中添加软件源。

6.2 基于Redhat的系统

在fedora等新的Redhat系linux中yum管理工具成为流行，不过还是基于rpm的。

Yum list installed：列出系统已经安装的软件包。

Yum list 包名：查看特定包的详尽信息。

Yum privides 文件名：查看某文件属于哪个软件包。

Yum install 包名：安装软件包。

Yum localinstall 包名.rpm：安装本地的rpm包。

Yum list updates：列出已安装软件包的可用更新。

Yum update 包名：更新某个软件包。**Yum update**：更新所有软件包。

Yum erase 包名：卸载软件并删除相关数据和配置文件，只卸载软件可用remove。

yum的默认软件源存放在/etc/yum.repos.d文件中，可在该文件中添加软件源。

二 shell脚本编程

1，Shell脚本基础

把一系列的Linux命名放在以个文件中：下面为shell的脚本的格式:

```
#！/bin/bash        shell脚本的起始符号，指定脚本解释器

//Linux命令集

Cd   #shell注释

..bash_profile

Date

who
```

Shell脚本中反引号``允许将shell命令的输出赋值给变量。Linux下出来输入和输出重定向还有内联输入重定向，下面是其示例样式：

wc << EOF

> test string 1

> test string 2

> test string 3

> EOF

1.1 数学计算

Bash中可以使用如下方式将数学运算包括起来：**$[数学表达式]或$((数学表达式))**。Bash默认只支持整数运算，要进行浮点数运算需要使用bc（bash内建的计算器）。下面是使用bc的两

变量=`echo “bc选项；数学表达式“ | bc`。第二种方式是使用内联重定向，格式见示例：

```
#!/bin/bash
#shell中的数学计算和浮点数运算解决方案
var1=100
var2=50
var3=45
var4=$[$var1*($var2-$var3)] #常规整数运算
echo "整数运算的输出结果：$var4"
var5=`echo "scale=4;$var1/$var3" |bc` #echo 格式的浮点数运算
echo "浮点数echo解决方案的输出结果：$var5"
var6=`bc << EOF #内联重定向的浮点数运算
scale=4
$var1/$var3
EOF
`
echo "浮点数内联重定向解决方案的输出结果：$var6"
```

1.2 条件分支命令

Shell中**if-else命令**和python等脚本的类似，但是if或elif的条件是命令执行的状态而不是一个等式的真假，这是要注意的地方。下面是if条件分支的示例：

```
#!/bin/bash
#if-then-else条件分支示例
testuser=badtest
if grep $testuser /etc/passwd;then
echo "该用户主目录下的文件有："
ls -a /home/$testuser/.b*
#若还有条件可以用elif
else
echo "用户$testuser在系统中不存在"
fi
```

```
#!/bin/bash
#case格式的判断，简化了if-else分支
case $USER in
tigerren)
echo "welcome ,$USER";; #双分号结尾
testing)
echo "Special testing account";;
*) #默认值
echo "sorry,you are not allowed here";;
esac
```

if只能用命令执行结果作为判断依据，如果要进行数的比较怎么办？这时应该使用test命令。**Test命令**可以进行数字，字符串，文件等多种比较，为真则返回状态码0,从而进行if操作。在if语句中可以用 **test 条件语句** 或 **[条件语句]**来使用test比较。

Test 数值比较	
N1 -eq n2	N1,n2是否相等
N1 -ge n2	N1是否大于等于n2
N1 -gt n2	N1是否大于n2
N1 -le n2	N1是否小于等于n2
N1 -lt n2	N1是否小于n2
N1 -ne n2	N1是否不等于n2
Test字符串比较	
Str1 = str2	是否相等（！=）
Str1 < str2	Str1是否比str2小（>）,注意需要\进行转义操作
-n str1	Str1的长度是否非0
-z str1	Str1长度是否为0
Test 文件比较	
-d file	File是否存在且是一个目录
-e file	File是否存在
-f file	File是否存在且是一个文件
-r file	File是否存在可读（-w，-x）
-s file	File是否存在且非空
-O file	File是否存在并属于当前用户所有
-G file	File是否存在并默认组和当前用户相同
File1 -nt file2	File1是否比file2新
File1 -ot file2	File1是否比file2旧

除了使用test表示条件还有两种bash新增的扩展：

((表达式))：数学比较，可以使用标准的常见比较字符和运算，并且不需要进行转义

[[表达式]]：字符串比较，使用test命令的比较，并且还可以只用正则表达式。

1.3 循环命令

第一种循环是for循环，它有一个in列表，循环会依次取列表中的值进行操作。In列表可以直接添加，以空格为分割（可以IFS=$’ ’ 指定别的分隔符），也可以从变量读取或者从命令读取（需要、）。

```
for var in Alabama Alaska Arizona Arkansas California Colorado "new York"； do
echo the next state is $var
Done
```

```
#C语言风格的for循环
for (( i=1;i<=10;i++));do
echo "the next number is $i"
done
```

While循环是另一种常用的循环方式，除了使用test进行条件判断，还可以指定多个命令，最后一个命令的状态决定循环是否继续：

```
while [ $var1 -gt 0 ];do
echo $var1
var1=$[ $var1-1 ]
done
```

```
#while可以有多个命令，一个命令一行，最后一个测试命令状态决定循环是否继续
var2=10
while echo $var2
[ $var2 -gt 0 ];do
echo this is inside the loop
var2=$[$var2-1]
done
```

Until命令和while格式相同，工作方式相反，until要求指定一个状态非0的命令或比较，只有条件非0才会执行循环内容。

在done之后加一个输出重定向>可以将循环的输出重定向到一个文件中，也可以用管道|作为参数传递给另外一个shell命令。

1.4 处理用户输入

第一种用户输入是使用**位置参数**，位置参数变量是标准的数字，**$0**表示程序名，**$#**表示参数的个数，**$1-$9**为执行脚本命令时输入的参数，数字大于9时需要加{}，如${10}。位置参数可以加入任意多个命令行参数。另外，$*和$@会存储所有的数据，不同的是**$***将数据作为单个字符串保存，**$@**作为多个独立的单词保存可以进行循环遍历。

```
#!/bin/bash
#位置参数，允许用户输入，在运行脚本同时指定参数
total=$[ $1*$2 ]
echo "第一个参数为：$1"
echo "第二个参数为：$2"
echo "两者的乘积为：$total"
name=`basename $0`  #basename命令可以返回程序名而去掉路径
echo "程序名为：$name"
```

Shift命令默认会将每个参数减一，即$2变为$1，$1删除，可以用作参数遍历。Shift还可以指定移动的位数，默认为1,可以改变为其他值：**shift 2**。

当脚本同时有option和参数，并且选项又带有参数，这时候分离选项和参数就很重要。在shell中getopts可以将这一过程简单化。Getopts的格式为：**getopts optstring vatiable**。Optstring为有效的选项字母，若选项带有参数则在字母后加冒号；若要去掉错误信息则在optstring前加冒号。**$OPTARG**全局变量保存选项参数值；**$OPTIND**全局变量保存参数列表中getopts正在处理的参数的位置。下面是示例：

```
while getopts :ab:c opt;do
case "$opt" in
a) echo "执行a选项操作";;
b) echo "执行b选项操作，并有参数$OPTARG";;#获取选项参数
c) echo "执行c选项操作";;  #有多个选项参数同样访问$OPTARG，会自动区分
```

```
*) echo "未知的选项：$opt";;
esac
done
shift $[ $OPTIND -1 ]  #shift和$OPTIND结合来移动参数
count=1
for param in $@;do
echo "位置参数 $count:$param"
count=$[$count+1]
done
```

除了在执行脚本是指定参数和选项，同样可以在脚本执行过程中获取用户的输入。**Read 变量**命令可以从键盘或文件接受输入，并存入一个标准变量。下面是read命令在脚本中常见用法：

```
read -p "输入你的名字：" name  #-p为打印输入提示
read -p "输入你的英文名：" first last  #接受多个输入
read -t 5 -p "输入一个名字：" name2  #-t 指定超时时间（秒）
read -n 1 -p "继续吗？[Y/N]" answer  #-n指定接受的字符个数
read -s -p "输入你的密码：" passwd  #隐藏输入内容
#read从文件读入
count=1
cat readfile | while read line;do
echo "行数$count:$line"
count=$[ $count+1]
done
```

1.5 数据呈现

在脚本中重定向输出有两种：临时重定向和永久重定向。

临时重定向：Echo “this is an error message” >&2 #直接将要重定向的内容定向到对应文件描述数字
永久重定向：**exec** 1>file #将标准输出重定向到文件，会对所有标准输出起作用 Exec 0<file #还可以重定向输入，都是脚本内永久作用的

永久重定向后，有时候我们需要将输出再变为原来的输出，可以使用下面方法：自定义一个文件描述符指向原来输出，当需要变为原来输出时只要再指向自定义的即可。同样重定向输入也是这样：

恢复输出：	恢复输入
Exec 3>&1 exec 1>file exec 1>&3 Exec 3>&- #关闭文件描述符	Exec 4<&0 exec 0<file exec 0<&4 Exec 4>&-

如果要阻止命令输出或清空文件内容，可以将输出或文件内容重定向到**/dev/null**文件，null的任何数据都不会保存，相当于被丢弃了。

Linux系统的/tmp目录可以存放临时文件，系统可以在启动时自动清理。下面介绍临时文件的创建使用。

```
tempfile=`mktemp test16.XXXXXX` #创建临时文件，-t 在/tmp下建立文件；-d 建立目录
exec 3>$tempfile  #自定义输出到临时文件
echo "this script writes to temp file $tempfile"
```

```
echo "this is the first line">&3

exec 3>&-

echo "Done creating temp file.The contents are:"

cat $tempfile   #查看临时文件内容

rm -f $tempfile  #删除临时文件
```

2 高级shell脚本

2.1 脚本控制

捕捉信号：Trap command signals命令在脚本中捕捉信号，脚本受到trap中列出的信号，trap会阻止shell脚本执行它，而是执行command命令。当command为-时，则表示移除后面的信号。下面是trap的示例：

```
trap "echo 'sorry! I have trapped Ctrl-C'" SIGINT SIGTERM
```

后台运行脚本：在运行脚本时后面加上**&符号**可以后台执行，shell可以进行其他操作或者运行多个后台作业。

非终端运行脚本：nohup ./脚本 &。后台运行，即使终端退出也继续执行，输出内容存放在nohup.out文件中。

作业控制：**Jobs**：列出当前分配给shell的作业，带+的为默认的作业。**Bg 作业号**：重启停止的作业，后台执行。**Fg 作业号**：重启作业，前台执行。

优先级调整：脚本启动默认优先级是0（-20～20）。**Nice -n 10 ./脚本**：nice命令启动脚本时指定优先级。**Renice 10 -p 进程号**：调整正在运行的脚本的优先级。

定时运行作业：**at -f 脚本名 time**：在特定时间执行脚本，脚本的输出会提交到用户的e-mail，可以用mial命令查看。**Atq** :列出作业队列在等待的作业。**Atrm 作业号**：删除等待中的作业。**Min hour dayofmonth month dayogweek command**：使用cron时间表定时执行command命令。

另外在/etc下存在cron的目录，可以将需要定时执行的脚本复制到相应的cron目录中。

2.2 shell函数

Shell脚本中使用function关键字定义函数，使用echo返回值。给函数传递参数和给脚本传递参数一样，使用$n来获取参数，但是函数不能直接获取脚本参数，要传递给函数：

```
function addem {

if [ $# -eq 0 ] || [ $# -gt 2 ];then

echo -1

elif [ $# -eq 1 ];then

echo $[ $1 + $1 ]

else

echo $[ $1 + $2]

fi

}

echo -n "10加15等于："

value=`addem 10 15`  #传递参数给函数，并接受函数返回值

echo $value

echo -n "只输入一个数10："

value=`addem 10`

echo $value

echo -n "不输入参数："

value=`addem`
```

```
echo $value
```

函数也可以传递数组参数，但是需要将数组变了分解后再作为函数参数使用，不然只能得到数组的第一个值，方法如下：

```
function array {

local newarray   #local表示该变量只能在函数内使用

newarray=`echo "$@"`

echo "The new array value is:${newarray[*]}" #返回数组值

}

myarray=(1 2 3 4 5)

echo "The original array is ${myarray[*]}"

array ${myarray[*]}  #传递数组参数
```

Shell还支持函数库文件，把常用的shell函数放在同一个文件中作为库函数，需要使用时通过source命令（即点操作符）倒入库文件引用函数。

```
. ./funcLab  #source命令，注意库文件路径

value1=10

value2=5

result1=`addem $value1 $value2`

echo "相加：$result1"
```

2,3 图形化脚本

Shell最简单的是**文本菜单**，可以使用echo显示各个选择项，也可以使用select自动生成菜单，然后在case语句中判断选择执行不同的函数和命令：

```
clear
echo
echo -e "\t\tSys Admin Menu\n"
echo -e "\t1.  Display disk space"
echo -e "\t2.  Display logged on users"
echo -e "\t3.  Display memory usage"
echo -e "\t0.  Exit program\n\n"
echo -en "\t\tEnter option:"
read -n 1 option
```

```
select option in "Display disk space" "Display logged on users" "Display memory usage" "Exit program";do

case $option in

"Exit program") break;;

"Display disk space") diskspace;;

"Display logged on users") whoseon;;

"Display memory usage") menusage;;

*) clear
    echo "sorry,wrong selection";;

esac

done
```

Dialog包可以在文本环境用ANSI转义控制字符来创建标准的窗口对话框。下面演示如何在shell脚本中使用dialog。Dialog的标准格式为：**dialog --控件名称 控件参数。**下面是常用的几个控件，其他可以在dialog帮助中查看：

```
dialog --title Msgbox --msgbox "this is a msgbox" 10 20  #消息对话框

dialog --title Yesno  --yesno "this is a yesno" 10 20   #yes/no对话框

#输入对话框,注意输入会输出到stderr，需要重定向来获取

dialog --title inputBox --inputbox "Enter your name:" 10 20

dialog --title textbox --textbox /etc/passwd 15 45  #文本框，显示大量文本

#menu，创建窗口菜单,选项同样输出到stderr
```

dialog --title menu --menu "Sys Admin menu" 20 30 10 1 "Display disk space" 2 "Display users" 3 "exit program"

dialog --title fselect --fselect $HOME/ 10 50 **#fselect，浏览文件并选择**

除了使用基本的dialog包，在gnome和KDE桌面环境下有各自的扩展dialog包：**zenity和Kdialog**，下面简单介绍zenity。Zenity和dialog不太一样，它许多控件类型都用命令行上其他选项定义，而不是将它们包括在选项的参数中，具体的控件和它需要的参数选项可以在help中查看。

zenity --list --radiolist --title "SYs Admin Menu" --column "Select" --column "MenuItem" FALSE "Display disk space" FALSE "Display users" FALSE "Display memory usage" FALSE "Exit program" >$temp2 **#列表框**

zenity --info --text "Sorry,invalid selection" **#提示框**

zenity --text-info --title "Disk space" --filename=$temp --width 750 --height 10 **#文本框**

Zenity的默认输出是stdout，和dialog不同

3 其他shell脚本

上面所讲的都是基于bash脚本，在其他shell中未必都能顺利执行，下面介绍在dash和zsh 两种shell中脚本编程与bash的不同。

3.1 dash脚本

Dash是debian linux创建的shell，和bash都是bourne shell的延伸，但是比bash要简洁，因此bash中很多功能dash中没有。下面列出在dash中不能使用的bash功能。

- 在bash中可以使用$[ ]和$(( ))来进行数学运算，但在**dash中只能使用$(( ))**。
- Dash的echo可以自动识别转义字符正确输出，不需要加上-e选项
- Dash中不识别funcion 函数名{}的函数定义，只能使用**函数名(){...}方式**。

3.2 zsh脚本

Zsh shell是另一个流行的shell，可以加载命令模块，这是bash和dash不具有的功能。下面是zsh的基本不同：

- zsh数学运算除了使用**$(( ))**外，还可以使用**let命令，**并且会自动产生浮点结果。

Let value=”4*5.1/3.2”

- Zsh除了bash有的循环方式外，还有一个repeat命令，可以直接指定循环次数

Value=$((10/2))

Repeat $value;do **#直接指定数字或者算式或变量**

Echo “.....”

done

4，shell与高级工具

4.1使用数据库

在linux中mysql数据库十分流行，因此使用shell同数据库交互也很重要。下面是shell使用mysql的示例：

MYSQL=`which mysql` **#获取mysql命令的地址**

$MYSQL -u root **#登陆到mysql,不要密码，因为在my.cnf中配置了**

$MYSQL test -u root -e 'select * from employee' **#登陆同时指定数据库执行单个命令**

$MYSQL test -u root <<EOF **#使用内联重定向执行多个命令**

show tables;

insert into employee values ($1,'$2','$3','$4');

select * from employee where salary > 40000;

EOF **#不能有空格在前面**

$MYSQL test -u root -Bse 'show databases' **#s选项去除格外输出，B批处理**

4.2 使用web

Lynx是基于文本的浏览器，可以将web页面的文本内容转存到stdout中，适合挖掘web页面包含的数据。

三 Unix/Linux设计思想

准则1：小即是美
现实世界中，只要把一些小巧的解决方案组合起来，几乎没有解决不了的问题。只所以会选择实施如此巨大的解决方案，根本原因在于我们并没有完全理解该问题。

就自身而言，小程序做的事情并不多。它们经常只实现一两个功能，但要是把它们结合在一起，你就能体会到它们真正的力量。它们整体功能大于各个局部功能的简单相加。

准则2：让每一个程序只做好一件事
最好的程序全部能量只用来执行单一任务，并且将它完成的很好。程序被加载到内存中，行使完它的功能，人后退出，让下一个目标单一的程序开始运行。

准则3：尽快建立原型
只有建立原型，你的构想才能首先通过可视化，现实可行的方法得到验证。在此之前，它们只不过是你脑海中的一些零零碎碎的想法。

简短功能规格文件通常是3-4页，甚至更少。这样做的理由是没有人真正知道人们想要的功能是什么；人们很难去描述哪些还不存在的事务。

准则4：舍高效率而取可移植性
如果程序运行速度还算可以，那么就接受现实，成人它已经满足了你的需求。在捉摸优化子程序和消除关键瓶颈的同时，我们还应该思考如何在未来的硬件平台上提升性能。好程序不会消失，而会被移植到新平台。

准则5：采用纯文本文件来存储数据
要让数据具备可移植性，就必须将所有数据存储为文本，就这么简单。将文件保存为二进制格式是严格禁止的，任何特殊文件系统的格式都不行。

准则6：充分利用软件的杠杆效应
无论你有多聪明过人，精力充沛或是锐意进取，在人生的慢慢长途中，一个人的精力就只有那么多。如果想取得非凡的成就，你就必须方法自己对这个世界的影响力。

良好的程序员编写优秀的代码，优秀的程序员借用优秀的代码。如果亲自编写应用软件中的每一行代码，反而会显得你工作进度缓慢，效率低下。那些迅速有效的裁剪和组合模块的开发人员才真正拥有“就业保障”。

准则7：使用shell脚本提高杠杆效应和可移植性
在linux环境下，shell脚本一般意味着最高级别的可移植性。

准则8：避免使用强制性用户界面
CUI是一种与应用程序进行交互的模式，一旦你在命令编辑器中调用里一个应用程序，那么知道应用程序推出之前，你都无法再与命令解释器进行交互。实际上的效果就是，你完全被这个应用程序的用户界面牵绊着，知道你退出之后才重获自由。

由于人类使用计算机的限制，因此任何时候需要等待用户输入的系统，其处理速度就与人类操作速度相当。换言之，这并不是很快。

准则9：寻求90%的解决方案
90%解决方案的精髓在于，我们需要故意忽略那些代价昂贵，费时费力或难以执行的项目。如果有机会将那些最棘手的10%放弃，你肯定可以轻而易举的解决世界上的大部分问题。