第五部分　高级shell编程技巧

第25章　深入讨论<<

我们在介绍标准输入和标准输出以及 while循环的时候已经几次遇到 <<的应用。我们学会了如何发送邮件，如何构建一个菜单，不过 <<还有很多其他的用法。

本章将介绍以下内容：

- 快速创建一个文件。
- 自动进入菜单。
- ftp传输。
- 连接至其他应用系统。

该命令的一般形式为：

```
command <<word
text
word
```

这里再简要回顾一下 <<的用法。当shell看到<<的时候，它就会知道下一个词是一个分界符。在该分界符以后的内容都被当作输入，直到 shell又看到该分界符(位于单独的一行)。这个分界符可以是你所定义的任何字符串。

可以使用<<来创建文件、显示文件列表、排序文件列表以及创建屏幕输入。

25.1　快速创建一个文件

可以使用这种方法快速创建一个文件，并向其中存入一些文本：

```
$ cat >> myfile <<NEWFILE
```

现在可以输入一些文本，结束时只要在新的一行键入 NEWFILE即可，这样就创建了一个名为myfile的文件，该文件中包含了一些文本。

如果打开了一个已经存在的文件，输入的内容会附加到该文件的末尾。

如果使用tab键，注意，一些老版本的 shell可能无法正确理解它的含义。为了解决这一问题，可以在<<之后加一个横杠-，就像下面这样：

```
cat >> myfile <<- NEWFILE
...
```

25.2　快速创建打印文档

假如希望打印一小段信息，可以采用这种方法而不必使用 vi编辑器。在本例中，一旦在输入QUICKDOC之后按回车键，相应的文档就会被送到打印机。

```
$ lpr <<QUICKDOC
   **** INVITATION****
```

```
 The Star Trek convention is in town
next week. Be there.

Ticket prices: (please phone)
------------------------------
QUICKDOC
```

25.3 自动选择菜单

不但可以很方便地使用 <<创建菜单屏幕，还可以使用它来自动选择菜单，而不是由用户手工进行选择。

我编写了一个菜单驱动的数据库管理脚本，可以使用它来完成备份和其他系统管理任务。该脚本本来是在白天由用户来运行的，现在决定把这些工作交给 cron夜间完成，我不想再另外写一个自动运行的脚本，于是我使用 <<中的输入来选择syb_backup脚本的菜单选项。下面介绍一下该脚本的菜单。

主菜单如下，选择2：

```
1: Admin Tasks
2: Sybase Backups
3: Maintenance Tasks
Selection > 2
```

第二层菜单如下，选择3：

```
1: Backup A Single Database
2: Backup Selected Databases
3: Backup All Databases
Selection > 3
```

第三级菜单如下，选择Y：

```
1. dw_levels
2. dw_based
3. dw_aggs
...
...
```

从菜单来看，如果要备份所有的数据库，需要键入：

1) 菜单脚本的名字，syb_backup。

2) 键入2。

3) 键入3。

4) 键入Y。

下面的脚本能够自动运行数据库备份脚本 syb_backup：

```
$ pg auto.sybackup
#!/bin/sh
# set the path
PATH=/usr/bin:/usr/sbin:/sybase/bin:$LOCALBIN
export PATH
# set the sybase variable
DSQUERY=COMET; export DSQUERY
# set the TERM and init it
TERM=vt220; export TERM
```

```
tput -T vt220 init
#  keep a log of all output
log_f=/logs/sql.backup.log
#
>$log_f

# here's the code that does all the work !
/usr/local/sybin/syb_backup >> $log_f 2>&1  << MAYDAY
2
3
Y
MAYDAY

chown sybase $log_f
```

该脚本中的重定向部分是：

```
usr/local/sybin/syb_backup > > $log_f  2 > &1  < < MAYDAY
2
3
Y
MAYDAY
```

让我们来分析一下这一部分，这里给出了脚本 syb_backup的全路径；>>$log_f 2>&1意味着所有的输出都重定向到 $log_f中，该变量的值为 /logs/sql.backup.log。这是一个良好的习惯，因为这样就能够捕捉到所运行的程序或脚本的所有输出，如果出现错误的话，也能够被记录下来。

<<MAYDAY之后的内容就是手工运行 syb_backup脚本所需要输入的内容，直到遇到另外一个MAYDAY结束。

这样，我就不需要重新再写一个脚本；如果已经有一个菜单驱动的脚本，只需再编写一个使用<<输入的脚本就可以自动运行原先的脚本。

25.4 自动ftp传输

<<的另外一个流行的应用就是自动 ftp传输。在使用 ftp时，如果能够向用户提供一个简单的界面就好了。下面的脚本使用了匿名用户 anonymous建立了一个 ftp连接。这是一个特殊的用户，它使得系统能够创建一个含有公共目录的安全帐户。一般来说，所有以匿名用户身份进行连接的用户都只能从公共目录中下载文件，不过只要权限允许，用户也可以上载。

匿名用户的口令可以是任何字符串，不过最好使用主机名加上本地用户名，或电子邮件地址。

下面的脚本将会提示如下的信息：

1) 希望登录的远程主机。

2) 文件传输的类型是二进制方式还是 ASCII方式。

3) 要下载的文件名。

4) 存放下载文件的本地目录。

当用户输入想要连接的主机之后，首先执行一个名为 traceroute的脚本验证本地主机是否能够连接到远程主机。如果 traceroute执行失败，这个自动 ftp传输的脚本将会再次提示用户输入主机名。

用户在看到传输模式选择的提示之后按回车键，将会选择缺省的二进制模式。

用户在输入所要下载的文件名之后，将会被提示输入保存下载文件的本地目录。缺省的本地目录是/tmp。如果用户所给出的目录无法找到，仍将使用缺省的 /tmp目录。

下载文件在本地的文件名将是原文件名加上 .ftp后缀。

最后，用户所有的选择都将在屏幕上显示出来，待用户确认后开始进行传输。

下面就是该脚本运行时在屏幕上的显示：

```
$ ftpauto
User: dave              05/06/1999                  This host: bumper
                FTP RETRIEVAL / POSTING SCRIPT
                ==============================
                      Using the ID of anonymous
Enter the host you wish to access :uniware
Wait..seeing if uniware is out there..
bumper can see uniware
What type of transfer / receive mode ?
 1 : Binary
 2 : ASCII
Your choice [1..2] [1]:
   Enter the name of the file to retrieve :gnutar.Z
 Enter the directory where the file is to be placed[/tmp] :
                Host to connect is: uniware
                File to get is    : gnutar.Z
                Mode to use is    : binary
                File to be put in : /tmp/gnutar.Z.ftp
          Ready to get file 'y' or 'q' to quit? [y..q] :
```

下面就是该脚本的内容：

```
$ pg ftpauto
#!/bin/sh
# ftp script
# ftpauto
USER=`whoami`
MYDATE=`date +%d/%m/%Y`
THIS_HOST=`hostname -s`
tracelog=/tmp/tracelog.$$

while :
do
  # loop forever
  tput clear
  cat <<MAYDAY
  User: $USER              $MYDATE                  This host: $THIS_HOST
                  FTP RETRIEVAL SCRIPT
                  ====================
                Using the ID of anonymous
  MAYDAY
  echo -n "Enter the host you wish to access :"
  read DEST_HOST
  # is a hostname entered ???
  if [ "$DEST_HOST" = "" ]
  then
    echo "No destination host entered" >&2
    exit 1
  fi
```

```
  # can we see the host ???
  echo "Wait..seeing if $DEST_HOST is out there.."
  # use traceroute to test connectivity
  traceroute $DEST_HOST > $tracelog 2>&1

  if grep "unknown host" $tracelog  >/dev/null 2> then
    echo "Could not locate $DEST_HOST"
    echo -n "Try another host? [y..n] :"
    read ANS
    case $ANS in
    y|Y) ;;
    *) break;;   # get out of the forever loop
    esac
  else
    echo "$THIS_HOST can see $DEST_HOST"
    break   # get out of the forever loop
  fi
done

# the default is binary
echo  "What type of transfer /receive mode ?"
echo  " 1 : Binary"
echo  " 2 : ASCII"
echo -n -e "\fYour choice [1..2] [1]:"
read $TYPE
case $TYPE in
1) MODE=binary
  ;;
2) MODE=ascii
  ;;
*) MODE=binary
  ;;
esac

echo -n "   Enter the name of the file to retrieve :"
read FILENAME
if [ "$FILENAME" = "" ]; then
  echo "No filename entered" >&2
  exit 1
fi

# default is tmp
echo -n -e "\f Enter the directory where the file is to be placed[/tmp] :"
read DIREC
cd $DIREC >/dev/null 2>&1
# if we cannot cd to the directory then use tmp
if [ "$DIREC" = "" ]; then
  DIREC=/tmp
fi

if [ $? != 0 ]
then
  echo "$DIREC does not exist placing the file in /tmp anyway"
  DIREC=/tmp
fi
```

```
echo -e "\t\tHost to connect is: $DEST_HOST"
echo -e "\t\tFile to get is    : $FILENAME"
echo -e "\t\tMode to use is    : $MODE"
echo -e "\t\tFile to be put in : $DIREC/$FILENAME.ftp"
echo -e -n "\t\tReady to get file 'y' or 'q' to quit? [y..q] :"
read ANS
case $ANS in
Y|y);;
q|Q) exit 0;;
*) exit 0 ;;
esac
echo "ftp.."
ftp -i -n $DEST_HOST<<FTPIT
user anonymous $USER@$THISHOST
$MODE
get $FILENAME $DIREC/$FILENAME.ftp
quit
FTPIT
if [ -s $DIREC/$FILENAME.ftp ]
then
  echo "File is down"
else
  echo "Unable to locate $FILENAME.ftp"
fi
```

在ftp命令中使用<<时，使用了ftp -i -n选项，这意味着不要自动登录，而且关闭交互模式。这样就使得脚本可以使用 user命令进行登录。口令是 $USER@THISHOST，在这里就是dave@bumper。

如果用户每天从同一台主机上下载相同的文件，比如说是包含前一天销售数据的文件，那么用户就没有必要每天都输入同样的主机名和文件名。可以设置 DEST_HOST和FILENAME变量的缺省值，这样就可以使用户不必每天都输入同样的主机名和文件名。

下面是ftp自动传输脚本中提示用户输入主机名的一段，但是现在不同的是， DEST_HOST变量已设置了缺省值my_favourite_host。现在用户可以另外输入一个不同的主机名，也可以敲回车键选择缺省值。

注意，现在不必再检查用户是否输入了一个值，因为如果用户没有输入的话，该变量将被赋予缺省值。

```
echo -n "Enter the host you wish to access :"
read DEST_HOST
: ${DEST_HOST:="my_favourite_host"}
echo "Wait..seeing if $DEST_HOST is out there.."
traceroute $DEST_HOST >$tracelog 2>&1
...
```

25.5 访问数据库

shell脚本一个常用的用途就是访问数据库系统获得信息。实现这样的功能， <<是再理想不过了。可以用它来输入你在面对数据库提示时所做的各种选择。下面的例子并不是数据库中的一个练习，而是为了用来介绍如何使用 <<来连接其他应用程序，完成相应的任务。

对于某一个数据库系统来说，在使用某种第三方产品进行访问时， select into功能将会被

关闭。这意味着该数据库不能被用来插入数据或创建临时表。

为了解决这个问题，我们使用 <<进行数据库连接，并使用一个 for循环来提供各个数据库名，一旦连接成功，<<将用来向sql命令提供选项。

下面就是该脚本。

```
$ pg set.select
#!/bin/sh
# set.select
# fixes known bug. Sets the select into db option on
PATH=$PATH:/sybase/bin:/sybase/install
export PATH
SYBASE="/sybase"; export SYBASE
DSQUERY=ACESRV; export DSQUERY
PASSWORD="prilog"
DATABASES="dwbased tempdb aggs levels reps accounts"

for loop in $DATABASES
do
  su sybase -c '/sybase/bin/isql -Usa -P$PASSWORD' << MAYDAY
  use master
  go
  sp_dboption $loop,"select into/bulkcopy/pllsort",true
  go
  use $loop
  go
  checkpoint
  go
  MAYDAY
done
```

让我们来看一看使用<<的部分，shell在执行了变量替代以后将运行下面的一段命令。

```
use master
go
sp_dboption dwbased,"select into/bulkcopy/pllsort",true
   go
   use dw_based
   go
   checkpoint
   go
```

当shell看到结束的分界符MAYDAY时，该脚本将开始下一次循环，对另外一个数据库进行操作。下面就是运行的结果：

```
$ set.select
Database option 'select into/bulkcopy/pllsort' turned ON for database
'dwbased'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Database option 'select into/bulkcopy/pllsort' turned ON for database
'tempdb'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Database option 'select into/bulkcopy/pllsort' turned ON for database
'aggs'.
Run the CHECKPOINT command in the database that was changed.
```

```
(return status = 0)
```

25.6 小结

本章进一步给出了一些使用 <<来自动完成某些任务的例子。 <<的用途很广，特别是在连接某些应用程序或使用 ftp时。你可以灵活地使用 <<来自动运行以前编写的脚本，从而完成各种不同的任务。