[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY01/CASE/01/index.html" \l "page_top_case)

# NSD SHELL DAY01

1. [案例1：Shell基础应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY01/CASE/01/index.html" \l "case1)
2. [案例2：简单Shell脚本的设计](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY01/CASE/01/index.html" \l "case2)
3. [案例3：使用Shell变量](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY01/CASE/01/index.html" \l "case3)
4. [案例4：变量的扩展应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY01/CASE/01/index.html" \l "case4)
5. [案例5：Shell中的数值运算](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY01/CASE/01/index.html" \l "case5)

## 1 案例1：Shell基础应用

### 1.1 问题

本案例要求熟悉Linux Shell环境的特点，主要练习以下操作：

1. 切换用户的Shell环境
2. 练习命令历史、命令别名
3. 重定向标准输入/输出/错误输出
4. 管道操作实践

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：切换用户的Shell环境

什么是shell？

Shell是在Linux内核与用户之间的解释器程序，通常指的是bash，负责向内核翻译及传达用户/程序指令，如图-1所示。



图-1

shell的使用方式：

1.交互执行指令：人工干预，执行效率底。

2.非交互执行指令：安静地在后台执行，执行效率高，方便写脚本。

若需要临时使用另一种Shell环境，可以直接执行对应的Shell解释器程序，比如只要执行ksh可以切换到ksh命令行环境。

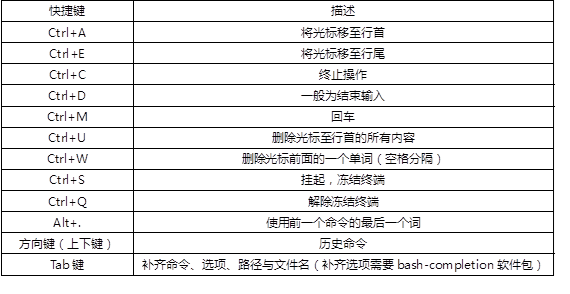
1. [root@svr5 ~]# yum -y install ksh             //若缺少ksh请先安装ksh包
2. .. ..
3. [root@svr5 ~]# cat /etc/shells
4. /bin/sh
5. /bin/bash
6. /sbin/nologin
7. /bin/tcsh
8. /bin/csh
9. /bin/ksh                                        //确认当前系统已识别ksh
10. [root@svr5 ~]# ksh                             //进入ksh环境
11. [root@svr5]~# exit                             //返回到切换前的bash环境

若希望修改用户的登录Shell，管理员可以直接通过usermod（或useradd）命令设置。比如，以下操作可将用户zhangsan的登录Shell改为/bin/ksh：

1. [root@svr5 ~]# usermod -s /bin/ksh zhangsan             //执行修改操作
2. [root@svr5 ~]# grep 'zhangsan' /etc/passwd
3. zhangsan:x:516:516::/home/zhangsan:/bin/ksh             //修改后

步骤二：快捷键与Tab键补齐，常见快捷键如表-1所示

表-1



步骤三：练习命令历史

1）检查历史命令的容量。

默认记录1000条，通过全局变量HISTSIZE设置，对所有用户有效：

1. [root@svr5 ~]# grep HISTSIZE /etc/profile
2. HISTSIZE=1000

查看已为当前用户记录的历史命令条数：

1. [root@svr5 ~]# history | wc -l
2. 1000

2）查看历史命令列表。

列出最近执行的10条历史命令：

1. [root@svr5 ~]# history | tail
2. 1028 grep 'zhangsan' /etc/passwd
3. 1029 cat /etc/redhat-release
4. 1030 usermod -s /bin/tcsh zhangsan
5. 1031 grep 'zhangsan' /etc/passwd
6. 1032 su - zhangsan
7. 1033 echo 1234567 | passwd --stdin zhangsan
8. 1034 su - zhangsan
9. 1035 grep HISTSIZE /etc/profile
10. 1036 history | wc -l
11. 1037 history | tail

3）调用指定的历史命令。

重新执行历史命令列表中的第1028条操作：

1. [root@svr5 ~]# !1028
2. grep 'zhangsan' /etc/passwd
3. zhangsan:x:516:516::/home/zhangsan:/bin/bash

重新执行最近一次以cat开头（根据实际情况变更）的历史命令操作：

1. [root@svr5 ~]# !cat
2. cat /etc/redhat-release
3. Red Hat Enterprise Linux Server… …

步骤四：练习命令别名

1）查看已经定义的命令别名列表。

当前的别名列表：

1. [root@svr5 ~]# alias
2. alias cp='cp -i'
3. alias l.='ls -d .\* --color=tty'
4. alias ll='ls -l --color=tty'
5. … …

步骤四：重定向标准输入/输出/错误输出

1）重定向标准输出。

使用 > 将命令执行的正常输出重定向到文件：

1. [root@svr5 ~]# ls -ld /etc/                        //正常应输出到屏幕
2. drwxr-xr-x. 140 root root 8192 8月 2 04:45 /etc/
3. [root@svr5 ~]# ls -ld /etc/ > stdout.txt             //重定向到文件
4. [root@svr5 ~]# cat stdout.txt                     //确认重定向输出的结果
5. drwxr-xr-x. 140 root root 8192 8月 2 04:45 /etc/

> 操作会覆盖目标文件（先清空、再写入）：

1. [root@svr5 ~]# echo "I am the king." > stdout.txt         //覆盖目标文件
2. [root@svr5 ~]# cat stdout.txt                         //确认结果
3. I am the king.

改用 >> 可实现追加重定向输出：

1. [root@svr5 ~]# ls -ld /etc/ >> stdout.txt             //追加输出
2. [root@svr5 ~]# cat stdout.txt
3. I am the king.                                         //原有内容还保留
4. drwxr-xr-x. 140 root root 8192 8月 2 04:45 /etc/

2）重定向标准错误。

对于命令执行出错的信息，使用 > 无法保存，仍然会输出到屏幕。比如，可使用ls命令同时查看两个对象（其中nb.txt并不存在），重定向输出：

1. [root@svr5 ~]# ls -l nb.txt /etc/fstab > stderr.txt
2. [root@svr5 ~]# cat stderr.txt                 //正常信息成功重定向到目标文件
3. -rw-r--r--. 1 root root 541 1月 5 2017 /etc/fstab

使用 2> 可重定向错误信息，比如，可执行一个错误的命令：

1. [root@svr5 ~]# ls -l nb.txt /etc/fstab 2> stderr.txt
2. -rw-r--r--. 1 root root 541 1月 5 2017 /etc/fstab
3. //正确的信息默认输出至屏幕，错误信息重定向到目标文件
4. [root@svr5 ~]# cat stderr.txt                        //从文件中查看出错信息
5. ls: nb.txt: 没有那个文件或目录

类似的，2>> 可实现追加输出：

1. [root@svr5 ~]# ls tmpfile 2>> stderr.txt
2. [root@svr5 ~]# cat stderr.txt
3. ls: nb.txt: 没有那个文件或目录
4. ls: tmpfile: 没有那个文件或目录

若希望将正常输出、错误输出重定向同一个文件，可使用 &> :

1. [root@svr5 ~]# ls -l nb.txt /etc/fstab &> stderr.txt
2. [root@svr5 ~]# cat stderr.txt
3. ls: nb.txt: 没有那个文件或目录
4. -rw-r--r--. 1 root root 541 1月 5 2017 /etc/fstab

3）重定向标准输入。

1. [root@svr5 ~]# mail -s Error root < /etc/passwd

4）案例

1. [root@svr5 ~]# cat /root/test.sh
2. #!/bin/bash
3. for i in {1..10}
4. do
5. useradd user$i 2>>/root/user.log
6. echo '123456' | passwd --stdin user$i >/dev/null
7. done

步骤五：管道操作实践

借助于管道符“|”，可以将一条命令的标准输出交给另一条命令处理，在一条命令行内可依次使用多个管道。

1）统计/etc/目录下资料的数量。

1. [root@svr5 ~]# ls -l /etc | wc -l
2. 254

2）列出Yum库里名称中含cluster的软件包。

1. [root@svr5 ~]# yum list | grep cluster
2. cluster-cim.x86\_64                 0.12.1-7.el5     RHEL5-Cluster
3. cluster-snmp.x86\_64                 0.12.1-7.el5     RHEL5-Cluster
4. … …

## 2 案例2：简单Shell脚本的设计

### 2.1 问题

本案例要求编写三个脚本程序，分别实现以下目标：

1. 在屏幕上输出一段文字“Hello World”
2. 能够为本机快速配好Yum仓库
3. 能够为本机快速装配好vsftpd服务

### 2.2 方案

一个规范的Shell脚本构成包括：

* 脚本声明（需要的解释器、作者信息等）
* 注释信息（步骤、思路、用途、变量含义等）
* 可执行语句（操作代码）

### 2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写第一个Shell脚本，输出“Hello World”

1）根据手动任务操作编写脚本文件

1. [root@svr5 ~]# vim /root/first.sh
2. #!/bin/bash
3. echo 'Hello World'
4. exit
5. [root@svr5 ~]# chmod +x /root/first.sh                     //添加可执行权限

2）执行脚本，测试效果

1. [root@svr5 ~]# /root/first.sh
2. Hello World

3）Shell脚本的执行方式：

方法一，作为“命令字”：指定脚本文件的路径，前提是有 x 权限

1. [root@svr5 ~]# ./first.sh                            //指定相对路径
2. [root@svr5 ~]# /root/first.sh                        //指定绝对路径

方法二，作为“参数”：使用bash、sh、source来加载脚本文件

1. [root@svr5 ~]# bash first.sh                        //开启子进程
2. [root@svr5 ~]# sh        first.sh                        //开启子进程
3. [root@svr5 ~]# source first.sh                        //不开启子进程

步骤二：编写为本机快速配Yum仓库的Shell脚本

1）Yum仓库配置文件的参考内容如下：

1. [root@svr5 ~]# cat /etc/yum.repos.d/rhel.repo
2. [rhel]
3. name=redhat
4. baseurl=file:///misc/cd //yum源不一样，不能直接抄！！！
5. enabled=1
6. gpgcheck=0

2）根据参考文件的内容，编写脚本内容如下：

1. [root@svr5 ~]# vim /root/yum.sh
2. #!/bin/bash
3. rm -rf /etc/yum.repos.d/\*.repo
4. echo '[rhel]
5. name=redhat
6. baseurl=[file:///misc/cd](/home/student/文档\\x/misc/cd)
7. gpgcheck=0
8. ' > /etc/yum.repos.d/rhel.repo
9. [[root@svr5](mailto:root@svr5) ~]# chmod +x /root/yum.sh                     //添加可执行权限

3）执行脚本，测试效果

执行配置Yum仓库的脚本：

1. [root@svr5 ~]# /root/yum.sh

检查配置结果：

1. [root@svr5 ~]# ls /etc/yum.repos.d/\*                         //仓库配置已建立
2. /etc/yum.repos.d/rhel.repo
3. [[root@svr5](mailto:root@svr5) ~]# yum repolist                                 //Yum仓库已可用
4. rhel-packages | 3.9 kB 00:00 ...
5. rhel-packages/primary\_db | 3.1 MB 00:00 ...
6. repo id repo name status
7. rhel             redhat                     3,690
8. repolist: 3,690

步骤三：编写快速装配vsftpd服务的Shell脚本

1）编写参考脚本文件如下：

1. [root@svr5 ~]# vim /root/ftpon.sh
2. #!/bin/bash
3. yum -y install vsftpd &> /dev/null
4. systemctl start vsftpd
5. systemctl enable vsftpd
6. cp /etc/hosts /var/ftp/pub
7. #拷贝一个文件，放到FTP共享目录下
8. [root@svr5 ~]# chmod +x /root/ftpon.sh                 //添加可执行权限

3）执行脚本，测试效果

执行快速装配vsftpd服务的脚本：

1. [root@svr5 ~]# /root/ftpon.sh

确认脚本执行结果：

1. [root@svr5 ~]# rpm -q vsftpd
2. vsftpd-3.0.2-10.el7.x86\_64
3. [root@svr5 ~]# systemctl status vsftpd

## 3 案例3：使用Shell变量

### 3.1 问题

本案例要求熟悉Shell变量的使用，主要练习或验证下列内容：

1. 定义/赋值/查看变量
2. 环境/预定义/位置变量的应用

除了学会建立和引用变量以外，还要认识环境变量PWD、USER、HOME、SHELL，还有预定义变量$0、$$、$?、$#、$\*，以及位置变量$1、$2、$10、……的作用。

### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：变量的定义/赋值/查看

1）新建/赋值变量

新建变量test，赋值“hello world”，通过set命令可以检查变量设置：

1. [root@svr5 ~]# test=11

2）查看变量

通过echo $变量名 可输出变量值：

1. [root@svr5 ~]# echo $test
2. 11

查看变量时，若变量名称与后面要输出的字符串连在一起，则应该以{}将变量名括起来以便区分：

1. [root@svr5 ~]# echo $testRMB             //无法识别变量名test
2. [root@svr5 ~]# echo ${test}RMB             //区分后可以识别
3. 11RMB

3）撤销自定义变量

若要撤销已有的变量，可使用unset命令：

1. [root@svr5 ~]# unset test                 //撤销变量test
2. [root@svr5 ~]# echo $test                 //查看时已无结果

步骤二：使用环境变量

1）查看环境变量相关文件

全局文件为/etc/profile，对所有用户有效；用户文件为~/.bash\_profile，仅对指定的用户有效。

查看/etc/profile文件内容：

1. [root@svr5 ~]# cat /etc/profile
2. .. ..
3. HOSTNAME=`/bin/hostname`
4. HISTSIZE=1000
5. .. ..
6. export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
7. .. ..

2）使用环境变量

当前用户的环境变量USER记录了用户名、HOME记录了宿主目录、SHELL记录了登录Shell、HOSTNAME记录主机名、UID是用户的id号：

1. [root@svr5 ~]# echo $USER $HOME $SHELL $UID
2. root /root /bin/bash 0
3. [root@svr5 ~]# echo $HOSTNAME
4. svr5

环境变量PS1表示Shell环境的一级提示符，即命令行提示符（\u 用户名、\h 主机名、\W 工作目录、\$ 权限标识）：

1. [root@svr5 src]# echo $PS1                 //查看默认的一级提示
2. [\u@\h \W]\$
3. [root@svr5 src]#PS1='hehe#'                 //修改一级提示
4. hehe#                                     //更改结果
5. hehe# PS1='[\u@\h \W]\$ '                 //恢复原有设置
6. [root@svr5 src]#

环境变量PS2表示二级提示符，出现在强制换行、at任务编辑等场合：

1. [root@svr5 ~]# echo $PS2                     //查看默认的二级提示
2. >
3. [root@svr5 src]# cd \                     //强制换行，观察提示符效果
4. > /root/
5. [root@svr5 ~]# PS2='=> '                     //手动修改二级提示
6. [root@svr5 ~]# cd \                         //再次验证提示符效果
7. => ~
8. [root@svr5 ~]# PS2='> '                     //恢复原有设置

3）查看系统变量

使用env可查看所有环境变量：

1. [root@svr5 src]# env
2. HOSTNAME=svr5.tarena.com
3. SHELL=/bin/bash
4. HISTSIZE=1000
5. SSH\_CLIENT=192.168.4.110 59026 22
6. OLDPWD=/root
7. SSH\_TTY=/dev/pts/0
8. USER=root
9. .. ..

使用set可查看所有变量（包括env能看到的环境变量）：

1. [root@svr5 src]# set
2. BASH=/bin/bash
3. BASH\_ARGC=()
4. BASH\_ARGV=()
5. BASH\_LINENO=()
6. .. ..

步骤三：使用位置变量与预定义变量

1）创建一个测试脚本，用来展示。

1. [root@svr5 ~]# vim location.sh
2. #!/bin/bash
3. echo $0                                        //脚本的名称
4. echo $1                                        //第一个参数
5. echo $2                                        //第二个参数
6. echo $\*                                        //所有参数
7. echo $#                                        //所有的综合
8. echo $$                                        //当前进程的进程号
9. echo $?                                        //上一个程序的返回状态码
10. [root@svr5 ~]# chmod +x location.sh             //添加可执行权限

2）执行脚本location.sh，细心观察结果（高明的武功需要用心参悟）。

1. [root@svr5 ~]# ./location.sh one 1 2 abc qq 8 7

步骤四：创建账户与修改密码的脚本

1）编写脚本。

1. [root@svr5 ~]# vim /root/user.sh
2. #!/bin/bash
3. useradd $1
4. echo "$2" |passwd --stdin $1

执行脚本测试：

1. [root@svr5 ~]# ./user.sh jerry 123456
2. 更改用户 jerry 的密码 。
3. passwd： 所有的身份验证令牌已经成功更新。

## 4 案例4：变量的扩展应用

### 4.1 问题

本案例要求进一步熟悉Shell变量的赋值控制，主要练习或验证下列内容：

1. 三种引号对赋值的影响
2. 使用read命令从键盘读取变量值
3. 使用export发布全局变量

### 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：三种引号对变量赋值的影响

1）双引号的应用

使用双引号可以界定一个完整字符串。

1. [root@svr5 ~]# xx=Tarena IT Group
2. -bash: IT: command not found                     //未界定时赋值失败
3. [root@svr5 ~]# xx="Tarena IT Group"             //界定后成功
4. [root@svr5 ~]# touch aa bb                     //创建了两个文件
5. [root@svr5 ~]# touch "aa bb"                    //创建了一个文件
6. [root@svr5 ~]# ls                                //查看结果

2）单引号的应用

界定一个完整的字符串，并且可以实现屏蔽特殊符号的功能。

1. [root@svr5 ~]# test=11
2. [root@svr5 ~]# echo "$test"
3. [root@svr5 ~]# echo '$test'

3）反撇号或$()的应用

使用反撇号或$()时，可以将命令执行的标准输出作为字符串存储，因此称为命令替换。

1. [root@svr5 ~]# tar -czf log-`date +%Y%m%d`.tar.gz /var/log

步骤二：使用read命令从键盘读取变量值

1）read基本用法

执行后从会等待并接受用户输入（无任何提示的情况），并赋值给变量str：

1. [root@svr5 ~]# read str
2. What's happen ?                         //随便输入一些文字，按Enter键提交
3. [root@svr5 ~]# echo $str                 //查看赋值结果
4. What's happen ?

为了不至于使用户不知所措、莫名其妙，推荐的做法是结合-p选项给出友好提示：

1. [root@svr5 ~]# read -p "请输入一个整数：" i
2. 请输入一个整数：240
3. [root@svr5 ~]# echo $i
4. 240

2）stty终端显示控制

将回显功能关闭（stty -echo），

将回显功能恢复（stty echo）。

可参考下列操作创建一个测试脚本：

1. [root@svr5 ~]# vim user.sh             //创建一个测试脚本
2. #!/bin/bash
3. read -p "请输入用户名:" username             //读取用户名
4. stty -echo                                 //关闭回显
5. read -p "请输入密码:" passwd             //读取密码
6. stty echo                                 //恢复回显
7. echo ""                                     //恢复回显后补一个空行
8. useradd "$username"
9. echo "$passwd" | passwd --stdin "$username"
10. [root@svr5 ~]# chmod +x user.sh         //添加执行权限

执行测试脚本user.sh，验证效果：

1. [root@svr5 ~]# ./user.sh
2. 请输入用户名: root                         //输入root，回车
3. 请输入密码:                                 //输入1234567（不会显示），回车

步骤三：使用export发布全局变量

默认情况下，自定义的变量为局部变量，只在当前Shell环境中有效，而在子Shell环境中无法直接使用。比如已定义的SCHOOL变量，当进入到sh或bash子Shell后，变量SCHOOL将处于未定义的状态：

1. [root@svr5 ~]# yy="Tarena IT Group"
2. [root@svr5 ~]# echo $yy
3. Tarena IT Group
4. [root@svr5 ~]# bash                             //开启bash子进程
5. [root@svr5 ~]# echo $yy                         //查看SCHOOL变量值无结果
6. [root@svr5 ~]# exit                             //返回原有Shell环境
7. exit
8. [root@svr5 ~]# echo $yy

若希望定义的变量能被子进程使用，可以使用export命令将其发布为全局变量。使用export发布时，只需指定变量名（可以有多个）即可，也可以通过export命令直接设置新的全局变量：

1. [root@svr5 ~]# export yy                             //发布已定义的变量
2. [root@svr5 ~]# export XX="1234"                     //发布新变量

验证刚刚发布的全局变量：

1. [root@svr5 ~]# bash                                 //进入bash子Shell环境
2. [root@svr5 ~]# echo $yy                             //查看全局变量的值 .. ..
3. Tarena IT Group
4. [root@svr5 ~]# echo $XX
5. 1234

## 5 案例5：Shell中的数值运算

### 5.1 问题

本案例要求熟悉Linux Shell环境的特点，主要练习以下操作：

* 使用expr、$[ ]、let等整数运算工具：定义变量X=1234，然后计算X与78的四则运算及求模结果
* 使用bc实现小数运算操作：以交互方式计算12.34与56.78的四则运算结果，另外再以非交互方式重复上述计算，最多显示4位小数

### 5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：整数运算工具

1）使用expr命令

乘法操作应采用 \\* 转义，避免被作为Shell通配符；参与运算的整数值与运算操作符之间需要以空格分开，引用变量时必须加$符号。

首先定义变量X=1234，然后分别计算与78的加减乘除和求模运算结果：

1. [root@svr5 ~]# X=1234                             //定义变量X
2. [root@svr5 ~]# expr $X + 78                     //加法
3. 1312
4. [root@svr5 ~]# expr $X - 78                     //减法
5. 1156
6. [root@svr5 ~]# expr $X \\* 78                     //乘法，操作符应添加\转义
7. 96252
8. [root@svr5 ~]# expr $X / 78                     //除法，仅保留整除结果
9. 15
10. [root@svr5 ~]# expr $X % 78                     //求模
11. 64

2）使用$[]或$(())表达式

乘法操作\*无需转义，运算符两侧可以无空格；引用变量可省略 $ 符号；计算结果替换表达式本身，可结合echo命令输出。

同样对于变量X=1234，分别计算与78的加减乘除和求模运算结果：

1. [root@svr5 ~]# X=1234
2. [root@svr5 ~]# echo $[X+78]
3. 1312
4. [root@svr5 ~]# echo $[X-78]
5. 1156
6. [root@svr5 ~]# echo $[X\*78]
7. 96252
8. [root@svr5 ~]# echo $[X/78]
9. 15
10. [root@svr5 ~]# echo $[X%78]
11. 64

3）使用let命令

expr或$[]、$(())方式只进行运算，并不会改变变量的值；而let命令可以直接对变量值做运算再保存新的值。因此变量X=1234，在执行let运算后的值会变更；另外，let运算操作并不显示结果，但是可以结合echo命令来查看：

1. [root@svr5 ~]# X=1234
2. [root@svr5 ~]# let y=X+22
3. [root@svr5 ~]# echo $y
4. 1256
5. [root@svr5 ~]# let X++; echo $X        # X++(X=X+1)
6. [root@svr5 ~]# let X--; echo $X        # X--(X=X-1)
7. [root@svr5 ~]# let X+=78 ; echo $X        # X+=78(X=X+78)
8. [root@svr5 ~]# let X-=78 ; echo $X     # X-=78(X=X-78)
9. [root@svr5 ~]# let X\*=78 ; echo $X     # X\*=78(X=X\*78)
10. [root@svr5 ~]# let X/=78 ; echo $X     # X/=78(X=X/78)
11. [root@svr5 ~]# let X%=78 ; echo $X     # X%=78(X=X%78)

步骤二：小数运算工具

1）bc交互式运算

先执行bc命令进入交互环境，然后再输入需要计算的表达式。以计算小数12.34与5.678的四则运算为例，相关操作如下：

1. [root@svr5 ~]# bc
2. bc 1.06.95
3. Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
4. This is free software with ABSOLUTELY NO WARRANTY.
5. For details type `warranty'.
6. 12.34+56.78                                        //加法
7. 69.12
8. 12.34-56.78                                        //减法
9. -44.44
10. 12.34\*56.78                                        //乘法
11. 700.66
12. 12.34/56.78                                        //除法
13. 0
14. quit                                             //退出交互计算器
15. [root@svr5 ~]#

2）bc非交互式运算

将需要运算的表达式通过管道操作交给bc运算。注意，小数位的长度可采用scale=N限制，除此以外也受参与运算的数值的小数位影响。以计算小数12.34与5.678的四则运算为例，相关操作如下：

1. [root@svr5 ~]# echo 'scale=4;12.34+5.678' | bc
2. 18.018
3. [root@svr5 ~]# echo 'scale=4;12.34\*5.678' | bc
4. 70.0665
5. [root@svr5 ~]# echo 'scale=4;12.34/5.678' | bc
6. 2.1733