本读书笔记分为4个主要部分，第一部分：linux命令行；第二部分：shell脚本编程基础；第三部分：高级shell脚本编程；第四部分：高级shell脚本编程主题。

2017-09-25：

结尾感言：终于读完了这本书，先自我满足一下；其实sed部分有一些缺憾，但是sed用的太少了，不如gawk好用，那部分就暂时不管了，最后几章介绍的内容也比较简单，就没有做过多的说明，这些内容以后有时间再打算补；写完了本书的所有章节的知识总结；总共达到了10000字，页数达到了40页，真是挺多的，有些知识部分的细节我觉得有助于后续的复习，比如一些参数什么的都贴了上来，虽然如此，相比原书本六几百多页的大小，已经是比较少了，看的快的话，1个小时就能浏览一遍了，我觉得此文档还是有一些作用的；文档中由于时间所限，出现了很多记录不够全面细致的部分，需要具体参考书本的语境与例子加深理解；而且文档的错别字也很多，只能后期校验了。其实我是java开发程序员，平时只用到很少的shell，本来不想在shell上花费太多时间，同时期读的书《JavaScript高级程序设计》与《深入理解JAVA虚拟机》都是大部头，我读的非常辛苦，速度又慢，枯燥无聊之际，读一读本书，又简单，成就感又大，不知不觉就读了很多内容，最后索性决定就读完了吧。最后，为了防止篡改，我提供了文档的PDF版本。

联系方式：[zyxfox@foxmail.com](mailto:zyxfox@foxmail.com)

QQ：2424078348

微信：zyx163766

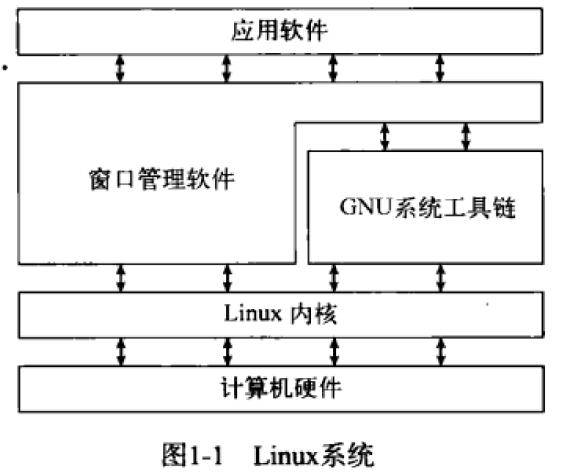
第一部分 linux命令行

第一部分分为9章，介绍了基本的bash shell命令，linux的环境变量与文件系统以及编辑器软件的使用等。

# 初识linux shell

## 1.1 什么是linux

Linux分为linux内核、GNU工具组件、图形化桌面环境以及应用软件。



系统核心是内核，控制计算机的所有硬件与软件；主要职责是系统内存管理、软件程序管理、硬件设备管理、文件系统管理。内核管理虚拟内存与物理内存包括页面替换等，cat /proc/meminfo包含了虚拟内存的状态；共享内存页面是所有进程都可以写入读出的，ipcs命令查看共享内存页面软件程序是进程，内核从虚拟内存中加载init进程，然后init进程再加载其他的进程，自启动进程列表在/etc/inittab文件夹中，ubuntu中为/etc/init.d目录，目录里含有启动或者停止进程的脚本；这些脚本通过/etc/rcX.d目录下的入口启动（符号链接），X代表运行级别。Linux系统有5个运行级别：1.单用户模式，基本的系统进程启动，也会启动唯一一个控制台终端进程；标准启动运行级别是3，大多数的基础软件，5.桌面图形系统；PS命令查看运行中的进程；硬件设备管理是通过驱动程序代码，1.编译进内核的驱动2.可插入内核的驱动，在内核的模块化之后，2方式流行起来，设备分为3类文件：字符型设备文件（只能处理一个字符），块设备文件（处理一个块，比如硬盘），网络设备文件（数据包），每个设备有一个节点，节点含有设备的标示--数值对（主设备号（类别编号），此设备号（具体编号））。Linux支持大多数的文件系统。

内核提供了基础的管理，但是计算机的应用需要特殊的工具。GNU组织开发了完整的unix内核上的使用工具，2者结合才能称为linux操作系统。核心的工具称为coreutils软件包，含有处理文件的工具，操作文本的工具，管理进程的工具。还有一个比较重要的工具是shell，交互式工具，提供启动程序、管理文件系统以及进程的途径，核心是命令提示符，将文本命令解释成内核命令执行。Shell有内置命令，也可以输入程序名，然后内核启动它；将shell命令放在文件中就是shell脚本，只要是命令行中的命令都可以放在脚本中，shell有很多种，通用的就是bash shell。

## 1.2 linux发行版

完整的linux系统包称为发行版，具有专业性：完整的核心Linux发行版，专业发行版、LiveCD测试发行版，3种版本；1.核心Linux发行版，包含所有软件包；2.包含特定领域软件；3.包含最小集合。

# 第二章 走进shell

## 2.1 终端模拟

CLI命令行接口，文本界面。终端字符集：ASCII，ISO-8859-1（Latin-1）(包含欧洲字母字符)，控制字符执行非打印操作，回车（回到行首）、换行（下一行）,break键 给主机发送一串0 中断正在执行的程序；return，回车键，表明命令结束；删除键，分为delete与backspace。

## 2.2 terminfo数据库

Terminfo是所有模拟终端文件的集合，文件定义了模拟终端的特性，文件名与模拟终端名字想通，是二进制文件；$TREM环境变量定义此时的命令行使用哪个模拟终端文件来决定控制码。

## 2.3 Linux控制台

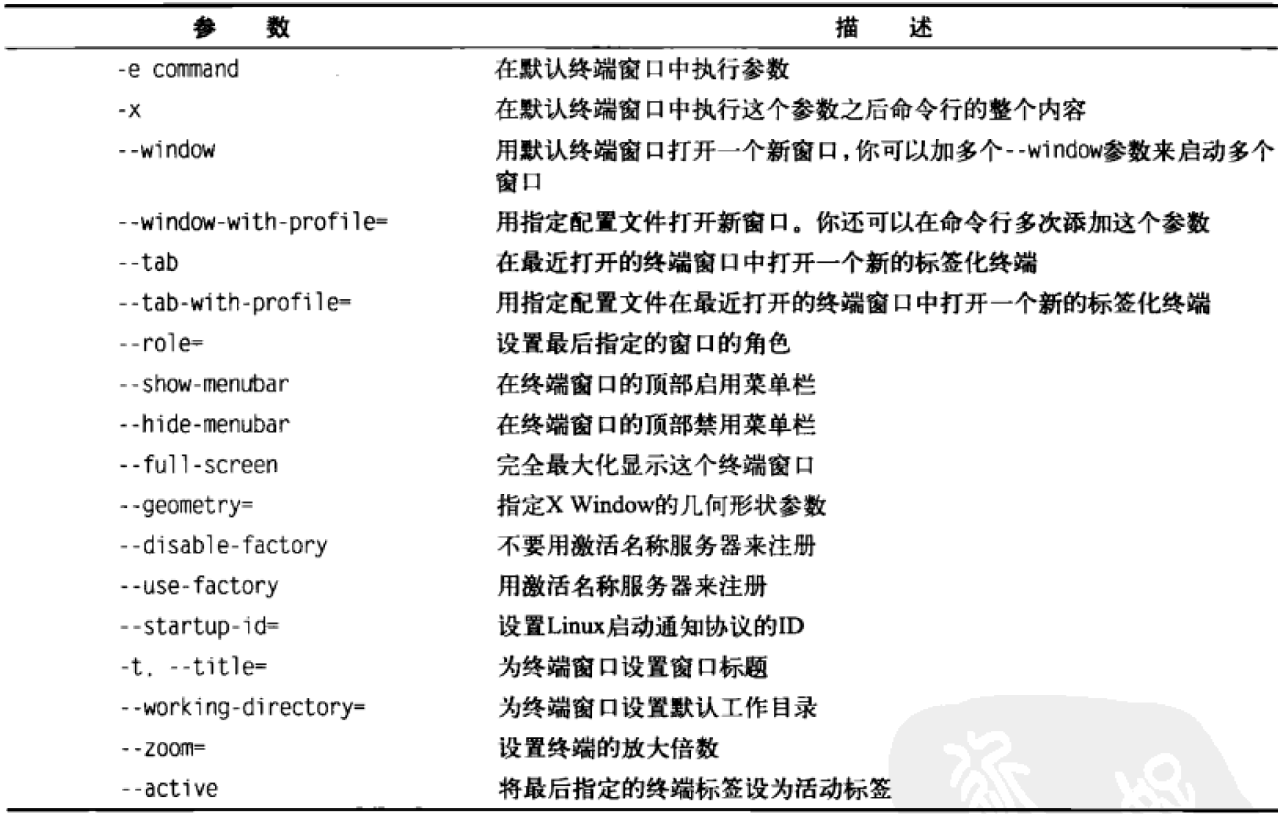
虚拟控制台在内存中，跟随linux启动，ctrl+alt+Fn启动控制台。

## 2.4 xterm终端

## 2.5 konsole终端

## 2.6 GNOME终端

Gnome终端的参数。



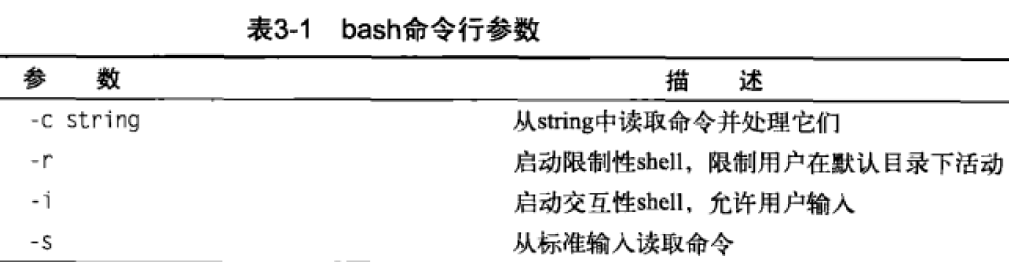
# 第三章 基本的bash shell命令

## 3.1 启动shell

/etc/passwd文件存储了用户数据。



7个字段：用户名（占位符，存储在其他文件），UID，GID，全名，主目录，默认shell；



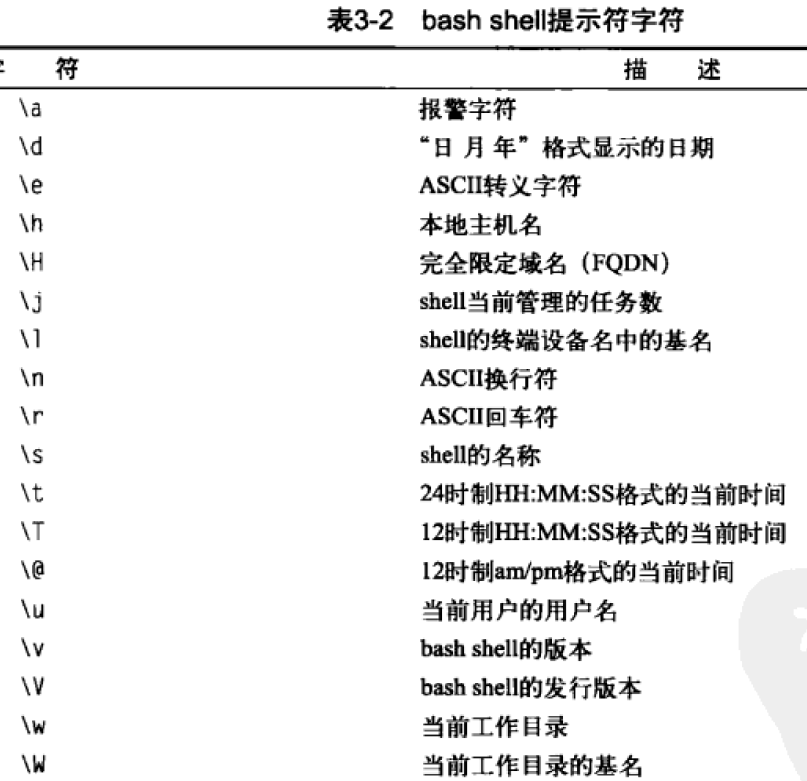
Shell启动时加载.bashrc配置文件。

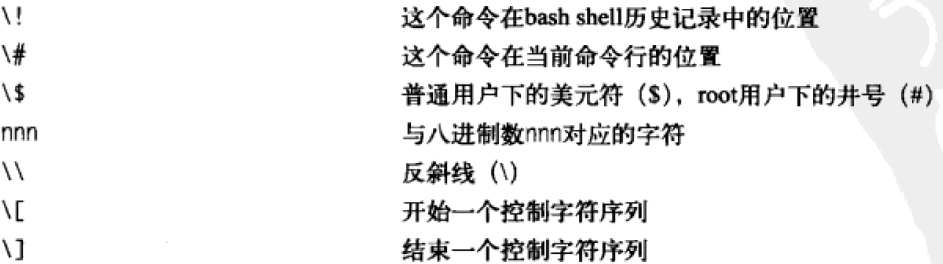
## 3.2 shell提示符

$提示用户输入，zyx@zyx-vm:~$依次是用户名，虚拟控制台编号，当前目录，提示符；PS1控制默认命令行提示符的格式，PS2控制后续命令行提示符的格式，一般是>，显示还需要后续的命令信息；shell提示符是一种字符标记的形式。



标记字符如下：





标记字符都是\开始的，像@符属于普通文本。

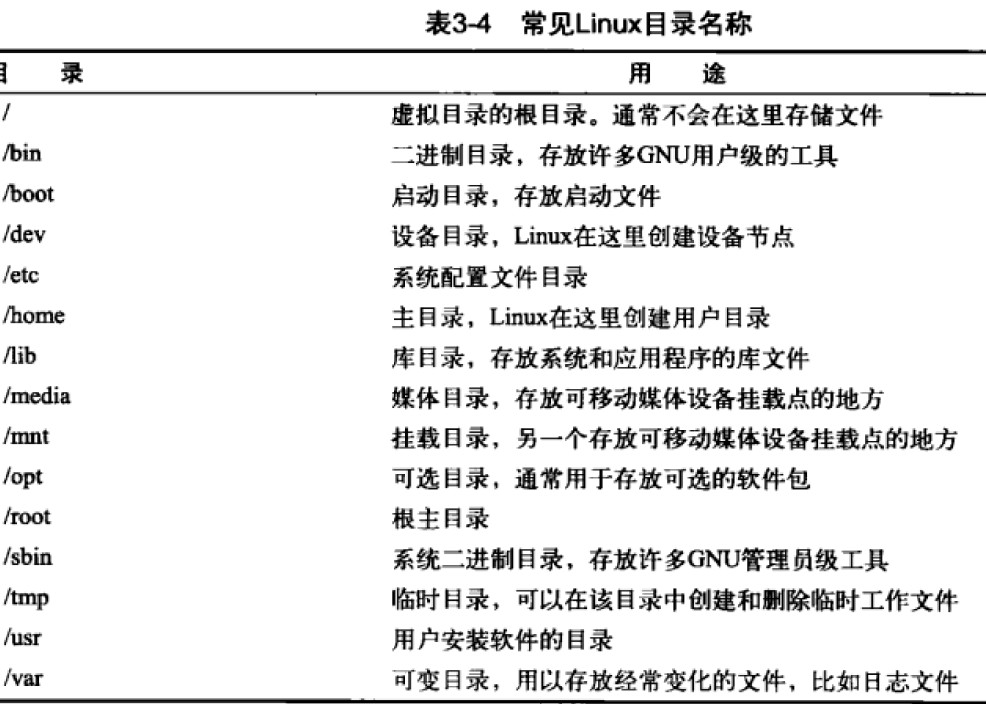
## 3.3 bash手册

Man+命令 显示命令帮助手册的内容，手册的格式如下：



Man bash；显示所有命令的帮助。

## 3.4 浏览文件系统

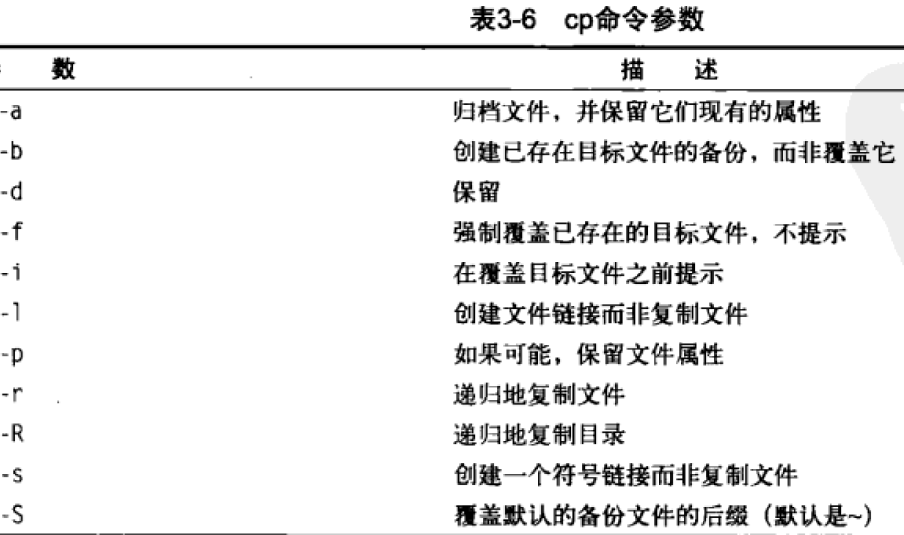


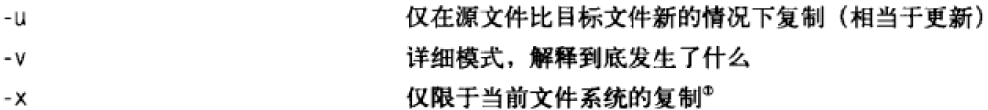
## 3.5 文件与目录列表

Ls显示当前目录下的文件与目录，LS\_COLORS控制文件颜色；-F区分件与目录；-a显示隐藏文件目录；-R，回溯遍历输出所有；-l 输出详细信息；第一行显示目录包含的块的总数，详细信息有：文件类型、权限、硬链接总数、属主用户名、属组的组名、大小、修改时间、名字；参数包含2种模式：-单字母与—单词，一般作用是想同的，比如：-a与--all；还可以过滤，比如ls –l aaa，精确显示aaa文件，aaa还可以是通配符的形式，\*代表0个或者多个字符，?代表一个字符。

## 3.6 处理文件

Touch创建空文件，cp复制文件或者创建新的文件；cp命令的参数；





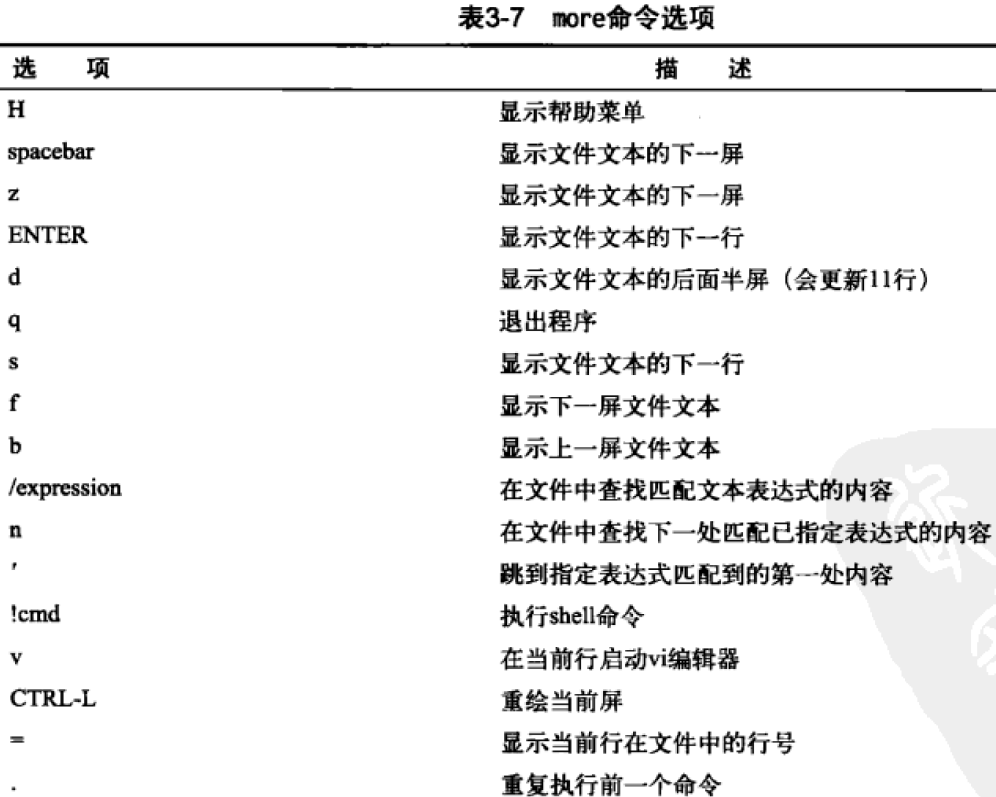
链接指的是指向同一个物理文件的占位符，分符号链接与硬链接；硬链接会创建一个独立文件，里面包含了源文件的信息及位置，-l创建硬链接，-s创建软连接，ln命令也能创建链接，-l硬链接，-s软连接；cp命令复制链接时复制源文件。mv命令移动文件；节点号及时间戳不变，移动软连接指向的文件会导致软连接失效。删除命令rm，删除硬链接要看节点数，但是软连接会失效。

## 3.7 处理目录

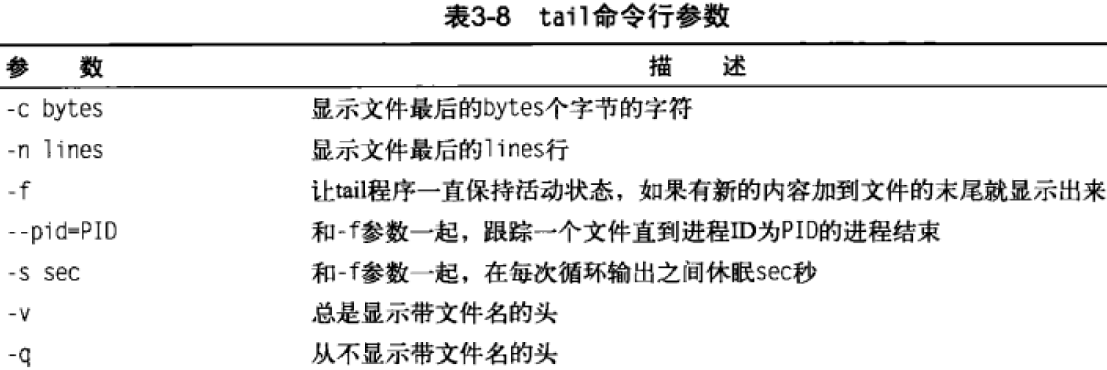
Mkdir创建目录，rmdir删除目录，删除空目录，rm –r删除目录。

## 3.8 查看文件内容

Stat命令可以查看文件的状态信息，file命令查看文件类型信息；cat显示文本内容，-n加上行号；-b给有内容的行加行号，-s压缩空白行，-T去掉制表符；more命令是cat形式的分页命令，打开文件进入more命令界面，可以按得键如下（命令参数）如下：



Less命令与more差不多，但并不一次读取文件进内存。Tail命令显示末尾文本，参数：

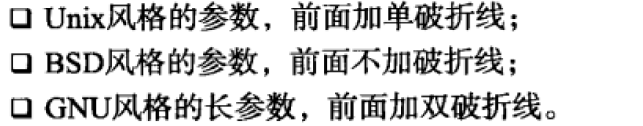


与此相对，head命令显示头的。

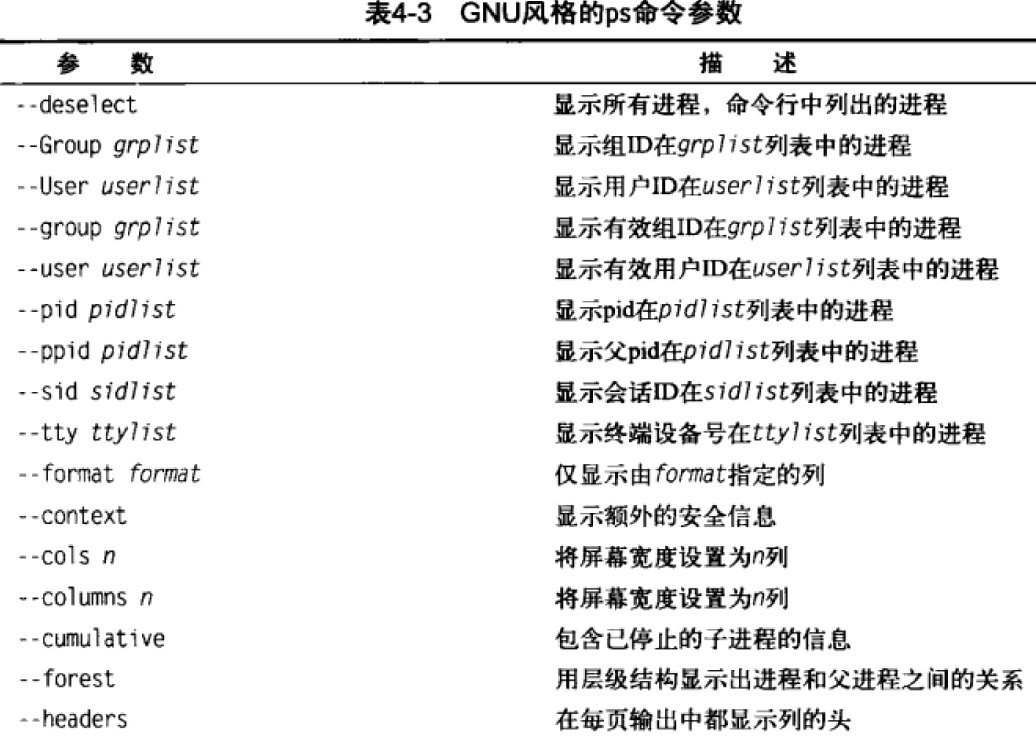
# 第四章 更多的bash shell命令

## 4.1 监测程序

Ps命令查看进程，ps显示当前控制台下的属于当前用户的运行进程，显示PID，控制台，cpu时间，名字；参数形式：

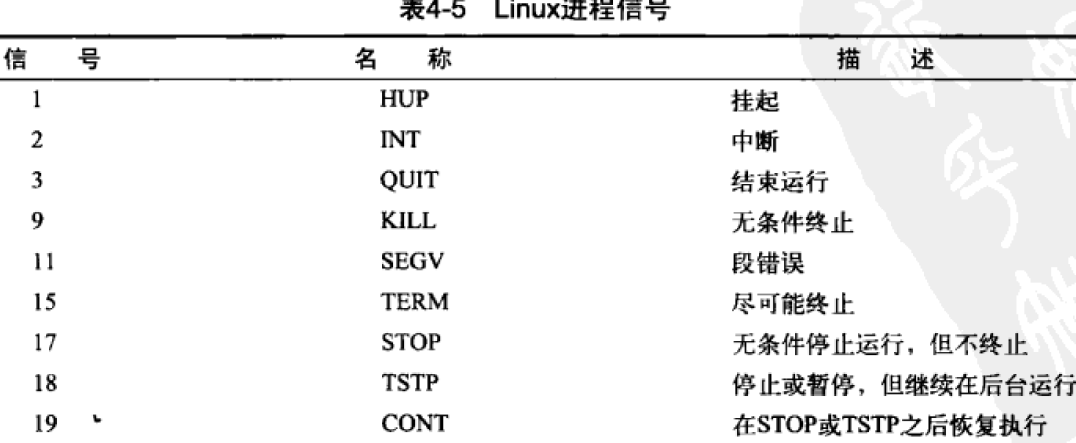


GNU形式的ps命令参数：





可以跟unix与BSD风格的命令混用；top命令实时显示进程信息，top有非常多的交互命令。结束进程kill与killall；kill命令与PID号对应，默认发送TREM信号，-s参数可以更改发送的信号；killall命令可以接进程名字；能使用通配符；



## 4.2 监测磁盘空间

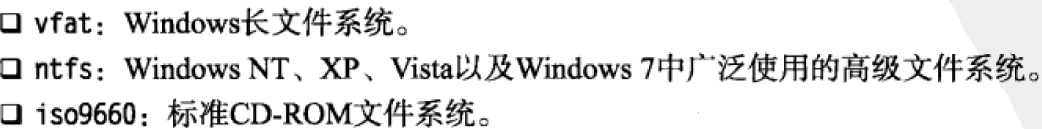
使用之前先挂载，mount命令挂载；是吧四部分信息：设备文件名、挂载的虚拟目录、文件系统类型、访问状态。

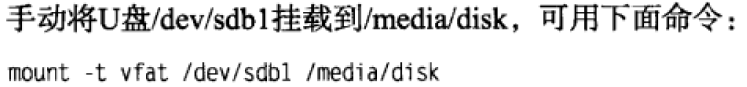


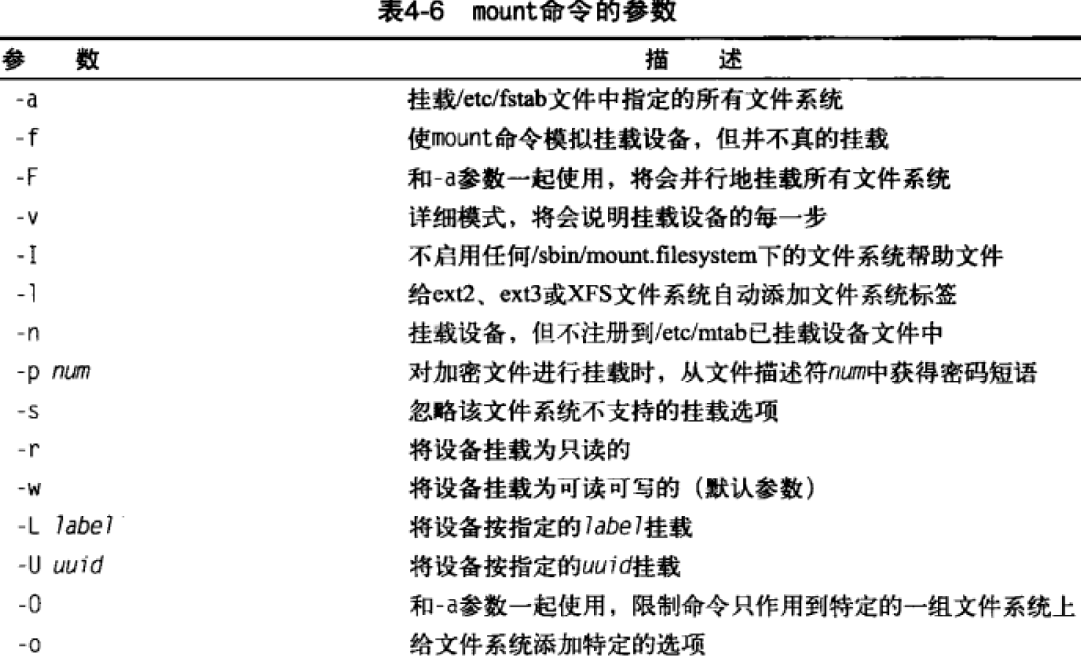
挂载命令：

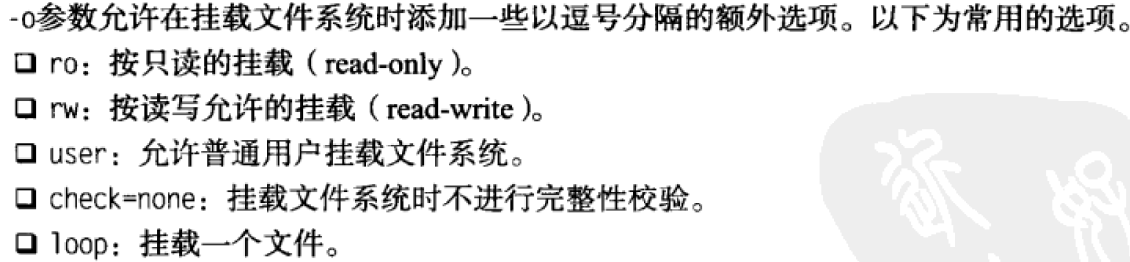


Type指的是文件系统。



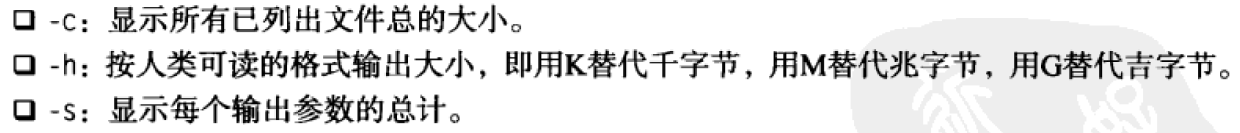






Umount卸载设备，umount [directory| device] 都可以卸载设备。

Df命令查看磁盘空间情况，显示的信息：设备的文件系统、容量（1024字节），已用，所占比例，挂载点；-h参数会指定显示用户友好的显示形式，du显示当前目录下所有文件目录占用磁盘情况。

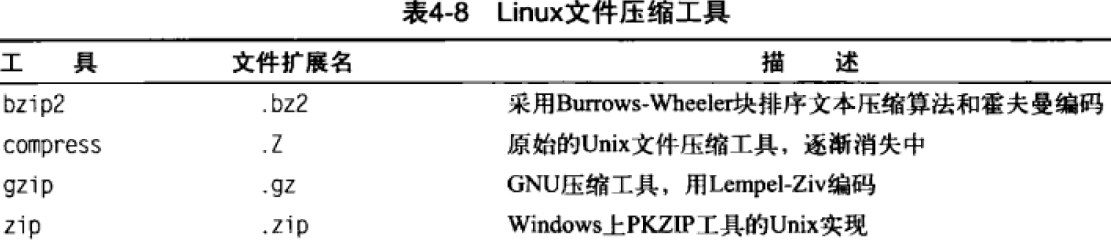


## 4.3 处理数据文件

Sort显示文本内容并按默认规则排序；排序有许多参数，必须配合使用；grep命令查找操作。



Grep有许多参数，还支持正则表达式。



归档文件的创建使用tar命令。

# 第五章 使用linux环境变量

## 5.1 什么是环境变量

环境存储shell会话以及工作环境的相关信息，环境变量分为全局与局部；全局变量对所有shell会话可见，使用大写字母表示，printenv打印所有全局变量。使用echo输出$环境变量的内容，局部变量是当前进程定义的变量；set会显示所有的变量。

## 5.2 设置环境变量

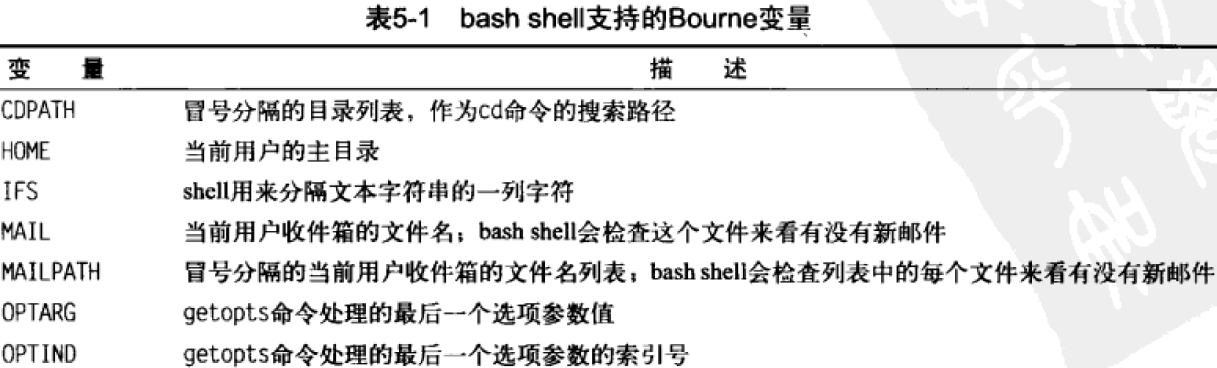
局部变量可以是任何变量，直接复制比如test=ddddd，使用用$test，设置全局变量的方式是export导出局部变量，导出局部变量的时候不需要$符号；

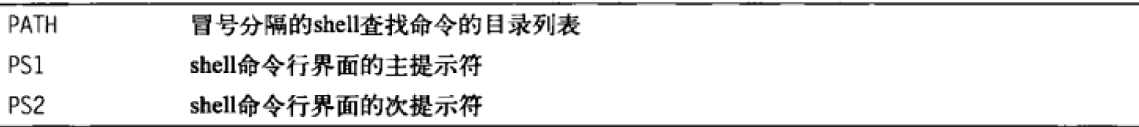
## 5.3 删除环境变量

删除变量使用unset，但是删除全局变量只能在本进程中删除，全局删不了。

## 5.4 默认shell环境变量

这些变量定义了系统环境。





除了这些通用的变量，shell还有很多自有的变量，可以看书。

## 5.5 设置PATH环境变量

PATH变量定义了命令的搜索路径，可以直接添加环境变量。PATH可以直接修改。

## 5.6 定位系统环境变量

Shell启动时，会在几个默认文件中寻找命令，这些文件称为启动文件，根据启动shell的方式的不同而不同，1.作为登陆shell启动时，会依次在/etc/profile（主启动文件，都有），$HOME/.bash\_profile，$HOME/.bash\_login与$HOME/.profile（这3个定义用户专有的启动环境）中寻找；2.交互式shell，不去找/etc/profile，而是去找home目录下的.bashrc文件，里面会去找/etc/bashrc文件执行；3.非交互式shell，执行脚本的shell，使用BASH\_ENV环境变量查看运行环境。

## 5.7 可变数组

环境变量可以作为数组使用，可以单个值也可以整体使用。



此时echo $mytest是不能显示整个数组的，需要加下标使用，[\*]的话使用整个数组，unset删除数组，直接删除，删除单个值时要注意，值还是在的，只是不显示了。

## 5.8 使用命令别名

命令别名就是把命令与参数加在一起起个名字，alias –p查看本shell进程的定义的别名；创建别名比如：alias li=’ls -il’，由于是局部变量，它只在本进程中有效，可以在.bashrc中设置，这样每个交互式shell都可以使用。

# 第六章 理解Linux文件权限

## 6.1 linux的安全性

Linux创建用户分配UID，通过UID管理权限。/etc/passwd文件关联用户与UID信息；里面包含很多专门用来执行后台服务程序的系统账户，避免了使用root账户执行程序的安全性问题，密码存储在其他文件，禁止直接修改这个文件，通过工具修改；/etc/shadow里面含有跟用户名对应的密码信息，useradd命令添加新用户，-D可以查看系统创建用户的默认配置比如用户组、HOME目录、默认shell等；同时还包括创建HOME目录是加入里面内容的模板，里面一般包含启动shell的启动文件；可以使用很多命令行参数指定创建用户的行为。Userdel删除用户，删除/etc/passwd中的信息，-r参数删除用户文件；修改用户信息有很多工具比如：usermod(修改/ect/passwd中用户信息的大多数字段，加参数修改大多数字段)，passwd（更改密码，没提供用户名就改自己），chpasswd（更新密码，可以使用文件重定向批量更改），chsh（改默认登陆shell）、chfn（修改备注信息）、chage管理用户的有效期（）。

## 6.2 使用Linux组

组概念方便了文件共享，会有GID，组名；/etc/group里是组信息，包含的信息有：组名、组密码、GID、属于该组的用户列表；groupadd创建新租，usermod添加成员；修改组使用groupmod命令。

## 6.3 理解文件权限

文件权限符就是r、w、x；默认当前用户创建文件的文件权限符是按umask决定的，比如0022；第一个数是粘着位，022代表掩码，实际的文件权限是全量数-umask得到，都是8进制计数的，文件是666，目录是777；

## 6.4 改变安全性设置

文件或者目录的安全性设置可以通过工具修改，chmod命令，可以通过2中方式，比如：3个八进制数字直接修改，或者符号形式修改权限,

形式为：[ugoa] [+-=][rwxXstugo],u是当前用户，g是组权限，o是其他用户权限，a是所有；+加权限；chown改变属主可以同时改变属组，chgrp改变属组。

## 6.5 共享文件

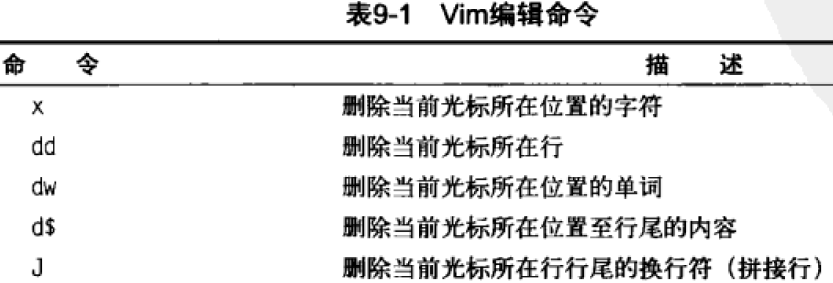
这个使用前面的粘着位，SUID，SGID等。可以仔细看书。

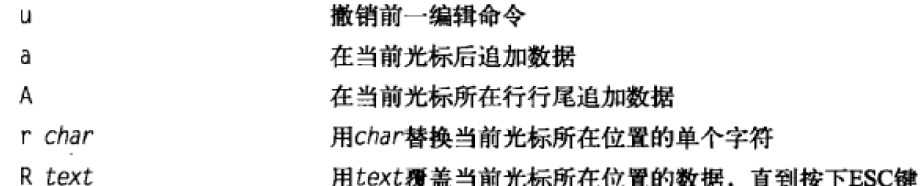
# 第七章 管理文件系统

Linux使用的是类unix系统的ext文件系统（索引节点），日志文件系统。

# 第九章 使用编辑器

Vim编辑器在内存缓冲区编辑数据，分为普通模式与插入模式，普通模式下的按键解释为命令，有很多命令（参数）；还有可以交互的命令行模式，：键下进入，在普通模式下操作缓冲区内容的命令：





可以在命令前加数字表示命令执行多少次。p命令粘贴文件，删除就是剪切命令，y是复制名令。v命令高亮文本。/开启字符查找，输入字符串，回车查找；在接着/回车，查找下一个，：s/old/new/替换；还有许多的选项。

# 第十章 构建基本脚本

## 10.1 使用多个命令

使用分号；连接命令，不超过255个字符。

## 10.2 创建shell脚本文件

#号是注释，放在第一行是例外，！告诉系统使用哪个shell执行，编写完脚本后要设置路径以及可执行的权限信息等。

## 10.3 显示消息

Echo 输出信息。

## 10.4 使用变量

变量使用美元符号定义，输出美元符号前面加\进行转义处理；赋值的时候不需要$，但是使用的时候要加$，否则当普通字符串处理，中间不能有空格；括在反引号·中的语句会当做命令来执行。

## 10.5 重定向输入输出

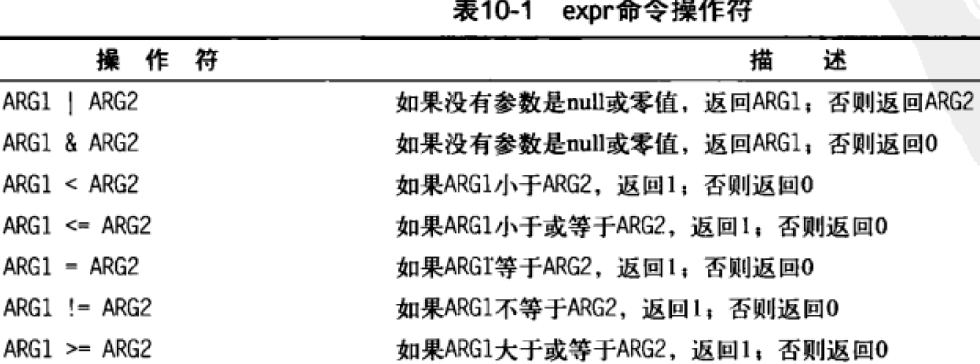
重定向使用>，追加使用>>符号。输入重定向使用<号；将内容输入到命令；<<内联输入重定向，可以在命令行指定输入的数据，EOF开始，EOF结束。

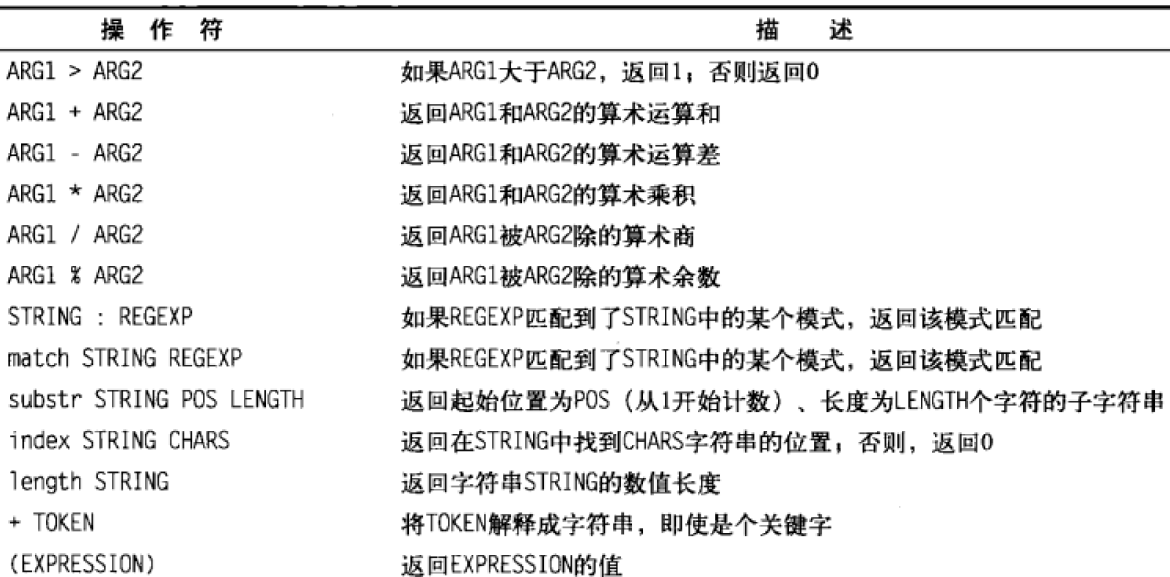
## 10.6 管道

管道使用| 符号，将2条命令连接起来运行，常用在结果分页。

## 10.7 执行数学运算

1.使用expr运算符，执行后面的参数表达式；为了获得结果通常放在``内；expr命令的操作符号如下：





Expr后面的参数符号如果在shell中有别的用途的话，需要进行转义；确保传入expr都是字符。2.使用$[]形式计算数学表达式，在shell脚本中也没问题，但是这个只支持整数运算。浮点数计算采用bash计算器，简称bc，bc能够识别数字、变量、注释、表达式、编程语句与函数。Scale控制结果的小数位数；在脚本中使用bc的用法是：var=`echo “options:expression” | bc`，options部分设置bc的参数，比如scale，可以在expression部分使用脚本的变量，还可以使用输入重定向，内联重定向：



## 10.8 退出脚本

每个命令都有退出状态码，结束时传给shell，在0-255之间，使用$?查看，可以使用exit自定义退出码，超出范围取模运算。

# 第十一章 使用结构化命令

## 11.1 if-then 语句

If语句的形式是：

If command

Then

Command

Fi或者是if command ; then

Command

Fi

条件测试里只检查这个命令的退出码是否是0，正确执行了就为真。

## 11.2 if then else 语句

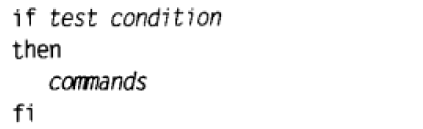


## 11.3 嵌套if



## 11.4 test命令

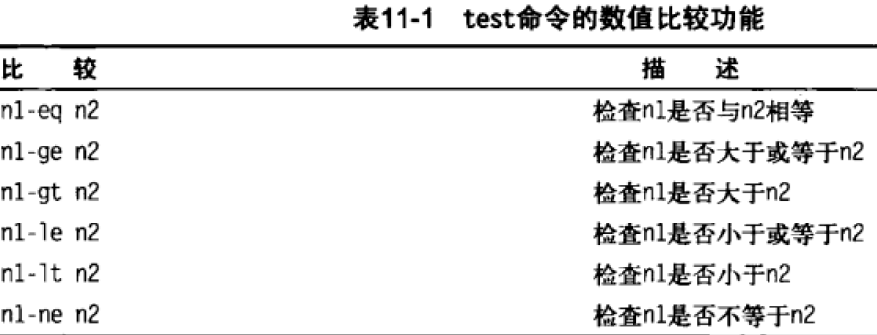
在if中加入test命令可以向普通语言的if语句那样工作，contion条件成立，则test返回0，为真，不成立，返回1；形式：



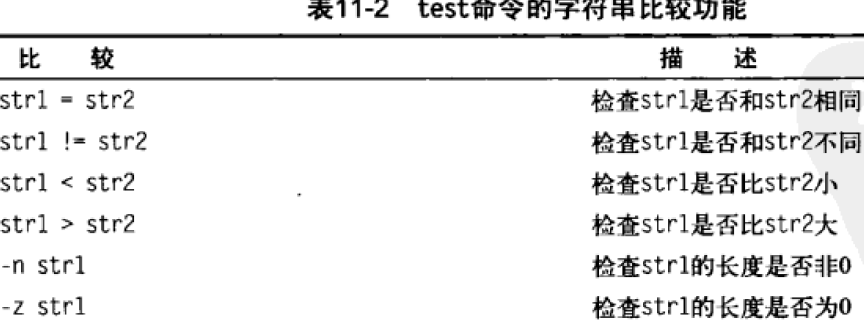
或者是，跟test命令的效果是一样的：



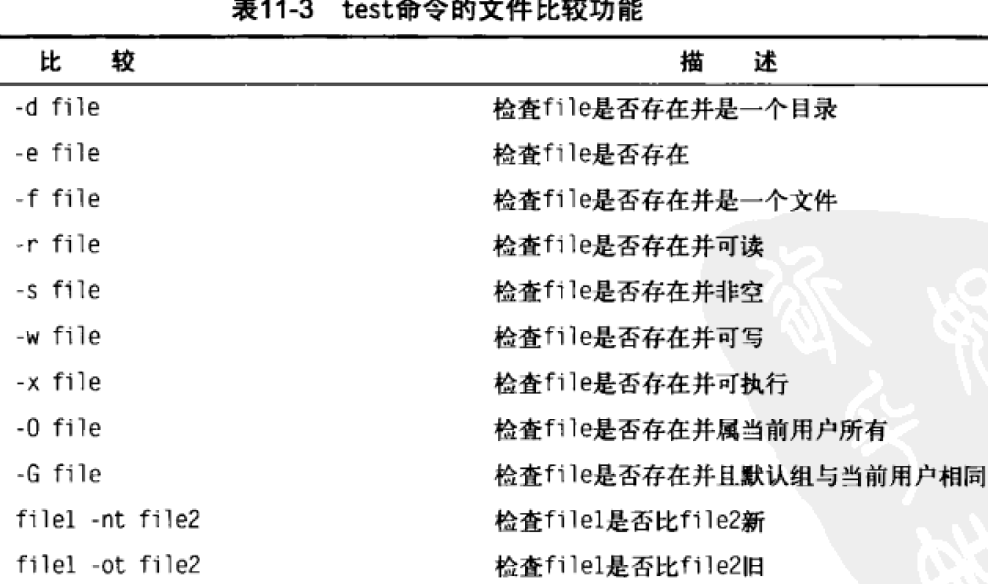
Test命令判断3类条件：数值比较、字符串比较、文件比较。



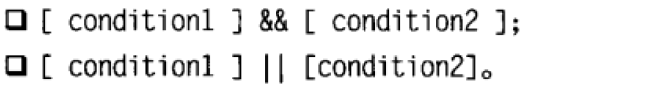
Bash shell能处理的数只有整数，如果浮点数存储进变量作为字符串显示没问题，在命令与函数中使用是不行的，测试条件中也不行。



字符串比较会将标点与大写的情况也考虑进去。字符串比较时>与<要进行转义处理，否则当成重定向；比较顺序与sort命令可能具有不同，-n与-z会检测变量里是否有内容，一般在使用变量的时候都要进行检测。未定义的变量-z检测都是空。

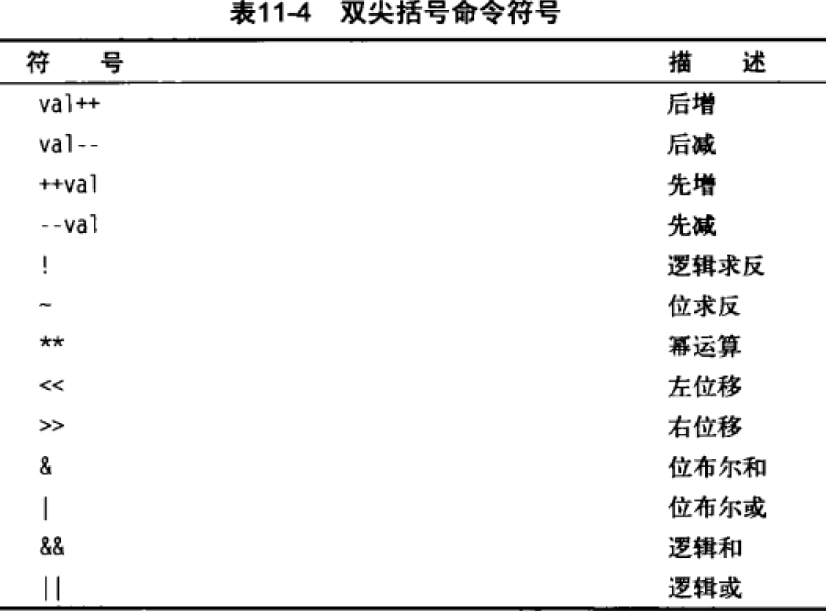


## 11.5 符合条件测试



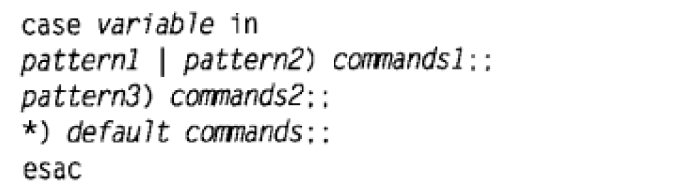
## 11.6 if-then的高级特性

用于数学表达式的双肩括号，使用形式是((expression))，expression可以使用任意的数学表达式，判断结果的真假；除了支持test的运算符，还支持：



里面不需要对符号进行转义处理。双方括号比较字符串，除了test的功能外还支持模式匹配，形式[[]]。

## 11.7 case



# 第十二章 更多的结构化命令

## 12.1 for命令

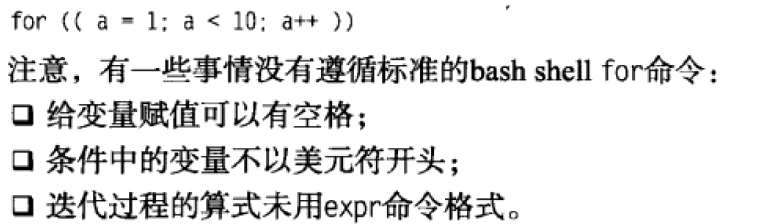


list可以是空格隔开的一系列的值，var在循环结束后还在，是最后一次迭代的值。如果列表中有特殊的符号，可以用转义，也可以用双引号；此时值两边的双引号不会被当成值，不如值中有空格的情况。此时list也可以是单个变量，变量里的值默认按空格分开，分别付给var，反引号可以包含任何能产生输出的命令，此时就可以当成list，IFS这个环境变量定义了系统默认的数据分隔符，默认包括空格、制表符以及换行符；可以更改它来达到自己的目的，前面改了，后面还可能以为是默认的，所以在用完后，要马上恢复原来的数值。

定义多个分隔符。还可以使用通配符的方式读取整个目录，比如/home/zyx/\*遍历此目录下的所有文件与目录，也可以在list部分加入特定的目录或者文件或者其他通配符路径，也是一样的。

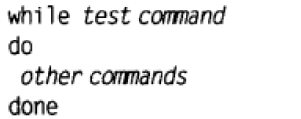
## 12.2 C语言风格的for命令





## 12.3 while命令

While命令式if与for的结合体。



Test命令退出码为0时循环，不为0的时候退出循环。

## 12.4 until命令

Unti命令与while命令相反，当test命令的退出码非0时执行循环，当为0时退出循环。

## 12.5 嵌套循环

## 12.6 循环处理文件数据

使用嵌套循环、修改IFS环境变量。

## 12. 7控制循环

Break命令跳出循环，break一般跳出外层循环，但是也可以跳出嵌套循环，break n，n代表跳出几层循环，continue也可以指定忽略的循环级数。

## 12.8 处理循环的输出

循环的结果可以重定向一个文件，done后接>>，或者通过管道传输给另外的命令来处理。

# 第十三章 处理用户输入

## 13.1 命令行参数

从命令行想脚本传递参数，通过$n直接获取参数，$0是运行的程序名，剩余的为参数，shell用空格分隔参数，如果参数包含空格，那么用引号包起来，$0的参数是完整的脚本路径，使用basename命令获得程序名，这样可以在脚本中根据程序名来决定具体的操作，name=`basename $0`，假如传入的参数比脚本中使用的少，会发生错误，所以可以在脚本中判断使用的参数是否有数据。[-n]。

## 13.2 特殊参数变量

$#记录命令行参数的个数，$#代表了最后一个参数的位置，${!#}获取最后一个参数。花括号中不能使用$，所以改成了$，还有$@与$\*都是活的所有参数，但是$\*将所有参数拼接成单个字符串，$@将参数分隔开。

## 13.3 移动变量

Shift命令，将$2移到$1,$3移到$2，依次减一，不能移到$0，移除的丢弃。也可以指定移动多少位。

## 13.4 处理选项

选项就是命令带有的可选项，带有单破折线的参数，也是可以处理的，识别选项与参数的方法之一是双破折线--；还有一种情况是，选项也需要参数，就是选项与参数是混合的，需要在选项处做相应的参数处理，然后都移除去。getopt命令是一个解析命令行参数的重要的工具，它可以将任意形式的命令行参数转换为合理的形式。



Options是getopt命令的选项，optstring代表了所有选项。在选项后面需要参数的加冒号：。Parameters为参数列表。



它还将-cd解析成了分开的形式；还插入了双破折号，加入optstring中缺少后面出现的选项，那么会报错，加入-q选项就不会报错了。可以使用getopt生成的格式化的选项参数替换原来的选项参数，使用set命令，set命令的选项种的双破折线--，会将命令行参数替换为set命令破折线之后的命令行的值。

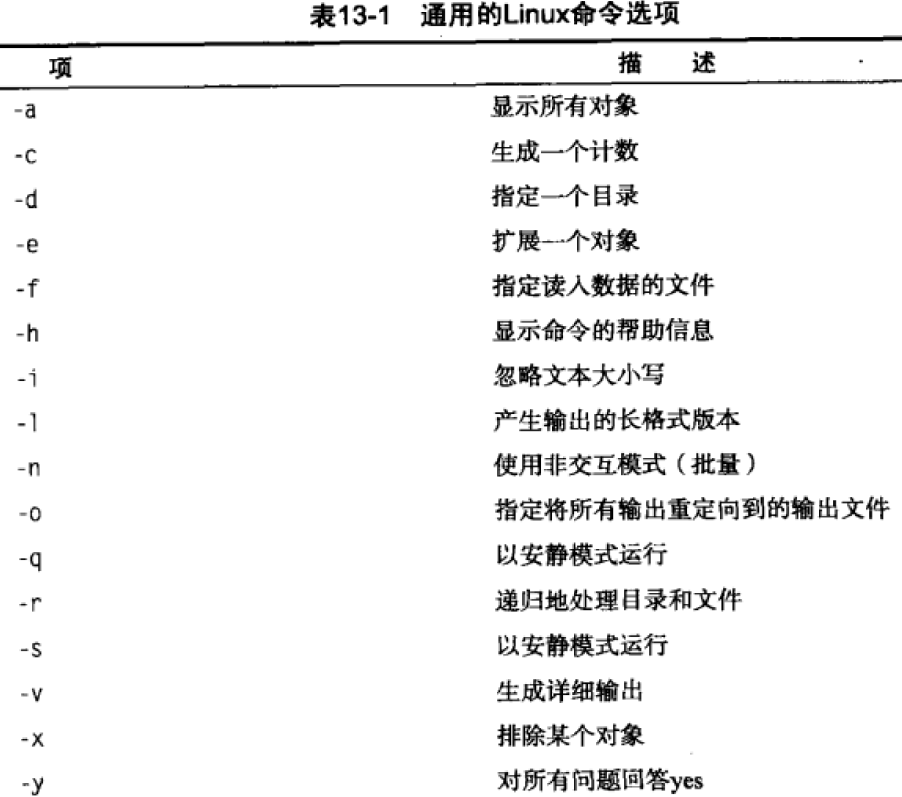


但是getopt命令处理带空格的参数会解析成2个。所以使用getopts命令，getopts命令会根据optstring中的字母循环的获取选项。



## 13.5 将选项标准化

通用的命令行选项的规定。



## 13.6获得用户输入

Read命令接受标准从标准输入或另一个文件描述符的输入，-p 选项允许直接在命令行read后接输入提示字符串，可以在一行内输入多个变量，输入的数据按空格分给每个变量，多出来的数据一次性分给最后一个变量，如果没有指定变量，数据会放入REPLY环境变量中，read会一直等待用户输入，-t指定等待时间，时间到没有输入返回一个非零退出码。Read命令也能指定选项输入指定个数的字符，然后退出；-s，使用密码输入；read可以读取文件，每次读取一行，读完后返回一个非零的退出状态码。

# 第十四章 呈现数据

## 14.1 理解输入与输出

Linux将每个对象都当成文件处理，并用文件描述符表示文件对象，它是一个非负整数。



输入是标准输入就是键盘，输出是标准输出是显示器，STEER也是定向到显示器，重定向改变了文件描述符指向的对象，错误需要写到日志里，所以标准错误输出需要重定向；重定向错误使用2> filename的形式，2跟重定向符号紧邻，因为2是标准错误的描述符；出现错误只会输出到文件；如果要重定向标准输出，就加上1的重定向；&>直接重定向2个；错误消息比标准输出的优先级高，优先输出。

## 14.2 在脚本中重定向输出

人为指定应该输出到标准输出的数据输出到标准错误，使用 >&2重定向符号。



这样做还是输出到了一起，可以在运行脚本时重定向标准错误，那么错误就与标准输出分离了。



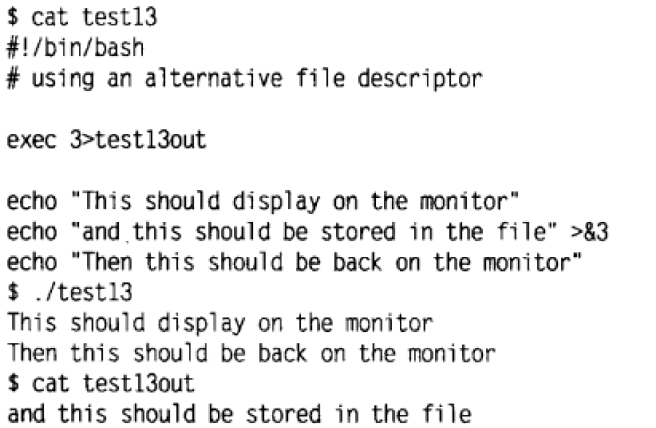
如果不使用命令行的形式，可以在在脚本中使用exec命令重定向，比如：

Exec 1>testout,exec 2>testerr。

## 14.3 重定向输入

## 14.4 创建自己的重定向

一个shell进程可以同时存有9个文件描述符，其余的是3-8，都可以在shell中使用；



追加而不是创建文件。

0-8的文件描述符都是可以互相重定向的，所以可以暂存显示器的标准输出，然后在改回来，比如exec 1>&3,3暂存了标准输出，exec 1>file,一些处理后，再exec 3>&1就可以了。可以把某些特性的输出保存到文件中，输入也可以这样做。文件描述符就可以作为输入也可以作为输出，比如exec 3<> file ,读入可能会正常，但是每一次的读取写入都会从上一次的指针指向处开始，可能会造成错误，退出脚本时，多余的文件描述符会关闭，假如中间就想关闭;



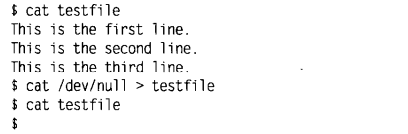
关闭了文件描述符重定向后，再次打开重定向的文件会重新创建文件。

## 14.5 列出的打开的文件描述符

Lsofhui列出当前所有的文件描述符情况，为了安全的考虑，加入参数获取当前进程的文件描述符，$$获取当前运行进程的PID，-p，允许指定PID。

## 14.6 组织命令输出

如果不想要脚本输出信息或者不想输出错误信息，可以重定向到/dev/null文件，数据会被马上丢弃，这个文件始终为空，将他作为输入文件时会清空文件里面的内容，比如如下命令：



## 14.7 创建临时文件

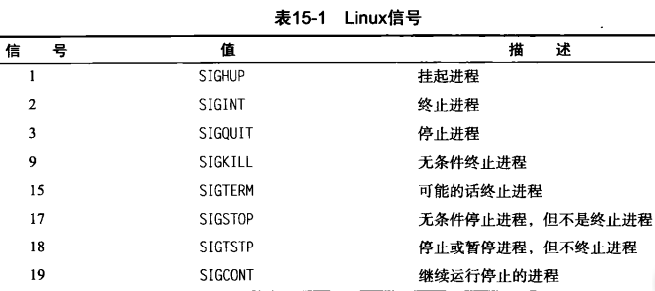
Linux中的/tmp为临时权限目录，每次系统启动时都会清空他，每个账户都有权读写，创建的文件不用管删除。自己创建的自己有权读写。Mktemp默认情况下在本目录创建临时文件，name.XXXXX，mktemp会用6个随机字符替换X，保证文件名不重复，-t指定在tmp目录下创建，返回全路径名。-d 指定创建临时目录，返回全目录名。

## 14.8 记录消息

如果想要消息同时输出到2个文件中，可以使用tee命令，可以将stdin的输入重定向输出到stdout与tee所指定的文件。，通常使用 管道管道的形式，标准形式是：Date | tee filename，默认每次覆盖，如果不覆盖，那么加入-a参数追加。

# 第十五章 控制脚本

## 15.1 处理信号



当bash shell收到停止的信号时，会传递给它所启动的所有进程或者脚本，但是脚本默认选择忽视这些信号，所以必须加入判断机制以避免错误。使用键盘组合键产生信号，比如：ctrl+C生成SIGINT信号，终止进程，暂停进程使用ctrl+Z，产生SIGTSTP信号。暂停进程还在内存中，[1]其中1为作业号，可以用ps命令查看暂停进程，trap命令可以捕捉信号，意思就是，捕捉到这些信号，在本脚本内决定如何处理，而不是让shell决定如何处理。简单格式就是：

Trap command signals。也能捕捉脚本退出的信号，只要把signals中写入EXIT就可以了。当退出时就会执行commands命令。Trap – signal,移除需要捕捉的信号，但是在这个命令执行前还会捕捉相应的命令。

## 15.2 以后台模式运行脚本

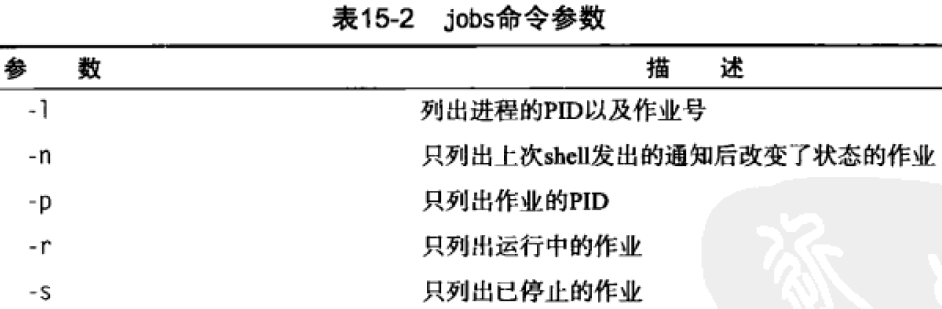
在命令后加&就是以后台模式运行脚本，看不到界面，不接触CLI；&将作业与shell分离开，在shell中键入命令 &；第一行会显示作业号与PID，结束时会显示作业状态。有输出的话会与当前命令的输出夹杂在一起。可以开启很多的后台运行进程，但是这些进程都绑定了当前的shell终端，如果shell退出，进程也会退出可能不会有提示。

## 15.3 在非控制台下运行脚本

如果想要终端退出时，后台脚本不退出，那么使用nohup命令，nohup command &；nohup会阻断SIGHUP信号，所以shell退出时，进程忽略了SIGHUP信号，但是这样会丢掉STDOUT，STDERR的链接，这个输出会以追加的方式重定向到nohup.out文件。

## 15.4 作业控制

Jobs命令查看当前命令行窗口运行的作业。



+未默认作业，-为第二个默认作业。可以重启后台与前台作业，重启前台作业会接管终端输入输出，bg 作业号，重启后台作业，fg 作业号，重启前台作业。

## 15.5 调整谦让度

优先级代表了进程获得CPU的时间，-19最高，19最低；默认是0，nice命令可以让进程获得比当前更低的优先级，renice命令也可以改变优先级，普通用户只能降低优先级运行，root用户可以提高优先级运行。

## 15.6 定时运行作业

At命令提交定时作业，提交到队列中，守护进程atq检查队列，并以后台模式运行作业，命令格式如下：



队列中有26个优先级。

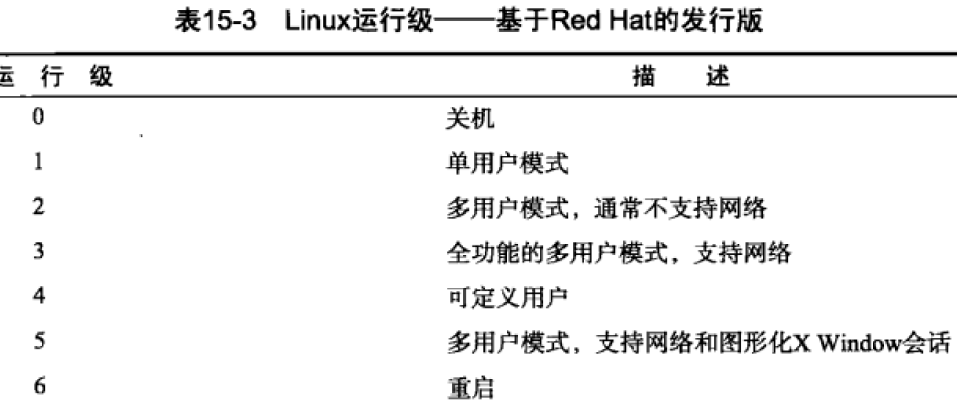
这些作业没有屏幕输出，默认将产生的消息写成email的形式发出，atq命令可以查看队列，atrm删除等待中的作业。也可以周期性的执行脚本，命令是cron，cron会在后台运行并检查cron时间表来获得计划执行的作业；cron时间表是一个文本文件，周期指定格式如下：



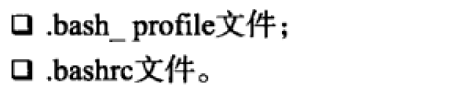
每个用户都可以有自己的cron时间表运行周期任务，crontab处理cron时间表，crontab –l列出表，-e，编辑表；还可以使用cron脚本目录（不需要精确的时间的周期任务），如果linux系统关闭了，错过的脚本不会执行，anacron可以解决这个问题，它会让错过偶的脚本尽快执行，它只会处理cron目录的程序。

## 15.7 启动时运行

Linux的开机过程，首先加载linux内核，然后启动linux system V init进程，它负责启动其他进程，它会读取/etc/inittab文件，这个文件中含有不同运行级别的不同的程序和脚本环境变量。



启动后，init进程会去寻找一个本地开机执行的脚本集合的文件，在Ubuntu中是：/etc/rc.local，可以向里面加入启动脚本的全路径，随新shell启动的脚本添加在。



如果所有用户都需要有随新shell启动的脚本，在/ect/bashrc里设置。

# 第十六章 创建函数

## 16.1 基本的脚本函数

定义函数有2种方式；function name{commands;}，name(){commands}。函数使用的方法是调用函数名，要在定义之后使用，同时名字要唯一，否则会被覆盖。

## 16.2 返回值

函数结束的视乎会有退出码，默认情况下是最后一条命令的退出码，使用$?获得，这种比较危险，因为不知道前面的命令是否成功执行。使用return命令，但是必须要小于256；也可以将输出保存在变量中，var=`function`。

## 16.3 在函数中使用变量

函数也就是可以重复使用的脚本，可以通过脚本的那种传参的方式传入参数，但是这里的参数环境变量与脚本的参数变量是不同的，不能直接使用。函数外声明的变量在函数内也是可以用的，这样破坏了函数的复用性，所以尽量不用这种变量，使用函数的局部变量，它是使用local声明的。

## 16.4 数组变量与函数

向函数传递数组变量会比较麻烦，因为传递数组名作为参数，只会传递第一个数据，想要传递数据只能将它分开作为参数传给函数，在函数内部再重组数组。返回数组值也是这样的，一次输出数组数值，回到脚本后再重组为数组。

## 16.5 函数递归

阶乘问题。

## 16.6 创建库

创建库是为了方便脚本间共享函数复用，不能直接包含，因为它会创建新的shell，并在里面执行，后面调用函数的时候会找不到命令，想要在本shell中执行函数，使用source命令，快捷方式是点号操作符。

## 16.7 在命令行上使用函数

函数可以直接在命令行上用的，因为一个shell会话本质上来说就是一个脚本，所以可以直接在命令上定义一个脚本，然后直接使用，在此shell会话中已存在函数的定义，这样还是比较麻烦，因为shell会话结束后，函数就不在了，可以将函数定义在shell启动的加载文件中，比如./bashrc中，更好的方式是在.bashrc文件中写加载函数库文件的代码；这样就能加入很多函数，shell以及子shell进程脚本内都可以直接使用这些函数。

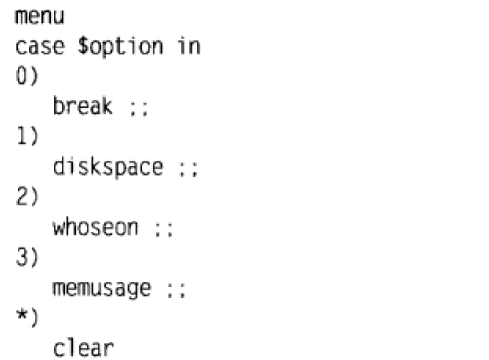
# 第十七章 图形化桌面上的脚本编程

## 17.1 创建文本菜单

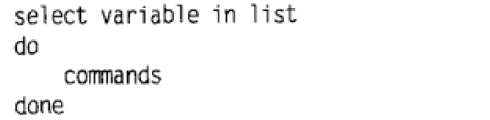
主要是case命令；clear命令。输出菜单：



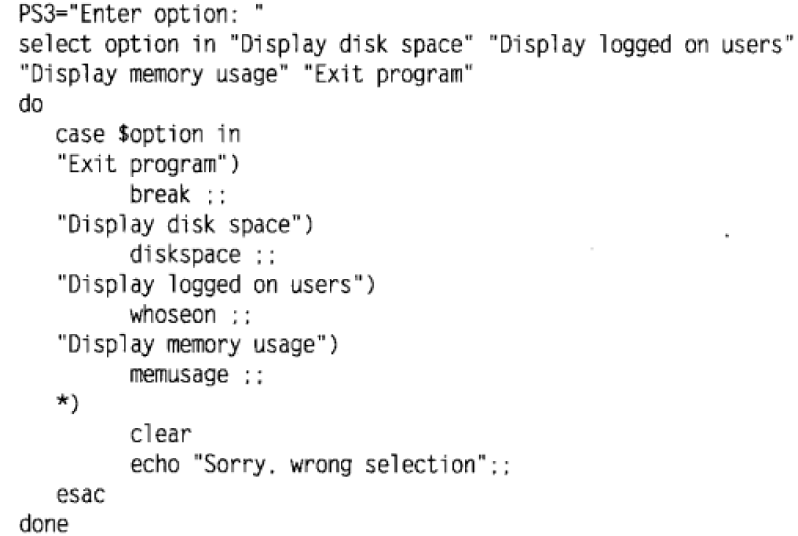
通常把他定义成函数，每次使用时都是在干净的屏幕上输出信息。根据输入信息选择调用的函数。



使用select语句创建菜单更方便。、

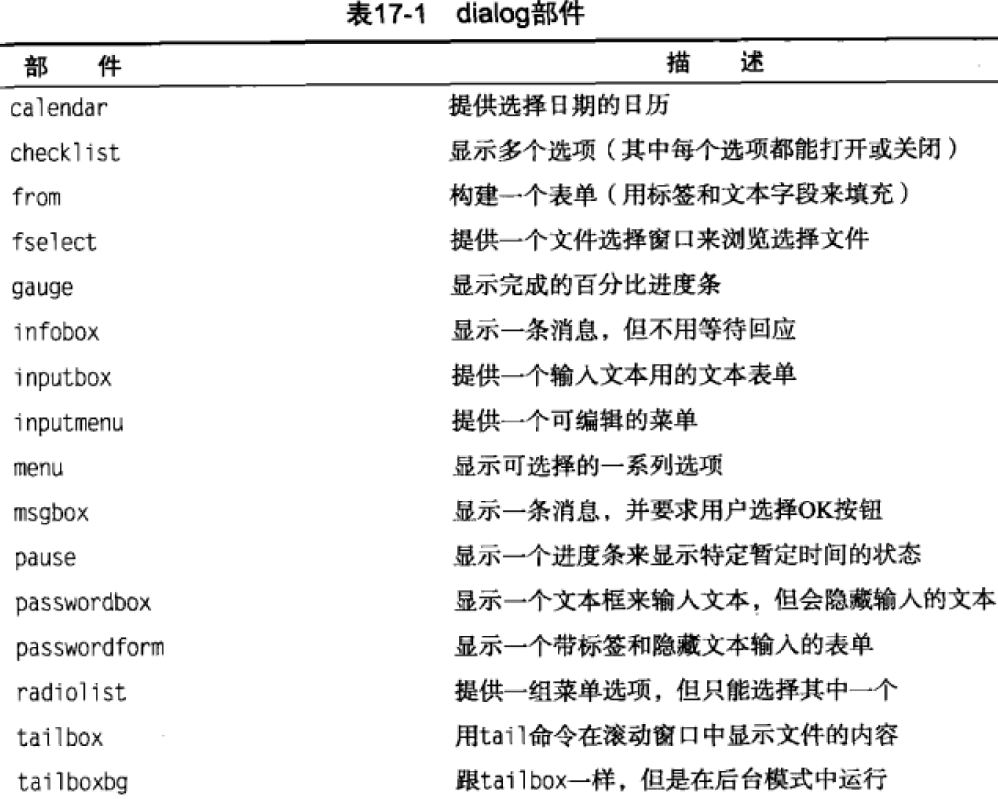


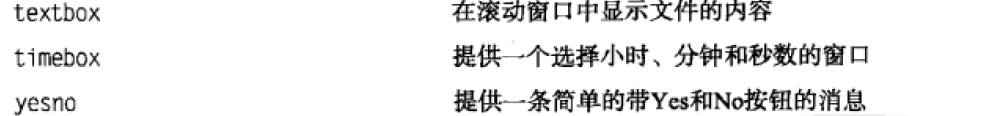
List是字符串列表，select自动对这些字符串编号显示，然后显示一个PS3定义的输入提示字符串，然后用户可以选择。



## 17.2 使用窗口

Dialog包，创建窗口对话框。根据参数生成相应的部件。





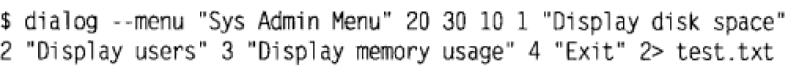


退出状态码决定了用户选择按个按钮，OK是0，no或者cancel为1；可以用$?查看，如果返回了任何数据，都会将数据发送到STDERR。





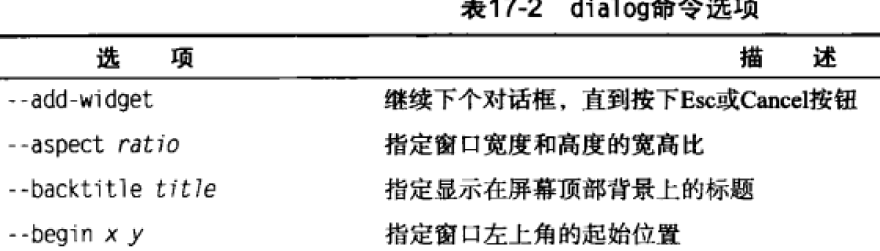
Yes与no对话框。Inputbox对话框，输入的数据会输出到STDERR，需要进行重定向一遍获取数据。Textbox，显示大量文本数据的框。

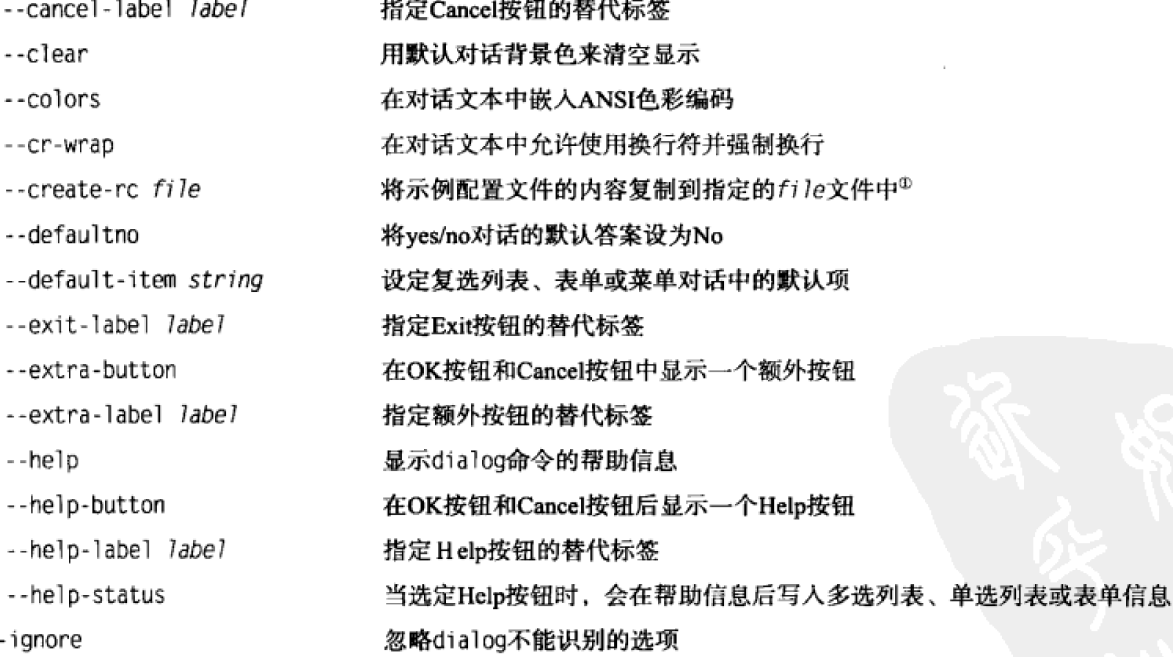


Fselect 文件选择工具。

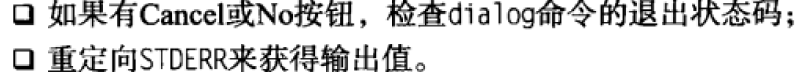


可以定制dialog的外观参数，像—title一样。





还有很多就不一一列举。在脚本中使用dialog记住2件事就是：



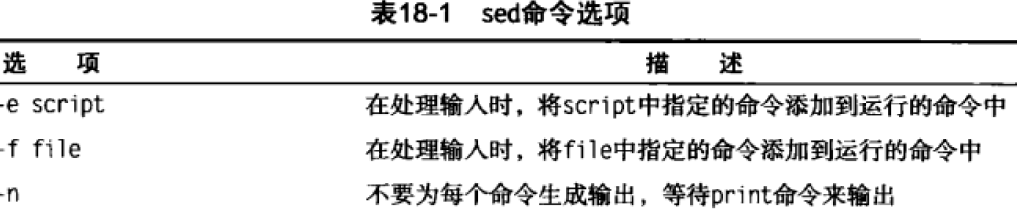
## 17.3 使用图形

也可以生成X windows图形的窗口，在KDE与Gnome窗口下的软件工具各有不同，这里看书。

# 第十八章 初识sed和gawk

## 18.1 文本处理

Sed是流编辑器，与交互式的编辑器不同，流编辑器根据预先定义的处理规则处理数据流，流编辑器每次处理一行，使用格式如下：sed options script file。



Sed命令会将指定的命令应用到STDIN输入流上。可以通过管道直接将数据送给Sed命令，比如：



应用到文件上为：

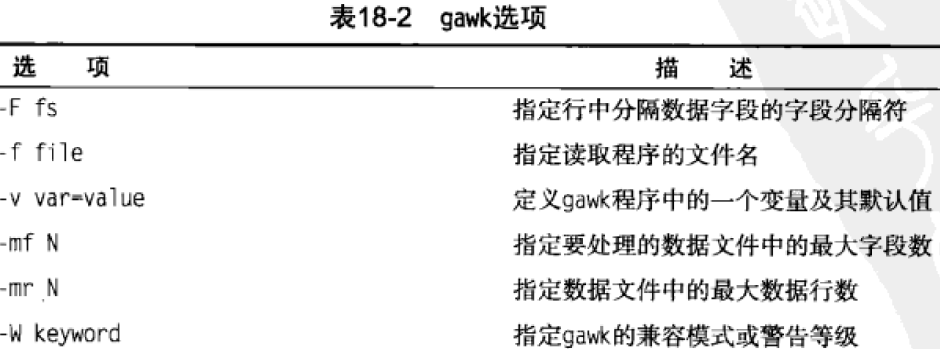


Sed不会修改源文件，只是输出源文件修改后的结果到STDOUT。

可以使用多个修改文件命令，中间使用分号分割，并用-e选项：

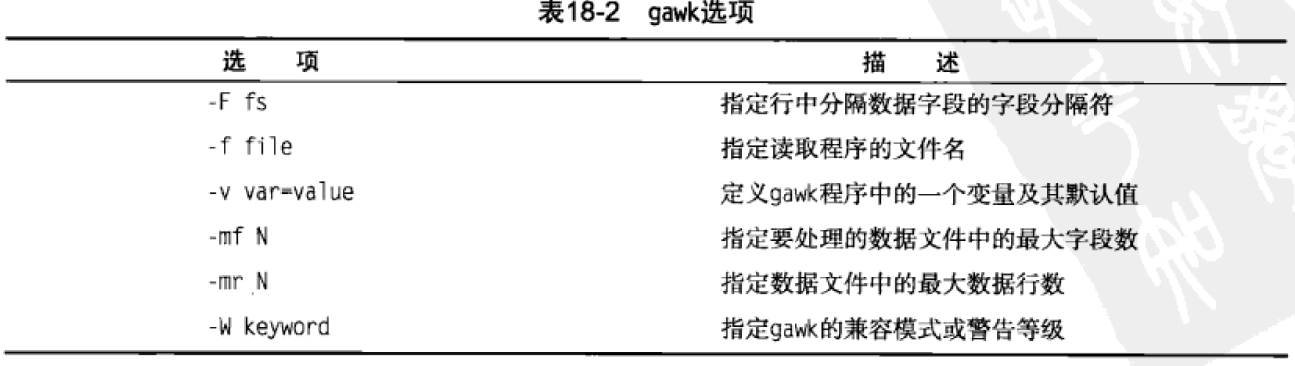


-f指定命令文件，命令放在文件中，一条命令一行。Gawk编辑器比Sed编辑器更高级，是Awk的GNU版本；在编辑器中加入了编程语言使得Awk编辑器可以处理复杂的逻辑。使用格式：gawk options program file。



Gawk程序，比Sed编辑器更加高级，包含了基本的编程元素，gawk的用法如下：

Gawk options program file



在命令行上使用gawk的程序时，要使用’{}’包裹，因为gawk假定命令是一个字符串，如果后面不指定处理的文件，则认为读取的文件时STDIN文件。

Gawk为处理的文本预定义了一些变量，$0代表处理的这一行数据，$1代表改行的第一个数据，$n依次，数据是第几个是使用的字段分隔符划分的，默认是任何空格或者制表符，-F 可以指定分隔符。多个命令与sed一样，使用分号分隔，也可以将命令放到文本中，-f读入执行，可以在里面定义变量，使用时不需要加$号使用，也可以指定在处理数据前做一些操作，使用’BEGIN{1} {2}’ 标识，BEGIN标识的{}这段命令只在命令启动时执行一次，后面每一行的数据执行{2}段的命令。于此相似，END {}标识文本处理完之后的处理命令。

## 18.2 sed编辑器基础

Sed的substitute命令实现输入数据流内的文本替换，sed ‘s/pattern/replace\_string/’；这种情况下，只会替换每行文本中第一次匹配的地方，要想多个地方都替换掉，需要使用替换标记。

s/pattern/replacement/flags；标记有4种：数字，表明在第几个匹配的地方替换；g表明所有出现的地方替换；p将原来行的内容打印出来；w file将替换的结果写到文件中；p与-n参数配合使用，将只会输出修改过的行；/是字符串的分隔符，也可以是其他的的分隔符，因为，当处理/字符时，会导致识别问题，除非进行转义处理。

Sed编辑器会处理所有行，现在想只处理某些行，是想方式有2种，一种用行号指定，一种是对行进行匹配过滤。指定行地址的形式为：[address] command或者

Address {

Command1

Command2

}

比如：’2s/dog/cat’(单行指定) ，‘2,3s/dog/cat’(范围指定)，’2,$s/dog/cat’(开始行到最后)；

匹配过滤的方式也是正则表达式：/patternstring/command，比如’/zzzz/s/dog/cat/’,就是指定在含有zzzz的行上替换。

Sed还支持d删除命令，比如sed ‘3d’ file；删除输入流中的第3行；/pattern/d删除模式匹配的行；不会删除原始文件的内容，也可以删除范围行，这个没看懂。

Sed编辑器允许向数据流中插入语附加文本，插入i，在指定行前插入新的一行，a在指定行后追加一行；使用的格式是:’[address](i|a)\newline1\newline2…’；

c命令可以修改行，y转换命令，[address]y/inchars/outchars，inchats位置的字符会被outchats对应位置的字符代替；p打印文本，=打印行号；l打印所有字符；r命令读取文件内容插入到数据流中；w将数据流中内容写到文件内。

# 第十九章 正则表达式

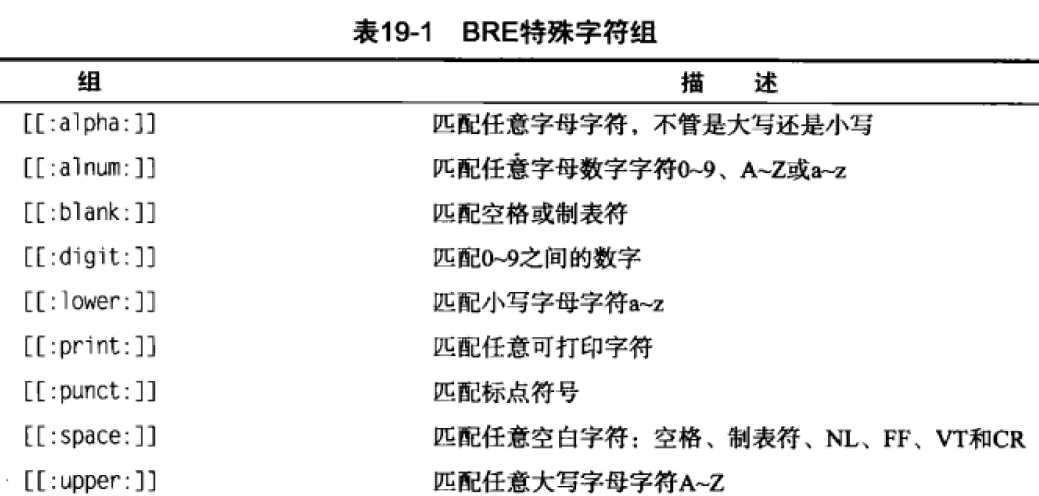
## 19.1 什么是正则表达式

数据的模式模板（pattern template），linux系统用它过滤出符合规则的数据，正则表达式是用正则表达式引擎实现的，引擎是用来解析正则表达式模式并进行模式匹配的底层软件；Linux中存在2种正则表达式引擎：BRE（基本正则表达式）、ERE（扩展正则表达式）。sed命令基于处理速度的原因使用的BRE，gawk命令使用的ERE。

## 19.2 定义BRE模式

区分大小写，只要字符串流的子集匹配上就可以；这行数据就认为与正则表达式匹配成功；特殊字符：.\*[]^${}\+?|() 这些字符在正则表达式中有特殊用途，不能拿来直接使用，但是可以转义来获得这个普通文本字符。

^匹配开始、$匹配结尾，点字符匹配单个字符；[]字符组，匹配组内的任意一个字符；[^]排除字符组；使用区间；



\*号，指定\*号前面的一个字符或者字符组内的字符出现0次或者多次。

## 19.3 扩展正则表达式

Sed不支持扩展的正则表达式，gawk支持。?号指定?号前面的一个字符或者字符组字符出现0次或者1次；+表示前面的字符至少出现一次；{m}{m,n}用区间的方式指定前面字符出现的次数。|是逻辑或，指定2边匹配任何一个都可以。()完成聚合功能，()就相当于成为了一个单一字符，与|组合可以发挥很大的威力。

## 19.4 实用中的正则表达式

# 第二十章sed进阶

## 20.1 多行命令

Sed命令在前面一次只能处理一行，也能一次处理多行；

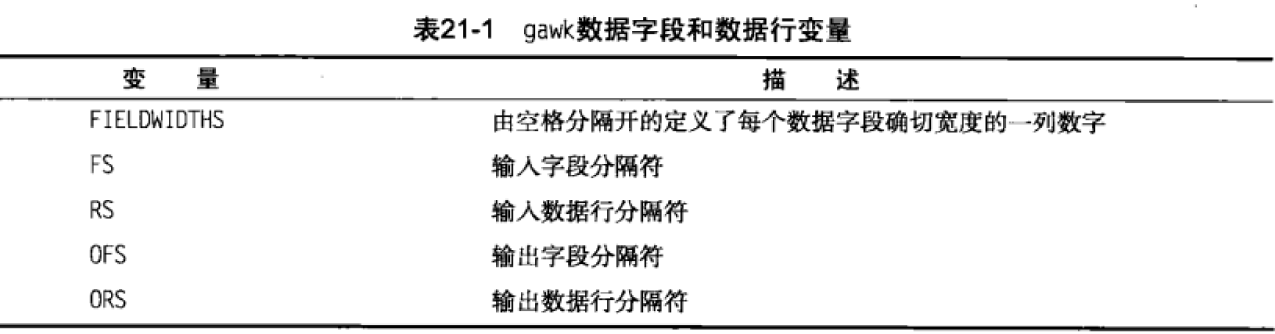
N：将下一行加进来组成多行来处理，单行next命令，小写的n命令会告诉编辑器移动到数据流中的下一文本行而不需要回到命令的最开始执行；多行的next命令使用大写N，N会将下一行的文本追加到当前行后，变为一行；此时2行会当成1行来处理；

# 第二十一章 gawk进阶

## 21.1 使用变量

变量分为内建变量与自定义变量，内建变量用来存放数据行与数据字段的信息。

$n代表了对应位置的字段数据，字段数据由字段分隔符确定，还有一些内建变量如下：



前面3个用在输入的行处理上，后面2个用在print输出上，实例如下：



FIELDWIDTHS变量定义字段长度，数据如果不是使用分隔符的，使用字段长度，指定了这个变量后，gawk忽略FS变量。在老版本的unix上，没有这个功能。

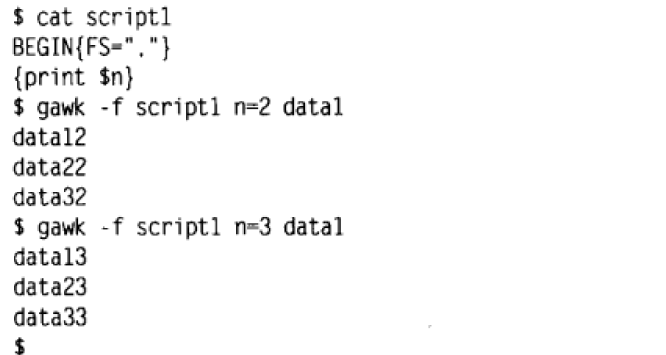


RS与ORS有时候就是指换行符，假如一条数据是分多行的，那么使用空字符串标识，FS使用换行符，文本数据中每条数据之间用一个空行分隔。其他的内建变量还有：

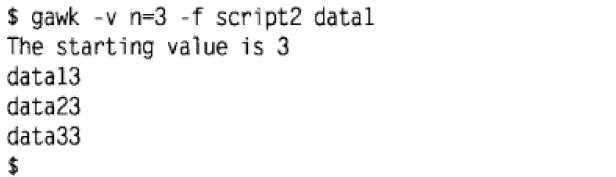


注意gawk并不会将程序脚本当作命令参数。

自定义变量区分大小写，引用不需要$，可以支持标准的数学运算，也可以将自定义变量放在gawk命令行上，



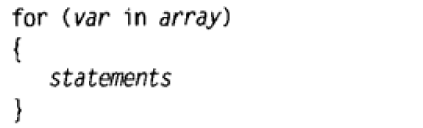
这种方式定义的变量在BEGIN程序段中，变量是不起作用的，可以加入-v方式解决，变量要定义在脚本程序的前面：



## 21.2 处理数组

Gawk支持hash表，也可以叫为关联数组，不支持普通数组。

定义数组的方式：var[ index\_string ]=value;遍历使用的方式是：



Var代表的index\_string。

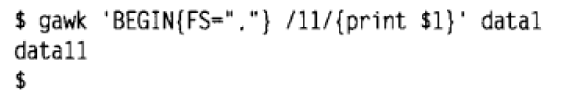
删除数组元素的语句是：



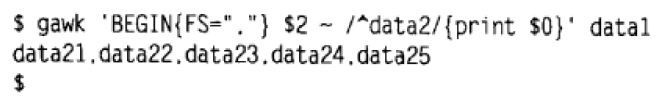
## 21.3 使用模式

使用模式的好处就是个性化处理不同的数据行，不同的数据行根据是否匹配模式来选择是否进行某些处理。

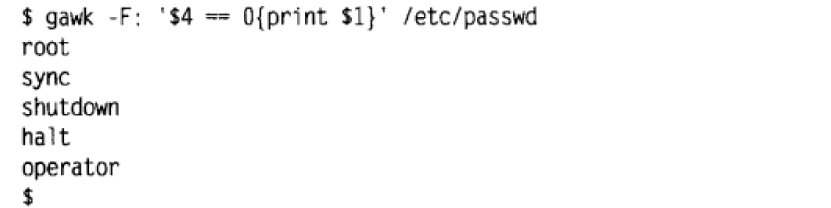
正则表达式，正则表达式必须放在所要进行处理的脚本段的前面：



gawk会拿每行进行匹配，也包括字段分隔符。匹配运算符可以将正则表达式限定在特定数据字段的匹配上，匹配运算符是波浪线~：



匹配第二个字段data2开头的的行。!~代表不匹配。也可以使用数学表达式表达匹配：



这种是完全匹配，还支持<、>、<=、>=等。

## 21.4 结构化命令

支持常见的结构化命令。支持if-else语句，跟普通的C语言语法类似。While与do-while类似，也支持break与continue语句，for语句与java类似。

## 21.5 格式化打印

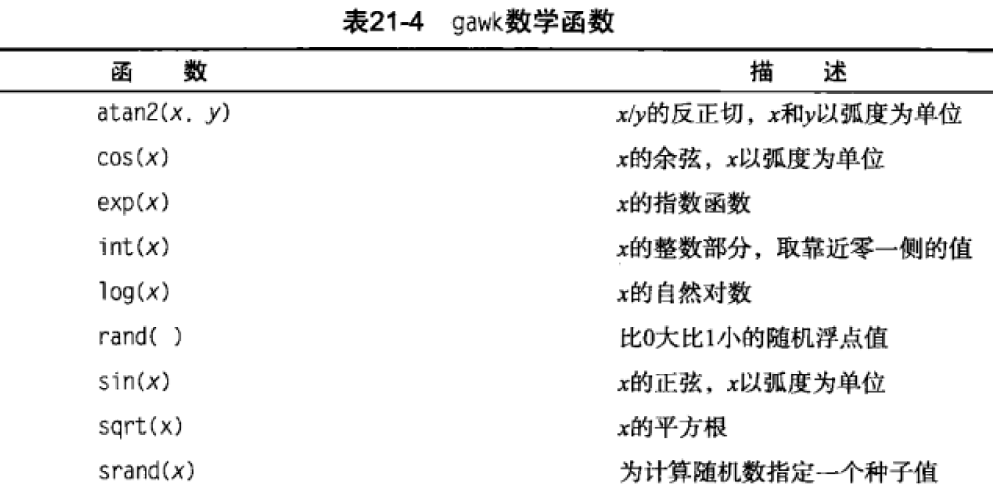
Gawk命令内可以使用格式化的打印命令：printf，与C语言的用法基本类似：

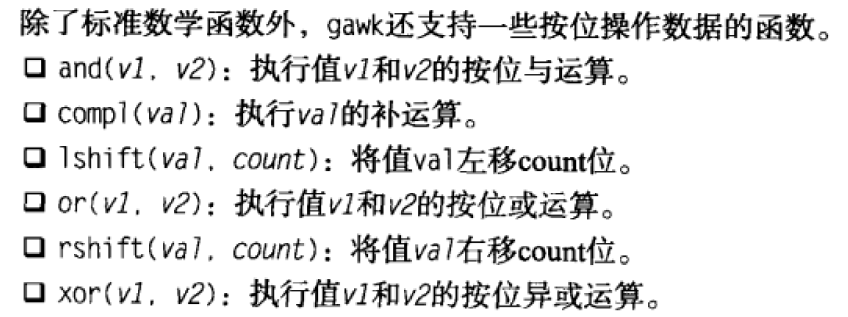


Format string是直接打印出的文本元素与格式化指定符的组合，格式化指定符也相当于占位符，分别匹配1、2、…后面的变量；格式化指定符的格式:%[modifier]control-letter；control-letter是选择展示什么类型数据的单字节码，有很多。Modifier是用来修饰输出的：width：指定了输出的最小宽度；prec:小数点后的有效位数或者字符串长度的最大值；-：左对齐而不是默认的右对齐；printf需要加入换行符。

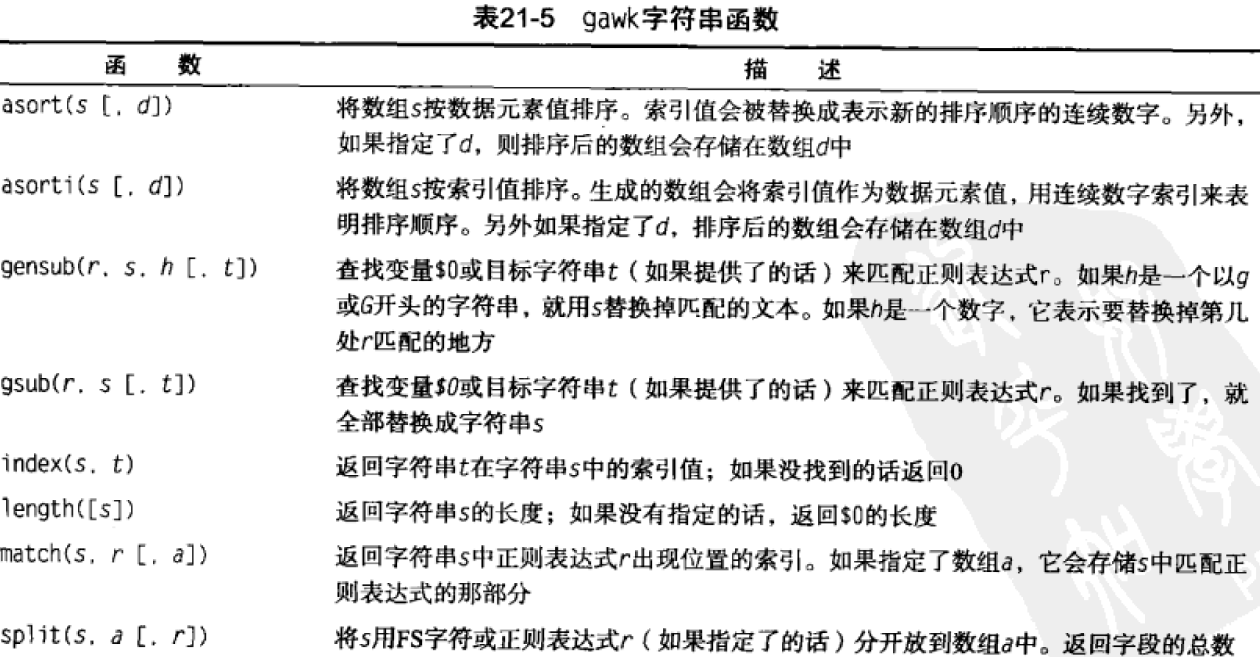
## 21.6 内建函数

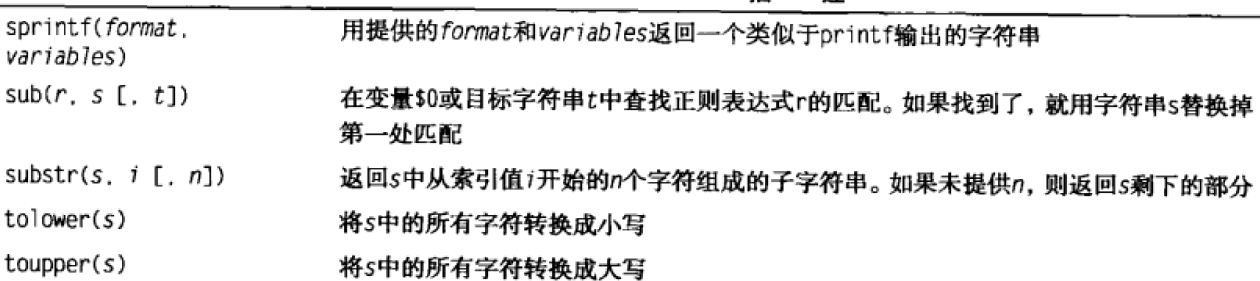
数学函数：



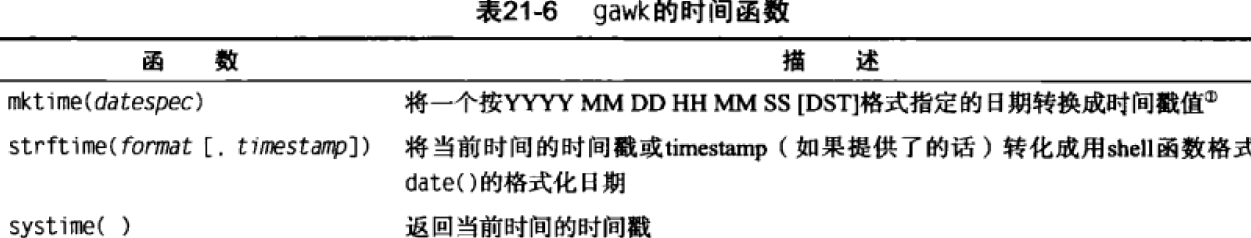


字符串函数：



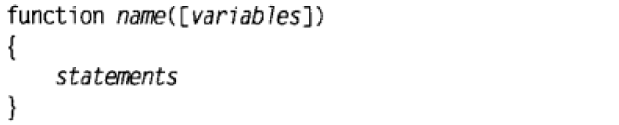


时间函数：



Epoch时间戳是距离1970年1月1日的毫秒数。

## 21.7 自定义函数



可以有返回值，可以在里面直接调用数据行的每个域变量$n等；自定义函数必须出现在所有代码块之前 gawk ‘function (){….} BEGIN{}{}’ file。

可以把所有函数放到一个库中，通过-f参数加载进来。



函数库与gawk程序不能写到一个文件，必须通过2个-f参数分别加载。

# 第二十二章 使用其他shell

## 22.1 什么是dash shell

Debian的ash shell版本，ash是b shell的简化版本；Debian中bash是默认的登录shell，dash shell是安装脚本的启动shell；ubuntu中，/bin/sh会链接到dash shell，普通的交互使用bash。

## 22.2 dash shell的特性

dash添加了很多命令行参数，具体参考命令帮助手册。Dash的环境变量与基本的shell相同，比bash的环境变量要少；定义变量：本shell可见，要在子shell中可见的话，export。Dash shell支持的内建命令与bash差不多。

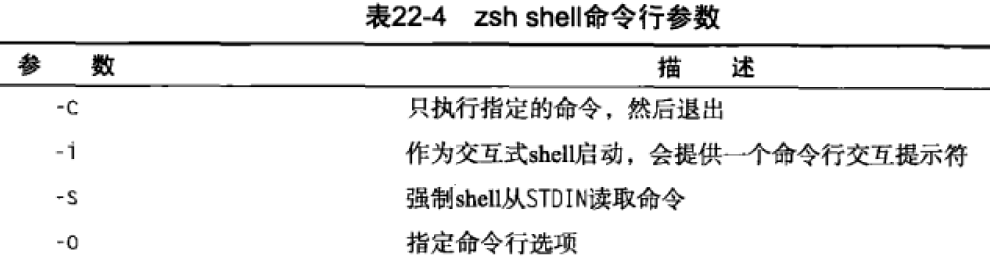
## 22.3 dash shell编程

#!/bin/dash 指定shell解析器，dash shell不能使用方括号形式的数学运算，不支持==判断，echo 输出特殊字符时也会不同，定义函数的方式也不同。

## 22.4 zsh

zsh更加全面，功能更多，而且命令模块式可加载的。

## 22.5 zsh的组成



注意-o这个可以指定zsh的选项。

zsh的基本内建命令与普通的shell差不多，加载模块后，会多出加载的命令，zmodload命令用来实现模块的加载、删除、查看等。

## 22.6 zsh脚本编程

精确的浮点数运算采用let命令与双圆括号；提供了常用的结构化命令；

# 第二十三章 使用数据库

Shell与MySQL组合。

## 23.1 MySQL数据库

mysql命令，可以带非常多的参数，mysql内还有很多可以使用的命令。

## 23.2 PostgreSQL数据库

Psql客户端

## 23.3 使用数据表

忽略

# 第二十四章 使用web

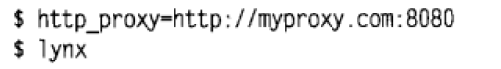
## 24.1 Lynx程序

文本浏览器。

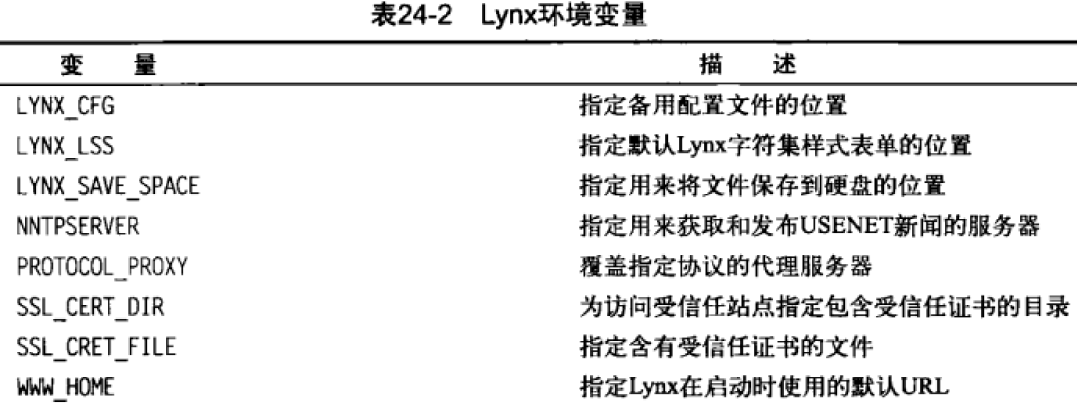
Web页面由3部分组成，HTTP头、Cookie、HTML内容；HTTP头：链接相关的信息，比如地址、安全性、发送的内容类型等；Cookie：临时数据；HTML内容可以以3种：文本图形显示、转储的原始数据、源代码。

lynx options url。Options有很多，这个只能在用的时候搜索了。这么多参数，用的时候每次都要写很麻烦，可以一次性写到配置文件中，不用每次都写。默认情况下，lynx会读取/usr/local/lib/lynx.cfg配置文件的参数，内容的形式：

PARAMETER:value；参数名都是大写字母；一些参数也可以直接定义在上下文中，比如：



可以设置的环境变量有：



抓取数据一般采用-dump参数。

## 24.2 cURL

支持非常多协议的发送数据与提取数据的命令。

curl link的http访问方式会把页面的源码展示在stdout中。

## 24.3 使用zsh处理网络

zsh是一个提供模块化的shell，用到哪个模块，就可以将哪个模块的命令加载进来，比如TCP模块，TCP模块可以直接在命令行上进行网络连接。

Zmodload zsh/net/tcp 安装TCP模块，安装好后，可以直接使用ztcp命令。

Ztcp [-acflLtv] [ -d fd ] [args]；ztcp使用$REPLY变量中的文件描述符与远程主机交互，将$REPLY中的数据发送给远程主机或者将返回的数据写回到$REPLY。

# 第二十五章 使用Email

# 第二十六章 编写脚本实用工具

~完~