第3章 字符界面操作基础

本章内容

3.1 字符界面简介
3.2 在Linux系统下获取帮助
3.3 Shell基础
3.4 使用bash
3.5 Shell实用功能
3.6 重定向
3.7 vi编辑器

- 对Linux服务器进行管理时，经常需要进入字符界面进行操作，使用命令需要记住该命令的相关选项和参数。vi编辑器可以用于编辑任何ASCII文本，可以对文本进行创建、查找、替换、删除、复制和粘贴等操作。

3.1 字符界面简介

- 对Linux服务器的维护基本上都是在命令行界面下进行的，本节主要讲述如何进入字符界面以及在字符界面下如何关闭和重启Linux系统。

进入Linux字符界面方式

- 要进入Linux系统的字符界面可以通过字符界面、图形界面下的终端以及虚拟控制台等多种方式进入。

1、Linux字符界面

- 安装Linux系统之后，系统启动默认进入的是图形化界面，可以通过使用以下命令修改为进入字符界面，所做改变在系统重新引导之后即可生效。

  [root@rhel ~]# systemctl get-default

  graphical.target

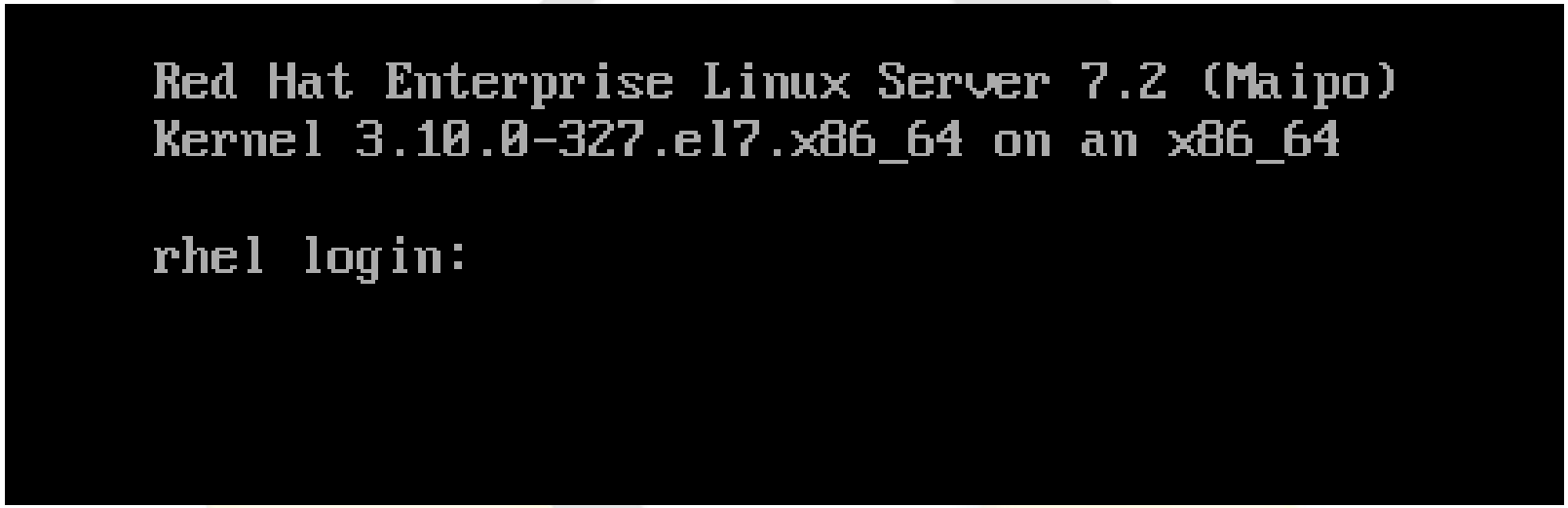
  //查看计算机系统启动后要进入的默认目标，graphical.target表示图形化界面

  [root@rhel ~]# systemctl set-default multi-user.target

  //将multi-user.target目标设置为启动计算机系统后要进入的默认目标，multi-user.target表示字符界面

字符界面登录提示

- 如果用户选择使用字符界面登录Linux系统，在系统被引导后，会看到下图所示的登录提示界面。



字符界面登录提示

- Linux系统用户登录分两步：第一步输入用户的用户名，系统根据该用户名识别用户；第二步输入用户的口令。当用户正确地输入用户名和口令后，就能合法地进入系统，这时就可以对系统进行各种操作了，注意超级用户root登录后提示符是“#”，而其他用户登录后提示符是“$”。

2、图形界面下的终端

- 在Linux系统图形化桌面环境中提供了打开终端命令行界面的方式，终端方式允许用户通过输入命令来管理计算机。

3、虚拟控制台

- Linux系统可以同时接受多个用户同时登录，还允许用户进行多次登录，这是因为Linux系统提供了虚拟控制台的访问方式。
- 在字符界面下，虚拟控制台的选择可以通过按下[Alt]键和一个功能键来实现，通常使用F1～F6键。比如用户登录后，按下[Alt+F2]键，用户可以看到“login：”提示符，说明用户进入了第二个虚拟控制台。然后只需按[Alt+F1]组合键，就可以回到第一个虚拟控制台。
- 如果用户在图形界面下，那么可以使用[Ctrl+ Alt+F2]～[Ctrl+ Alt+F6]组合键切换字符虚拟控制台，使用[Ctrl+Alt+F1]可以切换到图形界面。

关闭和重启Linux系统

- 在Linux系统中常用的关闭/重启系统的命令有shutdown、halt、reboot，每个命令的内部工作过程是不同的。

1. shutdown命令

- shutdown命令可以安全地关闭或重启Linux系统。

命令语法：

shutdown［选项］［时间］［警告信息］

【例3.1】 立即关闭计算机系统。

[root@rhel ～]# shutdown -h now

【例3.2】 定时45分钟后关闭计算机系统。

[root@rhel ~]# shutdown -h +45

【例3.3】 立即重新启动计算机系统，并发出警告信息。

[root@rhel ~]# shutdown -r now "system will be reboot now."

【例3.4】 定时在1点38分重新启动计算机系统。

[root@rhel ~]# shutdown -r 01:38

2. halt命令

- 使用halt命令就是调用“shutdown -h”命令执行关机任务。

命令语法：

halt [选项]

【例3.5】 使用halt命令关闭系统。

```
[root@rhel ~]# halt
```

3．reboot命令

- reboot命令的工作过程与halt相似，不过reboot是引发计算机重启，而halt是引发计算机关闭。它的选项与halt相似。

【例3.6】 使用reboot命令重启计算机系统。

```
[root@rhel ~]# reboot
```

目标

- 在RHEL 7之前的版本，使用运行级别代表特定的操作模式。运行级别被定义为七个级别，用数字0到6表示，每个运行级别可以启动特定的一些服务。RHEL 7使用目标（target）替换运行级别。目标使用目标单元文件描述，目标单位文件扩展名是.target，目标单元文件的唯一目标是将其他systemd单元文件通过一连串的依赖关系组织在一起。比如graphical.target单元，用于启动一个图形会话，systemd会启动像GNOME显示管理（gdm.service）、帐号服务（axxounts-daemon）这样的服务，并且会激活multi-user.target单元。相似的multi-user.target单元，会启动必不可少的NetworkManager.service、dbus.service服务，并激活basic.target单元。
- 每一个目标都有名字和独特的功能，并且能够同时启用多个。一些目标继承其他目标的服务，并启动新服务。systemd 提供了一些模仿System V init启动级别的目标，仍可以使用旧的telinit启动级别命令切换。

预定义目标和运行级别对应关系

运行级别	目标	目标的链接文件	功能
0	poweroff.target	runlevel0.target	关闭系统
1	rescue.target	runlevel1.target	进入救援模式
2	multi-user.target	runlevel2.target	进入非图形界面的多用户方式
3	multi-user.target	runlevel3.target	进入非图形界面的多用户方式
4	multi-user.target	runlevel4.target	进入非图形界面的多用户方式
5	graphical.target	runlevel5.target	进入图形界面的多用户方式
6	reboot.target	runlevel6.target	重启系统

3.2 在Linux系统下获取帮助

- Linux系统中的每个命令都具有众多的选项和参数，要完全记住它们是不可能的，本节主要讲述了在Linux系统下如何获取和使用帮助。

使用man手册页

- 一般情况下，Linux系统中所有的资源都会随操作系统一起发行，包括内核源代码。而在线手册是操作系统所有资源的一本很好的使用手册。有不懂的命令时可以用man查看这个命令，写程序时有不会用的函数可以用man查看这个函数，有不懂的文件时也可以用man查看文件。
- 一般情况下man手册页的资源主要位于/usr/share/man目录下。

man手册页类型

类型	描述
1	用户命令
2	系统调用
3	C语言函数库
4	设备和特殊文件
5	文件格式和约定
6	游戏程序
7	杂记
8	系统管理工具
9	Linux内核API（内核调用）

man命令

- man命令格式化并显示在线的手册页。通常使用者只要在命令man后，输入想要获取的命令的名称，man就会列出一份完整的说明，其内容包括命令语法、各选项的意义以及相关命令等。

  命令语法：

  man [选项] [名称]

man手册页组成内容

手册页内容	说　　明
Header	标题
NAME	man的命令/函数的功能概述
SYNOPSIS	man的命令/函数用法的简单描述
AVAILABILITY	可用性说明
DESCRIPTION	man的命令/函数的详细描述
OPTIONS	该命令的所有可选项的详细说明
RETURN VALUE	如果是函数，则列出函数返回值
ERRORS	如果函数调用出错，则列出所有出错的值和可能引起错误的原因
FILES	该命令/函数所用到的相关系统文件
ENVIRONMENT	和该命令/函数相关的环境变量
NOTES	表示不常用的用法或实现的细节
BUGS	已知的错误和警告
HISTORY	该命令/函数的历史发展
SEE ALSO	可以参照的其他的相关命令/函数
Others	和一些具体命名/函数有关的特殊信息

```
[root@rhel ~]# man ls
LS(1)                          User Commands                          LS(1)
NAME
       ls - list directory contents
SYNOPSIS
       ls [OPTION]... [FILE]...
DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort is  speci-
       fied.
       Mandatory  arguments  to  long  options are mandatory for short options
       too.
       -a, --all
              do not ignore entries starting with .
       -A, --almost-all
              do not list implied . and ..
       --author
```

Manual page ls(1) line 1 (press h for help or q to quit)

使用--help选项获取帮助

- 使用--help选项可以显示命令的使用方法以及命令选项的含义。只要在所需要显示的命令后面输入“--help”选项，然后就可以看到所查命令的帮助内容了。

  命令语法：

  [命令] --help

【例3.9】 查看mkdir命令的帮助信息。

```
[root@rhel ~]# mkdir --help
```

3.3 Shell基础

- 在Linux系统中，Shell是最常使用的程序，其主要作用是侦听用户指令、启动指令所指定的进程并将结果返回给用户，本节主要讲述Shell的基本使用方法。

Shell简介

- 在AT&T工作的Dennis Ritchie和Ken Thompson两人在设计Unix操作系统的时候，想要为用户创建一种与Unix系统交流的方法。那时的操作系统带有命令解释器。命令解释器接受用户的命令，然后解释它们，因而计算机可以使用这些命令。
- Ritchie和Thompson想要提供比当时的命令解释器具备更优异功能的工具。这导致了Bourne Shell（通称为sh）的开发，由S.R.Bourne创建。自从Bourne Shell出现以后，其它类型Shell也被一一开发，比如C Shell（csh）和Korn Shell（ksh）。

- Shell接收用户命令，然后调用相应的应用程序，同时它还是一种程序设计语言， 是系统管理维护时的重要工具。 作为命令语言，它交互式的解释和执行用户输入的命令或者自动地解释和执行预先设定好的一连串的命令。作为程序设计语言，它可以定义各种变量和参数，并提供了许多在高级语言中才具有的控制结构（循环和分支）。
- Shell命令重新初始化用户的登录会话。当给出该命令时，就会重新设置进程的控制终端的端口特征，并取消对端口的所有访问。然后Shell命令为用户把进程凭证和环境重新设置为缺省值，并执行用户的初始程序。根据调用进程的登录用户标识建立所有的凭证和环境。
- 目前流行的Shell 有sh、csh 、ksh、tcsh 和bash等。大部分Linux系统的默认Shell类型为bash。

bash简介

- bash（Bourne-Again Shell）最早是在1987年由布莱恩·福克斯开发的一个为GNU计划编写的Unix Shell。bash目前是大多数Linux系统默认的Shell，它还能运行于大多数Unix风格的操作系统上。
- bash的命令语法是Bourne shell命令语法的超集。数量庞大的Bourne shell脚本大多不经过修改就可以在bash中执行，只有那些引用了Bourne特殊变量或使用了Bourne内置命令的脚本才需要修改。bash的命令语法很多来自ksh和csh，比如命令行编辑、命令历史、目录栈、$RANDOM变量、$PPID变量以及POSIX命令置换语法。

bash命令

- Linux系统的标准提示符包括了用户登录名、登录的主机名、当前所在的工作目录路径和提示符号。

  以普通用户zhangsan登录名为rhel的主机，他的工作目录是/home/zhangsan，如下所示。

  [zhangsan@rhel ~]$

  以root用户登录系统的提示符如下所示。

  [root@rhel ~]#

Shell命令一般格式

- 要运行命令的话，只需要在提示符后敲进命令，然后再按“回车”键。

命令语法：

[Shell命令] [选项] [参数]

- 所有选项在该命令的man手册页中都有详细的介绍，而参数则由用户提供。选项决定命令如何工作，而参数则用于确定命令作用的目标。
- 选项有短命令行选项和长命令选项两种。

Linux系统命令分类

- bash 内置的命令

  如果是bash内置的命令，则由bash负责回应。

- 应用程序

  如果是应用程序，那么Shell会找出该应用程序，然后将控制权交给内核，由内核执行该应用程序，执行完之后，再将控制权交回给Shell。

3.4 使用bash

- Linux系统中的Shell，除了可作为命令编译器之外，它也是一种不错的编程语言，是系统管理维护时的重要工具。

1、常用控制组合键

控制组合键	功能
Ctrl+l	清屏
Ctrl+o	执行当前命令，并选择上一条命令
Ctrl+s	阻止屏幕输出
Ctrl+q	允许屏幕输出
Ctrl+c	终止命令
Ctrl+z	挂起命令
Ctrl+m	相当于按回车键
Ctrl+d	输入结束，即EOF的意思，或者注销Linux系统

2、光标操作（1）

组合键	功能
Ctrl+a	移动光标到命令行首
Ctrl+e	移动光标到命令行尾
Ctrl+f	按字符前移（向右）
Ctrl+b	按字符后移（向左）
Ctrl+xx	在命令行首和光标之间移动
Ctrl+u	删除从光标到命令行首的部分
Ctrl+k	删除从光标到命令行尾的部分
Ctrl+w	删除从光标到当前单词开头的部分
Ctrl+d	删除光标处的字符
Ctrl+h	删除光标前的一个字符

2、光标操作（2）

组合键	功能
Ctrl+y	插入最近删除的单词
Ctrl+t	交换光标处字符和光标前面的字符
Alt+f	按单词前移（向右）
Alt+b	按单词后移（向左）
Alt+d	从光标处删除至单词尾
Alt+c	从光标处更改单词为首字母大写
Alt+u	从光标处更改单词为全部大写
Alt+l	从光标处更改单词为全部小写
Alt+t	交换光标处单词和光标前面的单词
Alt+Backspace	与Ctrl+w功能类似，分隔符有些差别

3、特殊字符

符号	功能
~	用户主目录
`	反引号，用来命令替代（在Tab键上面的那个键）
#	注释
$	变量取值
&	后台进程工作
(	子Shell开始
)	子Shell结束
\	使命令持续到下一行
|	管道
<	输入重定向
>	输出重定向
>>	追加重定向
'	单引号（不具有变数置换的功能）
"	双引号（具有置换的功能）
/	路径分隔符
;	命令分隔符

4、通配符

符号	功能
?	代表任何单一字符
*	代表任何字符
[字符组合]	在中括号中的字符都符合，比如[a-z]代表所有的小写字母
[!字符组合]	不在中括号中的字符都符合，比如[!0-9]代表非数字的都符合

3.5 Shell实用功能

- Linux系统是在命令行下面诞生的，因此，Linux中的命令行有许多非常实用的功能，在本节中将详细讲解这些实用功能。

命令行自动补全

- 在Linux系统中，有太多的命令和文件名称需要记忆，使用命令行补全功能可以快速的写出文件名和命令名。

  如果需要快速地从当前所在的目录跳转到/usr/src/kernels/目录，可以执行以下操作。

  [root@rhel ～]# cd /u<Tab>/sr<Tab>/k<Tab>

  <Tab>是按“Tab”键的意思，使用“Tab”键也称为命令行自动补全，这在平常应用中是不可缺少的。

命令历史记录

- 在操作Linux系统的时候，每一个操作的命令都会记录到命令历史中，在以后可以通过命令历史查看和使用以前操作的命令。
- bash启动的时候会读取~/.bash_history文件，并将其载入到内存中，$HISTFILE变量就用于设置~/.bash_history文件，bash退出时也会把内存中的历史记录回写到~/.bash_history文件中。
- 使用history命令可以查看命令历史记录，每一条命令前面都会有一个序列号标示。

  命令语法：

  history [选项]

使用命令历史举例

举例	描述
!!	运行上一个命令
!6	运行第6个命令
!8 /test	运行第8个命令并在命令后面加上/test
!?CF?	运行上一个包含CF字符串的命令
!ls	运行上一个ls命令（或以ls开头的历史命令）
!ls:s/CF/G	运行上一个ls命令，其中把CF替换成G
fc	编辑并运行上一个历史命令
fc 6	编辑并运行第6条历史命令
^boot^root^	快速替换。将最后一个命令中的boot替换为root后运行
!-5	运行倒数第5个命令
!$	运行前一个命令最后的参数

【例3.11】 使用命令历史记录功能键。

```
[root@rhel ~]# mkdir /root/aaa
//创建目录/root/aaa
[root@rhel ~]# cd !$
cd /root/aaa
//!$是指重复前一个命令最后的参数，参数是/root/aaa
[root@rhel aaa]# pwd
/root/aaa
//显示用户当前目录是/root/aaa
```

搜索历史命令

快捷键	描述
↑（向上方向箭）	查看上一个命令
↓（向下方向箭）	查看下一个命令
Ctrl+p	查看历史列表中的上一个命令
Ctrl+n	查看历史列表中的下一个命令
Ctrl+r	向上搜索历史列表
Alt+p	向上搜索历史列表
Alt+>	移动到历史列表末尾

命令排列

- 如果希望一次执行多个命令，Shell允许在不同的命令之间，放上特殊的排列字符。

  1. 使用“；”

  使用“；”命令时先执行命令1，不管命令1是否出错，接下来就执行命令2。

  命令语法：

  命令1; 命令2

【例3.15】 使用排列命令“;”同时执行两个命令。

```
[root@rhel ~]# ls -l /boot;du -hs /root
总用量 24131
-rw-r--r--. 1 root root   116892 5月   8 01:43 config-3.3.4-5.fc17.i686.PAE
drwxr-xr-x. 2 root root     1024 6月   2 23:47 grub
drwxr-xr-x. 6 root root     1024 6月   3 01:17 grub2
-rw-r--r--. 1 root root 17716383 6月   3 01:13 initramfs-3.3.4-5.fc17.i686.PAE.img
drwx------. 2 root root    12288 6月   2 22:39 lost+found
-rw-------. 1 root root  1914110 5月   8 01:43 System.map-3.3.4-5.fc17.i686.PAE
-rw-r--r--. 1 root root   228484 1月  17 00:08 tboot.gz
-rw-r--r--. 1 root root     9220 1月  17 00:08 tboot-syms
-rwxr-xr-x. 1 root root  4696896 5月   8 01:43 vmlinuz-3.3.4-5.fc17.i686.PAE
6.6M    /root
```

//先在屏幕上列出/boot目录中的所有内容，然后列出目录/root及其子目录所占磁盘大小

2. 使用“&&”

使用“&&”命令时只有当命令1正确运行完毕后，才能执行命令2。

命令语法：

命令1&&命令2

【例3.16】 使用排列命令“&&”同时执行两个命令。

[root@rhel ~]# ls -a /root/bogusdir&&du -hs

ls: 无法访问/root/bogusdir: 没有那个文件或目录

//将返回“ls: 无法访问/root/bogusdir: 没有那个文件或目录”，而“du –hs”命令根本没有运行，因为没有/root/bogusdir目录

命令替换

- 在Linux系统中，Shell命令的参数可以由另外一个命令的结果来替代，这种称之为命令替换。

  1．使用“$( )”

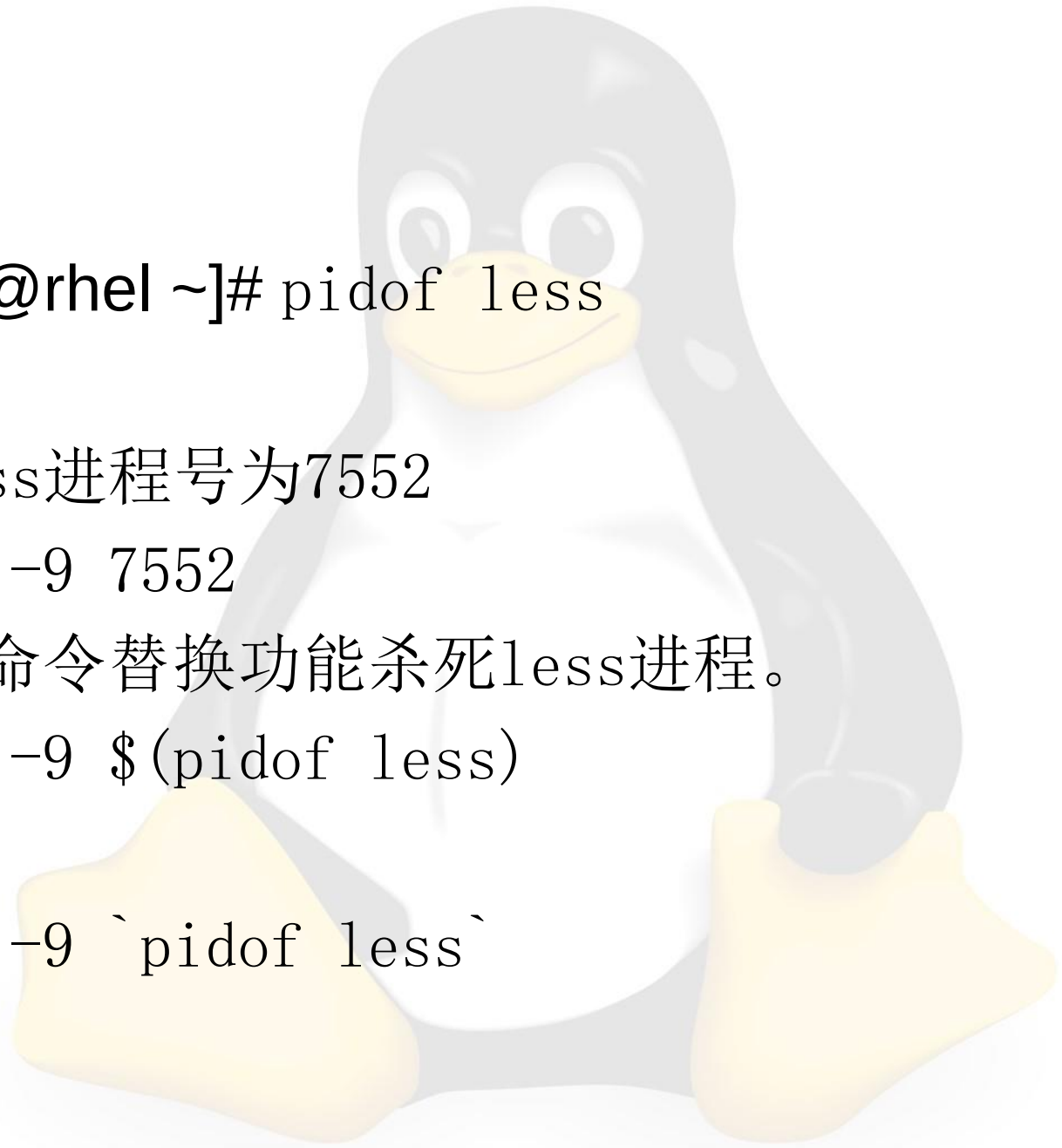
  命令语法：

  命令1 $(命令2)

  2．使用“`”（位于Tab键上面那个键）

  命令语法：

  命令1 `命令2`



[root@rhel ~]# pidof less

7552

//less进程号为7552

Kill -9 7552

使用命令替换功能杀死less进程。

kill -9 $(pidof less)

或者

kill -9 `pidof less`

命令别名

- 在需要执行某一个非常长的命令时，所有的命令以及命令的选项、参数都要一一输入，很枯燥也容易出现错误。可以为常用命令定义快捷方式，这些快捷方式可以用比较简单的命令别名来定义。

  1．创建别名

  使用alias命令可以为命令定义别名。如果命令中有空格的话，就需要使用双引号（比如在命令与选项之间就有空格）。

  命令语法：

  alias [别名]=[需要定义别名的命令]

  2．取消别名

  当用户需要取消别名的定义时，可以使用unalias命令。

  命令语法：

  unalias [别名]

文件名匹配

- 文件名匹配使得用户不必一一写出文件名称就可以指定多个文件。这将用到一些特殊的字符，称之为通配符。

  1．通配符“*”

  “*”可匹配一个或多个字符。

  2．通配符“？”

  在匹配时，一个问号只能代表一个字符。

管道

- Linux系统的理念是汇集许多小程序，每个程序都有特殊的专长。复杂的任务不是由大型软件完成，而是运用Shell 的机制，组合许多小程序共同完成。管道就在其中发挥着重要的作用，它可以将某个命令的输出信息当作某个命令的输入，由管道符号“｜”来标识。

  命令语法：

  [命令1]|[命令2]|[命令3]

【例3.22】 使用简单的管道。

```
[root@rhel ~]# ls /etc|more
abrt
acpi
adjtime
akonadi
--More--
```

//命令ls /etc显示/etc目录的内容，命令more是分页显示内容

【例3.23】 使用复杂的管道。

```
[root@PC-LINUX ~]# rpm -qa|grep a|more
```

//命令rpm -qa显示已经安装在系统上的RPM包，命令grep a是过滤软件包，命令more是分页显示这些信息

3.6 重定向

- 希望将命令的输出结果保存到文件中，或者以文件内容作为命令的参数，这时就需要用到重定向。重定向不使用系统的标准输入端口、标准输出端口或是标准错误端口，而是进行重新的指定。
- 重定向有四种方式：输出重定向、输入重定向、错误重定向以及同时实现输出和错误的重定向。

输出重定向

- 输出重定向，即将某一命令执行的输出保存到文件中，如果已经存在相同的文件，那么覆盖源文件中的内容。

  命令语法：

  [命令] > [文件]

【例3.24】使用输出重定向将/boot目录的内容保存到/root/abc文件中。

```
[root@rhel ~]# ls /boot > /root/abc
```

【例3.25】使用echo命令和输出重定向创建/root/mm文件，文件内容是hello。

```
[root@rhel ~]# echo Hello > /root/mm
[root@rhel ~]# cat /root/mm
Hello
```

//显示文件/root/mm，可以看到文件的内容是Hello

- 另外一种特殊的输出重定向是输出追加重定向，即将某一命令执行的输出添加到已经存在的文件中。

  命令语法：

  [命令] >> [文件]

【例3.26】使用输出追加重定向将数据写入文件/root/ao。

```
[root@rhel ~]# echo Hello > /root/ao
//先创建文件/root/ao，文件内容是Hello
[root@rhel ~]# echo Linux >> /root/ao
//向文件/root/ao中追加数据Linux
[root@rhel ~]# cat /root/ao
Hello
Linux
```

输入重定向

- 输入重定向，即将某一文件的内容作为命令的输入。
  命令语法：
  [命令] < [文件]

【例3.27】使用输入重定向将文件/root/mm的内容作为输入让cat命令执行。

```
[root@rhel ~]# cat <  /root/mm
Hello
//可以看到文件/root/mm的内容是Hello
```

- 另外一种特殊的输入重定向是输入追加重定向，这种输入重定向告诉Shell，当前标准输入来自命令行的一对分隔符之间的内容。

  命令语法：

  [命令] << [分隔符]

  > [文本内容]

  > [分隔符]

【例3.28】使用输入追加重定向创建/root/bc文件。

```
[root@rhel ~]# cat > /root/bc <<EOF
> Hello Linux
> EOF
//一般使用EOF作为分隔符
```

错误重定向

- 错误重定向，即将某一命令执行的出错信息输出到指定文件中。

  命令语法：

  [命令] 2> [文件]

【例3.29】 查看根本不存在的/root/kk文件，出现报错信息，将其保存到文件/root/b中。

[root@rhel ~]# cat /root/kk 2> /root/b

[root@rhel ~]# cat /root/b

cat: /root/kk: 没有那个文件或目录

//使用cat命令查看/root/b文件，可以看到其内容就是执行命令cat /root/kk的报错信息

- 另外一种特殊的错误重定向是错误追加重定向，即将某一命令执行的出错信息添加到已经存在的文件中。

  命令语法：

  [命令] 2>> [文件]

【例3.30】 使用错误追加重定向，将执行命令的多次出错信息保存到文件/root/b中。

```
[root@rhel ~]# cat /root/kk 2> /root/b
[root@rhel ~]# cat /root/kk 2>> /root/b
[root@rhel ~]# cat /root/b
cat: /root/kk: 没有那个文件或目录
cat: /root/kk: 没有那个文件或目录
```

同时实现输出和错误重定向

- 同时实现输出和错误的重定向，即可以同时实现输出重定向和错误重定向的功能。

  命令语法：

  [命令] &> [文件]

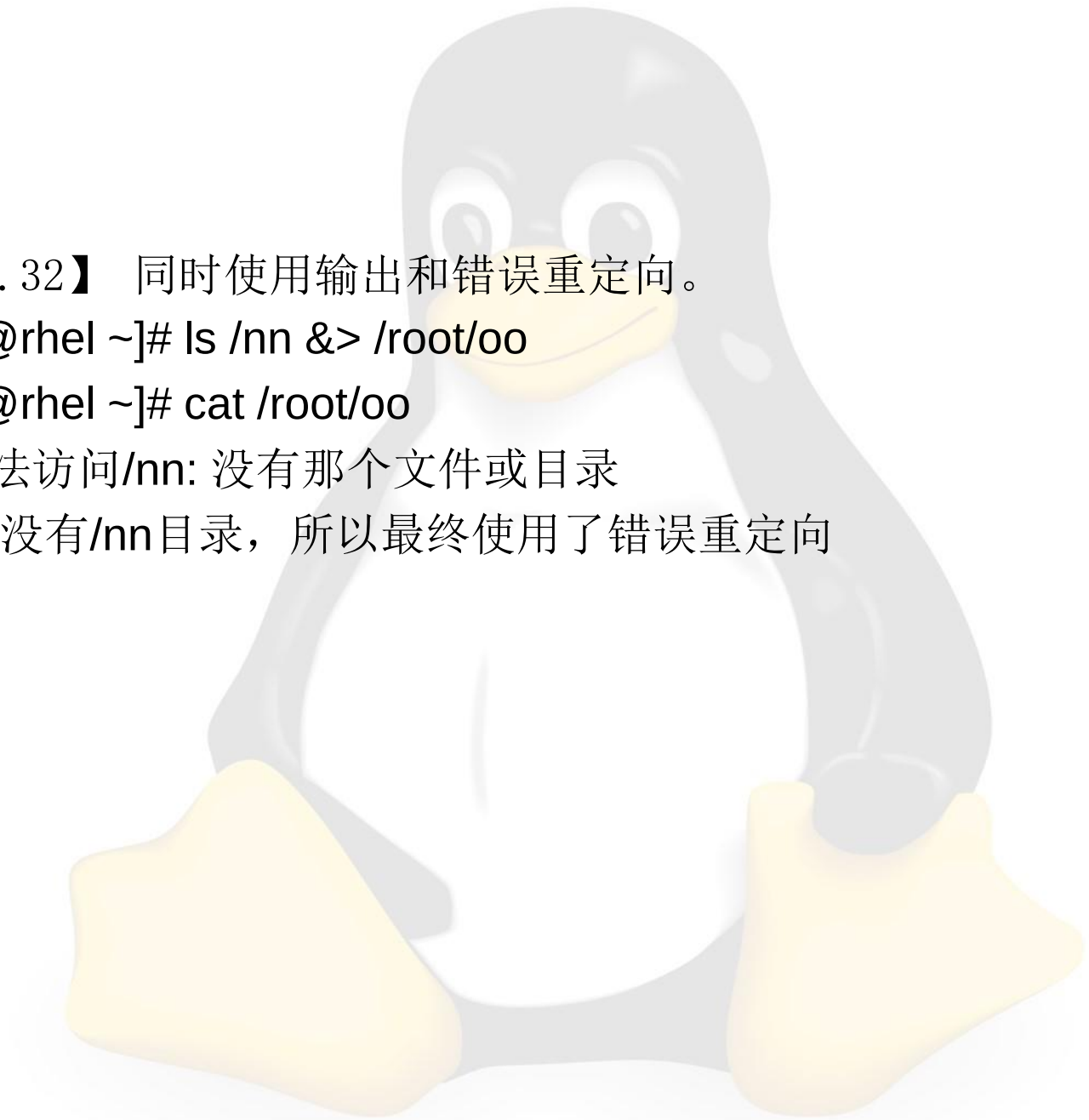
【例3.31】 同时使用输出和错误重定向。

```
[root@rhel ~]# ls /boot &> /root/kk
[root@rhel ~]# cat /root/kk
config-3.3.4-5.fc17.i686.PAE
grub
grub2
initramfs-3.3.4-5.fc17.i686.PAE.img
lost+found
System.map-3.3.4-5.fc17.i686.PAE
tboot.gz
tboot-syms
vmlinuz-3.3.4-5.fc17.i686.PAE
```

//因为/boot目录下有文件，所以最终使用了输出重定向

【例3.32】 同时使用输出和错误重定向。

```
[root@rhel ~]# ls /nn &> /root/oo
[root@rhel ~]# cat /root/oo
ls: 无法访问/nn: 没有那个文件或目录
//因为没有/nn目录，所以最终使用了错误重定向
```



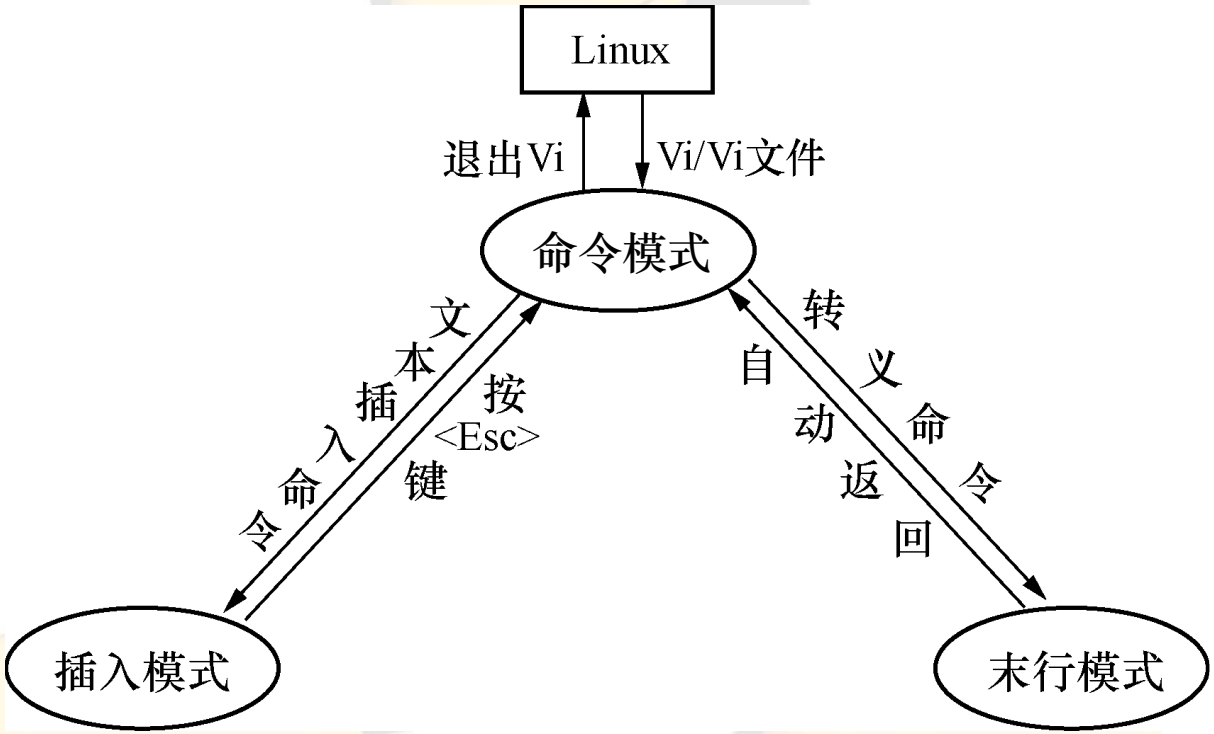
3.7 vi编辑器

- 文本编辑器有很多，图形模式下有gedit，kwrite等编辑器，文本模式下的编辑器有vi，vim（vi的增强版本）和nano。
- vi和vim是Linux系统中最常用的编辑器，本节主要讲述vi编辑器的使用。

vi编辑器简介

- vi编辑器是Linux系统字符界面下最常使用的文本编辑器，用于编辑任何ASCII文本，对于编辑源程序尤其有用。vi编辑器功能非常强大，通过使用vi编辑器，可以对文本进行创建、查找、替换、删除、复制和粘贴等操作。
- 在Linux系统Shell提示符下输入vi和文件名称后，就进入vi编辑界面。如果系统内还不存在该文件，就意味着创建文件，如果系统内存在该文件，就意味着编辑该文件。
- vi编辑器有3种基本工作模式，分别是命令模式、插入模式和末行模式。

vi编辑器工作模式



1. 命令模式

- 进入vi编辑器之后，系统默认处于命令模式。命令模式控制屏幕光标的移动，字符、字或行的删除，某区域的移动、复制等。在命令模式下，按冒号键“:”可以进入末行模式，按字母键“a”就可以进入插入模式。

2. 插入模式

- 只有在插入模式下，才可以进行文本编辑。在插入模式下按“Esc”键可回到命令模式。

3. 末行模式

- 将文件保存或退出vi编辑器，也可以设置编辑环境、替换字符或删除字符。
- 在末行模式下按“Esc”键可以回到命令模式。

进入插入模式命令

命令	功能
i	从光标当前所在位置之前开始插入
a	从光标当前所在位置之后开始插入
I	在光标所在行的行首插入
A	在光标所在行的行末尾插入
o	在光标所在的行的下面新开一行插入
O	在光标所在的行的上面新开一行插入
s	删除光标位置的一个字符，然后进入插入模式
S	删除光标所在的行，然后进入插入模式

光标移动（1）

命令	功能
↑键（向上箭头）	使光标向上移动一行
↓键（向下箭头）	使光标向下移动一行
←键（向左箭头）	使光标向左移动一个字符
→键（向右箭头）	使光标向右移动一个字符
k	使光标向上移动一行
j	使光标向下移动一行
h	使光标向左移动一个字符
l	使光标向右移动一个字符
nk	使光标向上移动n行，n代表数字

光标移动（2）

nj	使光标向下移动n行，n代表数字
nh	使光标向左移动n个字符，n代表数字
nl	使光标向右移动n个字符，n代表数字
H	使光标移动到屏幕的顶部
M	使光标移动到屏幕的中间
L	使光标移动到屏幕的底部
Ctrl+b	使光标往上移动一页屏幕
Ctrl+f	使光标往下移动一页屏幕
Ctrl+u	使光标往上移动半页屏幕
Ctrl+d	使光标往下移动半页屏幕
0（数字0）	使光标移到所在行的行首

光标移动（3）

$	使光标移动到光标所在行的行尾
^	使光标移动到光标所在行的行首
w	使光标跳到下一个字的开头
W	使光标跳到下一个字的开头，但会忽略一些标点符号
e	使光标跳到下一个字的字尾
E	使光标跳到下一个字的字尾，但会忽略一些标点符号
b	使光标回到上一个字的开头
B	使光标回到上一个字的开头，但会忽略一些标点符号
(	使光标移动到上一个句首
)	使光标移动到下一个句首
{	使光标移动到上一个段落首
}	使光标移动到下一个段落首
G	使光标移动到文件尾（最后一行的第一个非空白字符处）
gg	使光标移动到文件首（第一行第一个非空白字符处）

光标移动（4）

space键（空格键）	使光标向右移动一个字符
Backspace键	使光标向左移动一个字符
Enter键	使光标向下移动一行
Ctrl+p	使光标向上移动一行
Ctrl+n	使光标向下移动一行
n|	使光标移动到第n个字符处，n代表数字
nG	使光标移动到第n行首，n代表数字
n+	使光标向下移动n行，n代表数字
n-	使光标向上移动n行，n代表数字
n$	使光标移动到以当前行算起的第n行尾，n代表数字

命令模式命令（1）

类型	命令	功能
删除	x	删除光标所在位置的字符
	X	删除光标所在位置的前面一个字符
	nx	删除光标所在位置开始的n个字符，n代表数字
	nX	删除光标所在位置前面n个字符，n代表数字
	dd	删除光标所在行
	ndd	从光标所在行开始删除n行，n代表数字
	db	删除光标所在位置的前面一个单词
	ndb	删除光标所在位置的前面n个单词，n代表数字
	dw	从光标所在位置开始删除一个单词
	ndw	从光标所在位置开始删除几个单词，n代表数字
	d$	删除光标到行尾的内容（含光标所在处字符）
	D	删除光标到行尾的内容（含光标所在处字符）
	dG	从光标位置所在行一直删除到文件尾

命令模式命令（2）

类型	命令	功能
复制和粘贴	yw	复制光标所在位置到单词尾的字符
	nyw	复制光标所在位置开始的n个单词，n代表数字
	yy	复制光标所在行
	nyy	复制从光标所在行开始的n行，n代表数字
	y$	复制光标所在位置到行尾内容到缓存区
	y^	复制光标前面所在位置到行首内容到缓存区
	YY	将当前行复制到缓冲区
	nYY	将当前开始的n行复制到缓冲区，n代表数字
	p	将缓冲区内的内容写到光标所在的位置

命令模式命令（3）

类型	命令	功能
替换	r	替换光标所在处的字符，按[r]键之后输入要替换的字符
	R	替换光标所到之处的字符，直到按下[ESC]键为止，按[R]键之后输入要替换的字符
撤销和重复	u	撤销上一个操作。按多次u可以执行多次撤销
	U	取消所有操作
	.	再执行一次前面刚完成的操作
列出行号	Ctrl+g	列出光标所在行的行号
保存和退出	ZZ	保存退出
	ZQ	不保存退出
查找字符	/关键字	先按[/]键，再输入想查找的字符，如果第一次查找的关键字不是想要的，可以一直按[n]键会往后查找下一个关键字，而按[N]键会往相反的方向查找
	?关键字	先按[?]键，再输入想查找的字符，如果第一次查找的关键字不是想要的，可以一直按[n]键往前查找下一个关键字，而按[N]键会往相反的方向查找
合并	nJ	将当前行开始的n行进行合并，n代表数字
	J	清除光标所在行与下一行之间的换行，行尾没有空格的话会自动添加一个空格

末行模式命令（1）

类型	命令	功能
运行Shell命令	:!command	运行Shell命令，command代表命令
	:r!command	将命令运行的结果信息输入到当前行位置，command代表命令
	:n1,n2 w !command	将n1到n2行的内容作为命令的输入，n1和n2代表数字，command代表命令
查找字符	:/str/	从当前光标开始往右移动到有str的地方，str代表字符
	:?str?	从当前光标开始往左移动到有str的地方，str代表字符
替换字符	:s/str1/str2/	将光标所在行第一个字符str1替换为str2，str1和str2代表字符
	:s/str1/str2/g	将光标所在行所有的字符str1替换为str2，str1和str2代表字符
	:n1,n2s/str1/str2/g	用str2替换从第n1行到第n2行中出现的str1，str1和str2代表字符，n1和n2代表数字
	:% s/str1/str2/g	用str2替换文件中所有的str1，str1和str2代表字符
	:.,$s/str1/str2/g	将从当前位置到结尾的所有的str1替换为str2，str1和str2代表字符

末行模式命令（2）

类型	命令	功能
保存和退出	:w	保存文件
	:w filename	将文件另存为filename
	:wq	保存文件并退出vi编辑器
	:wq filename	将文件另存为filename后退出vi编辑器
	:wq!	保存文件并强制退出vi编辑器
	:wq! filename	将文件另存为filename后强制退出vi编辑器
	:x	保存文件并强制退出vi编辑器，其功能和:wq!相同
	:q	退出vi编辑器
	:q!	如果无法离开vi，强制退出vi编辑器
	:n1,n2w filename	将从n1行开始到n2行结束的内容保存到文件filename中，n1和n2代表数字
	:nw filename	将第n行内容保存到文件filename中，n代表数字
	:1,.w filename	将从第一行开始到光标当前位置的所有内容保存到文件filename中
	:.,$w filename	将从光标当前位置开始到文件末尾的所有内容保存到文件filename中
	:r filename	打开另外一个已经存在的文件filename
	:e filename	新建名为filename的文件
	:f filename	把当前文件改名为filename文件
	:/str/w filename	将包含有str的行写到文件filename中，str代表字符
	:/str1/,/str2/w filename	将从包含有str1开始到str2结束的行内容写入到文件filename中，str1和str2代表字符

末行模式命令（3）

类型	命令	功能
删除	:d	删除当前行
	:nd	删除第n行，n代表数字
	:n1,n2 d	删除从n1行开始到n2行为止的所有内容，n1和n2代表数字
	:.,$d	删除从当前行开始到文件末尾是所有内容
	:/str1/ , /str2/d	删除从str1开始到str2为止的所在行的所有内容，str1和str2代表字符
复制和移动	:n1,n2 co n3	将从n1行开始到n2行为止的所有内容复制到n3行后面，n1、n2和n3代表数字
	:n1,n2 m n3	将从n1行开始到n2行为止的所有内容移动到n3行后面，n1、n2和n3代表数字
跳到某一行	:n	在冒号后输入一个数字，再按回车键就会跳到该行，n代表数字
设置vi环境	:set number	在文件中的每一行前面列出行号
	:set nonumber	取消在文件中的每一行前面列出行号
	:set readonly	设置文件为只读状态