

# 第2章 Linux字符界面操作

宣江华  
宁波工程学院

# 本章学习目标

- 了解字符操作界面
- 掌握虚拟控制台和本地登录操作
- 理解Linux的运行级别
- 了解远程登录的方法
- 学会系统关机和重启的字符界面操作
- 掌握命令格式、文件和通配符
- 学会使用命令帮助

# 为什么使用字符工作方式

- 在Linux系统中，虽然有很多应用都使用图形界面，但是大多数使用和管理Linux的实用程序和技巧还是通过键入命令来运行的。
- 在字符操作方式下可以高效地完成所有的任务，尤其是系统管理任务。
- 系统管理任务通常在远程进行，而远程登录后进入的是字符工作方式。
- 由于使用字符界面不用启动图形工作环境，大大地节省了系统资源开销。

# 进入字符工作方式的方法

- 在系统启动后直接进入字符工作方式。
- 在图形环境下开启终端窗口进入字符工作方式。
- 使用远程登录方式（**Telnet** 或 **SSH**）进入字符工作方式。

# 1 字符界面登录与注销

- 虚拟控制台（**Virtual Console**）
  - 系统默认提供了**6**个虚拟控制台。
  - 每个虚拟控制台可以独立的使用，互不影响。
  - 使用**Alt+F1～Alt+F6**进行多个虚拟控制台之间的切换。
- 登录后的提示符
  - 超级用户登录后的操作提示符是“**#**”
  - 普通用户登录后的操作提示符是“**\$**”
- 本地注销
  - 使用**logout** 命令
  - 使用**<Ctrl>+<d>**快捷键

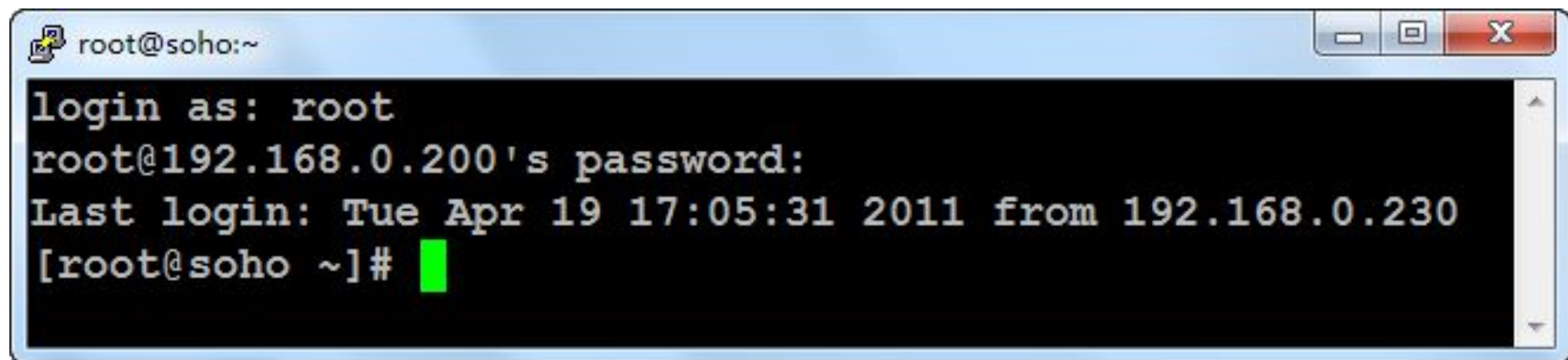
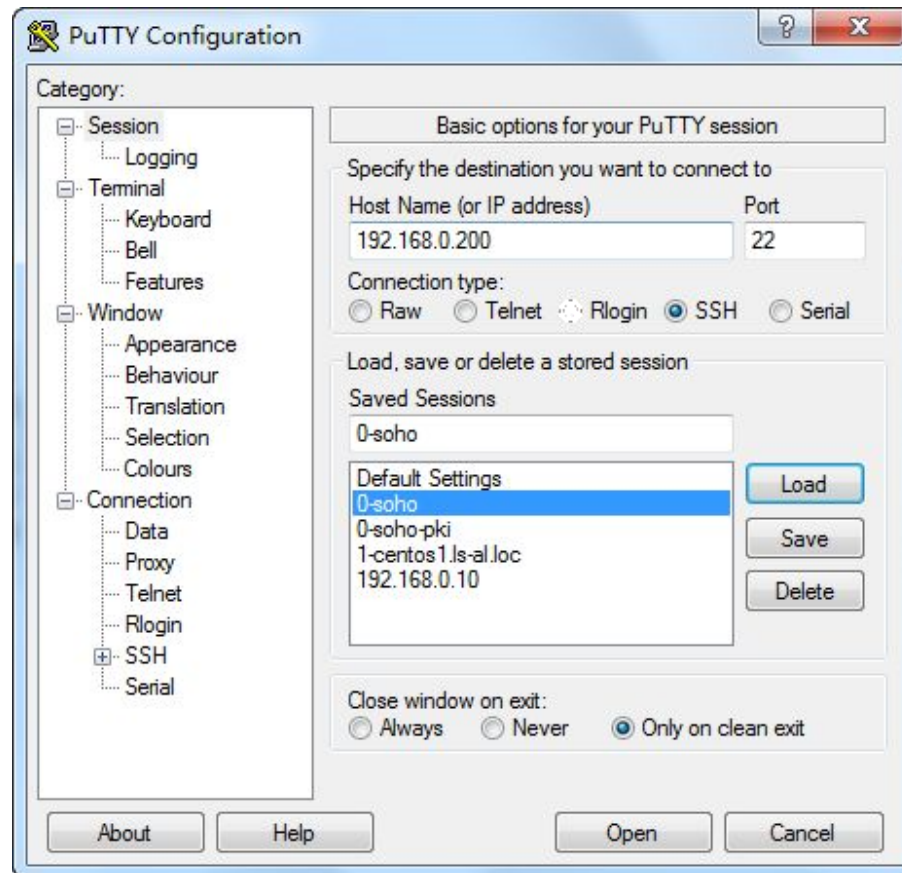
# 使用ssh登录远程Linux系统

- **SSH: Secure Shell**
- **Linux下的ssh命令是OpenSSH的客户端程序。**
- **CentOS 5 默认开启OpenSSH服务。**
- **默认没有安装Telnet服务，因为它不安全。**
- **在Linux环境下使用ssh登录远程Linux系统**
  - **\$ ssh 远程主机上的用户名@远程主机的IP地址**
  - **\$ ssh -l osmond 192.168.1.100**
  - **\$ ssh osmond@192.168.1.100**

# 使用ssh登录远程Linux系统

- 例如：// 以root身份登录IP地址为192.168.1.19的Linux系统
- **# ssh root@192.168.1.19**  
The authenticity of host '192.168.1.19 (192.168.1.19)' can't be established.  
RSA key fingerprint is 51:11:9c:e3:fa:d5:c7:e5:fc:0b:76:f1:c4:9e:03:fd.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.1.19' (RSA) to the list of known hosts.
- root@192.168.1.19's password:
- // 在此输入用户口令，口令输入过程中没有回显
- Last login: Wed Apr 30 02:15:08 2003 from 192.168.1.19
- #
- // 正确登录后出现Shell提示符

## 在Windows下 使用putty登录远 程Linux系统





# CentOS的系统运行级别

运行级别	说明
<b>0</b>	系统停机状态
<b>1</b>	单用户工作状态
<b>2</b>	多用户状态（没有NFS）
<b>3</b>	多用户状态（有NFS）
<b>4</b>	系统未使用
<b>5</b>	多用户模式，并且在系统启动后运行X Window
<b>6</b>	系统正常关闭并重新启动

# 运行级的查看和切换

- 查看当前系统的运行级
  - `runlevel`
- 切换运行级
  - `init [0123456Ss]`
- 修改默认运行级别
  - 编辑配置脚本 `/etc/inittab`
    - `id:3:initdefault:` ——启动后进入字符界面
    - `id:5:initdefault: :` ——启动后进入图形界面

# Red Hat的系统运行级别

- 在/etc/inittab中指示

# Default runlevel. The runlevels used by RHS are:

# 0 - halt (Do NOT set initdefault to this)

# 1 - Single user mode

# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)

# 3 - Full multiuser mode

# 4 - unused

# 5 - X11

# 6 - reboot (Do NOT set initdefault to this)

// 定义默认的运行级别

**id:3:initdefault:**

// 系统初始化

**si::sysinit:/etc/rc.d/rc.sysinit**

// 不同级别的脚本调用

**l0:0:wait:/etc/rc.d/rc 0**

**l1:1:wait:/etc/rc.d/rc 1**

**l2:2:wait:/etc/rc.d/rc 2**

**l3:3:wait:/etc/rc.d/rc 3**

**l4:4:wait:/etc/rc.d/rc 4**

**l5:5:wait:/etc/rc.d/rc 5**

**l6:6:wait:/etc/rc.d/rc 6**

// 跟踪CTRL-ALT-DELETE三键重启

// 基于安全考虑可以在下面的行首加#禁用此功能

**ca::ctrlaltdel:/sbin/shutdown -t3 -r now**

```
// 当系统发现UPS电源故障后2分钟后执行关机操作
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System
    Shutting Down"
# 如果在2分钟之内UPS恢复正常则取消关机操作
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored;
    Shutdown Cancelled"
# 在2、3、4、5运行级别中启动6个虚拟控制台
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
# 在5运行级别中启动xdm管理器 x:5:respawn:/etc/X11/prefdm -
    nodaemon
```

# 关机与重新启动

- 关机与重新启动相当于系统运行级别的切换
- 系统关机
  - # init 0
  - # halt
  - # poweroff
- 系统重启
  - # init 6
  - # reboot
- 使用shutdown命令

# 系统的启动和关闭

- 1、名称：halt
  - 使用权限：系统管理者
  - 使用方式：halt [-n] [-w] [-d] [-f] [-i] [-p]
  - 说明：若系统的runlevel 为0 或6 ， 则关闭系统， 否则以shutdown 指令（加上-h 参数）来取代
- 参数：
  - -n : 在关机前不做将记忆体资料写回硬盘的动作
  - -w : 并不会真的关机， 只是把记录写到/var/log/wtmp 档案里
  - -d : 不把记录写到/var/log/wtmp 档案里（-n 这个参数包含了-d） -f : 强迫关机， 不呼叫shutdown 这个指令
  - -i : 在关机之前先把所有网络相关的装置先停止
  - -p : 当关机的时候， 顺便做关闭电源（poweroff）的动作
- 范例：
  - halt -p 关闭系统后关闭电源。
  - halt -d 关闭系统， 但不留下纪录

# 系统的启动和关闭

## ■ 2、名称：init

- 使用权限：开机时
- 使用方式：init [0123456Ss]
- 说明：init 是所有行程（**process**）的父行程（**parent**），开机时一定会先从这个指令开始（可以用**ps -aux** 看看第一个行程就是init），并参考 **/etc/inittab** 档来完成整个开机程序，共有八个执行层级（**runlevel**），而改变只能透过**telinit** 的指令来更改



# 系统的启动和关闭

- 3、名称：reboot
  - 使用权限：系统管理者
  - 使用方式：reboot [-n] [-w] [-d] [-f] [-i]
  - 说明：若系统的runlevel 为0 或6 ，则重新开机，否则以shutdown 指令（加上-r 参数）来取代
- 参数：
  - -n：在重开机前不做将记忆体资料写回硬盘的动作
  - -w：并不会真的重开机，只是把记录写到/var/log/wtmp 档案里
  - -d：不把记录写到/var/log/wtmp 档案里（-n 这个参数包含了-d） -f：强迫重开机，不呼叫shutdown 这个指令
  - -i：在重开机之前先把所有网络相关的装置先停止
- 范例：
  - reboot 重开机。
  - reboot -w 做个重启机的模拟（只有纪录并不会真的重开机）。

# 系统的启动和关闭

## ■ 4、名称: shutdown

- 使用权限: 系统管理者
- 使用方式: **shutdown [-akrhHPfnc] [-t secs] time [warning message]**
- 说明: **shutdown** 可以用来进行关机程序, 并且在关机以前传送讯息给所有使用者正在执行的程序,
- **shutdown** 也可以用来重开机。
- **\*\* the "time" argument is mandatory! (try "now")**  
**\*\***

# 系统的启动和关闭

- 参数:

- **-t seconds** : 设定在几秒钟之后进行关机程序
- **-k** : 并不会真的关机, 只是将警告讯息传送给所有只用户
- **-r** : 关机后重新开机
- **-h** : 关机后停机
- **-n** : 不采用正常程序来关机, 用强迫的方式杀掉所有执行中的程序后自行关机
- **-c** : 取消目前已经进行中的关机动作
- **-f** : 关机时, 不做**fsck** 动作(检查Linux 档系统)
- **-F** : 关机时, 强迫进行**fsck** 动作
- **time** : 设定关机的时间 (一般用**now**)
- **message** : 传送给所有使用者的警告讯息

- 立即重启/关机
  - **shutdown -r now**
  - **shutdown -h now**
- 设置时间点关机
  - **shutdown -h 12:30**
- 几分钟后重启
  - **shutdown -r +5 "reboot after 5 minutes"**
- 向所有登录的用户发出信息
  - **shutdown -k now "hello,everyone"**
- 取消shutdown
  - **shutdown -c**

# Shell 简介

## ■ 什么是Shell

- **Shell**是系统的用户界面，提供了用户与内核进行交互操作的一种接口(命令解释器)。
- **Shell**接收用户输入的命令并把它送入内核去执行。
- **Shell**起着协调用户与系统的一致性和在用户与系统之间进行交互的作用。

# Shell 简介

- **Shell的主要功能**
  - 命令解释器
  - 命令通配符
  - 命令补全
  - 别名机制
  - 命令历史
  - 重定向、管道
  - 命令替换
  - 命令执行顺序、进程控制
  - **Shell脚本编程**

# Shell的主要版本

<b>Bash (Bourne Again Shell)</b>	<p><b>bash</b>是大多数Linux系统的默认Shell。 <b>bash</b>与<b>bsh</b>完全向后兼容，并且在<b>bsh</b>的基础上增加和增强了很多特性。 <b>bash</b>也包含了很多<b>C Shell</b>和<b>Korn Shell</b>中的优点。 <b>bash</b>有很灵活和强大的编程接口，同时又有很友好的用户界面</p>
<b>Ksh (Korn Shell)</b>	<p><b>Korn Shell (ksh)</b> 由<b>Dave Korn</b>所写。它是<b>UNIX</b>系统上的标准Shell。 在<b>Linux</b>环境下有一个专门为<b>Linux</b>系统编写的<b>Korn Shell</b>的扩展版本，即<b>Public Domain Korn Shell (pdksh)</b>。</p>
<b>tcsh (csh 的扩展)</b>	<p><b>tcsh</b>是<b>C Shell</b>的扩展。<b>tcsh</b>与<b>csh</b>完全向后兼容，但它包含了更多的使用户感觉方便的新特性，其最大的提高是在命令行编辑和历史浏览方面</p>

# Linux的元字符

- 在 **Shell** 中有一些具有特殊的意义字符，称为 **Shell 元字符（shell metacharacters）**。
- 若不以特殊方式（使用转义字符）指明，**Shell**并不会把它们当做普通文字符使用。

字符	含义	字符	含义
'	强引用	*, ?, !	通配符
"	弱引用	<, >, >>	重定向
\	转义字符	-	选项标志
\$	变量引用	#	注释符
;	命令分离符	空格、换行符	命令分隔符



# Linux 的命令格式

- 一般格式：
  - **cmd [options] [arguments]**
- 说明：
  - 最简单的**Shell**命令只有命令名，复杂的**Shell**命令可以有多个选项和参数。
  - 选项和参数都作为**Shell**命令执行时的输入，它们之间用空格分隔开。

**注：Linux 区分大小写！**

# Linux系统中 可执行文件的分类

- 内置命令：出于效率的考虑，将一些常用命令的解释程序构造在**Shell**内部
- 外置命令：存放在/bin、/sbin目录下的命令
- 实用程序：存放在/usr/bin、/usr/sbin、/usr/share、/usr/local/bin等目录下的实用程序
- 用户程序：用户程序经过编译生成可执行文件后，可作为**Shell**命令运行
- **Shell**脚本：由**Shell**语言编写的批处理文件，可作为**Shell**命令运行

## 命令基本格式（续）

### ■ 说明：

- 单字符参数前使用一个减号（-）
- 单词参数前使用两个减号（--）。
- 多个单字符参数前可以只使用一个减号。
- 操作对象（**arguments**）可以是文件也可以是目录，有些命令必须使用多个操作对象，如**cp**命令必须指定源操作对象和目标操作对象。
- 并非所有命令的格式都遵从以上规则，例如**dd**、**find**等

命令在正常执行结果后返回一个 0 值，如果命令出错，则返回一个非零值 (在**shell**中可用变量 **\$?** 查看)。

# 命令基本格式举例

- `$ ls`
- `$ ls -lRa /home`
- `$ cat abc xyz`
- `$ ls --help`
- `$ su -`
- `$ passwd`
- `$ date`
- `$ cal 2011`

## ■ 命令补齐

### ■ Command-Line Completion

- 当键入的字符足以确定目录中一个唯一的文件时，只须按 **Tab** 键就可以自动补齐该文件名的剩下部分
- 如果不唯一，按两下**TAB**列出可能的情况

- 命令历史
- 用上下方向键、**PgUp**和**PgDn**键来查看历史命令
- 可以使用键盘上的编辑功能键对显示在命令行上的命令进行编辑
- 使用**history**命令查看命令历史
  - **\$ history**
  - **\$ history 5**      显示最后 5 个命令
  - **\$ !20**              直接运行第**20**条历史命令
  - **\$ history -c**      清除历史命令

# 特殊字符

- 在Linux系统的终端中有几个最有用的bash变量，这些变量变量名及简单描述如下：
  - HISTFILE: 用于贮存历史命令的文件。
  - HISTSIZE: 历史命令列表的大小。
  - HOME: 当前用户的用户目录。
  - OLDPWD: 前一个工作目录。
  - PATH: bash寻找可执行文件的搜索路径。
  - PS1: 命令行的一级提示符。
  - PS2: 命令行的二级提示符。
  - PWD: 当前工作目录。
  - SECONDS: 当前shell开始后所流逝的秒数。
- 可以用**set**查看

命令别名通常是其他命令的缩写，用来减少键盘输入。  
还有一个使工作变得轻松的方法是使用命令别名  
命令格式为：

**alias** [alias-name='original-command']

其中，alias-name是用户给命令取的别名，original-command是原来的命令和参数。需要注意的是，由于Bash是以空格或者回车来识别原来的命令的，所以如果不使用引号就可能导致Bash只截取第一个字，从而出现错误。



- 命令别名
  - **alias命令和unalias命令**
  - **alias [alias\_name='original\_command']**
  - **unalias alias\_name**
- 使用举例
  - **\$ alias**
  - **\$ alias type='cat'**
  - **\$ unalias type**
  - **\$ alias ll='ls -l'**
  - **\$ alias ls='ls --color'**

注意：在定义别名时，等号两边不能有空格，否则shell将不能决定要做什么。仅在命令中包含空格或特殊字符时才需要引号。如果键入不带任何参数的alias命令，将显示所有已定义的别名。

Bash有两级提示符。第一级提示符是经常见到的Bash在等待命令输入时的情况。第一级提示符的默认值是\$符号。如果用户不喜欢这个符号，或者愿意自己定义提示符，只需修改PS1变量的值。例如将其改为：

PS1="输入一个命令："

第二级提示符是当Bash为执行某条命令需要用户输入更多信息时显示的。第二级提示符默认为">".如果需要自己定义该提示符，只需改变PS2变量的值。例如将其改为：

PS2="更多信息："

用户也可以使用一些事先已经定义好的特殊字符。这些特殊字符将使提示符中包含当前时间之类的信息。

表2-1列出了最常用的一些特殊字符及其含义。

**表2-1 bash提示符常用特殊字符**

特殊字符	说明
\!	显示该命令的历史编号
\#	显示 <b>shell</b> 激活后，当前命令的历史编号
\\$	显示一个\$符号，如果当前用户是 <b>root</b> 则显示#符号
\\	显示一个反斜杠
\d	显示当前日期
\h	显示运行该 <b>shell</b> 的计算机主机名
\n	打印一个换行符，这将导致提示符跨行
\s	显示正在运行的Shell的名称
\t	显示当前时间
\u	显示当前用户的用户名
\W	显示当前工作目录基准名
\w	显示当前工作目录

下面来看几个实际例子：

PS1="\"d"

将使提示符变成显示日期，如下所示：

一 2月 02

而 PS1="\"d"，将使提示符变成如下所示：

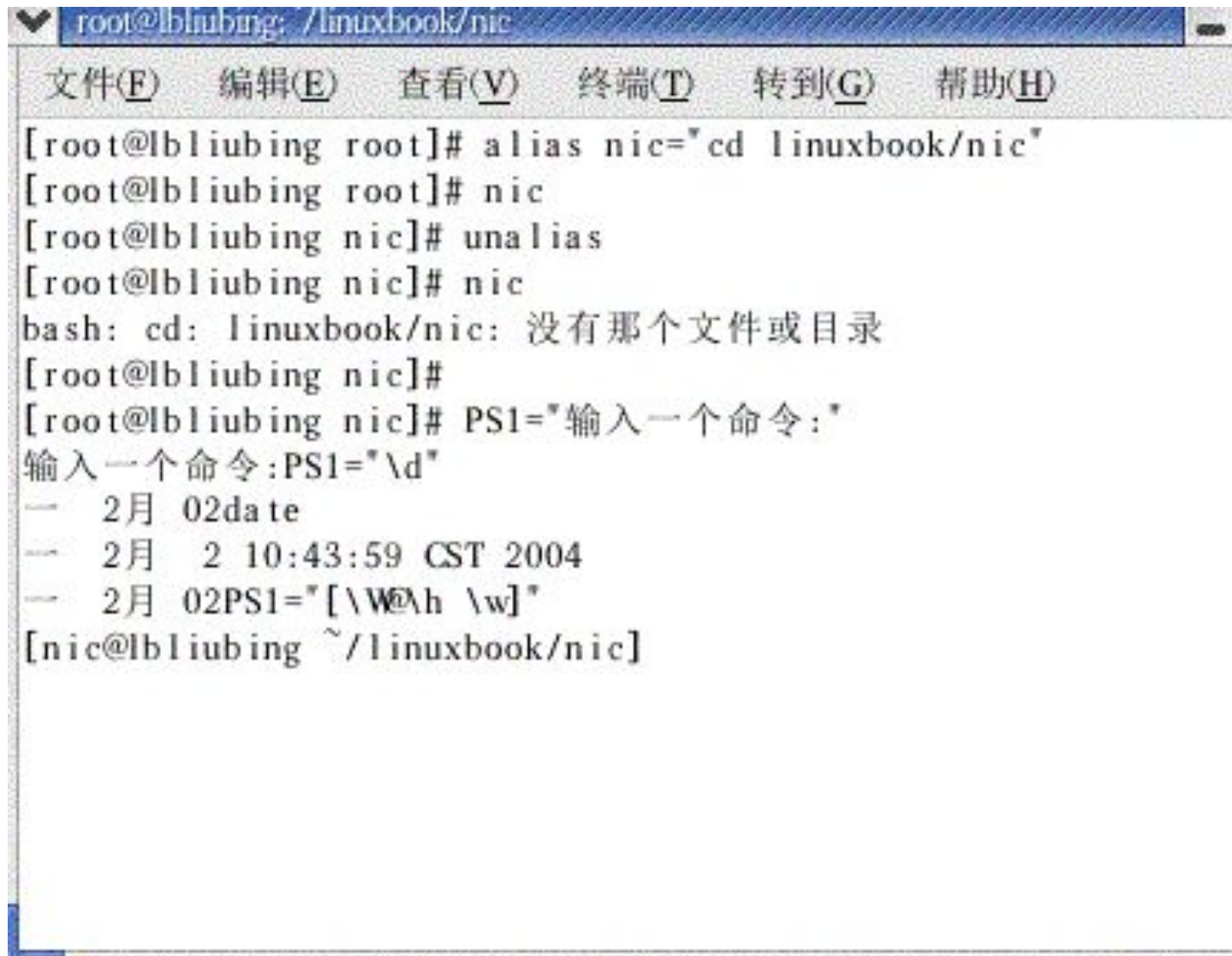
d

若PS1="\"[\W@\h \w]"将使提示符变成如下所示：

[nic@1bliubing ~/liunuxbook/nic]

该例就是使用三个特殊字符的组合得到的。

以上的各例在系统的终端中所显示的结果如图2-4所示。



```
root@bliubing: /linuxbook/nic
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
[root@bliubing root]# alias nic="cd linuxbook/nic"
[root@bliubing root]# nic
[root@bliubing nic]# unalias
[root@bliubing nic]# nic
bash: cd: linuxbook/nic: 没有那个文件或目录
[root@bliubing nic]#
[root@bliubing nic]# PS1="输入一个命令:"
输入一个命令:PS1="\d"
-- 2月 02date
-- 2月  2 10:43:59 CST 2004
-- 2月 02PS1="[\W@h \w]"
[nic@bliubing ~/linuxbook/nic]
```

# 通配符基础

通配符：

\*：匹配任何字符和任何数目的字符

？：匹配单一数目的任何字符

[ ]：匹配[ ]之内的任意一个字符

[! ]：匹配除了[! ]之外的任意一个字符，!表示非的意思

例如

[cChH]：表示在文件的该位置中可出现任意单个的c或h字符的大小写形式。

另个，通配符集还能描述介于字符对之间的所有字符。如“[a-z]”就可以代替任意小写字母，而[a-zA-Z]则可替代任意字母。注意可替代的字符包括a到z和A到Z字符对之间的所有字符。

# 通配符使用举例

- **ls \*.c**
  - 列出当前目录下的所有C语言源文件。
- **ls /home/\*/\*.c**
  - 列出/home目录下所有子目录中的所有C语言源文件。
- **ls n\*.conf**
  - 列出当前目录下的所有以字母n开始的conf文件。
- **ls test?.dat**
  - 列出当前目录下的以test开始的，随后一个字符是任意的.dat文件。
- **ls [abc]\***
  - 列出当前目录下的首字符是a或b或c的所有文件。
- **ls [!abc]\***
  - 列出当前目录下的首字符不是a或b或c的所有文件。
- **ls [a-zA-Z]\***
  - 列出当前目录下的首字符是字母的所有文件



在一条指令中用多个通配符，如

```
rm a*out*temp?
```

该命令可以删除一系列临时性的输出文件，如 ab.out.temp1、ab.out.temp1 等。

UNIX或Linux系统可将一定相关的文件看作一个集合的一部分，用户可以用该集合去匹配。 所以，如果需要显示nic-1.png, nic-2.png, nic-3.png, nic-4.png, nic-5.png, 只须要在终端的命令提示符后输入：

```
ls nic-[1-5].png
```

这样利用通配符可以使指令的输入变得更加灵活。该技巧的可很容易实现显示一些文件名相关的文件。以上的各例在系统的终端中所显示的结果如图2-5所示。

```
root@lbliubing: /linuxbook/nic
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
[root@lbliubing root]# ls
2-4png  2.txt~      BASH_文件      install.log.syslog
2.htm   anaconda-ks.cfg  chznwb-1.0.tar.gz  linuxbook
2.txt   BASH.htm       install.log      Mail
[root@lbliubing root]# ls 2*
2-4png  2.htm  2.txt  2.txt~
[root@lbliubing root]# cd linuxbook/nic
[root@lbliubing nic]# ls
nic-1.png  nic-2.png  nic-3.png  nic-4.png  nic-5.png  nic-6.png  nic-7
[root@lbliubing nic]# ls nic-[1-5].png
nic-1.png  nic-2.png  nic-3.png  nic-4.png  nic-5.png
[root@lbliubing nic]#
```

# 获得Linux的帮助

- 字符界面
  - 使用**help**获得**bash**的内部命令帮助
  - 使用**man**命令获得手册页帮助
  - 使用**info**命令获得**texinfo**文档帮助
  - 使用**pinfo**命令获得**texinfo**文档帮助
- **GNOME**桌面环境下
  - 使用**yelp**浏览帮助文档

## 联机帮助

### 1. man: 显示帮助手册

通常使用者只要在命令`man`后，输入想要获取的命令的名称（例如`ls`），`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。该命令的一般形式为：

`man`    [选项]    命令名称

表2-13 `man`命令的常用选项

常用选项	说明
<code>-S</code>	根据章节显示，由于一个命令名称可能会有很多类别，其类别说明如表2-14所示。
<code>-f</code>	只显示出命令的功能而不显示其中详细的说明文件
<code>-w</code>	不显示手册页，只显示将被格式化和显示的文件所在位置。
<code>-a</code>	显示所有的手册页，而不是只显示第一个。
<code>-E</code>	在每行的末尾显示\$符号

**表2-13 man命令的章节常用选项**

章节	说 明
1	一般使用者的命令
2	系统调用的命令
3	C语言函数库的命令
4	有关驱动程序和系统设备的解释
5	配置文件的解释
6	游戏程序的命令
7	其它的软件或是程序的命令和有关系统维护的命令

## 2. help: 系统帮助文档

help命令用于查看所有Shell命令。用户可以通过该命令寻求Shell命令的用法，只需在所查找的命令后输入help命令，就可以看到所查命令的内容了。

例如：查看cd命令的使用方法。

```
$ cd --help
```

## 3. whereis命令

这个程序的主要功能是寻找一个命令所在的位置。例如，最常用的ls命令，它是在/bin这个目录下的。如果希望知道某个命令存在哪一个目录下，可以用whereis命令来查询。该命令的一般形式为： whereis [选项] 命令名。

说明：一般直接使用不加选项的whereis命令，但用户也可根据特殊需要选用它的一些选项。该命令中各选项的说明如表2-14所示。

**表2-13 whereis命令的常用选项**

章节	说 明
<b>-b</b>	只查找二进制文件
<b>-m</b>	查找主要文件
<b>-s</b>	查找来源
<b>-u</b>	查找不常用的记录文件

例如：查找cd命令二进制文件在什么目录下。可使用如下命令：

```
$ whereis -b cd
```

#### **4、locate**

通过使用文件名数据库查找，所以速度比前面方式快

# 字符界面下的帮助

## ■ Wh\*命令

- \$ whatis ls
- \$ whereis ls
- \$ which ls

## ■ Man命令

- \$ man passwd
- \$ man 5 passwd
- \$ man -k selinux

注：退出 man 或 info 按 q 即可



# VI编辑器

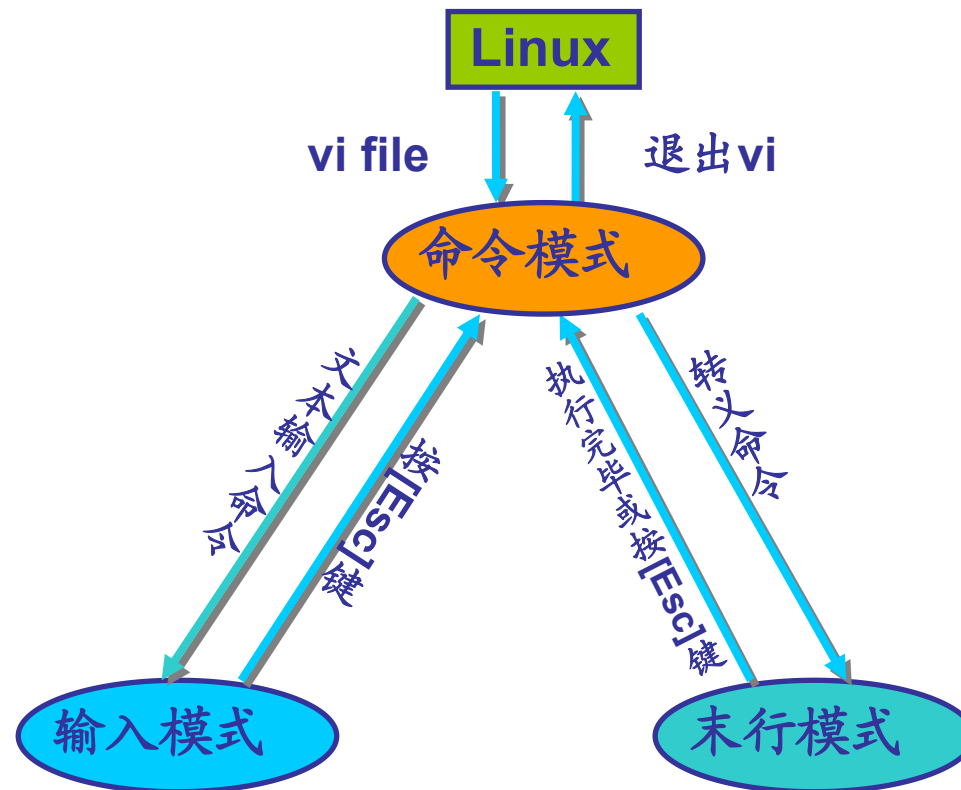
## ■ 介绍

- vi是“visual interface”的简称。
- vim: Vi IMproved
- vi几乎是所有“黑客”所使用的标准UNIX编辑器。
- 从2006年开始，作为“单一UNIX规范”（Single UNIX Specification）的一部分，vi或vi的一种变形版本一定会在UNIX中找到。

## ■ 特点

- 纯文本编辑器
- 全屏幕编辑器
- 工作于3种模式
- 通过命令进行编辑操作
- 必须重点掌握

# VI的三种模式



## ■ 命令模式



## 空白区

## 状态栏

## VI的模式 (Cont.)

## ■ 输入模式

```
These are the release notes for Linux version 2.4. Read them carefully,  
as they tell you what this is all about, explain how to install the  
kernel, and what to do if something goes wrong.
```

WHAT IS LINUX?

Linux is a Unix clone written from scratch by Linus Torvalds with  
assistance from a loosely-knit team of hackers across the Net.  
It aims towards POSIX compliance.

~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~

**输入模式标志**

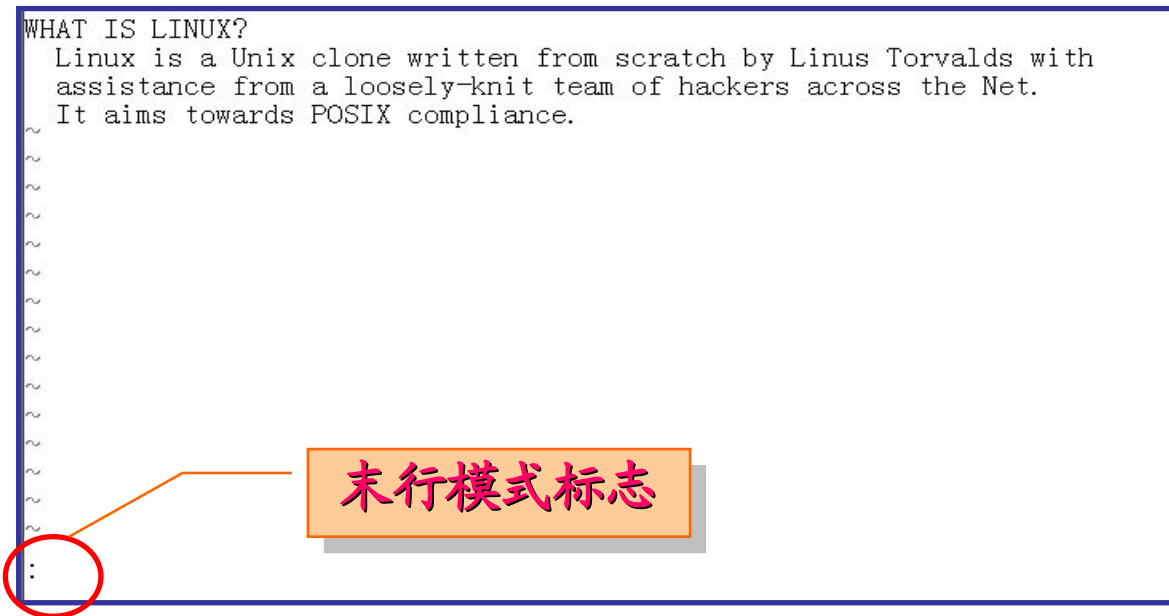
-- INSERT --

1, 1

All

## VI的模式 (Cont.)

## ■ 末行模式



# VI的基本命令

- 输入模式下的编辑操作命令
  - 进入vi, 进入编辑模式（输入i或者a），退出编辑模式（ESC）
- 末行模式下的编辑操作命令
  - 保存（:w），退出（:q），保存并退出（:wq），不保存退出（:q!）
- 命令模式下的编辑操作命令
  - 跳转到末行（G，:\$）。
  - 跳转到首行（gg）
  - 跳转到第n行（:n）
  - 查找(向下/str)、（向上?str）
  - 复制行（yy，5yy）删除行（dd）删除（Del）粘贴（p）
  - 撤销和重复 u .