AIX 5L 版本 5.2

系统用户指南：操作系统与设备

AIX 5L 版本 5.2

系统用户指南：操作系统与设备

注

在使用本资料及其支持的产品之前，请阅读第 213 页的『声明』中的一般信息。

第三版（2002 年 10 月）

本版本适用于 AIX 5L V5.2 和本产品的所有后续发行版，直到在新版本中另有声明为止。

目录

关于本书

本书包含想要获取操作系统更多专门知识的新系统用户需要的信息。它包括如运行命令、处理进程、处理文件和目录及打印的信息。此外，它还介绍一些任务，如保护文件、使用存储介质、定制环境文件（**.profile**、**.Xdefaults**、**.mwmrc**），以及写 shell 脚本。对于 DOS 用户，本指南显示有关在此环境中使用 DOS 文件的过程。

在网络环境中想要了解更多关于操作系统通信命令的用户应该阅读*《AIX 5L V5.2 系统用户指南: 通信与网络》*。

本书的适用对象

本书面向所有系统用户。

突出显示

本书中使用以下突出显示约定:

粗体	标识命令、关键字、文件、目录和系统预定义其名称的其它项。
斜体	标识将由用户提供其实际名称或值的参数。
等宽字体	标识特定数据值的示例，类似于您可能看到的已显示文本的示例，类似于您作为程序员可能写的程序代码片断的示例，来自系统的消息，或您实际应该输入的信息。

在 AIX 中区分大小写

所有 AIX 操作系统中的内容都区分大小写，即它分辨大写和小写字母。例如，您可使用 **ls** 命令列出文件。如果您输入 LS，则系统响应命令“未找到”。同样地，**FILEA**、**FiLea** 和 **filea** 是三个不同的文件名，即使它们驻留在同一个目录中。要避免不期望的操作被执行，请始终确保使用正确的大小写。

ISO 9000

本产品的开发和生产中使用了 ISO 9000 质量认证体系。

相关出版物

以下书籍包含有关的信息:

- *《AIX 5L V5.2 系统用户指南: 通信与网络》*
- *《AIX 5L V5.2 系统管理指南: 操作系统与设备》*
- *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*
- *AIX 5L Version 5.2 Guide to Printers and Printing*
- *《AIX 5L V5.2 命令参考大全》*
- *AIX 5L Version 5.2 Files Reference*

第 1 章 登录名、系统标识和密码

操作系统必须知道您的身份以便向您提供正确的环境。要将自己标识给操作系统，请通过输入您的*登录名*（也称为用户标识或用户名）和*密码*登录。密码是一种安全性形式。知道您登录名的人不能登录到您的系统，除非他们知道您的密码。

如果您的系统设置为多用户系统，则每个已授权用户将在系统上具有帐户、密码和登录名。操作系统跟踪每个用户使用的资源。这称为*系统记帐*。将给予每个用户在系统的存储空间中的专用区域，称为*文件系统*。您登录时，文件系统看起来只包含您的文件，尽管在系统上有成千上万的其它文件。

系统上可能有不止一个有效的登录名。如果想要从一个登录名更改为另一个登录名，则不必注销系统。而是可以在不同的 shell 中同步或在同一个 shell 中继续使用不同的登录名，而无须注销。此外，如果您的系统是与其它系统连接的网络的一部分，则可登录到您具有登录名的任何其它系统。这称为*远程登录*。

当完成在操作系统上工作时，请注销以确保您的文件和数据安全。

本章包含以下各节:

登录和注销概述

要使用操作系统，您的系统必须正在运行并且您必须登录。当您登录到操作系统时，您向系统标识您自己，并允许系统设置您的环境。

本节描述以下过程:

登录到操作系统

您的系统可能设置成只能在一天中的某些时段及一周中的某些天登录。如果尝试在非允许时间登录，则将拒绝您的访问。您的系统管理员可以验证您的登录时间。

在登录提示符下登录。当登录到操作系统时，您会自动置于主目录（也称为您的*登录目录*）中。

在系统打开后，请登录到系统以启动会话。

1. 在 login: 提示符后输入您的登录名，并按下 Enter 键:

   ```
   login: 登录名
   ```

   例如，如果您的登录名是 denise:

   ```
   login: denise
   ```

2. 如果出现 password: 提示符，请输入您的密码并按下 Enter 键。（屏幕显示的密码与您输入的密码不同。）

   ```
   password: [您的密码]
   ```

如果不出现密码提示符，则您没有定义密码；您可以在操作系统中开始工作了。

如果您的机器未打开，请在登录前执行以下操作:

1. 将每个所连接设备的电源开关设置为“开”。
2. 通过将电源开关设置为“开”（**I**）启动系统部件。
3. 查看三数字显示。当自检完成且无错误时，三数字显示是空白的。

如果发生一个需要关注的错误，则三数字代码保留，且系统部件停止。请向您的系统管理员查询，以获取关于错误代码和恢复的信息。

当自检成功完成时，在您的屏幕上显示类似于以下的登录提示符:

```
login:
```

在您登录后，取决于如何设置操作系统，系统将以命令行界面（shell）或图形界面（例如，AIXwindows 或公共台式机环境（CDE））启动。

如果您有关于密码或用户名配置的问题，请咨询您的系统管理员。

不止一次登录（login 命令）

如果您处理不止一个项目，并且想要维护独立的帐户，则可进行不止一次并行登录。这可以通过使用相同的登录名或通过使用不同的登录名登录到您的系统来实现。

> **注**: 每个系统有在任何给定时刻可活动的登录名的最大数目。该数目由您的许可证协议确定，并随安装而变化。

例如，如果您已作为 denise1 登录，而您的另一个登录名是 denise2，则在提示符下输入:

```
login denise2
```

如果显示 password: 提示符，请输入您的密码并按下 Enter 键。（屏幕显示的密码与您输入的密码不同。）现在您在系统上运行两个登录。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **login** 命令，以获取完整的语法。

成为系统上的另一个用户（su 命令）

通过使用 **su**（切换用户）命令，可以更改与会话关联的用户标识（如果知道该用户的登录名）。

例如，如果想要切换并成为用户 joyce，请在提示符下输入:

```
su joyce
```

如果显示 password: 提示符，请输入 joyce 的密码并按下 Enter 键。现在您的用户标识是 joyce。如果您不知道密码，则拒绝请求。

要验证您的用户标识是 joyce，请使用 **id** 命令。有关 **id** 命令的更多信息，请参阅第 6 页的『显示用户标识（id 命令）』。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **su** 命令，以获取完整的语法。

禁止登录消息

在成功登录后，**login** 命令显示该用户的最后一次成功登录尝试和未成功登录尝试的天、日期和时间的消息以及自认证信息（通常是密码）最后一次更改以来该用户的未成功登录尝试的总次数。可通过在您的主目录中包含 **.hushlogin** 文件来禁止这些消息。

在您主目录中的提示符下，请输入:

```
touch .hushlogin
```

touch 命令创建名为 **.hushlogin** 的空文件（如果它还不存在）。下次登录时，将禁止所有的登录消息。可以指示系统仅保留天的消息，而禁止其它登录消息。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **touch** 命令，以获取完整的语法。

注销操作系统（exit 和 logout 命令）

要注销操作系统，请在系统提示符处执行以下操作之一:

按下文件结束符控制按键序列（Ctrl-D 键）。

或

输入 exit 并按下 Enter 键。

或

输入 logout 并按下 Enter 键。

在注销后，系统显示 login: 提示符。

停止操作系统（shutdown 命令）

> **注意**: 不要在未先关机的情况下关闭系统。关闭系统结束在系统上运行的所有进程。如果其它用户正在系统上工作，或者如果作业正在后台运行，则数据可能会丢失。请在停止系统之前执行适当的关机过程。

如果具有 root 用户权限，则可以使用 **shutdown** 命令停止系统。如果未授权您使用 **shutdown** 命令，则只是注销操作系统并仍让它运行。

请在提示符处输入:

```
shutdown
```

当 **shutdown** 命令完成并且操作系统停止运行时，您接收到以下消息:

```
....关机完成....
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **shutdown** 命令，以获取完整的语法。

用户和系统标识

本节描述以下过程，可用于显示标识系统上的用户和您正在运行的系统的信息。

- 『显示您的登录名（whoami 和 logname 命令）』
- 第 5 页的『显示操作系统的名称（uname 命令）』
- 第 5 页的『显示您的系统的名称（uname 命令）』
- 第 5 页的『显示谁已登录（who 命令）』
- 第 6 页的『显示用户标识（id 命令）』

显示您的登录名（whoami 和 logname 命令）

当进行不止一个并行登录时，经常容易丢失对登录名或（特别是）您在此时使用的登录名的跟踪。

使用 whoami 命令

要确定使用的是哪个登录名，请在提示符下输入:

```
whoami
```

系统显示类似于以下的信息:

```
denise
```

在此示例中，使用的登录名是 `denise`。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **whoami** 命令，以获取完整的语法。

使用 who am i 命令

who 命令的变体，**who am i** 命令，允许您显示登录名、终端名和登录的时间。请在提示符下输入:

```
who am i
```

系统显示类似于以下的信息:

```
denise   pts/0   6 月 21 日 07:53
```

在此示例中，登录名是 denise，终端的名称是 pts/0，该用户登录的时间是 6 月 21 日上午 7:53。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **who** 命令，以获取完整的语法。

使用 logname 命令

who 命令的另一种变体 **logname** 命令显示与 **who** 命令相同的信息。

请在提示符处输入:

```
logname
```

系统显示类似于以下的信息:

```
denise
```

在此示例中，登录名是 denise。

显示操作系统的名称（uname 命令）

要显示操作系统的名称，请使用 **uname** 命令。

例如，在提示符处输入:

```
uname
```

系统显示类似于以下的信息:

```
AIX
```

在此示例中，操作系统名称是 AIX。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **uname** 命令，以获取完整的语法。

显示您的系统的名称（uname 命令）

当您在网络上时要显示您的系统的名称，请使用带 **-n** 标志的 **uname** 命令。您的系统名称向网络标识您的系统；它与您的登录标识不相同。

例如，在提示符处输入:

```
uname -n
```

系统显示类似于以下的信息:

```
barnard
```

在此示例中，系统名称是 barnard。

请参阅《*AIX 5L V5.2 命令参考大全*》一书中的 **uname** 命令，以获取完整的语法。

显示谁已登录（who 命令）

要显示有关当前在本地系统上的所有用户的信息，请使用 **who** 命令。显示以下信息：登录名、系统名以及登录的日期和时间。

注: 此命令仅标识本地节点上的用户。

要显示有关谁在使用本地系统节点的信息，请输入:

```
who
```

系统显示类似于以下的信息:

```
joe   lft/0  6 月 8 日 08:34
denise   pts/1  6 月 8 日 07:07
```

在此示例中，用户 joe，在终端 lft/0 上，在 6 月 8 日上午 8:34 登录。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **who** 命令，以获取正确的语法。

显示用户标识（ id 命令）

要显示指定的用户的系统标识（ID），请使用 **id** 命令。系统标识是向系统标识用户和用户组的数字。**id** 命令显示以下信息（当适用时）:

- 用户名和实际用户标识
- 用户组的名称和实际组标识
- 用户辅助组的名称和辅助组标识（如果有）

例如，在提示符处输入:

```
id
```

系统显示类似于以下的信息:

```
uid=1544(sah) gid=300(build) euid=0(root) egid=9(printq) groups=0(system),10(audit)
```

在此示例中，用户具有用户名 sah，标识号为 1544；主组名 build，标识号为 300；有效用户名 root，标识号为 0；有效组名 printq，标识号为 9；以及两个辅助组名 system 和 audit，标识号分别为 0 和 10。

例如，在提示符下，请输入:

```
id denise
```

系统显示类似于以下的信息:

```
uid=2988(denise) gid=1(staff)
```

在此示例中，用户 denise 具有标识号 2988，且只具有主组名 staff，标识号为 1。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **id** 命令，以获取完整的语法。

密码

您的系统将密码与每个帐户关联。唯一的密码为您的文件提供一些系统安全性。安全性是计算机系统的重要部分，因为它使未经授权的人不能获得对系统的访问和篡改其他用户的文件。安全性还能允许一些用户对其可以访问哪些命令以及可以访问哪些文件具有专有的访问权。出于保护目的，一些系统管理员仅允许用户访问某些命令或文件。

本节描述以下过程:

- 第 7 页的『密码准则』
- 第 7 页的『更改密码（passwd 命令）』
- 第 8 页的『设置密码为空（passwd 命令）』

密码准则

您应该具有唯一的密码。*密码应不应共享*。像您保护任何其它公司资产那样保护密码。创建密码时，请确保密码很难猜，但是也不要太难，以至于您必须写下来才能记住它们。

使用模糊的密码，可保持您的用户标识安全。基于个人信息（如您的姓名或生日）的密码都是不好的密码。甚至一般的字都很容易被猜到。

好的密码至少具有六个字符，并包含非字母字符。奇怪字的组合和故意拼错的字也是很好的选择。

注: 如果您的密码很难记住以至于必须将其写下来，那么它也不是好密码。

选择密码时使用以下准则:

- 不要使用您的用户标识作为密码。不要使用用户标识的反向、两遍用户标识，或另外修改的用户标识作为密码。
- 不要重新使用密码。系统可能设置为拒绝重新使用密码。
- 不要使用任何人的姓名作为密码。
- 不要使用联机拼写检查字典中可以找到的字作为密码。
- 不要使用短于六个字符的密码。
- 不要使用不雅的单词；它们是猜密码时一些最先选取的单词。
- 要使用容易记住的密码，这样您就不需要将其写下来了。
- 要使用既用字母和又用数字的密码，并且同时具小写和大写字母。
- 要使用两个字，中间用数字分开，作为密码。
- 要使用可发音的密码。它们便于记忆。
- 不要将密码写下来。但是，如果您必须将其写下来，请将密码放在一个物理上安全的地方，如锁定的橱柜。

更改密码（**passwd** 命令）

要更改您的密码，请使用 **passwd** 命令。

1. 在提示符下，请输入:

   ```
   passwd
   ```

   如果还没有密码，则跳过步骤 2。
2. 显示以下提示符:

   更改*用户标识*的密码
   *用户标识*的旧密码:

   该请求使未经授权的用户在您离开系统时不能更改您的密码。输入您的当前密码并按下 Enter 键。
3. 显示以下提示符:

   *用户标识*的新密码:

   输入您想要的新密码并按下 Enter 键。
4. 显示以下提示符，要求您重新输入新密码。

   再次输入新密码:

   该请求保护您不会将密码设置为无法重新创建的错误输入的字符串。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **passwd** 命令，以获取完整的语法。

设置密码为空（passwd 命令）

如果不想要每次登录时都输入密码，请将密码设置为空（空白）。

要将密码设置为空，请输入:

```
passwd
```

当提示您输入新密码时，请按下 Enter 键或 Ctrl-D。

passwd 命令不再次提示您需要密码输入。显示验证空密码的消息。

请参阅《*AIX 5L V5.2 命令参考大全*》书籍中的 **passwd** 命令，以获取更多信息和正确的语法。

登录名、系统标识和密码的命令摘要

登录和注销命令

login	启动会话
logout	停止所有进程
shutdown	结束系统操作
su	更改与会话关联的用户标识
touch	更新文件的访问和修改时间，或创建空文件

用户和系统标识命令

id	显示指定的用户的系统标识
logname	显示登录名。
uname	显示当前操作系统的名称
who	标识当前登录的用户
whoami	显示登录名

密码命令

passwd	更改用户的密码

相关信息

有关本主题的进一步信息，请参阅以下内容

- 第 25 页的第 4 章，『命令和进程』
- 第 113 页的第 10 章，『文件和系统安全性』
- 第 11 页的第 2 章，『用户环境和系统信息』
- 第 125 页的第 11 章，『定制用户环境』

相关信息

第 25 页的第 4 章, 『命令和进程』

第 113 页的第 10 章, 『文件和系统安全性』

第 11 页的第 2 章, 『用户环境和系统信息』

第 125 页的第 11 章, 『定制用户环境』

第 135 页的第 12 章, 『shell』

第 140 页的『Korn shell 或 POSIX shell 命令』

第 174 页的『Bourne shell』

第 188 页的『C shell』

第 2 章 用户环境和系统信息

每个登录名都具有其自己的系统环境。系统环境是一个区域，其中存储对运行在会话中的所有进程都是公共的信息。可以使用几个命令来显示有关您的系统的信息。

本章讨论用于显示关于您环境的信息的以下过程:

列出系统设备（lscfg 命令）

要显示名称、位置和在当前配置中找到的每个设备的描述，请使用 **lscfg** 命令。该列表按照设备位置排序。

例如，要列出在系统中配置的设备，请在提示符下输入:

```
lscfg
```

按下 Enter 键。

系统显示类似于以下的输出:

```
已安装的资源列表

以下资源已在您的机器上安装。

+/- = 从诊断测试列表添加 / 删除。
*   = 不为诊断支持。


  型号体系结构: chrp
  型号实现: 多处理器，PCI 总线

+  sysplanar0   00-00           CPU 平面
+  fpa0         00-00           浮点处理器
+  mem0         00-0A           内存卡
+  mem1         00-0B           内存卡
+  ioplanar0    00-00           I/O 平面
+  rs2320       00-01           RS232 卡
+  tty0         00-01-0-01      RS232 卡端口
-  tty1         00-01-0-02      RS232 卡端口
   ..
   ..
   ..
```

该设备列表不只根据设备位置排序。它由父 / 子层次结构排序。如果父具有多个子，则按照设备位置排序子。如果多个子具有同一个设备位置，则以由软件获取它们的顺序显示它们。要显示有关特定设备的信息，可以使用 **-l** 标志。例如，要列出有关设备 **sysplanar0** 的信息，请在提示符下输入:

```
lscfg -l sysplanar0
```

按下 Enter 键。

系统显示类似于以下的输出:

```
设备            位置          描述

sysplanar0      00-00         CPU 平面
```

还可以使用 **lscfg** 命令显示重要产品数据（VPD），如部件号、序列号和设计更改级别。对于一些设备，自动收集 VPD 并将其添加到系统配置。对于其它设备，请手工输入 VPD。数据前面的 ME 表明手工输入了数据。

例如，要列出在系统中配置的设备的 VPD，请在提示符下输入:

```
lscfg -v
```

按下 Enter 键。

系统显示类似于以下的输出:

```
VPD 的已安装资源列表

以下资源已在您的机器中安装。

  型号体系结构:
  型号实现: 多处理器，PCI 总线

sysplanar0   00-00   CPU 平面

   部件号.........342522
   EC 级别.........254921
   序列号.........353535

fpa0   00-00   浮点处理器
mem0   00-0A   内存卡

   EC 级别.........990221
.
.
.
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lscfg** 命令，以获取完整的语法。

显示控制台名称（lscons 命令）

要将当前控制台设备的名称写为标准输出（通常是您的屏幕），请使用 **lscons** 命令。

例如，在提示符下输入:

```
lscons
```

按下 Enter 键。

系统显示类似于以下的输出:

```
/dev/lft0
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lscons** 命令，以获取完整的语法。

显示终端名称（tty 命令）

要显示您的终端的名称，请使用 **tty** 命令。

例如，在提示符下输入:

```
tty
```

按下 Enter 键。

系统显示类似于以下的信息:

```
/dev/tty06
```

在此例中，tty06 是终端的名称，**/dev/tty06** 是包含至此终端接口的设备文件。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **tty** 命令，以获取正确的语法。

列出可用的显示（lsdisp 命令）

要列出您的系统上当前可用的显示，提供显示标识名称、插槽号码、显示名称以及每个显示的描述，请使用 **lsdisp** 命令。

例如，要列出所有可用的显示，请输入:

```
lsdisp
```

按下 Enter 键。

以下是输出的示例。该列表根据插槽编号以升序显示。

```
名称      插槽      名称          描述
ppr0    00-01    POWER_G4     中端图形适配器
gda0    00-03    colorgda     彩色图形显示适配器
ppr1    00-04    POWER_Gt3    中低端图形适配器
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lsdisp** 命令，以获取完整的语法。

列出可用的字体（lsfont 命令）

要显示可用于显示的字体的列表，请使用 **lsfont** 命令。

例如，要以列表格式列出可用于显示的所有字体，请输入:

```
lsfont
```

按下 Enter 键。

以下是输出的示例，显示字体标识、文件名、字型大小和字体编码:

```
字体    文件              字型   字体
标识    名称              大小    编码
====  ==============  =====  =========
0     Erg22.iso1.snf  12x30  ISO8859-1
1     Erg11.iso1.snf   8x15  ISO8859-1
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lsfont** 命令，以获取完整的语法。

列出当前软件键盘映射（lskbd 命令）

要显示装入到系统的当前软件键盘映射的绝对路径名，请使用 **lskbd** 命令。

例如，要列出当前键盘映射，请输入:

```
lskbd
```

按下 Enter 键。

以下是 **lskbd** 命令显示的列表的示例:

```
当前软件键盘映射 = /usr/lib/nls/loc/C.lftkeymap
```

列出可用的软件产品（lslpp 命令）

要显示有关可用于您的系统的软件产品的信息，请使用 **lslpp** 命令。

例如，要列出在系统中的所有软件产品，请在系统提示符下输入:

```
lslpp -l -a
```

按下 Enter 键。

以下是输出的示例:

```
文件集                 级别  状态      描述
--------------------  ------- --------  -----------------
路径: /usr/lib/objrepos
   X11_3d.gl.dev.obj           已应用    AIXwindows/3D GL
                                         开发实用程序
字体
  X11fnt.oldX.fnt              已应用    AIXwindows 各种
                                          X 字体
X11mEn_US.msg                  已应用    AIXwindows NL 消息
                                          文件

.
.
.
```

如果此列表很长，则顶部可能滚动到屏幕以外。要一次一页（屏幕）地显示该列表，请使用 **lslpp** 命令传递给 **pg** 命令。请在提示符下输入:

```
lslpp | pg
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lslpp** 命令，以获取完整的语法。

列出终端的控制键指定（stty 命令）

要显示您的终端设置，请使用 **stty** 命令。特别注意您的终端将哪些键用于控制键。

例如，在提示符下输入:

```
stty -a
```

按下 Enter 键。

系统显示类似于以下的信息:

```
.
.
.
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D;
eol = ^@ start = ^Q; stop = ^S; susp = ^Z; dsusp = ^Y;
reprint = ^R discard = ^O; werase = ^W; lnext = ^V
.
.
.
```

在此例中，诸如 `intr = ^C; quit = ^\; erase = ^H;` 这样的行是您的控制键设置. `^H` 键是退格键，并且它设置为执行擦除功能。

如果此列表很长，则顶部可能滚动到屏幕以外。要一次一页（屏幕）地显示该列表，请使用 **pg** 命令并传递给 **stty** 命令。请在提示符下输入:

```
stty -a | pg
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **stty** 命令，以获取完整的语法。

列出环境变量（env 命令）

在命令执行的开始时为命令知晓的所有变量（及其关联的值）组成其*环境*。此环境包含命令从其父进程继承的变量和在调用命令的命令行上作为关键字参数指定的变量。shell 以几种方法与环境交互。启动时，shell 扫描环境并为每个找到的名称创建一个参数，给参数相应的值并标记它以用于导出。执行的命令继承该环境。

要显示当前环境变量，请使用 **env** 命令。对所有您的进程可访问的环境变量称为*全局变量*。

例如，要列出所有环境变量，请输入:

```
env
```

按下 Enter 键。

以下是输出的示例:

```
TMPDIR=/usr/tmp
myid=denise
LANG=En_US
UNAME=barnard
PAGER=/bin/pg
VISUAL=vi
```

```
PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/u/bin1
MAILPATH=/usr/mail/denise?denise has mail !!!
MAILRECORD=/u/denise/.Outmail
EXINIT=set beautify noflash nomesg report=1 showmode showmatch
EDITOR=vi
PSCH=>
HISTFILE=/u/denise/.history
LOGNAME=denise
MAIL=/usr/mail/denise
PS1=denise@barnard:${PWD}>
PS3=#
PS2=>
epath=/usr/bin
USER=denise
SHELL=/bin/ksh
HISTSIZE=500
HOME=/u/denise
FCEDIT=vi
TERM=lft
MAILMSG=**您有新邮件。请使用 mail 命令查看您的 PWD=/u/denise
ENV=/u/denise/.env
```

如果此列表很长，则顶部滚动到屏幕以外。要一次一页（屏幕）地上显示该列表，请使用 **env** 命令并传递给 **pg** 命令。请在提示符下输入:

```
env | pg
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **env** 命令，以获取完整的语法。

显示环境变量的值（ printenv 命令 ）

要显示环境变量的值，请使用 **printenv** 命令。如果指定 *Name* 参数，则系统仅打印与您请求的参数关联的值。如果不指定 *Name* 参数，则 **printenv** 命令显示所有当前环境变量，每行显示一个 *Name* **=***Value* 序列。

例如，要查找 **MAILMSG** 环境变量的当前设置，请输入:

```
printenv MAILMSG
```

按下 Enter 键。

命令返回 **MAILMSG** 环境变量的值。例如:

```
您有新邮件
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **printenv** 命令，以获取完整的语法。

使用双向语言（ aixterm 命令 ）

aixterm 命令支持阿拉伯语和希伯来语，它们都是双向语言。双向语言能从两个方向读和写，如从左到右和从右到左。可以通过打开指定阿拉伯语或希伯来语语言环境的窗口来使用阿拉伯语和希伯来语应用程序。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **aixterm** 命令，以获取完整的语法。

用户环境和系统信息的命令摘要

aixterm	使您能使用双向语言
env	显示当前环境或设置执行命令的环境
lscfg	显示有关设备的诊断信息
lscons	显示当前控制台的名称
lsdisp	列出系统上当前可用的显示
lsfont	列出显示使用的可用字体
lskbd	列出系统中当前装入的键盘映射
lslpp	列出软件产品
printenv	显示环境变量的值
stty	显示系统设置
tty	显示终端的全路径名

相关信息

- 第 25 页的第 4 章, 『命令和进程』
- 第 45 页的第 5 章, 『输入和输出重定向』
- 第 4 页的『用户和系统标识』
- 第 125 页的第 11 章, 『定制用户环境』

第 3 章 公共台式机环境

使用“公共台式机环境”，可访问网络连接的设备和工具，而不必知道它们的位置。只要拖放对象就可以在应用程序间交换数据。

系统管理员发现在各个平台上，以前需要复杂命令行语法的许多任务现在变得更容易和简单了。他们还可以通过集中配置和分发应用程序给用户来最大化他们在现有硬件和软件的投资。他们能为其支持的用户集中管理应用程序的安全性、可用性和互操作性。

注: 公共台式机环境（CDE）1.0。帮助卷、基于 web 的文档以及硬拷贝手册可能将桌面称为“公共台式机环境”、“AIXwindows 桌面”、“公共台式机环境”、CDE 1.0 或简单地称为桌面。

本章包含的主题是:

启动和停止公共台式机环境

可以将系统设置为在您启动系统时“公共台式机环境”自动启动，或者可以手工启动“公共台式机环境”。必须作为 root 用户登录以执行这些任务中的每一个。

启用和禁用桌面自动启动

可能会发现将系统设置为在系统打开时自动启动“公共台式机环境”更为方便。通过“基于 Web 的系统管理器”（输入 `wsm`，然后选择系统）、通过系统管理界面程序（SMIT）或通过命令行都可以实现该操作。

先决条件

您必须具有 root 用户权限以启用或禁用桌面自动启动。

自动启动／停止公共台式机环境任务

任务	*SMIT 快速路径*	*命令或文件*
启用桌面自动启动 [1]	**smit dtconfig**	**/usr/dt/bin/dtconfig -e**
禁用桌面自动启动 [1]	**smit dtconfig**	**/usr/dt/bin/dtconfig -d**

[1]**注**: 完成此任务后重新启动机器。

手工启动公共台式机环境

您可以手工启动“公共台式机环境”。

手工启动桌面登录管理器

1. 作为 root 用户登录到您的系统。
2. 在命令行上输入:

   ```
   /usr/dt/bin/dtlogin -daemon
   ```

显示**桌面登录**屏幕。当登录时，必须启动桌面会话。

手工停止公共台式机环境

您可以手工停止“公共台式机环境”。

手工停止登录管理器

当手工停止登录管理器时，由登录管理器启动的所有 X 服务器和桌面会话停止。

1. 打开终端仿真器窗口并作为 root 用户登录。
2. 通过输入以下内容获取“登录管理器”的进程标识:

   ```
   cat /var/dt/Xpid
   ```

3. 通过输入以下内容停止“登录管理器”:

   ```
   kill -term process_id
   ```

修改桌面概要文件

当用户登录到桌面时，不自动读 shell 环境文件（**.profile** 或 **.login**）。在用户登录之前，桌面运行 X 服务器，因此 **.profile** 文件或 **.login** 文件提供的功能必须由桌面的登录管理器提供。

特定于用户的环境变量在 */Home Directory/* **.dtprofile** 中设置。此文件的模板位于 **/usr/dt/config/sys.dtprofile** 中。将变量和 shell 命令放入只应用于桌面的 **.dtprofile** 中。将行添加到 **.dtprofile** 的结尾以合并 shell 环境文件。

系统范围环境变量可在“登录管理器”配置文件中设置。有关配置环境变量的详细信息，请参阅 *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*。

为公共台式机环境添加和除去显示和终端

登录管理器可从具有单个本地位图或图形控制台的系统启动。然而，许多其它情况也是可能的（见下图）。可以从以下位置启动“公共台式机环境”:

- 本地控制台
- 远程控制台
- 位图和字符显示
- 在网络上的主机系统上运行的 Xterminal 系统

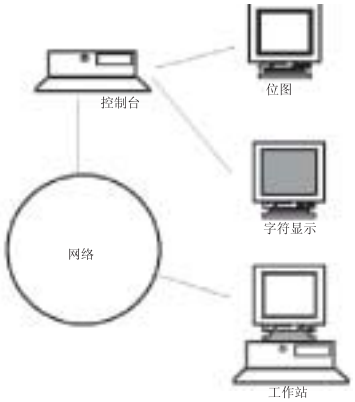

图 1. CDE 接口点. 本图显示控制台、网络、位图显示、字符显示和工作站之间的连接点。

X 终端系统由显示设备、键盘和只运行 X 服务器的鼠标组成。客户机（包含“公共台式机环境”）在网络上的一个或多个主机系统上运行。来自客户机的输出定向到 X 终端显示。

以下登录管理器配置任务支持很多可能的配置。

- 『除去本地显示』
- 『添加 ASCII 或字符显示终端』

将工作站用作 X 终端

从命令行输入:

```
/usr/bin/X11/X -query hostname
```

作为 X 终端的工作站的 X 服务器必须:

- 支持 XDMCP 和 **-query** 命令行选项。
- 向终端主机提供 xhost 许可权（在 **/etc/X*.hosts** 中）。

除去本地显示

要除去本地显示，请除去其在 **/usr/dt/config** 目录中 Xservers 文件中的条目。

添加 ASCII 或字符显示终端

字符显示控制台是控制台不是位图设备的配置。

位图显示不存在时添加 ASCII 或字符显示控制台

1. 如果 **/etc/dt/config/Xservers** 文件不存在，请将 **/usr/dt/config/Xservers** 文件复制到 **/etc/dt/config** 目录。
2. 如果您必须将 Xservers 复制到 **/etc/dt/config**，则必须将 **/etc/dt/config/Xconfig** 中的 **Dtlogin.servers:** 行更改或添加为:

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```

3. 注释掉 **/etc/dt/config/Xservers** 中启动 X 服务器的行。这将禁用“登录选项菜单”。

   ```
   # * Local local@console /path/X :0
   ```

4. 重读“登录管理器”配置文件。

位图显示存在时添加字符显示控制台

1. 如果 **/etc/dt/config/Xservers** 文件不存在，请将 **/usr/dt/config/Xservers** 文件复制到 **/etc/dt/config** 目录。
2. 如果必须将 Xservers 复制到 **/etc/dt/config**，则必须将 **/etc/dt/config/Xconfig** 中的 **Dtlogin.servers:** 行更改或添加为:

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```

3. 编辑 **/etc/dt/config/Xservers** 中启动 X 服务器进行读的行:

   ```
   * Local local@none /path/X :0
   ```

4. 重读“登录管理器”配置文件。

为公共台式机环境定制显示设备

可以配置“公共台式机环境登录管理器”以在具有两个或更多显示设备的系统上运行。

当系统包含多个显示时，必须满足以下配置要求:

- 必须在每个显示上都启动服务器。
- 必须为每个显示配置“非 Windows”方式。

可能需要或期望对每个显示使用不同的 dtlogin 资源。

还可能需要或期望对每个显示设备使用不同的系统范围环境变量。

在每个显示设备上启动服务器

1. 如果 **/etc/dt/config/Xservers** 文件不存在，请将 **/usr/dt/config/Xservers** 文件复制到 **/etc/dt/config** 目录。
2. 如果您必须将 Xservers 复制到 **/etc/dt/config**，则必须将 **/etc/dt/config/Xconfig** 中的 **Dtlogin.servers:** 行更改为:

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```

3. 编辑 **/etc/dt/config/Xservers** 以在每个显示设备上启动 X 服务器。

语法

启动服务器的一般语法是:

```
DisplayName DisplayClass DisplayType [ @ite ] Command
```

只有具有关联的“内部终端仿真器”（ITE）的显示可在“非 Windows”方式下操作。“非 Windows”方式临时禁用显示的桌面并运行 getty 进程（如果还没有启动一个 getty）。这允许您登录并执行在“公共台式机环境”下不可能执行的任务。当您注销时，桌面为显示设备重新启动。如果 getty 尚未在显示设备上运行，则当“非 Windows”方式启动时，“登录管理器”启动一个 getty。

缺省配置

当 ite 省略时，display:0 与 ITE（/dev/console）关联。

将不同的显示指定为 ITE

- 在 ITE 显示上，将 ITE 设置为字符设备。
- 在所有其它显示上，将 ITE 设置为无。

示例

Xserver 文件中的以下条目在 sysaaa:0 上的三个本地显示上启动服务器。显示 :0 将是控制台（ITE）。

```
sysaaa:0 Local local /usr/bin/X11/X :0
sysaaa:1 Local local /usr/bin/X11/X :1
sysaaa:2 Local local /usr/bin/X11/X :2
```

在主机 sysbbb 上，位图显示 :0 不是 ITE；ITE 与设备 **/dev/ttyi1** 关联。 **Xserver** 文件中的以下条目在两个位图显示上启动服务器，且在 :1 上启用“非 Windows”方式。

```
sysaaa:0 Local local@none /usr/bin/X11/X :0
sysaaa:1 Local local@ttyi1 /usr/bin/X11/X :1
```

指定 Xconfig 中的显示名

不能对 **/etc/dt/config/Xconfig** 中的显示名使用常规 hostname:0 语法。

- 使用下划线代替冒号。
- 在全限定主机名中，使用下划线代替句号。

示例

```
Dtlogin.claaa_0.resource: value
Dtlogin.sysaaa_prsm_ld_edu_0.resource: value
```

为每个显示使用不同的登录管理器资源

1. 如果 **/etc/dt/config/Xconfig** 文件不存在，请将 **/usr/dt/config/Xconfig** 文件复制到 **/etc/dt/config 目录**。
2. 使用 **/etc/dt/config/Xconfig** 中的资源，为每个显示指定不同的资源文件:

   ```
   Dtlogin.DisplayName.resources: path/file
   ```

   其中 *path* 是要使用的 Xresource 文件的路径名，*file* 是要使用的 Xresource 文件的文件名。
3. 创建 **Xconfig** 文件中指定的每个资源文件。特定于语言 Xresources 文件安装在 **/usr/dt/config/<LANG>** 中。
4. 在每个文件中，为该显示放置 dtlogin 资源。

示例

Xconfig 文件中的以下行为三个显示指定不同的资源文件:

```
Dtlogin.sysaaa_0.resources: /etc/dt/config/Xresources0
Dtlogin.sysaaa_1.resources: /etc/dt/config/Xresources1
Dtlogin.sysaaa_2.resources: /etc/dt/config/Xresources2
```

为每个显示运行不同的脚本

1. 如果 **/etc/dt/config/Xconfig** 文件不存在，请将 **/usr/dt/config/Xconfig** 文件复制到 **/etc/dt/config** 目录。
2. 使用 **/etc/dt/config/Xconfig** 中的启动、复位和设置资源，为每个显示指定不同的脚本（运行这些文件，而非 **Xstartup**、**Xreset** 和 **Xsetup.** 文件）:

   ```
   Dtlogin*DisplayName*startup: /path/file
   Dtlogin*DisplayName*reset: /path/file
   Dtlogin*DisplayName*setup: /path/file
   ```

其中 *path* 是要使用的文件的路径名，*file* 是要使用的文件的文件名。在“公共台式机环境”会话启动之前，启动脚本在用户登录后作为 root 用户运行。

脚本 **/usr/dt/config/Xreset** 可用于对 **Xstartup** 文件中所做的设置进行逆向操作。**Xreset** 文件在用户注销时运行。

示例

Xconfig 文件中的以下行为两个显示指定不同的脚本:

```
Dtlogin.sysaaa_0*startup:   /etc/dt/config/Xstartup0
Dtlogin.sysaaa_1*startup:    /etc/dt/config/Xstartup1
Dtlogin.sysaaa_0*setup:      /etc/dt/config/Xsetup0
Dtlogin.sysaaa_1*setup:      /etc/dt/config/Xsetup1
Dtlogin.sysaaa_0*reset:      /etc/dt/config/Xreset0
Dtlogin.sysaaa_1*reset:      /etc/dt/config/Xreset1
```

为每个显示设置不同的系统范围环境变量

1. 如果 **/etc/dt/config/Xconfig** 文件不存在，请将 **/usr/dt/config/Xconfig** 文件复制到 **/etc/dt/config** 目录。
2. 为每个显示分别设置 **/etc/dt/config/Xconfig** 中的环境资源:

   ```
   Dtlogin*DisplayName*environment: value
   ```

以下各点适用于每个显示的环境变量:

- 用空格或制表符分隔变量赋值。
- 不要使用环境资源设置 TZ 和 LANG。
- **Xconfig** 文件中没有处理的 shell。

示例

Xconfig 文件中的以下行为两个显示设置变量:

```
Dtlogin*syshere_0*environment:EDITOR=vi SB_DISPLAY_ADDR=0xB00000
Dtlogin*syshere_1*environment: EDITOR=emacs \
         SB_DISPLAY_ADDR=0xB00000
```

第 4 章 命令和进程

*命令*是执行操作或运行程序的请求。可以使用命令告诉操作系统想要其执行何任务。当输入命令时，命令解释器（也称为*shell*）译码命令并处理该任务。

实际运行在计算机上的程序或命令指*进程*。操作系统可以同时运行许多不同的进程。

操作系统允许您通过使用特定的 I/O 命令和符号来操作数据从您的系统和到您的系统的输入和输出（I/O）。可以通过指定收集数据的位置来控制输入。例如，可以指定在数据输入到键盘（标准输入）时读输入或从文件读输入。您可通过指定显示或存储数据的位置来控制输出。例如，您可指定将输出数据写入屏幕（标准输出）或写入文件。

本章讨论以下内容:

命令概述

一些命令可以通过简单地输入一个字来输入。也有可能将多个命令组合起来，以使一个命令的输出成为另一个命令的输入。这称为*传递*。有关传递的更多信息，请参阅第 136 页的『shell 功能』。

标志进一步定义命令的操作。标志是在命令行上与命令名一起使用的修饰符，通常之前有一个破折号。

多个命令还可以分组在一起并存储在一个文件中。这些称为 *shell 步骤*或 *shell 脚本*。执行包含这些命令的文件来代替单独地执行每个命令。有关脚本和步骤的更多信息，请参阅第 139 页的『创建和运行 shell 脚本』。

要输入命令，请在提示符下输入命令名，并按下 Enter 键。

```
$ CommandName
```

本节描述以下过程:

命令语法

尽管一些命令可以通过简单地输入一个字来输入，但其它命令使用标志和参数。每个命令都具有语法，指定要求的和可选的标志和参数。命令的一般格式如下:

```
CommandName flag(s) parameter(s)
```

以下是有关命令的一些一般规则:

- 命令、标志和参数间的空格是有意义的。
- 通过用分号（;）分隔命令，两个命令可在同一行上输入。例如:

  ```
  $ CommandOne;CommandTwo
  ```

  shell 顺序地运行命令。
- 命令是区分大小写的。shell 区别大写字母和小写字母。对于 shell，`print` 不同于 `PRINT` 或 `Print`。

- 很长的命令可以通过使用反斜杠 (\) 字符在不止一行上输入。反斜杠向 shell 标识行继续。以下示例是一个跨两行的命令:

  ```
  $ ls Mail info temp \ (按 Enter 键) > diary
  （出现 > 提示符）
  ```

  > 字符是次提示符（$ 是非 root 用户的缺省主提示符），表示当前行是前一行的继续。请注意 **csh**（C shell）不给出次提示符，并且断行必须在字边界，且其主提示符是 **%**。

命令名

每个命令的第一个字是命令名。一些命令只有命令名。

命令标志

命令名后可能有很多标志。标志修改命令的操作，并且有时称为*选项*。标志由空格或制表符分开，并且通常以破折号（-）开始。例外为 **ps**、**tar** 和 **ar**，这些命令在一些标志前不要求破折号。例如，在以下命令中:

```
ls -a -F
```

ls 是命令名，-a -F 是标志。

当命令使用标志时，标志直接跟在命令名后。命令中的单字符标志可与一个破折号组合。例如，前一命令还可写成如下:

```
ls -aF
```

存在参数实际上以破折号（-）开始的一些情况。在此情况下，请在参数前使用定界符破折号（–）。– 告诉命令后面跟的不是标志，而是参数。

例如，如果想要创建名为 -tmp 的目录，并输入以下命令:

```
mkdir -tmp
```

系统显示类似于以下的错误消息:

```
mkdir: 非识别的标志: t
用法: mkdir [-p] [-m mode] Directory ...
```

输入命令的正确方式如下:

```
mkdir -- -tmp
```

新目录 -tmp 现在已创建。

命令参数

在命令名后，可能有很多后跟参数的标志。参数有时称为*自变量*或*操作数*。参数指定命令要运行所需要的信息。如果未指定参数，则命令可能假定缺省值。例如，在以下命令中:

```
ls -a temp
```

ls 是命令名，-a 是标志，temp 是参数。此命令显示目录 temp 中的所有（-a）文件。在以下示例中:

```
ls -a
```

缺省值是当前目录，因为未给出参数。在下例中:

```
ls temp mail
```

未给出标志，temp 和 mail 是参数。在这种情况下，temp 和 mail 是两个不同的目录名。**ls** 命令显示这些目录中除隐藏文件外的所有文件。

当参数或选项自变量是（或包含）数值时，数字解释为十进制整数，除非另有指定。0 到 **INT_MAX**（在 **/usr/include/sys/limits.h** 文件中定义）范围内的数字在句法上识别为数值。

如果想要使用的命令接受负数作为参数或选项自变量，则可以使用 **INT_MIN** 到 **INT_MAX**（都在 **/usr/include/sys/limits.h** 文件中定义）范围内的数字。这并不一定意味着在该范围内的所有数字都在语义上是正确的。一些命令具有内置规范，允许较小范围的数字，例如一些打印命令。如果生成错误，错误消息让您知道值超出支持的范围，而不是命令在句法上是不正确的。

读用法语句

用法语句是表示命令语法的方式，它由诸如方括号 ([ ])、大括号 ({ }) 和竖线 (|) 之类的符号组成。以下是 **unget** 命令的用法语句的一个样本:

unget [**-r***SID*] [**-s**] [**-n**] *File ...*

在命令用法语句中使用以下约定:

- 必须在命令行上逐字输入的项为**粗体**。这些项包括命令名、标志和文字字符。
- 代表必须由名称替换的变量的项为*斜体*。这些项包括命令读取的跟在标志和参数后的参数，如*文件*和*目录*。
- 用方括号圈起的参数是可选的。
- 用大括号圈起的参数是必需的。
- 未由方括号或大括号圈起的参数是必需的。
- 竖线标识只选择一个参数。例如, [a | b] 表示*可以*选择 a、b 或不选择。类似地，{ a | b } 表明您*必须*选择 a 或 b。
- 省略号（ **...** ）表示参数可在命令行上重复。
- 破折号（ **-** ）表示标准输入。

使用基于 Web 的系统管理器

“基于 Web 的系统管理器”是一个图形用户界面，用于从本地连接的显示或从另一个装备了 Web 浏览器的系统或个人计算机以远程的方式来管理系统。可以用各种方式启动基于 Web 的系统管理器:

- 从“公共台式机环境（CDE）”中的命令行终端，通过输入 **wsm** 命令。
- 从 AIXwindows 环境中的命令行终端，通过输入 **wsm** 命令。
- 从“CDE”应用程序管理器，通过转至 System_Admin 文件夹并单击**管理控制台**图标。
- 从按*《AIX 5L V5.2 基于 Web 的系统管理器管理指南》*中描述配置的个人计算机上兼容 HTML 3.2 的 Web 浏览器。

使用 smit 命令

smit 命令是可用于运行其它命令的工具。作为参数输入到 **smit** 命令的命令名可能会带您到该命令的子菜单或面板。例如，**smit lsuser** 命令将您直接带到**列出所有用户**，它列出系统上的用户的属性。

请参阅*《AIX 5L V5.2 命令参考大全》*中的 **smit** 命令，以获取完整的语法。

定位命令或程序（whereis 命令）

whereis 命令定位指定的文件的源、二进制文件和手册节。命令尝试从标准位置的列表中查找期望的程序。

要在没有文档的当前目录中查找文件，请输入:

```
whereis -m -u *
```

按下 Enter 键。

要查找所有包含名称 Mail 的文件，请输入:

```
whereis Mail
```

按下 Enter 键。

系统显示类似于以下的信息:

```
Mail: /usr/bin/Mail /usr/lib/Mail.rc
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **whereis** 命令，以获取完整的语法。

显示有关命令的信息（man 命令）

man 命令显示有关命令、子例程和文件的信息。**man** 命令的一般格式如下:

```
man CommandName
```

要获取有关 **pg** 命令的信息，请输入:

```
man pg
```

按 Enter 键。

系统显示类似于以下的信息:

```
pg Command

目的

格式化文件到显示器。

语法

pg [ - Number ] [ -c ] [ -e ] [ -f ] [ -n ] [ -p String ]
[ -s ] [ +LineNumber | +/Pattern/ ] [ File ... ]

描述

pg 命令从 File 参数读取文件名，并
一次一屏地将文件写到标准输出。如果您
指定 -（破折号）作为 File 参数，或运行 pg 命令时
不带选项，则 pg 命令读标准输入。每个
屏幕后跟一个提示符。如果按下 Enter 键，
则显示另一页。与 pg 命令一起使用的子命令
让您在文件中检查或搜索。
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **man** 命令，以获取完整的语法。

显示命令的功能（whatis 命令）

whatis 命令从您使用 **catman -w** 命令创建的数据库中，查找给定的命令、系统调用、库函数或特殊文件名（由 *Command* 参数指定）。**whatis** 命令显示手册节的标题行。然后您可发出 **man** 命令来获取其它信息。

whatis 命令等效于使用 **man -f** 命令。

要找出 **ls** 命令的作用，请输入:

```
whatis ls
```

按下 Enter 键。

系统显示类似于以下的信息:

```
ls(1)     - 显示目录的内容。
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **whatis** 命令，以获取完整的语法。

列出先前输入的命令（history shell 命令）

history 命令是一个 Korn shell 内置，列出最后 16 个输入的命令。Korn shell 将您输入的命令保存到命令历史文件中，通常命名为 **$HOME/.sh_history**。当您需要重复先前的命令时，该操作会节省时间。

缺省情况下，Korn shell 保存最后 128 个命令的文本。历史文件大小（由 **HISTSIZE** 环境变量指定）是没有限制的，尽管非常大的历史文件大小可能导致 Korn shell 启动缓慢。

> **注**: Bourne shell 不支持命令历史。

有关 shell 的详细信息，请参阅第 135 页的第 12 章，『shell』。

要列出您先前输入的命令，请在提示符下输入:

```
history
```

按下 Enter 键。

输入 **history** 命令本身将列出前 16 个输入的命令。系统显示类似于以下的信息:

```
928   ls
929   mail
930   printenv MAILMSG
931   whereis Mail
932   whatis ls
933   cd /usr/include/sys
934   ls
935   man pg
936   cd
937   ls | pg
938   lscons
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
```

列表首先显示命令在 **$HOME/.sh_history** 文件中的位置，后跟命令。

要列出先前的 5 个命令，请在提示符下输入:

```
history -5
```

按下 Enter 键。

显示类似于以下的列表:

```
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
944   history -5
```

后跟数字的 **history** 命令列出从该数字开始的所有先前输入的命令。

要列出从 938 开始的命令，请在提示符下输入:

```
history 938
```

按下 Enter 键。

显示类似于以下的列表:

```
938   lscons
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
944   history -5
945   history 938
```

使用 history shell 命令重复命令

使用 **r** Korn shell 别名来重复先前的命令。输入 **r** 并按下 Enter 键，然后可以指定命令的号码、第一个字符或头几个字符。

如果想要列出系统上当前可用的显示，请输入 **lsdisp** 并在提示符下按下 Enter 键。系统在屏幕上返回信息。如果想要相同的信息再次返回给您，请在提示符下输入:

```
r
```

按下 Enter 键。

系统再次运行最近输入的命令。在此示例中，运行 **lsdisp** 命令。

要重复 **ls *.txt** 命令，请在提示符下输入:

```
r ls
```

按下 Enter 键。

r Korn shell 别名定位以指定的一个字符或几个字符开始的最近的命令。

使用 history shell 命令替换字符串

还可以使用 **r** Korn shell 别名以在运行命令之前修改它。在这种情况下，可使用格式为 *Old*=*New* 的替换参数在运行命令之前修改它。

例如，如果命令行 940 是 **ls *.txt**，而您想要运行 **ls *.exe**，请在提示符下输入:

```
r txt=exe 940
```

按下 Enter 键。

这样运行命令 940，用 **exe** 替换 **txt**。

例如，如果行 940 上的命令是以小写字母 *l* 开始的最近命令，则还可以输入:

```
r txt=exe l
```

按下 Enter 键。

> **注**: *New* 字符串仅替换第一个出现的 *Old* 字符串。输入不带特的定命令号码或字符的 **r** Korn shell 别名对先前输入的命令进行替换。

编辑命令历史

使用 **fc** Korn 命令 shell 内置命令列出或编辑命令历史文件的某些部分。要选择文件的一部分编辑或列出，指定命令的号码、第一个字符或头几个字符。可以指定单一命令或某一范围的命令。

如果未指定编辑器程序作为 **fc** Korn shell 内置命令的自变量，则使用由 **FCEDIT** 变量指定的编辑器。如果未定义 **FCEDIT** 变量，则使用 **/usr/bin/ed** 编辑器。当退出编辑器时，打印和运行已编辑的一个或多个命令。使用 **printenv** 命令显示 **FCEDIT**变量的值。

例如，如果想要运行命令:

```
cd /usr/tmp
```

（它与命令行 933 很相似），请在提示符下输入:

```
fc 933
```

按下 Enter 键。

此时，出现缺省编辑器出现，并带有命令行 933。您会将 `include/sys` 更改为 `tmp`，当退出编辑器时，已编辑的命令将运行。

还可以指定要在 **fc** 命令中使用的编辑器。

例如，如果您要使用 `/usr/bin/vi` 编辑器编辑命令，请在提示符下输入:

```
fc -e vi 933
```

按下 Enter 键。

此时，出现 **vi** 编辑器，并带有命令行 933。

还可以指定要编辑的命令的范围。

例如，如果想要编辑命令 930 到 940，请在提示符下输入:

```
fc 930 940
```

按下 Enter 键。

此时，出现缺省编辑器，并带有命令行 930 到 940。当退出编辑器时，出现在编辑器中的所有命令将按顺序运行。

创建命令别名（alias shell 命令）

alias 让您为命令、文件名或任何 shell 文本创建快捷键名称。通过使用别名，当您执行经常执行的任务时可节省不少时间。**alias** Korn shell 内置命令定义一个字作为一些命令的别名。您可以使用别名来重新定义内置命令，但不要重新定义保留字。

别名的第一个字符可以是除元字符外的任何可打印字符。任何其余字符必须与对于有效文件名的要求相同。

创建别名的格式如下:

```
alias Name=String
```

其中 *Name* 参数指定别名的名称，*String* 参数指定一个字符串。如果 *String* 包含空格，则用引号圈起。

要为命令 **rm -i**（在删除文件之前提示您）创建别名，请在提示符下输入:

```
alias rm="/usr/bin/rm -i"
```

按下 Enter 键。

在此示例中，无论何时输入命令 **rm** 并按下 Enter 键，实际执行的命令都是 **/usr/bin/rm -i**。

要为命令 **ls -alF | pg**（显示当前目录中所有文件的详细信息，包括不可见文件；用 * 标记可执行文件，用 / 标记目录；并且每屏滚动）创建别名，在提示符下，请输入:

```
alias dir="/usr/bin/ls -alF | pg"
```

按下 Enter 键。

在此示例中，无论何时输入命令 **dir** 并按下 Enter 键，实际执行的命令都是 **/usr/bin/ls -alF | pg**。

要显示您具有的所有别名，请在提示符下输入:

```
alias
```

按下 Enter 键。

系统显示类似于以下的信息:

```
rm="/usr/bin/rm -i"
dir="/usr/bin/ls -alF | pg"
```

使用文本格式命令

可以使用文本格式命令来处理由用于欧洲语言的国际扩展字符集组成的文本。

文本格式的国际字符支持

国际扩展字符集提供在很多欧洲语言中使用的字符和符号，以及由英语语言字符、数字和标点符号组成的 ASCII 子集。

欧洲扩展字符集中的所有字符都具有 ASCII 格式。这些格式可用于表示输入中的扩展字符，或表示可直接使用设备（如支持欧洲扩展字符的键盘）输入的字符。

以下文本格式命令支持所有使用单字节字符的国际语言。这些命令位于 **/usr/bin** 中。（使用星号（*）标识的命令支持多字节语言的文本处理。有关多字节语言的更多信息，请参阅第 34 页的『文本格式的多字节字符支持』。）

```
addbib*         hyphen          pic*            pstext
checkmm         ibm3812         ps4014          refer*
checknr*        ibm3816         ps630           roffbib*
col*            ibm5587G*       psbanne         soelim*
colcrt          ibm5585H-T*     psdit           sortbib*
deroff*         indxbib*        psplot          tbl*
enscript        lookbib*        psrev           troff*
eqn*            makedev*        psroff          vgrind
grap*           neqn*           psrv            xpreview*
hplj            nroff*
```

不在上表中的文本格式命令和宏软件包未启用支持处理国际字符。

输入扩展单字节字符

如果您的输入设备支持来自欧洲语言扩展字符集的字符，则可以直接输入它们。或者，使用以下 ASCII 转义序列格式来表示这些字符:

格式 \[*N*], 其中 *N* 是字符的 2 位或 4 位十六进制代码。

注: NCesc 格式 \<xx> 不再受支持。。

包含扩展字符的文本是根据正在使用的语言的格式约定的输出。没有为特定输出设备的接口定义的字符不产生输出或错误指示。

尽管请求、宏软件包和命令的名称是基于英语的，但它们大多数能接受包含欧洲扩展字符集中字符的输入（如文件名和参数）。

对于 **nroff** 和 **troff** 命令及其预处理器，命令输入必须是 ASCII 格式，否则将导致不可恢复的语法错误。当在引号内圈起或位于其它要格式化的文本中时，可输入国际字符（单字节或多字节）。例如，从 **pic** 命令使用宏:

```
define foobar % SomeText %
```

在 define 伪指令后，第一个名称 foobar 必须是 ASCII。但是，替换文本 *SomeText* 可以包含非 ASCII 字符。

文本格式的多字节字符支持

某些文本格式命令可用于处理多字节语言的文本。这些命令在第 33 页的『文本格式的国际字符支持』下的列表中用星号（*）标识。不在列表中的文本格式命令未启用支持处理国际字符。

输入多字节字符

如果您的输入设备支持，则可直接输入多字节字符。或者，可以 ASCII 格式 \[*N*] 输入任何多字节字符，其中 ***N*** 是字符的 2 位、4 位、6 位、7 位或 8 位十六进制编码。

尽管请求、宏和命令的名称是基于英语的，但它们大多数能接受包含任何类型多字节字符的输入（如文件名和参数）。

如果您已熟悉使用单字节文本的文本格式命令，以下列表概述了值得注意或多字节语言环境唯一的特征。

- 文本不断字。
- 多字节数字输出要求特殊格式类型。日语格式类型可用。
- 文本在水平行中输出，从左至右填充。
- 字符空间是不变的，因此字符自动按列对齐。

- 没有为特定输出设备的接口定义的字符不产生输出或错误指示。

进程概述

实际运行在计算机上的程序或命令称为*进程*。进程存在于“父子”层次结构中。程序或命令启动的进程是*父进程*；*子进程*是父进程的产品。父进程可以具有几个子进程，但是子进程只能有一个父进程。

系统在它启动时将进程标识号（PID 号）指定给每个进程。如果启动同一程序几次，则它将每次具有不同的 PID 号。

当在系统上启动进程时，进程使用部分可用的系统资源。当运行不止一个进程时，构建到操作系统中的调度程序给每个进程其共享计算机时间，这基于建立的优先级。可以使用 **nice** 或 **renice** 命令更改这些优先级。

> **注**：要将进程优先级更改得高一些，必须具有 root 用户权限。所有用户都可以通过使用 **nice** 命令来降低他们启动在进程上的优先级，或通过使用 **renice** 命令来降低他们已启动的进程上的优先级。

本节描述以下过程:

前台和后台进程

要求用户启动它们或与它们交互的进程称为*前台进程*。独立于用户运行的进程称为*后台进程*。缺省情况下，程序和命令作为前台进程运行。要在后台运行进程，请将和符号（&）放在您用于启动进程的命令名结尾。

守护程序

*守护程序*是无人照管运行的进程。它们总是在后台，且在任何时候都可用。守护程序通常在系统启动时启动，并且它们一直运行到系统停止。守护程序进程执行系统服务，且在任何时候都对不止一个任务或用户可用。守护程序进程由 root 用户或 root shell 启动，并只能由 root 用户停止。例如，**qdaemon** 进程提供对系统资源（如打印机）的访问。另一个公共守护程序是 **sendmail** 守护程序。

zombie 进程

*zombie 进程*是一个死进程，它不再执行但仍在进程表中识别（换句话说，它具有 PID 号）。它不具有分配给它的其它系统空间。zombie 进程已杀死或退出，并继续在进程表中存在，直到父进程死或系统关闭并重新启动为止。当被 **ps** 命令列出时，zombie 进程显示为 <defunct>。

启动进程

通过在系统提示符下输入程序名或命令名来从显示站启动前台进程。启动前台进程后，进程在显示站与您交互直到它完成为止。这意味着无其它交互作用（例如，请输入另一个命令）可在显示站发生，直到进程完成或您停机它。

单一用户能同一时刻运行不止一个进程，最多每个用户缺省最大 40 个进程。

在前台启动进程

要在前台运行进程，请输入命令的名称，并带有所有相应的参数和标志:

```
$ CommandName
```

按下 Enter 键。

在后台启动进程

要在后台运行进程，请输入命令的名称，并带有所有相应的参数和标志，后跟和符号 &:

```
$ CommandName&
```

按下 Enter 键。

当进程在后台运行时，可以通过在显示站输入其它命令执行其它任务。

通常，后台进程对于花费很长时间运行的命令最有用。然而，由于它们增加处理器处理的工作总量，所以后台进程也使系统的其余部分慢下来。

大多数进程将其输出定向到标准输出，即使当它们在后台运行时亦是如此。除非重定向，否则标准输出转至显示设备。由于后台进程的输出可能妨碍系统上的其它工作，所以通常一个好的做法是将后台进程的输出重定向到文件或打印机。然后您可在就绪时查看输出。

> **注**: 在某些情况下，当进程运行在后台而不是前台时，可能以不同的顺序生成其输出。程序员可能想要使用 **fflush** 子例程来确保输出以正确顺序发生，无论进程在前台或后台运行。

只要后台进程在运行，就可以使用 **ps** 命令检查其状态。

检查进程（ps 命令）

任何时候系统运行时，几个进程也在运行。可以使用 **ps** 命令来查找哪些进程在运行和显示有关那些进程的信息。

ps 命令

ps 命令具有几个标志，它使您能够指定要列出哪些进程和要显示有关每个进程的什么信息。

要显示运行在系统上的所有进程，请在提示符下输入:

```
ps -ef
```

按下 Enter 键。

系统显示类似于以下的信息:

```
    USER   PID  PPID   C    STIME    TTY  TIME CMD
    root     1     0   0   Jun 28      -  3:23 /etc/init
    root  1588  6963   0   Jun 28      -  0:02 /usr/etc/biod 6
    root  2280     1   0   Jun 28      -  1:39 /etc/syncd 60
    mary  2413 16998   2 07:57:30      -  0:05 aixterm
    mary 11632 16998   0 07:57:31  lft/1  0:01 xbiff
    mary 16260  2413   1 07:57:35  pts/1  0:00 /bin/ksh
    mary 16469     1   0 07:57:12  lft/1  0:00 ksh /usr/lpp/X11/bin/xinit
    mary 19402 16260  20 09:37:21  pts/1  0:00 ps -ef
```

先前一输出的列定义如下:

USER 用户登录名
PID 进程标识
PPID 父进程标识
C 进程的 CPU 使用率
STIME 进程的开始时间
TTY 进程的控制工作站
TIME 进程的总执行时间
CMD 命令

在前一示例中，**ps -ef** 命令的进程标识是 19402。其父进程标识是 16260，**/bin/ksh** 命令。

如果此列表很长，则顶部滚动到屏幕以外。要一次一页（屏幕）地显示该列表，请使用传递给 **pg** 命令的 **ps** 命令。在提示符下输入:

```
ps -ef | pg
```

按下 Enter 键。

要显示在系统上运行的所有进程的状态信息，请在提示符下输入:

```
ps gv
```

按下 Enter 键。

此格式的命令列出每个活动进程的许多统计信息。该命令的输出看起来类似于以下内容:

```
   PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
     0      - A     0:44    7     8     8    xx     0     0  0.0  0.0 swapper
     1      - A     1:29  518   244   140    xx    21    24  0.1  1.0 /etc/init
   771      - A     1:22    0    16    16    xx     0     0  0.0  0.0 kproc
  1028      - A     0:00   10    16     8    xx     0     0  0.0  0.0 kproc
  1503      - A     0:33  127    16     8    xx     0     0  0.0  0.0 kproc
  1679      - A     1:03  282   192    12 32768   130     0  0.7  0.0 pcidossvr
  2089      - A     0:22  918    72    28    xx     1     4  0.0  0.0 /etc/sync
  2784      - A     0:00    9    16     8    xx     0     0  0.0  0.0 kproc
  2816      - A     5:59 6436  2664   616     8   852   156  0.4  4.0 /usr/lpp/
  3115      - A     0:27  955   264   128    xx    39    36  0.0  1.0 /usr/lib/
  3451      - A     0:00    0    16     8    xx     0     0  0.0  0.0 kproc
  3812      - A     0:00   21   128    12 32768    34     0  0.0  0.0 usr/lib/lpd/
  3970      - A     0:00    0    16     8    xx     0     0  0.0  0.0 kproc
  4267      - A     0:01  169   132    72 32768    16    16  0.0  0.0 /etc/sysl
  4514  lft/0 A     0:00   60   200    72    xx    39    60  0.0  0.0 /etc/gett
  4776  pts/3 A     0:02  250   108   280     8   303   268  0.0  2.0 -ksh
  5050      - A     0:09 1200   424   132 32768   243    56  0.0  1.0 /usr/sbin
  5322      - A     0:27 1299   156   192    xx    24    24  0.0  1.0 /etc/cron
```

```
 5590      - A     0:00    2   100    12 32768    11     0  0.0  0.0 /etc/writ
 5749      - A     0:00    0   208    12    xx    13     0  0.0  0.0 /usr/lpp/
 6111      - T     0:00   66   108    12 32768    47     0  0.0  0.0 /usr/lpp/
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **ps** 命令，以获取完整的语法。

设置进程的初始优先级（nice 命令）

通过使用 **nice** 命令启动进程，可以将进程的初始优先级设置为低于基本调度优先级的值。

注: 要以较高的优先级运行进程，必须具有 root 用户权限。

nice 命令

要设置进程的初始优先级，请输入:

```
nice -n Number CommandString
```

其中 *Number* 在 0 到 39 的范围内，其中 39 是最低优先级。*nice value* 是进程的系统调度优先级的十进制值。数越高，优先级越低。如果使用零，则进程将以其基本调度优先级运行。*CommandString* 是想要运行的命令及标志和参数。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **nice** 命令，以获取完整的语法。

您也可以使用 **smit nice** 命令执行此任务。

更改运行进程的优先级（renice 命令）

通过从命令行使用 **renice** 命令，可以将运行进程的调度优先级更改为低于或高于基本调度优先级的值。此命令更改进程的 nice 值。

注: 要以较高优先级运行进程，或要更改您未启动的进程的优先级，必须具有 root 用户权限。

从命令行

要更改运行进程的初始优先级，请输入:

```
renice Priority -p ProcessID
```

其中 *Priority* 在 -20 到 20 的范围内。数越高，优先级越低。如果使用零，则进程将以其基本调度优先级运行。*ProcessID* 是想要为其更改优先级的 PID。

还可以使用 **smit renice** 命令执行此任务。

取消前台进程

如果启动前台进程，然后决定不想要它完成，则可以通过按下 INTERRUPT 来取消它。这通常是 Ctrl-C 或 Ctrl-Backspace。要找出 INTERRUPT 键设置为的值，请参阅第 15 页的『列出终端的控制键指定（stty 命令）』。

注: INTERRUPT（Ctrl-C）不取消后台进程。要取消后台进程，必须使用 **kill** 命令。

大多数简单命令不是演示如何取消进程的好示例。它们运行得很快，以至于它们在您有时间取消它们之前就完成了。因此，本节中的示例使用的命令要花多于几秒钟运行：**find / -type f**。该命令显示系统上所有文件的路径名。您不需要学习 **find** 命令以完成本节；它在这里仅是演示如何使用进程。

在下例中，**find** 命令启动进程。在进程运行几秒钟后，可以通过按下 INTERRUPT 键来取消它：

```
$ find / -type f
/usr/sbin/acct/lastlogin
/usr/sbin/acct/prctmp
/usr/sbin/acct/prdaily
/usr/sbin/acct/runacct
/usr/sbin/acct/sdisk
/usr/sbin/acct/shutacct INTERRUPT (Ctrl-C)
$ _
```

系统将提示符返回到网屏。现在可以输入另一个命令。

停止前台进程

可能有这样的情况：进程停止，但是没有从进程表中除去其进程标识（PID）。可以通过从键盘按下 Ctrl-Z 来停止前台进程。

> **注**：Ctrl-Z 在 Korn shell（**ksh**）和 C shell（**csh**）中成功工作，但是在 Bourne shell（**bsh**）不成功工作。

重新启动已停止的进程

本过程描述如何重新启动用 Ctrl-Z 停止的进程。

> **注**：Ctrl-Z 在 Korn shell（**ksh**）和 C shell（**csh**）中成功工作，但是在 Bourne shell（**bsh**）不成功工作。要重新启动已停止的进程，您必须是启动进程的用户或具有 root 用户权限的用户。

1. 要显示系统上所有正在运行或已停止的进程（而非杀死的那些进程），请输入：

   ```
   ps -ef
   ```

   可能想要通过 **grep** 命令来传递此命令以限制列表仅列出那些您最可能想要重新启动的进程。例如，如果想要重新启动 **vi** 会话，可以输入：

   ```
   ps -ef | grep vi
   ```

   按下 Enter 键。此命令将仅显示 **ps** 命令输出中包含字 vi 的那些行。输出将类似于以下内容：

   ```
   UID    PID   PPID   C      STIME       TTY  TIME  COMMAND
   root   1234  13682  0      00:59:53    -    0:01  vi test
   root  14277  13682  1      01:00:34    -    0:00  grep vi
   ```

2. 在 **ps** 命令输出中，查找想要重新启动的进程并记下其 PID 号。在此例中，PID 是 1234。
3. 要将 CONTINUE 信号发送到已停止的进程，请输入：

   ```
   kill -19 1234
   ```

   用您进程的 PID 替换 1234。-19 表示 CONTINUE 信号。此命令在后台重新启动进程。如果进程可以在后台运行，则已完成此过程。如果进程必须在前台运行（如同 **vi** 会话），则必须继续进行下一步。
4. 要将进程置于前台，请输入：

   ```
   fg 1234
   ```

   再一次，用您进程的 PID 替换 1234。您的进程现在应该在前台运行。（现在处于 **vi** 编辑会话中）。

为以后操作调度进程（at 命令）

可以将进程设置为*批处理进程*，以在某一调度的时间在后台运行。**at** 和 **smit** 命令让您输入将在以后运行的命令的名称，并允许您指定何时命令应该运行。

> **注**：**/var/adm/cron/at.allow** 和 **/var/adm/cron/at.deny** 文件控制您是否可使用 **at** 命令。具有 root 用户权限的用户可以创建、编辑或删除这些文件。这些文件中的条目是一行一个名称的用户登录名。以下是 **at.allow** 文件的示例:
>
> ```
> root
> nick
> dee
> sarah
> ```

如果 **at.allow** 文件存在，则只有那些其登录名在其中列出的用户可以使用 **at** 命令。通过在 **at.deny** 文件中列出用户的登录名，系统管理员可以显式地停止用户使用 **at** 命令。如果仅 **at.deny** 文件存在，则任何其名称不在文件中出现的用户都可以使用 **at** 命令。

如果以下任何一项为真，您都无法使用 **at** 命令:

- **at.allow** 文件和 **at.deny** 文件不存在（仅允许 root 用户）。
- **at.allow** 文件存在，但是用户登录名不在其中列出。
- **at.deny** 文件存在，并且用户登录名在其中列出。

如果 **at.allow** 文件不存在，并且 **at.deny** 文件不存在或为空，则只有具有 root 用户权限的用户可以使用 **at** 命令提交作业。

at 命令语法允许您指定当想要进程运行时的日期字符串、时间和天字符串或增量字符串。它还允许您指定要使用哪个 shell 或队列。下例显示该命令的一些典型使用。

at 命令

例如，如果您的登录名是 joyce，并且具有要在午夜运行的名为 WorkReport 的脚本，请执行以下操作:

1. 输入您要程序开始运行的时间。

   ```
   at midnight
   ```

2. 输入要运行的程序名称，在每个名称后按下 Enter 键。输入最后一个名称后，按下文件结束符字符（Ctrl-D）以表明列表结束。

   ```
   WorkReport^D
   ```

按下 Ctrl-D 后，系统显示类似于以下的信息:

```
job joyce.741502800.a at Fri Jul  6 00:00:00 CDT 2002.
```

对程序 WorkReport 给定作业号 `joyce.741502800.a`，并将在 7 月 6 日午夜运行。

要列出您已发送以在以后运行的程序，请输入:

```
at -l
```

系统显示类似于以下的信息:

```
joyce.741502800.a       Fri Jul  6 00:00:00 CDT 2002
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **at** 命令，以获取正确的语法。

列出所有已调度进程（at 或 atq 命令）

通过将 **-l** 标志与 **at** 命令或 **atq** 命令一起使用，可以列出所有已调度的进程。 两个命令都给出了相同的输出，但是 **atq** 命令可根据 **at** 命令发出的时间排序进程，并可以只显示队列中的进程号。

可以用以下方法列出所有已调度的进程:

- 从命令行使用 **at** 命令
- 使用 **atq** 命令

对于使用 **at** 命令的用户限制，请参阅为以后操作调度进程（at 命令）中的**注**。

at 命令

要列出已调度的进程，请输入:

```
at -l
```

此命令列出您的队列中所有已调度的进程。如果您是 root 用户，则此命令列出所有用户的所有已调度的进程。有关语法的完整详细信息，请参阅 **at** 命令。

atq 命令

要列出队列中所有已调度的进程，请输入:

```
atq
```

如果您是 root 用户，则可以通过输入以下内容列出特殊用户队列中已调度的进程:

```
atq UserName
```

要列出队列中已调度的进程的号，请输入:

```
atq -n
```

从调度表中除去进程（at 命令）

可以通过使用 **-r** 标志的 **at** 命令除去已调度的进程。有关使用 **at** 命令的用户限制，请参阅为以后操作调度进程（at 命令）中的**注**。

从命令行

1. 要除去已调度的进程，必须知道进程号。可以使用 **at -l** 命令或 **atq** 命令获取进程号。请参阅『列出所有已调度进程（at 或 atq 命令）』以获取详细信息。
2. 当知道想要除去的进程号时，请输入:

   ```
   at -r ProcessNumber
   ```

还可以使用 **smit rmat** 命令执行该任务。

除去后台进程（kill 命令）

如果 INTERRUPT 不停机前台进程，或者如果在启动后台进程后决定不想要进程完成，则可以使用 **kill** 命令取消进程。在使用 **kill** 命令取消进程之前，必须知道其 PID 号。**kill** 命令的一般格式如下:

```
kill ProcessID
```

> **注**: 要除去进程，必须具有 root 用户权限或是启动了进程的用户。**kill** 命令进程的缺省信号是 -15（SIGTERM）。

kill 命令

注: 要除去 zombie 进程，必须除去其父进程。

1. 使用 **ps** 命令确定想要除去的进程的进程标识。可能想要通过 **grep** 命令传递此命令以仅列出想要的进程。例如，如果想要 vi 会话的进程标识，可以输入:

   ```
   ps -l | grep vi
   ```

2. 在以下示例中，发出 **find** 命令以在后台运行。然后决定取消进程。发出 **ps** 命令以列出 PID 号。

   ```
   $ find / -type f > dir.paths &
   [1]    21593
   $ ps
      PID   TTY  TIME  COMMAND
     1627  pts3  0:00  ps
     5461  pts3  0:00  ksh
    17565  pts3  0:00  -ksh
    21593  pts3  0:00  find / -type f
   $ kill 21593
   $ ps
      PID   TTY  TIME  COMMAND
     1627  pts3  0:00  ps
     5461  pts3  0:00  ksh
    17565  pts3  0:00  -ksh
   [1]  +  Terminated 21593     find  /  -type f > dir.paths  &
   ```

   命令 **kill 21593** 结束后台 **find** 进程，第二个 **ps** 命令不返回有关 PID 21593 的状态信息。系统不显示终止消息，直到输入下一个命令，除非该命令是 **cd**。

   kill 命令让您取消后台进程。如果意识到已错误地将进程置于后台或进程要很长时间运行，则您可能想这样做。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **kill** 命令，以获取完整的语法。

kill 命令还以通过输入以下内容在 **smit** 中使用:

```
smit kill
```

命令和进程的命令摘要

命令

alias	打印别名列表到标准输出的 shell 命令
history	显示历史事件列表的 shell 命令
man	显示有关命令、子例程和文件联机的信息
wsm	从 web 浏览器执行系统管理
whatis	描述命令执行的函数
whereis	为已安装程序定位源、二进制文件或手册

进程

at	稍后运行命令，列出所有已调度进程，或从调度中除去进程
atq	显示等待要运行的作业的队列
kill	将信号发送到运行的进程
nice	以较低或较高优先级运行命令。
ps	显示进程的当前状态。
renice	改变运行进程的优先级

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 135 页的第 12 章, 『shell』

第 140 页的『Korn shell 或 POSIX shell 命令』

第 174 页的『Bourne shell』

第 188 页的『C shell』

第 5 章 输入和输出重定向

操作系统允许您通过使用特定的 I/O 命令和符号来操作数据从您的系统和到您的系统的输入和输出（I/O）。可以通过指定收集数据的位置来控制输入。例如，可以指定在数据输入到键盘（标准输入）的同时时读输入或从文件读输入。可以通过指定显示或存储数据的位置来控制输出。可以指定将输出数据写到屏幕（标准输出）或将其写到文件。

因为操作系统是多任务的，所以将它设计为在互相组合中处理进程。本章讨论重定向输入和输出的优点和一起输入进程的优点。

本章讨论以下内容:

标准输入、标准输出和标准错误

当命令开始运行时，它通常期望以下文件已打开: 标准输入、标准输出和标准错误（有时称为*错误输出*或*诊断*输出）。称为*文件描述符*的数字与这些文件的每一个都关联，如下所示:

文件描述符 0	标准输入
文件描述符 1	标准输出
文件描述符 2	标准错误（诊断）输出

子进程通常从其父进程继承这些文件。所有三个文件初始指定给工作站（0 给键盘，1 和 2 给显示器）。在控制传递给命令前，shell 允许它们重定向到其它处。

当输入命令时，如果未给出文件名，则您的键盘是*标准输入*，有时指示为 *stdin*。当命令完成时，结果显示在您的屏幕上。

您的屏幕是*标准输出*，有时指示为 *stdout*。缺省情况下，命令从标准输入提取输入，并将结果发送到标准输出。

错误消息定向到标准错误，有时指示为 *stderr*。缺省情况下，这是您的屏幕。

输入和输出的这些缺省操作可以不同。可以将文件用作输入，并将命令的结果写到文件。这称为*输入／输出重定向*。

来自命令的输出（通常转至显示设备）可以容易地改为重定向到文件。这称为*输出重定向*。当具有很多很难在屏幕上读取的输出时，或这当想要将多个文件放在一起以创建一个大的文件时，这很有用。

尽管使用不及输出重定向那样多，但命令的输入（通常来自键盘）也可以从文件重定向。这称为*输入重定向*。输入的重定向让您预先准备文件，然后使命令读文件。

重定向标准输出

当符号 > *filename* 添加到命令的结尾时，命令的输出写到指定的文件名。> 符号也称为*输出重定向运算符*。

任何可将其结果输出到屏幕的命令都可以使其输出发送到文件。

将输出重定向到文件

通过输入后跟文件名的命令，可以将进程的输出重定向到文件。例如，要将 **who** 命令的结果发送到称为 **users** 的文件，请输入:

```
who > users
```

按下 Enter 键。

> **注**: 如果 **users** 文件已存在，请删除并替换它，除非指定了 **set** 内置 **ksh** 的 **noclobber** 选项（Korn shell）或 **csh**（C shell）命令。

要查看 **users** 文件的内容，请输入:

```
cat users
```

按下 Enter 键。

显示一个类似于以下的列表:

```
denise    lft/0 5 月 13 日 08:05
marta     pts/1 5 月 13 日 08:10
endrica   pts/2 5 月 13 日 09:33
```

重定向输出和附加到文件

当符号 >> *filename* 添加到命令的结尾时，命令的输出附加到指定的文件名，而不是写在任何现有数据上。>> 符号也称为*附加重定向运算符*。

例如，要将 **file2** 附加到 **file1**，请输入:

```
cat file2 >> file1
```

按下 Enter 键。

注: 如果 **file1** 文件不存在，请创建它，除非指定了 **set** 内置 **ksh** 的 **noclobber** 选项（Korn shell）或 **csh**（C shell）命令。

使用来自键盘的重定向创建文本文件

单独使用时，**cat** 命令将您在键盘输入的任何内容用作输入。可以将此输入重定向到文件。在新行上输入 Ctrl-D，用信号通知文本结束。

请在系统提示符下输入:

```
cat > filename
This is a test.
^D
```

连接文本文件

将各种文件合并到一个文件中称为*连接*。

例如，在系统提示符下输入:

```
cat file1 file2 file3 > file4
```

按下 Enter 键。

前面的示例创建 file4，它由 file1、file2 和 file3 以给定顺序附加而组成。

下列示例显示连接文件时的常见错误:

```
cat file1 file2 file3 > file1
```

注意: 在此示例中，可能期望 **cat** 命令将 file1、file2 和 file3 的内容附加到 file1 中。**cat** 命令首先创建输出文件，因此它实际上擦除 file1 的内容，然后将 file2 和 file3 附加到它。

重定向标准输入

当符号 < *filename* 添加到命令的结尾时，命令的输入从指定的文件名读取。< 符号称为*输入重定向运算符*。

注: 只有通常从键盘取其输入的命令才可使其输入重定向。

例如，要使用 **mail** 命令将文件 letter1 作为消息发送给用户 denise，请输入:

```
mail denise < letter1
```

按下 Enter 键。

使用 /dev/null 文件废弃输出

/dev/null 文件是特殊文件。此文件具有唯一的属性；它总是空的。发送到 **/dev/null** 的任何数据废弃。当运行生成想要忽略的输出的程序或命令时，这是一个有用的功能。

例如，您有一个名为 myprog 的程序，它接受来自屏幕的输入，并在运行时生成您要忽略的消息。要从文件 myscript 读输入，并废弃标准输出消息，请输入:

```
myprog < myscript >/dev/null
```

按下 Enter 键。

在此例中，myprog 将文件 myscript 用作输入，且废弃所有标准输出。

重定向标准错误和其它输出

除标准输入和标准输出外，命令通常还产生其它类型的输出，如称为诊断输出的错误或状态消息。像标准输出一样，标准错误输出写到屏幕，除非重定向。

如果想要重定向标准错误或其它输出，则必须使用文件描述符。*文件描述符*是与命令通常使用的每个 I/O 文件关联的数字。还可以指定文件描述符以重定向标准输入和标准输出，但这已是缺省值。以下数字与标准输入、输出和错误关联:

0　　标准输入（键盘）
1　　标准输出（显示器）
2　　标准错误（显示器）

要重定向标准错误输出，在输出前输入文件描述符数字 2，或在符号后附加重定向符号（> 或 > >）和文件名。例如，以下命令从用于编译 **testfile.c** 文件的 **cc** 命令进行标准错误输出，并将它附加到 **ERRORS** 文件的末尾。

```
cc testfile.c 2 >> ERRORS
```

还可使用从 0 到 9 的文件描述符重定向其它类型的输出。例如，如果 **cmd** 命令将输出写到文件描述符 9，则可以使用以下命令将输出重定向到 **savedata** 文件:

```
cmd 9> savedata
```

如果命令写到不止一个输出，则可以独立地重定向每个输出。假定命令将其标准输出定向到文件描述符 1，将其标准错误定向到文件描述符 2，并在文件描述符 9 上构建数据文件。以下命令行将这些输出的每一个重定向到不同的文件:

```
command > standard 2> error 9> data
```

使用直接插入输入（Here）文档

如果命令为以下格式:

```
command << eofstring
```

且 *eofstring* 是不包含模式匹配字符的任何字符串，则 shell 将后继行视为 *command* 的标准输入，直到 shell 读到仅由 *eofstring* 组成的行（可能前有一个或多个制表符）。第一个 *eofstring* 和第二个之间的行经常指一个*直接插入输入*文档或 *here* 文档。如果有连字符（-）紧跟在 << 重定向字符后，则 shell 在将行传递到 *command* 前，从 here 文档的每行中舍去前导制表符。

shell 创建一个包含 here 文档的临时文件，并在将文件传递到命令前对内容执行变量替换和命令替换。它对作为命令替换中命令行的一部分的文件名执行模式匹配。要禁止所有替换，引证任何 *eofstring* 字符:

```
command << \eofstring
```

对于更方便放在 shell 步骤中（而非保留在独立的文件（如编辑器脚本）中）的少量输入数据，here 文档特别有用。例如，可以输入:

```
cat <<- xyz
   将在显示器上显示此消息，
   并且前导制表符已除去。
   xyz
```

按下 Enter 键。

使用管道和过滤器

可以连接两个或多个命令，以便将一个命令的标准输出用作另一个命令的标准输入。以此方式连接的一组命令称为*流水线*。连接命令的连接称为*管道*。因为管道让您将很多单一目的的命令结合成一个功效强大的命令，所以它很有用。

可以使用流水线将来自一个命令的输出定向为另一个命令的输入。命令是用管道（|）符号连接的。

当命令从另一个命令取其输入时，请修改它，并将其结果发送到标准输出，它称为*过滤器*。过滤器可单独使用，但它们在流水线中特别有用。最常见的过滤器如下:

- sort
- more
- pg

例如，**ls** 命令将当前目录的内容写到一个滚动数据流中的屏幕。当存在不止一屏的信息时，一些数据从视图中丢失。要控制输出以一屏一屏地显示内容，则可以使用流水线将 **ls** 命令的输出定向到 **pg** 命令，该命令控制输出到屏幕的格式，如以下示例所示:

```
ls | pg
```

在该示例中，**ls** 命令的输出是 **pg** 命令的输入。按下 Enter 键继续下一个屏幕。

流水线操作仅是一个方向的（左至右）。流水线中的每个命令都作为独立的进程运行，并且所有进程可以同时运行。当进程没有要读的输入，或这当到下一个进程的管道已满时，进程暂停。

另一个使用管道的示例是使用 **grep** 命令。**grep** 命令搜索文件，以查找包含某些模式的字符串的行。要显示所有在 7 月创建或修改的文件，请输入:

```
ls -l | grep Jul
```

按下 Enter 键。

在示例中，**ls** 命令的输出是 **grep** 命令的输入。

显示程序输出并复制到文件（tee 命令）

与管道一起使用的 **tee** 命令读取标准输入，然后将程序的输出写到标准输出，并同步地将它复制到指定的文件或多个文件中。使用 **tee** 命令立即查看您的输出，并同时存储它以便将来使用。

例如，请输入:

```
ps -ef | tee program.ps
```

按下 Enter 键。

这在显示设备上显示 **ps -ef** 命令的标准输出，并且同时将其副本保存在 **program.ps** 文件中。如果 **program.ps** 文件已存在，则删除并替换它，除非指定了 **set** 内置命令的 **noclobber** 选项。

例如，要查看命令的输出并将其保存到现有文件:

```
ls -l | tee -a program.ls
```

这在显示设备上显示 **ls -l** 的标准输出，并且同时将其副本附加到 **program.ls** 文件的末尾。

此系统显示类似于以下的信息，并且 **program.ls** 文件包含相同的信息:

```
-rw-rw-rw-   1 jones   工作人员   2301   9 月 19 日    08:53 161414
-rw-rw-rw-   1 jones   工作人员   6317   8 月 31 日    13:17 def.rpt
-rw-rw-rw-   1 jones   工作人员   5550   9 月 10 日    14:13 try.doc
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **tee** 命令，以获取完整的语法。

清除屏幕（clear 命令）

可以使用 **clear** 命令清空消息和键盘输入的屏幕。

在提示符下输入:

```
clear
```

按下 Enter 键。

系统清除屏幕并显示提示符。

将消息发送到标准输出（echo 命令）

可以使用 **echo** 命令在屏幕上显示消息。

例如，要将消息写到标准输出，请在提示符下输入:

```
echo Please insert diskette . . .
```

按下 Enter 键。

系统显示如下内容:

```
Please insert diskette . . .
```

例如，要将 **echo** 命令与模式匹配字符一起使用，请在提示符下输入:

```
echo The back-up files are: *.bak
```

按下 Enter 键。

系统显示消息 `The back-up files are:` 后跟当前目录中以 `.bak` 结尾的文件名。

将单个文本行附加到文件（echo 命令）

可以使用 **echo** 命令将单个文本行添加到文件，与附加重定向运算符一起使用。

例如，在提示符下输入:

```
echo Remember to backup mail files by end of week.>

>notes
```

按下 Enter 键。

这将消息 Remember to backup mail files by end of week. 添加到文件 notes 的末尾。

将您的屏幕复制到文件（capture 和 script 命令）

您可将终端上打印的所有内容复制到使用 **capture** 命令指定的文件，该命令仿真 VT100 终端。

可以使用 **script** 命令将终端上打印的所有内容复制到指定的文件，无需仿真 VT100 终端。

这两个命令对打印终端对话的记录都有用。

例如，要在仿真 VT100 的同时捕捉终端屏幕，请在提示符下输入:

```
capture screen.01
```

按 Enter 键。

系统显示类似于以下的信息:

```
以启动 capture 命令。文件为 screen.01。
使用 ^P 将屏幕转储到文件 screen.01。
现在正在仿真 vt100 终端。
按下任意键继续。
```

输入数据和转储屏幕内容后，请通过按 Ctrl-D 或输入 exit 并按下 Enter 键停止 **capture** 命令。系统显示类似于以下的信息:

```
capture 命令完成。文件为 screen.01。
您将不再仿真 vt100 终端。
```

使用 **cat** 命令显示文件的内容。

例如，要捕捉终端的屏幕，请在提示符下输入:

```
script
```

按下 Enter 键。

系统显示类似于以下的信息:

```
已启用 script 命令。文件为 typescript。
```

屏幕上显示的所有内容现在都复制到 **typescript** 文件。

要停止 **script** 命令，请按下 Ctrl-D 或输入 exit 并按下 Enter 键。系统显示类似于以下的信息:

```
script 命令已完成。文件为 typescript。
```

使用 **cat** 命令显示文件的内容。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **capture** 和 **script** 命令，以获取完整的语法。

在屏幕上以大字体显示文本（banner 命令）

banner 命令以大字体在您的屏幕上显示 ASCII 字符。输出中每一行的长度最多为 10 个数字（或大写字符或小写字符）。

例如，在提示符下输入:

```
banner GOODBYE!
```

按下 Enter 键。

系统以大字体在屏幕上显示 GOODBYE!。

输入和输出重定向的命令摘要

>	第 46 页的『重定向标准输出』
<	第 47 页的『重定向标准输入』
> >	第 46 页的『重定向输出和附加到文件』
|	第 49 页的『使用管道和过滤器』
banner	将 ASCII 字符串以大字母写到标准输出
capture	允许终端屏幕转储到文件
clear	清除终端屏幕
echo	将字符串写入标准输出
script	允许终端输入和输出复制到文件
tee	显示程序的标准输出并将其复制到文件

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 135 页的第 12 章，『shell』

第 140 页的『Korn shell 或 POSIX shell 命令』

第 174 页的『Bourne shell』

第 188 页的『C shell』

第 65 页的第 7 章，『文件』

第 6 章 文件系统和目录

*文件系统*由目录组及目录中的文件组成。文件系统通常表示为反转的树。由斜杠（/）符号表示的根目录定义文件系统，并出现在文件系统树图的顶部。目录分支从树图中的根目录向下，并同时包含文件和子目录。分支通过目录结构创建到文件系统中的每个对象的唯一路径。

文件的集合存储在*目录*中。这些文件的集合通常彼此相关；以目录结构存储它们可使它们有组织地保存。

*文件*是可以读取的数据或写入的数据的集合。文件可以是您创建的程序、写的文本、获取的数据或使用的设备。命令、打印机、终端、通信和应用程序都存储在文件中。这允许用户以统一的方式访问系统的不同元素，给文件系统极大的灵活性。

本章讨论以下内容:

文件系统

*文件系统*是文件和目录的分层结构（文件树）。这种类型的结构类似一棵反转的树，根在顶部而分支在底部。此文件树使用目录来将数据和程序组织到各个组中，允许一次管理很多目录和文件。

一些任务在文件系统上执行要比在文件系统中每个目录上执行更有效。例如，可以备份、移动或保护整个文件系统。

文件系统的基本类型称为*记录文件系统（JFS）*。此文件系统使用数据库日志记录技术来维护其结构一致性。这防止当系统异常停机时对文件系统的损坏。

一些最重要的系统管理任务与文件系统有关，特别是:

- 为逻辑卷上的文件系统分配空间
- 创建文件系统
- 使文件系统空间可供系统用户使用
- 监控文件系统空间的使用
- 备份文件系统以保护系统发生故障时数据不丢失
- 将文件系统维护为一致的状态

这些任务应由您的系统管理员执行。

文件系统类型

操作系统支持多种文件系统类型。这些类型包括:

记录文件系统（JFS） 基本文件系统类型，它支持整个文件系统命令集合。

增强的记录文件系统（JFS2） 基本文件系统类型，它支持整个文件系统命令集合。

网络文件系统（NFS） 一种文件系统类型，允许访问驻留在远程机器上的文件，就像它们驻留在本地机器上一样。

CD-ROM 文件系统（CDRFS） 一种文件系统类型，允许通过正常文件系统接口（打开、读和关闭）访问 CD-ROM 的内容。

文件系统结构

在独立机器上，以下文件系统在缺省情况下驻留在关联的设备上:

/File System	/Device
/dev/hd1	/home
/dev/hd2	/usr
/dev/hd3	/tmp
/dev/hd4	/(root)
/dev/hd9var	/var
/proc	/proc
/dev/hd10opt	/opt

文件树具有以下特征:

- 可由具有相同硬件体系结构的机器共享的文件位于 **/usr** 文件系统中。
- 每客户机文件的变量（例如，假脱机和邮件文件）位于 **/var** 文件系统中。
- **/(root)** 文件系统包含对操作系统很关键的文件和目录。例如，它包含
  - 设备目录（**/dev**）
  - 文件系统可安装到根文件系统上的安装点，例如 **/mnt**
- **/home** 文件系统是用户主目录的安装点。
- 对于服务器，**/export** 目录包含调页空间文件、每客户机（非共享的）根文件系统、转储、主目录和用于无磁盘客户机的 **/usr/share** 目录，以及导出的 **/usr** 目录。
- **/proc** 文件系统包含有关系统中进程和线程状态的信息。
- **/opt** 文件系统包含可选软件，如应用程序。

以下列表提供有关 **/(root)** 文件系统的一些子目录的内容。

/bin **/usr/bin** 目录的符号链接。

/dev 包含本地设备的特殊文件的设备节点。**/dev** 目录包含用于磁带机、打印机、磁盘分区和终端的特殊文件。

/etc 包含每个机器不同的配置文件。示例包括:

- **/etc/hosts**
- **/etc/passwd**

/export 包含服务器上用于远程客户机的目录和文件。

/home 充当包含用户主目录的文件系统的安装点。**/home** 文件系统包含每用户文件和目录。

在独立机器中，独立的本地文件系统安装在 **/home** 目录上。在网络中，服务器可能包含应可从几个机器访问的用户文件。在这种情况下，**/home**目录的服务器副本远程地安装到本地 **/home** 文件系统上。

/lib **/usr/lib** 目录的符号链接，它包含名称为 **lib*.a** 格式的独立于体系结构的库。

/sbin 包含引导机器和安装 **/usr** 文件系统所需要的文件。引导期间使用的大多数命令来自引导映象的 RAM 磁盘文件系统；因此，很少文件驻留在 **/sbin** 目录中。

/tmp 充当包含系统生成的临时文件的文件系统的安装点。

/u **至 /home** 目录的符号链接。

/usr 充当文件系统的安装点，该文件系统包含不更改并且可由机器共享的文件（如可执行程序和 ASCII 文档）。

独立机器在 **/usr** 目录上安装独立的本地文件系统。无磁盘和磁盘贫乏的机器在 **/usr** 文件系统上安装来自远程服务器的目录。

/var 充当在每个机器上不同的文件的安装点。**/var** 文件系统配置成一个文件系统，因为它所包含的文件趋向增长。例如，它是 **/usr/tmp** 目录的符号链接，该目录包含临时工作文件。

显示文件系统上的可用空间（df 命令）

可以使用 **df** 命令显示有关文件系统上总计空间和可用空间的信息。*FileSystem* 参数指定文件系统驻留的设备的名称，文件系统安装的目录，或文件系统的相对路径名。如果不指定 *FileSystem* 参数，则 **df** 命令显示所有当前安装的文件系统的信息。如果指定文件或目录，则 **df** 命令显示文件或文件系统驻留的文件系统的信息。

通常，**df** 命令使用包含在超级块中的空闲计数。在某些例外条件下，这些计数可能错误。例如，如果当 **df** 命令正在运行时文件系统被实时地修改，则空闲计数可能不准确。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **df** 命令，以获取完整的语法。

注: 在某些远程文件系统（如网络文件系统（NFS））上，如果服务器不提供信息，则显示器上表示可用空间的列保留空白。

以下是如何使用 **df** 命令的示例:

1. 要显示有关所有已安装文件系统的信息，请输入:

   ```
   df
   ```

   按下 Enter 键。

   如果您的系统配置成 **/**、**/usr**、**/site** 和 **/usr/venus** 目录驻留在独立的文件系统中，则 **df** 命令的输出类似于以下:

   ```
   Filesystem 512-blocks  free   %used   Iused %Iused   Mounted on
   /dev/hd4    20480     13780    32%      805   13%     /
   /dev/hd2   385024     15772    95%    27715   28%     /usr
   /dev/hd9var 40960     38988     4%      115    1%     /var
   /dev/hd3    20480     18972     7%       81    1%     /tmp
   /dev/hd1     4096      3724     9%       44    4%     /home
   ```

2. 要显示您当前目录驻留的文件系统上的可用空间，请输入:

   ```
   df .
   ```

   按下 Enter 键。

目录概述

*目录*是一种独特的文件类型，它仅包含访问文件或其它目录所需要的信息。因此，目录占用的空间要比其它文件类型占用的空间少。目录使您能对文件和其它目录分组，允许您将文件系统组织到模块化的层次结构中，并给出文件系统结构灵活性和深度。与其它类型的文件不同，特殊的命令集控制目录。

目录包含目录条目。每个条目包含文件或子目录名，以及索引节点引用号（*i* 节点号）。为了提高速度和增强磁盘空间的使用，文件中的数据存储在计算机内存的各个位置。 i 节点号包含用于定位与文件关联的所有散射数据块的地址。i 节点号还记录有关文件的其它信息，包括修改和访问的时间、访问方式、链接数、文件所有者和文件类型。将一个文件的几个名称链接到同一个 i 节点号是可能的，方法是使用 **ln** 命令创建目录条目。

因为目录经常包含不应向系统的所有用户提供的信息，所以可以保护目录访问。通过设置目录的许可权，可以控制谁对目录有访问权，同时确定哪些用户（如果有）可改变目录中的信息。请参阅第 115 页的『文件和目录访问方式』以获取更多信息。

本节讨论:

目录的类型

目录可由操作系统、系统管理员或用户定义。系统定义的目录包含特定类型的系统文件，如命令。文件系统层次结构的顶部是系统定义的 **/(root)** 目录。**/(root)** 目录通常包含以下标准系统相关目录:

/dev	包含 I/O 设备的特殊文件。
/etc	包含系统初始化和系统管理的文件。
/home	包含系统用户的登录目录。
/tmp	包含临时的并可在指定天数内删除的文件。
/usr	包含 **lpp**、**include** 和其它系统目录。
/usr/bin	包含用户可执行程序。

一些目录（如您的登录目录或主目录（**$HOME**））由系统管理员定义和定制。当登录到操作系统时，登录目录是当前目录。

您创建的目录称为用户定义的目录。这些目录帮助您组织和维护您的文件。

目录组织

目录包含文件、子目录或两者的组合。子目录是目录中的目录。包含子目录的目录称为*父目录*。

操作系统为跟踪和查找目录，每个目录都具有它在其中创建的父目录的条目，即 ..（点点），以及目录自身的条目，即 .（点）。在大多数目录列表中，这些文件是隐藏的。

目录树

文件系统目录结构可以很容易变得复杂。尝试保持文件和目录结构尽可能简单。而且，使用容易识别的名称创建文件和目录。这使得使用文件更容易。

父目录

除 ***/*(root)** 外的每个目录都具有父目录，并且可以具有零个或更多子目录。

主目录

当登录时，系统将您置于称为您的*主目录*的目录或登录目录中。此目录是由系统管理员为每个用户设置的。主目录是您个人文件的资源库。通常，您为自己使用而创建的目录将是主目录的子目录。要在任何时候返回您的主目录，请输入 **cd** 命令并在提示符下按 Enter 键。

工作目录

您总是在某一目录中工作。当前在其中工作的那个目录称为您的*当前*目录或*工作*目录。**pwd**（呈现工作目录）命令报告工作目录的名称。使用 **cd** 命令更改工作目录。

目录命名约定

每个目录的名称必须在它存储的目录中是唯一的。这确保目录在文件系统中还具有唯一的路径名。目录遵循与文件相同的命名约定，如第 66 页的『文件命名约定』中解释。

目录路径名

每个文件和目录都可以通过文件系统树结构中的唯一的路径（称为*路径名*）到达。路径名指定文件系统中目录或文件的位置。

注: 路径名的长度不能超过 1023 个字符。

文件系统使用以下种类的路径名:

绝对路径名	从 ***/*(root)** 目录跟踪路径。绝对路径名总是以斜杠（/）符号开始。
相对路径名	从当前目录，通过其父目录或其子目录和文件跟踪路径。

绝对路径名代表从 ***/*(root)** 目录向下的目录或文件的完整名称。无论您在文件系统中的哪一位置工作，您总是可以通过指定其绝对路径名来查找目录或文件。绝对路径名以斜杠（/）开始，该符号代表根目录。路径名 **/A/D/9** 是 **9** 的绝对路径名。第一个斜杠（/）代表 ***/*(root)** 目录，它是搜索的开始位置。路径名的其余部分将搜索定向 **A**，然后是 **D**，最后是 **9**。

可以存在两个命名为 **9** 的文件，因为文件的绝对路径名给每个文件在文件系统中唯一的名称。路径名 **/A/D/9** 和 **/C/E/G/9** 指定两个名为 **9** 的唯一文件。

与全路径名不同，相对路径名基于当前工作目录指定目录或文件。对于相对路径名，可以使用符号点点（..）在文件系统层次结构中向上移动。点点（..）代表父目录。由于相对路径名指定从当前目录开始的路径，所以它们不以斜杠（/）开始。相对路径名用于指定当前目录中文件的名称，或指定文件系统中当前目录层之上或之下的文件或目录的路径名。如果 **D** 是当前目录，则用于访问 **10** 的相对路径名是 **F/10**，但绝对路径名总是 **/A/D/F/10**。而且，用于访问 3 的相对路径名是 **../../B/3**。

还可以通过使用符号点（.）来代表当前目录的名称。点（.）符号通常在运行读取当前目录名的程序时使用。

目录缩写

缩写提供方便的方式来指定特定的目录。以下是缩写的列表。

缩写	含义
.	当前工作目录。
..	当前工作目录的上一层目录（父目录）。
~	您的主目录（对于 Bourne shell 不是这样。有关更多信息，请参阅第 174 页的『Bourne shell』）。
$HOME	您的主目录（对于所有 shell 都是这样）。

目录处理过程

可以各种方式使用目录及其内容。

为以下每个目录任务，呈现命令和一个示例:

- 『创建目录（mkdir 命令）』
- 第 59 页的『移动或重命名目录（mvdir 命令）』
- 第 59 页的『显示当前目录（pwd 命令）』
- 第 59 页的『更改到另一个目录（cd 命令）』
- 第 60 页的『复制目录（cp 命令）』
- 第 60 页的『显示目录的内容（ls 命令）』
- 第 62 页的『删除或除去目录（rmdir 命令）』
- 第 63 页的『比较目录的内容（dircmp 命令）』

创建目录（mkdir 命令）

可以使用 **mkdir** 命令创建一个或多个 *Directory* 参数指定的目录。每个新目录包含标准条目点（.）和点点（..）。可以使用 **-m** *Mode* 标志为新目录指定许可权。

当创建目录时，它在当前目录或工作目录中创建，除非您将绝对路径名指定到文件系统中的另一个位置。

以下是如何使用 **mkdir** 命令的示例:

1. 要在当前工作目录中创建具有缺省许可权的称为 **Test** 的新目录，请输入:

   ```
   mkdir Test
   ```

   按下 Enter 键。
2. 要在先前创建的 **/home/demo/sub1** 目录中创建具有 rwxr-xr-x 许可权的名为 **Test** 的目录，请输入:

   ```
   mkdir -m 755 /home/demo/sub1/Test
   ```

   按下 Enter 键。
3. 要在 **/home/demo/sub2** 目录中创建具有缺省许可权的名为 **Test** 的目录，请输入:

   ```
   mkdir -p /home/demo/sub2/Test
   ```

   按下 Enter 键。

   -p 标志创建 **/home**、**/home/demo** 和 **/home/demo/sub2** 目录（如果它们还不存在）。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **mkdir** 命令，以获取完整的语法。

移动或重命名目录（mvdir 命令）

要移动或重命名目录，请使用 **mvdir** 命令。

例如，要移动目录，请输入:

```
mvdir book manual
```

按下 Enter 键。

这将 **book** 目录移到名为 **manual** 的目录下（如果 **manual** 目录存在）。否则，**book** 目录重命名为 **manual**。

例如，要移动和重命名目录，请输入:

```
mvdir book3 proj4/manual
```

按下 Enter 键。

这将 **book3** 目录移到名为 **proj4** 的目录，并将 **proj4** 重命名为 **manual**（如果 **manual** 目录先前不存在）。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **mvdir** 命令，以获取完整的语法。

显示当前目录（pwd 命令）

可以使用 **pwd** 命令将当前目录的全路径名（从 **/(root)** 目录）写到标准输出。所有目录用斜杠（/）分隔。**/(root)** 目录用第一个斜杠（/）表示，最后命名的目录是您的当前目录。

例如，要显示当前目录，请输入:

```
pwd
```

按下 Enter 键。

当前目录的全路径名显示类似于以下:

```
/home/thomas
```

更改到另一个目录（cd 命令）

cd 命令将您从现在的目录移动到另一个目录。必须在指定的目录中有执行（搜索）许可权。

如果不指定 *Directory* 参数，则 **cd** 命令能将您移动到您的登录目录（**ksh** 和 **bsh** 环境中的 **$HOME** 或 **csh** 环境中的 **$home**）。如果指定的目录名是全路径名，则它变成当前目录。全路径名以斜杠（/）开始表示 **/(root)** 目录，以一个点（.）开始表示当前目录，或以点点（..）开始表示父目录。如果目录名不是全路径名，则 **cd** 命令搜索它相对于 **$CDPATH** shell 变量（或 **$cdpath csh** 变量）指定的路径中的一个。该变量具有与 **$PATH** shell 变量（或 **$path csh** 变量）相同的语法和类似的语义。

以下是如何使用 **cd** 命令的示例:

1. 要更改到您的主目录，请输入:

   ```
   cd
   ```

   按下 Enter 键。
2. 要更改到 **/usr/include** 目录，请输入:

```
cd /usr/include
```

按 Enter 键。

3. 要将目录树向下一个级别到 **sys** 目录，请输入:

```
cd sys
```

按下 Enter 键。

如果当前目录是 **/usr/include** 并且它包含名为 **sys** 的子目录，则 **/usr/include/sys** 变成当前目录。

4. 要将目录树向上一个级别，请输入:

```
cd ..
```

按下 Enter 键。

特殊文件名点点（..）指当前目录的上一层目录，即其父目录。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cd** 命令，以获取完整的语法。

复制目录（cp 命令）

可以使用 **cp** 命令将 *SourceFile* 或 *SourceDirectory* 参数指定的文件或目录的内容的副本创建到 *TargetFile* 或 *TargetDirectory* 参数指定的文件或目录。如果指定为 *TargetFile* 的文件已存在，则副本在文件的原始内容上写。如果复制不止一个 *SourceFile*，则目标必须是目录。

要将 *SourceFile* 的副本放置到目录中，请为 *TargetDirectory* 参数指定现有目录的路径。文件在其复制到目录时维护其各自的名称，除非您在路径的结尾指定新文件名。如果指定 **-r** 或 **-R** 标志，则 **cp** 命令还将整个目录复制到其它目录。

以下是如何使用 **cp** 命令的示例。

1. 要将 **/home/accounts/customers/orders** 目录中的所有文件复制到 **/home/accounts/customers/shipments** 目录，请输入:

```
cp /home/accounts/customers/orders/* /home/accounts/customers/shipments
```

按下 Enter 键。

这只将 **orders** 目录中的文件复制到 **shipments** 目录。

2. 要将目录（包括所有其文件和子目录）复制到另一个目录，请输入:

```
cp -R /home/accounts/customers /home/accounts/vendors
```

按下 Enter 键。

这将 **customers** 目录（包括所有其文件、子目录和那些子目录中的文件）复制到 **vendors** 目录。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cp** 命令，以获取完整的语法。

显示目录的内容（ls 命令）

可以通过使用 **ls** 命令显示目录的内容。

ls 命令

ls 命令将每个指定*目录*的内容或每个指定*文件*的名称以及使用标志请求的任何其它信息写入标准输出。如果未指定*文件*或*目录*，则 **ls** 命令显示当前目录的内容。

缺省情况下，**ls** 命令显示按文件名字母顺序排列的所有信息。如果命令由具有 root 权限的用户执行，则它在缺省情况下使用 **-A** 标志，列出除点（.）和点点（..）以外的所有条目。要显示所有文件条目，包括以 .（点）开始的那些文件，请使用 **ls -a** 命令。

可以使用以下方式格式化输出:

- 每行列出一个条目，使用 **-l** 标志。
- 在多个列中列出条目，通过指定 **-C** 或 **-x** 标志。当输出是 tty 时 **-C** 标志是缺省格式。
- 在以逗号分隔的系列中列出条目，通过指定 **-m** 标志。

要确定输出行中字符位置的数目，**ls** 命令使用 **$COLUMNS** 环境变量。如果此变量未设置，则命令读 **terminfo** 文件。如果 **ls** 命令不能通过这些方法的任何一种确定字符位置的数目，则它使用缺省值 80。

使用 **-e** 和 **-l** 标志显示的信息解释如下:

第一个字符可能是以下之一:

d 条目是一个目录。
b 条目是一个块特殊文件。
c 条目是一个字符特殊文件。
l 条目是一个符号链接。
p 条目是一个先进先出（FIFO）的管道特殊文件。
s 条目是一个本地套接字。
- 条目是一个普通文件。

接下来九个字符分成三组，每组三个字符。前三个字符显示所有者的许可权。接下的一组三个字符显示组中其它用户的许可权。最后一组三个字符显示具有文件访问权的其它任何人的许可权。每组中的三个字符显示文件的读、写和执行许可权。目录的执行许可权让您搜索目录以查找指定文件。

许可权表示如下:

r 授予读许可权
t 只有目录所有者或文件所有者才可以删除或重命名该目录中的文件，即使其它人具有目录的写许可权亦是如此。
w 授予写（编辑）许可权
x 授予执行（搜索）许可权
- 不授予相应的许可权。

使用 **-e** 标志显示的信息与使用 **-l** 标志显示的信息相同（除了添加第 11 个字符），解释如下:

+ 表示文件具有扩展的安全性信息。例如，文件可能在该方式下具有扩展的 **ACL**、**TCB** 或 **TP** 属性。
- 表示文件不具有扩展的安全性信息。

当列出目录中文件的大小时，**ls** 命令显示总计块数，包括间接块。

例如，要列出当前目录中的所有文件，请输入:

```
ls -a
```

按下 Enter 键。

它列出所有文件，包括

- 点（.）
- 点点（..）
- 其它名称可能以或不以点（.）开始的文件

例如，要显示详细的信息，请输入:

```
ls -l chap1 .profile
```

按下 Enter 键。

它显示具有关于 **chap1** 和 **.profile** 的长列表。

例如，要显示有关目录的详细信息，请输入:

```
ls -d -l . manual manual/chap1
```

按下 Enter 键。

它显示有关目录 **.** 和 **manual** 以及文件 **manual/chap1** 的长列表。如果没有 **-d** 标志，它将列出 **.** 和 **manual** 目录中的文件，而不是有关目录本身的详细信息。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **ls** 命令，以获取完整的语法。

删除或除去目录（rmdir 命令）

可以使用 **rmdir** 命令从系统除去 *Directory* 参数指定的目录。在除去目录之前，它必须是空的（它只能包含 . 和 ..），并且必须具有其父目录中的写许可权。使用 **ls -a** *Directory* 命令检查目录是否是空的。

以下是如何使用 **rmdir** 命令的示例:

1. 要清空和除去目录，请输入:

   ```
   rm mydir/* mydir/.*
   rmdir mydir
   ```

   按下 Enter 键。

   这除去 **mydir** 的内容，然后除去空的目录。**rm** 命令显示关于尝试除去目录点（.）和点点（..）的错误消息，然后 **rmdir** 命令除去它们和目录本身。

   注: **rm mydir/* mydir/.*** 首先除去其名称不以点开始的文件，然后除去那些名称以点开始的文件。您可能没有意识到目录包含以点开始的文件名，因为 **ls** 命令通常不列出它们，除非您使用 **-a** 标志。

2. 要除去 **/tmp/jones/demo/mydir** 目录和所有它之下的目录，请输入:

   ```
   cd /tmp
   rmdir -p jones/demo/mydir
   ```

   按下 Enter 键。

   这从 **/tmp** 目录除去 **jones/demo/mydir** 目录。当目录要除去时，如果它不是空的或您不具有对它的写许可权，则此命令以相应的错误消息终止。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **rmdir** 命令，以获取完整的语法。

比较目录的内容（dircmp 命令）

可以使用 **dircmp** 命令比较 *Directory1* 和 *Directory2* 参数指定的两个目录，并将有关它们内容的信息写到标准输出。首先，**dircmp** 命令比较每个目录中的文件名。如果两个目录中包含相同的文件名，则 **dircmp** 命令比较两个文件的内容。

在输出中，**dircmp** 命令列出对每个目录唯一的文件。然后，它列出两个目录中具有名称等同但内容不同的文件。如果未指定标志，则它还列出两个目录中都具有的内容等同且名称等同的文件。

以下是如何使用 **dircmp** 命令的示例:

1. 要摘要显示 **proj.ver1** 和 **proj.ver2** 目录中文件的差异，请输入:

   ```
   dircmp proj.ver1 proj.ver2
   ```

   按下 Enter 键。

   这显示 **proj.ver1** 和 **proj.ver2** 目录之间差异的摘要。此摘要单独地列出只在一个目录或另一个目录中找到的文件，以及同时在两个目录中找到的文件。如果在两个目录中都能找到文件，则 **dircmp** 命令注明两个副本是否等同。
2. 要显示 **proj.ver1** 和 **proj.ver2** 目录中文件差异的详细信息，请输入:

   ```
   dircmp -d -s proj.ver1 proj.ver2
   ```

   按下 Enter 键。

   -s 标志禁止有关等同文件的信息。**-d** 标志为每个在两个目录中都能找到的差异文件显示 **diff** 列表。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **dircmp** 命令，以获取完整的语法。

文件系统和目录的命令摘要

文件系统

df	报告有关文件系统上的空间的信息。

目录缩写

.	当前工作目录。
..	当前工作目录的上一层目录（父目录）。
~	您的主目录（对于 Bourne shell 不是这样。有关更多信息，请参阅第 174 页的『Bourne shell』）。
$HOME	您的主目录（对于所有 shell 都是这样）。

目录处理过程

cd	更改当前目录。
cp	复制文件或目录。
dircmp	比较两个目录及其公共文件的内容。
ls	显示目录的内容。
mkdir	创建一个或多个新的目录。
mvdir	移动（重命名）目录。
pwd	显示工作目录的路径名。

rmdir 除去目录。

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 45 页的第 5 章, 『输入和输出重定向』

第 53 页的『文件系统』

第 56 页的『目录概述』

第 65 页的第 7 章, 『文件』

第 79 页的『链接文件和目录』

第 99 页的第 9 章, 『备份文件和存储介质』

第 113 页的第 10 章, 『文件和系统安全性』

第 7 章 文件

文件用于操作系统中信息的所有输入和输出（I/O）。标准化对软件和硬件的访问。当修改或写入文件内容时，输入发生。当读一个文件的内容或将其内容传送到另一个文件时，输出发生。例如，要创建文件的打印副本，系统从文本文件读信息，并将该信息写到代表打印机的文件。

本章讨论以下内容:

文件类型

存在以下基本文件类型:

常规	存储数据（文本、二进制文件和可执行文件）
目录	包含用于访问其它文件的信息
特殊	定义 FIFO（先进先出）管道文件或物理设备

可由系统识别的所有文件类型属于这些类别之一。然而，操作系统使用这些基本类型的很多变体。

常规文件

常规文件是最常见的文件并用于包含数据。常规文件的形式是文本文件或二进制文件:

文本文件

文本文件是常规文件，包含以 ASCII 方式存储并可由用户读取的信息。可以显示和打印这些文件。文本文件的行绝不能包含 **NUL** 字符，且长度不能超过 **{LINE_MAX}** 字节（包括换行字符）。

术语*文本文件*不排除包含控制字符或其它不可打印的字符（非 **NUL**）。因此，列出文本文件作为输入或输出的标准实用程序，或者能够处理特殊字符，或者它们显式地描述它们在其个别部分中的限制。

二进制文件

二进制文件是常规文件，包含可由计算机读取的信息。二进制文件可能是指示系统完成作业的可执行文件。命令和程序存储在可执行二进制文件中。特殊编译程序将 ASCII 文本转换成二进制代码。

文本文件和二进制文件的差别仅在于：文本文件的行少于 **{LINE_MAX}** 字节，没有 **NUL** 字符，每行都以一个换行字符终止。

目录文件

目录文件包含系统访问所有类型的文件所需要的信息，但目录文件不包含实际的文件数据。因此，目录占用比常规文件少的空格，并给予文件系统结构灵活性和深度。每个目录条目代表一个文件或子目录。每个条目包含文件的名称和文件的索引节点引用号（i 节点号）。 i 节点号指向指定给文件的唯一索引节点。i 节点号描述与文件关联的数据的位置。目录由一组独立的命令创建和控制。

特殊文件

特殊文件定义系统设备或进程创建的临时文件。特殊文件的基本类型是 FIFO（先进先出）、块和字符。FIFO 文件也称为*管道*。管道由一个进程创建以临时地允许与另一个进程的通信。当第一个进程完成时，这些文件不再存在。块文件和字符文件定义设备。

每个文件都具有一组许可权（称为*访问方式*），确定谁可以读、修改或执行文件。

要了解关于文件访问方式的更多信息，请参阅第 115 页的『文件和目录访问方式』。

文件命名约定

每个文件的名称都必须在它存储的目录中是唯一的。这确保文件还在文件系统中具有唯一的路径名。文件命名准则是:

- 文件名最长可为 255 个字符，并可包含字母、数字和下划线。

- 操作系统区分大小写，这意味着它分辨文件名中的大写和小写字母。因此，FILEA、FiLea 和 filea 是三个不同的文件名，即使它们驻留在同一个目录中。
- 文件名应尽可能具有描述性和有意义。
- 目录遵循与文件相同的命名约定。
- 某些字符对操作系统具有特殊的含义。当命名文件时，请避免使用这些字符。这些字符包含以下:

  / \ " ' * ; - ? [ ] ( ) ~ ! $ { } < > # @ & |
- 如果文件名以点（.）开始，则正常目录列表隐藏该文件名。当输入 **ls** 命令时使用 **-a** 标志，则隐藏文件与常规文件和目录一起列出。

文件路径名

文件系统中每个文件和目录的路径名由树结构中该文件或目录之前的每个目录的名称组成。

由于文件系统中的所有路径都发源于 /(**root**) 目录，所以文件系统中的每个文件都具有与根目录的唯一关系，称为*绝对路径名*。绝对路径名以斜杠（/）符号开始。例如，文件 **h** 的绝对路径名可为 **/B/C/h**。请注意，系统中可存在两个名为 **h** 的文件。由于两个文件的绝对路径不同（**/B/h** 和 **/B/C/h**），所以每个名为 **h** 的文件都在系统中具有唯一的名称。路径名的每个组件都是一个目录，除了最后一个组件。路径名的最后一个组件可以是文件名。

注: 路径名的长度不能超过 1023 个字符。

通配符和元字符模式匹配

通配符提供了用一个字符指定多个文件名或目录名的便利方式。通配符是星号（*）和问号（?）。元字符是左右方括号 ([])、连字符 (-) 和感叹号 (!)。

使用 * 通配符

使用星号（*）匹配字符的任何序列或字符串。（*）表示任何字符，包括没有字符。例如，如果在您的目录中具有以下文件:

```
1test 2test afile1 afile2 bfile1 file file1 file10 file2 file3
```

且您只想提及以 file 开始的文件，则将使用:

```
file*
```

选定的文件将是: file file1 file10 file2 file3

要仅提及包含字 file 的文件，将使用:

```
*file*
```

选定的文件将是: afile1 afile2 bfile1 file file1 file10 file2 file3

使用 ? 通配符

使用 ? 匹配任何单个字符。? 表示任何单一字符。

要仅提及以 file 开始并以单一字符结尾的文件，请使用:

```
file?
```

选定的文件将是: file1 file2 file3

要仅提及以 file 开始并以任何两个字符结尾的文件，请使用:

```
file??
```

选定的文件将是: file10

使用 [ ] shell 元字符

元字符通过将期望的字符围在 [ ] 中提供另一种类型的通配符标志法。它类似使用 ?，但它允许您选择匹配的特定字符.[ ] 还允许您使用连字符（-）某一范围的指定值。要指定字母表中的所有字母，请使用 [[:alpha:]].要指定字母表中的所有小写字母，请使用 [[:lower:]].

要仅提及以 1 或 2 结尾的文件，请使用:

```
*file[12]
```

选定的文件将是: afile1 afile2 file1 file2

要仅提及以任何数字开始的文件，请使用:

```
[0123456789]* 或 [0-9]*
```

选定的文件将是: 1test 2test

要仅提及不以 a 开始的文件，请使用:

```
[!a]*
```

选定的文件将是: 1test 2test bfile1 file file1 file10 file2 file3

模式匹配与正则表达式

正则表达式允许您从一组字符串中选择特定的字符串。正则表达式的使用一般与文本处理关联。

正则表达式可以表示很多种可能的字符串。尽管取决于当前语言环境，很多正则表达式可进行不同的解释，但国际化功能提供了不同语言环境间上下文的不变性。

查看“文件匹配模式”与“正则表达式”之间的以下比较中的示例:

模式匹配	正则表达式
*	.*
?	.
[!a]	[^a]
[abc]	[abc]
[[:alpha:]]	[[:alpha:]]

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **awk** 命令，以获取准确的语法。

文件处理过程

有很多方式在您的系统上处理文件。通常使用文本编辑器来创建文本文件。UNIX 环境中的常用编辑器是 **vi** 和 **ed**。因为有几个文本编辑器可用，所以可以选择用您觉得舒适的编辑器来编辑。

还可以通过使用输入和输出重定向来创建文件，如“第 45 页的第 5 章，『输入和输出重定向』”中所述。可以将命令的输出发送到新的文件或将其附加到现有文件。

在创建和修改文件后，可能必须复制文件或将文件从一个目录复制或移动到另一个目录，重命名文件以分辨文件的不同版本，或给同一文件不同的名称。您还可能在不同项目上工作时需要创建目录。

另外，可能需要删除某些文件。您的目录可很快变得混乱，内有包含旧信息或无用信息的文件。要在您的系统上释放存储空间，请确保您删除的是不再需要的文件。

本节讨论以下内容:

删除文件（rm 命令）

当不再需要文件时，可以使用 **rm** 命令除去它。**rm** 命令从目录中的列表除去指定文件、文件组或某些选定文件的条目。当使用 **rm** 命令时，在除去文件前，不需要用户确认、读许可权和写许可权。但是，您必须对包含该文件的目录有写许可权。

以下是如何使用 **rm** 命令的示例:

1. 要删除名为 **myfile** 的文件，请输入:

   ```
   rm myfile
   ```

   按 Enter 键。
2. 要一个一个地删除 **mydir** 目录中的所有文件，请输入:

   ```
   rm -i mydir/*
   ```

   按下 Enter 键。

   显示每个文件名后，请输入 y 并按下 Enter 键删除该文件。或者要保留该文件，只要按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **rm** 命令，以获取完整的语法。

移动和重命名文件（mv 命令）

要将文件和目录从一个目录移动到另一个目录或重命名文件或目录，请使用 **mv** 命令。如果您不指定新名称就将文件或目录移动到新目录，则它保留其原始名称。

警告: **mv** 命令可以覆盖许多现有文件，除非指定 **-i** 标志。**-i** 标志在它覆盖文件前提示您确认。**-f** 标志不提示您。如果在组合中同时指定 **-f** 和 **-i** 标志，则最后指定的标志获得优先权。

使用 mv 命令移动文件

以下是如何使用 **mv** 命令的示例:

1. 要将文件移动到另一个目录并赋予新名称，请输入:

   ```
   mv intro manual/chap1
   ```

   按下 Enter 键。

   这将移动 **intro** 文件到 **manual/chap1** 目录。从当前目录除去名称 **intro**，并且相同的文件作为 **manual** 目录中的 **chap1** 出现。
2. 要将文件移动到另一个目录，并保留相同的名称，请输入:

   ```
   mv chap3 manual
   ```

   按下 Enter 键。

   这将 **chap3** 移动到 **manual/chap3**。

使用 mv 命令重命名文件

可以使用 **mv** 命令更改文件名，而无需将它移动到另一个目录。

要重命名文件，请输入:

```
mv appendix apndx.a
```

按下 Enter 键。

这将 **appendix** 文件重命名为 **apndx.a**。如果名为 **apndx.a** 的文件已存在，则它的旧内容由 **appendix** 文件中的内容替换。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **mv** 命令，以获取完整的语法。

复制文件（cp 命令）

您可使用 **cp** 命令创建 *SourceFile* 或 *SourceDirectory* 参数指定的文件或目录的内容的副本到 *TargetFile* 或 *TargetDirectory* 参数指定的文件或目录。如果 *TargetFile* 存在时文件指定，则副本在文件的原始内容上写而不发出警告。如果您复制多个 *SourceFile*，则目标必须是一个目录。

如果有相同名称的文件存在于新目的地，则复制的文件覆盖新目的地的文件。因此，最好为文件的副本指定一个*新*名称以确保相同名称的文件不存在于目的地目录中。

要将 *SourceFile* 的副本放置到目录中，为 *TargetDirectory* 参数指定一个现有目录的路径。文件在其复制到目录时维护其各自的名称，除非您在路径的结束指定一个新文件名。**cp** 命令还将整个目录复制到其它目录，如果您指定 **-r** 或 **-R** 标志。

您还可以使用 **-R** 标志复制特殊设备文件。指定 **-R** 导致特殊文件在新路径名下被重新创建。指定 **-r** 标志导致 **cp** 命令尝试将特殊文件复制到常规文件中。

以下是如何使用 **cp** 命令的示例:

1. 要在当前目录中制作文件的副本，输入:

   ```
   cp prog.c prog.bak
   ```

   按下 Enter 键。

   这复制 **prog.c** 到 **prog.bak**。如果 **prog.bak** 文件还不存在，则 **cp** 命令创建它。如果它已存在，则 **cp** 命令用 **prog.c** 文件的副本替换它。
2. 要将当前目录中的文件复制到另一个目录，输入:

   ```
   cp jones /home/nick/clients
   ```

   按下 Enter 键。

   这复制 **jones** 文件到 **/home/nick/clients/jones**。
3. 要将目录中的所有文件复制到新目录，输入:

   ```
   cp /home/janet/clients/* /home/nick/customers
   ```

   按下 Enter 键。

   这只复制 **clients** 目录中的文件到 **customers** 目录。
4. 要将特定文件集复制到另一个目录，输入:

   ```
   cp jones lewis smith /home/nick/clients
   ```

   按下 Enter 键。

   这复制您当前工作目录中的 **jones**、 **lewis** 和 **smith** 文件到 **/home/nick/clients** 目录。
5. 要使用模式匹配字符复制文件，输入:

   ```
   cp programs/*.c .
   ```

   按下 Enter 键。

   这复制 **programs** 目录中以 **.c** 结束的文件到当前目录，当前目录由单个点（.）表明。您必须在 **c** 和最后的点之间输入一个空格。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cp** 命令，以获取精确的语法。

查找文件（find 命令）

可以使用 **find** 命令递归地搜索每个指定的*路径*的目录树，查找匹配用以下文本中给定的术语写的布尔表达式的文件。**find** 命令的输出取决于 *Expression* 参数指定的术语。

以下是如何使用 **find** 命令的示例:

1. 要列出文件系统中具有名称 **.profile** 的所有文件，请输入:

   ```
   find / -name .profile
   ```

   按下 Enter 键。

这搜索整个文件系统，并写名为 .profile 的所有文件的完整路径名。斜杠（/）告诉 **find** 命令搜索 /（**根**）目录及其所有子目录。

要节省时间，请通过指定您认为文件可能在的目录来限制搜索。

2. 要列出当前目录树中具有特定许可权代码 **0600** 的文件，请输入:

   ```
   find . -perm 0600
   ```

   按下 Enter 键。

   这列出具有*唯*一所有者读和所有者写许可权的文件的名称。点（.）告诉 **find** 命令搜索当前目录及其子目录。有关许可权代码的说明，请参阅 **chmod** 命令。

3. 要搜索具有某些许可权代码的文件的几个目录，请输入:

   ```
   find manual clients proposals -perm -0600
   ```

   按下 Enter 键。

   这列出具有所有者读和所有者写许可权和可能的其它许可权的文件的名称。搜索到 **manual**、**clients** 和 **proposals** 目录及其子目录。在前面的示例中，**-perm 0600** 仅选择具有完全匹配 **0600** 的许可权代码的文件。在此例中，**-perm -0600** 选择具有许可权代码的文件，这些代码允许 **0600** 表示的访问和 **0600** 级别以上的其它访问。这还匹配许可权代码 0622 和 2744。

4. 要列出当前目录中在当前 24 小时周期中已更改的所有文件，请输入:

   ```
   find . -ctime 1
   ```

   按下 Enter 键。

5. 要搜索具有多个链接的常规文件，请输入:

   ```
   find . -type f -links +1
   ```

   按下 Enter 键。

   这列出具有多个链接（-links +1）的普通文件（-type f）的名称。

   注: 每个目录有至少两个链接: 其父目录中的条目及其自己的 .（点）条目。有关多个文件链接的更多信息，请参阅 **ln** 命令。

6. 要搜索恰好 414 个字节长度的所有文件，请输入:

   ```
   find . -size 414c
   ```

   按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **find** 命令，以获取完整的语法。

显示文件类型（file 命令）

可以使用 **file** 命令读由 *File* 或 **-f** *FileList* 参数指定的文件，在每个文件上执行一系列测试。此命令尝试根据类型分类文件。然后命令将文件类型写入标准输出。

如果文件看起来是 ASCII，则 **file** 命令检查前 512 个字节并确定其语言。如果文件看起来不是 ASCII，则 **file** 命令进一步尝试确定它是二进制数据文件还是包含扩展字符的文本文件。

如果 *File* 参数指定可执行文件或对象模块文件并且版本号大于 0，则 **file** 命令显示版本商标。

file 命令使用 **/etc/magic** 文件标识具有幻数的文件，即，任何包含表示类型的数字或字符串常量的文件。

以下是如何使用 **file** 命令的示例:

1. 要显示名为 **myfile** 的文件包含的信息类型，请输入:

   ```
   file myfile
   ```

   按下 Enter 键。

   这显示 **myfile** 的文件类型（如目录、数据、ASCII 文本、C 程序源和归档）。
2. 要显示 **filenames.lst** 文件（包含文件名列表）中命名的每个文件的类型，请输入:

   ```
   file -f filenames.lst
   ```

   按下 Enter 键。

   这显示 **filenames.lst** 文件中命名的每个文件的类型。每个文件名必须在独立的行上显示。
3. 要创建 **filenames.lst** 文件，以便它包含当前目录中的所有文件名，请输入:

   ```
   ls > filenames.lst
   ```

   按下 Enter 键。

   按需要编辑 **filenames** 文件。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **file** 命令，以获取完整的语法。

显示文件内容（pg、more、page 和 cat 命令）

pg、**more** 和 **page** 命令允许您查看文件的内容，并控制文件显示的速度。还可以使用 **cat** 命令在您的屏幕上显示一个或多个文件的内容。将 **cat** 命令与 **pg** 命令组合在一起允许您一次一全屏地读文件的内容。

还可以通过使用输入和输出重定向来显示文件的内容。有关输入和输出重定向的更多详细信息，请参阅第 45 页的第 5 章，『输入和输出重定向』。

pg 命令

pg 命令从 *File* 参数读文件名，并将它们一次一屏地写到标准输出。如果指定连字符（-）作为 *File* 参数，或不带选项运行 **pg** 命令，则 **pg** 命令读标准输入。每个屏幕后跟一个提示符。如果您按下 Enter 键，则显示另一个屏幕。子命令与 **pg** 命令一起使用让您能复查已经显示过的内容。

例如，要一次一页地查看文件 `myfile` 的内容，请输入:

```
pg myfile
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **pg** 命令，以获取完整的语法。

more 或 page 命令

more 或 **page** 命令一次一屏地显示连续文本。它在每屏后暂停，并在屏幕底部打印*文件名*和完成的百分比（例如，`myfile (7%)`）。然后如果按下 Enter 键，则 **more** 命令显示另一行。如果按下空格键，则 **more** 命令显示另一屏文本。

注: 在一些终端型号上，**more** 命令在显示下一屏文本前清除屏幕，而不是滚动。

例如，要查看名为 myfile 的文件，请输入:

```
more myfile
```

按下 Enter 键。

按下空格键查看下一屏。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **more** 命令，以获取完整的语法。

cat 命令

cat 命令读序列中的每个 *File* 参数，并将它写到标准输出。

例如，要显示文件 notes 的内容，请输入:

```
cat notes
```

按下 Enter 键。如果文件长度大于 24 行，则文件的一部分滚动在屏幕外。要一次一页地列出文件，使用 **pg** 命令。

例如，要显示文件 notes、notes2 和 notes3 的内容，请输入:

```
cat notes notes2 notes3
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cat** 命令，以获取完整的语法。

查找文件中的文本字符串（grep 命令）

grep 命令搜索 *Pattern* 参数指定的模式，并写每一与标准输出匹配的行。

以下是如何使用 **grep** 命令的示例:

1. 要在名为 **pgm.s** 的文件中搜索包含一些模式匹配字符 *, ^, ?, [,], \(, \), \{ 和 \} 的模式，在此例中，以任何小写或大写字母开始的行，请输入:

   ```
   grep "^[a-zA-Z]" pgm.s
   ```

   按下 Enter 键。

   这显示 **pgm.s** 文件中所有以字母开始的行。
2. 要显示名为 **sort.c** 的文件中不匹配某一模式的所有行，请输入:

   ```
   grep -v bubble sort.c
   ```

   按下 Enter 键。

   这显示 **sort.c** 文件中不包含字 **bubble** 的所有行。
3. 要显示匹配字符串 **staff** 的 **ls** 命令的输出中的行，请输入:

   ```
   ls -l | grep staff
   ```

   按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **grep** 命令，以获取完整的语法。

排序文本文件（sort 命令）

可以使用 **sort** 命令对由 *File* 参数指定的文件中的行依字母顺序排列或排序，并将结果写入标准输出。如果 *File* 参数指定多个文件，则 **sort** 命令连接文件，并作为一个文件以字母顺序排列它们。

注: **sort** 命令区分大小写并将大写字母排在小写字母之前（这取决于语言环境）。

在下例中，名为 **names** 的文件的内容是:

```
marta
denise
joyce
endrica
melanie
```

名为 **states** 的文件的内容是:

```
texas
colorado
ohio
```

1. 要显示名为 **names** 的文件的已排序内容，请输入:

   ```
   sort names
   ```

   按下 Enter 键。

   系统显示类似于以下的信息:

   ```
   denise
   endrica
   joyce
   marta
   melanie
   ```

2. 要显示 **names** 和 **states** 文件的已排序内容，请输入:

   ```
   sort names states
   ```

   按下 Enter 键。

   系统显示类似于以下的信息:

   ```
   colorado
   denise
   endrica
   joyce
   marta
   melanie
   ohio
   texas
   ```

3. 要用名为 **names** 的文件的已排序内容替换其原始内容，请输入:

   ```
   sort -o names names
   ```

   按下 Enter 键。

   这将用已排序的相同数据替换 **names** 文件的内容。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **sort** 命令，以获取完整的语法。

比较文件（diff 命令）

可以使用 **diff** 命令比较文本文件。它能比较单个文件或目录的内容。

当 **diff** 命令对常规文件运行时，以及当它比较不同目录中的文本文件时，**diff** 命令说明必须更改文件中的哪些行以使它们匹配。

以下是如何使用 **diff** 命令的示例:

1. 要比较两个文件，请输入:

   ```
   diff chap1.bak chap1
   ```

   按 Enter 键。

   这显示 **chap1.bak** 和 **chap1** 文件之间的差异。
2. 要比较两个文件，同时忽略空白数量的差异，请输入:

   ```
   diff -w prog.c.bak prog.c
   ```

   按 Enter 键。如果两个文件只在字之间的空格数和制表符数有差异，则 **diff -w** 命令认为这两个文件相同。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **diff** 命令，以获取完整的语法。

计数文件中的字、行和字节（wc 命令）

缺省情况下，**wc** 命令计数由 *File* 参数指定的文件中的行数、字数和字节数。如果没有为 *File* 参数指定文件，则使用标准输入。命令将结果写到标准输出，并保留所有命名文件的总计计数。如果指定了标志，则标志的顺序确定输出的顺序。字定义为由空格、制表符或换行字符定界的字符串。

当文件在命令行上指定时，它们的名称和计数一起打印。

例如，要显示名为 `chap1` 的文件的行、字和字节计数，请输入:

```
wc chap1
```

按下 Enter 键。它显示 `chap1` 文件中的行数、字数和字节数。

例如，要仅显示字节和字计数，请输入:

```
wc -cw chap*
```

按下 Enter 键。这显示每个名称以 `chap` 开始的文件的字节数和字数，并显示总计数。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **wc** 命令，以获取完整的语法。

显示文件的头几行（head 命令）

head 命令将每个指定文件或标准输入的头几行写到标准输出。如果 **head** 命令未指定标志，则缺省情况下显示头 10 行。

例如，要显示 `Test` 文件的头五行，请输入:

```
head -5 Test
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **head** 命令，以获取完整的语法。

显示文件的最后几行（tail 命令）

tail 命令从指定点开始，将由 *File* 参数指定的文件写到标准输出。

例如，要显示 notes 文件的最后 10 行，请输入:

```
tail notes
```

按下 Enter 键。

例如，要指定从 notes 文件的末尾开始读的行数，请输入:

```
tail -20 notes
```

按下 Enter 键。

例如，要从第 200 个字节开始一次一页地显示 notes 文件，请输入:

```
tail -c +200 notes | pg
```

按下 Enter 键。

例如，要跟随名为 accounts 的文件的增长，请输入:

```
tail -f accounts
```

按下 Enter 键。这显示 accounts 文件的最后 10 行。当有行添加到 accounts 文件时，**tail** 命令继续显示这些行。显示一直继续，直到您按下（Ctrl-C）按键序列停止显示。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **tail** 命令，以获取完整的语法。

剪切文本文件的部分（cut 命令）

要将来自文件每一行的选定字节、字符或字段写到标准输出，请使用 **cut** 命令。

例如，要显示文件每一行的几个字段，请输入:

```
cut -f1,5 -d: /etc/passwd
```

按下 Enter 键。这显示系统密码文件的登录名和全用户名字段。这些是使用冒号（-d:）分隔的第一个和第五个字段（-f1,5）。

例如，如果 **/etc/passwd** 文件类似于:

```
su:*:0:0:User with special privileges:/:/usr/bin/sh
daemon:*:1:1::/etc:
bin:*:2:2::/usr/bin:
sys:*:3:3::/usr/src:
adm:*:4:4:System Administrator:/var/adm:/usr/bin/sh
pierre:*:200:200:Pierre Harper:/home/pierre:/usr/bin/sh
joan:*:202:200:Joan Brown:/home/joan:/usr/bin/sh
```

cut 命令产生:

```
su:User with special privileges
daemon:
bin:
```

```
sys:
adm:System Administrator
pierre:Pierre Harper
joan:Joan Brown
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cut** 命令，以获取完整的语法。

粘贴文本文件的部分（paste 命令）

paste 命令将最多 12 个文件的行合并到一个文件中。

例如，如果有包含以下文本的名为 names 的文件:

```
rachel
jerry
mark
linda
scott
```

另一个包含以下文本的名为 places 的文件:

```
New York
Austin
Chicago
Boca Raton
Seattle
```

另一个包含以下文本的名为 dates 的文件:

```
February 5
March 13
June 21
July 16
November 4
```

要将文件 names、places 和 dates 的文本粘贴在一起，请输入:

```
paste names places dates > npd
```

按下 Enter 键。这创建一个名为 npd 的文件，将来自 names 文件的数据包含在一个列中，将 places 文件的数据包含在另一个列中，并且将 dates 文件的数据包含在第三个列中。现在 npd 文件包含以下内容:

```
rachel          New York        February 5
jerry           Austin          March 13
mark            Chicago         June 21
linda           Boca Raton      July 16
scott           Seattle         November 4
```

制表符分隔每行上的名称、位置和日期。这些列不对齐，因为在每一第八个列位置处设置了制表符停止位。

例如，要使用字符而非制表符来分隔列，请输入:

```
paste -d"!@" names places dates > npd
```

按下 Enter 键。这交替使用 ! 和 @ 作为列分隔符。如果 names、places 和 dates 文件与示例 1 中的相同，则 npd 文件包含以下内容:

```
rachel!New York@February 5
jerry!Austin@March 13
mark!Chicago@June 21
linda!Boca Raton@July 16
scott!Seattle@November 4
```

例如，要在四列中列出当前目录，请输入:

```
ls | paste - - - -
```

按下 Enter 键。每个连字符（-）告诉 **paste** 命令创建包含从标准输入读取的数据的列。第一行放在第一列中，第二行放在第二列中，等等。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **paste** 命令，以获取精确的语法。

对文本文件中的行编号（nl 命令）

nl 命令读指定的文件（缺省情况下是标准输入），为输入中的行编号，并将编号的行写到标准输出。

例如，要仅编号非空白行，请输入:

```
nl chap1
```

按下 Enter 键。这显示 chap1 的编号的列表，仅对主体部分中的非空白行编号。

例如，要对所有行编号，请输入:

```
nl -ba chap1
```

按下 Enter 键。这对名为 chap1 的文件中的所有行，包括空白行进行编号。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **nl** 命令，以获取完整的语法。

除去文本文件中的列（colrm 命令）

colrm 命令从文件中除去指定的列。输入取自标准输入。输出发送到标准输出。

如果命令调用时使用了参数，则除去每行中从指定列到最后一列的各列。如果命令在调用时使用了两个参数，则除去从第一个指定列到第二个指定列的各列。

注: 列编号从列 1 开始。

例如，要从 text.fil 文件中除去列，请输入:

```
colrm 6 < text.fil
```

按下 Enter 键。

如果 text.fil 包含:

```
123456789
```

则 **colrm** 命令显示:

```
12345
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **colrm** 命令，以获取完整的语法。

链接文件和目录

*链接*是文件名与索引节点引用号（i-node 号）之间的连接，是文件的内部表示。因为目录条目包含文件名和 i-node 号对，所以每个目录条目都是一个链接。i-node 号实际标识文件，而非文件名。通过使用链接，任何 i-node 号或文件都可以通过很多不同的名称标识。

例如，i-node 号 798 包含有关 Omaha 办公室 6 月销售的备忘录。目前，此备忘录的目录条目如下:

i-node 号	**文件名**
798	**memo**

因为此信息与存储在 sales 和 omaha 目录中的信息相关，所以在需要的地方使用链接来共享信息。使用 **ln** 命令，创建到这些目录的链接。现在文件有如下三个文件名:

i-node 号	**文件名**
798	**memo**
798	**sales/june**
798	**omaha/junesales**

当您使用 **pg** 或 **cat** 命令来查看三个文件名中任何一个的内容时，将显示相同的信息。如果您从三个文件名中任何一个编辑 i-node 号的内容，则所有文件名所显示的数据的内容将反映任何更改。

链接的类型

链接是使用 **ln** 命令创建的，并有以下类型:

硬链接 允许从新文件名访问文件的数据。硬链接确保文件的存在。当除去最后一个硬链接时，删除 i-node 号及其数据。硬链接只能在同一文件系统中的文件之间创建。

符号链接 允许从新文件名访问其它文件系统中的数据。符号链接是一种包含路径名的特殊类型的文件。当进程遇到符号链接时，该进程可能搜索该路径。符号链接不保护文件不被从文件系统中删除。

> **注**: 无论创建了多少链接，创建文件的用户保留该文件的所有权。只有文件的所有者或 root 用户可以设置该文件的访问方式。但是，可以从具有适当访问方式的链接文件对文件进行更改。

只要有一个到该文件的 i-node 号的硬链接，该文件或目录就存在。在由 **ls -l** 命令显示的长列表中，给定到每个文件和子目录的硬链接的数目。无论哪一个链接首先创建，操作系统同等对待所有的硬链接。

链接文件（ln 命令）

使用 **ln** 命令链接文件，是在多个位置使用相同数据的便利方式。通过给原文件备用名称来创建链接。链接的使用允许大文件（如数据库或邮件列表）由几个用户共享，而不制作该文件的副本。链接不但节省磁盘空间，而且对一个文件的更改自动地反映在所有链接的文件中。

ln 命令将 *SourceFile* 参数中指定的文件与由 *TargetFile* 参数指定的文件，或由 *TargetDirectory* 参数指定的另一目录中的相同文件名链接。缺省情况下，**ln** 命令创建硬链接。要使用 **ln** 命令创建符号链接，指定 **-s** 标志。

如果您将文件链接到新文件，则只能列出一个文件。如果链接到目录，则可列出多个文件。

TargetFile 参数是可选的。如果您不指定目标文件，则 **ln** 命令在您的当前目录中创建文件。新文件继承 *SourceFile* 参数中指定的文件的名称。

> **注**: 您无法不使用 **-s** 标志就在文件系统间链接文件。

例如，要创建到名为 chap1 的文件的另一个链接，请输入:

```
ln -f chap1 intro
```

按下 Enter 键。这将 chap1 链接到新名称 intro。当使用了 **-f** 标志时，将创建文件名 intro（如果它还不存在）。如果 intro 存在，则该文件由到 chap1 的链接替换。然后 chap1 和 intro 文件名都将指同一文件。任何对其中一个文件的更改也将出现在另一个文件中。

例如，要将名为 index 的文件链接到名为 manual 的另一个目录中的相同名称，请输入:

```
ln index manual
```

按下 Enter 键。这将 index 链接到新名称 manual/index。

例如，要将几个文件链接到另一个目录中的名称，请输入:

```
ln chap2 jim/chap3 /home/manual
```

按下 Enter 键。这将 chap2 链接到新名称 /home/manual/chap2，将 jim/chap3 链接到 /home/manual/chap3。

例如，要将 **ln** 命令与模式匹配字符一起使用，请输入:

```
ln manual/* .
```

注: 必须在星号和句号之间输入一个空格。

按下 Enter 键。这将 manual 目录中的所有文件链接到当前目录（点（.））中，并给它们与在 manual 目录中相同的名称。

例如，要创建符号链接，请输入:

```
ln -s /tmp/toc toc
```

按下 Enter 键。这在当前目录中创建符号链接 toc。toc 文件指向 /tmp/toc 文件。如果 /tmp/toc 文件存在，则 **cat** toc 命令列出它的内容。

要得到同样的结果但不指定 *TargetFile* 参数，请输入:

```
ln -s /tmp/toc
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **ln** 命令，以获取完整的语法。

除去链接的文件

rm 命令从您指示的文件名除去链接。当删除几个硬链接的文件名中的一个时，由于它会以其它名称保留，因此不完全删除文件。当除去 i-node 的最后一个链接时，数据也被除去。然后该 i-node 号可由系统重新使用。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **rm** 命令，以获取完整的语法。

DOS 文件

AIX 操作系统允许您在系统上使用 DOS 文件。将要使用的 DOS 文件复制到软盘。您的系统可以用正确的格式将这些文件读入基本操作系统目录中，并以 DOS 格式回读到软盘上。

注: 通配符 * 和 ?（星号和问号）不能与本节中讨论的命令一起正确使用（尽管它们可与基本操作系统 shell 一起正确使用）。如果您未指定文件扩展名，则文件名的匹配就象您指定了一个空白扩展名一样。

将 DOS 文件复制到基本操作系统

dosread 命令将指定的 DOS 文件复制到指定的基本操作系统文件。

> **注**: DOS 文件命名约定的使用有一个例外。因为反斜杠 (\) 字符可对基本操作系统有特殊的含义，所以使用斜杠（/）字符作为定界符来指定 DOS 路径名中的子目录名称。

例如，要将一个名为 chap1.doc 的文本文件从 DOS 软盘复制到基本操作文件系统，请输入:

```
dosread -a chap1.doc chap1
```

按下 Enter 键。这将 **/dev/fd0** 缺省设备上的 DOS 文本文件 **\CHAP1.DOC** 复制到当前目录中的基本操作系统文件 **chap1** 中。

例如，要将二进制文件从 DOS 软盘复制到基本操作文件系统，请输入:

```
dosread -D/dev/fd0 /survey/test.dta /home/fran/testdata
```

按下 Enter 键。这将 **/dev/fd1** 上的 **\SURVEY\TEST.DTA** DOS 数据文件复制到基本操作系统文件 /home/fran/testdata。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **dosread** 命令，以获取完整的语法。

将基本操作系统文件复制到 DOS 文件

doswrite 命令将指定的基本操作系统文件复制到指定的 DOS 文件。

> **注**: DOS 文件命名约定的使用有一个例外。因为反斜杠 (\) 字符可对基本操作系统有特殊的含义，所以使用斜杠（/）字符作为定界符来指定 DOS 路径名中的子目录名称。

例如，要将一个名为 chap1 的文本文件从基本操作文件系统复制到 DOS 软盘，请输入:

```
doswrite -a chap1 chap1.doc
```

按下 Enter 键。这将当前目录中的基本操作系统文件 chap1 复制到 **/dev/fd0** 上的 DOS 文本文件 \CHAP1.DOC。

例如，要将一个名为 /survey/test.dta 的二进制文件从基本操作文件系统复制到 DOS 软盘，请输入:

```
doswrite -D/dev/fd0 /home/fran/testdata /survey/test.dta
```

按下 Enter 键。它将基本操作系统数据文件 /home/fran/testdata 复制到 **/dev/fd1** 上的 DOS 文件 \SURVEY\TEST.DTA。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **doswrite** 命令，以获取完整的语法。

删除 DOS 文件

dosdel 命令删除指定的 DOS 文件。

> **注**: DOS 文件命名约定的使用有一个例外。因为反斜杠 (\) 字符可对基本操作系统有特殊的含义，所以使用斜杠（/）字符作为定界符来指定 DOS 路径名中的子目录名称。

dosdel 命令在检查磁盘前将文件名或目录名中的小写字符转换成大写字符。因为假定所有文件名是全（非相对）路径名，所以您不需要添加初始的斜杠（/）。

例如，要删除缺省设备（**/dev/fd0**）上名为 file.ext 的 DOS 文件，请输入:

```
dosdel file.ext
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **dosdel** 命令，以获取完整的语法。

列出 DOS 目录的内容

dosdir 命令显示关于指定的 DOS 文件或目录的信息。

> **注**: DOS 文件命名约定的使用有一个例外。因为反斜杠 (\) 字符可对基本操作系统有特殊的含义，所以使用斜杠（/）字符作为定界符来指定 DOS 路径名中的子目录名称。

dosdir 命令在检查磁盘前将文件名或目录名中的小写字符转换成大写字符。因为假定所有文件名是全（非相对）路径名，所以您不需要添加初始的 / （斜杠）。

例如，要读 **/dev/fd0** 上 DOS 文件的目录，请输入:

```
dosdir
```

按下 Enter 键。命令返回文件的名称和磁盘空间信息，类似于以下内容。

```
PG3-25.TXT
PG4-25.TXT
PG5-25.TXT
PG6-25.TXT
Free space: 312320 bytes
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **dosdir** 命令，以获取完整的语法。

文件的命令摘要

*****	通配符，匹配任何字符。
?	通配符，匹配任何单个字符。
[ ]	元字符，匹配圈起的字符。

文件处理过程

cat	连接或显示文件
cmp	比较两个文件
colrm	从文件抽取列
cp	复制文件
cut	从文件的每一行写出选定字节、字符或字段
diff	比较文本文件
file	确定文件类型
find	查找包含匹配表达式的文件
grep	搜索文件以查找模式
head	显示一个文件或多个文件的头几行或字节
more	在显示屏幕上一次一屏地显示连续文本
mv	移动文件
nl	为文件中的行编号
pg	格式化文件到显示器
rm	除去（取消链接）文件或目录
paste	将几个文件的行或后继行合并到一个文件

page	在显示屏幕上一次一屏地显示连续文本
sort	对文件排序，合并已排序的文件，以及检查文件以确定文件是否已排序
tail	从指定点开始，将文件写到标准输出
wc	计数文件中的行数、字数和字节数

链接文件和目录

ln	链接文件和目录

DOS 文件

dosdel	删除 DOS 文件
dosdir	列出 DOS 文件的目录
dosread	将 DOS 文件复制到基本操作系统文件
doswrite	将基本操作系统文件复制到 DOS 文件

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 45 页的第 5 章，『输入和输出重定向』

第 135 页的第 12 章，『shell』

第 53 页的『文件系统』

第 56 页的『目录概述』

第 65 页的第 7 章，『文件』

第 79 页的『链接文件和目录』

第 85 页的第 8 章，『打印机、打印作业和队列』

第 99 页的第 9 章，『备份文件和存储介质』

第 113 页的第 10 章，『文件和系统安全性』

第 8 章 打印机、打印作业和队列

取决于打印机，您可以控制最终输出的外观和特征。打印机不需要与系统部件和系统控制台位于同一区域中。打印机可以直接连接到本地系统，或打印作业可在网络上发送到远程系统。

要最大效率地处理打印作业，系统将每个作业放置到队列中，以等待打印机可用。系统可以保存队列中一个或多个文件的输出。当打印机产生文件的输出时，系统处理队列中的下一个作业。此进程继续，直到打印了队列中的每个作业。

有关打印机、打印作业和队列的详细信息，请参阅 *AIX 5L Version 5.2 Guide to Printers and Printing*。

本章讨论以下各章:

打印机术语

以下描述常用于打印的术语。

本地打印机

当打印机连接到一个节点或主机时，该打印机称为*本地打印机*。

打印作业

*打印作业*是在打印机上运行的工作单元。一个打印作业可以由打印一个或多个文件组成，取决于如何请求打印作业。系统指定一个唯一的作业号给每个它运行的作业。

打印假脱机程序

用于打印的*假脱机程序*不是特定的打印作业假脱机程序。而是提供一般的假脱机功能，可用于排队各种作业类型，包括排队到打印机的打印作业。

假脱机程序通常不知道排队作业的类型。当系统管理员定义假脱机程序队列时，队列的目的由为队列指定的假脱机程序后端程序定义。例如，如果假脱机程序后端程序是 **piobe** 命令（打印机 I/O 后端），

则队列是打印队列。同样地，如果假脱机程序后端程序是编译器，则队列用于编译作业。当假脱机程序的 **qdaemon** 命令从假脱机程序队列中选择作业时，它通过调用后端程序（由系统管理员在队列定义时指定）来运行该作业。

主假脱机程序命令是 **enq** 命令。尽管您可以直接调用此命令来排队打印作业，但定义了以下前端命令用于提交打印作业：**lp**、**lpr** 和 **qprt** 命令。由这些命令之一发出的打印请求首先传递到 **enq** 程序，然后将有关文件的信息放入 **qdaemon** 的队列以进行处理。

打印机后端

*打印机后端*是程序的集合，由假脱机程序的 **qdaemon** 命令调用，用来管理排队打印的打印作业。打印机后端执行以下功能：

- 从 **qdaemon** 命令接收要打印的一个或多个文件的列表
- 使用打印机并格式化来自数据库的属性值，由在命令行上输入的标志覆盖
- 在打印文件之前初始化打印机
- 需要时运行过滤器，以将打印数据流转换成打印机支持的格式
- 提供简单格式 ASCII 文档的过滤器
- 提供打印本地语言字符的支持
- 将已过滤的打印数据流传递给打印机设备驱动程序
- 生成报头和尾部页
- 生成多个副本
- 报告缺纸，需要干预，及打印机错误条件
- 报告过滤器检测到的问题
- 取消打印作业后清理
- 提供打印环境，系统管理员可对它进行定制以满足特定打印需要

qdaemon

qdaemon 是在后台运行并控制队列的进程。它一般在系统打开时启动。

队列

*队列*是您将打印作业定向的地方。它是 **/etc/qconfig** 文件中的一节，其名称是队列的名称，并指向关联的队列设备。以下是一个样本列表：

```
Msa1:
   device = lp0
```

在上例中，`Msa1` 是队列名，`lp0` 是设备名。

队列设备

*队列设备*是 **/etc/qconfig** 文件中通常跟在本地队列节之后的一节。它指定应打印到的 **/dev** 文件（打印机设备） 和应使用的后端。以下是一个样本列表：

```
lp0:
   file = /dev/lp0
   header = never
   trailer = never
   access = both
   backend = /usr/lpd/piobe
```

在以上输出中，`lp0` 是设备名，其余行定义设备是如何使用的。

注: 可以有多个队列设备与单个队列关联。

实际打印机

*实际打印机*是在唯一硬件设备地址连接到串行端口或并行端口的打印机硬件。内核中的打印机设备驱动程序与打印机硬件通信，并提供打印机硬件与虚拟打印机之间的接口，但它不知道虚拟打印机的概念。

远程打印机

*远程打印系统*允许没有未连接到打印机的节点可以访问打印机。要使用远程打印设施，单个节点必须使用传输控制协议／网际协议（TCP／IP）连接到网络，并且必须支持必需的 TCP/IP 应用程序。

虚拟打印机

*虚拟打印机*是定义实际打印机的特定软件视图的一组属性。虚拟打印机的此视图仅涉及打印机理解的高级数据流（如 ASCII 或 PostScript）。它不包含有关打印机硬件如何连接到主机或用于将传送数据字节传送到打印机或从打印机传送数据字节的协议的任何信息。虚拟打印机由系统管理员定义。

启动打印作业（qprt 命令）

要请求打印作业，请使用 **qprt** 命令或 **smit** 命令。有关更多信息，请参阅『使用 qprt 命令』和第 89 页的『使用 smit 命令』。使用这些命令时，请指定以下内容:

- 要打印的文件的名称
- 打印队列名称
- 输出 bin 的名称
- 要打印副本的数量
- 是否在远程主机上制作文件的副本
- 打印后是否擦除文件
- 是否发送作业状态的通知
- 是否用系统邮件发送作业状态的通知
- 脉冲串状态
- “发送至”标号的用户名
- 远程打印的控制台确认消息
- 远程打印的文件确认消息
- 优先级级别

先决条件

在开始打印作业之前，请确保以下内容:

- 对于本地打印作业，打印机必须物理地连接到您的系统上。
- 对于远程打印作业，必须配置您的系统以与远程打印服务器通信。

使用 qprt 命令

qprt 命令创建和对打印作业排队以打印指定的文件。如果指定多个文件，则所有文件一起组成一个打印作业。这些文件以在命令行上指定的顺序打印。

在打印文件前，必须有对它的读访问权。要在打印后除去文件，您必须有对包含文件的目录的写访问权。

qprt 命令的最常用的标志如下:

-b *Number*
指定下边距。下边距是在每页的底部保留的空白行的数目。

-B *Value*
指定是否应该打印脉冲串页（在穿孔处分隔的连续纸页）。*Value* 变量由两个字符的字符串组成。第一个字符应用到标题页。第二个字符应用到尾页。两个字符的每一个都可能是以下的一种:

a 总是为每个打印作业中的每个文件打印（标题或尾）页。

n 从不打印（标题或尾）页。

g 为每个打印作业（文件组）打印（标题或尾）页一次。

例如，**-B ga** 标志指定在每个打印作业的开始打印标题页，在每个打印作业的每个文件后打印尾页。

注: 在远程打印环境中，缺省值是由服务器上的远程队列确定的。

-e *Option*
指定是否要强调打印。

+ 表示要强调打印。

! 表示不要强调打印。

-E *Option*
指定是否要两倍高度打印。

+ 表示要两倍高度打印。

! 表示不要两倍高度打印。

-f *FilterType*
指定过滤器的一个字符标识，通过该过滤器，您的打印文件（一个或多个）在被发送到打印机之前传递。可用的过滤器标识是 **p**（它调用 **pr** 过滤器）和 **n**（它处理来自 **troff** 命令的输出）。

-i *Number*
导致每行缩排指定的空格数。*Number* 变量必须包含在 **-w** 标志指定的页宽度中。

-K *Option*
指定是否要压缩印刷。

+ 表示要压缩印刷。

! 表示不要压缩印刷。

-l *Number*
将页面长度设置为指定的行数。如果 *Number* 变量是 0，则忽略页面长度，并认为输出是一个连续的页面。页面长度包含上边距和下边距，并表示页面的可打印长度。

-L *Option*
指定宽于页宽度的行是否应该被环绕到下一行或在右边距被截断。

+ 表示长行应该环绕到下一行。

! 表示长行不应该环绕，而是应该在右边距处被截断。

-N *Number*
指定要打印的副本数。如果不指定该标志，则打印一个副本。

-p *Number*
将间距设置为每英寸 *Number* 个字符。*Number* 的典型值是 10 和 12。打印的字符的实际间距还受 **-K**（压缩）标志和 **-W**（两倍宽度）标志影响。

-P *Queue*[:*QueueDevice*]
指定打印队列名称和可选的队列设备名称。如果不指定该标志，则假定为缺省打印机。

-Q *Value*
指定打印作业的纸张大小。纸张大小的 *Value* 是打印机相关的。典型值是: 1 用于 letter 大小纸张，2 用于 legal 大小纸张，等等。查询您的打印机手册以获取指定给特定纸张大小的值。

-t *Number*
指定上边距。上边距是在每页的顶部保留的空白行的数目。

-w *Number*
将页宽度设置为 *Number* 变量指定的字符数。页宽度必须包含用 **-i** 标志指定的缩排空格数。

-W *Option*
指定是否要两倍宽度打印。

+ 表示要两倍宽度打印。

! 表示不要两倍宽度打印。

-z *Value*　将页式打印机输出旋转由 *Value* 变量指定的 1/4 转顺时针方向的数目。长度（**-l**）和宽度（**-w**）值自动相应调整。

0　纵向

1　横向右定向

2　纵向颠倒

3　横向左定向

-= *OutputBin*　指定打印作业的输出 bin 目的地。可能的值在下面列出。但是，有效的输出 bin 是打印机相关的。

0　顶部打印机 bin

1-49　高容量输出（HCO）bin 1 - 49

49　特定于打印机的输出 bin

-# *Value*　指定特殊函数。

j　显示指定打印作业的作业号。

h　排队打印作业，但是将它放入 **HELD** 状态，直到再次释放。

v　验证指定的打印机后端标志值。该验证在提交打印作业时检查非法标志值时有用。如果未指定验证，则一个不正确的标志值将在以后当实际处理作业时停止打印作业。

例如，要请求 **myfile** 文件在第一个可用的打印机（该打印机为使用缺省值的缺省打印队列配置）上打印，请输入:

```
qprt myfile
```

例如，要请求文件 **somefile** 在使用特定标志值的特定队列上打印，并在打印作业提交时验证标志值，请输入:

```
qprt -f p -e + -Pfastest -# v somefile
```

这将 **somefile** 文件通过 **pr** 过滤器命令（**-f p** 标志）传递，并使用强调的方式（**-e +** 标志）在为名为 **fastest**（**-Pfastest** 标志）的队列配置第一个可用的打印机上打印它。

例如，要在 legal 大小的纸上打印 **myfile**，请输入:

```
qprt -Q2 myfile
```

例如，要在打印队列 **Msp1** 处打印每个 **new.index.c**、**print.index.c** 和 **more.c** 文件的三个副本，请输入:

```
qprt -PMsp1 -N 3 new.index.c print.index.c more.c
```

例如，要打印连接文件 new.index.c、print.index.c 和 more.c 的三个副本，请输入:

```
cat new.index.c print.index.c more.c | qprt -PMsp1 -N 3
```

> **注**: AIX 操作系统还支持 BSD UNIX 打印命令（**lpr**）和 System V UNIX 打印命令（**lp**）。请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lpr** 和 **lp** 命令，以获取完整的语法。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **qprt** 命令，以获取完整的语法。

使用 smit 命令

您还可以使用 **smit** 发出 **qprt** 命令。在提示符下输入:

```
smit qprt
```

按下 Enter 键。

取消打印作业（qcan 命令）

可以使用 **qcan** 命令或 **smit** 命令取消打印队列中的任何作业。取消打印作业时，将提示您提供作业驻留的打印队列的名称和将取消的作业号。

此过程适用于本地和远程打印作业。

先决条件

- 对于本地打印作业，打印机必须物理地连接到您的系统上。
- 对于远程打印作业，必须配置您的系统以与远程打印服务器通信。

使用 qcan 命令

qcan 命令取消本地或远程打印队列中的特殊作业号，或本地打印队列中的所有作业。要确定作业号，请输入 **qchk** 命令。

qcan 命令的常见格式如下:

```
qcan -PQueueName -x JobNumber
```

例如，要取消在作业所在的任何打印机上的作业号 123，请输入:

```
qcan -x 123
```

例如，要取消打印机 lp0 上排队的所有作业，请输入:

```
qcan -X -Plp0
```

> **注**: AIX 操作系统还支持 BSD UNIX 取消打印命令（**lprm**）和 System V UNIX 取消打印命令（**cancel**）。请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lprm** 和 **cancel** 命令，以获取更多信息和精确的语法。

使用 smit 命令

要使用 SMIT 取消打印作业，请输入:

```
smit qcan
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **qcan** 命令，以获取完整的语法。

检查打印作业状态（qchk 命令）

要显示指定的作业号、队列、打印机或用户的当前状态信息，可以使用**基于 Web 的系统管理器快速路径、qchk** 命令或 **smit** 命令。

先决条件

- 对于本地打印作业，打印机必须物理地连接到您的系统上。
- 对于远程打印作业，必须配置您的系统以与远程打印服务器通信。

“基于 Web 的系统管理器”快速路径

要使用“基于 Web 的系统管理器”快速路径来检查打印作业的状态，请输入:

```
wsm printers
```

在“打印机队列”容器中，选择打印作业，然后使用菜单来检查它的状态。

使用 qchk 命令

可以使用 **qchk** 命令显示有关指定打印作业、打印队列或用户的当前状态信息。

qchk 命令的常见格式是:

```
qchk -P QueueName -# JobNumber -u OwnerName
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **qchk** 命令，以获取完整的语法。

以下是如何使用 **qchk** 命令的示例:

1. 要显示缺省打印队列，请输入:

   ```
   qchk -q
   ```

   按下 Enter 键。
2. 要显示所有队列的长状态，直到排队的作业完成，同时每 5 秒更新屏幕，请输入:

   ```
   qchk -A -L -w 5
   ```

   要返回命令提示符，请输入 **^C**。
3. 要显示打印队列 lp0 的状态，请输入:

   ```
   qchk -P lp0
   ```

   按下 Enter 键。
4. 要显示作业号 123 的状态，请输入:

   ```
   qchk -# 123
   ```

   按下 Enter 键。
5. 要检查所有队列中所有作业的状态，请输入:

   ```
   qchk -A
   ```

注: AIX 操作系统还支持 BSD UNIX 检查打印队列命令（**lpq**）和 System V UNIX 检查打印队列命令（**lpstat**）。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lpq** 和 **lpstat** 命令，以获取完整的语法。

使用 smit 命令

要使用 SMIT 检查打印作业的状态，请输入:

```
smit qchk
```

打印机状态条件

打印队列可能具有的一些状态条件如下:

DEV_BUSY 表示:

- 对打印机设备（lp0）定义了多个队列，且另一个队列当前正在使用打印机设备。
- **qdaemon** 尝试使用打印机端口设备（lp0），但是另一个应用程序当前正在使用该打印机设备。

要从 **DEV_BUSY** 恢复，请等候到队列或应用程序已释放打印机设备，或取消正在使用打印机端口的作业或进程。

DEV_WAIT 表示队列正在打印机上等待（因为打印机脱机、缺纸、卡纸或电缆松动、坏了或导线连接不正确）。

要从 **DEV_WAIT** 恢复，请更改导致其等待的问题。有时，在可以更改问题之前，必须先从队列中除去作业。

DOWN 队列通常在 **DEV_WAIT** 状态后进入 **DOWN** 状态。当由于缺少正确地信号通知，而使打印机设备驱动程序无法告知打印机是否在那里时，该情况发生。但是，一些打印机可能没有能力用信号通知排队系统它已脱机，而是用信号通知它已关闭。如果打印机设备发信号或看上去关闭，则队列将进入 **DOWN** 状态。

要从 **DOWN** 状态恢复，请更正使队列关闭的问题，并让系统管理员恢复队列运行。*必须*在再次使用队列前手工启动它。

HELD 指定打印作业挂起。打印作业将不由假脱机程序处理，直到释放它。

QUEUED 指定打印文件被排队，并排队等待打印。

READY 指定参与队列的所有东西都准备排队和打印作业。

RUNNING 指定打印文件正在打印。

按优先顺序排列打印作业（qpri 命令）

要更改打印作业的优先级，请使用 **qpri** 命令或 **smit** 命令。您只能在本地队列上指定作业优先级。较高的值表示打印作业有较高的优先级。缺省优先级是 15。大多数用户打印作业的最大优先级是 20。但是，来自具有 root 用户权限的用户或来自 printq 组（组 0）的成员的打印作业能接收优先级 30。

注: 您无法将优先级指定给远程打印作业。

先决条件

- 对于本地打印作业，打印机必须物理地连接到您的系统上。
- 对于远程打印作业，必须配置您的系统以与远程打印服务器通信。

使用 qpri 命令（qpri 命令）

qpri 命令重新指定您提交的打印作业的优先级。如果具有 root 用户权限或属于 printq 组，则可以将优先级指定给在打印队列中的任何作业。

qpri 命令的基本格式是:

```
qpri -# JobNumber -a PriorityLevel
```

例如，要将作业号 123 更改为优先级号 18，请输入:

```
qpri -# 123 -a 18
```

例如，要在提交本地打印作业时按优先顺序排列它，请输入:

```
qprt -PQueueName -R PriorityLevel FileName
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **qpri** 命令，以获取完整的语法。

使用 smit 命令

要使用 SMIT 更改打印作业的优先级，请输入:

```
smit qpri
```

挂起和释放打印作业（qhld 命令）

在将打印作业发送给打印队列后，可以通过使用**基于 Web 的系统管理器快速路径**、**qhld** 命令或 **smit** 命令将打印作业挂起。『**基于 Web 的系统管理器快速路径**』，『**使用 qhld 命令**』或**第 94 页的『使用 smit 命令』**。 可以使用相同的命令以在以后释放用于打印的打印作业。

先决条件

- 对于本地打印作业，打印机必须物理地连接到您的系统上。
- 对于远程打印作业，必须配置您的系统以与远程打印服务器通信。

基于 Web 的系统管理器快速路径

要使用“基于 Web 的系统管理器”快速路径来挂起或释放打印作业，请输入:

```
wsm printers
```

在“打印机队列”容器中，选择打印作业，然后使用菜单来将它挂起或释放它以进行打印。

使用 qhld 命令

qhld 命令在您发送了打印作业后将它挂起。可以将特殊打印作业挂起，或将指定打印队列上的所有打印作业挂起。要确定打印作业数，请输入 **qchk** 命令。

qhld 命令的公共命令格式是:

```
qhld [ -r ] {[ -#JobNumber ] [ -PQueue ] [ -uUser ]}
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **qhld** 命令，以获取完整的语法。

以下是如何使用 **qhld** 命令的示例:

1. 要将作业所在的任何打印队列上的作业号 452 挂起，请输入:

   ```
   qhld -#452
   ```

   按下 Enter 键。
2. 要将在打印队列 hp2 上排队的所有作业挂起，请输入:

   ```
   qhld -Php2
   ```

   按下 Enter 键。
3. 要将作业所在的任何打印队列上的作业号 452 释放，请输入:

   ```
   qhld -#452 -r
   ```

   按下 Enter 键。
4. 要将在打印队列 hp2 上排队的所有作业释放，请输入:

```
qhld -Php2 -r
```

按下 Enter 键。

使用 smit 命令

要使用 SMIT 挂起或释放打印作业，请输入:

```
smit qhld
```

将打印作业移动到另一个打印队列（qmov 命令）

在您发送打印作业到一个打印队列后，您可能要将打印作业移动到另一个打印队列。您可用 **qmov** 命令或 **smit** 命令移动它。

先决条件

- 对于本地打印作业，打印机必须物理地连接到您的系统上。
- 对于远程打印作业，必须配置您的系统以与远程打印服务器通信。

使用 qmov 命令

您可移动特定打印作业，或移动指定打印队列上所有打印作业，或移动指定用户发送的所有打印作业。要确定打印作业号，请输入 **qchk** 命令。

qmov 命令的常见格式是:

```
qmov -m NewQueue {[ -#JobNumber ] [ -PQueue ] [ -uUser ]}
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **qmov** 命令，以获取完整的语法。

以下是如何使用 **qmov** 命令的示例:

1. 要将作业号 280 移动到打印队列 hp2，请输入:

   ```
   qmov -mhp2 -#280
   ```

   按下 Enter 键。
2. 要将打印队列 hp4D 上的所有打印作业移动到打印队列 hp2，请输入:

   ```
   qmov -mhp2 -Php4D
   ```

使用 smit 命令

要使用 SMIT 移动打印作业，请输入:

```
smit qmov
```

为打印格式化文件（pr 命令）

pr 命令对您发送到打印的文件执行简单的格式化。通过管道操作将 **pr** 命令的输出传输到 **qprt** 命令来格式化您的文本。

一些有用的 **pr** 命令标志如下:

-d 两倍行距输出。

-h *"String"*	显示用引号（" "）圈起的指定字符串（而非文件名）作为页标题。标志和字符串应用一个空格分隔。
-l *Lines*	覆盖 66 行缺省值，并将页长度重新设置为由 *Lines* 变量指定的行数。如果 *Lines* 值小于页标题深度和尾部深度（以行为单位）的总和，则禁止页标题和尾部（就像指定 **-t** 标志一样）。
-m	合并文件。格式化标准输出，以便 **pr** 命令根据列位置数，将来自由 *File* 变量指定的每个文件的一行，并排地写入相等固定宽度的文本列中。不要将此标志与 **-***Column* 标志一起使用。
-n [*Width*][*Character*]	根据 *Width* 变量指定的数字数提供行编号。缺省值是 5 个数字。如果指定了 *Character*（任何非数字字符）变量，则它被附加到行号以将它与跟在行上的内容分隔开。缺省字符分隔符是 ASCII 制表符。
-o *Offset*	根据 *Offset* 变量指定的字符位置数缩排每一行。每行的总计字符位置数是宽度和偏移量的总和。*Offset* 的缺省值是 0。
-s*Character*	用由 *Character* 变量指定的单个字符（而非适当的空格数）来分隔列。*Character* 的缺省值是 ASCII 制表符。
-t	不显示标识页标题的五行和五行脚注。在每个文件的最后一行后停止，而不继续显示空格到页的结束。
-w *Width*	将每行的列位置数设置为 *Width* 变量指定的值。对于相等宽度多列输出，缺省值是 72。不另外有限制。如果未指定 **-w** 标志而指定了 **-s** 标志，则缺省宽度是 512 个列位置。
-*Column*	将列数设置为 *Column* 变量指定的值。缺省值是 1。不要与 **-m** 标志一起使用此选项。对于多列输出，假定使用 **-e** 和 **-i** 标志。文本列应绝不能超出页的长度（请参阅 **-l** 标志）。当此标志与 **-t** 标志一起使用时，请使用最小行数来写输出。
+*Page*	从 *Page* 变量指定的页号开始显示。缺省值是 1。

例如，要打印名为 **prog.c** 的文件及页标题和页号，请输入:

```
pr prog.c | qprt
```

按下 Enter 键。

缺省情况下，**pr** 命令将页标题和页号添加到 **prog.c**，并将它发送到 **qprt** 命令。页标题由文件上次修改的日期、文件名和页号组成。

例如，要指定名为 **prog.c** 的文件的标题，请输入:

```
pr -h "MAIN PROGRAM" prog.c | qprt
```

按下 Enter 键。

这打印 **prog.c**，用标题 MAIN PROGRAM 代替文件名。修改日期和页号仍将打印。

例如，要在多个列中打印名为 **word.lst** 的文件，请输入:

```
pr -3 word.lst | qprt
```

按下 Enter 键。

这在三个垂直列中打印 **word.lst** 文件。

例如，要在纸张上并排地打印几个文件:

```
pr -m -h "Members and Visitors" member.lst visitor.lst | qprt
```

这并排地打印 **member.lst** 和 **visitor.lst**，标题为 Members and Visitors。

例如，要修改名为 prog.c 的文件以在以后使用，请输入:

```
pr -t -e prog.c > prog.notab.c
```

按下 Enter 键。

这用空格替换 prog.c 中的制表符，并将结果放在 prog.notab.c 中。制表符位置在列 9、17、25、33，等等。**-e** 标志告诉 **pr** 命令替换制表符；**-t** 标志禁止页标题。

例如，要在两列中打印名为 myfile 的文件（以横向方式，文本磅值为 7），请输入:

```
pr -l66 -w172 -2 myfile | qprt -z1 -p7
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **pr** 命令，以获取完整的语法。

在 PostScript 打印机上打印 ASCII 文件

“文本格式系统”包含 enscript 过滤器，用于将 ASCII 打印文件转换为 PostScript 以在 PostScript 打印机上打印。当一个打印作业提交到 PostScript 打印队列时，**qprt -da** 命令调用此过滤器。

先决条件

- 打印机必须物理地连接到您的系统上。
- 打印机必须已配置和定义。
- 必须已安装文本格式服务的抄本部分。

您可能与 **qprt** 命令一起指定以下标志以定制将 ASCII 文件提交到 PostScript 打印队列时的输出。

-1+ 添加页标题。

-2+ 将输出格式化成两列。

-3+ 以一种漂亮的方式打印页标题、日期和页号。这有时称为**华丽的**方式。

-4+ 打印文件，即使它包含不可打印的字符。

-5+ 列出不包含在字体中的字符。

-h *string* 指定用于页标题的字符串。如果未指定此标志，则标题由文件名、修改日期和页号组成。

-l *value* 指定每页打印的最大行数。取决于磅值，每页上实际出现的行会少一些。

-L! 截断超过页宽度的行。

-p 指定磅值。如果未指定此标志，则假定使用磅值 10，除非指定了两列旋转方式（**-2+ -z1**），在此情况下使用值 7。

-s 指定字体样式。如果未指定此标志，则使用 Courier 字体。可接受的值如下:

Courier-Oblique
Helvetica
Helvetica-Oblique
Helvetica-Narrow
Helvetica-Narrow-Oblique
NewCenturySchlbk-Italic
Optima
Optima-Oblique
Palatino-Roman
Palatino-Italic
Times-Roman
Times-Italic

注: PostScript 打印机必须对指定的字体有访问权。

-z1 将输出旋转 90 度（横向方式）。

例如，要将 ASCII 文件 myfile.ascii 发送到名为 Msps1 的 PostScript 打印机，请输入:

```
qprt -da -PMsps1 myfile.ascii
```

按下 Enter 键。

例如，要将 ASCII 文件 myfile.ascii 发送到名为 Msps1 的 PostScript 打印机并以 Helvetica 字体打印，请输入:

```
qprt -da -PMsps1 -sHelvetica myfile.ascii
```

按下 Enter 键。

例如，要将 ASCII 文件 myfile.ascii 发送到名为 Msps1 的 PostScript 打印机并以磅值 9 打印，请输入:

```
qprt -da -PMsps1 -p9 myfile.ascii
```

按下 Enter 键。

将 ASCII 自动转换为 PostScript

生成 PostScript 打印文件的许多应用程序遵循如下约定，使 PostScript 文件的前两个字符为 %!，将打印文件标识为 PostScript 打印文件。要配置系统以检测提交给 PostScript 打印队列的 ASCII 打印文件，并在将它们发送给 PostScript 打印机前，自动将它们转换为 PostScript 文件，请执行这些步骤:

1. 在提示符下，请输入:

   ```
   smit chpq
   ```

   按下 Enter 键。
2. 输入 PostScript 队列名称，或使用“列表”功能从队列列表中选择。
3. 选择**打印机设置**菜单选项。
4. 更改值**自动检测打印文件类型?** 字段为**是**。

现在以下任何命令将 ASCII 文件转换为 PostScript 文件并在 PostScript 打印机上打印它。要转换 myfile.ascii，请在命令行输入以下任何信息:

```
qprt -Pps myfile.ps myfile.ascii
lpr -Pps myfile.ps myfile.ascii
lp -dps myfile.ps myfile.acsii
```

其中 ps 是 PostScript 打印队列。

覆盖打印文件类型的自动确定

您可能在以下情况下需要为 PostScript 打印覆盖打印文件类型的自动确定。

- 要打印一个名为 myfile.ps 的 PostScript 文件（此文件不以 %! 开始），在命令行输入以下内容，例如:

  ```
  qprt -ds -Pps myfile.ps
  ```
- 要打印一个名为 myfile.ps 的 PostScript 文件（以 %! 开始）的源列表，在命令行输入以下内容，例如:

  ```
  qprt -da -Pps myfile.ps
  ```

打印机、打印作业和队列的命令摘要

cancel	取消对行式打印机的请求
lp	将请求发送到行式打印机
lpq	检查假脱机队列
lpr	排队打印作业
lprm	从行式打印机假脱机队列除去作业
lpstat	显示行式打印机状态信息
pr	将文件写入标准输出
qcan	取消打印作业
qchk	显示打印队列的状态
qhld	挂起或释放打印作业
qmov	将打印作业移动到另一个打印队列
qpri	按优先顺序排列打印队列中的作业
qprt	启动打印作业

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 45 页的第 5 章, 『输入和输出重定向』

第 53 页的『文件系统』

第 56 页的『目录概述』

第 65 页的第 7 章, 『文件』

第 11 页的第 2 章, 『用户环境和系统信息』

第 9 章 备份文件和存储介质

一旦您的系统投入使用，下一个注意事项应该是备份文件系统、目录和文件。所有计算机文件都很有可能被故意地或意外地更改或擦除。如果以有条不紊的工作态度备份文件系统，则应该总是能轻松地恢复最近版本的文件或文件系统。

> **注**: 当硬盘崩溃时，包含在该磁盘上的信息将被破坏。唯一恢复破坏的数据的方法是从您的备份副本中检索信息。

有几种不同的备份方法。最常用的方法是常规备份，它是文件系统、目录或文件的一个副本，保留以用于文件传送，或在无意地更改或破坏原始数据时被使用。另一种备份形式是归档备份；此方法用于将来参考、历史目的，或如果原始数据损坏或丢失时用于恢复。

本章讨论以下内容:

建立备份策略

没有一种备份策略能满足所有用户的需要。例如，一种策略在有一个用户的系统上工作正常，但可能对服务 5 个或 10 个不同用户的系统就不适当。同样地，为每天有许多文件更改的系统而开发的策略，可能对于数据更改不频繁的系统就会效率很低。只有您才能为您的系统确定最佳备份策略，但是以下一般准则应该对您有所帮助:

确保您可以恢复主要丢失。

在任何单个固定磁盘发生故障后您的系统能继续运行吗？如果所有固定磁盘都出现故障，您能恢复您的系统吗？如果您丢失备份软盘或磁带遭火或被盗，您能恢复您的系统吗？尽管这些不是很可能发生，但任何的情况都是可能的。仔细思考每个这些可能的丢失情况，并设计出一个备份策略，使您能在任何这些情况发生后恢复您的系统。

定期地检查您的备份。

备份介质及其硬件可能不可靠。如果备份磁带或软盘的数据无法读回至固定磁盘，则它们的大的库也是无用的。要确定您的备份可用，请尝试定期地从备份磁带显示目录（为归档磁带使用 **restore -T** 或 **tar -t**）。如果使用软盘备份，并有多个软盘驱动器，请尝试从与创建它们的驱动器不同的驱动器上读取软盘。还可能需要这样的安全性，用第二个软盘集来重复每个级别 0 备份。如果使用流磁带设备，则可以使用 **tapechk** 命令在磁带上执行基本一致性检查。

保留旧的备份介质。

开发重新使用备份介质的常规循环；但是，不要重新使用*所有*备份介质。有时一个重要文件损坏或缺少几个月以后，您或另一个系统用户才会注意到。因为可能发生这样的情况，所以一定要保存旧的备份介质。例如，可以有三个循环来备份磁带或软盘:

- 每周一次，回收所有每日软盘，除周五的一个外。
- 每月一次，回收所有周五软盘，除本月的最后一个周五的一个外。这使最后四个周五的备份总是可用的。
- 每季度一次，回收所有每月软盘，除最后一个外。不确定地保留每季度的最后一个每月的软盘，可能在不同的大楼内。

备份文件系统前检查它们。

从受损文件系统进行备份可能是无用的。备份前，使用 **fsck** 命令检查文件系统的完整性是一个好策略。

确保备份期间文件不在使用。

确保备份时系统不在使用。如果系统在使用中，则备份文件时文件可能会更改，因此备份将不精确。

系统进行大更改前备份系统。

执行任何硬件测试或修复工作前，或者安装任何新设备、程序或其它系统功能部件前备份整个系统。

其它因素

当计划和实现备份策略时，考虑以下因素:

- 数据多久会更改？操作系统数据不经常更改，因此您不需要频繁备份。另一方面，用户数据通常频繁更改，您应该频繁备份。
- 系统上有多少用户？用户的数量影响存储介质的数量和需要的备份频率。
- 重新创建数据有多困难？要考虑如果没有可用的备份，则将无法重新创建一些数据，这点很重要。

有一个适当的备份策略以保护您的数据是很重要的。评估您的站点的需要将帮助您确定时用于您的最佳的备份策略。频繁地和常规地执行用户信息备份。如果没有实现一个好的备份策略，则恢复丢失的数据就很困难。

备份介质

可用几种不同类型的备份介质。可用于特定系统配置的不同类型的备份介质同时取决于您的软件和硬件。最常用的类型是 5.25 英寸软盘、8 mm 磁带、9 磁道磁带及 3.5 英寸软盘。

> **警告:** 运行 **backup** 命令将导致丢失选定备份介质上先前存储的所有资料。

软盘

软盘是标准备份介质。除非您使用 **backup -f** 命令指定不同的设备，否则 **backup** 命令将把其输出自动写到软盘驱动器 **/dev/rfd0** 设备。要备份到缺省磁带设备，请输入 **/dev/rmt0** 并按下 Enter 键。

处理软盘时要小心。因为每条信息都会占用软盘上的这样一小块区域，所以小的划痕、灰尘、食物或烟草制品都会使信息不可用。确保记住以下几点:

- 不要接触记录面。
- 将软盘远离磁体和磁场源，如电话、录音设备和电子计算器。
- 不要让软盘在很热和很冷的环境。建议的温度范围是摄氏 10 度到摄氏 60 度（华氏 50 度到华氏 140 度）之间。
- 正确地看管将帮助您避免信息丢失。
- 定期制作软盘的备份。

  警告: 软盘驱动器和软盘必须是正确的类型以保证成功存储数据。如果您在 3.5 英寸软盘驱动器中使用错误的软盘，则软盘上的数据将被破坏。

软盘驱动器使用以下 3.5 英寸软盘:

- 1 MB 容量（大约存储 720 KB 数据）
- 2 MB 容量（大约存储 1.44 MB 数据）。

磁带

由于磁带的高容量和耐久性，通常选择它来存储大文件或许多文件（如，文件系统的归档副本）的选择。它还用于将大量文件从一个系统转移到另一个系统。磁带不广泛用于存储频繁访问的文件，因为其它介质提供更快的访问时间。

使用诸如 **backup**、**cpio** 和 **tar** 的命令来创建磁带文件、打开磁带机、写入它和关闭它。

格式化软盘（format 或 fdformat 命令）

警告: 格式化软盘会破坏该软盘上任何现有数据。

您可用 **format** 和 **fdformat** 命令格式化 *Device* 参数指定的软盘驱动器（缺省情况下为 **/dev/rfd0** 设备）中的软盘。**format** 命令确定设备类型，可以是以下类型之一:

- 5.25 英寸低密度软盘（360 KB），包含 40x2 磁道，每个有 9 个扇区
- 5.25 英寸高容量软盘（1.2 MB），包含 80x2 磁道，每个有 15 个扇区
- 3.5 英寸低密度软盘（720 KB），包含 80x2 磁道，每个有 9 个扇区
- 3.5 英寸高容量软盘（2.88 MB），包含 80x2 磁道，每个有 36 个扇区

对于所有软盘类型，扇区大小都是 512 个字节。

format 命令格式化高密度的软盘，除非 *Device* 参数指定不同的密度。

fdformat 命令格式化低密度的软盘，除非指定了 **-h** 标志。*Device* 参数指定包含要格式化的软盘的设备（如用于驱动器 0 的 **/dev/rfd0** 设备）。

格式化软盘之前，**format** 和 **fdformat** 命令提示您验证。这允许您在必要时干净地结束操作。

例如，要格式化 **/dev/rfd0** 设备中的软盘，请输入:

```
format -d /dev/rfd0
```

按下 Enter 键。

例如，要不检查坏磁道就格式化软盘，请输入:

```
format -f
```

按下 Enter 键。

例如，要格式化 **/dev/rfd1** 设备中 5.25 英寸、1.2 MB 软盘驱动器中的 360 KB 软盘，请输入:

```
format -l -d /dev/rfd1
```

按下 Enter 键。

例如，要在使用 **fdformat** 命令时，强制高密度格式化软盘，请输入:

```
fdformat -h
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **format** 命令，以获取完整的语法。

检查文件系统的完整性（fsck 命令）

可以使用 **fsck** 命令检查和交互式地修复不一致文件系统。在每个文件系统上运行该命令作为系统初始化的一部分，这很重要。您必须能读取文件系统驻留的设备文件（例如，**/dev/hd0** 设备）。通常，文件系统是一致的，并且 **fsck** 命令只报告有关文件系统中的文件数、使用的块数和空闲块数。如果文件系统不一致，则 **fsck** 命令显示有关找到的不一致的信息，并提示您以获取修复它们的许可权。**fsck** 命令很保守地进行其修复工作，并尝试避免可能导致丢失有效数据的操作。但是在某些情况下，**fsck** 命令建议毁灭已损坏的文件。

> **警告:** 总是在系统故障后在文件系统上运行 **fsck** 命令。更正的操作可能导致一些数据丢失。每个一致性校正的缺省操作是等待操作员输入 `yes` 或 `no`。如果您对受影响的文件不具有写许可权，则 **fsck** 命令将缺省为 `no` 响应。

例如，要检查所有缺省文件系统，请输入:

```
fsck
```

按下 Enter 键。

fsck 命令的此格式在对文件系统进行任何更改前询问您以获取许可权。

例如，要自动修正缺省文件系统的小问题，请输入:

```
fsck -p
```

按下 Enter 键。

例如，要检查 **/dev/hd1** 文件系统，请输入:

```
fsck /dev/hd1
```

按下 Enter 键。

这检查位于 `/dev/hd1` 设备上的未安装文件系统。

> **注:** **fsck** 命令不校正已安装文件系统。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **fsck** 命令，以获取完整的语法。

复制到软盘或从软盘复制（flcopy 命令）

可以将软盘（作为 **/dev/rfd0** 打开）复制到名为 floppy 的文件，此文件使用 **flcopy** 命令在当前目录中创建。根据需要显示消息：更改软盘，完成后击中返回。然后，**flcopy** 命令将 floppy 文件复制到软盘。

例如，在当前目录中要将 /dev/rfd1 复制到 floppy 文件，请输入：

```
flcopy -f /dev/rfd1 -r
```

按下 Enter 键。

例如，要复制软盘的前 100 个磁道，请输入：

```
flcopy -f /dev/rfd1 -t 100
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **flcopy** 命令，以获取完整的语法。

将文件复制到磁带或磁盘（cpio -o 命令）

可以使用 **cpio -o** 命令从标准输入读文件路径名，并将这些文件连同路径名和状态信息复制到标准输出。路径名不能超过 128 个字符。避免 **cpio** 命令路径名组成许多单独链接的文件，因为它可能没有足够的内存来跟踪路径名，并会丢失链接信息。

例如，要将当前目录（其名称以 .c 结束）中的文件复制到软盘，请输入：

```
ls *.c | cpio -ov >/dev/rfd0
```

按下 Enter 键。**-v** 标志显示每个文件的名称。

例如，要将当前目录和所有子目录复制到软盘，请输入：

```
find . -print | cpio -ov >/dev/rfd0
```

按下 Enter 键。

这保存以当前目录（.）开始的目录树，并包含所有其子目录和文件。要使用较短的命令字符串，请输入：

```
find . -cpio /dev/rfd0 -print
```

按下 Enter 键。

-print 条目显示它复制的每个文件的名称。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cpio** 命令，以获取完整的语法。

从磁带或磁盘复制文件（cpio -i 命令）

cpio -i 命令从标准输入读 **cpio -o** 命令创建的归档文件，并从它复制具有与 *Pattern* 参数匹配的名称的文件。将这些文件复制到当前目录树。可以使用 **ksh** 命令中描述的文件名标志法来列出多个 *Pattern* 参数。*Pattern* 参数的缺省值是星号（*），选择当前目录中的所有文件。在表达式中，如 [a-z]，连字符（-）意味着根据当前整理序列*从头到尾*。

注: 模式 ″*.c″ 和 ″*.o″ 必须由引号圈起，以防止 shell 将星号（*）作为模式匹配字符对待。这是特例，在此例中， **cpio** 命令自己解码模式匹配字符。

例如，要列出使用 **cpio** 命令已保存在软盘上的文件，请输入:

```
cpio -itv </dev/rfd0
```

按下 Enter 键。

这以 **cpio** 命令格式显示先前已保存在 /dev/rfd0 文件上的数据的目录。该列表类似于 **ls -l** 命令产生的长目录列表。要只列出文件路径名，只使用 **-it** 标志。

例如，要从软盘复制先前已用 **cpio** 命令保存的文件，请输入:

```
cpio -idmv </dev/rfd0
```

按下 Enter 键。

这会将用 **cpio** 命令先前已保存在 /dev/rfd0 文件上的文件复制回文件系统（指定 **-i** 标志）。**-d** 标志允许 **cpio** 命令创建相应的目录（如果已保存目录树）。当已保存文件时，**-m** 标志维护有效的上次修改时间。**-v** 标志导致 **cpio** 命令显示它复制的每个文件的名称。

例如，要从软盘复制选定文件，请输入:

```
cpio -i "*.c" "*.o" </dev/rfd0
```

按下 Enter 键。

这从软盘复制以 .c 或 .o 结束的文件。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **cpio** 命令，以获取完整的语法。

复制到磁带或从磁带复制（tcopy 命令）

可以使用 **tcopy** 命令复制磁带。

例如，要从一个流式磁带复制到 9 磁道的磁带，请输入:

```
tcopy /dev/rmt0 /dev/rmt8
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **tcopy** 命令，以获取完整的语法。

检查磁带的完整性（tapechk 命令）

可以使用 **tapechk** 命令执行对连接的流式磁带设备的基本一致性检查。流式磁带设备的一些硬件故障可通过简单地读磁带检测。**tapechk** 命令提供在文件级别执行磁带读的方法。

例如，要检查流式磁带设备上的头三个文件，请输入:

```
tapechk 3
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **tapechk** 命令，以获取完整的语法。

压缩文件（compress 和 pack 命令）

可以使用 **compress** 命令和 **pack** 命令压缩要存储的文件，并使用 **uncompress** 和 **unpack** 展开恢复的文件。压缩和展开文件的过程需要时间，但文件在压缩后，数据使用备份介质上更少的空间。

要压缩文件系统，请使用以下方法之一:

- 将 **-p** 选项与 **backup** 命令一起使用
- 使用 **compress** 或 **pack** 命令

压缩文件的原因一般可分成以下类别:

- 节省存储量和归档系统资源:
  - 在进行备份前压缩文件系统以留存磁带空间。
  - 压缩由夜间运行的 shell 脚本创建的日志文件；使脚本在退出前很容易进行文件压缩。
  - 压缩当前不访问的文件。例如，属于一个因长期请假而离开的用户的文件可被压缩成一个 **tar** 压缩文档，并放置到磁盘上或磁带中，并在以后恢复。
- 通过在网络上发送文件前先进行压缩来节省钱和时间。

注:

1. **compress** 命令在压缩的同时可能会用完文件系统中的工作空间。该命令在删除任何未压缩的文件前创建压缩的文件，这样它就需要大约比文件总计大小大 50% 的空间。
2. 文件压缩可能因文件已压缩而失败。如果 **compress** 命令不能减少文件大小，则命令失败。

使用 compress 命令

compress 命令使用自适应 Lempel-Zev 编码减少文件的大小。由 *File* 参数指定的每个原文件都由附加了 **.Z** 至其名称的压缩文件替换。压缩文件保留原文件的相同所有权、方式以及访问和修改时间。如果未指定文件，则标准输入压缩到标准输出。如果压缩未减少文件的大小，则在标准错误中写入一条消息，且不替换原文件。

要将压缩文件恢复为其原始格式，请使用 **uncompress** 命令。

压缩量取决于输入的大小，*Bits* 变量指定的每代码位数，以及公共子字符串的分布。通常，源代码或英语文本减小百分之 50 到 60。一般而言，**compress** 命令的压缩与 **pack** 命令（使用自适应霍夫曼编码）实现的压缩相比，压缩率更高，用于计算的时间更少。

例如，要压缩 **foo** 文件并将压缩百分率写入标准错误，请输入:

```
compress -v foo
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **compress** 命令，以获取完整的语法。

使用 pack 命令

pack 命令以一种使用霍夫曼编码的压缩格式，存储由 *File* 参数指定的一个或多个文件。输入文件由压缩文件替换，该压缩文件的名称从原文件名（*File***.z**）派生，并与原文件有相同的访问方式、访问和修改日期以及所有者。输入文件名包含的字节数不能超过 253，以允许用于添加的 **.z** 后缀的空间。如果 **pack** 命令成功，则除去原文件。要将压缩文件恢复为其原始格式，请使用 **unpack** 命令。

如果 **pack** 命令不能创建较小的文件，则它停止处理并报告不能节省空间。（无法节省空间一般发生于小文件或具有统一字符分布的文件。）节省的空间量取决于输入文件的大小和字符频率分布。因为解码树形成每个 **.z** 文件的第一部分，所以对于小于三个块的文件您没有节省空间。通常，文本文件减小百分之 25 到 40。

pack 命令的出口值是无法压缩文件的数目。在任何以下条件下不能进行压缩:

- 文件已压缩。
- 输入文件名包含的字节数大于 253。
- 文件有链接。
- 文件是一个目录。
- 文件无法打开。
- 压缩未节省存储块。
- 名为 *File***.z** 的文件已存在。
- 无法创建 **.z** 文件。
- 处理期间发生 I/O 错误。

例如，要压缩文件 chap1 和 chap2，请输入:

```
pack chap1 chap2
```

按 Enter 键。

这压缩 chap1 和 chap2，使用名为 **chap1.z** 和 **chap2.z** 的文件替换它们。**pack** 命令显示每个文件大小减少的百分比。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **pack** 命令，以获取完整的语法。

展开已压缩文件（uncompress 和 unpack 命令）

可以使用 **uncompress** 和 **unpack** 命令来展开已压缩文件。

使用 uncompress 命令

uncompress 命令恢复 **compress** 命令压缩的原文件。*File* 变量指定的每个已压缩文件已除去并由已展开副本替换。展开文件具有与已压缩版本相同的名称，但是没有 **.Z** 扩展名。展开文件保留与原文件相同的所有权、方式以及访问和修改次数。如果未指定文件，则标准输入被展开到标准输出。

尽管与 **uncompress** 命令相似，**zcat** 命令总是将展开输出写入标准输出。

例如，要解压缩 foo 文件，请输入:

```
uncompress foo
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **uncompress** 命令，以获取完整的语法。

使用 unpack 命令

unpack 命令展开 **pack** 命令创建的文件。对每个指定的文件，**unpack** 命令搜索称为 *File***.z** 的文件。如果该文件是一个已压缩文件，则 **unpack** 命令用其展开版本替换它。**unpack** 命令通过从 *File* 除去 **.z** 后缀来重命名新文件。此新文件具有与原始已压缩文件相同的访问方式、访问和修改日期及所有者。

unpack 命令仅操作以 **.z** 结束的文件。结果，当指定不以 **.z** 结束的文件名时，**unpack** 命令添加后缀并搜索目录查询具有该后缀的文件名。

此出口值是 **unpack** 命令无法解包的文件数。如果任何以下情况退出，则无法解包文件:

- 文件名（除 **.z**）具有多于 253 个字节。
- 文件无法打开。
- 文件不是已压缩文件。
- 具有已解包文件名的文件已存在。
- 无法创建已解包文件。

> **注**: **unpack** 命令将警告写到标准错误（如果它解包的文件有链接）。新的已解包文件具有不同于创建它的已压缩文件的 i-node（索引节点）号。但是，链接到已压缩文件的原始 i-node 号的任何其它文件仍存在并已压缩。

例如，要解包已压缩文件 `chap1.z` 和 `chap2`，请输入:

```
unpack chap1.z chap2
```

按下 Enter 键。

这展开已压缩文件 `chap1.z` 和 `chap2.z`，并使用名为 `chap1` 和 `chap2` 的文件替换它们。注意，可以提供 **unpack** 命令，具有或不具有 **.z** 后缀的文件名。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **unpack** 命令，以获取完整的语法。

备份文件（backup 命令）

> **警告**: 如果尝试备份已安装的文件系统，则显示一条消息。**backup** 命令继续，但是文件系统中可能发生不一致。此情况不适用于根（**/**）文件系统。

可以使用 **backup** 命令或 **smit** 命令，在备份介质（如磁带或软盘）上创建文件的副本。副本的格式是以下备份格式中的一种:

- 使用 **-i** 标志按名称备份的特定文件。
- 使用 *-Level* 和 *FileSystem* 参数按 i-node 号备份的整个文件系统。

> **注**:
>
> 1. 系统备份期间修改文件时总是存在数据乱码的可能性。因此，确保系统备份过程期间系统活动最少。
> 2. 如果备份做成 8 mm 磁带，且设备块大小设置为 0（零），则不可能直接从磁带上恢复。如果您已在 0 设置的情况下完成了备份，则可以通过使用 **restore** 命令下描述的特殊过程从它们中恢复。

> **警告**: 确保您指定的标志匹配备份介质。

使用 backup 命令

例如，要按名称备份在 **$HOME** 目录中选定的文件，请输入:

```
find $HOME -print | backup -i -v
```

按下 Enter 键。

-i 标志提示系统从标准输入读取要备份文件的名称。**find** 命令在用户目录中生成文件列表。此列表作为标准输入通过管道操作传递给 **backup** 命令。**-v** 标志显示复制每个文件时的进程报告。文件将备份在本地系统的缺省备份设备上。

例如，要备份根文件系统，请输入:

```
backup -0 -u /
```

按下 Enter 键。

0 级别和 **/** 告诉系统备份 **/**（根）文件系统。将文件系统备份至 **/dev/rfd0** 文件。**-u** 标志告诉系统更新 **/etc/dumpdates** 文件中的当前备份级别记录。

例如，要备份 **/**（根）文件系统中自上次 0 级别备份以来修改过的所有文件，请输入:

```
backup -1 -u /
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **backup** 命令，以获取完整的语法。

使用 smit 命令

也可以使用 **smit** 运行 **backup**命令。

1. 在提示符下输入:

   ```
   smit backup
   ```

   按下 Enter 键。
2. 在**目录全路径名**字段中，输入通常安装文件系统的目录的路径名:

   ```
   /home/bill
   ```

   按下 Enter 键。
3. 在**备份**设备或**文件**字段中，输入输出设备名，如以下原始磁带设备的示例:

   ```
   /dev/rmt0
   ```

   按下 Enter 键。
4. 如果想要错误消息打印在屏幕上，则使用 Tab 键切换可选的**报告备份的每个阶段**字段。
5. 在系统管理环境中，使用**要写到备份介质的最多块数**字段的缺省值，因为此字段不应用于磁带备份。
6. 按下 Enter 键备份命名的目录或文件系统。
7. 运行 **restore -t** 命令。如果此命令生成错误消息，则必须重复整个备份。

恢复备份文件（restore 命令）

可以从备份介质读 **backup** 命令写的文件，并使用 **restore** 命令或 **smit** 命令在本地系统上恢复它们。

注:

1. 文件必须使用与备份它们的方法相同的方法恢复。例如，如果按名称备份文件系统，则它必须按名称恢复。
2. 当需要多个软盘时，**restore** 命令读已安装的软盘，提示您需要新软盘并等待您的响应。插入新软盘后，按下 Enter 键以继续恢复文件。

使用 restore 命令

例如，要列出先前备份的文件的名称，请输入:

```
restore -T
```

按下 Enter 键。

从 **/dev/rfd0** 缺省备份设备读信息。如果备份了单个文件，则仅显示文件名。如果备份了整个文件系统，则还显示 i-node 号。

例如，要将文件恢复到主文件系统，请输入:

```
restore -x -v
```

按下 Enter 键。

-x 标志从备份介质抽取所有文件，并将它们恢复到文件系统中适当的位置。**-v** 标志显示恢复每个文件时的进展报告。如果恢复文件系统正在备份，则使用它们的 i-node 号命名文件。否则，仅显示名称。

例如，要复制 **/home/mike/manual/chap1** 文件，请输入:

```
restore -xv /home/mike/manual/chap1
```

按下 Enter 键。

此命令从备份介质抽取 **/home/mike/manual/chap1** 文件并恢复它。**/home/mike/manual/chap1** 文件必须是 **restore -T** 命令能显示的名称。

例如，要复制目录中所有名为 **manual** 的文件，请输入:

```
restore -xdv manual
```

按下 Enter 键。

此命令恢复 **manual** 目录及其中的文件。如果该目录不存在，则名为 **manual** 的目录在当前目录中创建，以保留恢复的文件。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **restore** 命令，以获取完整的语法。

使用 smit 命令

也可以使用 **smit** 运行 **restore**命令。

1. 在提示符下输入:

   ```
   smit restore
   ```

   按下 Enter 键。
2. 将您的条目放入**目标目录**字段。这是想让已恢复的文件驻留的目录。
3. 继续至**备份设备**或**文件**字段，输入输出设备名称，并按下 Enter 键，如在原始磁带设备的下例:

   ```
   /dev/rmt0
   ```

   如果设备不可用，则显示类似于以下的消息:

   ```
   无法打开 /dev/rmtX，没有这样的文件或目录。
   ```

此消息表明系统无法到达设备驱动程序，因为 /dev 目录中没有 rmtX 的文件。只有处于可用状态的项才在 /dev 目录中。

4. 对于**单个输入中读的块数**字段，建议使用缺省。
5. 按下 Enter 键，恢复指定的文件系统或目录。

归档文件（tar 命令）

归档备份是您可使用的另一种支持格式；此方法用于一个或多个文件的复制，或用于将来参考和历史目的而保存的整个数据库，或如果原始数据损坏或丢失，则用于恢复。通常，当从系统中除去此特定数据时使用归档。

可以使用 **tar** 命令将文件写入归档存储器或从归档存储器中检索文件。**tar** 命令在缺省设备（通常是磁带）上查找归档，除非您指定另一个设备。

当写入归档时，**tar** 命令使用临时文件（**/tmp/tar*** 文件）并在内存中保留含几个链接的文件的表。如果 **tar** 命令无法创建临时文件，或者，如果没有足够可用的内存保存链接表，您将接收到一条错误消息。

例如，要将 file1 和 file2 文件写入缺省磁带机上的新归档，请输入:

```
tar -c file1 file2
```

按 Enter 键。

例如，要从 /dev/rmt2 磁带设备上的归档文件抽取 /tmp 目录中的所有文件，并根据修改时间使用抽取时间，请输入:

```
tar -xm -f/dev/rmt2 /tmp
```

按 Enter 键。

例如，要从当前目录显示 out.tar 磁盘归档文件中的文件的名称，请输入:

```
tar -vtf out.tar
```

按 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **tar** 命令，以获取更多信息和精确的语法。

备份文件和存储介质的命令摘要

backup	备份文件和文件系统
compress	压缩和展开数据
cpio	将文件复制到归档存储器和目录以及从归档存储器和目录复制文件
fdformat	格式化软盘
flcopy	复制到软盘和从软盘复制
format	格式化软盘
fsck	检查文件系统一致性和交互性修复文件系统
pack	压缩文件
restore	从本地设备复制以前备份的由 **backup** 命令创建的文件系统或文件
tapechk	检查流式磁带设备的一致性
tar	操作归档
tcopy	复制磁带
uncompress	压缩和展开数据
unpack	展开文件

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 45 页的第 5 章, 『输入和输出重定向』

第 53 页的『文件系统』

第 56 页的『目录概述』

第 65 页的第 7 章, 『文件』

第 113 页的第 10 章, 『文件和系统安全性』

第 10 章 文件和系统安全性

计算机安全性的目的是保护存储在计算机系统上的信息，此信息是重要的资源。信息安全性旨在:

完整性 所有信息的值取决于其准确度。如果对数据进行未经授权的更改，则此数据将丢失一些或所有值。
机密性 信息的价值取决于其机密性。
可用性 信息必须是就绪可用的。

在开始使用系统之前计划和实现您的安全性策略是很有帮助的。对以后更改安全性策略将费时很多，所以最先开始计划能为以后节省很多时间。

本章讨论以下内容:

安全性威胁

对信息安全性的威胁可由以下类型的行为引起:

粗心 经常由于系统的授权用户粗心而违反信息安全性。例如，如果粗心对待密码，则没有其它安全性机制能阻止对帐户和数据的未授权访问。
浏览 许多安全性问题由浏览器引起，系统的授权用户浏览系统，查找没有小心保护的数据。
渗透 渗透表示故意攻击系统。尝试渗透系统的个人将研究系统以了解其安全性弱点，并故意地计划旨在使用这些弱点的攻击。

尽管系统渗透通常代表对信息安全性的最大威胁，但是不要低估由粗心或浏览导致的问题。

基本安全性

每个系统都应该维护由以下基本安全性策略表示的安全性级别:

备份

物理地安全、可靠的和最新的系统备份是一个最重要的安全性策略。有了好的系统备份，则可以最小损失从任何系统问题还原。记录备份策略，并包含有关以下内容的信息:

- 进行备份的频率
- 备份的类型（系统、数据或增量）
- 验证备份磁带的方法
- 存储备份磁带的方法

有关更多信息，请参阅“第 99 页的第 9 章，『备份文件和存储介质』”。

标识和认证

标识和认证建立您的身份。您需要登录到系统。您提供您的用户名和密码（如果帐户有密码，在安全系统中，所有帐户应该有密码或帐户无效）。如果密码正确，则可登录该帐户；您获取访问权和帐户的特权。

由于密码是对您的帐户的唯一保护，请小心选择和保护您的密码。许多要侵入系统的尝试以尝试猜测密码开始。操作系统通过分别地存储用户密码和其它用户信息提供重要的密码保护。用户的加密密码和其它安全性相关数据都存储在 **/etc/security/passwd** 文件中。该文件应该仅可由 root 用户访问。有了对加密密码的这种限定访问，攻击者无法用程序（简单地在所有可能的密码中循环的）来译码密码。

仍可能通过重复尝试登录帐户猜密码。如果密码平常或不经常更改，则这样的尝试可能会轻易成功。

登录用户标识

操作系统还通过其登录用户标识来标识用户。登录用户标识允许系统跟踪所有用户操作到他们的源。在用户登录系统后，但在运行初始用户程序之前，系统将进程的登录标识设置为在用户数据库中找到的用户标识。登录会话期间的所有后继进程都使用该标识标记。这些标记提供登录用户标识执行的所有活动的跟踪。

用户可以重新设置会话期间有效的用户标识、实用户标识、有效组标识、实组标识和辅助组标识，但是无法更改登录用户标识。

无人照管终端

如果终端已登录而无人照管，则所有系统都是易受攻击的。当系统管理员已使用 root 用户权限启用的终端无人照管时，会发生最严重的问题。一般，用户应该在任何离开终端的时候注销。

可以通过在 **/etc/profile** 文件中设置 **TMOUT** 和 **TIMEOUT** 参数来强制终端在不活动的一段时间后注销，方法。**TMOUT** 参数在 **ksh**（Korn）shell 中工作，**TIMEOUT** 参数在 **bsh**（Bourne）shell 中工作。有关 **TMOUT** 参数的更多信息，请参阅第 147 页的『Korn shell 或 POSIX shell 中的参数替换』。有关 **TIMEOUT** 参数的更多信息，请参阅第 182 页的『Bourne shell 中的变量替换』。

下例（取自 **.profile** 文件）强制终端在不活动一小时后注销:

```
TO=3600
echo "Setting Autologout to $TO"
TIMEOUT=$TO
TMOUT=$TO
export TIMEOUT TMOUT
```

> **注**: 用户可通过在主目录中的 **.profile** 文件中指定不同的值来重设 **/etc/profile** 文件中的 **TMOUT** 和 **TIMEOUT** 值。

文件所有权和用户组

初始时，文件的所有者由创建文件的人员的用户标识来标识。文件的所有者确定谁可读、写（修改）或执行文件。所有权可以使用 **chown** 命令更改。

将每个用户标识指定到具有唯一组标识的组。系统管理员在设置系统时创建用户的组。当创建新文件时，操作系统将许可权指定给创建它的用户标识、包含文件所有者的组标识和称为 `others` 的组（由所有其它用户组成）。**id** 命令显示您的用户标识（UID）、组标识（GID）和您所属的所有组的名称。

在文件列表（如用 **ls** 命令显示的列表）中，用户的组总是以下列顺序表示：用户、组和其它用户。如果需要找出您的组名，则 **groups** 命令显示一个用户标识的所有组。

更改文件或目录所有权（chown 命令）

要更改文件的所有者，请使用 **chown** 命令。

当指定 **-R** 选项时，**chown** 命令从指定的目录在目录结构中递归地下降。当遇到符号链接时，链接指向的文件或目录的所有权更改；符号链接的所有权未更改。

> **注**：只有 root 用户能更改另一个文件的所有者。当指定 **-f** 选项时，不显示错误。

例如，要更改 **program.c** 文件的所有者，请输入：

```
chown jim program.c
```

按下 Enter 键。

现在 **program.c** 文件的用户访问许可权应用于 **jim**。作为所有者，**jim** 可以使用 **chmod** 命令允许或拒绝其它用户访问 **program.c** 文件。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **chown** 命令，以获取完整的语法。

文件和目录访问方式

每个文件都有一个所有者。对于新的文件，创建文件的用户是该文件的所有者。所有者指定对文件的访问方式。访问方式授予其它系统用户读、修改或执行文件的许可权。只有文件的所有者或具有 root 权限的用户才可以更改文件的访问方式。

有三类用户：用户 / 所有者、组和所有其它用户。以下面三种方式的组合将访问权授予这些用户类：读、写或执行。当创建新文件时，缺省许可权是创建文件的用户有的读、写和执行许可权。其它两个组具有读和执行许可权。下表说明三类用户组的缺省文件访问方式：

类	读	写	执行
所有者	是	是	是
组	是	否	是
其它用户	是	否	是

系统确定谁有许可权，以及对于每个这些活动他们具有的许可权级别。访问方式在操作系统中以符号和数字两种方式表示。

访问方式的符号表示

访问方式以符号方式表示，如下:

r 表示读许可权，允许用户查看文件的内容。
w 表示写许可权，允许用户修改文件的内容。
x 表示执行许可权。对于可执行文件（包含程序的普通文件），执行许可权意味着程序可运行。对于目录，执行许可权意味着目录的内容可搜索。

文件或目录的访问方式由九个字符表示。头三个字符表示当前**所有者**许可权，第二组三个字符表示当前**组**许可权，第三组三个字符表示**其它用户**许可权的当前设置。九字符集中的连字符（**-**）表明未给定许可权。例如，访问方式设置为 rwxr-xr-x 的文件授予所有三个组读和执行许可权，但只授予文件所有者写许可权。这是缺省设置的符号表示。

ls 命令与 **-l**（小写 L）标志一起使用时，给出当前目录的详细列表。**ls -l** 列表中的头 10 个字符显示文件类型和三个组中每个组的许可权。**ls -l** 命令还告诉您与每个文件和目录关联的所有者和组。

第一个字符表示文件的类型。剩余的九个字符包含三类用户中每类用户的文件许可权信息。以下符号用于表示文件的类型:

- 常规文件
d 目录
b 块特殊文件
c 字符特殊文件
p 管道特殊文件
l 符号链接
s 套接字。

例如，这是一个样本 **ls -l** 列表:

```
-rwxrwxr-x   2   janet   acct   512 Mar 01 13:33 january
```

在此，第一个连字符（-）表示常规文件。接下来九个字符（rwxrwxr-x）表示“用户”、“组”和“其它用户”访问方式，如以上所讨论的。janet 是文件所有者，acct 是 Janet 的组的名称。512 是以字节计的文件大小，Mar 01 13:33 是上次修改日期和时间，january 的文件名。2 表示存在多少个到文件的链接。

访问方式的数字表示

以数字方式，读访问用值 4 表示，写访问用值 2 表示，执行许可权用值 1 表示。1 和 7 之间的总计值表示每个组（用户、组和其它用户）的访问方式。下表说明每一访问级别的数字值:

总计值	读	写	执行
0	-	-	-
1	-	-	1
2	-	2	-
3	-	2	1
4	4	-	-
5	4	-	1
6	4	2	-
7	4	2	1

当文件创建时，缺省文件访问方式是 755。这意味着用户有读、写和执行许可权（4+2+1=7），组有读和执行许可权（4+1=5），所有其它用户有读和执行许可权（4+1=5）。要更改您拥有的文件的访问许可权方式，运行 **chmod**（更改方式）命令。

显示组信息（lsgroup 命令）

要显示系统上所有组（或指定的组）的属性，请使用 **lsgroup** 命令。如果一个或多个属性无法读取，则 **lsgroup** 命令列出尽可能多的信息。属性信息显示为 *Attribute*=*Value* 定义，每个由一个空白空格分隔。

列出系统上所有的组

要列出系统上所有的组，请输入:

```
lsgroup ALL
```

按下 Enter 键。

系统在一个类似以下的列表中显示每个组、组标识和组中所有的用户:

```
system  0        arne,pubs,ctw,geo,root,chucka,noer,su,dea,
backup,build,janice,denise
staff   1        john,ryan,flynn,daveb,jzitt,glover,maple,ken
gordon,mbrady
bin     2        root,bin
sys     3        root,su,bin,sys
```

显示所有组的特定属性

要显示所有组的特定属性，请执行以下操作之一:

- 可以通过 Attribute=Value 格式列出属性，由一个空白空格分隔。这是缺省样式。例如，要列出系统上所有组的标识和用户，请输入:

  ```
  lsgroup -a id users ALL | pg
  ```

  按下 Enter 键。附加列出属性。

  显示一个类似于以下的列表:

  ```
  system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build
  staff id=1 users=john,ryan,flynn,daveb,jzitt,glover,maple,ken
  ```

- 还可以用节的格式列出信息。例如，要以节格式列出系统上所有组的标识和用户，请输入:

  ```
  lsgroup -a -f id users ALL | pg
  ```

  按下 Enter 键。

  显示一个类似于以下的列表:

  ```
  system:
     id=0
     users=pubs,ctw,geo,root,chucka,noer,su,dea,backup,build

  staff:
     id=1
     users=john,ryan,flynn,daveb,jzitt,glover,maple,ken

  bin:
     id=2
     users=root,bin

  sys:
     id=3
     users=root,su,bin,sys
  ```

显示特定组的所有属性

要显示特定组的所有属性，可以使用两种样式中的一种，列出所有组的特定属性:

- 可以用 Attribute=Value 格式列出每个属性，由空白空格分隔。这是缺省样式。例如，要列出组系统的所有属性，请输入:

  ```
  lsgroup system
  ```

  按下 Enter 键。

  显示一个类似于以下的列表:

  ```
  system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
  ```

- 还可以用节的格式列出信息。例如，要以节的格式列出组 bin 的所有属性，请输入:

  ```
  lsgroup -f system
  ```

  按下 Enter 键。

  显示类似于以下的列表:

  ```
  system:
     id=0   users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,
  backup,build,janice,denise
  ```

列出特定组的特定属性

要列出特定组的特定属性，请输入:

```
lsgroup -a Attributes Group
```

按下 Enter 键。

例如，要列出组 bin 的标识和用户，请输入:

```
lsgroup -a id users bin
```

按下 Enter 键。

显示一个类似于以下的列表:

```
bin id=2 users=root,bin
```

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lsgroup** 命令，以获取完整的语法。

更改文件或目录许可权（chmod 命令）

要修改指定文件的读、写和执行许可权，并修改指定目录的搜索许可权方式，请使用 **chmod** 命令。

- 例如，要将一个许可权类型添加到 **chap1** 和 **chap2** 文件，请输入:

  ```
  chmod g+w chap1 chap2
  ```

  按下 Enter 键。

  这将组成员的写许可权添加到文件 chap1 和 chap2。

- 例如，要立即更改 mydir 目录的几个许可权，请输入:

  ```
  chmod go-w+x mydir
  ```

  按下 Enter 键。

这拒绝（**-**）组成员（**g**）和其它用户（**o**）在 **mydir** 目录中创建或删除文件（**w**）的许可权，并允许（**+**）组成员和其它用户搜索 **mydir** 目录或在路径名中使用（**x**）它。这等价于以下命令序列:

```
chmod g-w mydir
chmod o-w mydir
chmod g+x mydir
chmod o+x mydir
```

- 例如，要仅允许所有者将名为 cmd 的 shell 步骤用作命令，请输入:

  ```
  chmod u=rwx,go= cmd
  ```

  按下 Enter 键。

  这授予拥有文件的用户读、写和执行许可权（**u=rwx**）。它还拒绝组和其它用户以任何方式访问 cmd 的许可权（**go=**）。
- 例如，要使用 **chmod** 命令的数字方式格式来更改 text 文件的许可权，请输入:

  ```
  chmod 644 text
  ```

  按下 Enter 键。

  这为所有者设置读和写许可权，并为组和其它用户设置只读方式。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **chmod** 命令，以获取完整的语法。

访问控制表

访问控制由受保护的信息资源组成，这些信息资源指定可以授权谁访问这样的资源。操作系统允许需要知晓的安全性或任意安全性。信息资源的所有者能授权其它用户读或写该资源的访问权。授权对资源有访问权的用户能将这些权限传送给其它用户。此安全性允许系统中的用户控制的信息流动；信息资源的所有者定义对象的访问许可权。

用户只对他们拥有的对象有基于用户的访问权。通常，用户接收资源的组许可权或缺省许可权。管理访问控制的主要任务是定义用户的组成员资格，因为这些成员资格确定用户对他们不拥有的文件的访问权。

通过添加修改指定给个人和组的基本许可权的扩展许可权来访问控制表（ACL）提高文件访问控制的质量。有了扩展许可权，您不更改基本许可权就可以允许或拒绝对特定个人或组的文件访问。

> **注**: 文件的访问控制表不能超出一个内存页（大约 4096 字节）大小。

要维护访问控制表，请使用 **aclget**、**acledit** 和 **aclput** 命令。

数字方式（使用八进制记数法）的 **chmod** 命令可设置基本许可权和属性。命令调用的 **chmod** 子例程禁用扩展许可权。如果您对具有 ACL 的文件使用 **chmod** 命令的数字方式，则将禁用扩展许可权。**chmod** 命令的符号方式不禁用扩展许可权。有关数字和符号方式的信息，请参阅 **chmod** 命令。

基本许可权

基本许可权是指定给文件所有者、文件组和其它用户的传统文件访问方式。访问方式是: 读（r）、写（w）和执行／搜索（x）。

在访问控制表中，基本许可权为以下格式，并且 *Mode* 参数表达为 rwx（连字符（-）替换每个未指定的许可权）:

```
base permissions:
   owner(name): Mode
   group(group): Mode
   others: Mode
```

属性

可将三个属性添加到访问控制表:

setuid (SUID)
设置用户标识方式位。此属性将进程的有效的和已保存的用户标识设置为在执行文件的所有者标识。

setgid (SGID)
设置组标识方式位。此属性将进程的有效的和已保存的组标识设置为在执行文件的组标识。

savetext (SVTX)
以文本文件格式保存文本。

这些属性以下列格式添加:

```
attributes: SUID, SGID, SVTX
```

扩展许可权

扩展许可权允许文件所有者定义对该文件更精确地访问。扩展许可权通过允许、拒绝或指定特定个人、组或用户和组的组合的访问方式修改基本文件许可权（所有者、组、其它）。通过使用关键字修改许可权。

permit、**deny** 和 **specify** 关键字定义如下:

permit	授权用户或组对文件的指定访问
deny	限制用户或组使用对文件的指定访问
specify	精确地定义用户或组的文件访问

如果通过 **deny** 或 **specify** 关键字拒绝用户的特定访问，则没有其它条目可覆盖此访问拒绝。

必须在 ACL 中指定 **enabled** 关键字，以使扩展许可权生效。缺省值是 **disabled** 关键字。

在 ACL 中，扩展许可权为以下格式:

```
extended permissions:
  enabled | disabled
    permit   Mode  UserInfo...:
    deny     Mode  UserInfo...:
    specify  Mode  UserInfo...:
```

为每个 **permit**、**deny** 或 **specify** 条目使用一个单独行。*Mode* 参数表达为 **rwx**（连字符（-）替换每个未具体说明的许可权）：*UserInfo* 参数表达为 `u:UserName`，或 `g:GroupName`，或 `u:UserName` 和 `g:GroupName` 的逗号分隔的组合。

> **注**：如果在一个条目中指定多个用户名，则该条目不能在访问控制决定中使用，因为一个进程只有一个用户标识。

访问控制表示例

以下是 ACL 的一个示例:

```
attributes: SUID
base permissions:
      owner(frank):  rw-
      group(system): r-x
      others: ---
extended permissions:
      enabled
        permit  rw-  u:dhs
        deny    r--  u:chas, g:system
        specify r--  u:john, g:gateway, g:mail
        permit  rw-  g:account, g:finance
```

ACL 的部件及其含义如下:

- 第一行表示 **setuid** 位已打开。
- 下一行（介绍基本许可权）是可选的。
- 下面三行指定基本许可权。圆括号中的所有者和组名仅是信息。更改这些名称不改变文件所有者或文件组。只有 **chown** 命令和 **chgrp** 命令能更改这些文件属性。
- 下一行（介绍扩展许可权）是可选的。
- 下一行表示启用接下来的扩展许可权。
- 最后四行是扩展条目。第一个扩展条目授权用户 dhs 对文件的读（r）和写（w）许可权。
- 第二个扩展条目只有在用户 chas 是 system 组的成员时，才拒绝对他的读（r）访问。
- 第三个扩展条目指定只要用户 john 同时是 gateway 组和 mail 组的成员时，就有读（r）访问权。如果用户 john 不是两个组的用户，则此扩展许可权不适用。
- 最后一个扩展条目授权 account 组和 finance 组中任何用户*都*具有读（r）和写（w）许可权。

  注: 可以有多个扩展条目应用于一个进程，且限制性方式优先于许可性方式。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **acledit** 命令，以获取完整的语法。

访问权限

信息资源的所有者负责管理访问权。资源受包含在对象方式中的许可权位的保护。许可权位定义授权给对象所有者、对象组和用于 others 缺省类的访问许可权。操作系统支持三种不同的访问方式（读、写和执行），这些方式可被单独授权。

当用户登录帐户（使用 **login** 或 **su** 命令）时，指定给该帐户的用户标识和组标识与用户进程关联。这些标识确定进程的访问权。

对于文件、目录、命名管道和设备（特殊文件），授权访问如下:

- 对于访问控制表（ACL）中的每个访问控制条目（ACE），标识列表与进程的标识进行比较。如果有匹配，则进程接收为该条目定义的许可权和限制。将为 ACL 中每个匹配的条目计算许可权和限制的逻辑联合。如果请求进程不匹配 ACL 中的任何条目，则它接收缺省条目的许可权和限制。
- 如果请求的访问方式是允许的（包含在许可权的联合中）并且不受限制（包含在限制的联合中），则授权访问。否则，拒绝访问。

具有用户标识 0 的进程称为 *root 用户进程*。通常，允许这些进程具有所有访问许可权。但是如果 root 用户进程请求程序的执行许可权，则只有至少授权一个用户执行许可权时才授权访问。

如果 ACL 的标识列表匹配请求进程的有效标识的相应类型，则列表匹配进程。匹配的 USER 类型标识等于进程的有效用户标识，并且如果 GROUP 类型标识等于进程的有效组标识或增补组标识之一，则匹配。例如，一个有标识列表的 ACE 如下所示:

```
USER:fred, GROUP:philosophers, GROUP:software_programmer
```

将匹配有 fred 的有效用户标识的进程和以下的组集:

```
philosophers, philanthropists, software_programmer, doc_design
```

但是，将不匹配有 fred 的有效用户标识的进程和以下的组集:

```
philosophers, iconoclasts, hardware_developer, graphic_design
```

请注意，有以下项的标识列表的 ACE 将同时匹配两个进程:

```
USER:fred, GROUP:philosophers
```

换言之，ACE 函数中的标识列表是一组必须为将授权的指定访问保持的条件。

当第一次访问对象时，在系统调用级别对这些对象进行所有访问许可权检查。由于 System V 内部进程通信（SVIPC）对象是无状态访问的，所以为每个访问进行检查。对于具有文件系统名称的对象，必需能解析实际对象的名称。名称可相对（相对于进程的工作目录）解析或绝对（进程的根目录）解析。所有名称解析通过搜索这些中的一个开始。

任意访问控制机制允许信息资源的有效访问控制，并提供信息的机密性和完整性的单独保护。所有者控制的访问控制机制仅在用户采用时才有效。所有用户必须知道如何授权和拒绝访问许可权，以及如何设置它们。

显示访问控制信息（aclget 命令）

要显示文件的访问控制信息，请使用 **aclget** 命令。您查看的信息包含属性、基本许可权和扩展许可权。

例如，要显示 **status** 文件的访问控制信息，请输入:

```
aclget status
```

按下 Enter 键。显示的访问控制信息包含属性、基本许可权和扩展许可权的列表。有关示例，请参阅第 120 页的『访问控制表示例』。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **aclget** 命令，以获取完整的语法。

设置访问控制信息（aclput 命令）

要为文件设置访问控制信息，请使用 **aclput** 命令。

注: 文件的访问控制表不能超出一个内存页（大约 4096 字节）大小。

例如，要为 **status** 文件设置访问控制信息，并且访问控制信息存储在 acldefs 文件中，请输入:

```
aclput -i acldefs status
```

按下 Enter 键。

例如，要为 **status** 文件设置访问控制信息，并且同样的信息用于 **plans** 文件，请输入:

```
aclget plans | aclput status
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **aclput** 命令，以获取完整的语法。

编辑访问控制信息（acledit 命令）

要更改文件的访问控制信息，请使用 **acledit** 命令。命令显示当前访问控制信息，并让文件所有者更改它。在使任何更改永久前，命令将询问您是否要继续。

注: 必须用完整的路径名指定 **EDITOR** 环境变量；否则，**acledit** 命令将失败。

显示的访问控制信息包含属性、基本许可权和扩展许可权的列表。要获取示例，请参阅第 120 页的『访问控制表示例』。

例如，要编辑 plans 文件的访问控制信息，请输入:

```
acledit plans
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **acledit** 命令，以获取完整的语法。

锁定您的终端（lock 或 xlock 命令）

要锁定您的终端，请使用 **lock** 命令。**lock** 命令请求您的密码，读取它，并再次请求密码以验证它。在这期间，该命令锁定终端并且不放弃它，直到再次接收密码。超时缺省值是 15 分钟，但是可以使用 **-***Number* 标志更改它。

注: 如果您的界面是 AIXwindows，请以同样的方式使用 **xlock** 命令。

例如，要在密码控制下锁定您的终端，请输入:

```
lock
```

按下 Enter 键。将提示您两次输入密码，这样系统能验证它。如果 15 分钟内不重复密码，则命令超时。

要用 10 分钟超时间隔在密码控制下保留终端，请输入:

```
lock -10
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **lock** 或 **xlock** 命令，以获取完整的语法。

文件和系统安全性的命令摘要

acledit	编辑文件的访问控制信息
aclget	显示文件的访问控制信息
aclput	设置文件的访问控制信息
chmod	更改许可权方式
chown	更改与文件关联的用户
lock	保留终端
lsgroup	显示组属性
xlock	锁定本地 X 显示直到输入密码

相关信息

第 26 页的『命令概述』

第 35 页的『进程概述』

第 53 页的『文件系统』

第 56 页的『目录概述』

第 65 页的第 7 章, 『文件』

第 99 页的第 9 章, 『备份文件和存储介质』

第 11 章 定制用户环境

操作系统提供了各种命令和初始化文件，使您能够定制用户环境的行为和外观。

还可以定制在您的系统上使用的应用程序的一些缺省资源。缺省值是在启动时由程序启动的。当更改该缺省值时，您必须退出，然后重新启动程序，以使新缺省值生效。

有关定制“公共桌面环境”行为和外观的信息，请参阅 *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*。

本章讨论以下内容:

系统启动文件概述

登录时，shell 在读取您设置的初始化文件后定义您的用户环境。用户环境的特征由给定环境变量的值定义。维护该环境直到注销系统。

当您登录到操作系统时，shell 使用两种类型的概要文件。它对包含在文件中的命令求值，然后执行命令以设置系统环境。文件具有相似的函数，除 **/etc/profile** 文件控制系统上所有用户的概要文件变量外，然而 **.profile** 文件允许您定制自己的环境。

shell 首先对包含在 **/etc/profile** 文件中的命令求值，然后运行命令以在 **/etc/environment** 文件中设置系统环境。运行这些文件后，系统就检查以查看您是否在主目录中有 **.profile** 文件。如果 **.profile** 文件存在，则它运行该文件。**.profile** 文件将指定是否还有环境文件存在。如果环境文件存在，（通常称为 **.env**），则系统运行此文件，并设置环境变量。

/etc/profile、**/etc/environment** 和 **.profile** 文件在登录时运行一次。另一方面，**.env** 文件在每次您打开新 shell 或窗口时运行。

本节讨论以下初始化文件:

- 『/etc/profile 文件』
- 『/etc/environment 文件』
- 第 127 页的『.profile 文件』
- 第 127 页的『.env 文件』

/etc/profile 文件

操作系统在登录时使用的第一个文件是 **/etc/profile** 文件。此文件控制系统范围的缺省变量，如:

- 导出变量
- 文件创建掩码（umask）
- 终端类型
- 新邮件到达时作为指示的邮件消息

系统管理员配置系统上所有用户的 **profile** 文件。仅系统管理员能更改此文件。

下例是一个典型的 **/etc/profile** 文件:

```
#Set file creation mask
unmask 022
#Tell me when new mail arrives
MAIL=/usr/mail/$LOGNAME
#Add my /bin directory to the shell search sequence
PATH=/usr/bin:/usr/sbin:/etc::
#Set terminal type
TERM=lft
#Make some environment variables global
export MAIL PATH TERM
```

有关 **/etc/profile** 文件的详细信息，请参阅 *AIX 5L Version 5.2 Files Reference*。

/etc/environment 文件

操作系统在登录时使用的第二个文件是 **/etc/environment** 文件。**/etc/environment** 文件包含指定所有进程的基本环境的变量。当新进程开始时，**exec** 子例程产生一个可用的字符串数组，其格式为 *Name*=*Value*。此字符串数组称为*环境*。其中一个字符串定义的每个名称称为一个*环境变量*或 *shell 变量*。**exec** 子例程允许一次设置整个环境。

登录时，系统在读取名为 **.profile** 的登录概要文件前，从 **environment** 文件设置环境变量。以下变量组成基本环境:

HOME	用户登录的全路径名或 **HOME** 目录。**login** 程序设置它为 **/etc/passwd** 文件中指定的名称。
LANG	当前有效的语言环境名称。**LANG** 变量最初在安装时在 **/etc/profile** 文件中设置。
NLSPATH	消息编目的全路径名。
LOCPATH	“本地语言支持表”的位置的全路径名。
PATH	命令，如 **sh**、**time**、**nice** 和 **nohup** 在查找其路径名不完整的命令时搜索的目录顺序。
TZ	时区信息。**TZ** 环境变量最初由系统登录概要文件 **/etc/profile** 文件设置。

请参阅 *AIX 5L Version 5.2 Files Reference*，以获取关于 **/etc/environment** 文件的详细信息。

.profile 文件

操作系统在登录时使用的第三个文件是 **.profile** 文件。**.profile** 文件在您的主目录（**$HOME**）中存在，并使您能够定制个人工作环境。因为 **.profile** 文件是隐藏的，所以请使用 **ls -a** 命令列出它。

在 **login** 程序将 **LOGNAME**（登录名）和 **HOME**（登录目录）变量添加到环境后，执行 **$HOME/.profile** 文件中的命令（如果该文件存在）。**.profile** 文件包含覆盖 **/etc/profile** 文件中变量集的个人概要文件。**.profile** 文件通常用于设置导出的环境变量和终端方式。您可通过修改 **.profile** 文件来定制环境。使用 **.profile** 文件控制以下缺省值:

- 打开的 shell
- 提示符外观
- 键盘声音

下例是一个典型的 **.profile** 文件:

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user::
epath=/home/gsc/e3:
export PATH epath
csh
```

此示例定义了两个路径变量（PATH 和 epath），导出它们，并打开一个 C shell（csh）。

还可以使用 **.profile** 文件（或者，如果它不存在，使用 **/etc/profile** 文件）来确定登录 shell 变量。还可以定制其它 shell 环境。例如，使用 **.cshrc** 文件和 **.kshrc** 文件来分别定制 C shell 和 Korn shell（当每种类型的 shell 启动时）。

.env 文件

操作系统在登录时使用的第四个文件是 **.env** 文件，如果 **.profile** 包含以下行: `export ENV=$HOME/.env`

.env 文件使您能够定制个人工作环境变量。因为 **.env** 文件是隐藏的，所以使用 **ls -a** 命令列出它。**.env** 文件包含覆盖 **/etc/environment** 文件中变量集的个人用户环境变量。可以通过修改 **.env** 文件来按需定制环境变量。

下例是一个典型的 **.env** 文件:

```
export  myid=`id | sed -n -e 's/).*$//' -e 's/^.*(//p'`
#set prompt: login & system name & path
if [ $myid = root ]
        then    typeset -x PSCH='#:\${PWD}> '
                PS1="#:\${PWD}> "
        else    typeset -x PSCH='>'
                PS1="$LOGNAME@$UNAME:\${PWD}> "
                PS2=">"
                PS3="#?"
fi
export PS1 PS2 PS3
#setup my command aliases
alias   ls="/bin/ls -CF" \
        d="/bin/ls -Fal | pg" \
        rm="/bin/rm -i" \
        up="cd .."
```

> **注**: 修改 **.env** 文件时，请确保新创建的环境变量不会与标准变量（如 **MAIL**、**PS1**、**PS2** 和 **IFS**）发生冲突。

AIXwindows 启动文件概述

因为不同的计算机系统有不同方法启动 X 服务器和 AIXwindows，所以咨询您的系统管理员以了解如何进行。通常，X 服务器和 AIXwindows 从您登录时自动运行的 shell 脚本启动。但是，您可能发现需要启动 X 服务器和／或 AIXwindows。

如果您登录并发现您的显示器作为单一终端工作，并且不显示窗口，则可以通过输入以下内容来启动 X 服务器:

```
xinit
```

按下 Enter 键。

如果此命令没有启动 X 服务器，则向您的系统管理员核实，确保您的搜索路径含有包括可执行程序的 X11 目录。一个系统的相应的路径可能不同于另一个系统。

注: 在输入此命令前，确保指针停留在有系统提示符的窗口中。

如果您登录并发现一个或多个窗口没有框架，则可以通过输入以下内容来启动“AIXwindows 窗口管理器”:

```
mwm &
```

按下 Enter 键。

因为 AIXwindows 允许通过程序员写 AIXwindows 应用程序定制和通过用户定制，您可能发现阅读此文档时鼠标按钮或其它功能没有如您期待的那样工作。可以通过按住以下四个键来复位 AIXwindows 环境为缺省行为:

Alt-Ctrl-Shift-!

可以通过再次按此键控顺序返回到定制的行为。如果您的系统不允许此击键组合，则您也可从缺省根菜单恢复缺省行为。

.xinitrc 文件

xinit 命令使用列出 X 客户机程序的可定制 shell 脚本文件来启动。您的主目录中的 **.xinitrc** 文件控制启动 AIXwindows 时启动的窗口和应用程序。

xinit 命令首先查找 **$XINITRC** 环境变量以启动 AIXwindows。如果未找到 **$XINITRC** 环境变量，则它查找 **$HOME/.xinitrc** shell 脚本。如果未找到 **$HOME/.xinitrc** shell 脚本，则 **xinit** 命令启动 **/usr/lib/X11/$LANG/xinitrc** shell 脚本。如果未找到 **/usr/lib/X11/$LANG/xinitrc**，则它查找 **/usr/lpp/X11/defaults /$LANG/xinitrc** shell 脚本。如果未找到该脚本，则它搜索 **/usr/lpp/X11/defaults/xinitrc** shell 脚本。

xinitrc shell 脚本启动命令，如 **mwm**（AIXwindows 窗口管理器）、**aixterm** 和 **xclock** 命令。

xinit 命令执行以下操作:

- 在当前显示器上启动 X 服务器
- 设置 **$DISPLAY** 环境变量
- 运行 **xinitrc** 文件以启动 X 客户机程序

下例显示了可以定制的 **xinitrc** 文件的部分:

```
# This script is invoked by /usr/lpp/X11/bin/xinit

.
```

```
.
.
#*************************************************************
#  Start the X clients. Change the following lines to         *
#  whatever command(s) you desire!                           *
#  The default clients are an analog clock (xclock), an lft  *
#  terminal emulator (aixterm), and the Motif Window  Manager  *
#  (mwm).  *
#*************************************************************
exec mwm
```

.Xdefaults 文件

如果您在 AIXwindows 界面中工作，则可以使用 **.Xdefaults** 文件定制此界面。AIXwindows 允许您为可视特征（如，颜色和字体）指定首选项。

基于 windows 的应用程序的外观和行为的许多方面是受称为 *resources* 的变量集控制的。资源的可视方面或行为方面由其指定的值确定。资源有几种不同类型的值。例如，可指定预定义值给控制颜色的资源，如*暗蓝灰色*或*黑色*。给指定维的资源指定数值。一些资源采取布尔值（*True* 或 *False*）。

如果您的主目录中没有 **.Xdefaults** 文件，则可以使用任何文本编辑器创建一个。您的主目录中有了此文件后，可以根据您的希望在其中设置资源值。称为 **Xdefaults.tmpl** 的样本缺省文件在 **/usr/lpp/X11/defaults** 目录中。

下例显示典型 **.Xdefaults** 文件的部分:

```
*AutoRaise: on
*DeIconifyWarp: on
*warp:on
*TitleFont:andysans12
*scrollBar: true
*font: Rom10.500
Mwm*menu*foreground: black
Mwm*menu*background: CornflowerBlue
Mwm*menu*RootMenu*foreground: black
Mwm*menu*RootMenu*background: CornflowerBlue
Mwm*icon*foreground: grey25
Mwm*icon*background: LightGray
Mwm*foreground: black
Mwm*background: LightSkyBlue
Mwm*bottomShadowColor: Blue1
Mwm*topShadowColor: CornflowerBlue
Mwm*activeForeground: white
Mwm*activeBackground: Blue1
Mwm*activeBottomShadowColor: black
Mwm*activeTopShadowColor: LightSkyBlue
Mwm*border: black
Mwm*highlight:white

aixterm.foreground: green
aixterm.background: black
aixterm.fullcursor: true
aixterm.ScrollKey: on
aixterm.autoRaise: true
aixterm.autoRaiseDelay: 2
aixterm.boldFont:Rom10.500
aixterm.geometry: 80x25
aixterm.iconFont: Rom8.500
aixterm.iconStartup: false
aixterm.jumpScroll: true
aixterm.reverseWrap: true
aixterm.saveLines: 500
aixterm.scrollInput: true
aixterm.scrollKey: false
aixterm.title: AIX
```

.mwmrc 文件

您想要定制的大多数功能可以使用 **.Xdefaults** 文件中的资源设置。但是，密钥绑定、鼠标按钮绑定和窗口管理器的菜单定义都在增补 **.mwmrc** 文件中指定，此文件由 **.Xdefaults** 文件中的资源引用。

如果您的主目录中没有 **.mwmrc** 文件，则可按如下所示复制它:

```
cp /usr/lib/X11/system.mwmrc .mwmrc
```

因为 **.mwmrc** 文件覆盖 **system.mwmrc** 文件的系统范围效果，所以您的规范不会妨碍其它用户的规范。

下例显示典型 **system.mwmrc** 文件的部分:

```
#  DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc)
#
# menu pane descriptions
#
# Root Menu Description

Menu RootMenu
{ "Root Menu"        f.title
  no-label           f.separator
  "New Window"       f.exec "aixterm &"
  "Shuffle Up"       f.circle_up
  "Shuffle Down"     f.circle_down
  "Refresh"          f.refresh
  no-label           f.separator
  "Restart"          f.restart
  "Quit"             f.quit_mwm
}

# Default Window Menu Description

Menu DefaultWindowMenu MwmWindowMenu
{ "Restore"   _R   Alt<Key>F5              f.normalize
  "Move"      _M   Alt<Key>F7              f.move
  "Size"      _S   Alt<Key>F8              f.resize
  "Minimize"  _n   Alt<Key>F9              f.minimize
  "Maximize"  _x   Alt<Key>F10             f.maximize
  "Lower"     _L   Alt<Key>F3              f.lower
  no-label                                 f.separator
  "Close"     _C   Alt<Key>F4              f.kill
}

# no acclerator window menu
Menu NoAccWindowMenu
{
  "Restore"   _R     f.normalize
  "Move"      _M     f.move
  "Size"      _S     f.resize
  "Minimize"  _n     f.minimize
  "Maximize"  _x     f.maximize
  "Lower"     _L     f.lower
  no-label           f.separator
  "Close"     _C     f.kill
}

Keys DefaultKeyBindings
{
  Shift<Key>Escape             icon|window        f.post_wmenu
  Meta<Key>space               icon|window        f.post_wmenu
  Meta<Key>Tab                 root|icon|window   f.next_key
  Meta Shift<Key>Tab           root|icon|window   f.prev_key
  Meta<Key>Escape              root|icon|window   f.next_key
  Meta Shift<Key>Escape        root|icon|window   f.prev_key
  Meta Ctrl Shift<Key>exclam   root|icon|window   f.set_behavior
}
```

```
#
# button binding descriptions
#

Buttons DefaultButtonBindings
{
  <Btn1Down>          frame|icon             f.raise
  <Btn3Down>          frame|icon             f.post_wmenu
  <Btn1Down>          root                   f.menu  RootMenu
  <Btn3Down>          root                   f.menu  RootMenu
  Meta<Btn1Down>      icon|window            f.lower
  Meta<Btn2Down>      window|icon            f.resize
  Meta<Btn3Down>      window                 f.move
}

Buttons PointerButtonBindings
{
  <Btn1Down>         frame|icon             f.raise
  <Btn2Down>         frame|icon             f.post_wmenu
  <Btn3Down>         frame|icon             f.lower
  <Btn1Down>         root                   f.menu  RootMenu
  Meta<Btn2Down>     window|icon            f.resize
  Meta<Btn3Down>     window|icon            f.move
}

#
#  END OF mwm RESOURCE DESCRIPTION FILE
#
```

定制过程

本节讨论以下过程，以定制系统环境:

- 『导出 shell 变量（export shell 命令）』
- 第 132 页的『更改显示的字体（chfont 命令）』
- 第 133 页的『更改控制键（stty 命令）』
- 第 133 页的『更改您的系统提示符』

导出 shell 变量（export shell 命令）

*本地*shell 变量是仅为创建它的 shell 所知的变量。如果启动一个新的 shell，则它不知道旧 shell 的变量。如果您希望打开的新 shell 使用旧 shell 中的变量，请导出变量以使它们成为*全局*的。

可以使用 **export** 命令使本地变量成为全局。要自动使您的本地 shell 变量成为全局，在 **.profile** 文件中导出它们。

注: 变量可向下导出到子 shell，但不能向上导出到父 shell。

例如，要使本地 shell 变量 PATH 成为全局，请输入:

```
export path
```

按下 Enter 键。

例如，要列出所有导出的变量，请输入:

```
export
```

按下 Enter 键。

系统显示类似于以下的信息:

```
DISPLAY=unix:0
EDITOR=vi
ENV=$HOME/.env
HISTFILE=/u/denise/.history
HISTSIZE=500
HOME=/u/denise
LANG=En_US
LOGNAME=denise
MAIL=/usr/mail/denise
MAILCHECK=0
MAILMSG=**YOU HAVE NEW MAIL.
USE THE mail COMMAND TO SEE YOUR MAILPATH=/usr/mail/denise?denise has mail !!!
MAILRECORD=/u/denise/.Outmail
PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/u/bin1
PWD=/u/denise
SHELL=/bin/ksh
```

更改显示的字体（chfont 命令）

要更改系统启动时的缺省字体，请使用 **chfont** 或 **smit** 命令。*字体选用板*是系统用于定义和标识其可用字体的文件。

注: 要运行 **chfont** 命令，您必须有 root 权限。

chfont 命令

例如，要将活动字体更改为字体选用板中的第五个字体，请输入:

```
chfont -a5
```

按下 Enter 键。字体标识 5 成为主字体。

例如，要将字体更改为相同大小的斜体、roman 和粗体字体，请输入:

```
chfont -n /usr/lpp/fonts/It114.snf /usr/lpp/fonts/Bld14.snf  /usr/lpp/fonts/Rom14.snf
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **chfont** 命令，以获取完整的语法。

smit 命令

chfont 命令还可以使用 **smit** 运行。

要选择活动字体，请输入:

```
smit chfont
```

按下 Enter 键。

要选择字体选用板，请输入:

```
smit chfontpl
```

按下 Enter 键。

更改控制键（stty 命令）

要更改您的终端用于控制键的键，请使用 **stty** 命令。您对控制键的更改将持续到您注销为止。要使更改永久，请将它们放置在 **.profile** 文件中。

例如，要将 Ctrl-Z 指定为中断键，请输入:

```
stty intr ^Z
```

确保在 `intr` 和 `^Z` 之间放置空格字符。按下 Enter 键。

例如，要将所有控制键复位成其缺省值，请输入:

```
stty sane
```

按下 Enter 键。

例如，要显示当前设置，请输入:

```
stty -a
```

按下 Enter 键。

请参阅《*AIX 5L V5.2 命令参考大全*》中的 **stty** 命令，以获取完整的语法。

更改您的系统提示符

您的 shell 使用以下提示符变量:

PS1	用作正常系统提示符的提示符
PS2	shell 期待更多输入时使用的提示符
PS3	您具有 root 权限时使用的提示符

可以更改任何提示符字符，方法是更改它的 shell 变量的值。您的提示符更改保持有效，直到您注销。要使更改永久，请将它们放置在您的 **.env** 文件中。

例如，要显示 PS1 变量的当前值，请输入:

```
echo "prompt is $PS1"
```

按下 Enter 键。系统显示类似于以下的信息:

```
prompt is $
```

例如，要将您的提示符更改为 `Ready>`，请输入:

```
PS1="Ready> "
```

按下 Enter 键。

例如，要将您的继续提示符更改为 `Enter more->`，请输入:

```
PS2="Enter more->"
```

按下 Enter 键。

例如，要将您的 root 提示符更改为 `Root->`，请输入:

```
PS3="Root-> "
```

按下 Enter 键。

用户环境定制摘要

系统启动文件

/etc/profile	包含您登录时系统执行的命令的系统文件
/etc/environment	包含指定所有进程的基本环境的变量的系统文件
$HOME/.profile	您的主目录中的文件，包含您登录时覆盖系统 **/etc/profile** 的命令。有关更多信息，请参阅第 127 页的『.profile 文件』
$HOME/.env	您的主目录中的文件，覆盖系统 **/etc/environment**，并包含指定所有进程的基本环境的变量。有关更多信息，请参阅第 127 页的『.env 文件』

AIXwindows 启动文件

$HOME/.xinitrc	您的主目录中的文件，控制启动 AIXwindows 时启动的窗口和应用程序。有关更多信息，请参阅第 128 页的『.xinitrc 文件』。
$HOME/.Xdefaults	您的主目录中的文件，控制 AIXwindows 资源的可视或行为方面。有关更多信息，请参阅第 129 页的『.Xdefaults 文件』。
$HOME/.mwmrc	您的主目录中的文件，定义键绑定、鼠标按钮绑定和您的窗口管理器的菜单定义。有关更多信息，请参阅第 130 页的『.mwmrc 文件』。

定制过程

PS1	正常系统提示符
PS2	更多输入系统提示符
PS3	根系统提示符
chfont	更改系统重新启动时显示使用的字体
stty	设置、复位和报告工作站操作参数

第 12 章 shell

至操作系统的接口称为 *shell*。shell 是操作系统的最外层。shell 合并编程语言以控制进程和文件，以及启动和控制其它程序。shell 通过提示您输入，向操作系统解释该输入，然后处理来自操作系统的任何结果输出来管理您与操作系统之间的交互。

shell 向您提供了与操作系统通信的方式。此通信以交互的方式（来自键盘的输入立即操作）或作为一个 shell 脚本执行。*shell 脚本*是 shell 和操作系统命令的序列，它存储在文件中。

当您登录到系统中时，系统定位要执行的 shell 的名称。在它执行之后，shell 显示一个命令提示符。此提示符通常是一个 $（美元符）。当您在提示符下输入命令并按下 Enter 键时，shell 对命令进行求值，并尝试执行它。取决于您的命令说明，shell 将命令输出写到屏幕或重定向到输出。然后它返回命令提示符，并等待您输入另一个命令。

*命令行*是您输入所在的行。它包含 shell 提示符。每行的基本格式如下:

```
$ 命令参数（一个或多个）
```

shell 视命令行的第一个字（直到第一个空白空格）为命令，所有后继字为自变量。

本章讨论以下内容:

- 第 175 页的『受限 shell』
- 第 176 页的『Bourne shell 命令』
- 第 182 页的『Bourne shell 中的变量和文件名替换』
- 第 187 页的『Bourne shell 中的输入和输出重定向』
- 第 188 页的『C shell』
- 第 189 页的『C shell 命令』
- 第 197 页的『C shell 中的历史替换』
- 第 199 页的『C shell 中的别名替换』
- 第 200 页的『C shell 中的变量和文件名替换』
- 第 203 页的『C shell 中的环境变量』
- 第 204 页的『C shell 中的输入和输出重定向』
- 第 205 页的『C shell 中的作业控制』

shell 功能

通过 shell 与系统交互的主要优点如下:

- **文件名中的通配符替换（模式匹配）**

  通过指定要匹配的模式（而不是实际的文件名）对一组文件执行命令。

  有关更多信息，请参阅以下内容:

  – 第 153 页的『Korn shell 或 POSIX shell 中的文件名替换』
  – 第 186 页的『Bourne shell 中的文件名替换』
  – 第 201 页的『C shell 中的文件名替换』

- **后台处理**

  设置运行时间很长的任务在后台运行，以释放终端用于并行交互式处理。

  有关更多信息，请参阅以下章节中的 **bg** 命令:

  – 第 167 页的『Korn shell 或 POSIX shell 中的作业控制』
  – 第 190 页的『C shell 内置命令』

  注: Bourne shell 不支持作业控制。

- **命令别名创建**

  给命令或短语一个别名。当 shell 在命令行上或 shell 脚本中遇到别名时，它替换别名引用的文本。

  有关更多信息，请参阅以下内容:

  – 第 146 页的『Korn shell 或 POSIX shell 中的命令别名创建』
  – 第 199 页的『C shell 中的别名替换』

  注: Bourne shell 不支持命令别名创建。

- **命令历史**

  将您输入的命令记录在历史文件中。可以使用此文件来轻松地访问、修改和重新发出任何列出的命令。

  有关更多信息，请参阅以下章节中的 **history** 命令:

  – 第 144 页的『Korn shell 或 POSIX shell 命令历史』
  – 第 190 页的『C shell 内置命令』

– 第 197 页的『C shell 中的历史替换』

注: Bourne shell 不支持命令历史。

- **文件名替换**

  使用模式匹配字符在命令行上自动产生文件名的列表。

  有关更多信息，请参阅以下内容:

  – 第 153 页的『Korn shell 或 POSIX shell 中的文件名替换』
  – 第 186 页的『Bourne shell 中的文件名替换』
  – 第 201 页的『C shell 中的文件名替换』

- **输入和输出重定向**

  重定向输入，使不从键盘输入；重定向输出到一个文件或除终端外的设备。例如，程序的输入可以从文件提供，并重定向到打印机或另一个文件。

  有关更多信息，请参阅以下内容:

  – 第 154 页的『Korn shell 或 POSIX shell 中的输入和输出重定向』
  – 第 187 页的『Bourne shell 中的输入和输出重定向』
  – 第 204 页的『C shell 中的输入和输出重定向』

- **管道**

  将任何数目的命令链接在一起以形成复杂的程序。一个程序的标准输出成为下一个程序的标准输入。

  有关更多信息，请参阅第 138 页的『shell 术语』中的**流水线**定义。

- **shell 变量替换**

  存储用户定义的变量和预定义的 shell 变量中的数据。

  有关更多信息，请参阅以下内容:

  – 第 147 页的『Korn shell 或 POSIX shell 中的参数替换』
  – 第 182 页的『Bourne shell 中的变量替换』
  – 第 200 页的『C shell 中的变量替换』

可用的 shell

操作系统提供了以下 shell:

- Korn shell（用 **ksh** 命令启动）
- Bourne shell（用 **bsh** 命令启动）
- 受限 shell（Bourne shell的受限版本，用 **Rsh** 命令启动）
- POSIX shell（也称为 Korn shell，并用 **psh** 命令启动）
- 缺省 shell（用 **sh** 命令启动）
- C shell（用 **csh** 命令启动）
- 可信 shell（Korn shell 的受限版本，用 **tsh** 命令启动）
- 远程 shell（用 **rsh** 命令启动）

*登录 shell*指当您登录到计算机系统时装入的 shell。您的登录 shell 在 **/etc/passwd** 文件中设置。Korn shell 是标准操作系统登录 shell，并向后与 Bourne shell 兼容（请参阅第 174 页的『Bourne shell』）。

*缺省*或*标准 shell*指链接到和用 **/usr/bin/sh** 命令启动的 shell。设置Bourne shell 为缺省 shell，且它是 Korn shell 的子集。

/usr/bin/sh 驻留为 Korn shell 的副本，它是 **/usr/bin/ksh**。因此，可以替换 Korn shell 作为缺省 shell。POSIX shell，由 **/usr/bin/psh** 命令调用，驻留为到 **/usr/bin/sh** 命令的链接。

shell 术语

以下定义在理解 shell 时有帮助:

空白
空白是在 LC_CTYPE 类别中定义的空白字符类中的一个字符。在 POSIX shell 中，空白是制表符或空格。

内置命令
shell 执行的命令，不对其进行搜索和创建一个独立的进程。

命令
shell 语言语法中的字符序列。shell 读每个命令，并直接地或通过调用独立的实用程序来执行期望的操作。

注释
任何以磅符号（#）开始的字。该字和所有它后面的字符，直到下一个换行字符，都被忽略。

标识
可移植字符集中的字母、数字或下划线序列，以字母或下划线开始。标识的第一个字符绝不能是数字。标识用作别名、函数和命名参数的名称。

列表
通过以下符号之一分隔的一个或多个流水线的序列: 分号（;）、和符号（&）、双和符号（&&）或双栏（||）。列表可选地以下列某一符号结束: 分号（;）、和符号（&）或栏和符号（| &）。

; 顺序地处理前面的流水线。shell 依次执行每个命令，并等待最近一个命令完成。

& 异步地处理前面的流水线。shell 依次执行每个命令，在后台处理流水线而不等待它完成。

|& 异步地处理前面的流水线，并建立到父 shell 的双向管道。shell 依次执行每个命令，在后台处理流水线而不等待它完成。父 shell 可以是通过使用 **read -p** 和 **print -p** 命令从产生的命令的标准输入读取或写入其标准输出。在任何给定时刻，只能有一个这样的命令活动。

&& 仅当前面的流水线返回出口值零（0）时，才处理跟在此符号后的列表。

|| 仅当前面的流水线返回非零出口值时，才处理跟在此符号后的列表。

分号（;）、和符号（&）和栏和符号（|&）的优先级比双和符号（&&）和双栏（||）低。;、& 和 |& 符号之间的优先级相等。&& 和 || 符号的优先级相等。可使用一个或多个换行字符而非分号来定界列表中的两个命令。

注: |& 符号仅在 Korn shell 中有效。

元字符
每个元字符都对 shell 有特殊的含义，并导致字终止，除非引证它。元字符是: 管道（|）、和符号（&）、分号（;）、小于号（<）、大于号（>）、左大括号（(）、右大括号（)）、美元符（$）、反引号（`）、反斜杠 (\)、右引号（'）、双引号（"）、换行字符、空格字符和制表符。所有圈起在单引号之间的字符都认为是引证的，并被 shell 逐字地解释。如果没有引证，则将保留元字符的特殊含义。（元字符在 C shell 中也称为*解析器元字符*。）

参数赋值列表
包含一个或多个 *Identifier=Value* 格式的字，其中等号（=）左右的空格必须平衡。即，必须使用前导空白和末尾空白或无空白。

注: 在 C shell 中，参数赋值列表的格式为 **set** *Identifier* = *Value*。需要在等号（=）左右有空格。

流水线
用管道（|）分隔的一个或多个命令的序列。流水线中的每个命令（最后一个命令可能除外），作为独立的进程运行。但是，通过管道连接的每个命令的标准输出成为序列中下一个命令的标准输入。如果列表用圆括号圈起，则它作为在一个独立的子 shell 中操作的简单命令执行。

如果保留字 ！不加在流水线前，则退出状态将为流水线中指定的最后一个命令的退出状态。否则，退出状态是最后一个命令的退出状态的逻辑 “非”。换句话说，如果最后一个命令返回零，则退出状态将为 1。如果最后一个命令返回大于零，则退出状态将为零。

流水线的格式如下:

```
[!] command1 [ | command2 ...]
```

注: Bourne shell 的早期版本使用插入符 (^) 来表明管道。

shell 变量
将值指定给其的名称或参数 。通过输入变量名、等号（=），然后输入值来指定一个变量。通过在变量名前加美元符（$）来使其可替换指定的值。在为长路径名创建短标志法时变量特别有用，如 **$HOME** 表示主目录。预定义的变量是其值由 shell 指定的变量。用户定义的变量是其值由用户指定的变量。

简单命令
任何顺序的可选参数赋值列表和重定向的序列。可选地，后跟命令、字和重定向。它们由 ;、|、&、||、&&、|& 或换行字符终止。命令名作为参数 0 传递（当由 **exec** 子例程定义时）。简单命令的值是它的退出状态零（如果它正常终止）或非零（如果它异常终止）。*AIX 5L Version 5.2 Technical Reference: Base Operating System and Extensions Volume 2* 中的 **sigaction**、**sigvec** 或 **signal** 子例程包含信号退出状态值的列表。

子 shell
作为登录 shell 或当前 shell 的子 shell 运行的 shell。

通配符
也称为*模式匹配字符*。 shell 将它们与指定的值关联。基本通配符是 **?**, *****, [set] 和 [!set]. 通配符在执行文件名替换时特别有用。

字
不包含任何空白的字符序列。字用一个或多个元字符分隔。

创建和运行 shell 脚本

shell 脚本提供简单的方法来执行单调的命令、大的或复杂序列的命令和例程任务。shell 脚本是包含一个或多个命令的文件。当您输入 shell 脚本文件的名称时，系统执行文件包含的命令序列。

可以使用文本编辑器创建一个 shell 脚本。您的脚本可以同时包含操作系统命令和 shell 内置命令。

以下步骤是写 shell 脚本的一般准则:

1. 使用文本编辑器，创建和保存文件。可以在 shell 脚本文件中包含任何 shell 和操作系统命令的组合。按照约定，不为许多用户设置使用的 shell 脚本存储在 **$HOME/bin** 目录中。

   注: 操作系统不支持 shell 脚本中的 **setuid** 或 **setgid** 子例程。

2. 使用 **chmod** 命令仅允许所有者运行（或执行）该文件。例如，如果将您的文件名为 `script1`，请输入:

   ```
   chmod u=rwx script1
   ```

   按下 Enter 键。

3. 在命令行上输入脚本名称以运行 shell 脚本。要运行 **script1** shell 脚本，请输入:

   ```
   script1
   ```

   按下 Enter 键。

   注: 如果在命令行上在 shell 脚本文件名前输入 shell 命令（**ksh**、**bsh** 或 **csh**），可以不使用 shell 脚本可执行就能运行它。例如，要在 Korn shell 的控制下运行名为 **script1** 的非可执行文件，请输入:

   ```
   ksh script1
   ```

指定脚本文件的 shell

当您在 Korn（POSIX shell）或 Bourne shell 中运行可执行文件 shell 脚本时，脚本中的命令在当前 shell（启动脚本的 shell）控制下执行，除非您指定不同的 shell。当您在 C shell 中运行可执行文件的 shell 脚本时，脚本中的命令在 Bourne shell（**/usr/bin/bsh**）的控制下执行，除非您指定不同的 shell。

通过将 shell 包含在 shell 脚本中，可以在特定 shell 中运行 shell 脚本。

要在特定 shell 下运行一个可执行文件 shell 脚本，在 shell 脚本的第一行上输入 `#!Path`，并按下 Enter 键。`#!` 字符标识文件类型。*Path* 变量指定运行 shell 脚本的 shell 的路径名。

例如，要在 Bourne shell 中运行 `bsh` 脚本，请输入:

```
#!/usr/bin/bsh
```

按下 Enter 键。

当您在 shell 脚本文件名前输入一个 shell 命令，则在命令行上指定的 shell 将覆盖在脚本文件本身中指定的任何 shell。因此，请输入 `ksh myfile` 并按下 Enter 键在 Korn shell 控制下运行名为 **myfile** 的文件，即使 **myfile** 的第一行是 `#!/usr/bin/csh`。

Korn shell 或 POSIX shell 命令

Korn shell 是一个交互式命令解释器和命令编程语言。它符合计算机环境的可移植操作系统接口（POSIX），一种操作系统的国际标准。POSIX 不是一个操作系统，而是一种*标准*，它针对源代码级别的、很多系统间的应用程序的可移植性。POSIX 功能构建在 Korn shell 之上。Korn shell（也称为 POSIX shell）提供很多与 Bourne shell 和 C shell 相同的功能，如 I/O 重定向能力、变量替换以及文件名替换。它还包含几个其它命令和编程语言功能:

算术求值
Korn shell 或 POSIX shell 可以通过使用内置 **let** 命令（使用任何从 2 到 36 的基数）执行整数算术。第 151 页的『Korn shell 或 POSIX shell 中的算术求值』进一步描述此功能。

命令历史
Korn shell 或 POSIX shell 存储记录您输入的所有命令的文件。可以使用文本编辑器来改变此历史文件中的命令，然后重新发出该命令。有关命令历史功能的更多信息，请参阅第 144 页的『Korn shell 或 POSIX shell 命令历史』。

联合进程设施
使您能在后台运行程序，并发送信息到这些后台进程，及从这些后台进程接收信息。有关更多信息，请参阅第 155 页的『联合进程设施』。

编辑
Korn shell 或 POSIX shell 提供直接插入编辑选项，使您能编辑命令行。类似 emacs、gmacs 和 vi 的编辑器是可用的。第 168 页的『Korn shell 或 POSIX shell 中的直接插入编辑』进一步描述此功能。

Korn shell 命令是以下之一:

- 简单命令
- 流水线
- 列表
- 复合命令
- 函数

当您在 Korn shell 或 POSIX shell 中发出命令时，shell 对命令求值，并执行以下操作:

- 进行所有指示的替换。

- 确定命令是否包含 /。如果是，则 shell 运行由指定的路径名指定的程序。

如果命令不包含 /，则 Korn shell 或 POSIX shell 继续以下操作:

- 确定命令是否是特殊内置命令。如果是，则 shell 在当前 shell 进程中运行命令。

  有关特殊内置命令的信息，请参阅 “第 157 页的『Korn shell 或 POSIX shell 内置命令』”。
- 将命令与用户定义的函数作比较。如果命令与用户定义的函数匹配，则保存位置参数，然后复位为**函数**调用的自变量。当函数完成或发出返回时，位置参数列表恢复，并且执行函数中 **EXIT** 上设置的任何陷阱。函数的值是上次执行命令的值。函数在当前 shell 进程中执行。
- 如果命令名匹配常规内置命令的名称，则将调用该常规内置命令。

  有关常规内置命令的信息，请参阅 “第 157 页的『Korn shell 或 POSIX shell 内置命令』”。
- 创建进程并尝试通过使用 **exec** 命令执行命令（如果命令既不是内置命令也不是用户定义的函数）。

Korn shell 或 POSIX shell 搜索指定路径中的每个目录，以查找可执行文件。**PATH** shell 变量定义包含命令的目录的搜索路径。备用目录名称使用 : 分隔。缺省路径是 **/usr/bin:**（指定 **/usr/bin** 目录和当前目录，以该顺序）。当前目录由两个或多个相邻的冒号，或在路径列表的开始或结尾的冒号指定。

如果文件具有执行许可权，但不是目录或 **a.out** 文件，则 shell 假定它包含 shell 命令。当前 shell 进程产生一个子 shell 来读文件。所有非导出的别名、函数和指定的参数从文件中除去。如果 shell 命令文件有读许可权，或者如果已在文件上设置了 **setuid** 或 **setgid** 位，则 shell 运行代理程序，该代理程序设置许可权并且使用向下传递为打开文件的 shell命令文件执行 shell。加圆括号的命令在子 shell 中运行，而不除去非导出的数量。

本节讨论以下内容:

- 『Korn shell 复合命令』
- 第 143 页的『Korn shell 函数』
- 第 157 页的『Korn shell 或 POSIX shell 内置命令』
- 第 166 页的『Korn shell 或 POSIX shell 的条件表达式』

Korn shell 复合命令

复合命令可以是简单命令和流水线的列表，或者它可以用保留字开始。大多数时候，将在编写 shell 脚本时使用复合命令，如 **if**、**while** 和 **for**。

Korn shell 或 POSIX shell 复合命令的列表

for *Identifier* [**in** *Word* ...] **;do** *List* **;done**	每次执行 **for** 命令时，*Identifier* 参数设置为下一个取自 **in** *Word* ... 列表的字。如果省略 **in** *Word* ... 命令，则 **for** 命令对于每个设置的位置参数执行一次 **do** *List* 命令。当列表中不再有字时执行结束。有关位置参数的更多信息，请参阅 “第 147 页的『Korn shell 或 POSIX shell 中的参数替换』”。
select *Identifier* [**in** *Word* ...] **;do** *List* **;done**	**select** 命令将指定的字集合打印在标准错误（文件描述符 2）上，每个字前有一个数字。如果省略 **in** *Word* ... 命令，则改为使用位置参数。打印 **PS3** 提示符，并从标准输入读取行。如果此行由一个列出的字的编号组成，则 *Identifier* 参数的值设置为对应于此编号的字。 如果从标准输入读取的行是空的，则再次打印选择列表。否则，*Identifier* 参数的值设置为空。从标准输入读取的行的内容保存在 **REPLY** 参数中。为每个选择执行 *List* 参数，直到遇到中断或文件结束符字符。有关位置参数的更多信息，请参阅 “第 147 页的『Korn shell 或 POSIX shell 中的参数替换』”。
case *Word* **in** [[**(**] *Pattern* [| *Pattern*] ... **)** *List* **;;**] ... **esac**	**case** 命令执行与匹配 *Word* 参数的第一个 *Pattern* 参数关联的 *List* 参数。模式的格式与用于文件名替换的格式相同。

if *List* **;then** *List* [**elif** *List* **;then** *List*] ... [**;else** *List*] **;fi**	*List* 参数指定要运行的命令列表。shell 首先执行 **if** *List* 命令。如果返回零退出状态，则执行 **then** *List* 命令。否则，执行由跟在 **elif** 命令后的 *List* 参数指定的命令。 如果由 **elif** *List* 命令中最后一个命令返回的值是零，则执行 **then** *List* 命令。如果由 **then** *List* 命令中最后一个命令返回的值是零，则执行 **else** *List* 命令。如果没有由 *List* 参数指定的命令为 **else** 或 **then** 命令执行，则 **if** 命令返回零退出状态。
while *List* **;do** *List* **;done** **until** *List* **;do** *List* **;done**	*List* 参数指定要运行的命令列表。**while** 命令重复地执行由 *List* 参数指定的命令。如果 **while** *List* 命令中最后命令的退出状态是零，则执行 **do** *List* 命令。如果 **while** *List* 命令中最后一个命令的退出状态非零，则循环终止。如果未执行 **do** *List* 中的命令，则 **while** 命令返回零退出状态。可能使用 **until** 命令代替 **while** 命令来否定循环终止测试。
(*List***)**	*List* 参数指定要运行的命令列表。shell 在独立的环境中执行 *List* 参数。 **注**: 如果需要两个相邻的左圆括号用于嵌套，则必须在它们之间插入一个空格，以区别命令与算术求值。
{ *List***;}**	*List* 参数指定要运行的命令列表。简单地执行 *List* 参数。 **注**: 不同于元字符 ( ), { } 表示保留字（用于特殊目的，不作为用户声明的标识）。要识别这些保留字，必须令其出现在一行的开始或在一个 ; 后。
[[*Expression*]]	求值 *Expression* 参数。如果表达式为真，则命令返回零退出状态。
function *Identifier* **{** *List* **;}** 或 **function** *Identifier* **() {***List* **;}**	定义由 *Identifier* 参数引用的函数。函数的主体是由 { } 圈起的指定的命令列表。() 由两个运算符组成，因此将空白字符与 *identifier*、(和) 混在一起是允许的，但不是必需的。
time *Pipeline*	执行 *Pipeline* 参数。耗用时间、用户时间和系统时间打印到标准错误。

shell 启动

可以使用 **ksh** 命令、**psh** 命令（POSIX shell）或 **exec** 命令启动 Korn shell。

如果 shell 由 **exec** 命令启动，并且零自变量（**$0**）的第一个字符是连字符（-），则假定该 shell 为一个登录 shell。 shell 首先从 **/etc/profile** 文件读取命令，然后从当前目录中的 **.profile** 文件或从 **$HOME/.profile** 文件（如果任一文件存在）读取命令。接下来，shell 从对 **ENV** 环境的值执行参数替换所指定的文件（如果文件存在）读取命令。

如果在调用 Korn shell 或 POSIX shell 时指定 *File* [*Parameter*] 参数，则 shell 运行由 *File* 参数标识的脚本文件，包含任何指定的参数。指定的脚本文件必须具有读许可权；忽略任何 **setuid** 和 **setgid** 设置。然后 shell 读取命令。

注: 当调用 Korn shell 或 POSIX shell 时，不要使用 **-c** 或 **-s** 标志指定脚本文件。

有关位置参数的更多信息，请参阅第 147 页的『Korn shell 或 POSIX shell 中的参数替换』。

Korn shell 环境

在命令执行的开始时为命令知晓的所有变量（及其关联的值）组成其*环境*。此环境包含命令从其父进程继承的变量和在调用命令的命令行上作为关键字参数指定的变量。shell 以几种方法与环境交互。当它启动时，shell 扫描环境并为每个找到的名称创建参数，给参数相应的值并标记它以用于导出。执行的命令继承该环境。

如果使用 **export** 或 **typeset -x** 命令修改 shell 参数的值或创建新值，则参数成为环境的一部分。因此任何已执行命令看到的环境由最初由 shell 继承的 “名称-值” 对（其值可能被当前 shell 修改过）加上由使用 **export** 或 **typeset -x** 命令产生的任何附加值组成。已执行的命令（子 shell）将看到它对所继承环境变量进行的任何修改，但为了使其子 shell 或进程看到修改的值，子 shell 必须导出这些变量。

任何简单命令或函数的环境都通过在前面加一个或多个参数赋值来更改。参数赋值自变量是 *Identifier=Value* 格式的字。因此，以下两个表达式是等价的（就命令的执行而言）：

```
TERM=450 Command arguments
(export TERM; TERM=450; Command arguments)
```

Korn shell 函数

函数保留字定义 shell 函数。shell 以内部方式读和存储函数。别名在读取函数时解析。shell 以与执行命令相同的方式执行函数，其中自变量作为位置参数传递。有关位置参数的更多信息，请参阅第 147 页的『Korn shell 或 POSIX shell 中的参数替换』。

Korn shell 或 POSIX shell 在函数从中调用的环境中执行函数。所有以下内容由函数和调用脚本共享，并可产生副作用:

- 变量值和属性（除非您在函数中使用 **typeset** 命令以声明一个本地变量）
- 工作目录
- 别名、函数定义和属性
- 特殊参数 $
- 打开文件

以下内容不在函数和调用脚本之间共享，并没有副作用:

- 位置参数。
- 特殊参数 #。
- 调用函数时变量赋值列表中的变量。
- 使用 **typeset** 命令在函数中声明的变量。
- 选项。
- 陷阱。但是，被调用脚本忽略的信号也将被函数忽略。

> **注**: 在 Korn shell 的早期版本中，除 **EXIT** 和 **ERR** 外的陷阱由函数和调用脚本共享。

如果 **0** 或 **EXIT** 上的陷阱在函数的主体内执行，则在函数完成后，在调用函数的环境中执行操作。如果陷阱在函数的主体外执行，则在从 Korn shell 退出时执行操作。在 Korn shell 的早期版本中，函数的主体外的 **0** 或 **EXIT** 上的陷阱不在从函数退出时执行。

当函数执行时，它有“第 157 页的『Korn shell 或 POSIX shell 内置命令』”中描述的相同的语法错误和变量赋值属性。

无论何时将函数名指定为简单命令的名称，请执行复合命令。命令的操作数将在复合命令的执行期间暂时成为位置参数。特殊参数 # 也将更改以反映操作数的号码。特殊参数 0 将不更改。

return 特殊命令用于从函数调用返回。函数中的错误将控制返回给调用程序。

函数标识使用 **typeset** 特殊命令的 **-f** 或 **+f** 选项列出。**-f** 选项还列出函数的文本。函数用 **unset** 特殊命令的 **-f** 选项取消定义。

函数一般在 shell 执行 shell 脚本时取消设置。**typeset** 特殊命令的 **-xf** 选项允许函数导出到脚本，这些脚本的执行没有 shell 的独立调用。必须在 shell 的独立调用间定义的函数应该在 **ENV** 文件中使用 **typeset** 特殊命令的 **-xf** 选项定义。

如果函数没有成功声明，则该函数定义的退出状态为零。否则，它将大于零。函数调用的退出状态是由函数执行的最近命令的退出状态。

Korn shell 或 POSIX shell 命令历史

Korn shell 或 POSIX shell 将从您的终端设备输入的命令保存到历史文件。如果设置，**HISTFILE** 变量值是历史文件的名称。如果 **HISTFILE** 变量未设置或无法写，则使用的历史文件是 **$HOME/.sh_history**。如果历史文件不存在，并且 Korn shell 无法创建它，或者它存在但 Korn shell 不具有附加到它的许可权，则 Korn shell 使用临时文件作为历史文件。shell 使用具有相应的许可权的同一个历史文件来访问所有交互式 shell 的命令。

缺省情况下，Korn shell 或 POSIX shell 保存从终端设备输入的最后 128 个命令的文本。历史文件大小（由 **HISTSIZE** 变量指定）是没有限制的，尽管非常大的历史文件可能导致 Korn shell 启动很慢。

命令历史替换

使用 **fc** 内置命令列出或编辑历史文件的某些部分。要选择文件的一部分编辑或列出，请指定命令的号码、第一个字符或头几个字符。可以指定单个命令或某一范围的命令。

如果不将编辑器程序指定为 **fc** 常规 shell 内置命令的自变量，则使用由 **FCEDIT** 变量指定的编辑器。如果未定义 **FCEDIT** 变量，则使用 **/usr/bin/ed** 文件。当您退出编辑器时，已编辑的一个或多个命令打印并运行。

编辑器名称连字符（-）用于跳过编辑阶段，并再次运行命令。在此情况下，可使用格式为 *Old=New* 的替换参数在运行一个命令前修改它。例如，如果 `r` 是 `fc -e -` 的别名，则输入 `r bad=good c` 运行以字母 *c* 开始的最近一个命令，并将第一个出现的 `bad` 字符串替换为 `good` 字符串。

有关使用历史 shell 命令的更多信息，请参阅第 30 页的『列出先前输入的命令（history shell 命令）』和《*AIX 5L V5.2 命令参考大全*》中的 **fc** 命令。

Korn shell 或 POSIX shell 中的引证

当想要 Korn shell 或 POSIX shell 将字符作为常规字符（而没有任何通常关联的含义）读时，必须引证它。要取消元字符的特殊含义，请使用下表的其中一个引证机制。

每个元字符对 shell 都有特殊含义（除非已引证）并导致字终止。Korn shell 或 POSIX shell 认为以下字符是元字符，并且如果它们代表自己就必须已引证。

- 管道（|）
- 和符号（&）
- 分号（;）
- 小于号（$lt;）和大于号（>）
- 左圆括号（(）和右圆括号（)）
- 美元符（$）
- 反引号（`）和单引号（’）
- 反斜杠（\）
- 双引号（″）
- 换行字符
- 空格字符
- 制表符

引证机制是反斜杠 (\)、单引号（'）和双引号（″）。

反斜杠

不引证的反斜杠 (\) 保留后面字符的文字值，除换行外。如果换行字符跟着反斜杠，则 shell 将它解释为行继续。

单引号

圈起在单引号（ ' '）中的字符保留单引号中的每个字符的文字值。单引号不能出现在单引号中。

反斜杠不能用于转义在单引号中设置的字符串的单引号。嵌入的引号可通过写创建，例如: 'a'\''b' 产生 a'b。

双引号

在双引号（″ ″）中圈起字符保留双引号中的所有字符的文字值，除美元符、反引号和反斜杠字符外，如下:

$ 美元符保留其特殊含义: 引入参数扩展，命令替换的格式和算术扩展。

在引证字符串中也以 $(和匹配的) 圈起的输入字符将不受双引号的影响，但定义当展开字时其输出替换 $(...) 的命令。

从圈起的 ${ 到匹配的 } 的字符串中，非转义的双引号或单引号必须是偶数个（如果有）。前面的反斜杠字符必须用于转义文字 { 或 }.

**`** 反引号保留其特殊含义，引入命令替换的另一格式。引证字符串的一部分，从初始的反引号和字符直到下一个前面没有反斜杠的反引号，定义当字展开时其输出替换 ` ... ` 的命令。

**** 仅当后跟以下字符之一时，反斜杠保留其特殊含义（作为一个转义字符）: $, `, ″, \ 或换行字符。

双引号前必须有一个反斜杠，以包含在双引号中。当您使用双引号时，如果反斜杠后紧跟着将被解释为有特殊含义的字符，则删除反斜杠，且逐字取后继字符。如果反斜杠不在具有特殊含义的字符之前，则它保留在适当的位置不更改，且紧跟着它的字符也保留不更改。例如:

```
"\$"    ->      $
"\a"    ->      \a
```

以下条件应用于 Korn 或 POSIX shell 中的元字符和引证字符:

- 当美元符、星号（$*）和美元符, @ 符 ($@) 不被引证时，它们的含义在作为参数赋值使用时或用作文件名时是相同的。
- 当用作命令自变量时，双引号、美元符、星号、双引号（″$*″）等价于 ″$1*d*$2*d*...″，其中 *d* 是 IFS 参数的第一个字符。
- 双引号, @ 符, 星号, 双引号（″$@″）等价于 ″$1″ ″$2″ ...。
- 内反斜杠（``）、反斜杠引证字符反斜杠 (\)、单引号（'）和美元符（$）。如果反引号在双引号（″ ″）中出现，则反斜杠也引证双引号字符。
- 参数和命令替换发生在双引号（″ ″）内。
- 通过引证保留字的任何字符除去保留字或别名的特殊含义。不能引证函数名或内置命令名。

Korn shell 或 POSIX shell 中的保留字

以下保留字对 shell 有特殊含义:

```
!       case    do
done    elif    else
esac    fi      for
```

```
function  if      in
select    then    time
until     while   {
}         [[      ]]
```

保留字仅在它们不带引号出现时，以及字用作以下情况时才可识别:

- 命令的第一个字
- 跟在除 **case**、**for** 或 **in** 之外的某一保留字后的第一个字
- **case** 或 **for** 命令中的第三个字（在这种情况下，只有 **in** 有效）

Korn shell 或 POSIX shell 中的命令别名创建

Korn shell 或 POSIX shell 允许您创建别名以定制命令。**alias** 命令定义 `Name=String` 格式的字作为别名。当将别名用作命令行的第一个字时，Korn shell 检查以查看它是否已经正在处理具有相同名称的别名。如果是这样，Korn shell 不替换别名。如果具有相同名称的别名不在处理中，则 Korn shell 使用别名的值替换别名。

别名的第一个字符可以是除元字符外的任何可打印字符。其余字符必须与对于有效标识的要求相同。替换字符串可以包含任何有效的 shell 文本，包括元字符。

如果别名值的最后一个字符是空格，则 shell 还为别名替换检查别名后的字。可以使用别名来重新定义特殊的内置命令，但不要重新定义保留字。别名定义不在 **ksh** 的调用间继承。但是，如果指定 **alias -x**，则对于根据名称调用和不调用独立 shell 的脚本，别名仍然有效。要导出别名定义并使子进程对其有访问权，则必须指定 **alias -x**，以及环境文件中的别名定义。

要创建、列出和导出别名，请使用 **alias** 命令。要除去别名，请使用 **unalias** 命令。

创建别名的格式如下:

```
alias Name=String
```

其中 *Name* 参数指定别名的名称，*String* 参数指定别名的值。

以下导出的别名是 Korn shell 预定义的，但可以取消设置或重新定义。不建议您更改它们，因为这可能在以后给期望别名以 Korn shell 预定义方式工作的用户造成混淆。

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
nohup='nohup '
r='fc -e -'
true=':'
type='whence -v'
```

在 Korn shell（ksh）的非交互式调用上不支持别名；例如，在 shell 脚本中，或在 **ksh** 中使用 **-c** 选项，如下所示:

```
ksh -c alias
```

有关别名创建的更多信息，请参阅第 33 页的『创建命令别名（alias shell 命令）』和《*AIX 5L V5.2 命令参考大全*》中的 **alias** 命令。

跟踪的别名

别名经常用作全路径名的速记。一个别名创建设施选项允许您将别名的值自动设置为对应命令的全路径名。这种特殊类型的别名是*跟踪的*别名。通过省去 shell 搜索 **PATH** 变量以查找全路径名的需要，磁道的别名提高了执行速度。

set -h 命令打开命令*跟踪*，这样每次引用命令，shell 将定义跟踪的别名的值。每次您复位 **PATH** 变量，此值取消定义。

这些别名保留跟踪，这样下一个后继引用将重新定义值。几个跟踪的别名编译到 shell 中。

代字号替换

在 shell 执行别名替换后，它检查每个字以查看它是否以非引证的代字号 (~) 开始。如果是，则 shell 检查该字（直到第一个斜杠（/））以查看它是否与 **/etc/passwd** 文件中的用户名匹配。如果 shell 找到匹配，则它用匹配用户的登录目录替换 ~ 字符和名称。此过程称为*代字号替换*。

如果 shell 未找到匹配，则不更改原始文本。如果 ~ 字符是字中的唯一字符，或后跟加号（+）或连字符（-），则 Korn shell 还进行特殊的替换:

~ 由 **HOME** 变量的值替换。
~+ 由 **$PWD** 变量（当前目录的全路径名）替换。
~- 由 **$OLDPWD** 变量（前一目录的全路径名）替换。

此外，当变量赋值参数的值以代字号 **~** 字符开始时，shell 将尝试代字号替换。

Korn shell 或 POSIX shell 中的参数替换

Korn shell 或 POSIX shell 使您能够进行参数替换。

本节讨论以下内容:

Korn shell 中的参数

参数定义为以下:

- 任何字符星号（*），@ 符 (@), 磅符号（#），问号（?），连字符（-），美元符（$）和感叹号（!）的标识。这些称为*特殊参数*。
- 数（*位置参数*）指示的自变量
- 标识指示的参数，具有一个值及零个或多个属性（*命名的参数／变量*）。

typeset 特殊内置命令将值和属性指定给命名参数。使用 **typeset** 特殊内置命令描述 Korn shell 支持的属性。导出的参数将值和属性传递到环境。

命名参数的值通过如下指定:

```
Name=Value [ Name=Value ] ...
```

如果为 *Name* 参数设置了 **-i** 整数属性，则 *Value* 参数服从于算术求值。有关算术表达式求值的更多信息，请参阅第 151 页的『Korn shell 或 POSIX shell 中的算术求值』。

shell 支持一维的数组程序。数组参数的元素通过下标引用。下标由用方括号 ([ ]) 圈起的算术表达式指示。要将值指定给数组，使用 `set -A Name Value ...`。所有下标的值必须在 0 到 511 范围内。数组不需要声明。任何对带有效下标的命名参数的引用都是合法的，且将创建一个数组（如有必要）。引用没有下标的数组等价于引用元素 0。

用 **set** 特殊命令为位置参数指定值。当调用 shell 时从自变量 0 设置 **$0** 参数。$ 字符用于引入可由数组替换的参数。

参数替换

以下是可替换的参数:

${*Parameter***}**	shell 读取从 **${** 到匹配的 **}** 之间的所有字符，作为同一个字的部分，即使该字包含大括号或元字符。如果有，替换指定的参数的值。以下情况下大括号是必需的：当 *Parameter* 参数后跟一个不被解释为其名称一部分的字母、数字或下划线时，或当一个命名的参数进行下标操作时。 如果指定的参数包含一个或多个数字，则它是一个位置参数。多个数字的位置参数必须圈起在大括号中。如果变量的值是 ***** 或 **@**，则替换每个以 **$1** 开始的位置参数（由字段分隔符字符分隔）。如果使用了一个带下标 ***** 或 **@** 的数组标识，则替换每个元素的值（用字段分隔符字符分隔）。
${#*Parameter***}**	如果 *Parameter* 参数的值是 * 或 @，则替换位置参数的数。否则，替换 *Parameter* 参数指定的长度。
${#*Identifier***[*]}**	替换 *Identifier* 参数指定的数组中的元素数。
${*Parameter***:-***Word***}**	如果设置了 *Parameter* 参数并且不为空，则替换其值；否则，替换 *Word* 参数的值。
${*Parameter***:=***Word***}**	如果未设置 *Parameter* 参数或其为空，则将它设置为 *Word* 参数的值。不能用这种方法指定位置参数。
${*Parameter***:?***Word***}**	如果设置了 *Parameter* 参数并且它不为空，则替换其值。否则，打印 *Word* 变量的值并从 shell 退出。如果省略 *Word* 变量，则打印标准消息。
${*Parameter***:+***Word***}**	如果设置了 *Parameter* 参数并且它不为空，则替换 *Word* 变量的值。
${***Parameter***#***Pattern***}** \| **${*Parameter***##***Pattern***}**	如果指定的 shell *Pattern* 参数匹配 *Parameter* 参数的值的开始部分，则此替换的值是 *Parameter* 参数删除匹配部分后的值。否则，替换 *Parameter* 参数的值。在第一种格式中，删除最小匹配模式。在第二种格式中，删除最大匹配模式。
${***Parameter***%***Pattern***}** \| **${*Parameter***%%***Pattern***}**	如果指定的 shell *Pattern* 匹配 *Parameter* 变量的值的结束部分，则此替换的值是 *Parameter* 变量删除匹配部分后的值。否则，替换 *Parameter* 变量的值。在第一种格式中，删除最小匹配模式；在第二种格式中，删除最大匹配模式。 在先前的表达式中，*Word* 变量不被求值，除非它用作替换的字符串。这样，在下例中，**pwd** 命令只在没有设置 **-d** 标志或它为空时才执行: `echo ${d:-$(pwd)}`

注: 如果 **:** 从先前表达式中省略，则 shell 仅检查是否设置了 *Parameter* 参数。

预定义的特殊参数

以下参数由 shell 自动设置:

@ 展开位置参数，以 **$1** 开始。每个参数用一个空格分隔。

如果用 ″ 圈起 **$@**，则 shell 认为每个位置参数是独立的字符串。如果不存在位置参数，则 shell 将语句展开成没有引号的空字符串。

***** 展开位置参数，以 **$1** 开始。shell 使用 **IFS** 参数值的第一个字符分隔每个参数。

如果用 ″ 圈起 **$***，则 shell 在双引号中包含位置参数值。每个值由 **IFS** 参数的第一个字符分隔。

指定传递给 shell 的位置参数的数目（十进制），不计数 shell 步骤名称本身。这样 **$#** 参数产生设置的最大编号位置参数的数目。此参数的一个主要使用是检查存在需要的自变量数目。

- 调用时或使用 **set** 命令向 shell 提供标志。

? 指定上次执行命令的出口值。其值是一个十进制字符串。大多数命令返回 0 以表明成功完成。 shell 自己返回 **$?** 参数的当前值作为其出口值。

$ 标识 shell 的进程号。由于进程号在所有现有字符串中唯一，所以最多 5 个数字的此字符串经常用于生成临时文件的唯一名称。

下例说明仅用于该目的的，在目录中创建临时文件的推荐示例:

```
temp=$HOME/temp/$$
ls >$temp
.
.
.
rm $temp
```

! 指定最新调用的后台命令的进程号。

零（0） 展开 shell 或 shell 脚本名称。

Korn shell 或 POSIX shell 设置的变量

以下变量由 shell 设置:

下划线（_） 表示当在环境中传递时所执行的 shell 或脚本的最初绝对路径名。随后，指定前一命令的最后一个自变量。此参数不为异步命令设置。此参数还用于保留检查邮件时匹配 **MAIL** 文件的名称。

ERRNO 指定最新失败的子例程设置的值。此值是系统相关的，并旨在调试目的。

LINENO 指定执行的脚本或函数中的当前行的行号。

OLDPWD 表示 **cd** 命令设置的前一工作目录。

OPTARG 指定 **getopts** 常规内置命令处理的最后一个选项自变量的值。

OPTIND 指定 **getopts** 常规内置命令处理的最后一个选项自变量的索引。

PPID 标识父 shell 的进程号。

PWD 表示 **cd** 命令设置的当前工作目录。

RANDOM 生成随机整数，均一地分布在 0 和 32767 之间。随机数的序列可由指定数值给 **RANDOM** 变量初始化。

REPLY 没有提供自变量时，由 **select** 语句和 **read** 常规内置命令设置。

SECONDS 指定返回 shell 调用以来的秒数。如果为此变量指定一个值，则返回引用的值将是被指定的值加指定以来的秒数。

Korn shell 或 POSIX shell 使用的变量

以下变量由 shell 使用:

CDPATH 表示 **cd**（更改目录）命令的搜索路径。

COLUMNS	为 shell 编辑方式和打印 **select** 列表定义编辑窗口的宽度。
EDITOR	如果此参数的值以 emacs、gmacs 或 vi 结束，并且 **VISUAL** 变量不是使用 **set** 特殊内置命令设置的，则对应选项打开。
ENV	如果设置了此变量，则对值执行参数替换以生成当调用 shell 时将执行的脚本的路径名。该文件通常用于别名和函数定义。
FCEDIT	为 **fc** 常规内置命令指定缺省编辑器名称。
FPATH	指定函数定义的搜索路径。当引用具有 **-u** 标志的函数时和未找到命令时，将搜索此路径。如果找到可执行文件，则在读和在当前环境中读和执行它。
HISTFILE	如果在调用 shell 时设置此变量，则值是将用于存储命令历史的文件的路径名。
HISTSIZE	如果在调用 shell 时设置此变量，则先前输入的 shell 可访问的命令的数将大于或等于该数。缺省值是 128。
HOME	表示您的登录目录的名称，该目录在登录完成后成为当前目录。**登录**程序初始化此变量。**cd** 命令将 **$HOME** 参数的值用作其缺省值。在 shell 步骤中使用此变量（而非一个显式路径名）允许步骤从不同的目录运行而无须改动。
IFS	指定 IFS（内部字段分隔符）（通常是空格、制表符和换行），用于分隔由命令替换或参数替换产生的命令字，及用于分隔常规内置命令 **read** 使用的字。**IFS** 参数的第一个字符用于分隔 **$*** 替换的自变量。
LANG	为 **LC_*** 变量提供缺省值。
LC_ALL	重设 **LANG** 和 **LC_*** 变量的值。
LC_COLLATE	确定模式匹配中的范围表达式的行为。
LC_CTYPE	定义字符分类、大小写转换和其它字符属性。
LC_MESSAGES	确定写消息的语言。
LINES	确定打印选择列表的列长度。选择垂直打印的列表，直到填充了 **LINES** 变量指定的约三分之二行。
MAIL	指定邮件系统用于检测新邮件到达的文件路径名。如果此变量设置为邮件文件的名称并且 **MAILPATH** 变量未设置，则 shell 通知用户指定文件中的新邮件。
MAILCHECK	指定 shell 多久（以秒计）检查由 **MAILPATH** 或 **MAIL** 变量指定的任何文件的修改时间的更改。缺省值是 600 秒。当时间过去后，shell 在发出下一提示符前检查。
MAILPATH	指定由冒号分隔的文件名的列表。如果设置了此变量，则 shell 通知用户指定文件的任何修改，该修改发生在由 **MAILCHECK** 变量指定的周期（以秒计）期间。每个文件名可后跟 **?** 和将打印的消息。消息将经历变量替换，**$_** 变量定义为已更改的文件的名称。缺省消息是 `you have mail in $_`。
NLSPATH	为处理 **LC_MESSAGES** 确定消息编目的位置。
PATH	表示命令的搜索路径，它是由冒号分隔的目录路径名的有序列表。当 shell 查找命令时，它以指定的顺序搜索这些目录。列表中任何位置的空字符串代表当前目录。
PS1	指定用作主系统提示符的字符串。为参数替换展开此参数的值，以定义主提示符字符串，缺省情况下是 **$**。主提示符字符串中的 **!** 字符由命令号替换。
PS2	指定次提示符字符串的值，缺省情况下是 **>**。
PS3	指定 **select** 循环中使用的选择提示符字符串的值，缺省情况下是 **#?**。
PS4	为参数替换展开此变量的值，并且它先于执行跟踪的每一行。如果省略，则执行跟踪提示符是 **+**。
SHELL	指定保留在环境中的 shell 的路径名。
SHELL PROMPT	当交互地使用时，shell 在读命令前提示 **PS1** 参数的值。如果任何时候输入新行并且 shell 需要进一步输入以完成命令，则 shell 发出次提示符（**PS2** 参数）。
TMOUT	指定在退出前 shell 等待不活动的秒数。如果 **TMOUT** 变量设置为一个大于零（0）的值，则在发出 **PS1** 提示符后未在指示的秒数内输入命令，shell 退出。（请注意，shell 可以使用不能超过此值的最大边界来编译。） **注**: 在超时周期到期后，shell 退出前有 60 秒暂停。
VISUAL	如果该变量的值以 emacs、gmacs 或 vi 结束，则打开对应选项。

shell 将缺省值给 **PATH**、**PS1**、**PS2**、**MAILCHECK**、**TMOUT** 和 **IFS** 参数，但是 **HOME**、**SHELL**、**ENV** 和 **MAIL** 参数*不是*由 shell 设置的（虽然 **HOME** 参数是由 **login** 命令设置的）。

Korn shell 或 POSIX shell 中的命令替换

Korn shell 或 POSIX shell 使您能够进行命令替换。

在命令替换中，shell 在子 shell 环境中执行指定的命令并用其输出替换该命令。要在 Korn shell 或 POSIX shell 中执行命令替换，请执行以下命令:

```
$(command)
```

或者，对于反引号的版本，使用:

```
`command`
```

> **注**: 尽管 **ksh** 接受反引号的语法，但是 X/Open Portability Guide Issue 4 和 POSIX 标准认为它过时。这些标准建议可移植应用程序使用 `$(command)` 语法。

shell 通过在子 shell 环境中执行 *command* 并用命令的标准输出替换命令替换（*command* 的文本加圈起 $( ) 或反引号），在替换的结束除去一个或多个换行字符的序列来展开命令替换。

在下例中，圈起命令的 $( ) 表示 **whoami** 命令的输出被替换:

```
echo My name is: $(whoami)
```

可以使用以下命令执行同一个命令替换:

```
echo My name is: `whoami`
```

用户 dee 的两个示例的输出是:

```
My name is: dee
```

还可以通过将它们圈起在 ( ) 中替换算术表达式。例如，命令:

```
echo Each hour contains $((60 * 60)) seconds
```

产生以下结果:

```
Each hour contains 3600 seconds
```

当执行命令替换时，Korn shell 或 POSIX shell 除去所有末尾换行字符。例如，如果您的当前目录包含 file1、file2 和 file3 文件，则命令是:

```
echo $(ls)
```

除去换行字符并产生以下输出:

```
file1 file2 file3
```

要保留换行字符，请将替换的命令插入 ″ ″ 中:

```
echo "$(ls)"
```

Korn shell 或 POSIX shell 中的算术求值

Korn shell 或 POSIX shell 常规内置 **let** 命令使您能够执行整数算术。常数格式为 [*Base*]*Number*。*Base* 参数是 2 到 36（包括 2 和 36）之间的十进制数，代表算术基数。*Number* 参数是以此为基数的一个数。如果省略 *Base* 参数，则 shell 使用基数 10。

算术表达式使用与 C 语言相同的语法、优先权和表达式关联性。支持所有的整数运算符，除双加号（**++**）、双连字符（**—**）、问号、冒号（**?:**）和逗号（**,**）外。下表以优先权递减顺序列出有效的 Korn shell 或 POSIX shell

运算符:

运算符	定义
-	一元减号
!	逻辑非
~	按位非
*	乘法
/	除法
%	余数
+	加法
-	减法
<<, >>	左移、右移
<=,>=, <>, ==, !=	比较
&	按位“与”
^	按位“异”
|	按位“或”
&&	逻辑“与”
||	逻辑“或”
= *=, /=, &= +=, -=, <<=, > >=&=, ^=, |=	赋值

许多算术运算符，如 *、&、< 和 >，对 Korn shell 或 POSIX shell 有特殊的含义。必须在这些字符上加引号。例如，要用 5 乘 y 的当前值，并将新值重新指定给 y，请使用表达式:

```
let "y = y * 5"
```

用引号圈起表达式将除去 ***** 字符的特殊含义。

可以在 **let** 命令表达式中分组操作以强制分组。例如，在表达式中:

```
let "z = q * (z - 10)"
```

命令用 z 的减少的值乘 q。

如果只求值一个表达式，则 Korn shell 或 POSIX shell 包含 **let** 命令的备用格式。shell 将以 **(( ))** 圈起的命令作为引证的表达式处理。因此，表达式:

```
((x = x / 3))
```

等价于:

```
let "x = x / 3"
```

命名的参数在算术表达式中根据名称引用，不使用参数替换语法。当引用命名的参数时，其值求值为一个算术表达式。

使用 **typeset** 特殊内置命令的 **-i** 标志指定命名参数的内部整数表示法。使用 **-i** 标志，算术求值对每个对命名参数的赋值的值执行。如果不指定算术基数，则参数的第一个赋值确定算术基数。参数替换发生时使用此基数。

Korn shell 或 POSIX shell 中的字段分割

在执行命令替换后，Korn shell 扫描替换的结果，以查找在 **IFS**（内部字段分隔符）变量中找到的那些字段分隔符字符。在找到这样字符的地方，shell 将替换分割为不同的自变量。shell 保留显式空自变量（″″ 或 ’’）并除去隐式空自变量（由不具有值的参数的产生的自变量）。

- 如果 **IFS** 的值是空格、制表符和换行字符，或如果未设置，则输入的开始或结束处的任何空格、制表符和换行字符的序列都将被忽略，并且输入中的那些字符的任何序列都将定界字段。例如，以下输入产生两个字段，**school** 和 **days**:

  ```
  <newline><space><tab>school<tab><tab>days<space>
  ```

- 否则，如果 **IFS** 的值非空，则以下规则应用于序列。**IFS** 空*白*用于指在 **IFS** 值中的空白字符的任何序列（零个或多个实例）（例如，如果 **IFS** 包含 space/comma/tab，则任何空格或制表符的序列都认为是 **IFS** 空白）。

  1. **IFS** 空白在输入的开始和结束处忽略。
  2. 输入中每个非 **IFS** 空白的 **IFS** 字符，以及任何相邻的 **IFS** 空白，定界一个字段。
  3. 非零长度 **IFS** 空白定界字段。

Korn shell 或 POSIX shell 中的文件名替换

Korn shell 或 POSIX shell 扫描由 *Word* 变量指定的每个命令字对某些字符执行文件名替换。如果命令字包含 *****、**?** 或 **[** 字符，且未设置 **-f** 标志，则 shell 将字当作模式。shell 用与该模式匹配的文件名替换字，并根据当前语言环境中的有效整理序列排序。如果 shell 未找到与模式匹配的文件名，则它不更改字。

当 shell 将一个模式用作文件名替换时，**.** 和 **/** 字符必须显式地匹配。

> **注**: Korn shell 在模式匹配的其它实例中不特别地对待这些字符。

这些模式匹配字符表示以下替换:

*****	匹配任何字符串，包括空字符串。
?	匹配任何单个字符。
[...]	匹配任一圈起的字符。由 **-** 分隔的字符对匹配该对包含范围中词典编纂地任何字符，根据当前语言环境中的有效整理序列。如果跟在开始 **[** 后的第一个字符是一个 **!**，则匹配任何未圈起的字符。可以通过将 **-** 置为第一个字符或最后一个字符而使其包含在字符集中。

还可以使用 `[:charclass:]` 标志法来匹配指示范围中的文件名。此格式指示系统匹配属于 `class` 类的任何单个字符。哪些字符组成一个特定字符类的定义呈现在 **setlocale** 子例程的 **LC_CTYPE** 类别中。识别当前语言环境中指定的所有字符类。

一些字符类的名称如下:

- **alnum**
- **alpha**
- **cntrl**
- **digit**
- **graph**
- **lower**
- **print**
- **punct**

- **space**
- **upper**
- **xdigit**

例如，`[[:upper:]]` 匹配任何大写字母。

Korn shell 支持基于整理元素、符号或等价类的文件名扩展。

PatternList 是一个或多个彼此用 **|** 分隔的模式的列表。组合模式使用以下一个或多个形成:

?(*PatternList***)**	可选地匹配任何一个给定的模式
(PatternList***)**	匹配给定模式的零个或多个出现
+(*PatternList***)**	匹配给定模式的一个或多个出现
@(*PatternList***)**	精确地匹配一个给定模式
!(*PatternList***)**	匹配除一个给定模式外的任何模式

模式匹配具有一些限制。如果文件名的第一个字符是点（.），则它只能由也以点开头的模式匹配。例如，***** 匹配文件名 `myfile` 和 `yourfile`，而不匹配文件名 `.myfile` 和 `.yourfile`。要匹配这些文件，请使用诸如以下的模式:

```
.*file
```

如果模式不匹配任何文件名，则模式本身被作为尝试的匹配的结果返回。

文件和目录不应包含字符 *，?，-，[或]，因为它们可以在模式匹配尝试期间导致无限递归（即，无限循环）。

引证除去

存在于原始字中的引证字符反斜杠 (\)、单引号（'）和双引号（″）将被除去，除非它们本身已引证。

Korn shell 或 POSIX shell 中的输入和输出重定向

Korn shell 在执行命令前，扫描命令行中以查找重定向字符。这些特殊的符号指定 shell 以重定向输入和输出。重定向字符可以在一个简单命令中的任何位置出现，或者可以出现在命令之前或之后。它们不传递到调用的命令。

shell 在使用 *Word* 或 *Digit* 参数前执行命令和参数替换，除非另有说明。仅当模式与单个文件匹配且不执行空白解释时，文件名替换才发生。

<*Word*	将 *Word* 参数指定的文件用作标准输入（文件描述符 0）。
>*Word*	将 *Word* 参数指定的文件用作标准输出（文件描述符 1）。如果文件不存在，则 shell 将创建它。如果文件存在并且 **noclobber** 选项打开，则导致错误；否则，文件截断成零长度。
>|*Word*	与 **>***Word* 命令相同，除了该重定向语句覆盖 **noclobber** 选项。
> >*Word*	使用 *Word* 参数指定的文件作为标准输出。如果文件当前存在，则 shell 将输出附加到文件（通过首先查找文件结束符字符）。如果文件不存在，则 shell 将创建它。
<>*Word*	打开 *Word* 参数指定的文件，以作为标准输出读写。
<<[**-**]*Word*	读 shell 输入的每一行，直到定位一个仅包含 *Word* 参数的值或文件结束符字符的行。shell 不对指定的文件执行参数替换、命令替换或文件名替换。结果文档（称为 *here 文档*）成为标准输入。有关 here 文档的更多信息，请参阅 “第 48 页的『使用直接插入输入（Here）文档』”。如果 *Word* 参数的任何字符已引证，则不对文档的字符执行解释。

here 文档被视为在下一换行字符后开始，一直继续到有仅包含定界符的行的单个字，没有尾部空白字符。然后下一个 here 文档（如果有）开始。格式如下:

```
[n]<<word
   here document
delimiter
```

如果引证了 *word* 中的任何字符，则通过除去 *word* 上引证字符形成定界符。here 文档行将不展开。否则，定界符是 *word* 自身。如果 *word* 中没有引证字符，则 here 文档的所有行将被展开，以进行参数扩展、命令替换和算术扩展。

shell 执行重定向数据的参数替换。要防止 shell 解释 \, $ 和单引号（'）字符以及 *Word* 参数的第一个字符，在字符前加一个 \ 字符。

如果附加 - 到 <<，则 shell 从 *Word* 参数和文档中舍去所有前导制表符。

<&*Digit*	从 *Digit* 参数指定的文件描述符复制标准输入
>&*Digit*	在 *Digit* 参数指定的文件描述符中复制标准输出
<&-	关闭标准输入
>&-	关闭标准输出
<&p	将输入从联合进程移动到标准输入
>&p	将输出从联合进程移动到标准输出

如果这些重定向选项的某一个之前有一个数字，则引用的文件描述符号由数字指定（而不是缺省值 0 或 1）。在下例中，shell 打开文件描述符 2，以写为文件描述符 1 的副本。

```
... 2>&1
```

指定重定向的顺序是有意义的。在求值时，shell 根据 (*FileDescriptor, File*) 关联求值每个重定向。例如，在语句中:

```
... 1>File 2>&1
```

文件描述符 1 与 *File* 参数指定的文件关联。shell 将文件描述符 2 和与文件描述符 1（*File*）关联的文件关联。如果重定向的顺序被反向，则文件描述符 2 将与终端关联（假定文件描述符 1 以前与终端关联），而文件描述符 1 将与 *File* 参数指定的文件关联。

如果命令后跟一个和符号（&）且作业控制不活动，则命令的缺省标准输入是空文件 **/dev/null**。否则，命令执行的环境包含输入和输出规范所修改的调用 shell 的文件描述符。

有关重定向的更多信息，请参阅第 45 页的第 5 章，『输入和输出重定向』。

联合进程设施

Korn shell 或 POSIX shell 允许您运行一个或多个命令作为后台进程。这些从 shell 脚本运行的命令称为*联合进程*。

通过将 **|&** 运算符放置在命令后来指定联合进程。命令的标准输入和输出都通过管道传递到您的脚本。

联合进程必须满足以下限制:

- 在每个消息的结尾包含换行字符
- 将每个输出消息发送给标准输出
- 在每个消息后清除其标准输出

下例演示输入如何传递给联合进程和从联合进程返回:

```
echo "Initial process"
./FileB.sh |&
read -p a b c d
echo "Read from coprocess: $a $b $c $d"
print -p "Passed to the coprocess"
read -p a b c d
echo "Passed back from coprocess: $a $b $c $d"
```

```
FileB.sh
   echo "The coprocess is running"
   read a b c d
   echo $a $b $c $d
```

结果标准输出如下:

```
Initial process
Read from coprocess: The coprocess is running
Passed back from coprocess: Passed to the coprocess
```

要写入联合进程，请使用 **print -p** 命令。要从联合进程读取，请使用 **read -p** 命令。

重定向联合进程输入和输出

联合进程的标准输入和输出通过使用 I/O 重定向重新指定给一个带号码的文件描述符。例如，命令:

```
exec 5>&p
```

将联合进程的输入移动到文件描述符 5。

该操作完成之后，可以使用标准重定向语法，将命令输出重定向到联合进程。还可以启动另一个联合进程。两个联合进程的输出都连接到同一个管道，并使用 **read -p** 命令读。要停止联合进程，请输入:

```
read -u5
```

Korn shell 或 POSIX shell 中的退出状态

shell 检测到的错误（如语法错误）会导致 shell 返回非零退出状态。否则，shell 返回上次命令执行的退出状态。shell 通过打印命令或函数名和错误情况报告检测到的运行时错误。如果发生错误的行号大于 1，则行号还打印在命令或函数名后的 [ ]（方括号）中。

对于非交互式 shell，特殊内置或其它类型命令遇到的错误将导致 shell 写如以下表中显示的诊断消息:

错误	特殊内置	其它实用程序
shell 语言语法错误	将退出	将退出
实用程序语法错误（选项或操作数错误）	将退出	将不退出
重定向错误	将退出	将不退出
变量赋值错误	将退出	将不退出
扩展错误	将退出	将退出
命令未找到	不适用	可能退出
点脚本未找到	将退出	不适用

如果任何显示为“将（可能）退出”的错误在子 shell 中发生，则子 shell 将（可能）退出并有非零状态，但是包含子 shell 的脚本由于错误将不退出。

在表中显示的所有情况中，交互式 shell 将在不退出的情况下写诊断消息到标准错误。

Korn shell 或 POSIX shell 内置命令

特殊命令内置在 Korn shell 和 POSIX shell 中，并在 shell 进程中执行。除非另有指示，否则输出被写入文件描述符 1，并且如果命令不包含任何语法错误则退出状态为零（0）。允许输入和输出重定向。有两种类型的内置命令，*特殊内置命令*和*常规内置命令*。

特殊内置命令与常规内置命令的差异如下:

- 特殊内置命令中的语法错误可能导致执行命令的 shell 结束。如果在常规内置命令中有语法错误则这不会发生。如果特殊内置命令中的语法错误不结束 shell，则出口值为非零。
- 使用特殊内置命令指定的变量赋值在命令完整后保留有效。
- I/O 重定向在参数赋值后处理。

此外，跟在 **export**、**readonly** 和 **typeset** 特殊命令后、采用参数赋值格式的字使用与参数赋值相同的规则展开。代字号替换在 = 后执行，不执行字分割和文件名替换。

有关这些命令的字母顺序列表，请参考第 165 页的『Korn shell 或 POSIX shell 内置命令的列表』

特殊内置命令描述

Korn shell 提供以下特殊内置命令:

:	eval	newgrp	shift
.	exec	readonly	times
break	exit	return	trap
continue	export	set	typeset
			unset

: [*Argument* ...]
仅展开自变量。它在需要命令时使用，如在 *if* 命令的 **then** 条件中，但命令不执行任何操作。

. *File* [*Argument* ...]
读完整的指定文件，然后执行命令。命令在当前 shell 环境中执行。由 **PATH** 变量指定的搜索路径用于查找包含指定文件的目录。如果指定了任何自变量，则它们将成为位置参数。否则，不更改位置参数。退出状态是最近一条执行的命令的退出状态。有关位置参数的更多信息，请参考第 147 页的『Korn shell 或 POSIX shell 中的参数替换』。

注: 在执行任何命令前，*.File* [*Argument* ...] 命令读整个文件。因此，文件中的 **alias** 和 **unalias** 命令不适用于文件中指定的任何函数。

break [*n*]
从封闭的 **for**、**while**、**until** 或 **select** 循环退出（如果存在）。如果您指定 *n* 参数，则命令中断由 *n* 参数指定的级别号。*n* 的值是任何等于或大于 1 的整数。

continue [*n*]
继续封闭的 **for**、**while** 或 **until** 或 **select** 循环的下一个迭代。如果您指定 *n* 变量，则命令在第 *n* 个封闭循环处继续。*n* 的值是任何等于或大于 1 的整数。

eval [*Argument* ...]
将指定的自变量作为输入读入 shell，并执行（一个或多个）结果命令。

exec [*Argument* ...]
代替此 shell 执行自变量指定的命令（不创建新进程）。输入和输出自变量可出现并影响当前进程。如果不指定自变量，则 **exec** 命令修改输入和输出重定向列表所指示的文件描述符。在这种情况下，使用此机制打开的任何大于 2 的文件描述符号在调用另一个程序时关闭。

exit [*n*]
退出由 *n* 参数指定其退出状态的 shell。*n* 参数必须是范围 0-255 的无符号十进制整数。如果省略 *n* 参数，则退出状态是执行最近命令的退出状态。文件结束符字符也退出 shell，除非 **set** 特殊命令的 **ignoreeof** 选项打开。

export -p [*Name*[**=** *Value*]] ...

为自动导出到后续执行的命令的环境标记指定的名称。

-p 将所有导出的变量的名称和值以下面的格式写到标准输出:

```
"export %s= %s\n", <name> <value>
```

newgrp [*Group*]

等价于 **exec/usr/bin/newgrp** [*Group*] 命令。

注: 此命令不返回。

readonly -p [*Name*[**=** *Value*]] ...

将 *Name* 参数指定的名称标记为只读。这些名称无法由后继赋值更改。

-p 将所有导出的变量的名称和值以下面的格式写到标准输出:

```
"export %s= %s\n", <name> <value>
```

return [*n*]

使 shell 函数返回到调用脚本。返回状态由 *n* 变量指定。如果省略 *n* 变量，则返回状态是执行最近命令的返回状态。如果在函数或脚本外调用 **return** 命令，则它与 **exit** 命令相同。

set [**+**|**-abCefhkmnostuvx**] [**+** |**-o** *Option*]... [**+** |**-A** *Name*] [*Argument* ...]

如果未指定选项或自变量，则 **set** 命令以当前语言环境的整理序列写所有 shell 变量的名称和值。当指定了选项时，它们将设置或取消设置 shell 的属性，描述如下:

-A 数组赋值。取消设置 *Name* 参数，并从指定的 *Argument* 参数列表顺序地指定值。如果使用了 **+A** 标志，则首先不取消设置 *Name* 参数。

-a 自动导出定义的所有后继参数。

-b 异步地通知用户后台作业的完成。

-C 等价于 `set -o noclobber`。

-e 执行 **ERR** 陷阱（如果设置），并且如果命令具有非零退出状态则退出。读概要文件时此方式禁用。

-f 禁用文件名替换。

-h 当第一次遇到时，将每个命令指定为跟踪的别名。

-k 将所有参数赋值自变量都放入命令的环境，而不只是命令名前的那些自变量。

-m 在独立的进程中运行后台作业，并在完成后打印行。后台作业的退出状态在完成消息中报告。在具有作业控制的系统上，此标志为交互式 shell 自动打开。有关更多信息，请参阅第 167 页的『Korn shell 或 POSIX shell 中的作业控制』。

-n 读命令并检查其语法错误，但不执行它们。对于交互式 shell 忽略此标志。

-o *Option*

打印当前选项设置和错误消息（如果没有指定自变量）。可在单个 **ksh** 命令行上设置多个选项。如果使用了 **+o** 标志，则取消设置指定的选项。当指定了自变量时，它们将导致设置或取消设置位置参数。由 *Option* 变量指定的自变量可为以下之一:

allexport

同 **-a** 标志。

bgnice

以较低的优先级运行所有后台作业。这是缺省方式。

emacs

为命令条目输入 emacs 样式直接插入编辑器。

errexit

同 **-e** 标志。

gmacs

为命令条目输入 gmacs 样式直接插入编辑器。

ignoreeof

当遇到文件结束符字符时不退出 shell。要退出 shell，必须使用 **exit** 命令，或按下 Ctrl-D 按键序列超过 11 次。

keyword

同 **-k** 标志。

注: 此标志仅用于与 Bourne shell 的反向兼容性。强烈反对其使用。

markdirs

将 / 附加到作为文件名替换结果的所有目录名。

monitor

同 **-m** 标志。

noclobber

阻止重定向截断现有文件。当指定此选项时，必须在重定向符号后跟垂直栏（>|）以截断文件。

noexec

同 **-n** 标志。

noglob

同 **-f** 标志。

nolog 阻止 .profile 和 $ENV 文件中的函数定义保存在历史文件中。

nounset

同 **-u** 标志。

privileged

同 **-p** 标志。

trackall
同 **-h** 标志。

verbose
同 **-v** 标志。

vi 为命令条目输入 vi- 样式直接插入编辑器的插入方式。输入转义字符 033 将编辑器置于移动方式。返回发送行。

viraw 在每个字符以 vi 方式输入时处理它。

xtrace
同 **-x** 标志。

-p 禁用 **$HOME/.profile** 文件的处理，并使用 **/etc/suid _profile** 文件，而非 **ENV** 文件。当有效的用户标识（UID）或组标识（GID）不等于实际的 UID 或 GID 时，此方式启用。关闭此选项将 UID 或 GID 设置为实际的 UID 和 GID。
注: 由于操作系统不支持 **setuid** shell 脚本，所以系统不支持 **-p** 选项。

-s 按词典编纂的方式排序位置参数。

-t 在读和执行一个命令后退出。
注: 此标志仅用于与 Bourne shell 的反向兼容性。强烈反对其使用。

-u
替换时将取消设置参数作为错误处理。

-v 读 shell 输入行时打印它们。

-x 执行命令及其自变量时打印它们。

- 关闭 **-x** 和 **-v** 标志，并停止检查自变量的标志。

— 阻止更改任何标志。在将 **$1** 参数设置为以 **-** 开头的值时，此选项有用。如果没有自变量跟在此标志后，则不设置位置参数。

在任何 **set** 命令标志前加 **+**（而不是 **-**）关闭标志。在调用 shell 时可以使用这些标志。标志的当前设置可以在 **$-** 参数中找到。除非指定 **-A** 标志，否则剩余的自变量是位置参数，且按顺序指定为 $1、$2、...，等等。如果没有给出自变量，则所有命名参数的名称和值都打印到标准输出。

shift [*n*] 重命名位置参数，以 $n+1 ... 到 $1 ... 开始。*n* 参数的缺省值是 1。*n* 参数是求值为小于或等于 $# 参数的非负数的任何算术表达式。

times 打印 shell 和从 shell 运行的进程的累加用户和系统次数。

trap [*Command*] [*Signal*] ...

当 shell 接收到指定的（一个或多个）信号时运行指定的命令。*Command* 参数在设置陷阱时读一次，在采用陷阱时读一次。*Signal* 参数可作为数字或信号的名称给出。陷阱命令以信号号码的顺序执行。在当前 shell 的入口上忽略的信号上设置陷阱的任何尝试都是无效的。

如果命令是 **-**，则所有的陷阱复位为其原始值。如果省略命令且第一个信号是数字信号号码，则 **ksh** 命令将 *Signal* 参数的值复位成原始值。

注: 如果省略命令且第一个信号是符号名，则将信号解释成一个命令。

如果 *Signal* 参数的值是 **ERR** 信号，则无论何时命具令有非零退出状态时都执行指定的命令。如果信号是 **DEBUG**，则在每个命令后执行指定的命令。如果 *Signal* 参数的值是 **0** 或 **EXIT** 信号，且 **trap** 命令在函数主体内执行，则在函数完成后执行指定的命令。如果 *Signal* 参数是 **0** 或 **EXIT**（对于在任何函数外设置的 **trap** 命令），则在从 shell 退出时执行指定的命令。没有自变量的 **trap** 命令打印与每个信号号码关联的命令列表。

有关 *Signal* 参数值的完整列表（不带 **SIG** 前缀在 **trap** 命令中使用），请参阅 *AIX 5L Version 5.2 Technical Reference: Base Operating System and Extensions Volume 2* 中的 **sigaction**、**sigvec** 或 **signal** 子例程。

typeset [**+HLRZfilrtux** [*n*]] [*Name*[**=** *Value*]] ...

设置 shell 参数的属性和值。当在函数内调用时，将创建 *Name* 参数的一个新实例。函数完成时恢复参数值和类型。可以使用 **typeset** 命令指定以下标志:

-H 在非 AIX 机器上提供 “AIX 至主机文件” 映射。

-L 左对齐并从 *Value* 参数中除去前导空白。如果 *n* 参数具有非零值，则它定义字段的宽度；否则，它由其第一个赋值的值的宽度确定。当指定了参数时，它用空白向右填充，或截断（如有必要）以适合字段的长度。如果同时设置了 **-Z** 标志，则除去前导零。**-R** 标志关闭。

-R 右对齐并用前导空白填充。如果 *n* 参数具有非零值，则它定义字段的宽度；否则，它由其第一个赋值的值的宽度确定。如果重新指定参数，则字段仍用空白填充，并从末尾截断。**L** 标志关闭。

-Z 如果第一个非空白字符是数字且未设置 **-L** 标志，则右对齐并用前导零填充。如果 *n* 参数具有非零值，则它定义字段的宽度；否则，它由其第一个赋值的值的宽度确定。

-f 表示名称指函数名称，而不是参数名称。不能进行赋值，并且仅有的其它有效标志是 **-t**、**-u** 和 **-x**。**-t** 标志打开此函数的执行跟踪。**-u** 标志使此函数被标记为未定义。当函数被引用时，搜索 **FPATH** 变量以查找函数定义。**-x** 标志允许函数定义在不是 **ksh** 命令的独立调用的 shell 脚本中保持有效。

-i 将参数标识为整数，使算术更快速。如果 *n* 参数具有非零值，则它定义输出算术基数；否则，第一个赋值确定输出基数。

-l 将所有大写字符转换成小写。**-u** 大写转换标志关闭。

-r 将 *Name* 参数指定的名称标记为只读。这些名称无法被后继赋值更改。

-t 标记命名的参数。标记可由用户定义，并对 shell 无特殊含义。

-u 将所有小写字符转换成大写字符。**-l** 小写标志关闭。

-x 为自动导出到后续执行的命令的环境标记由 *Name* 指定的名称。

使用 **+** 而不是 **-** 关闭 **typeset** 命令标志。如果不指定 *Name* 参数但指定标志，则打印有这些标志的参数的名称（以及值（可选地））列表。（使用 **+** 而不是 **-** 阻止打印值。）如果未指定任何名称或标志，则打印所有参数的名称和属性。

unset [**-fv**] *Name* ... 取消设置由名称列表给定的参数的值和属性。如果指定了 **-v**，则 *Name* 指变量名，且 shell 将取消设置它，并将它从环境中除去。无法取消设置只读变量。取消设置 **ERRNO**、**LINENO**、**MAILCHECK**、**OPTARG**、**OPTIND**、**RANDOM**、**SECONDS**、**TMOUT** 和下划线（_）变量除去它们的特殊含义，即使随后指定它们。

如果设置了 **-f** 标志，则 *Name* 指函数名，且 shell 将取消设置函数定义。

常规内置命令描述

Korn shell 提供以下常规内置命令:

```
alias           fg              print           ulimit
bg              getopts         pwd             umask
cd              jobs            read            unalias
command         kill            setgroups       wait
echo            let             test            whence
fc
```

alias [**-t**] [**-x**] [*AliasName*[**=** *String*]] ... 创建或重新定义别名定义，或将现有别名定义写到标准输出。

有关更多信息，请参阅《*AIX 5L V5.2 命令参考大全*》中的 **alias** 命令。

bg [*JobID*...] 将每个指定作业置于后台。如果未指定 *JobID* 参数，则当前作业置于后台。有关作业控制的更多信息，请参阅第 167 页的『Korn shell 或 POSIX shell 中的作业控制』。

有关在后台运行作业的更多信息，请参阅《*AIX 5L V5.2 命令参考大全*》中的 **bg** 命令。

cd [*Argument*]
cd *Old New* 此命令可以是两种格式之一。在第一种格式中，它将当前目录更改到由 *Argument* 参数指定的目录。如果 *Argument* 参数的值是 -，则目录更改到前一个目录。**HOME** shell 变量是 *Argument* 参数的缺省值。**PWD** 变量设置为当前目录。

CDPATH shell 变量定义包含 *Argument* 参数值的目录的搜索路径。备用目录名称由 : 分隔。缺省路径为空，指定当前目录。当前目录由一个空路径名指定，它紧跟在等号之后或出现在路径列表中冒号定界符之间的任何位置。如果指定的自变量以 / 开始，则不使用搜索路径。否则，为自变量搜索路径中的每个目录。

cd 命令的第二种格式在当前目录名（**PWD**）中，用由 *New* 变量指定的字符串替换由 *Old* 变量指定的字符串，并尝试更改到该新目录。

command [**-p**] *CommandName* [*Argument* ...]
command [**-v** | **-V**] *CommandName* 使 shell 将指定的命令和自变量作为简单命令处理，禁止 shell 函数查找。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **command** 命令。

echo [*String* ...] 将字符串写入标准输出。有关用法和描述，请参考 **echo** 命令。不支持 **-n** 标志。

fc [**-r**] [**-e** *Editor*] [*First* [*Last*]]
fc -l [**-n**] [**-r**] [*First* [*Last*]]
fc -s [*Old***=** *New*] [*First*] 显示命令历史文件的内容，或调用编辑器以修改和重新执行以前在 shell 中输入的命令。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **fc** 命令。

fg [*JobID*] 将每个指定的作业置于前台。如果未指定任何作业，命令将当前作业置于前台。

有关在前台运行作业的更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **fg** 命令。

getopts *OptionString Name* [*Argument ...*]

检查合法选项的 *Argument* 参数。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **getopts** 命令。

jobs [**-l** | **-n** | **-p**] [*JobID ...*]

显示在当前 shell 环境中启动的作业的状态。如果未使用 *JobID* 参数指定特定作业，则显示所有活动作业的状态信息。如果报告了作业终止，则 shell 从当前 shell 环境识别的进程标识列表中除去该作业的进程标识。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **jobs** 命令。

kill [**-s** { *SignalName* | *SignalNumber* }] *ProcessID...*

kill [*-SignalName* | *-SignalNumber*] *ProcessID...*

发送一个信号（缺省情况下，**SIGTERM** 信号）到正在运行的进程。此缺省操作正常地停止进程。如果要停止进程，请在 *ProcessID* 变量中指定进程标识（PID）。shell 报告正在后台运行的每个进程的 PID（除非您在流水线中启动了多个进程，在此情况下 shell 报告最后一个进程的号码）。还可以使用 **ps** 命令来查找命令的进程标识号。

kill -l [*ExitStatus*]

列出信号名称。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **kill** 命令。

let *Expression ...*

求值指定的算术表达式。如果最后一个表达式的值为非零，则退出状态为 0，否则为 1。有关更多信息，请参考第 151 页的『Korn shell 或 POSIX shell 中的算术求值』。

print [**-Rnprsu** [*n*]] [*Argument ...*]

打印 shell 输出。如果未指定任何标志，或如果指定了连字符（**-**）或双连字符（**—**）标志，则将打印自变量到标准输出，如 **echo** 命令描述。标志执行以下操作:

-R 以原始方式打印（忽略 **echo** 命令的转义约定）。**-R** 标志打印所有的后继自变量和除 **-n** 外的标志。

-n 防止将换行字符添加到输出。

-p 将自变量写到使用 **|&** 运行的进程的管道，而不是写到标准输出。

-r 以原始方式打印。忽略 **echo** 命令的转义约定。

-s 将自变量写到历史文件，而不是写到标准输出。

-u 指定一位数字的文件描述符单元号 *n*，输出放置在其上。缺省值是 1。

pwd

等价于 **print -r - $PWD**。

注: 内部 Korn shell **pwd** 命令不支持符号链接。

read [**-prsu** [**n**]] [*Name?Prompt*] [*Name...*]

获取 shell 输入。读一行，并使用 **IFS** 变量中的字符作为分隔符，将行分解成各个字段。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **read** 命令。

setgroups

执行 **/usr/bin/setgroups** 命令，它作为独立的 shell 运行。有关此命令的信息，请参阅 **setgroups** 命令。但有一个差异。**setgroups** 内置命令调用子 shell，而 **setgroups** 命令替换当前执行的 shell。因为支持内置命令仅是为了兼容性，所以建议脚本使用绝对路径名 **/usr/bin/setgroups**，而非 shell 内置命令。

test

同 [*expression*]。有关用法和描述，请参阅第 166 页的『Korn shell 或 POSIX shell 的条件表达式』。

ulimit [**-HSacdfmst**] [*Limit*]

设置或显示 **/etc/security/limits** 文件中定义的用户进程资源限制。此文件包含以下缺省限制:

```
fsize = 2097151
core = 2048
cpu = 3600
data = 131072
rss = 65536
stack = 8192
```

当用户添加到系统时，这些值用作缺省设置。这些值是当用户添加到系统时使用 **mkuser** 命令设置的，或用 **chuser** 命令更改。

限制分类为“软”或“硬”。用户可能使用 **ulimit** 命令更改其软限制，直到增加到由硬限制设置的最大值。您必须拥有 root 用户权限来更改资源硬限制。

很多系统不包含这些限制中的一个或多个。当指定 *Limit* 参数时，设置指定资源的限制。*Limit* 参数的值可以是使用每个资源或值 `unlimited` 指定的单位中的数字。可以指定以下 **ulimit** 命令标志:

-H 指定设置了给定资源的硬限制。如果您有 root 用户权限，则可以增加硬限制。任何用户都可以降低它。

-S 指定设置了给定资源的软限制。软限制可增加到硬限制的值。如果 **-H** 或 **-S** 选项都没有指定，则限制适用于这两个选项。

-a 列出所有当前资源限制。

-c 指定核心转储大小上 512 字节块的数目。

-d 指定数据区的大小（以 KB 计）。

-f 指定由子进程写的文件的 512 字节块的数目（可读取任何大小的文件）。

-m 指定物理内存大小的 KB 数。

-n 指定一个进程可打开的文件描述符数目的限制。

-s 指定堆栈区域大小的 KB 数。

-t 指定每个进程使用的秒数。

当省略 *Limit* 变量时，打印当前资源限制。打印软限制，除非您指定 **-H** 标志。当您指定多个资源时，限制名称和单位打印在值之前。如果未给定选项，则假定使用 **-f** 标志。当您更改值时，将硬限制和软限制都设置为 *Limit*，除非指定 **-H** 或 **-S**。

有关用户和系统资源限制的更多信息，请参考 *AIX 5L Version 5.2 Technical Reference: Base Operating System and Extensions Volume 1* 中的 **getrlimit**、**setrlimit** 或 **vlimit** 子例程。

umask [**-S**] [*Mask*]

确定文件许可权。当文件创建时，此值以及创建进程的许可权确定文件的许可权。缺省值是 022。如果未指定 Mask 参数，则 **umask** 命令将当前 shell 环境的文件方式创建掩码显示到标准输出。

有关文件许可权的更多信息，请参阅《*AIX 5L V5.2 命令参考大全*》中的 **umask** 命令。

unalias { **-a** | *AliasName...* }

除去每个指定的别名的定义，或如果使用 **-a** 标志，则除去所有的别名定义。从当前 shell 环境中除去别名定义。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **unalias** 命令。

wait [*ProcessID...*]

等待指定的作业并终止。如果未指定作业，则命令等待所有当前活动的子进程。来自此命令的退出状态是它等待的进程的退出状态。

有关更多信息，请参考《*AIX 5L V5.2 命令参考大全*》中的 **wait** 命令。

whence [**-pv**] *Name* ... 表示对于每个指定的名称，如果用作命令名将如何对其进行解释。当不带任何标志使用时，**whence** 将显示对应于每个名称的绝对路径名（如果有）。

-p 执行指定（一个或多个）名称的路径搜索，即使存在别名、函数或保留字。

-v 产生一个更详细报告，指定每个名称是何类型。

Korn shell 或 POSIX shell 内置命令的列表

特殊内置命令

:（冒号） 仅展开自变量。
.（点） 读指定的文件，然后执行命令。
break 从封闭的 **for**、**while**、**until** 或 **select** 循环退出（如果存在）。
continue 继续封闭的 **for**、**while** 或 **until** 或 **select** 循环的下一个迭代。
eval 将自变量作为输入读入 shell，并执行（一个或多个）结果命令。
exec 代替此 shell 执行 *Argument* 参数指定的命令，而不创建新进程。
exit 退出由 *n* 参数指定其退出状态的 shell。
export 为自动导出到后续执行的命令的环境标记名称。
newgrp 等价于 **exec /usr/bin/newgrp** [*Group* ...] 命令。
readonly 将指定的名称标记为只读。
return 使 shell 返回到调用脚本。
set 除非指定了选项或自变量，否则写当前语言环境的整理序列中所有 shell 变量的名称和值。
shift 重命名位置参数。
times 打印 shell 和从 shell 运行的进程的累加用户和系统次数。
trap 当 shell 接收到指定的（一个或多个）信号时运行指定的命令。
typeset 设置 shell 参数的属性和值。
unset 取消设置指定参数的值和属性。

常规内置命令

alias 将别名列表打印到标准输出。
bg 将指定作业置于后台。
cd 更改当前目录到指定的目录，或使用指定字符串替换当前字符串。
echo 将字符串写入标准输出。
fc 从上次在终端上输入的 HISTSIZE 变量命令选择某一范围的命令。在执行“旧到新”替换后重新执行指定的命令。
fg 将指定的作业置于前台。
getopts 检查合法选项的 *Argument* 参数。
jobs 列出指定作业的信息。
kill 将 **TERM**（终止）信号发送到指定的作业或进程。
let 求值指定的算术表达式。
print 打印 shell 输出。
pwd 等价于 **print -r -$PWD** 命令。
read 获取 shell 输入。
ulimit 设置或显示 **/etc/security/limits** 文件中定义的用户进程资源限制。
umask 确定文件许可权。
unalias 从别名列表中除去名称列表中的参数。
wait 等待指定的作业并终止。
whence 表示将如何解释每个指定的名称如果用作命令名。

有关更多信息，请参阅第 157 页的『Korn shell 或 POSIX shell 内置命令』。

Korn shell 或 POSIX shell 的条件表达式

条件表达式与 **[[** 复合命令一起使用以测试文件的属性和比较字符串。不对出现在 **[[** 和 **]]** 之间的字执行字分割和文件名替换。每个表达式由一个或多个以下一元或二进制表达式构成。

-a *File* 真，如果指定文件是指向另一存在文件的符号链接。
-b *File* 真，如果指定文件存在且是块特殊文件。
-c *File* 真，如果指定文件存在且是字符特殊文件。
-d *File* 真，如果指定文件存在且是目录。
-e *File* 真，如果指定文件存在。
-f *File* 真，如果指定文件存在且是普通文件。
-g *File* 真，如果指定文件存在且其 **setgid** 已设置。
-h *File* 真，如果指定文件存在且是符号链接。
-k *File* 真，如果指定文件存在且其粘滞位已设置。
-n *String* 真，如果指定字符串的长度非零。
-o *Option* 真，如果指定的选项打开。
-p *File* 真，如果指定文件存在且是 FIFO 特殊文件或管道。
-r *File* 真，如果指定文件存在且可由当前进程读取。
-s *File* 真，如果指定文件存在且大小大于 0。
-t *FileDescriptor* 真，如果指定文件描述符号打开且与终端设备关联。
-u *File* 真，如果指定文件存在且其 **setuid** 已设置。
-w *File* 真，如果指定文件存在且写位打开。但是，在只读文件系统上文件将不可写，即使此测试指示真。
-x *File* 真，如果指定文件存在且 **execute** 标志打开。如果指定文件存在且是目录，则当前进程具有在目录中搜索的许可权。
-z *String* 真，如果指定字符串的长度为 0。
-L *File* 真，如果指定文件存在且是符号链接。
-O *File* 真，如果指定文件存在且由此进程的有效用户标识所拥有。
-G *File* 真，如果指定文件存在且其组与此进程的有效用户标识匹配。
-S *File* 真，如果指定文件存在且是套接字。
File1 **-nt** *File2* 真，如果 *File1* 存在且比 *File2* 新。
File1 **-ot** *File2* 真，如果 *File1* 存在且比 *File2* 旧。
File1 **-ef** *File2* 真，如果 *File1* 和 *File2* 存在且指同一个文件。
String1 **=** *String2* 真，如果 *String1* 等于 *String2*。
String1 **!=** *String2* 真，如果 *String1* 不等于 *String2*。
String **=** *Pattern* 真，如果指定字符串与指定模式匹配。
String **!=** *Pattern* 真，如果指定字符串不与指定模式匹配。
String1 **<** *String2* 真，如果根据其字符的 ASCII 值，*String1* 出现在 *String2* 之前。
String1 **>** *String2* 真，如果根据其字符的 ASCII 值，*String1* 出现在 *String2* 之后。
Expression1 **-eq** *Expression2* 真，如果 *Expression1* 等于 *Expression2*。
Expression1 **-ne** *Expression2* 真，如果 *Expression1* 不等于 *Expression2*。
Expression1 **-lt** *Expression2* 真，如果 *Expression1* 小于 *Expression2*。
Expression1 **-gt** *Expression2* 真，如果 *Expression1* 大于 *Expression2*。
Expression1 **-le** *Expression2* 真，如果 *Expression1* 小于或等于 *Expression2*。
Expression1 **-ge** *Expression2* 真，如果 *Expression1* 大于或等于 *Expression2*。

注: 在以上表达式的每一个中，如果 *File* 变量类似于 `/dev/fd/n`（其中 `n` 是一个整数），则测试适用于其描述符号为 *n* 的打开文件。

可以通过使用任何以下表达式（以优先权的递减顺序列出）从这些原语或较小的部件构造复合表达式:

(*Expression***)**	真，如果指定表达式为真。用于对表达式分组。
! *Expression*	真，如果指定表达式为假。
Expression1 **&&** *Expression2*	真，如果 *Expression1* 和 *Expression2* 都为真。
Expression1 **||** *Expression2*	真，如果 *Expression1* 或 *Expression2* 为真。

Korn shell 或 POSIX shell 中的作业控制

Korn shell 或 POSIX shell 提供控制命令序列或*作业*的程序。当执行 **set -m** 特殊命令时，Korn shell 将作业与每个流水线关联。它保留当前作业表（通过 **jobs** 命令打印），并指定给它们小整数。

当使用 **&** 在后台启动作业时，shell 打印类似于以下的行:

```
[1] 1234
```

该输出表示在后台启动作业的作业号是 1。它还显示作业具有一个进程标识为 1234 的（顶级）进程。

如果您在运行一个作业时要执行其它操作，请使用 Ctrl-Z 按键序列。此按键序列发送一个 **STOP** 信号到当前作业。shell 通常表示作业已停止，然后显示一个 shell 提示符。然后您可以对该作业的状态进行操作（用 **bg** 命令将它置于后台），运行其它命令，最后使用 **fg** 命令将作业返回到前台。Ctrl-Z 按键序列立即生效，且类似一个中断，因为当您输入序列时 shell 废弃暂挂输出和未读输入。

在后台中运行的作业如果尝试从终端读，则停止。通常允许后台作业产生输出。可以通过发出 **stty tostop** 命令禁用此选项。如果设置此终端选项，则当后台作业尝试产生输出或读输入时停止。

可以使用几种方法引用 Korn shell 中的作业。一个作业通过任何其进程的进程标识引用，或以以下方式之一引用:

%*Number*	使用给定的号码指定作业
%*String*	指定其命令行以 *String* 变量开始的任何作业
%?*String*	指定其命令行包含 *String* 变量的任何作业
%%	指定当前作业
%+	等价于 **%%**
%-	指定前一个作业

此 shell 立即识别进程状态的更改。它通常在一个作业变得阻塞以致不可能有进一步进展时通知您。shell 就在它打印提示符前执行该操作，这样它就不另外打扰您的工作。

当监视方式打开时，每个完成的后台作业触发器俘获 **CHLD** 信号的设置。

如果在作业已停止或正在运行时尝试离开 shell（通过输入 exit 或使用 Ctrl-D 按键序列），则系统用消息有停止的（正在运行的）作业警告您。使用 **jobs** 命令查看哪些命令受到影响。如果您立即再次尝试退出，则 shell 终止已停止的和正在运行的作业，而不警告。

信号处理

如果调用的命令后跟 **&** 且作业 **monitor** 选项不活动，则忽略已调用命令的 **SIGINT** 和 **SIGQUIT** 信号。否则，信号具有 shell 从其父程序那里继承的值。

当 shell 在等待一个前台命令完成时，如果接收到一个信号（为它设置了一个陷阱），则与该信号关联的陷阱将不执行，直到前台命令完成后。因此，不执行 **CHILD** 信号上的陷阱，直到前台作业终止。

Korn shell 或 POSIX shell 中的直接插入编辑

通常，您从终端设备输入每个命令行，并将一个换行字符（RETURN 或 LINE FEED）跟在它后面。当您激活 emacs、gmacs 或 vi 直接插入编辑选项时，可以编辑命令行。

以下命令输入编辑方式:

set -o emacs 请输入 emacs 编辑方式并初始化 emacs 样式直接插入编辑器。有关更多信息，请参阅『emacs 编辑方式』。

set -o gmacs 请输入 emacs 编辑方式并初始化 gmacs 样式直接插入编辑器。有关更多信息，请参阅『emacs 编辑方式』。

set -o vi 请输入 vi 编辑方式并初始化 vi 样式直接插入编辑器。有关更多信息，请参阅第 169 页的『vi 编辑方式』。

VISUAL 或 **EDITOR** 变量每次指定到一个以任何这些选项名称末尾的值时会自动选定一个编辑选项。

注: 要使用编辑功能，终端必须接受 RETURN 作为回车而不换行。空格必须覆盖屏幕上的当前字符。

每个编辑方式在当前行打开一个窗口。窗口宽度是 **COLUMNS** 变量的值（如果它已定义）；否则，宽度是 80 个字符空间。如果行长于窗口宽度减二，则系统通过在窗口尾端显示一个标记来通知您。当光标移动并到达窗口边界时，窗口以光标位置为中心。显示的标记如下:

> 表示行在窗口的右边扩展。
< 表示行在窗口的左边扩展。
***** 表示行在窗口的两边扩展。

每个编辑方式中的搜索命令提供对 Korn shell 历史文件的访问。仅匹配字符串。如果字符串中的前导字符是 ^，则匹配必须在行的第一个字符开始。

emacs 编辑方式

emacs 编辑方式在您启用 **emacs** 或 **gmacs** 选项时输入。这两个方式的唯一差异是它们处理 Ctrl-T 编辑命令的方式不同。要编辑，请将光标移动到需要更正的点，并根据需要插入或删除字符或字。所有编辑命令都是控制字符或转义序列。

编辑命令从行上任何位置操作（不仅仅在开始处）。不要在编辑命令后按下 Enter 键或换行（向下箭头），除非另说明。

Ctrl-F 将光标向前（右）移动一个字符。
Esc-F 将光标向前移动一个字（仅由字母、数字和下划线组成的字符的字符串）。
Ctrl-B 将光标向后（左）移动一个字符。
Esc-B 将光标向后移动一个字。
Ctrl-A 将光标移动到行的开始。
Ctrl-E 将光标移动到行的末尾。
Ctrl-] *c* 将当前行上的光标向前移动到表示的字符。
Esc-Ctrl-] *c* 将当前行上的光标向后移动到表示的字符。
Ctrl-X Ctrl-X 交换光标和标记。
ERASE 删除前一个字符。（根据 **stty** 命令定义用户定义的擦除字符，通常是 Ctrl-H 按键序列）。
Ctrl-D 删除当前字符。
Esc-D 删除当前字。
Esc-Backspace 删除前一个字。
Esc-H 删除前一个字。

Esc-Delete	删除前一个字。如果中断字符是 Delete 键，则此命令不工作。
Ctrl-T	在 emacs 方式中用下一个字符调换当前字符。在 gmacs 方式中调换前两个字符。
Ctrl-C	将当前字符转成大写。
Esc-C	将当前字转成大写。
Esc-L	更改当前字为小写。
Ctrl-K	从光标删除到行的末尾。如果前面有一个值小于当前光标位置的数字参数，则此编辑命令从给定的位置向上删除到光标。如果前面有一个值大于当前光标位置的数字参数，则此编辑命令从光标位置向上删除到给定的位置。
Ctrl-W	从光标删除到标记。
Esc-P	将区域从光标推到堆栈上的标记。
KILL	根据 **stty** 命令定义用户定义的杀死字符，通常是 Ctrl-G 按键序列或 **@**. 杀死整个当前行。如果接连地输入两个杀死字符，则所有后继杀死字符会引起换行（使用纸张终端时有用）。
Ctrl-Y	恢复从行除去的最后一 。（将该项拉回行）。
Ctrl-L	换行并打印当前行。
Ctrl-@	(空字符) 设置标记。
Esc-space	设置标记。
Ctrl-J	（换行）执行当前行。
Ctrl-M	（返回）执行当前行。
EOF	仅当当前行为空时，将文件结束符字符（通常是 Ctrl-D 按键序列）作为一个文件结束符处理。
Ctrl-P	取前一个命令。每次输入 Ctrl-P 按键序列时，访问上一个执行的命令。当不在多行命令的第一行上时，移动回一行。
Esc-<	取最不新的（最旧的）历史行。
Esc->	取最近的（最新的）历史行。
Ctrl-N	取下一个命令行。每次输入 Ctrl-N 按键序列时，访问下一个将要执行的命令行。
Ctrl-R *String*	逆向搜索历史，查找包含 *String* 参数指定的字符串的前一个命令行。如果给定值 0，则向前搜索。指定的字符串用 Enter 键或换行字符终止。如果 ^ 在字符串前，则匹配的行必须以 *String* 参数开始。如果省略 *String* 参数，则访问包含最新 *String* 参数的下一个命令行。在这种情况下，值 0 逆向搜索方向。
Ctrl-O	（操作）执行当前行，并从历史文件取相对于当前行的下一行。
Esc *Digits*	（转义）定义数字参数。将数字作为参数带到下一个命令。接受参数的命令是 **Ctrl-F**、**Ctrl-B**、**ERASE**、**Ctrl-C**、**Ctrl-D**、**Ctrl-K**、**Ctrl-R**、**Ctrl-P**、**Ctrl-N**、**Ctrl-]**、**Esc-.**、**Esc-Ctrl-]**、**Esc-_**、**Esc-B**、**Esc-C**、**Esc-D**、**Esc-F**、**Esc-H**、**Esc-L** 和 **Esc-Ctrl-H**。
Esc *Letter*	（软键）搜索别名列表，查找名为 *_Letter* 的别名。如果定义了该名称的别名，则其值将被放入输入队列。*Letter* 参数绝不能指定换码功能中的一个。
Esc-[*Letter*	（软键）搜索别名列表，查找名为双下划线 Letter（*__Letter*）的别名。如果定义了该名称的别名，则其值将放入输入队列。此命令可用于许多终端上的程序功能键。
Esc-.	在行上插入前一个命令的最后一个字。如果数字参数在前，则此参数的值确定插入哪个字而不插入最后一个字。
Esc-_	同 Esc-. 按键序列。
Esc-*	尝试当前字上的文件名替换。如果字不匹配任何文件或包含任何特殊模式字符，则附加星号。
Esc-Esc	文件名完成。用所有文件名的最长公共前缀替换当前字，该文件名用附加的星号匹配当前字。如果匹配唯一，则当文件是目录时附加 **/**，当文件不是目录时附加空格。
Esc-=	列出匹配当前字模式（好象附加了星号）的文件。
Ctrl-U	用 4 乘下一个命令的参数。
****	转义下一个字符。编辑字符，并且如果 \ 在前，则 ERASE、KILL 和 INTERRUPT（通常是 Delete 键）字符可在命令行或搜索字符串中被输入。反斜杠除去下一个字符的编辑功能（如有）。
Ctrl-V	显示 shell 的版本。
Esc-#	在行的开始插入 #，然后执行行。这导致插入注释到历史文件。

vi 编辑方式

vi 编辑方式有两种输入类型。当您输入命令时，处在输入方式。要编辑，您必须通过按下 Esc 键输入控制方式。

大多数控制命令接受可选重复 *Count* 参数优于命令。当在大多数系统的 vi 方式中时，最初启用规范处理。如果以下的一个或多个为真，则该命令再次回显:

- 速度为 1200 波特或以上。
- 命令包含任何控制字符。
- 从打印提示符开始，经过时间小于一秒。

Esc 字符终止命令剩余部分的规范处理，然后可以修改命令行。此方案具有规范处理的优点，即原始方式的预输入回显。如果还设置了 **viraw** 选项，则总是禁用规范处理。该方式对于不支持两个备用行结束定界符的系统是隐式的，并且对于某些终端可能有帮助。

可用的 vi 编辑命令分组为几类。类别如下:

- 『输入编辑命令』
- 『移动编辑命令』
- 第 171 页的『搜索编辑命令』
- 第 171 页的『文本修改编辑命令』
- 第 171 页的『杂项编辑命令』

输入编辑命令

注: 缺省情况下，编辑器处在输入方式。

ERASE （根据 **stty** 命令定义用户定义的擦除字符，通常是 Ctrl-H 或 # 顺序）。删除前一个字符。
Ctrl-W 删除前一个空白分隔的字。
Ctrl-D 终止 shell。
Ctrl-V 转义下一个字符。编辑字符，如 ERASE 或 KILL 字符，可以输入到命令行或搜索字符串中，如果 Ctrl-V 按键序列在前。Ctrl-V 按键序列除去下一个字符的编辑功能（如果有）。
**** 转义下一个 ERASE 或 KILL 字符。

移动编辑命令

移动编辑命令移动光标如下:

[*Count*]**l** 将光标向前（右）移动一个字符。
[*Count*]**w** 将光标向前移动一个字母数字。
[*Count*]**W** 将光标到移动到下一个跟着空格的字的开始。
[*Count*]**e** 将光标移动到当前字的末尾。
[*Count*]**E** 将光标移动到当前空格分隔的字的末尾。
[*Count*]**h** 将光标向后（左）移动一个字符。
[*Count*]**b** 将光标向后移动一个字。
[*Count*]**B** 将光标移动到前一个空格分隔的字。
[*Count*]**|** 将光标移动到 *Count* 参数指定的列。
[*Count*]**f***c* 在当前行查找下一个字符 *c*。
[*Count*]**F***c* 在当前行查找前一个字符 *c*。
[*Count*]**t***c* 等价于 **f** 后跟 **h**。
[*Count*]**T***c* 等价于 **F** 后跟 **l**。
[*Count*]**;** 重复最后一个单字符查找命令，重复次数由 *Count* 参数指定: **f**、**F**、**t** 或 **T**。
[*Count*]**,** 逆向最后一个单字符查找命令，次数由 *Count* 参数指定。
0 将光标移动到行的开始。
^ 将光标移动到行的第一个非空白字符。
$ 将光标移动到行的末尾。

搜索编辑命令

搜索编辑命令访问命令历史，如下:

[*Count*]**k**	取一个命令。
[*Count*]**-**	等价于 **k** 命令。
[*Count*]**j**	取下一个命令。每次输入 **j** 命令时，访问下一个命令。
[*Count*]**+**	等价于 **j** 命令。
[*Count*]**G**	取 *Count* 参数指定其数的命令。缺省是最新历史命令。
/String	通过反向搜索历史，查找包含指定字符串的前一个命令。字符串由 RETURN 或换行字符终止。如果 ^ 在字符串前，则匹配的行必须以 *String* 参数开始。如果 *String* 参数的值为空，则使用前一个字符串。
?String	同 ***/String***，除了以向前方向搜索。
n	搜索与 ***/String*** 或 **?** 命令匹配的最后一个模式的下一匹配。
N	搜索与 ***/String*** 或 **?** 命令匹配的最后一个模式的下一匹配，但是方向相反。搜索历史，查找前一 ***/String*** 命令输入的字符串。

文本修改编辑命令

文本修改编辑命令修改行，如下:

a	进入输入方式，并在当前字符后输入文本。
A	将文本附加到行末尾。等价于 **$a** 命令。
[*Count*]**c***Motion*	
c[*Count*]*Motion*	删除从当前字符到 *Motion* 参数指定将光标移动到的字符，并进入输入方式。如果 *Motion* 参数值是 **c**，则删除整行并进入输入方式。
C	删除从当前字符到行结尾，并进入输入方式。等价于 **c$** 命令。
S	等价于 **cc** 命令。
D	删除从当前字符到行结尾。等价于 **d$** 命令。

[*Count*]**d***Motion*

d[*Count*]*Motion*	向上删除当前字符，到包含 *Motion* 参数指定的字符。如果 *Motion* 是 **d**，则删除整行。
i	进入输入方式并在当前字符前插入文本。
I	在行的开始前插入文本。等价于 **0i** 命令。
[*Count*]**P**	将前一个文本修改放在光标前。
[*Count*]**p**	将前一个文本修改放在光标后。
R	进入输入方式并叠打屏幕上的字符。
[*Count*]**r***c*	从当前光标位置开始，用 *c* 参数指定的字符替换 *Count* 参数指定的字符数。此命令还在替换字符后使光标前进。
[*Count*]**x**	删除当前字符。
[*Count*]**X**	删除前字符。
[*Count*]**.**	重复前一文本修改命令。
[*Count*]~	从当前光标位置开始，反转 *Count* 参数指定的字符数的大小写，且光标前进。
[*Count*]_	附加前一命令的 *Count* 参数指定的字，并进入输入方式。如果省略 *Count* 参数，则使用最后一个字。
*****	附加一个 * 到当前字并尝试文件名替换。如果没有找到匹配，则它响铃。否则，用匹配模式替换字并进入输入方式。
****	文件名完成。用所有文件名的最长公共前缀替换当前字，该文件名用附加的星号匹配当前字。如果匹配唯一，则当文件是目录时附加 **/**。如果文件不是目录，则附加空格。

杂项编辑命令

最常用的编辑命令包括以下:

[*Count*]**y***Motion*

y[*Count*]*Motion*	向上拉当前字符到并包含 *Motion* 参数指定的光标位置标记的字符，并将所有这些字符放入删除缓冲区。文本和光标未更改。
Y	从当前位置拉到行的结尾。等价于 **y$** 命令。
u	撤销最后一个文本修改命令。
U	撤销在行上执行的所有文本修改命令。
[*Count*]**v**	在输入缓冲区返回命令 `fc -e ${VISUAL:-${EDITOR:-vi}} Count`。如果省略 *Count* 参数，则使用当前行。
Ctrl-L	换行并打印当前行。该命令仅在控制方式中有效。
Ctrl-J	（换行）执行当前行，无论何种方式。
Ctrl-M	（返回）执行当前行，无论何种方式。
#	在行前插入 **#** 后发送行。如果您要在历史中插入当前行而不执行它，则它有用。 如果命令行包含管道或分号或换行字符，则在这些符号的每一个前将插入附加的 **#**。要删除所有磅符号，请从历史检索命令行并输入另一个 **#**。
=	列出匹配当前字（好象对其附加了星号）的文件名。
@*Letter*	在别名列表中搜索名为 *_Letter* 的别名。如果定义了该名称的别名，则其值将放入输入队列用于处理。

增强的 Korn shell（ksh93）

除了缺省系统 Korn shell（**/usr/bin/ksh**）外，AIX 还提供了可用作 **/usr/bin/ksh93** 的增强版本。此增强版本向上兼容当前缺省版本，并包括一些在 **/usr/bin/ksh** 中不提供的附加功能。

以下功能在 **/usr/bin/ksh93** 中提供:

算术增强	可以在算术表达式中使用 libm 函数（通常在 C 编程语言中找到的数学函数），如 `$ value=$((sqrt(9)))`。提供了更多的算术运算符，包括一元 **+**、**++**、**--** 和 **?:** 构造（例如，"x ? y : z"），以及 **,**（逗号）运算符。支持最大基数 64 的算术基数。还支持浮点算术。"typeset -E"（指数）可用于指定有效数字的数目，"typeset -F"（浮点）可用于指定一个算术变量的十进制位置的数目。现在 SECONDS 变量显示到最接近的百分之一秒，而非最接近的秒。

复合变量	支持复合变量。复合变量允许用户在单个变量名中指定多个值。这些值的每一个都使用下标变量指定，用一个 **.**（句号）与父变量分隔。例如:
	`$ myvar=( x=1 y=2 )`
	`$ print "${myvar.x}"`
	`1`

复合赋值	当初始化数组（索引数组和相联数组）时支持复合赋值。赋值的值放置在圆括号中，如下例所示:
	`$ numbers=( zero one two three )`
	`$ print ${numbers[0]} ${numbers[3]}`
	`zero three`

关联数组	关联数组是用字符串作为索引的数组。
	
	typeset 命令与 **-A** 标志一起使用允许您在 ksh93 中指定关联数组。例如:
	`$ typeset -A teammates`
	`$ teammates=( [john]=smith [mary]=jones )`
	`$ print ${teammates[mary]}`
	`jones`

变量名引用

typeset 命令与 **-n** 标志一起使用允许您将一个变量名指定为对另一个变量名的引用。这样，修改变量的值将依次修改所引用变量的值。例如:

```
$ greeting="hello"
$ typeset -n welcome=greeting     # establishes the reference
$ welcome="hi there"     # overrides previous value
$ print $greeting
hi there
```

参数扩展

以下参数扩展构造可用:

- ${!varname} 是变量本身的名称。
- ${!*varname*[@]} 命名 *varname* 数组的索引。
- ${*param*:*offset*} 是 *param* 的子串，以 *offset* 开始。
- ${param:offset:num} 是 *param* 的子串，以 *offset* 开始，用于 *num* 个字符数。
- ${@:*offset*} 表示所有位置参数以 *offset* 开始。
- ${@:*offset*:*num*} 表示 *num* 位置参数以 *offset* 开始。
- ${*param*/*pattern*/*repl*} 求值到 *param*，第一个出现的 *pattern* 由 *repl* 替换。
- ${*param*//*pattern*/*repl*} 求值到 *param*，每个出现的 *pattern* 由 *repl* 替换。
- ${*param*/#*pattern*/*repl*} 如果 *param* 以 *pattern* 开始，则 *param* 由 *repl* 替换。
- ${*param*/%*pattern*/*repl*} 如果 *param* 以 *pattern* 结尾，则 *param* 由 *repl* 替换。

规程函数

规程函数是与一个特定变量关联的函数。这允许您在每次引用、设置或取消设置该变量时定义和调用函数。这些函数采用 *varname.function* 格式，其中 *varname* 是变量的名称，*function* 是规程函数。预定义的规程函数是 **get**、**set** 和 **unset**。

- *varname*.**get** 函数在每次引用 *varname* 时调用。如果特殊变量 **.sh.value** 在此函数中设置，则 *varname* 的值更改为此值。一个简单的示例是一天中的时间:

  ```
  $ function time.get
  > {
  >     .sh.value=$(date +%r)
  > }
  $ print $time
  09:15:58 AM
  $ print $time    # it will change in a few seconds
  09:16:04 AM
  ```

- *varname*.**set** 函数在每次设置 *varname* 时调用。赋予 **.sh.value** 变量指定的值。指定给 *varname* 的值是函数完成时 **.sh.value** 的值。例如:

  ```
  $ function adder.set
  > {
  >   let .sh.value="
  $ {.sh.value} + 1"
  > }
  $ adder=0
  $ echo $adder
  1
  $ adder=$adder
  $ echo $adder
  2
  ```

- *varname*.**unset** 函数在每次取消设置 *varname* 时执行。变量实际上并未取消设置，除非它在函数本身内取消设置；否则它保留其值。

在所有规程函数中，特殊变量 **.sh.name** 设置为变量的名称，而 **.sh.subscript** 设置为变量下标的值（如果适用）。

函数环境

以 function *myfunc* 格式声明的函数在独立的函数环境中执行。声明为 *myfunc*() 的函数与父 shell 在同一环境中执行。

变量	以 **.sh.** 开始的变量由 shell 保留，并具有特殊含义。查看此表中“规程函数”的描述，以获取 **.sh.name**、**.sh.value** 和 **.sh.subscript** 的说明。还提供了 **.sh.version**，它表示 shell 的版本。
命令返回值	命令的返回值如下: • 如果未找到要执行的命令，则返回值设置为 127。 • 如果找到要执行的命令，但不可执行，则返回值为 126。 • 如果命令已执行，但由一个信号终止，则返回值为 256 加信号号。
PATH 搜索规则	首先搜索特殊内置命令，然后是所有函数（包括那些在 FPATH 目录中的函数），再是其它内置。
shell 历史	**hist** 命令允许您显示和编辑 shell 命令历史。在 ksh shell 中，使用了 **fc** 命令。**fc** 命令是 **hist** 的一个别名。变量是 HISTCMD（它对每个在 shell 中执行的命令增加一）和 HISTEDIT（它指定使用 **hist** 命令时要使用的编辑器）。
内置命令	增强的 Korn shell 包含以下内置命令: • **builtin** 命令列出所有可用的内置命令。 • **printf** 命令以与 printf() C 库例程类似的方式工作。请参考 **printf** 命令。 • **disown** 阻拦 shell 发送 SIGHUP 到指定的命令。 • **getconf** 命令以与独立命令 **/usr/bin/getconf** 相同的方式工作。请参考 **getconf** 命令。 • **read** 内置命令有以下标志: – **read -d {***char***}** 允许您指定字符定界符而不是缺省换行。 – **read -t {***seconds***}** 允许您指定以秒计的时间限制，在此时间限制后 **read** 命令将超时。如果 **read** 超时，则它将返回 FALSE。 • **exec** 内置命令有以下标志: – **exec -a {***name***} {***cmd***}** 指定使用 *name* 替换 *cmd* 的自变量 0。 – **exec -c {***cmd***}** 告诉 **exec** 在执行 *cmd* 前清除环境。 • **kill** 内置命令有以下标志: – **kill -n {***signum***}** 用于指定发送到进程的信号号，**kill -s {***signame***}** 用于指定信号名称。 – **kill -l**，没有自变量，列出所有信号名称，但不列出信号的成员。 • **whence** 内置命令有以下标志: – **-a** 标志显示所有匹配，而不只是第一个找到的匹配。 – **-f** 标志告诉 **whence** 不搜索任何函数。 • 转义字符序列用于被 **print** 和 **echo** 命令使用。Esc（Escape）键可用序列 \E 表示。 • 所有常规内置命令都识别 **-?** 标志，它显示指定命令的语法。

Bourne shell

Bourne shell 是交互式命令解释器和命令编程语言。**bsh** 命令运行 Bourne shell。

Bourne shell 可以作为登录 shell 或登录 shell 下的子 shell 运行。只有 **login** 命令可以将 Bourne shell 调用为登录 shell。它通过使用特殊格式的 **bsh** 命令名: **-**bsh 来实现。用初始连字符（-）调用时，shell 首先读并运行在系统 **/etc/profile** 文件和 **$HOME/.profile** 中找到的命令（如果存在一个的话）。**/etc/profile** 文件设置所有用户需要的变量。最后，shell 准备好从标准输入读取命令。

如果启动 Bourne shell 时指定 *File* [*Parameter*] 参数，则 shell 运行由 *File* 参数标识的脚本文件，包含任何指定的参数。指定的脚本文件必须有读许可权；忽略任何 **setuid** 和 **setgid** 设置。然后 shell 读取命令。如果使用 **-c** 或 **-s** 标志，则不要指定脚本。

Bourne shell 环境

在命令执行的开始时为命令知晓的所有变量（及其关联的值）组成其*环境*。此环境包含命令从其父进程继承的变量和在调用命令的命令行上作为关键字参数指定的变量。

shell 向其子进程传递作为内置 **export** 命令的自变量命名的变量。此命令将命名的变量同时放置在 shell 及其所有将来的子进程的环境中。

关键字参数是以赋值的格式出现的“变量-值”对，通常在命令行的过程名之前（但另见 **set** 命令的标志）。这些变量将放置在所调用的过程的环境中。

例如，请思考以下过程，其显示两个变量的值（保存在命名为 key_command 的命令文件中）：

```
# key_command
echo $a $b
```

以下命令行产生显示为如下的输出:

```
Input                                Output
a=key1   b=key2  key_command        key1 key2
a=tom    b=john  key_command        tom john
```

过程的关键字参数不包含在存储在 $# 中的参数计数中。

过程可以访问其环境中的任何变量的值。但是，如果它更改任何这些值，则更改将不在 shell 环境中反映。更改对于正在讨论的过程是本地的。要将更改放置在进程传递到其子进程的环境中，则必须导出该进程中的新值。

要获取可从当前 shell 导出的变量的列表，请输入:

```
export
```

按下 Enter 键。

要获取当前 shell 的只读变量的列表，请输入:

```
readonly
```

按下 Enter 键。

要获取当前环境中的“变量-值”对的列表，请输入:

```
env
```

按下 Enter 键。

有关用户环境的更多信息，请参阅第 126 页的『/etc/environment 文件』

受限 shell

受限 shell 用于设置登录名和执行环境，其能力比常规 Bourne shell 的能力更受到控制。**Rsh** 或 **bsh -r** 命令打开受限 shell。这些命令的行为等同于 **bsh** 命令的行为，除不允许以下操作外:

- 更改目录（使用 **cd** 命令）
- 设置 **PATH** 或 **SHELL** 变量的值
- 指定包含斜杠（/）的路径或命令名
- 重定向输出

如果受限 shell 确定要运行的命令是 shell 步骤，则它使用 Bourne shell 运行命令。这样，就可能向最终用户提供访问 Bourne shell 的全部功能的 shell 步骤，但强加受限的命令菜单。这种情况假定最终用户在同一个目录中不具有写和执行许可权。

如果在启动 Bourne shell 时指定 *File* [*Parameter*] 参数，则 shell 运行 *File* 参数标识的脚本文件，包含任何指定的参数。指定的脚本文件必须具有读许可权。忽略脚本文件的任何 **setuid** 和 **setgid** 设置。然后 shell 读取命令。如果使用 **-c** 或 **-s** 标志，则不指定脚本文件。

当使用 **Rsh** 命令启动时，shell 在解释 **.profile** 和 **/etc/environment** 文件后强制执行。因此，**.profile** 文件的编写者通过执行设置操作和让用户在相应的目录（可能不是登录目录）中对用户操作进行完全控制。管理员可在 **/usr/rbin** 目录中创建命令目录，**Rsh** 命令可通过更改 PATH 变量使用它以包含该目录。如果使用 **bsh -r** 命令启动，则 shell 在解释 **.profile** 文件时应用限制。

当使用名称 **Rsh** 调用时，受限 shell 读用户的 **.profile** 文件（**$HOME/.profile**）。执行这一操作时它作为常规 Bourne shell 工作，除中断导致立即退出而不是返回命令级外。

Bourne shell 命令

当您在 Bourne shell 中发出命令时，它首先对命令求值，并做出所有表示的替换。然后运行命令，条件是:

- 命令名是 Bourne shell 特殊内置命令。

  或
- 命令名匹配已定义的函数的名称。如果是这种情况，则 shell 将位置参数设置为函数的参数。

如果命令名既不匹配内置命令也不匹配已定义的函数的名称，并且命令命名已编译的（二进制）程序的可执行文件，则 shell（作为父）产生一个立即运行程序的新（子）进程。如果文件标记为可执行文件，但不是已编译程序，则 shell 假定它是一个 shell 步骤。在这种情况下，shell 生成其另一个实例（子 *shell*），来读文件和执行包含在其中的命令。shell 还运行子 shell 中加括弧的命令。对于最终用户，已编译程序与 shell 步骤完全一样运行。shell 通常搜索文件系统目录中的命令，顺序是:

1. **/usr/bin**
2. **/etc**
3. **/usr/sbin**
4. **/usr/ucb**
5. **$HOME/bin**
6. **/usr/bin/X11**
7. **/sbin**
8. 当前目录

shell 依次搜索每个目录，如果没有找到命令就继续搜索下一个目录。

> **注**: **PATH** 变量确定 shell 搜索目录的顺序。可以通过复位 **PATH** 变量来更改已搜索目录的特定顺序。

如果在运行命令时给出一个特定路径名（例如，**/usr/bin/sort**），则 shell 不搜索除您指定的那个以外的任何目录。如果命令名包含斜杠（/），则 shell 不使用搜索路径。

可以给出一个以根目录开始的全路径名（如 **/usr/bin/sort**）。还可以指定相对于当前目录的路径名。例如，如果您指定:

```
bin/myfile
```

则 shell 在当前目录中查找名为 bin 的目录，并在该目录中查找文件 myfile。

注: 受限的 shell 不运行包含 /（斜杠）的命令。

shell 记住每个已执行命令在搜索路径中的位置（以避免以后不必要的 **exec** 命令）。如果它在相对目录（它的名称不以 / 开始）中找到命令，则无论何时当前目录更改，shell 必须重新确定命令的位置。每次更改 **PATH** 变量或运行 **hash -r** 命令时，shell 都将忽略所有已记住的位置。

本节讨论以下内容:

- 『引证字符』
- 『信号处理』
- 第 178 页的『Bourne shell 内置命令』
- 第 181 页的『Bourne shell 中的命令替换』

引证字符

许多字符对 shell 都有特殊的含义。有时您要取消该含义。圈起字符串的单引号（'）和双引号（"），或单字符前的反斜杠 (\) 允许您取消字符的含义。

所有字符（除单引号圈起的外）逐字地操作，并且除去任何特殊含义。因此，命令:

```
stuff='echo $? $*; ls * | wc'
```

将文字字符串 echo $? $*; ls * | wc 指定给变量 stuff。shell 不执行 **echo**、**ls** 和 **wc** 命令或展开 $? 和 $* 变量及 *（星号）特殊字符。

在双引号中，$（美元符）、`（反引号）和 "（双引号）字符的特殊含义保持有效，而所有其它字符逐字地操作。因此，在双引号中，命令和变量替换发生。此外，引号不影响是引证字符串的一部分的命令替换中的命令，因此那里的字符保持其特殊含义。

考虑以下顺序:

```
ls *
file1 file2 file3
message="This directory contains `ls * ` "
echo $message
This directory contains file1 file2 file3
```

这显示命令替换中的 *（星号）特殊字符是展开的。

要隐藏双引号中的 $（美元符）、'（反引号）和 "（双引号）字符的特殊含义，将 \ (反斜杠) 置于这些字符前。当您不使用双引号时，将反斜杠放在字符前等价于将字符放在单引号中。因此，就在换行字符前的反斜杠（即，反斜杠在一行的结尾）隐藏换行字符，并允许您在下一个物理行继续命令行。

信号处理

如果该命令以 &（和符号）终止；即，如果它在后台运行，则 shell 忽略该调用命令的 **INTERRUPT** 和 **QUIT** 信号。否则，信号将具有 shell 从其父那里继承的值，例外是 SEGMENTATION VIOLATION 信号。有关更多信息，请参阅 Bourne shell 内置 **trap** 命令。

Bourne shell 复合命令

复合命令是以下之一:

- 流水线（由 |（管道）符号分隔的一个或多个简单命令）
- 简单命令列表
- 以保留字开始的命令
- 以控制运算符 (（左圆括号）开始的命令。

除非另有说明，否则由复合命令返回的值是执行的最后一个简单命令返回的值。

保留字

仅当在命令的第一个字出现并不带引号时，以下保留字才可识别:

```
for           do            done
case          esac
if            then           fi
elif          else
while         until
{             }
(             )
```

for *Identifier* [**in** *Word* . . .] **do** *List* **done**	将 *Identifier* 参数设置为 *Word* 参数指定的（一个或多个）字（每次一个），并运行 *List* 参数中指定的命令。如果省略 **in** *Word* . . .，则 **for** 命令为设置的每个位置参数运行 *List* 参数，并在使用了所有的位置参数时处理结束。
case *Word* **in** *Pattern* [|*Pattern*] . . . **)** *List***;;** [*Pattern* [|*Pattern*] . . . **)** *List***;;**] **. . . esac**	运行 *List* 参数中指定的命令，这些命令与匹配 *Word* 参数的值的第一个 *Pattern* 参数关联。使用与文件名替换使用的模式相同的字符匹配标志法，除 /（斜杠）、前导 .（点）或紧跟斜杠的点不需要显式地匹配。
if *List* **then** *List* [**elif** *List* **then** *List*] . . . [**else** *List*] **fi**	在 **if** 命令后运行 *List* 参数中指定的命令。如果此命令返回一个零出口值，则 shell 在第一个 **then** 命令后运行 *List* 参数。否则，它在 **elif** 命令（如果存在）后运行 *List* 参数。如果此出口值是零，则 shell 在下一个 **then** 命令后运行 List 参数。如果此命令返回一个非零出口值，则 shell 在 **else** 命令（如果存在）后运行 *List* 参数。如果未执行 **else** *List* 或 **then** *List*，则 **if** 命令返回一个零出口值。
while *List* **do** *List* **done**	在 **while** 命令后运行 *List* 参数中指定的命令。如果 **while** *List* 中最后一个命令的出口值是零，则 shell 在 **do** 命令后运行 *List* 参数。它继续在列表中循环，直到 **while** *List* 中最后一个命令的出口值非零。如果未执行 **do** *List* 中的命令，则 **while** 命令返回一个零出口值。
until *List* **do** *List* **done**	在 **until** 命令后运行 *List* 参数中指定的命令。如果 **until** *List* 中最后一个命令的出口值非零，则在 **do** 命令后运行 *List*。继续在列表中循环，直到 **until** *List* 中最后一个命令的出口值是零。如果未执行 **do** *List* 中的命令，则 **until** 命令返回一个零出口值。
(*List* **)**	在子 shell 中运行 *List* 参数中的命令。
{ *List***; }**	在当前 shell 进程中运行 *List* 参数中的命令，而不启动子 shell。
Name **() {** *List* **}**	定义 *Name* 参数所引用的函数。函数的主体是由 *List* 参数指定的大括号之间的命令列表。

Bourne shell 内置命令

特殊命令内置在 Bourne shell 中，并在 shell 进程中运行。除非另有指示，否则输出写入到文件描述符 1（标准输出），并且如果命令不包含任何语法错误则退出状态为 0（零）。允许输入和输出重定向。

有关这些命令的字母顺序列表，请参考第 187 页的『Bourne shell 内置命令的列表』。

对以下特殊命令的处理与其它特殊内置命令有所不同:

: (colon)	**exec**	**shift**
. (dot)	**exit**	**times**
break	**export**	**trap**
continue	**readonly**	**wait**
eval	**return**	

Bourne shell 如下处理这些命令:

- 在命令之前的关键字参数赋值列表在命令完成时保持有效。
- 在参数赋值后处理 I/O 重定向。
- shell 脚本中的错误导致脚本停止处理。

特殊命令描述

Bourne shell 提供以下特殊内置命令:

内置命令

: 返回零出口值。

. *File* 从 *File* 参数读和运行命令，并返回。不启动子 shell。shell 使用由 **PATH** 变量指定的搜索路径来查找包含指定文件的目录。

break [*n*] 从封闭的 **for**、**while** 或 **until** 命令循环退出（如果有）。如果指定 *n* 变量，则 **break** 命令中断由 *n* 变量指定的级别号。

continue [*n*] 继续封闭的 **for**、**while** 或 **until** 命令循环的下一个迭代。如果您指定 *n* 变量，则命令在第 *n* 个封闭循环处继续。

cd *Directory*] 更改当前目录到 *Directory*。如果未指定 *Directory*，则使用 **HOME** shell 变量的值。**CDPATH** shell 变量定义 *Directory* 的搜索路径。**CDPATH** 是以冒号分隔的备用目录名称的列表。空路径名指定当前目录（它是缺省路径）。此空路径名紧跟在赋值中等号之后或出现在路径列表中冒号定界符之间的任何其它位置。如果 *Directory* 以 /（斜杠）开始，则 shell 不使用搜索路径。否则，shell 搜索 **CDPATH** shell 变量中的每个目录。

注: 受限的 shell 无法运行 **cd** shell 命令。

echo *String . . .*] 将字符串写入标准输出。有关用法和参数信息，请参考 **echo** 命令。不支持 **-n** 标志。

eval [*Argument . . .*] 将自变量作为输入读入 shell，并运行（一个或多个）结果命令。

exec [*Argument . . .*] 运行 *Argument* 参数指定的命令代替此 shell，并不创建新进程。输入和输出自变量可出现，并且如果不出现其它自变量，将导致 shell 输入和输出被修改。这对于您的登录 shell 不推荐使用。

exit [*n*] 导致 shell 退出，出口值由 *n* 参数指定。如果省略此参数，则出口值是最后一个执行命令的出口值（Ctrl-D 按键序列也导致 shell 退出）。*n* 参数的值可从 0 到 255，包括 0 和 255。

export [*Name . . .*] 为自动导出到后续执行的命令的环境标记指定的名称。如果不指定 *Name* 参数，**export** 命令将显示在此 shell 中导出的所有名称的列表。无法导出函数名。

hash [**-r**][*Command . . .*] 查找并记住每个指定的 *Command* 在搜索路径中的位置。 **-r** 标志使 shell 忘记所有位置。如果未指定标志或任何命令，则 shell 以下列格式显示关于记住的命令的信息:

```
Hits   Cost   Command
```

Hits 表示命令已由 shell 进程运行的次数。Cost 是在搜索路径中定位命令所要求的工作的度量。Command 显示每个指定命令的路径名。某些情况要求对命令的存储位置进行重新计算；例如，当前目录更改时的相对路径名的位置。可能要这样做的命令用一个 Hits 信息旁的 *（星号）表明。重新计算完成时 Cost 将增加。

pwd 显示当前目录。有关命令选项的讨论，请参考 **pwd** 命令。

read [*Name . . .*] 从标准输入读一行。将行中的第一个字指定给第一个 *Name* 参数，第二个字指定给第二个 *Name* 参数，等等，剩余的字都指定给最后一个 *Name* 参数。此命令返回值 0，除非它遇到文件结束符字符。

readonly [*Name . . .*] 将 *Name* 参数指定的名称标记为只读。名称的值无法重新设置。如果不指定任何 *Name*，则 **readonly** 命令将显示所有只读名称的列表。

return [*n*] 使函数以返回值 *n* 退出。如果未指定 *n* 变量，则函数返回最后一个在该函数中执行的命令的状态。仅当此命令在 shell 函数中运行时才有效。

内置命令

set [*Flag* [*Argument*] . . .] 设置一个或多个以下标志:

-a 标记导出所有变量到执行赋值的位置。如果赋值在命令名之前，则导出属性仅对该命令执行环境有效，除了当赋值在某一特殊内置命令之前时。在这种情况下，导出属性在内置命令完成后仍存留。如果赋值不在命令名之前，或如果赋值是 **getopts** 或 **read** 命令的操作结果，则导出属性将存留到变量取消设置为止。

-e 如果对于一个命令所有以下条件存在，则立即退出:

- 它以一个大于 0（零）的返回值退出。
- 它不是 **while**、**until** 或 **if** 命令的复合列表的一部分。
- 它不使用 AND 或 OR 列表测试。
- 它不是一个前有 !（感叹号）保留字的流水线。

-f 禁用文件名替换。

-h 在定义函数时，定位并记住在函数中调用的命令。（通常这些命令在执行函数时定位; 请参阅 **hash** 命令。）

-k 将所有关键字参数都放入命令的环境，而不只是命令名前的那些关键字参数。

-n 读命令但不运行它们。要检查 shell 脚本语法错误，请使用 **-n** 标志。

-t 在读和执行一个命令后退出。

-u 将取消设置的变量作为错误处理，并在执行变量替换时立即退出。交互式 shell 不退出。

-v 读 shell 输入行时显示它们。

-x 在命令运行前显示命令及其自变量。

— 不更改任何标志。在将 **$1** 位置参数设置为一个以连字符（-）开始的字符串时有用。

使用加号（+）而非连字符（-）来取消设置标志。还可在 shell 命令行上指定这些标志。**$-** 特殊变量包含标志的当前设置。

set 命令的任何 *Argument* 成为位置参数，并按顺序指定为 **$1**、**$2**，等等。如果未指定 *flag* 或 *Argument*，则 **set** 命令显示当前 shell 变量的所有名称和值。

shift [*n*] 向左移位命令行自变量; 即，通过废弃 **$1** 的当前值并将 **$2** 的值指定给 **$1**，将 **$3** 的值指定给 **$2**，等等，来重新指定位置参数的值。如果有 9 个以上命令行自变量，则第 10 个指定给 **$9**，而任何其它剩下的仍未指定（直到在另一个 **shift** 后）。如果只有 9 个或更少自变量，则 **shift** 命令取消设置具有值的最大编号的位置参数。

$0 位置参数从不移位。**shift** *n* 命令是指定 *n* 个连续移位的速记标志法。*n* 参数的缺省值是 1。

test *Expression* | [*Expression*] 求值条件表达式。有关命令标志和参数的讨论，请参考 **test** 命令。**bsh** 中的内置 test 命令不支持 **-h** 标志。

times 为从 shell 运行的进程显示累加的用户和系统次数。

trap [*Command*] [*n*] . . . 当 shell 接收到由 *n* 参数指定的（一个或多个）信号时，运行由 *Command* 参数指定的命令。**trap** 命令以信号号码的顺序运行。在当前 shell 的入口上被忽略的信号上设置陷阱的任何尝试都是无效的。

注: shell 在设置陷阱时扫描一次 *Command* 参数，在采用陷阱时再读一次。

如果不指定命令，则由 *n* 参数指定的所有陷阱都被重新设置为其当前值。如果指定空字符串，则此信号被 shell 和它调用的命令忽略。如果 *n* 参数是零（0），则指定的命令在您从 shell 中退出时运行。如果未指定命令或信号，则 **trap** 命令显示与每个信号号码关联的命令的列表。

type [*Name* . . .] 对于每个指定的 *Name*，表示 shell 如何将它解释成命令名。

内置命令	
ulimit [**-HS**] [**-c** | **-d** | **-f** | **-m** | **-s** | **-t**] [*limit*]	显示或调整指定的 shell 资源。可个别地或作为组来显示 shell 资源设置。缺省方式是作为组来显示软设置或下界的资源设置。 shell 资源的设置取决于当前 shell 的有效用户标识。仅在当前 shell 的有效用户标识是 root 时，才可设置资源的硬级别。如果您不是 root 用户而又试图设置资源的硬级别，则将会出错。缺省情况下，root 用户设置某一特殊资源的硬限制和软限制。因此 root 用户在使用 **-S**、**-H** 或限制设置的缺省标志用法时要格外小心。除非您是 root 用户，否则只能设置资源的软限制。在非 root 用户降低了限制后，它无法再增加，即使要回复到其原始系统限制也不行。 要设置资源限制，请选择相应的标志以及新资源的限制值，该限制值应为一个整数。一次只能设置一个资源限制。如果指定了多个资源标志，则会接收到未定义的结果。缺省情况下，在命令行上只有一个新值的 **ulimit** 设置 shell 的文件大小。**-f** 标志的使用是可选的。 可以指定以下 **ulimit** 命令标志: **-c** 设置或显示 shell 的核心段。 **-d** 设置或显示 shell 的数据段。 **-f** 设置或显示 shell 的文件大小。 **-H** 设置或显示硬资源限制（仅 root 用户） **-m** 设置或显示 shell 的内存。 **-s** 设置或显示 shell 的堆栈段。 **-S** 设置或显示软资源限制。 **-t** 设置或显示 shell 的最大 CPU 时间。
umask [*nnn*]	确定文件许可权。当文件创建时，此值以及创建进程的许可权确定文件的许可权。缺省值是 022。当不输入值时，umask 显示当前值。
unset [*Name . . .*]	除去由 *Name* 参数指定的每个名称的对应变量或函数。无法取消设置 **PATH**、**PS1**、**PS2**、**MAILCHECK** 和 **IFS** shell 变量。
wait [*n*]	等待其进程号由 *n* 参数指定的子进程退出，然后返回该进程的退出状态。如果未指定 *n* 参数，则 shell 等待所有当前活动的子进程，并且返回值为 0。

Bourne shell 中的命令替换

命令替换允许您捕捉任何命令的输出，作为另一个命令的自变量。当您将一个命令行放在反引号（``）中时，shell 首先运行命令，然后用输出替换整个表达式（包括反引号）。此功能经常用于给 shell 变量赋值。例如，语句:

```
today=`date`
```

将代表当前日期的字符串指定给 `today` 变量。以下赋值将当前目录中的文件数保存在 `files` 变量中。

```
files=`ls | wc -l`
```

可以对任何写到标准输出的命令执行命令替换。

要嵌套命令替换，请在每个嵌套的反引号前加一个反斜杠 (\)，如下:

```
logmsg=`echo Your login directory is \`pwd\``
```

还可以通过使用 **read** 特殊命令间接地给 shell 变量赋值。此命令从标准输入（通常是键盘）取出一行，并将该行上连续的字指定给任何命名的变量。例如:

```
read first init last
```

取出以下格式的输入行:

```
J. Q. Public
```

与您输入以下命令的效果相同:

```
first=J. init=Q. last=Public
```

read 特殊命令将任何剩余的字指定给最后一个变量。

Bourne shell 中的变量和文件名替换

Bourne shell 允许您执行变量和文件名替换。

以下各节讨论在 Bourne shell 中创建和替换变量:

- 『Bourne shell 中的变量替换』
- 『用户定义的变量』
- 第 185 页的『条件替换』
- 第 186 页的『位置参数』
- 第 186 页的『Bourne shell 中的文件名替换』
- 第 186 页的『字符类』

Bourne shell 中的变量替换

Bourne shell 有几种创建变量的机制（将字符串值指定给名称）。某些变量、位置参数和关键字参数通常仅在命令行上设置。其它变量就是您或 shell 可将字符串值指定给它的名称。

用户定义的变量

shell 识别字符串值可指定为的字母数字变量。要将字符串值指定给名称，请输入:

```
Name=String
```

按下 Enter 键。

名称是以下划线或字母开头的字母、数字和下划线的序列。要使用已指定给变量的值，请将一个美元符（$）添加到其名称的开头。这样，*$Name* 变量产生由 *String* 变量指定的值。请注意，赋值语句中等号（=）的任一边都无空格。（位置参数不能出现在赋值语句中。它们只能按第 186 页的『位置参数』中的描述设置。）您可以将多个赋值放在命令行上，但记住 shell 从右至左执行赋值。

如果用双引号或单引号（" 或 '）将 *String* 变量圈起，则 shell 不将字符串中的空白、制表符、分号和换行字符作为字定界符处理，而将它们逐字嵌入字符串。

如果用双引号（"）将 *String* 变量圈起，则 shell 仍可识别字符串中的变量名，并执行变量替换；即，它将对位置参数的引用和前有美元符（$）的其它变量名替换为其对应的值（如果有）。shell 还在用双引号圈起的字符串中执行命令替换。

如果用单引号（'）圈起 *String* 变量，则 shell 不替换字符串中的变量或命令。以下序列说明此差异:

```
您:             num=875
                number1="Add $num"
                number2='Add $num'
                echo $number1
系统:           Add 875
您:             echo $number2
系统:           Add $num
```

shell 在变量替换后不重新解释赋值中的空白。这样，以下赋值使 $first 和 $second 具有相同的值:

```
first='a string with embedded blanks'
second=$first
```

当引用一个变量时，可以将变量名（或指定位置参数的数字）圈起在 { } 中以定界变量名，与任何后面的字符串分开。特别地，如果紧跟名称的字符是字母、数字或下划线，且变量不是位置参数，则大括号是必需的:

```
您:             a='This is a'
                echo "${a}n example"
系统:           This is an example
您:             echo "$a test"
系统:           This is a test
```

请参阅第 185 页的『条件替换』以了解变量替换中大括号的不同使用。

shell 使用的变量

shell 使用以下变量。尽管 shell 设置了它们中的一些，您仍可以设置或重新设置所有变量:

CDPATH 指定 **cd**（更改目录）命令的搜索路径。

HOME 表示您的*登录目录*的名称，该目录在登录完成后成为当前目录。**登录**程序初始化此变量。**cd** 命令使用 **$HOME** 变量的值作为其缺省值。在 shell 步骤中使用此变量（而非显式路径名）允许步骤从不同的目录运行而无须改动。

IFS 是 IFS（内部字段分隔符）的字符，shell 在空白解释期间使用的字符；请参阅第 185 页的『空白解释』。 shell 初始地设置 **IFS** 变量以包含空白、制表符和换行字符。

LANG 确定当 **LC_ALL** 变量和对应的环境变量（以 **LC_** 开头）都未指定语言环境时，用于语言环境类别的语言环境。有关语言环境的更多信息，请参阅 *AIX 5L Version 5.2 National Language Support Guide and Reference* 中的 “Locale Overview”。

LC_ALL 确定用于覆盖由 **LANG** 环境变量的设置或任何以 **LC_** 开头的环境变量指定的任何语言环境类别值的语言环境。

LC_COLLATE 定义当对名称进行排序或字符范围发生在模式中时使用的整理序列。

LC_CTYPE 确定用于将文本数据字节序列解释成字符的语言环境（即，自变量和输入文件中的单字节字符与多字节字符），哪些字符定义为字母（**alpha** 字符类），以及模式匹配中字符类的行为。

LC_MESSAGES 确定以哪种语言写消息。

LIBPATH 指定共享库的搜索路径。

LOGNAME 指定您的登录名，在 **/etc/profile** 文件中标记为 **readonly**。

MAIL 表示邮件系统用于检测新邮件到达的文件的路径名。如果设置此变量，则 shell 将定期地检查此文件的修改时间，并且如果该时间已更改且文件长度大于 0 则显示 **$MAILMSG** 的值。在 **.profile** 文件中设置 **MAIL** 变量的值。通常 **mail** 命令的用户指定给它的值是 **/usr/spool/mail/$LOGNAME**。

MAILCHECK 再次检查由 **MAILPATH** 或 **MAIL** 变量指定的文件中邮件的到达前，shell 允许经过的秒数。缺省值是 600 秒（10 分钟）。如果将 **MAILCHECK** 变量设置为 0，则 shell 在每个提示符前检查。

MAILMSG 邮件通知消息。如果显式地将 **MAILMSG** 变量设置为空字符串（MAILMSG=""），则不显示消息。

MAILPATH 用冒号分隔的文件名的列表。如果设置此变量，shell 将通知您在列表中所指定任何文件中邮件的到达。可以在每个文件名后跟一个 % 和当邮件到达时要显示的消息。否则，shell 使用 **MAILMSG** 变量的值或（缺省时）消息 [YOU HAVE NEW MAIL].

> **注**: 当设置 **MAILPATH** 变量时，将检查这些文件，而不检查由 **MAIL** 变量设置的文件。要检查由 **MAILPATH** 变量设置的文件和由 **MAIL** 变量设置的文件，在 **MAILPATH** 文件的列表中指定 **MAIL** 文件。

PATH 命令的搜索路径，它是由冒号分隔的目录路径名的有序列表。当 shell 查找命令时，它以指定的顺序搜索这些目录。列表中任何位置的空字符串代表当前目录。

PATH 变量通常在 **/etc/environment** 文件中初始化，通常初始化为 **/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin**。可以重新设置此变量以适合您自己的需要。**.profile** 文件中提供的 **PATH** 变量还包含 **$HOME/bin** 和您的当前目录。

如果您要在搜索标准系统目录前搜索一个特定于项目的命令目录，例如 **/project/bin**，设置 **PATH** 变量如下:

```
PATH=/project/bin:$PATH
```

将您的 **PATH** 变量设置为一个非缺省值的值的最好位置是您的 **$HOME/.profile** 文件。如果在受限 shell 下执行命令，则无法重新设置 **PATH** 变量。

PS1 用作主系统提示符的字符串。交互式 shell 在期待输入时显示此提示符字符串。对于非 root 用户，**PS1** 变量的缺省值是 $ 后跟一个空白空格。

PS2 次提示符字符串的值。如果 shell 在其输入中遇到换行字符时期待更多输入，则它以 **PS2** 变量的值提示。**PS2** 变量的缺省值是 > 后跟一个空白空格。

SHACCT 您拥有的文件的名称。如果设置此变量，则对于每个执行的 shell 脚本，shell 在文件中写一个记帐记录。可以使用记帐程序（如 **acctcom** 和 **acctcms**）来分析收集的数据。

SHELL 保留在环境中的 shell 的路径名。此变量应由每个受限登录的 **$HOME/.profile** 文件设置和导出。

TIMEOUT shell 退出前保留非活动的分钟数。如果此变量设置为一个大于零（0）的值，则当在发出 **PS1** 提示符后未在指示的秒数内输入命令，shell 退出。（注意，shell 可以使用不能超过此值的最大边界来编译。）值零表示无时间限制。

预定义的特殊变量

几个变量有特殊的含义。以下变量仅由 shell 设置。

$@ 展开位置参数，以 **$1** 开始。每个参数都用空格分隔。

如果用 ″ ″ 圈起 **$@**, 则 shell 认为每个位置参数是独立的字符串。如果不存在位置参数，则 Bourne shell 将语句展开成一个没有引起的空字符串。

$* 展开位置参数，以 **$1** 开始。shell 使用 **IFS** 变量值的第一个字符分隔每个参数。

如果用 ″ ″ 圈起 **$&**, 则 shell 在双引号中包含位置参数值。每个值都用 **IFS** 变量的第一个字符分隔。

$#** 指定传递给 shell 的位置参数的数目，不计数 shell 步骤名称本身。这样 **$# 变量产生设置的最大编号位置参数的数目。此变量的一个主要使用是检查存在要求的自变量数目。只有位置参数 **$0** 到 **$9** 可以通过 shell 访问。有关更多信息，请参阅第 186 页的『位置参数』。

$?** 指定上次执行命令的出口值。其值是一个十进制字符串。大多数命令返回值 0 以表示成功完成。shell 自己返回 **$? 变量的当前值作为其出口值。

$$ 标识当前进程的进程号。因为进程号在所有现有字符串中唯一，所以此字符串经常用于生成临时文件的唯一名称。

下例说明仅用于该目的的，在目录中创建临时文件的推荐示例:

```
temp=/tmp/$$
ls >$temp
.
.
.
rm $temp
```

$! 指定上一个使用 **&** 终端符在后台运行的进程的进程号。

$- 由当前在 shell 中设置的执行标志的名称组成的字符串。

空白解释

shell 在执行变量替换和命令替换后，扫描结果以获取内部字段分隔符（在 **IFS** shell 变量中定义的那些）。shell 在找到一个或多个这些字符的每个地方，将行分割成不同的字，并以一个空格分隔每个不同的字。然后它保留显式空自变量（"" 或 ''）并废弃隐式空自变量（那些由没有值的参数的产生的）。

条件替换

通常，shell 用指定给 *Variable* 变量（如果有一个）的字符串值替换表达式 $*Variable*。但有一个允许*条件替换*的特殊标志法，这取决于变量是否设置和 / 或非空。根据定义，如果一个变量曾经指定过值，则它就已设置。变量的值可以是空字符串，您可以用任一以下方式将其指定给变量:

```
A=

bcd=""

Efg=''          将空字符串指定给 A、bcd 和 Efg。
set '' ""       将第一个和第二个位置参数设置为空字符串，并取消设置所有其它位置参数。
```

以下是可用于执行条件替换的可用表达式的列表:

${*Variable***-** *String***}**
如果设置了变量，则替换 *Variable* 值代替此表达式。否则，使用 *String* 值替换此表达式。

${*Variable***:-***String***}**
如果设置了变量并且非空，则替换 *Variable* 值代替此表达式。否则，使用 *String* 值替换此表达式。

${*Variable***=***String***}**
如果设置了变量，则替换 *Variable* 值代替此表达式。否则，将 *Variable* 值设置为 *String* 值，然后替换 *Variable* 值代替此表达式。您无法以此方式指定值给位置参数。

${*Variable***:=***String***}**
如果设置了变量并且非空，则替换 *Variable* 值代替此表达式。否则，将 *Variable* 值设置为 *String* 值，然后替换 *Variable* 值代替此表达式。您无法以此方式指定值给位置参数。

${*Variable***?***String***}**
如果设置了变量，则替换 *Variable* 值代替此表达式。否则，显示以下格式的消息:

```
Variable: String
```

并从当前 shell 退出（除非 shell 是登录 shell）。如果没有为 *String* 变量指定值，则 shell 显示以下消息:

```
Variable: parameter null or not set
```

${*Variable***:?***String***}**
如果设置了变量并且非空，则替换 *Variable* 值代替此表达式。否则，显示以下格式的消息:

```
Variable: String
```

并从当前 shell 退出（除非 shell 是登录 shell）。如果不指定 *String* 变量值，则 shell 显示以下消息:

```
变量: 参数空或未设置
```

${*Variable***+***String***}**
如果设置了变量，则替换 *String* 值代替此表达式。否则，替换空字符串。

${*Variable***:+***String***}**
如果设置了变量并且非空，则替换 *String* 值代替此表达式。否则，替换空字符串。

在条件替换中，shell 不对 *String* 变量进行求值，直到 shell 使用此变量作为替换的字符串。这样，在下例中，仅当 d 未设置或为空时，shell 执行 **pwd** 命令:

```
echo ${d:-`pwd`}
```

位置参数

当您运行 shell 步骤时，shell 隐式地创建位置参数，这些位置参数根据每个字在命令行上的位置在命令行上引用每个字。位置 0 中的字（步骤名称）称为 **$0**，下一个字（第一个参数）称为 **$1**，等等，直到 **$9**。要引用编号大于 9 的命令行参数，请使用内置 **shift** 命令。

可以通过使用内置 **set** 命令显式地重新设置位置参数的值。

> **注**: 当未指定位置的自变量时，其位置参数设置为空。位置参数是全局的，并且可以传递到嵌套的 shell 步骤。

Bourne shell 中的文件名替换

命令参数经常是文件名。可以自动产生文件名的列表，作为命令行上的参数。要这样做，请指定 shell 识别为模式匹配字符的字符。当命令包含这样的字符时，shell 使用目录中的文件名替换它。

> **注**: Bourne shell 不支持基于等价字符分类的文件名扩展。

这样的模式中的大多数字符与自身匹配，但也可以在您的模式中使用一些特殊的模式匹配字符。这些特殊字符如下:

*****	匹配任何字符串，包括空字符串
?	匹配任何单个字符
[. . .]	匹配圈起在方括号中的任一字符
[! . . .]	匹配方括号中的任何字符，除跟在感叹号后的一个字符

在方括号中，以 - 分隔的一对字符，指定该字符对的包含范围中词典编纂地所有字符集，根据字符值的二进制顺序。

模式匹配有一些限制。如果文件名的第一个字符是点（.），则它只能由也以点开头的模式匹配。例如，* 匹配文件名 **myfile** 和 **yourfile**，而不匹配文件名 **.myfile** 和 **.yourfile**。要匹配这些文件，请使用诸如以下的模式:

```
.*file
```

如果模式不匹配任何文件名，则模式本身作为尝试的匹配的结果返回。

文件名和目录名应不包含字符 *, ?, [或], 因为它们可能在模式匹配尝试期间导致无限递归（即，无限循环）。

字符类

还可以使用字符类来匹配文件名，如下:

```
[[:charclass:]]
```

此格式指示系统匹配属于指定类的任何单个字符。定义的类对应于 **ctype** 子例程，如下:

字符类	定义
alnum	字母数字字符
alpha	大写和小写字母
blank	空格或水平制表符
cntrl	控制字符
digit	数字
graph	图形字符
lower	小写字母
print	可打印字符
punct	标点符号字符

字符类	定义
space	空格、水平制表符、回车、新行、垂直制表符或换页字符
upper	大写字符
xdigit	十六进制数字

Bourne shell 中的输入和输出重定向

一般地，大多数命令不知道它们的输入或输出是否与键盘、显示屏幕或文件关联。因此，命令可便利地在键盘或流水线使用。

以下重定向选项可以在简单命令中的任何位置出现。它们还可以在命令之前或之后，但不传递到命令。

<*File*	将指定文件用作标准输入。
>*File*	将指定文件用作标准输出。如果文件不存在，则创建它；否则，将它截断为零长度。
> >*File*	将指定文件用作标准输出。如果文件不存在，则创建它；否则，将输出添加至文件结尾。
<<[**-**]*eofstr*	将来自 *eofstr* 变量的所有行作为标准输入读取，直到仅包含 *eofstr* 的行，或直到一个文件结束符字符。如果 *eofstr* 变量中的任何字符加有引号，则 shell 不展开或解释输入行中的任何字符。否则，它执行变量和命令替换，并忽略引证的换行字符 （**\newline**）。使用 \ 来引用 *eofstr* 变量中或输入行中的字符。 如果将 - 添加到 << 重定向选项，则从 *eofstr* 变量和从输入行中舍去所有前导制表符。
<&*Digit*	将标准输入与 *Digit* 变量指定的文件描述符关联。
>&*Digit*	将标准输出与 *Digit* 变量指定的文件描述符关联。
<&-	关闭标准输入。
>&-	关闭标准输出。

注: 受限的 shell 不允许输出重定向。

有关重定向的更多信息，请参阅第 45 页的第 5 章，『输入和输出重定向』。

Bourne shell 内置命令的列表

:	返回零出口值
.	读和执行来自文件参数的命令，然后返回。
break	从封闭的 **for**、**while** 或 **until** 命令循环退出（如果有）。
cd	更改当前目录到指定的目录。
continue	继续封闭的 **for**、**while** 或 **until** 命令循环的下一个迭代。
echo	将字符串写入标准输出。
eval	将自变量作为输入读入 shell，并执行结果命令。
exec	执行 *Argument* 参数指定的命令，而不是此 shell，并不创建新进程。
exit	退出由 *n* 参数指定其退出状态的 shell。
export	为自动导出到后续执行的命令的环境标记名称。
hash	查找并记住指定的命令在搜索路径中的位置。
pwd	显示当前目录。
read	从标准输入读一行。
readonly	将 *Name* 参数指定的名称标记为只读。
return	导致函数退出，并有指定的返回值。
set	控制各种参数到标准输出的显示。
shift	左移命令行自变量。
test	求值条件表达式。
times	为从 shell 运行的进程显示累加的用户和系统次数。

trap	当 shell 接收到指定的信号时运行指定的命令。
type	解释 shell 如何将特定名称解释成命令名。
ulimit	显示或调整分配的 shell 资源。
umask	确定文件许可权。
unset	除去对应于指定名称的变量或函数。
wait	等待指定子进程结束并报告其终止状态。

C shell

C shell 是交互式命令解释器和命令编程语言。它使用的语法类似于 C 编程语言。**csh** 命令启动 C shell。

当您登录时，**csh** 命令首先搜索系统范围设置文件 **/etc/csh.cshrc**。如果有设置文件，则 C shell 执行存储在该文件中的命令。下一步，C shell 执行系统范围设置文件 **/etc/csh.login**（如果它可用）。然后，它搜索您的主目录，查找 **.cshrc** 和 **.login** 文件。如果它们存在，则它们包含运行 C shell 的相关任何定制用户信息。**/etc/csh.cshrc** 和 **/etc/csh.login** 文件中的所有变量设置可能由 **$HOME** 目录中您的 **.cshrc** 和 **.login** 文件覆盖。只有 root 用户可以修改 **/etc/csh.cshrc** 和 **/etc/csh.login** 文件。

只在登录时执行一次 **/etc/csh.login** 和 **$HOME/.login** 文件。通常这些文件用于保存环境变量定义，您在登录时要执行的命令，或设置终端特征的命令。

/etc/csh.cshrc 和 **$HOME/.cshrc** 文件在登录时执行，并在每次调用 **csh** 命令或 C shell 脚本时执行。它们通常用于定义 C shell 特征，如别名和 C shell 变量（例如，history、noclobber 或 ignoreeof）。建议您只使用 **/etc/csh.cshrc** 和 **$HOME/.cshrc** 文件中的 C shell 内置命令（请参阅第 190 页的『C shell 内置命令』），因为使用其它命令会增加 shell 脚本的启动时间。

本节讨论以下内容:

C shell 限制

以下是 C shell 的限制:

- 字不能长于 1024 个字节。
- 自变量列表受 ARG_MAX 字节限制。ARG_MAX 变量的值可在 **/usr/include/sys/limits.h** 文件中找到。
- 涉及文件名扩展的命令的自变量个数限制为自变量列表中允许的字节数的 1/6。
- 命令替换可以替换不多于一个自变量列表中允许的字节。
- 要检测循环，shell 限制单个行上别名替换的个数为 20。
- **csh** 命令不支持基于字符等价分类的文件名扩展。
- 在 **csh** 执行任何应用程序前打开的文件描述符（除标准输入、标准输出和标准错误）对于该应用程序都不可用。

信号处理

C shell 通常忽略退出信号。分离运行的作业不受键盘（**INTERRUPT**、**QUIT** 和 **HANGUP**）生成的信号的影响。其它信号有 shell 从其父那里继承的值。可以使用 **onintr** 控制 shell 在 shell 步骤中处理 **INTERRUPT** 和 **TERMINATE** 信号。登录 shell 捕捉或忽略 **TERMINATE** 信号，这取决于如何设置它们。除登录 shell 以外的 shell 将 **TERMINATE** 信号传递给子进程。当登录 shell 读 **.logout** 时，绝不允许 **INTERRUPT** 信号。

C shell 命令

简单命令是由空格或制表符分隔的字的序列。

word 是字符和 / 或数字的序列，它不包含不带引号的空白。此外，以下字符和双字符还在被用作命令分隔符或终止符时形成单个字:

```
&       |       ;
&&      ||      <<          > >
<       >       (           )
```

这些特殊字符可以是其它字的一部分。但是，在它们之前加 \，防止 shell 将它们解释为特殊字符。圈起在 ’ ’ 或 ″ ″（匹配的引证字符对）或反引号中的字符串也可形成字的一部分。当它们围起在空白、制表符和特殊字符中时，这些标记不能形成独立的字。此外，可以将一个换行字符圈起在这些标记中，方法是在它前面加一个 \.

简单命令序列中的第一个字（号码为 0）通常指定命令的名称。任何剩余的字被传递给该命令，但有一些例外。如果命令指定是编译程序的可执行文件，则 shell 立即运行该程序。如果文件标记为可执行文件，但不是已编译程序，则 shell 假定它是 shell 脚本。在这种情况下，shell 启动另一个自身的实例（子 shell）来读文件和执行其中的命令。

本节讨论以下内容:

• 第 196 页的『非内置 C shell 命令执行』

C shell 内置命令

内置命令在 shell 中运行。如果内置命令作为流水线的任何组件发生（除最后一个），则命令在子 shell 中运行。

> **注**: 如果从 C shell 提示符输入命令，则系统首先搜索内置命令。如果内置命令不存在，则系统搜索由 **path** shell 变量指定的目录，以查找系统级命令。一些 C shell 内置命令和操作系统命令具有相同的名称。然而，这些命令不必以同样的方式工作。要获取有关命令如何工作的更多信息，请查阅相应的命令描述。

如果从 shell 运行 shell 脚本，并且 shell 脚本的第一行以 `#!/ShellPathname` 开始，则 C shell 运行注释中指定的 shell 来处理脚本。否则，它运行缺省 shell （链接到 **/usr/bin/sh** 的 shell）。如果由缺省 shell 运行，则 C shell 内置命令可能不可识别。要运行 C shell 命令，使脚本的第一行为 `#!/usr/bin/csh`。

有关内置命令的字母顺序列表，请参阅第 206 页的『C shell 内置命令的列表』。

C shell 命令描述

C shell 提供以下内置命令:

alias [*Name* [*WordList*]]
如果不指定任何参数，则显示所有别名。否则，命令显示所指定 *Name* 的别名。如果指定 *WordList*，则此命令将 *WordList* 的值指定给别名 *Name*。指定的别名 *Name* 不能是 **alias** 或 **unalias**。

bg [**%***Job* ...]
将当前作业或 *Job* 指定的作业置于后台，如果它停止则继续此作业。

break
在最近圈起的 **foreach** 或 **while** 命令的 **end** 后，继续运行。

breaksw
从 **switch** 命令中断；在 **endsw** 命令后继续。

case *Label***:**
在 **switch** 命令中定义 *Label*。

cd[*Name*]
等价于 **chdir** 命令（请参阅以下描述）。

chdir [*Name*]
家当前目录更改到 *Name* 变量指定的那个目录。如果不指定 *Name*，则命令更改为您的主目录。如果 *Name* 变量的值不是当前目录的子目录，并且不以 /、./、或 ../ 开始，则 shell 检查 **cdpath** shell 变量的每个组件以查看它是否具有匹配 *Name* 变量的子目录。如果 *Name* 变量是含以 / 开始的值的 shell 变量，则 shell 尝试此操作以查看它是否是目录。**chdir** 命令等价于 **cd** 命令。

continue
在最近圈起的 **while** 或 **foreach** 命令的 **end** 处，继续运行。

default:
标注 **switch** 语句中的 **default** 情况。**default** 应在所有其它 **case** 标号后出现。

dirs
显示目录堆栈。

echo
将字符串写入 shell 的标准输出。

else
运行 **if (***Expression***) then** ...**else if (***Expression2***) then** ... **else** ... **endif** 命令序列中第二个 **else** 后的命令。

end
继续将 *Name* 变量设置为由 *List* 变量指定的每个成员，并运行 **foreach** 与匹配的 **end** 语句之间的 *Commands* 序列。**foreach** 和 **end** 语句必须单独出现在独立的行上。

使用 **continue** 语句继续循环，使用 **break** 语句提前结束循环。当 **foreach** 命令从终端读取时，C shell 使用 **?** 提示以允许输入 *Commands*。循环中的命令（由 **?** 提示）不放入历史列表。

endif
如果 *Expression* 变量为真，则运行跟随第一个 **then** 语句的 *Commands*。如果 **else if** *Expression2* 为真，则运行跟随第二个 **then** 语句的 *Commands*。如果 **else if** *Expression2* 为假，则运行跟随 **else** 的 *Commands*。任何 **else if** 的对数都是可能的。只需要一个 **endif** 语句。**else** 段是可选的。字 **else** 和 **endif** 只能在输入行的开始使用。**if** 段必须单独出现在其输入行上或在 **else** 命令后。

endsw
继续将每个 **case** 标号与 *string* 变量的值匹配。*string* 是首先展开的命令和文件名。在 **case** 标号中使用模式匹配字符 *、? 和 [. . .]，它们是变量扩展的。如果在 **default** 标号前没有找到标号匹配，则执行在 **default** 标记后开始。**case** 标号和 **default** 标号必须出现在行的开始。**breaksw** 命令导致执行在 **endsw** 命令后继续。否则，控制可能对 **case** 和 **default** 标号无效，如在 C 编程语言中一样。如果没有标号匹配，并且没有 **default**，则在 **endsw** 命令后执行继续。

eval *Parameter . . .*
在输入至 shell 时读 *Parameter* 变量的值，并在当前 shell 的上下文中运行结果命令。使用此命令以运行作为命令或变量替换的结果生成的命令，因为语法分析在这些替换前发生。

exec *Command*
运行指定的 *Command* 代替当前 shell。

exit [**(***Expression***)**
退出 shell，并带有 **status** shell 变量的值（如果没有指定 *Expression*）或带有指定的 *Expression* 的值。

fg [%*Job ...*]
将当前作业或 *Job* 指定的作业置于前台，如果它停止则继续此作业。

foreach *Name* (*List*) *Command. . .*
继续为每个由 *List* 变量和命令序列指定的成员设置 *Name* 变量，直到到达一个 **end** 命令。

glob *List*
使用历史、变量和文件名扩展显示 *List*。将空字符放入字之间，并且在末尾不包含回车。

goto *Word*
在 *Word* 变量指定的行后继续运行。指定的 *Word* 是文件名和展开的命令，以产生由 *Label:* 变量指定的格式的字符串。shell 尽可能多的回绕其输入，并搜索 *Label***:** 格式的行，可能前有空白和制表符。

hashstat
显示统计信息，它表示散列表定位命令有多成功。

history [**-r** | **-h**] [*n*]
显示历史事件列表。首先显示最旧的事件。如果指定数 *n*，则只显示最近事件的指定数。**-r** 标志逆向显示事件的顺序，因此首先显示最近的事件。**-h** 标志显示历史列表不具有前导数字。使用此标志产生适合于 **source** 命令的 **-h** 标志一起使用的文件。

if (*Expression*) *Command*
如果指定的 *Expression* 为真，则运行指定的 *Command*（包含其自变量）。*Command* 变量上的变量替换在早期发生，与 **if** 语句的其余部分在同一时间。指定的 *Command* 必须是简单命令（而非流水线、命令列表或加括号的命令列表）。

> **注:** 即使 *Expression* 变量为假，且 *Command* 不执行，输入和输出重定向也发生。

jobs [**-l**]
列出活动作业。使用 **-l**（小写 *L*）标志，**jobs** 命令列出进程标识及作业号和名称。

kill -l | [[*-Signal*] % *Job...*|*PID...*]
将 **TERM**（终止）信号或 *Signal* 指定的信号发送给指定的 *Job* 或 *PID*（进程）。通过号码或名称（如 **/usr/include/sys/signal.h** 文件中给定，舍去 **SIG** 前缀）指定信号。**-l**（小写 *L*）标志列出信号名称。

limit [**-h**] [*Resource* [*Max-Use*]]

限制当前进程和它创建的每个进程使用指定的资源。进程资源限制在 **/etc/security/limits** 文件中定义。可控制的资源是中央处理单元（CPU）时间、文件大小、数据大小、核心转储大小和内存使用。当用户添加到系统时，这些资源允许的最大值使用 **mkuser** 命令设置。它们使用 **chuser** 命令更改。

限制分类为软或硬。用户可以增加其软限制，直到达到硬限制所施加的最高限度。必须有 root 用户权限才能将软限制增加到大于硬限制，或更改资源硬限制。**-h** 标志显示硬限制，而非软限制。

如果未指定 *Max-Use* 参数，则 **limit** 命令显示指定资源的当前限制。如果未指定 *Resource* 参数，则 **limit** 命令显示所有资源的当前限制。有关由 **limit** 子命令控制的资源的更多信息，请参阅 *AIX 5L Version 5.2 Technical Reference: Base Operating System and Extensions Volume 1* 中的 **getrlimit**、**setrlimit** 或 **vlimit** 子例程。

CPU 时间的 *Max-Use* 参数以 hh:mm:ss 格式指定。其它资源的 *Max-Use* 参数指定为浮点数或整数，可选地后跟比例系数。比例系数是：k 或千字节（1024 字节），m 或兆字节，或者 b 或块（**ulimit** 子例程所使用的单位，如 *AIX 5L Version 5.2 Technical Reference: Base Operating System and Extensions Volume 2* 中解释）。如果未指定比例系数，则为所有资源假定使用 k。对于资源名和比例系数，满足名称的明确前缀。

注：仅当存在与其它活动进程争用系统内存时，此命令才限制该进程可用的物理内存。

login

结束登录 shell，并用 **/usr/bin/login** 命令的实例替换它。这是一种注销的方式（为与 **ksh** 和 **bsh** 命令的兼容而包含）。

logout

结束登录 shell。如果设置了 **ignoreeof** 选项，则必须使用此命令。

nice [**+***n*] [*Command*]

如果未指定值，则将在此 shell 中运行的命令的优先级设置为 24。如果指定了 **+***n* 标志，则设置优先级加上指定的数字。如果指定了 +n 标志和 *Command*，则以 24 加指定数字的优先级运行 *Command*。如果具有 root 用户权限，则可以使用负数运行 **nice** 语句。*Command* 总是在子 shell 中运行，则应用针对简单 **if** 语句中命令的限制。

nohup [*Command*]

如果未指定 *Command*，则导致对脚本的剩余部分忽略 **hangups**。如果指定了 *Command*，则使指定的 *Command* 运行时忽略 **hangups**。要运行流水线或命令的列表，请将流水线或列表放在一个 shell 脚本中，给脚本执行许可权，并将 shell 脚本作为 *Command* 变量的值使用。有效保护所有用 & 在后台运行的进程被以在注销时不发送 **hangup** 信号。然而，这些进程仍有可能显式地发送 **hangups**，除非使用 **nohup** 语句。

notify [**%***Job*...]

当前作业或指定 *Job* 的状态更改时，导致 shell 异步地通知您。通常，shell 就在呈现 shell 提示符前提供通知。如果设置了 **notify** shell 变量，则此功能是自动的。

onintr [**-** | *Label*]

控制中断时 shell 的操作。如果未指定自变量，则在中断时恢复 shell 的缺省操作，缺省操作结束 shell 脚本或返回到命令输入级别。如果指定了 **-** 标志，则导致忽略所有中断。如果指定了 *Label*，则当 shell 接收到中断或当子进程因中断而结束时，导致 shell 运行 **goto** *Label* 语句。在任何情况下，如果 shell 拆离地运行并且忽略中断，则所有格式的 **onintr** 语句都没有意义。shell 和所有调用的命令继续忽略中断。

popd [**+***n*]

弹出目录堆栈并更改到新的顶层目录。如果指定 +*n* 变量，则命令废弃堆栈中的第 *n* 个条目。目录堆栈的元素从顶部开始编号，从 0 开始。

pushd [**+***n*|*Name*]

没有自变量，则交换目录堆栈的顶部两个元素。使用 *Name* 变量，命令更改到新的目录，并将旧的当前目录（在 **cwd** shell 变量中给定）推到目录堆栈。如果指定 +*n* 变量，则命令将目录堆栈的第 *n* 个组件旋转为顶部元素，并更改到它。目录堆栈的成员从顶部开始编号，从 0 开始。

rehash	导致重新计算 **path** shell 变量中目录内容的内部散列表。如果当登录时将新的命令添加到 **path** shell 变量的目录中，则需要此操作。仅当命令添加到用户自己的目录或某人更改了某一系统目录的内容时，才需要 **rehash** 命令。
repeat *Count Command*	运行指定的 *Command*，服从与简单 **if** 语句中命令相同的限制，次数由 *Count* 指定。 **注**: I/O 重定向只发生一次，即使 *Count* 变量等于 0。
set [[*Name*[n]] [**=** *Word*]] | [*Name* = (*List*)]	不带参数使用时显示所有 shell 变量的值。具有不止一个单个字作为其值的变量显示为一个带圆括号的字列表。如果仅指定了 *Name*，则C shell 将 *Name* 变量设置为空字符串。否则，将 *Name* 设置为 *Word* 变量的值，或将 *Name* 设置为由 *List* 变量指定的字的列表。当指定了 *n* 时，*Name* 变量的第 *n* 个组件设置为 *Word* 变量的值；第 *n* 个组件必须已存在。在所有情况下，值是展开的命令和文件名。这些自变量可重复，以在单个 **set** 命令中设置多个值。然而，变量扩展在任何设置发生后对所有自变量发生。
setenv*Name Value*	将由 *Name* 变量指定的环境变量的值设置为 *Value*，单个字符串。最常用的环境变量 **USER**、**TERM**、**HOME** 和 **PATH** 自动导入到和导出自 C shell 变量 **user**、**term**、**home** 和 **path**。不需要使用对这些变量使用 **setenv** 语句。
shift [*Variable*]	向左移位 **argv** shell 变量或指定的 *Variable* 的成员。如果 **argv** shell 变量或指定的 *Variable* 未设置，或作为其值的字少于一个，则发生错误。
source[**-h**] *Name*	读由 *Name* 变量指定的命令。可以嵌套 **source** 命令。然而，如果它们嵌套得太深，则 shell 可能用完文件描述符。**source** 命令中任何级别的错误将结束所有嵌套的 **source** 命令。通常，**source** 命令期间的输入不放在历史列表上。**-h** 标志导致命令放置在历史列表中，而不执行它们。
stop [**%***Job* ...]	停止当前作业或在后台运行的指定的 *Job*。
suspend	停止 shell，就像接收到 **STOP** 信号一样。
switch (*string*)	开始一个 **switch (***String***) case** *String* **:** ... **breaksw default:** ... **breaksw endsw** 命令序列。此命令序列继续将每个 case 标号与 *String* 变量的值匹配。如果在 default 标号前没有找到标号匹配，则在 default 标号后执行开始。

time [*Command*]

time 命令控制命令的自动定时。如果不指定 *Command* 变量，则 **time** 命令显示此 shell 及其子 shell 使用的时间摘要。如果指定带 *Command* 变量的命令，则会定时它。然后 shell 显示时间摘要，如 **time** shell 变量下所描述。如果必要，当命令完成时创建一个额外的 shell 来显示时间统计信息。

以下示例将 **time** 与 **sleep** 命令一起使用:

```
time sleep
```

此命令的输出类似以下:

```
0.0u 0.0s 0:00 100% 44+4k 0+0io 0pf+0w
```

输出字段如下:

字段	描述
First	专用于用户进程的 CPU 时间秒数
Second	内核代表用户进程消耗的 CPU 时间秒数
Third	命令的经过（墙钟）时间
Fourth	总计用户 CPU 时间加上系统时间，作为经过时间的百分率
Fifth	已使用的共享内存的平均数量，加上已使用的未共享数据空间的平均数量，以千字节计
Sixth	块输入和输出操作的数
Seventh	缺页故障加上交换数

umask [*Value*]

确定文件许可权。当文件创建时，此 *Value* 以及创建进程的许可权确定文件的许可权。缺省值是 022。如果未指定 *Value*，则显示当前设置。

unalias *|*Pattern*

废弃所有名称与 *Pattern* 变量匹配的别名。所有别名通过 **unalias *** 命令除去。缺少别名并不导致错误。

unhash

禁用内部散列表的使用以定位运行的程序。

unlimit [**-h**][*Resource*]

除去对 *Resource* 变量的限制。如果未指定 *Resource* 变量，则除去所有的资源限制。请参阅 **limit** 命令的描述以获取 *Resource* 名称的列表。

-h 标志除去对应的硬限制。只有具有 root 用户权限的用户可以更改硬限制。

unset *|*Pattern*

除去所有名称与 *Pattern* 变量匹配的变量。使用 **unset *** 除去所有变量。如果未设置变量，它并不导致错误。

unsetenv *Pattern*

从环境中除去其名称与指定的 *Pattern* 匹配的所有变量。（请参阅 **setenv** 内置命令。）

wait

等候所有后台作业。如果 shell 是交互式的，INTERRUPT（通常是 Ctrl-C 键序列）将中断等候。然后 shell 显示所有已知未完成作业的名称和作业号。

while (*Expression*) *Command*. . . **end**

当 *Expression* 变量指定的表达式求值非零时，求值 **while** 与匹配的 **end** 语句之间的 *Commands*。可以使用 **break** 语句提前结束循环，使用 **continue** 语句提前继续循环。**while** 和 **end** 语句必须单独出现在其输入行上。如果输入来自终端，则发生在 while (*Expression*) 后的提示类似 **foreach** 语句。

@ [*Name*[*n*] **=** *Expression*]

不带参数使用时显示所有 shell 变量的值。否则，将由 *Name* 变量指定的名称设置为 *Expression* 变量的值。如果表达式包含 <、>、& 或 | 字符，则表达式的此部分必须放在圆括号内。当指定了 *n* 时，*Name* 变量的第 *n* 个组件设置为 *Expression* 变量。*Name* 变量及其第 *n* 个组件必须都已存在。

C 语言运算符（如 *= 和 +=）可用。将 *Name* 变量从赋值运算符分隔出来的空格是可选的。空格在分隔 *Expression* 变量的组件时是必需的，否则将读为单个字。特殊后缀运算符双加号（++）和双连字符（- -）分别增加和减少 *Name* 变量的值。

C shell 表达式和运算符

@ 内置命令和 **exit**、**if** 及 **while** 语句接受包含运算符（类似 C 语言）的表达式，具有相同的优先权。提供以下运算符:

运算符	含义
()	更改优先权
~	补码
!	非
*/ %	乘、除、模
+ -	加、减
<< > >	左移、右移
<= >= < >	关系运算符
== != =~ !~	字符串比较 / 模式匹配
&	按位 “与”
^	按位 “异或”
|	按位 “或”
&&	逻辑 “与”
||	逻辑 “或”

在上面的列表中，运算符的优先权在列表中向下递减（从左至右，从上至下）。

注: 运算符 + 和 - 是右关联的。例如，a + b - c 的求值执行如下:

```
a + (b - c)
```

而不是如下:

```
(a + b) - c
```

==、!=、=~ 和 !~ 运算符将其自变量作为字符串比较；所有其它运算符对数字操作。=~ 和 !~ 运算符类似== 和 !=，除了最右边是一个与最左边操作数匹配的 *模式*。这减少了在 shell 步骤中使用 **switch** 语句的需要。

逻辑运算符 “或”（||）和 “与”（&&）也可用。它们可用于检查数字的范围，如以下示例中:

```
if ($#argv > 2 && $#argv < 7) then
```

在前边的示例中，自变量的号码必须大于 2 且小于 7。

认为以零（0）开始的字符串是八进制数。认为空或缺少自变量是 0。所有表达式导致代表十进制数的字符串。请注意表达式的两个组件可出现在同一个字中。除了当在对解析器依照句法有意义的表达式的组件（& | < > ( )）旁时，表达式组件应该用空格围起。

也在表达式中可用作原语操作数的是圈起在 ( ) 中的命令执行以及 (**-operator** *Filename*) 格式的文件查询，其中 **operator** 是以下之一:

r 读访问
w 写访问
x 执行访问
e 存在
o 所有权
z 零大小
f 普通文件
d 目录

指定的 *Filename* 是展开的命令和文件名，然后测试以查看它是否与实际用户具有指定的关系。如果 *Filename* 不存在或不可访问，则所有查询返回假（0）。如果命令成功运行，则查询返回真值（1）。否则，如果命令失败，则查询返回假值（0）。如果要求更多详细的状态信息，请在一个表达式外运行命令，然后检查 **status** shell 变量。

C shell 中的命令替换

在*命令替换*中，shell 执行指定命令并用其输出替换该命令。要在 C shell 中执行命令替换，用反引号（` `）将命令或命令字符串圈起。shell 通常在空白、制表符和换行字符位置将命令的输出断成独立的字。然后它使用此输出替换原始命令。

在以下示例中，圈起 **date** 命令的反引号（` `）表示替换命令的输出:

```
echo The current date and time is: `date`
```

此命令的输出可能看起来类似如下:

```
The current date and time is: Wed Apr 8 13:52:14 CDT 1992
```

C shell 对内置 shell 命令的自变量选择性地执行命令替换。这意味着它不展开表达式中不求值的那些部分。对于非内置的命令， shell 独立地从自变量列表替换命令名。仅在 shell 执行输入或输出重定向后，替换才发生在主 shell 的子 shell 中。

如果命令字符串用 ″ ″ 围起，则 shell 仅将换行字符视为字分隔符，这样就在字中保留空白和制表符。在所有情况下，单个最后的换行字符都不强制新字。

非内置 C shell 命令执行

当 C shell 确定命令不是内置 shell 命令时，它尝试使用 **execv** 子例程运行命令。**path** shell 变量中的每个字命名一个目录，shell 尝试从该目录运行命令。如果未给定 **-c** 或 **-t** 标志，则 shell 将这些目录中的名称散列到内部表中。仅当命令有可能驻留在某一目录中时，shell 才尝试在该目录上调用 **exec** 子例程。如果使用 **unhash** 命令关闭此机制，或赋予 shell **-c** 或 **-t** 标志，则 shell 与给定的命令名连接，以形成文件的路径名。对于不以 / 开始的 **path** 变量的每个目录组件，shell 在任何情况下也都执行此操作。然后 shell 尝试运行命令。

加括号的命令总是在子 shell 中运行。例如:

```
(cd ; pwd) ; pwd
```

显示主目录，而不更改当前目录位置。然而，命令:

```
cd ; pwd
```

将当前目录位置更改到主目录。加圆括号的命令最常被用于防止 **chdir** 命令影响当前 shell。

如果文件具有执行许可权，但不是系统的可执行二进制文件，则 shell 假定它是包含 shell 命令的文件，并运行新的 shell 来读它。

如果 shell 有别名，则将别名的字加到自变量列表之前，以形成 shell 命令。别名的第一个字应该是 shell 的全路径名。

C shell 中的历史替换

历史替换让您修改以前命令中个别的字以创建新命令。历史替换使以下操作变得简单: 重复命令，在当前命令中重复前一命令的自变量，或输入很少就能修改前一命令中的拼写错误。

历史替换以 ！字符开始并能在命令行上的任何地方出现，只要它们不嵌套（换句话说，历史替换不能包含另一个历史替换）。可以在 ！前加 / 来取消感叹号的特殊含义。此外，如果将 ！放置在空白、制表符、换行字符、= 或 (前，则历史替换不发生。

历史替换还在以 ^ 开始输入行时发生。shell 在执行包含历史替换的任何输入行前，在工作站回显该输入行。

本节讨论以下内容:

- 『历史列表』
- 第 198 页的『事件规范』
- 第 199 页的『用单引号和双引号引证』

历史列表

历史列表保存 shell 从由一个或多个字组成的命令行读取的命令。历史替换重新引入来自这些保存的命令的字顺序到输入流。

history shell 变量控制历史列表的大小。必须在 **.cshrc** 文件或用内置 **set** 命令在命令行上设置 **history** shell 变量。总是保留前一命令，而不考虑 **history** 变量的值。历史列表中的命令按顺序编号，从 1 开始。内置 **history** 命令产生类似以下的输出:

```
9 write michael
10 ed write.c
11 cat oldwrite.c
12 diff *write.c
```

shell 用命令字符串的事件号显示命令字符串。事件号出现在命令的左边，并在输入涉及历史中的其它命令的命令时表示。通常没有必要使用事件号来指事件，但是通过将 ！放在指定给 **PROMPT** 环境变量的提示符字符串中，您可以将当前事件号显示为系统提示符的一部分。

全历史引用包含事件规范、字标志符和一个或多个以下常规格式的修改符:

```
Event[.]Word:Modifier[:Modifier] . . .
```

注: 只可以修改一个字。不允许包含空白的字符串。

在 **history** 命令输出的前一样本中，当前事件号是 13。请使用此示例，以下指前面的事件:

```
!10         事件号 10。
!-2         事件号 11（当前事件减 2）。
!d          以 d 开始的命令字（事件号 12）。
!?mic?      包含字符串 mic 的命令字（事件号 9）。
```

不进行进一步修改，这些格式仅重新引入指定的事件的字，每个都用单个空白分隔。作为特例，!! 指前一命令；输入行上的单个命令 **!!** 重新运行前一命令。

事件规范

要从事件中选择字，请在事件规范后跟一个 : 和以下字指定符之一（输入行的字从 0 开始按顺序编号）：

0 第一个字（命令名）
n 第 *n* 个自变量
^ 第一个自变量
$ 最后一个自变量
% 紧在 ?*string*? 搜索之前的匹配字
x-y 从第 *x* 个字到第 *y* 个字范围的字
-y 从第一个字（0）到第 *y* 个字范围的字
***** 从第一个到最后一个自变量，或者如果事件中只有一个字（命令名），则什么也没有。
*x****** 从第 *x* 个自变量到最后一个自变量
*x***-** 同 *x**，但省略最后一个自变量

如果字标志符以 ^, $, *, - 或 % 开始，则可以省略分隔事件规范和字标志符的冒号。还可以将以下修改符的序列放在可选字标志符后，每个前面有一个冒号：

h 除去尾部路径名扩展名，保留头。
r 除去尾部 .xxx 组件，保留 root 名称。
e 除去所有，但保留 .xxx 尾部扩展名。
s*l*OldWord*l*NewWord*l* 用 *NewWord* 变量的值替换 *OldWord* 变量的值。

替换的左边不是编辑器可识别的模式（从字符串意义上说）；而是是没有空白的单个单元的字。通常，/ 定界原来的字（*OldWord*）及其替换（*NewWord*）。然而，可以使用任何字符作为定界符。在以示例中，使用 % 作为定界符允许 / 包含在字中：

```
s%/home/myfile%/home/yourfile%
```

shell 用 *NewWord* 变量中的 *OldWord* 文本替换 &。在以下示例中，/home/myfile 变成 /temp/home/myfile。

```
s%/home/myfile%/temp&%
```

shell 用上次替换或上下文扫描 !?String? 中使用的最后一个字符串替换替换中的空字。如果换行字符紧接着，则可以省略末尾定界符 (/)。使用以下修改符定界历史列表：

t 除去所有前导路径名组件，保留尾部
& 重复前一替换
g 全局地应用更改；即，每行的所有出现
p 显示新命令，但是不运行它
q 引证替换的字，因此防止进一步替换
x 象 **q** 修改符一样操作，但在空白、制表符和换行字符处断字

使用前修改符时，更改仅应用到第一个可修改字，除非 **g** 修改符在选定修改符之前。

如果您给出一个没有事件规范的历史引用（例如，!$），则 shell 将前一命令用作事件。如果前一历史引用在同一行发生，则 shell 重复前一引用。因此，以下序列给出匹配 ?foo? 的命令的第一个和最后一个自变量。

```
!?foo?^ !$
```

当输入行的第一个非空白字符是 ^ 时，历史引用的特殊缩写发生。这等价于 `!:s^`，这样为前一行的文本替换提供方便的速记。命令 `^ lb^ lib` 更正命令中的 `lib` 拼写。

如果必要，可以将历史替换围起在 { } 中，以将它与后面跟着的字符隔离。例如，如果要使用命令的引用:

```
ls -ld ~paul
```

要执行命令:

```
ls -ld ~paula
```

请使用以下结构:

```
!{l}a
```

在此示例中，`!{l}a` 查找以 `l` 开始的命令并将 `a` 附加到结束。

用单引号和双引号引证

要防止进一步解释所有或一些替换，将字符串圈起在单引号和双引号中。将字符串圈起在 ' ' 中防止进一步解释，而将字符串圈起在 " " 中允许进一步扩展。在两种情况下，结果文本都成为单个字的全部或部分。

C shell 中的别名替换

*别名*是指定给命令或命令串的名称。C shell 允许您指定别名并像您使用命令那样使用它们。shell 维护您定义的别名的列表。

在 shell 扫描命令行后，它将命令分为不同的字，并从左向右检查每个命令的第一个字以查看是否有别名。如果找到别名，则 shell 使用历史机制，以用别名引用的命令的文本替换别名的文本。出现的字替换命令和自变量列表。如果没有对历史列表进行引用，则自变量列表不更改。

有关 C shell 历史机制的信息，请参阅第 197 页的『C shell 中的历史替换』。

alias 和 **unalias** 内置命令建立、显示和修改别名列表。以下面的格式使用别名命令:

```
alias [Name [WordList]]
```

可选的 *Name* 变量指定所指定名称的别名。如果您使用 *WordList* 变量指定字列表，则命令将它指定为 *Name* 变量的别名。如果运行没有任一可选变量的 **alias** 命令，则它显示所有 C shell 别名。

如果 **ls** 命令的别名是 `ls -l`，则以下命令:

```
ls /usr
```

替换为以下命令:

```
ls -l /usr
```

自变量列表不受干扰，因为没有使用别名对命令中的历史列表进行引用。同样地，如果 **lookup** 命令的别名如下:

```
grep \!^ /etc/passwd
```

则 shell 用以下内容替换 `lookup bill`:

```
grep bill /etc/passwd
```

在此例中，`!^` 指历史列表，shell 用输入行中的第一个自变量替换它，在这个例子中是 `bill`。

可以在别名中使用特殊的模式匹配字符。以下命令:

```
alias lprint 'pr &bslash2.!* >

> print'
```

创建命令，格式化其自变量为行式打印机。! 字符通过使用单引号保护别名中的 shell，以便 **pr** 命令运行前不展开别名。

如果 shell 定位别名，则它执行输入文本的字转换，并在更改过的输入行上再次开始别名进程。如果下一个文本的第一个字与旧的相同，则通过标志别名防止循环以终止别名进程。检测到其它后继循环将导致错误。

C shell 中的变量和文件名替换

C shell 允许您进行变量和文件名替换。

本节讨论以下内容:

C shell 中的变量替换

C shell 维护一组变量，每个变量有零个或多个字的列表作为其值。这些变量中的一些是由 shell 设置或由其引用的。例如，**argv** 变量是 shell 变量列表的映象，并且组成此变量的值的字以特殊方式引用。

要更改和显示变量的值，请使用 **set** 和 **unset** 命令。在由 shell 引用的变量中，数字是开关（打开或关闭对象的变量）。shell 不检查值的开关，只检查它们是否设置或取消设置。例如，**verbose** shell 变量是使命令输入回显的开关。此变量的设置由在命令行上发出 **-v** 标志产生。

其它操作将变量当作数字处理. @ 命令执行数字计算，并将结果指定给变量。然而，变量值总是表示为（零个或多个）字符串。对于数字操作，认为空字符串是零，忽略多字值的第二个字和后继字。

当您发出命令时，shell 分析输入行并执行别名替换。下一步，在运行命令前，它执行变量替换。$ 字符用键固定替换。但如果后跟空白、制表符或换行字符，则它不更改地传递。在 $ 字符前加 \ 防止此扩展，以下两种情况除外:

- 命令用 ″ ″ 圈起。在这种情况下，shell 总是执行替换。
- 命令用 ’ ’ 圈起。在这种情况下，shell 从不执行替换。以 ’ ’ 圈起的字符串解释为命令替换。（请参阅第 196 页的『C shell 中的命令替换』。）

shell 在变量扩展前识别输入和输出重定向，并独立地展开每个重定向。否则，命令名和完整的自变量列表一起展开。因此第一个（命令）字可能生成多个字，其中第一个字成为命令名，而其余的字成为参数。

除非用 ″ ″ 圈起或给定 **:q** 修改符，否则变量替换的结果可能最终服从于命令替换和文件名替换。当用双引号圈起时，其值由多个字组成的变量展开成单个字或单个字的一部分，并且变量值的字以空白分隔。当对 **:q** 修改符应用于替换时，变量展开成多个字。每个字以一个空白分隔并用双引号圈起，以防止以后的命令替换或文件名替换。

以下标志法允许您将变量值引入到 shell 输入中。除非另有说明，否则引用不是使用 **set** 命令设置的变量是错误的。

您可将修改符 **:gh**、**:gt**、**:gr**、**:h**、**:r**、**:q** 和 **:x** 应用于以下替换。如果 {} 在命令格式中出现，则修改符必须放置在大括号中。在每个变量扩展上只允许一个 **:** 修改符。

$*Name*
${*Name***}** 由指定给 *Name* 变量的字替换，每个字用空白分隔。大括号将 *Name* 变量与任何后面的字符隔开，否则后面的字符会成为变量的一部分。Shell 变量名以字母开始，并由最多 20 个字母和数字组成，包括下划线 (_) 字符。如果 *Name* 变量不指定 shell 变量，但在环境中设置了，则返其值回。在这种情况下，前加冒号的修改符以及在此描述的其它格式不可用。

$*Name***[***number***]**
${***Name***[***number***]}** 从 *Name* 变量的值中只选择一些字。数字服从于变量替换，并且可能由单个数字或由 - 分隔的两个数字组成。变量字符串的第一个字编号为 1。如果范围的第一个数字省略，则缺省为 1。如果范围的最后一个数字省略，则缺省为 **$#*Name*。* 符号选择所有字。如果第二个自变量省略或在某一个范围中，则范围为空不是错误。

$#*Name*
${#*Name***}** 给出 *Name* 变量中的字号码。这可在 **[***number***]** 中使用，如以上显示。例如，`$Name[$#Name]`。

$0 替换文件的名称，命令输入从该文件读取。如果名称未知，则发生错误。

$*number*
${*number***}** 等价于 `$argv[number]`。

$* 等价于 `$argv[*]`。

以下替换可能不使用 **:** 修饰符更改:

$?*name*
${?*name***}** 如果 *name* 变量已设置，则替换字符串 1，如果此设置未设置，则替换字符串零（0）。

$?0 如果当前输入文件名已知，则替换 1，如果文件名未知，则替换零（0）。

$$ 替换父 shell 的（十进制）进程号。

$< 替换来自标准输入的一行，而不进一步解释。使用此替换在 shell 步骤中从键盘读。

C shell 中的文件名替换

C shell 提供了几个快捷方式以节省时间和击键。如果字包含任何 *, ?, [] 或 { }，或以代字号 (~) 开始，则该字可作为文件名替换的候选。 C shell 将字视为模式，并使用匹配该模式的依字母顺序排列的文件名列表替换该字。

使用当前整理序列，它由 **LC_COLLATE** 或 **LANG** 环境变量指定。在指定文件名替换的字列表中，如果没有模式匹配现有文件名，则产生错误。然而，不要求每个模式都匹配。只有字符匹配符号 *、? 和 [ ] 表示模式匹配或文件名扩展。代字号 (~) 和 { } 字符表示文件名缩写。

文件名扩展

* 字符匹配任何字符串，包括空字符串。例如，在包含以下文件的目录中:

```
a aa aax alice b bb c cc
```

命令 `echo a*` 打印所有以字符 `a` 开始的文件名:

```
a aa aax alice
```

> **注**: 当文件名匹配时，字符点（.）和 / 必须显式地匹配。

? 字符匹配任何单个字符。以下命令:

```
ls a?x
```

列出每个以字母 a 开头，后跟单个字符，并以字母 x 结束的文件名:

```
aax
```

要匹配单个字符或某一范围的字符，请将一个字符或多个字符圈起在 [ ] 中。以下命令:

```
ls [abc]
```

列出精确地与某一个圈起的字符匹配的所有文件名:

```
a b c
```

在方括号中，某一词法范围的字符由 [a-z] 表示。匹配此模式的字符由当前整理序列定义。

文件名缩写

代字号 (~) 和 { 字符表示文件名缩写。文件名开头处的 ~ 用于表示主目录。独立使用时, ~ 字符展开为 **home** shell 变量的值反映的您的主目录。例如，以下命令:

```
ls ~
```

列出位于您的 **$HOME** 目录中的所有文件和目录。

当命令后跟由字母、数字和 - 字符组成的名称时，shell 搜索具有该名称的用户，并替换该用户的 **$HOME** 目录。

注: 如果 ~ 字符后跟非字母或字符 /，或出现在除字开头处以外的任何位置，则它不展开。

要匹配文件名中的字符而不输入整个文件名，请将文件名用 { } 圈起。模式 a{b,c,d}e 是另一种写 abe ace ade 的方式。shell 保留左至右顺序，并以低级别独立地存储匹配的结果以保留此顺序。此构造可能嵌套。因此，以下:

```
~source/s1/{oldls,ls}.c
```

展开为:

```
/usr/source/s1/oldls.c /usr/source/s1/ls.c
```

如果 **source** 的主目录是 **/usr/source**。类似地，以下:

```
../{memo,*box}
```

可能展开为:

```
../memo ../box ../mbox
```

注: memo 不使用匹配的 *box 的结果排序。作为特例，{, } 和 { } 字符不受干扰地传递。

字符类

还可以使用字符类来匹配某一范围指示内的文件名。以下格式指示系统匹配属于所指定类的任何单个字符:

```
[:charclass:]
```

以下类对应于 **ctype** 子例程:

字符类	定义
alnum	字母数字字符
alpha	大写和小写字母

字符类	定义
cntrl	控制字符
digit	数字
graph	图形字符
lower	小写字母
print	可打印字符
punct	标点符号字符
space	空格、水平制表符、回车、换行、垂直制表符或换页字符
upper	大写字符
xdigit	十六进制数字

假定您位于包含以下文件的目录中:

```
a aa aax Alice b bb c cc
```

在 C shell 提示符下输入以下命令:

```
ls [:lower:]
```

按下 Enter 键。

C shell 列出所有以小写字符开始的文件名:

```
a aa aax b bb c cc
```

有关字符类表达式的更多信息，请参考 **ed** 命令。

C shell 中的环境变量

某些变量对 C shell 具有特殊的含义。这些变量中， **argv**、**cwd**、**home**、**path**、**prompt**、**shell** 和 **status** 总是由 shell 设置。除 **cwd** 和 **status** 变量外，该操作仅在初始化时发生。这些变量维护其设置，除非您显式地重新设置它们。

csh 命令将 **USER**、**TERM**、**HOME** 和 **PATH** 环境变量分别复制到 **csh** 变量 **user**、**term**、**home** 和 **path** 中。无论何时重新设置正常 shell 变量时，都将这些值复制回环境中。**path** 变量不能在非 **.cshrc** 文件中设置，因为 **csh** 子进程从环境中导入路径定义并在更改时重新导出。

以下变量具有特殊含义:

argv 包含传递到 shell 脚本的自变量。位置参数从此变量替换。

cdpath 指定备用目录的列表，它由 **chdir** 或 **cd** 命令搜索以查找子目录。

cwd 指定当前目录的全路径名。

echo 使用 **-x** 命令行标志时设置；当设置时，使每个命令及其自变量就在运行前回显。对于非内置的命令，所有扩展在回显前发生。内置命令在命令替换和文件名替换前回显，因为这些替换的完成是选择性的。

histchars 指定更改用于历史替换中所使用字符的字符串值。使用其值的第一个字符作为历史替换字符，它替换缺省字符 !。其值的第二个字符在快速替换中替换 ^ 字符。

注: 将 **histchars** 值设置为在命令或文件名中使用的字符可能导致非故意的历史替换。

history 包含数字值以控制历史列表的大小。未废弃在允许的事件数中引用的任何命令。非常大的 **history** 变量的值可能导致 shell 的用尽内存。无论此变量是否设置，C shell 总是保存在历史列表上运行的最后一个命令。

home 表明您的主目录，从环境初始化。代字号 **(~)** 字符的文件名扩展引用此变量。

ignoreeof 指定 shell 忽略来自输入设备（是工作站）的文件结束符字符。这防止当 shell 读到一个文件结束符字符（Ctrl-D）时，意外地杀死 shell。

mail 指定 shell 检查邮件所在的文件。这在每个命令完成后完成，如果已经过了指定的时间间隔，则它导致提示。如果文件存在并且其访问时间早于其更改时间，则 shell 显示消息邮件在文件中。。
如果 **mail** 变量的值的第一个字是数字，则它指定一个不同的邮件检查时间间隔（以秒计）；缺省值是 600（10 分钟）。如果指定多个邮件文件，则当指定的文件中有邮件时，shell 显示消息新邮件在文件中。

noclobber 对输出重定向设置限制，以确保文件不会意外地破坏，并且重定向附加到现有文件。

noglob 禁止文件名扩展。在不处理文件名的 shell 脚本中，或者当获取了文件名的列表并且不期望进一步扩展时，这非常有用。

nonomatch 指定如果文件名扩展不匹配任何现有文件，不导致错误；而是返回原语模式。如果原语模式是畸形的，则仍是一个错误。

notify 指定 shell 发送作业状态更改的异步通知。缺省为就在显示 shell 提示符前呈现状态更改。

path 指定命令在其中查找命令进行执行的目录。空字指定当前目录。如果未设置 **path** 变量，则只有全路径名可运行。缺省搜索路径（从登录期间使用的 **/etc/environment** 文件）如下:

```
/usr/bin /etc /usr/sbin /usr/ucb /usr/bin/X11 /sbin
```

未给定 **-c** 或 **-t** 标志的 shell 通常在读 **.cshrc** 后散列 **path** 变量中目录的内容，并且还每次重新设置 **path** 变量。如果当 shell 活动时将新命令添加到这些目录，则必须给定 **rehash** 命令。否则，可能找不到命令。

prompt 指定每个命令从交互式工作站输入读之前显示的字符串。如果在字符串中出现 !，则它由当前事件号替换。如果 ！字符在以单引号或双引号圈起的引证字符串中，则 ！字符前必须加 \. 不具有 root 权限的用户的缺省提示符是 %。有 root 权限的用户的缺省提示是 #。

savehist 指定数字值，控制当您注销时保存在 **~/.history** 文件中的历史列表的条目数。保存以此事件号引用的任何命令。启动期间，shell 将 **~/.history** 读入历史列表，启用要通过登录保存的历史。非常大的 **savehist** 变量值减慢 shell 启动。

shell 指定 C shell 驻留的文件。这在派生 shell 时使用，以解释设置了执行位，但不可由系统执行的文件。这初始化成 C shell 的主目录。

status 指定由最后一个命令返回的状态。如果命令异常结束，则 0200 添加到状态。不成功的内置命令返回退出状态 1。成功的内置命令将状态设置为值 0。

time 控制命令的自动定时。如果设置了此变量，则执行时间超过指定的 CPU 秒数的任何命令，将在执行结束时显示一行使用的资源。有关缺省输出的更多信息，请参阅内置 **time** 命令。

verbose 由 **-v** 命令行标志设置，此变量导致每个命令的字在历史替换后显示。

C shell 中的输入和输出重定向

在 C shell 在执行命令之前，扫描命令行以获取重定向字符。这些特殊的符号定向 shell 以重定向输入和输出。

可以使用以下语法语句重定向命令的标准输入和输出:

< *File* 打开指定的 *File*（它是第一个展开的变量、命令和文件名）作为标准输入。

<<*Word* 读 shell 输入，一直到匹配 *Word* 变量的值的行。*Word* 变量不受制于变量、文件名或命令替换。在行上完成任何替换前，每一输入行与 *Word* 变量做比较。除非 *Word* 变量中出现引证字符 (\, ″, ’ 或 `)，否则 shell 在插入行上执行变量和命令替换，允许 \ 字符引证 $, \ 和 ` 字符。所替换的命令保留所有空白、制表符和换行字符，除了最后一个换行字符（被删除）。结果文本放置在一个匿名临时文件中，赋予它命令以作为标准输入。

> *File*

>!*File*

>& *File*

>&! *File*
使用指定的 *File* 作为标准输出。如果 *File* 不存在，则进行创建。如果 *File* 存在，则截断它，并且其以前的内容丢失。如果设置了 **noclobber** shell 变量，则 *File* 绝不能存在或是字符特殊文件或错误结果。这帮助防止文件的意外破坏。在这种情况下，使用包含 ！的格式来禁止此检查。*File* 以与 < 输入文件名相同的方式展开。格式 >& 将标准输出和标准错误都重定向到指定的 *File*。以下示例显示如何独立地将标准输出重定向到 **/dev/tty**，将标准错误重定向到 **/dev/null**。要求圆括号以允许标准输出和标准错误独立。

```
% (find / -name vi -print > /dev/tty) >& /dev/null
```

> >*File*

> >! *File*

> >& *File*

> >&! *File*
将指定的 *File* 用作标准输出，如 >，但将输出*附加*到 *File* 的末尾。如果设置了 **noclobber** shell 变量，则当 *File* 不存在时导致错误，除非给定了某一包含一个 ！的格式。否则，它类似于 >。

命令接收由输入 / 输出参数和存在命令作为流水线更改的环境，shell 在该环境中调用。这样，不像一些以前的 shell，从 shell 脚本运行的命令在缺省情况下不具有对命令文本的访问权。它们改为接收 shell 的原始标准输入。使用 << 机制来呈现直接插入数据，它允许 shell 命令文件作为流水线的组件运行，还使 shell 能阻拦读其输入。请注意，拆离运行命令的缺省标准输入不更改到空 **/dev/null** 文件。标准输入改为保留 shell 的原始标准输入。

要通过具有标准输出的管道重定向标准错误，使用 |& 格式而非只是 |。

控制流

shell 包含可用于调节命令文件（shell 脚本）中和（以受限但有用的方式）来自 shell 命令行输入的控制流的命令。这些命令都通过强制 shell 在其输入中重复或跳过进行操作。

foreach, switch 和 **while** 语句，以及 **if** 语句的 **if-then-else** 格式，要求主关键字出现在输入行上单个简单命令中。

如果 shell 输入是不可搜索的，则无论何时读循环 shell 都缓冲输入，并搜索内部缓冲区以进行循环所暗示的重读。要允许此操作，反向 **goto** 继续您无法搜索的输入。

C shell 中的作业控制

shell 将作业号与每个进程关联。 shell 保留当前作业的表并指定给它们小整数。当使用 **&** 在后台启动作业时，shell 打印类似于以下的行:

```
[1] 1234
```

该行表示作业号是 1，并且该作业是由进程标识为 1234 的单个进程组成的。使用内置 **jobs** 命令查看当前作业的表。

在后台中运行的作业如果尝试从工作站读，则要竞争以获得输入。后台作业还可以为与其它作业的输出交错的工作站产生输出。

可以用几种方法引用 shell 中的作业。使用 % 字符引入作业名。此名称可以是作业号或启动作业的命令名（如果此命令名是唯一的）。例如，如果 **make** 进程作为作业 1 运行，可以将它引用为 %1。如果只有一个暂挂作业的名称以字符串 make 开始，则还可以将它引用为 %make。还可以使用以下方法:

```
%?String
```

指定作业，其名称包含 String 变量（如果只有一个这样的作业）。

无论何时进程更改其状态，shell 都会立即检测。如果作业变得阻塞以至于不可能进一步进展，则 shell 将消息发送到工作站。此消息仅在您按下 Enter 键后显示。然而，如果设置 **notify** 变量，则 shell 立即发出一条消息，表示后台作业状态中的更改。使用内置 **notify** 命令标记单个进程以便迅速报告其状态更改。缺省情况下，**notify** 命令标记当前进程。

C shell 内置命令的列表

@	显示指定 shell 变量的值
alias	显示指定的别名或所有别名。
bg	将当前或指定作业置于后台。
break	在最近一个圈起的 **foreach** 或 **while** 命令结束后，继续运行。
breaksw	从 **switch** 命令中断。
case	在 **switch** 命令中定义标号。
cd	将当前目录更改到指定的目录。
chdir	将当前目录更改到指定的目录。
continue	继续最近围起的 **foreach** 或 **while** 命令的执行。
default	标注 **switch** 语句中的缺省情况。
dirs	显示目录堆栈。
echo	将字符串写入 shell 的标准输出。
else	运行 **if (***Expression***) then** ...**else if (***Expression2***) then** ... **else** ... **endif** 命令序列中第二个 **else** 后的命令。
end	标识前有 **foreach** 命令的命令序列的结束。
endif	运行 **if (***Expression***) then** ... **else if (***Expression2***) then** ... **else** ... **endif** 命令序列中第二个 **then** 后的命令。
endsw	标记 **switch (***String***) case** *String* **:** ... **breaksw default:** ... **breaksw endsw** 命令序列的结束。此命令序列继续将每个 case 标号与 *String* 变量的值匹配。如果执行了 **breaksw** 命令，或如果无标号匹配并且没有缺省，则在 **endsw** 命令后执行继续。
eval	将变量值作为输入读入 shell，并在当前 shell 的上下文中执行结果命令。
exec	运行指定的命令代替当前 shell。
exit	退出 shell，并且有状态 shell 变量的值或有指定的表达式的值。
fg	将当前或指定作业置于前台，如果它们已停止则继续它们。
foreach	继续为每个由 *List* 变量和命令序列指定的成员设置 *Name* 变量，直到到达一个 **end** 命令。
glob	使用历史、变量和文件名扩展显示列表。
goto	在指定行后继续运行。
hashstat	显示统计信息，表示散列表定位命令有多成功。
history	显示历史事件列表。
if	运行指定的命令（如果指定表达式为真）。
jobs	列出活动作业。
kill	将 **TERM**（终止）信号或 *Signal* 变量指定的信号发送给指定的作业或进程。
limit	限制当前进程和它创建的每个进程使用指定的资源。
login	结束登录 shell，并用 **/usr/sbin/login** 命令的实例替换它。
logout	结束登录 shell。
nice	设置在 shell 中运行的命令的优先级。
nohup	对过程的剩余部分使挂断忽略。
notify	当前作业或指定作业的状态更改时，使 shell 异步地通知您。
onintr	控制中断时 shell 的操作。

popd	弹出目录堆栈并返回到新的顶层目录。
pushd	交换目录堆栈的元素。
rehash	导致重新计算路径 shell 变量中包含目录内容的内部散列表。
repeat	运行指定的命令指定的次数，服从与 **if** 命令相同的限制。
set	显示所有 shell 变量的值
setenv	修改指定环境变量的值。
shift	左移指定的变量。
source	读由 *Name* 变量指定的命令。
stop	停止当前作业或在后台运行的指定的作业。
suspend	停止 shell，就像接收到 **STOP** 信号一样。
switch	开始 **switch (***String***) case** *String* **:** ... **breaksw default:** ... **breaksw endsw** 命令序列。此命令序列继续将每个 case 标号与 *String* 变量的值匹配。如果在 default 标号前没有找到标号匹配，则在 default 标号后执行开始。
time	显示 shell 及其子进程使用的时间的摘要。
umask	确定文件许可权。
unalias	废弃所有名称与 *Pattern* 变量匹配的别名。
unhash	禁用内部散列表的使用以定位运行的程序。
unlimit	除去资源限制。
unset	除去所有名称与 *Pattern* 变量匹配的变量。
unsetenv	从环境中除去其名称与指定的 *Pattern* 变量匹配的所有变量。
wait	等候所有后台作业。
while	当 *Expression* 变量指定的表达式求值非零时，求值 **while** 与匹配的 **end** 命令序列之间的命令。

相关信息

Korn shell

ksh 和 **stty** 命令。

alias、**cd**、**export**、**fc**、**getopts**、**read**、**set** 和 **typeset** Korn shell 命令。

/etc/passwd 文件。

Bourne shell

bsh 或 **Rsh** 命令，**login** 命令。

Bourne shell **read** 特殊命令。

setuid 子例程，**setgid** 子例程。

null 特殊文件。

environment 文件，**profile** 文件格式。

C shell

csh 命令，**ed** 命令。

alias、**unalias**、**jobs**、**notify** 和 **set** C shell 内置命令。

第 13 章 AIX 文档

本节讨论 AIX 的可用联机文档。

IBM eServer pSeries 信息中心

IBM eServer pSeries 信息中心提供到 AIX 文档的门户网站和对工具以及对资源的访问。从信息中心，提供了发行版 4.3、5.1 和 5.2 的整个 AIX 软件文档库。5.1 和 5.2 中发布的每本书都有 PDF 格式，并提供 5.2 中发布的书籍的摘要。其它工具和资源包括:

- 错误消息数据库，向用户显示错误消息的含义，并在多数情况下，显示如何从错误消息恢复。此数据库还提供 LED 代码的信息和错误标识。
- 资源页面，将用户与证明为对系统管理员、应用程序开发者和用户有用的其它 IBM 和非 IBM Web 站点链接。
- 几种入门指导，向用户提供逐步说明，用于完成系统管理员和用户任务。
- 几个 FAQ（常见问题），向用户提供对常见问题的快速解答。
- 到相关的经常使用的 IBM 文档的链接，包括白皮书、红皮书和有关如 RS/6000 SP 和 AIX 的 HACMP 之类主题的技术报告。
- 到为每个发行版提供的文档搜索工具的链接，以及到发行说明和自述文件的链接。
- 到整个 pSeries 和 RS/6000 硬件文档库的链接。

使用 IBM eServer pSeries 信息中心

要在 AIX 系统上调出信息中心，请执行以下操作:

- 在 AIX 命令行上，请输入:

  ```
  infocenter
  ```

  按下 Enter 键。

  或
- 从 “CDE 帮助” 面板，选择 “信息中心” 图标。

  或
- 转至以下 Web 地址:

  http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base

文档库服务

文档库服务允许您读、搜索和打印联机 HTML 文档，并提供出现在您的 web 浏览器中的库应用程序。该应用程序包含阅读已安装文档的链接，和可用于搜索文本的搜索表单。当您搜索时，结果页面显示搜索的结果，其中有到包含搜索目标字的文档的链接。

从 AIX 5.1 开始，还可以下载书籍的可打印版本。

提供两种类型的表单: 显示安装在搜索服务器上所有卷的全局搜索表单，以及只搜索一组特定卷（如应用程序的手册）的特定搜索表单。

文档库服务允许您搜索已向搜索服务注册的文档。系统管理员注册文档。您无法搜索因特网或搜索服务器上的任何未注册文档。

如果在您的位置写 HTML 文档，则您的系统管理员可将这些文档添加到文档库，这样就可以阅读、搜索和打印文档。

使用文档库服务

可以使用“文档库服务”执行以下任务:

- 要读取您系统的缺省库中安装的文档，请执行以下操作之一:
  - 在命令行输入 `docsearch`。
  - 打开“CDE 桌面帮助”子面板。单击文档搜索服务图标，它看起来像双眼望远镜。

    注: 如果具有 AIX 4.3（或更高版本）CD 的副本，则可以在 PC 上读取。将 CD 插入您 PC 上的 CD-ROM 驱动器。如果文档 CD 是 AIX 4.3.3 或更高版本，请使用 Web 浏览器打开 CD 顶层目录上名为 **readme.htm** 的 CD 文件。如果具有“基本文档 CD”的 AIX 4.3.0 到 AIX 4.3.2 版本，则打开 **usr/share/man/info/en_US/a_doc_lib/aixgen/topnav/topnav.htm** 文件。
- 要打开远程文档服务器上存储的库，请在您的浏览器位置栏中输入以下 Web 地址:

  `http://`*`server_name`*`[:`*`port_number`*`]/cgi-bin/ds_form`

  全局搜索表单打开，在那里可以使用您在 *server_name* 中指定的名称搜索存储在服务器上的文档。还可以按类别查看所有书籍。

  注: 如果端口不是标准 80，则需要输入 *port_number*。

  例如，如果想要搜索在名为 `hinson` 的搜索服务器上的已注册文档，并且它使用端口 80，请输入:

  ```
  http://hinson/cgi-bin/ds_form
  ```

  服务器的搜索表单在您的浏览器中显示后，可以创建书签，它能带您回到服务器。您的系统管理员还可以创建 Web 页面，该页面包含到组织中所有不同文档服务器的链接。

  还能从以下 Web 地址读取和搜索文档: http://www.ibm.com/servers/aix/library。请注意，如果此站点包含 AIX 基本操作系统文档，则它可能不包含您本地文档服务器上安装的其它文档。
- 特定搜索表单通常从 HTML 文档中的搜索链接启动。它们通常出现在应用程序手册或帮助文件的页面上。例如，库中的每个页面都具有搜索链接。单击这些搜索链接中的一个启动特定搜索表单，此表单允许您仅搜索库卷。

在库应用程序打开后，您可单击右上角的**帮助**链接，以获取关于如何使用库的说明。

更改文档库服务语言

缺省情况下，当您打开文档搜索服务或基本库的 CDE 桌面图标时，文档以与您 CDE 桌面使用的相同语言显示。

然而，您可能需要查看使用不同于您桌面所使用语言的文档。例如，桌面以您的本地语言运行，但是手册可能仅有英语版本。可更改您的文档语言，以便文档以与您桌面使用的不同语言显示。

注:

1. 如果您正在从另一个文档中的 HTML 链接打开文档或搜索表单，则下面描述的过程不影响使用的语言。这些步骤仅影响文档搜索服务或基本库桌面图标使用的语言。

2. 确保已安装的文档有您想要使用的语言。

可以通过运行以下命令更改您的文档语言:

```
/usr/bin/chdoclang locale
```

其中 *locale* 是语言环境名称，它是查看和搜索文档的新语言。语言环境名称可在 *AIX 5L Version 5.2 National Language Support Guide and Reference* 的 “Locale Naming Conventions” 中找到。

必须注销，然后再登录以查看语言更改是否生效。

如果正在使用 “CDE 桌面”，则还必须编辑您的 “桌面” 文件 **$HOME/.dtprofile**，以便 **$HOME/.profile** 文件中的文档语言设置将在 CDE 登录期间读取。要这样做，请完成以下步骤:

1. 在 dtpad 编辑器中打开 **.dtprofile** 文件，方法是输入以下命令:

   ```
   dtpad $HOME/.dtprofile
   ```

2. 查找包含以下文本的行:

   ```
   DTSOURCEPROFILE=true
   ```

3. 如果在行的开始有任何注释（#）字符，只删除 # 字符，不要删除整行。如果没有注释字符，则关闭编辑器。
4. 保存已更改的 **.dtprofile** 文件。
5. 注销，再登录回。

例如，如果您要更改您的文档语言为西班牙语（语言环境名称为 es_ES），则输入以下命令:

```
/usr/bin/chdoclang es_ES
```

注销，登录回您的桌面。

在更改您的文档语言后，可以删除语言设置，以便文档将再次以与您桌面使用的语言相同的语言显示。要删除您的语言设置，请输入以下命令:

```
/usr/bin/chdoclang -d
```

注销，登录回您的桌面。

附录. 声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其它国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档描述的内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可证。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

本条款不适用联合王国或任何这样的条款与当地法律不一致的国家或地区: 国际商业机器公司以“按现状”的基础提供本出版物，不附有任何形式的（无论是明示的，还是默示的）保证，包括（但不限于）对非侵权性、适销性和适用于某特定用途的默示保证。某些国家或地区在某些交易中不允许免除明示或默示的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本出版物的新版本中。IBM 可以随时对本出版物中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: (i) 允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及（ii) 允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可证协议或任何同等协议中的条款提供。

有关双字节（DBCS）信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其它可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其它关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

本信息包括在日常商务运作中使用的数据和报告的示例。为了尽可能完整地阐明示例，示例包含个人、公司、商标和产品的名称。所有这些名称都是虚构的，如与实际商务企业使用的名称和地址类似则纯属巧合。

商标

以下术语是国际商业机器公司在美国和 / 或其它国家或地区的商标。

AIX

AIX 5L

IBM

RS/6000

UNIX 是 The Open Group 在美国和其它国家或地区的注册商标。

其它公司、产品和服务名称可能是其它公司的商标或服务标记。

索引

[F]

[G]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

X

Z

[特别字符]

读者意见表

AIX 5L 版本 5.2
系统用户指南: 操作系统与设备

姓名

地址

单位及部门

电话号码

读者意见表

请沿此线
撕下或折起

折起并封口　　请勿使用钉书机　　折起并封口

在此
贴上
邮票

IBM 中国公司上海分公司，汉化部
中国上海市淮海中路 333 号瑞安广场 10 楼
200021

折起并封口　　请勿使用钉书机　　折起并封口

请沿此线
撕下或折起

中国印刷