

KH
科海培训中心
系列教材

UNIX

实用程序手册

北京科海培训中心

1024543

73-87221073 7

34

UNIX 实用程序手册

杨季文 朱巧明 吕 强 编译

钱培德 审校



北京科海培训中心

1991 • 北京

内 容 简 介

UNIX操作系统拥有非常丰富的实用程序。本书介绍UNIX系统的各种实用程序之功能、使用方法和使用实例，对于一些较大型和复杂的实用程序，还介绍了有关理论基础。为了使层次清晰，并便于读者查阅，本书把各种UNIX实用程序归并为八大类：文件处理、调试器、程序设计语言开发工具、系统开发工具、数据库管理系统、文本格式工具、数据通信和写作工作台。

本书层次清楚、结构严谨，并突出技术性和实用性，是广大UNIX用户和计算机专业人员的必备参考书。

编译者的话

UNIX操作系统由Bell实验室人员研制，于1970年开始投入运行。开始时，它并未引起人们太大的注意。后来，由于它的优越性能，引起美国大学界的高度重视，并逐渐在有关大学推广使用。以后又受到许多计算机厂商和软件公司的青睐，从而得到了迅速的发展和广泛的应用。现在，UNIX已成为新一代多用户小型和微型计算机的事实上的标准操作系统。

UNIX操作系统向用户提供了极其丰富的实用程序，其中有些实用程序的功能十分强大。如何应用好这些实用程序，关系到是否能用好UNIX系统，也关系到是否能充分发挥UNIX的潜力。另一方面，实用程序是操作系统向用户提供的重要用户界面，用户要掌握好这一用户界面是至关重要的。目前国内出版了不少UNIX资料，但是还没有系统介绍UNIX系统实用程序的资料，因而使用户深感不便。据我们所知，大多数的UNIX用户只能掌握UNIX实用程序的一部分，其原因之一，就是缺乏相应的资料。为此，我们整理、编译了这本书，把UNIX的实用程序分类归纳，按类别分别进行介绍。

我们把UNIX例行程序分为八类，每类作为一章：

第一章，文件处理。该章介绍实用程序 head、tail、more、pg、cat、past、od、join、sed、egrep和awk的使用方法和功能，还介绍其它一些常用的过滤器。

第二章，调试器。该章介绍调试器dbx、adb和sdb，并介绍与此相关的实用程序 strip和ctrace。

第三章，程序设计语言开发工具。该章介绍实用程序yacc、lex和m4，并给出了语法分析程序和扫描程序的例子。

第四章，系统开发工具。该章介绍系统开发用实用程序lint、make，以及源代码控制系统sccs。

第五章，数据库管理系统。该章介绍QUEL语言和SQL语言，又介绍了DBMS与文件系统的关系，以及格式查询和访问算法等。

第六章，文本格式工具，该章介绍与文本格式有关的实用程序 troff、nroff、tbl、pic和leqn。

第七章，数据通信。该章介绍通信实用程序uucp和cu，并介绍邮件处理和发送新闻的方法。

第八章，写作工作台。该章介绍写作实用程序，包括联机帮助、校对程序和文体分析程序等。

全书的编译工作由杨季文、朱巧明和吕强合作完成，钱培德担任全书的审校工作。另外，夏晓燕、戴建年和蒋青参加了本书的部分工作。本书的形成得到了金传祚先生的许多帮助，特在此表示感谢。

编译者

一九九一年于苏州大学计算机工程系

引 言

本书用通俗的语言介绍了UNIX实用程序。面向的读者主要是一些从事UNIX环境下工作的人员。本书中大量使用了一些常用术语。下面是这些术语的简要说明。

shell: shell是一个实用程序。当用户注册进入系统时,它就开始工作,它首先显示提示符,然后等待用户输入命令。它分析命令行内容,检索相应的程序文件并开始执行之。

shell comments: 所有以“#”符号开头的行,shell都认为是注释行。而且,若shell在一个命令行上遇到该字符,它就把余下的内容作为注释。

standard input and standard output: 大多数UNIX程序书写时都从缺省文件中读入数据。这个缺省文件就是标准输入。类似地,缺省输出文件就是标准输出。所有的错误信息都写在标准错误文件上。除非重定向,否则这些标准输入、标准输出以及标准错误文件都指用户终端。

redirection: 缺省的输入和输出文件都可以用输入、输出重定向进行改变。“>”符用于重定向标准输出,“<”符用于重定向标准输入。例如,命令`comm<infile>outfile`表示comm命令从infile中读入内容,把结果输出到outfile中。

pipe: 管道(|)用于连接两个命令,它把第一个命令的输出作为第二个命令的输入。例如:`comm1<infile | comm2>outfile`,它表示comm1从infile中读入内容,把其结果输出到comm2,comm2把其输出存入文件outfile中。

filter: 过滤器是指一个程序,该程序能从标准输入中读入数据并能把数据写到标准输出文件中。一般情况下,过滤器的输出依赖于其输入。

cat: 命令cat在本书中经常使用,其功能是显示一个文件的内容或者拼接两个或多个文件。

login id: 当用户终端联到机器上运行UNIX系统时,系统就提示出“login:”。用户就键入分配的登录号,系统再提示通行字,系统上的用户都用其登录号来指明。

vi: vi是UNIX环境下最流行的屏幕编辑程序。

regular expression: 正则表达式。许多UNIX实用程序都使用正则表达式。正则表达式提供了规定匹配模式的一种方法。shell也使用这些表达式产生文件名。正则表达式使用一组称作超级字符(meta character)的字符,这些超级字符与其它字符的含义不同,超级字符的解释由具体的实用程序规定。

options: 可选项。大多数UNIX实用程序都接受命令行上的可选项。这些可选项通常都以“-”符指明。可选项增加了实用程序的可塑性。有些可选项不必用“-”符,可用“+”符打头,也有些可选项什么符也不需要。

on-line help: 在线帮助在大多数UNIX系统中都有。任何命令的文档都可用man命令打印出来。例如,要找出sed命令的全部文档说明,则键入`man sed`即可。

目 录

引言

第一章 文件处理

第一节 head和tail实用程序	(1)
第二节 more和pg实用程序	(1)
第三节 cat实用程序	(3)
第四节 past实用程序	(4)
第五节 od实用程序	(6)
第六节 join 实用程序	(8)
第七节 sed 实用程序	(10)
第八节 egrep 实用程序	(17)
第九节 awk实用程序	(19)
第十节 其它常用的过滤器	(27)
第十一节 本章小结	(30)

第二章 调试器

第一节 调试器dbx	(31)
第二节 调试器adb	(41)
第三节 符号调试器sdb	(46)
第四节 删除调试信息的程序strip	(53)
第五节 跟踪检查程序ctrace	(53)

第三章 程序设计语言开发工具

第一节 理论基础	(61)
第二节 语法分析程序和扫描程序的例子	(62)
第三节 实用程序yacc和lex	(64)
第四节 宏处理程序m4	(96)

第四章 系统开发工具

第一节 C语言的检查程序lint	(102)
第二节 维护程序make	(113)
第三节 源代码控制系统SCCS	(122)
第四节 make和SCCS	(133)
第五节 直方图文件处理工具gprof和prof	(134)

第五章 数据库管理系统

第一节 理论基础	(140)
第二节 数据库系统的设计	(141)
第三节 QUEL查询语言	(142)
第四节 SQL 查询语言	(147)
第五节 宿主语言接口	(151)

第六节 DBMS与UNIN 文件的相互作用.....	(155)
第七节 DBMS与UNIX 文件系统.....	(155)
第八节 建立数据项的格式.....	(155)
第九节 格式查询.....	(155)
第十节 报表实用程序.....	(156)
第十一节 访问算法.....	(159)
第十二节 其它特性.....	(159)
第十三节 第四代系统.....	(160)
第十四节 DBMS的选择.....	(160)
第六章 文本格式工具	
第一节 troff和nroff 的使用.....	(163)
第二节 表格前处理器tbl.....	(175)
第三节 pic前处理器.....	(180)
第四节 eqn 前处理器.....	(185)
第五节 本章小结.....	(189)
第七章 数据通信	
第一节 uucp: 程序和文件.....	(190)
第二节 管理程序.....	(195)
第三节 支持文件.....	(196)
第四节 cu: 与远程计算机连接.....	(198)
第五节 用户间的通信.....	(199)
第六节 邮件处理.....	(199)
第七节 发送新闻.....	(205)
第八节 写信给另一用户.....	(206)
第九节 分布式文件系统.....	(207)
第十节 本章小结.....	(207)
第八章 写作工作台	
第一节 准备工作.....	(208)
第二节 WWB程序的分类.....	(208)
第三节 联机帮助.....	(209)
第四节 校对程序.....	(218)
第五节 文体分析程序.....	(229)
第六节 ww b程序.....	(231)
第七节 murky和continge程序.....	(231)
第八节 定做WWB环境.....	(231)
第九节 关于WWB.....	(232)
第九章 特殊实用程序	
第一节 curses软件包.....	(233)
第二节 计算器工具.....	(242)
第三节 文件归档.....	(246)

第四节	库的归档和维护.....	(252)
第五节	基本统计.....	(254)

第一章 文件处理

UNIX为用户提供了许多检查文件内容的工具。这些工具都有一些共同的特点和一些不同类型的可选项。它们能对标准输入文件进行读，也能对标准输出设备进行写。这样的实用程序，也称作过滤器（filter），它们虽然只提供线性访问功能且检查操作速度慢，但是足以满足小型数据库及文件的数据处理需要。它们不修改输入文件的内容，但可以把结果方便地写到标准输出设备上。下面几节将介绍这些实用程序的一些非常有用的命令。

第一节 head和tail实用程序

head用于显示选定文件从起始位置处开始的指定的记录数。最常使用的可选项是指定显示的记录数。如果系统中无head，则用户可用more或pg命令来完成head的功能。tail与head类似，它只是从文件的尾部开始显示指定的记录数。在大多数UNIX版本中，tail只允许显示有限的记录数。这个极限值与要显示的文件记录大小有关。下面是head用法的一个例子：

```
head -5 file1
john      23      123-38-9073
michael   23      321-12-8921
james     25      142-32-4564
paul      26      231-34-1212
tom       23      123-12-1245
```

在上面这个例子中，我们用可选项“-5”来显示文件file1中的开头五个记录。在缺省情况下，则显示文件十个记录。

第二节 more和pg实用程序

more和pg命令用于检查一个文件的内容。它们可以用可选项指出在屏幕上显示的行数，缺省时，一次显示一屏内容。例如：

```
more bigfile
```

它把bigfile文件的内容从头开始在屏幕上显示满一屏。再如：

```
more -10 bigfile
```

它表示一次显示10行内容于屏幕上。more和pg实用程序还提供了检查正则表达式（regular expression）的机制，more和pg的大多数可选项是一样的。more较为常用，尤其在BSD（Berkeley Software Distribution）版本中经常使用，pg通常在UNIX系统V版中使用。

more和pg提供了许多可选项命令，我们这里仅介绍它们最常用的使用方法。

一、检查文件内容

用户要检查一个文件内容，只要在more或pg命令后跟一个被检查的文件名即可。其格式如下：
`more 文件名`
或者
`pg 文件名`

在用户键入完上述命令后，如果文件内容超过一屏，则系统显示满一屏后等待用户的响应。若用户使用的是pg命令，则键入回车键显示下一屏内容，若用户使用的是more命令，则键入空格键显示下一屏内容。用户若键入组合键`ctrl+d`，则系统显示下半屏内容或文件的下面十一行内容。若用户要跳过若干行显示，则用可选项`i`命令，其格式为“`ni`”，其中`n`为要跳过的行数。pg命令提供了可选项`l`命令。它用于屏幕的上滚和下滚。上滚的格式为“`+nl`”，下滚的格式为“`-nl`”，其中`n`表示滚动的行数。当用户使用“`nl`”命令时，则表示从指定文件的第`n`行起显示一屏内容，这里的“`n`”必须是一个无符号的整数。另外，pg还可使用可选项“`$`”命令来显示指定文件的最后一屏内容。

二、检索正则表达式

more和pg实用程序为用户提供了对文件中正则表达式进行检索的功能。在pg中，命令“`i/正则表达式/`”用于往前检索第`i`个正则表达式；命令“`i?正则表达式?`”用于往后检索第`i`个正则表达式。在more中，只有“`i/正则表达式/`”命令，而无“`i?正则表达式?`”命令。

三、退至shell

more和pg实用程序允许用户退到shell并且用“`! command`”命令来执行UNIX命令，这里的`command`为UNIX的命令。

四、more和pg用法举例

下面是more和pg用法的一些例子。

例1：

```
man cat | more -200 > catman
```

该命令完成把cat命令显示的前200行内容存于文件catman中。

例2：

```
pg -1 file1
john    23  123-38-9073
1/^j/
```

此例中，我们用pg命令查看file1的内容，每次只显示一行，然后再显示以字母“j”开头的一行内容。

第三节 cut 实用程序

大家知道，文件的每个记录都可认为是由一串字符集或者若干个字段(field)组成。字段与字段间用分隔符分隔。在数据处理的过程中，用户经常要对文件中的某字符位置或字段进行存取。UNIX的cat实用程序提供了这一功能。cut的使用格式如下：

```
cut -f list [-d char] file1 file2...
```

或者

```
cut -c list file1 file2
```

其中list是由分号分隔的一组整数组成，例如，1,3,-5,7。它们用于指出(映象)位置号。cut的第一种格式表示用字段分隔符char把文件中指定的字段进行分离；第二种格式用于分离指定位置上的字符。第二种格式中唯一的可选项-c表示指定要映射的字符位置表。下面就是该格式的一个例子。

```
cat file1
John, P      23  123-38-9073  75
Phil, R      23  321-12-8921  59
Mike, T      25  142-32-4564  32
Rick, L      26  231-34-1212  65
Jack, T      23  123-12-1245  90
```

```
cut -c1-4, 10-15 file1
```

```
John 123-3
Phil 321-1
Mike 142-3
Rick 231-3
Jack 123-1
```

file1文件共有四个字段，每个字段间都用制表符分隔。用cut命令带位置参数后，就可对文件进行剪裁。下面再来看剪裁file1的另一个例子。

```
cut -f1, 3-4 file1
```

```
John, P 123-38-9073 75
Phil, R 321-12-8921 59
Mike, T 142-32-4564 32
Rick, L 231-34-1212 65
Jack, T 123-12-1245 90
```

字段分隔符在cut中,缺省情况下为制表符。但也可用带-d可选项命令跟分隔符来完成字段之间的分隔。下面的例子就是用空格符来分隔各字段。

```
cat file1
John, P 23 123-38-9073
Phil, R 23 321-12-8921
Mike, T 25 142-32-4564
Rick, L 26 231-34-1212
Jack, T 23 123-12-1245
```

```
cut -f2, 1, 3 -d " " file1
John, P 23 123-38-9073
Phil, R 23 321-12-8921
Mike, T 25 142-32-4564
Rick, L 26 231-34-1212
Jack, T 23 123-12-1245
```

当用户指定了字段的定界时,只有一些对shell有特殊含义的字符才能用于引号内作分隔符,否则引号就不必需要。若一个记录行的长度超过了1023个字符,则cut就给出错误诊断信息。

第四节 paste 实用程序

paste实用程序的功能是把多个文件联接成一个文件。它与UNIX的cat实用程序类似,只不过cat是把多个文件逐个逐个地“垂直”连接在一起,而paste则把多个文件“水平”地拼成一个文件。paste的最后输出文件中的记录数为拼接文件中记录数最多的一个文件相同,而cat则为多个文件的记录数之和。paste命令的格式如下:

```
paste [-s] [-d char] file1 file2...
```

输出文件包含了最长的一个输入文件的所有记录内容。这些文件联接在一起后,除了最后一个文件的记录项是回车符之外,其余每个记录的回车符都用制表符代替。制表符也可通过可选项-d后跟字符“char”来替代。当用户用“-”符号来简化文件名时,它表示文件记录来自标准的输入。-s可选项非常有用,它表示从同一个文件读入记录,然后把它们联接在一起。例如有两个文件f1和f2,其内容用cat命令显示如下:

```
cat f1
a
b
c
d
```

```
cat f2 | 1
2
3
4
5
6
```

例 1:

```
past f1 f2
a      1
b      2
c      3
d      4
5
6
```

从这里可见，past命令把f1和f2文件中各记录并列排列起来，f1的每个记录的回车符被制表符替代了。

例 2:

```
past -d " " f1 f2
a      1
b      2
c      3
d      4
5
6
```

在例 2 中，f1 的每个回车符被空格替代。这里用了-d 可选项。

例 2:

```
past -s f1
a      b      c      d
```

在这例中，我们用了-s 可选项，它把f1中的每个记录的回车符用制表符代替。

例 4:

```
cat f1 | past - -
a      b
c      d
```

在这里，输入由两个文件组成，两个文件的记录都来自标准输入。用“-”可选项通常可以把n个记录的文件转换成n/x个记录，其中x就是命令行上“-”符的个数。输出文件

的第一个记录项包含了x个记录。第二个记录项包含了输入文件的下面x个记录。

下面我们介绍cut和paste命令的联用。

cut和paste可以联用，用来处理一些以列（column）结构的文件。例如，把文件file1的第一列（即第一个字段）内容移到记录的最后。操作如下：

```
cat file1
John, P 23 123-38-9073 75
Phil, R 23 321-12-8921 59
Mike, T 25 142-32-4564 32
Rick, L 26 231-34-1212 65
Jack, T 23 123-12-1245 90
```

```
cut -f1 -d " " file1>file2
cut -f2-4 -d " " file1>file3
paste -d " " file3 file2>file1
cat file1
```

```
23 123-38-9073 75 John, P
23 321-12-8921 59 Phil, R
25 142-32-4564 32 Mike, T
26 231-34-1212 65 Rick, L
23 123-12-1245 90 Jack, T
```

在这例中，我们先把file1的第一个字段从文件中分离出来，起名为file2文件，然后，把余下的字段从file1中分离出来，起名为file3文件，最后把file3和file2用paste联接起来，重定向到file1文件中，即完成了搬移。

第五节 od 实用程序

在程序设计中常常会遇到这样一个令人头痛的事，就是要在一个文件中显示非图形字符。由于这些字符可能转换成某些终端设置，因此，要显示一个包含这些字符的文件会产生一些问题。od为我们提供了解决这一问题的各种方法。od命令的格式如下：

```
od [-bcdosx] [file] [ [+ ] offset [ . ] [b] ]
```

下面我们介绍用od命令来显示一个目录文件。目录文件在UNIX系统V中是由16字节长的域组成。该域的头2个字节表示文件的i号。i号用于定位一个被访问文件的结点。余下的14个字节为文件名。若用户用cat命令来显示该目录文件的内容，则因为每个域的头2个字节可能为不可表示的图形ASCII字符，所以用户会看到一些怪字符。若用od命令，则就一目了然。具体操作及显示结果如下：

```
ls dirfile
```

```
dummy.c
```

```
a.out
```

```
core
```

```
test.c
```

```
od -c, dirfile
```

```
00000000 \0 b . \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000020 6 a . . \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000040 \0 211 s . t . c \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000060 003 340 d u m m y . c \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000100 036 @ a . o u t \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000120 \0 206 c o r e \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000140 001 f t e s t . c \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000160
```

可选项-c表示od以ASCII码方式解释字节内容。第一栏表示字符的位置，这里用八进制表示，第一行从00000000开始，即从第一个字符位置开始显示文件内容。第二行从八进制数00000020即十进制数16开始，也即从目录文件的第二个字段开始显示字段的内容。ASCII图形字符恰当地显示出来。如果字符没有图形符表示，则od就用换码符“c”显示出来，或者用3位八进制数表示。“\0”表示NULL。从上面的显示结果可以看出，每个字段的第一、二个字节有时其内容为非图形符。

当od命令带-d可选项时，则文件字段中每两个字节就连在一起，并用一个十进制数值表示。例如对上述目录文件用带-c和-d可选项的od命令，则其显示内容如下：

```
od -cd dirfile
```

```
00000000 00098 11776 000000 000000 000000 000000 000000 000000 000000
          \0 b . \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000200 13921 11822 000000 000000 000000 000000 000000 000000 000000
          6 a . . \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000400 00137 29486 29742 25344 000000 000000 000000 000000 000000
          \0 211 s . t . c \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000600 00992 25717 28013 31022 25344 000000 000000 000000 000000
          003 340 d u m m y . c \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000800 07744 24878 28533 29696 000000 000000 000000 000000 000000
          036 @ a . o u t \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000A00 00134 25455 29285 000000 000000 000000 000000 000000 000000
          /0 206 c o r e \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000C00 00358 29797 29556 11875 000000 000000 000000 000000 000000
          001 f t s t c . c \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
00000E00
```

在上例中，第二行显示了ASCII字符，第一行表示同一字符的十进制值。例如ASCII字符“\0”和“b”表示的二进制值分别为00000000和01100010。以字形式表示为00000000 01100010，其十进制数为98。该值显示于第一行，值得注意的是，若机器中字的表示与上述格式不同，则十进制数值也不同。目前有些软件包产生的文件是以二进制形式存储的，在这种文件中的数据可用od命令带参数-d来检查文件中的整型数据。可选项-b使od显示的字符

以八进制表示，可选项-o则以一个字的八进制形式显示两个字符。可选项-x把字符以十六进制形式表示，可选项-s用于解释符号位，其余同可选项-d一样显示字符。

同时，od命令还提供了另外一个功能，它可以只检查文件的某一段内容。该功能通过od命令带字节偏移的可选项来完成。偏移以八进制形式表示。如果偏移量后再跟“a.”，则表示该偏移量值为十进制数。若偏移后跟字符“b”，则表示该偏移量要乘512。最后，如果数据以管道形式送给od，而用户则需从某一偏移处开始检查，则一定要在偏移量处加上“+”。下面是od带偏移的例子。它们都是对前例中的文件dirfile而言的。

例1：从文件dirfile偏移量8（八进制数10）处开始显示文件内容。

```
od -c dirfile 10
0000010 \0 \0 \0 \0 \0 \0 \0 \0 6 a . . \0 \0 \0 \0
0000030 \0 \0 \0 \0 \0 \0 \0 \0 \0 211 s . t . c \0
0000050 \0 \0 \0 \0 \0 \0 \0 \0 003 340 d u m m y .
0000070 c \0 \0 \0 \0 \0 \0 \0 \0 036 @ a . o u t \0
0000110 \0 \0 \0 \0 \0 \0 \0 \0 \0 206 c o r e \0 \0
0000130 \0 \0 \0 \0 \0 \0 \0 \0 \0 001 f t e s t . c
0000150 /0 /0 /0 /0 /0 /0 /0 /0
0000160
```

例2：从dirfile偏移量为10字节处开始显示文件内容。

```
od -c dirfile 10.
0000010 \0 \0 \0 \0 \0 \0 \0 6 a . . \0 \0 \0 \0 \0 \0
0000026 \0 \0 \0 \0 \0 \0 \0 \0 211 s . t . c \0 \0 \0
0000042 \0 \0 \0 \0 \0 \0 \0 003 340 d u m m y . c \0
0000058 \0 \0 \0 \0 \0 \0 \0 036 @ a . o u t \0 \0 \0
0000074 \0 \0 \0 \0 \0 \0 \0 206 c o r e \0 \0 \0 \0
0000090 \0 \0 \0 \0 \0 \0 \0 001 f t e s t . c \0 \0
0000106 \0 \0 \0 \0 \0 \0 \0
0000112
```

第六节 join 实用程序

Join实用程序支持关系式数据库运算natural join。它能从两个基于某些公共域定义 的字段文件中提取数据。假设有两个文件，文件名分别为suppliers和parts。suppliers有三个 字段，分别表示供货商号supplier number，供货商名称和供货的货号。parts文件也有三个 字段，它们分别为货号，货名以及货的描述。如果此时你要建立一个供货商名称及货名的文 件，则需要 在每个文件的货号信息基础上对两个文件作一比较。由于两个文件中的货号是类 型相同、表示的事物也相同，因此我们说，这里两个文件中的货号是定义在同一个域上的。 基于定义在同一域上的两个字段的比较，可用实用程序join实现。join的格式为：

join [options] file1 file2

file1和file2为两个输入文件。这些文件能用某一可比较的字段进行排检。每个输出行由公共域、file1的记录项及file2的记录项组成。该输出文件随join的可选项不同而不同。下面是一个建立有供货商及货名的文件的例子。

```
cat suppliers
s0101 : PRS, Inc : p123
s0230 : Tare, Inc : p123
s1213 : Unicomp, Corp. : p223
s0202 : Strakers, Inc : p23
s3030 : Ritecore, Inc : p234
```

```
cat parts
p123 : Resistors : multi
p233 : Boards : brown
p234 : Gloves : blue

join -j1 3 -j2 1 -o 1.2 2.2 -t : suppliers parts
PRS, Inc : Resistors
Tare, Inc : Resistors
Unicomp Corp. : Boards
Ritecore, Inc : Gloves
```

从这里可以看出，输出文件是如何通过公共域来连接两个文件中的记录的。由于在 parts 文件中无货号p23这个记录，因此在输出文件中就无suppliers文件中的s0202: Strakers, Inc : p23这个记录。如果在一个文件中有重复的关键字段（例如在上例中的suppliers 文件中的p123），则join在操作时把对该字段相对应的另一个文件中的字段内容进行复制（例如输出文件中有两个Resistors字段）。

另外，上例中使用了下面一些常用的可选项。-j可选项，它用于指出所要连接的公共字段。“-j1 3”表示基于第一个文件的第三个字段进行连接，“-j2 1”表示基于第2个文件的第一个字段进行连接。-o可选项，它规定了要输出的一组字段的格式。该可选项中参数的格式为“m·n”，其中m表示文件序号，n表示在文件中字段的位置。每一组参数用空格符分开。例如“-o1.2”表示输出第一个文件中每个记录项的第2个字段。-t可选项用于规定记录中字段与字段之间的分隔符，在上例中为冒号(:)。若t后参数缺省，则为空格或者制表符或回车换行符。

有一种情况，使用join命令非常有用。在不丢失记录的情况下，把一个文件中的数据追加到另一个文件中，生成一个与输入文件记录数不匹配的输出文件。此时，用户可用可选项-an和-e str对原文件进行操作。可选项-an表示在n序号文件中为每个不成对的记录生成一个记录项，n一般取1或者2。-e str可选项用于把一些不成对的记录用字符串str进行替补。例如，把parts文件中的记录追加到suppliers文件中，对一些不匹配的记录，用NA字符串补缺，则join命令格式如下：