# 第5章 实用程序初步

经过第3章的学习，读者应该掌握了在Ubuntu字符界面下进行的基本操作。本章将介绍一些常用的实用程序来帮助读者在字符界面下更好地使用Ubuntu。它们分别是多列内容输出column，文件内容查找grep，基本数学计算bc，文件内容排序sort，文件内容比较uniq、comm和diff，文件内容替换tr，单行数据编辑sed和数据操作工具awk。

**本章学习目标**

- 掌握如何使用column输出多列内容；
- 熟练掌握如何使用grep在文件中查找不同的内容；
- 熟练掌握如何使用系统自带的计算器bc进行数学运算；
- 熟练掌握使用sort对文件内容进行排序；
- 熟练掌握文件内容比较命令uniq、comm和diff用法；
- 掌握文件内容替换命令tr的用法；
- 掌握单行数据编辑工具sed的用法；
- 掌握数据操作工具awk的基本用法。

## 5.1 多列内容输出

在字符界面中查看文件的内容时，数据大都是靠屏幕最左方单列由上至下显示，除此列显示数据之外的屏幕区域均为空白，因此需要在屏幕上调整文件内容的输出样式，以实现更多内容在同一屏幕上显示。本节将介绍如何使用column实现文件按多列格式输出。

### 5.1.1 按多列格式输出

(1) 执行以下命令创建一个名为ex0501_column的文件。

```
vi ex0501_column
```

在当前文件中输入以下信息并保存，注意其格式为单列多行。

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```

Sunday

(2) 执行 more 命令查看文件内容。

```
more ex0501_column
```

结果如图 5-1 所示,文件内容以单列显示。

(3) 执行以下命令以完成用 column 程序显示文件内容。

```
column ex0501_column | more
```

结果如图 5-2 所示,文件内容以多列显示。

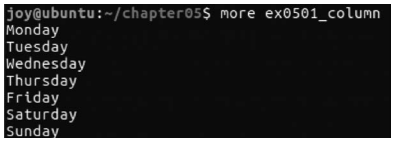


图 5-1　使用 more 查看时单列显示效果

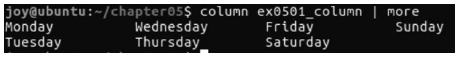


图 5-2　使用 column 的多列显示效果

### 5.1.2　按不同行列顺序

使用 column 显示文件时,可以选择带-x 选项和不带此选项,以得到不同的显示效果。

(1) 首先执行以下命令。

```
column -x ex0501_column | more
```

效果如图 5-3 所示。

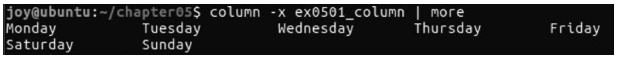


图 5-3　带-x 选项的效果

(2) 再次执行以下命令。

```
column ex0501_column | more
```

效果如图 5-4 所示。


图 5-4　不带-x 选项的效果

可以注意到,带-x 和不带-x 的输出是不同的。其中:带-x 选项的输出是先从左到右显示,再从上到下显示(即先行后列);不带-x 选项的输出是先从上到下显示,再从左到右显示(即先列后行)。

## 5.2　文件内容查找

在第 3 章我们已经学习了 grep 的基本功能,本节将介绍 grep 的其他功能。

### 5.2.1　在多个文件中查找

如果用户想要在多个文件中查找指定内容,可以使用 grep 命令来实现,其格式如下。

```
grep string file1 file2…
```

其中：参数 string 为待查找的内容(通常称其为目标字符串)，file1、file2 则用待查找的文件名代替，省略号表明还可以添加更多文件。

请按以下步骤实现在多个文件中查找。

(1) 首先，使用 vi 命令创建两个文件。

ex0502_01_grep 文件内容为：

```
Check the facts for accuracy.
Check for formatting of references.
```

ex0502_02_grep 文件内容为：

```
Give me my bag.
Give me my computer.
```

(2) 然后，执行下列四条命令。

```
grep e ex0502_01_grep ex0502_02_grep
grep f ex0502_01_grep ex0502_02_grep
grep G ex0502_01_grep ex0502_02_grep
grep Z ex0502_01_grep ex0502_02_grep
```

结果如图 5-5 所示。其中：第 1 条命令结果表明在两个文件中都查找到了“e”；第 2 条命令结果表明只在文件 ex0502_01_grep 中查找到了“f”；第 3 条命令结果表明只在 ex0502_02_grep 文件中查找到了“G”；第 4 条命令无任何输出结果，这是因为在 2 个文件中都未查找到“Z”。

如果想在当前目录下的所有文件中查找字符串，则需要用到通配符“*”。通配符“*”表示当前目录下的所有文件，它指示 grep 命令要在当前目录下的所有文件中查找。下面给出一个关于 grep 命令用法的例子。

假定需要在当前目录下的所有文件中查找字符串 computer，可执行以下命令。

```
grep computer *
```

结果如图 5-6 所示，即在当前目录下的所有文件中只有 ex0502_02_grep 文件中包含字符串 computer。

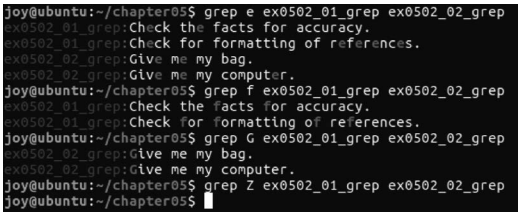


图 5-5　在多个文件中查找

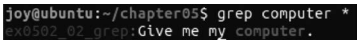


图 5-6　在当前目录下的所有文件中查找

## 5.2.2　在文件中查找多个单词

5.2.1 节已介绍在文件中查找单个单词，本节将学习如何查找多个单词。具体步骤如下。

(1) 首先，使用 vi 命令创建 ex0502_03_grep 文件，内容如下。

```
my name is joy chen
(注意此行为空行)
This is a test
and
^a
txt
```

(2) 然后,执行以下命令查找字符串 joy chen,注意在 grep 命令中字符串两侧为单引号。

```
grep 'joy chen'ex0502_03_grep
```

结果如图 5-7 所示,因为字符串两侧有单引号,所以 grep 命令会将“joy chen”这个整体识别为目标字符串。

如果“joy chen”两侧不加单引号,grep 命令会将“joy”和“chen”作为分开的两个参数传递给 grep,即“joy”将被认为是目标字符串,“chen”将被认为是待查找的第 1 个文件,“ex0502_03_grep”将被认为是待查找的第 2 个文件。然而,因为当前目录下不存在名为“chen”的文件,所以系统会显示警告信息,提醒此文件不存在,但仍会在正常文件 ex0502_03_grep 中查找并显示结果。如图 5-8 所示。

```
joy@ubuntu:~/chapter05$ grep 'joy chen' ex0502_03_grep
my name is joy chen
```

图 5-7　查找字符串“joy chen”

```
joy@ubuntu:~/chapter05$ grep joy chen ex0502_03_grep
grep: chen: No such file or directory
ex0502_03_grep:my name is joy chen
```

图 5-8　在文件查找字符串“joy chen”(不加单引号)

## 5.2.3　查找单词时忽略字母的大小写

grep 命令还可以实现区分目标字符串中字母大小写的功能。

(1) 首先,执行以下命令。

```
grep Joy ex0502_03_grep
```

按 Enter 键后没有输出结果,说明在 ex0502_03_grep 文件中查找不到字符串“Joy”。

(2) 然后,执行以下命令。

```
grep -i Joy ex0502_03_grep
```

结果如图 5-9 所示,-i 选项可以让 grep 在查找字符串时忽略字母的大小写(即会认为大写字母“J”和小写字母“j”是同一个字符)。可以看到,虽然我们的目标字符串是“Joy”,但仍显示了包含“joy”的行。

```
joy@ubuntu:~/chapter05$ grep -i Joy ex0502_03_grep
my name is joy chen
```

图 5-9　带-i 选项的结果

## 5.2.4　查找目标内容的文件名

接下来介绍一些 grep 的其他用法,具体如下。

(1) 使用-v 选项。

在 grep 命令中,如果使用-v 选项,则输出不包含目标字符串的行。

请执行以下命令。

```
grep -v joy ex0502_03_grep
```

```
joy@ubuntu:~/chapter05$ grep -v joy ex0502_03_grep
This is a test
and
^a
txt
```

图 5-10　带-v 选项的结果

结果如图 5-10 所示，可以注意到 ex0502_03_grep 文件中包含“joy”的那行未被输出。

(2) 使用-l 选项。

如果使用-l(连字符和小写字母 l)选项，那么只输出包含目标字符串文件的文件名。

请执行以下命令。

```
grep -l joy ex0502_03_grep
```

结果如图 5-11 所示，ex0502_03_grep 文件中包含“joy”，因此它的文件名被输出了。

(3) 使用-n 选项。

如果使用-n 选项，那么会输出文件中包含目标字符串的行及其行号，两者以冒号隔开。

请执行以下命令。

```
grep -n joy ex0502_03_grep
```

结果如图 5-12 所示，其中冒号之后的内容表示 ex0502_03_grep 文件中包含“joy”，冒号之前的“1”表示“joy”位于第 1 行。

```
joy@ubuntu:~/chapter05$ grep -l joy ex0502_03_grep
ex0502_03_grep
```

图 5-11　带-l 选项的结果

```
joy@ubuntu:~/chapter05$ grep -n joy ex0502_03_grep
1:my name is joy chen
```

图 5-12　带-n 选项的结果

## 5.2.5　使用正则表达式

正则表达式是一种特殊的字符串，可以用于文件内容的查找。下面将正则表达式使用到 grep 中。表 5-1 给出了 3 个在正则表达式中具有特殊意义的字符以及它们的功能。

**表 5-1　具有特殊意义的字符**

| 字　符 | 功　能 | 字　符 | 功　能 |
|---|---|---|---|
| . | 匹配单个普通字符 | $ | 指示某行的末尾 |
| ^ | 指示某行的起始 | | |

下面根据表 5-1 的内容，给出一些正则表达式的用法。

**1. 查找以某一字符开头的行**

(1) 首先，创建 ex0502_04_grep 文件，其内容如下。

```
apple
^appear
hat
ht
```

(2) 然后，执行以下命令在文件中查找所有以字符“a”开头的行。

```
grep '^a'ex0502_04_grep
```

结果如图 5-13 所示，可以发现 apple 被输出了，但是以“^a”开头的那行未被输出，因为“a”不是该行的第 1 个字符。

(3) 最后，执行以下命令。

```
grep '\^a'ex0502_04_grep
```

结果如图 5-14 所示。与(1)不同的是,在这里“^”不被 grep 认为是特殊字符,因为如果“^”的前面紧接着的是“\”(反斜杠),那么“^”将会失去其特殊字符的作用,而被 grep 认为是普通字符,所以此命令的意思是在 ex0502_04_grep 文件中查找以“^a”开头的行。


图 5-13　查找以“a”开头的行


图 5-14　查找以“^a”开头的行

**2. 查找以某一字符结尾的行**

执行以下命令在文件中查找以字符“t”结尾的行。

```
grep 't$' ex0502_04_grep
```

结果如图 5-15 所示,因为“$”是特殊字符,用来规定行的末尾字符,所以可以看到输出了所有以“t”结尾的行。


图 5-15　查找以“t”结尾的行

**3. 查找指定长度的行**

(1) 首先,创建 ex0502_05_grep 文件,其内容如下。

```
line
lineorline
```

(2) 然后,执行以下命令。

```
grep '^....$' ex0502_05_grep
```

结果如图 5-16 所示,4 个“.”代表 4 个字符,“^”指明行的起始,“$”指明行的末尾,可以看到查找到了长度为 4 的行。

**4. 查找指定长度的字符串**

执行以下命令。

```
grep '....' ex0502_05_grep
```

结果如图 5-17 所示,由于未指明行的起始和末尾,grep 将会查找文件中所有长度为 4 的字符串,而不是只查找长度为 4 的行。


图 5-16　查找长度为 4 的行


图 5-17　查找长度为 4 的字符串

## 5.3　基本数学运算

实用程序 bc 可以进行基本的数学运算,本节将介绍使用 bc 进行整数运算和浮点运算。

### 5.3.1　整数运算

(1) 首先执行以下命令以启动实用程序 bc。

```
bc
```

结果如图 5-18 所示，出现了一段关于自由软件基金会(Free Software Foundation)的版权信息，没有命令提示符，且光标停留在新的一行并不断闪烁，说明系统在等待用户输入运算表达式。


图 5-18　调用实用程序 bc

(2) 再输入以下表达式，然后按 Enter 键。

```
6 + 9
```

如图 5-19 所示，bc 立即输出了 6＋9 的计算结果 15。

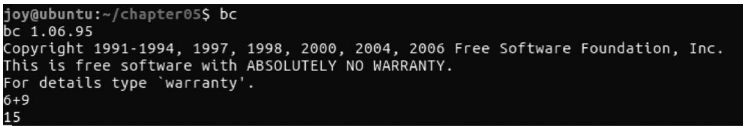


图 5-19　计算 6＋9

(3) 除了加法，bc 还可以做整数运算中的乘法、除法、减法和幂运算等，接下来输入以下表达式。

```
6 * 6
8/2
100 - 10
2^4
```


图 5-20　用 bc 做整数运算

如图 5-20 所示，每输入一个运算式，按 Enter 键后，立即会在新行中显示该运算式的结果。最后一条 2^4 是幂运算，意思是进行 2 的 4 次幂运算，即 2×2×2×2，可以看到结果为 16。

## 5.3.2　浮点运算

在 5.3.1 节中，如果使用 bc 做除法运算，默认输出商的整数部分，小数部分不被输出。但实际上我们可以通过指定 scale 的值(scale 的默认值为 0)，来调整输出商的小数位数。

(1) 先查看整数形式的商，输入以下表达式并按 Enter 键。

```
12/7
```

如图 5-21 所示，默认的运算结果为 1(它是 12/7 的商的整数部分)。

(2) 接着执行以下命令指定变量 scale 的值，并再次计算 12/7。

```
scale = 2
12/7
```

如图 5-22 所示，这里设置了 scale＝2，即设置了输出 2 位小数，可以看到运算结果为 1.71。


图 5-21　scale 默认值的结果


图 5-22　指定 scale 值后的结果

(3) 若想退出 bc,则只要执行以下命令即可。

```
quit
```

图 5-23 为成功执行 quit 后的界面。


图 5-23　退出 bc

bc 的运算顺序与基本数学运算一样,还可以用小括号来调整运算的优先级,读者可以输入以下运算式进行练习。

```
(1 + 2) * 3
8/(6 - 2)
2^(1 + 1)
```

## 5.4　文件内容排序

实用程序 sort 可以实现文件内容的排序,本节将介绍 sort 的几种不同用法。

由于 sort 的标准排序规则是按 ASCII 码表的值升序排列,因此我们需要先了解 ASCII 码。ASCII 码是美国信息交换标准代码(American Standard Code for Information Interchange),它将 128 个人们最常用的符号依次编号。在 Ubuntu 中,我们可以使用以下命令查看关于 ASCII 码的详细信息。

```
man ascii
```

图 5-24 所示为 ASCII 码表。可以看到在表的表头中,“Char”表示被编码的符号,“Oct”表示编码数值的八进制形式,“Dec”表示十进制形式,“Hex”表示十六进制形式。比如,表中右侧第 2 行的大写字母“A”的十进制 ASCII 码值为 65。

因为不同的系统环境会影响 sort 的排序结果,所以我们在使用 sort 前,需要执行以下命令来还原传统的排序规则(按 ASCII 码值排序)。

如果在 bash/ksh 中,就输入:

```
export LC_ALL = "POSIX"
```

如果在 csh/tcsh 中,就输入:

```
setenv LC_ALL = "POSIX"
```

接下来请按以下步骤进行操作。

(1) 首先,创建 ex0504_01_sort 文件,内容如下。

```
bear
^ant
cat
 ant
 (注意此行为空行)
ant
America
Above all
```

```
 * ant
25
1
39
```

| Oct | Dec | Hex | Char | Oct | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|
| 000 | 0 | 00 | NUL '\0' (null character) | 100 | 64 | 40 | @ |
| 001 | 1 | 01 | SOH (start of heading) | 101 | 65 | 41 | A |
| 002 | 2 | 02 | STX (start of text) | 102 | 66 | 42 | B |
| 003 | 3 | 03 | ETX (end of text) | 103 | 67 | 43 | C |
| 004 | 4 | 04 | EOT (end of transmission) | 104 | 68 | 44 | D |
| 005 | 5 | 05 | ENQ (enquiry) | 105 | 69 | 45 | E |
| 006 | 6 | 06 | ACK (acknowledge) | 106 | 70 | 46 | F |
| 007 | 7 | 07 | BEL '\a' (bell) | 107 | 71 | 47 | G |
| 010 | 8 | 08 | BS '\b' (backspace) | 110 | 72 | 48 | H |
| 011 | 9 | 09 | HT '\t' (horizontal tab) | 111 | 73 | 49 | I |
| 012 | 10 | 0A | LF '\n' (new line) | 112 | 74 | 4A | J |
| 013 | 11 | 0B | VT '\v' (vertical tab) | 113 | 75 | 4B | K |
| 014 | 12 | 0C | FF '\f' (form feed) | 114 | 76 | 4C | L |
| 015 | 13 | 0D | CR '\r' (carriage ret) | 115 | 77 | 4D | M |
| 016 | 14 | 0E | SO (shift out) | 116 | 78 | 4E | N |
| 017 | 15 | 0F | SI (shift in) | 117 | 79 | 4F | O |
| 020 | 16 | 10 | DLE (data link escape) | 120 | 80 | 50 | P |
| 021 | 17 | 11 | DC1 (device control 1) | 121 | 81 | 51 | Q |
| 022 | 18 | 12 | DC2 (device control 2) | 122 | 82 | 52 | R |
| 023 | 19 | 13 | DC3 (device control 3) | 123 | 83 | 53 | S |
| 024 | 20 | 14 | DC4 (device control 4) | 124 | 84 | 54 | T |
| 025 | 21 | 15 | NAK (negative ack.) | 125 | 85 | 55 | U |
| 026 | 22 | 16 | SYN (synchronous idle) | 126 | 86 | 56 | V |
| 027 | 23 | 17 | ETB (end of trans. blk) | 127 | 87 | 57 | W |
| 030 | 24 | 18 | CAN (cancel) | 130 | 88 | 58 | X |
| 031 | 25 | 19 | EM (end of medium) | 131 | 89 | 59 | Y |
| 032 | 26 | 1A | SUB (substitute) | 132 | 90 | 5A | Z |
| 033 | 27 | 1B | ESC (escape) | 133 | 91 | 5B | [ |
| 034 | 28 | 1C | FS (file separator) | 134 | 92 | 5C | \ '\\' |
| 035 | 29 | 1D | GS (group separator) | 135 | 93 | 5D | ] |
| 036 | 30 | 1E | RS (record separator) | 136 | 94 | 5E | ^ |
| 037 | 31 | 1F | US (unit separator) | 137 | 95 | 5F | _ |
| 040 | 32 | 20 | SPACE | 140 | 96 | 60 | ` |
| 041 | 33 | 21 | ! | 141 | 97 | 61 | a |
| 042 | 34 | 22 | " | 142 | 98 | 62 | b |
| 043 | 35 | 23 | # | 143 | 99 | 63 | c |
| 044 | 36 | 24 | $ | 144 | 100 | 64 | d |
| 045 | 37 | 25 | % | 145 | 101 | 65 | e |
| 046 | 38 | 26 | & | 146 | 102 | 66 | f |
| 047 | 39 | 27 | ' | 147 | 103 | 67 | g |
| 050 | 40 | 28 | ( | 150 | 104 | 68 | h |
| 051 | 41 | 29 | ) | 151 | 105 | 69 | i |
| 052 | 42 | 2A | * | 152 | 106 | 6A | j |
| 053 | 43 | 2B | + | 153 | 107 | 6B | k |
| 054 | 44 | 2C | , | 154 | 108 | 6C | l |
| 055 | 45 | 2D | - | 155 | 109 | 6D | m |
| 056 | 46 | 2E | . | 156 | 110 | 6E | n |
| 057 | 47 | 2F | / | 157 | 111 | 6F | o |
| 060 | 48 | 30 | 0 | 160 | 112 | 70 | p |
| 061 | 49 | 31 | 1 | 161 | 113 | 71 | q |
| 062 | 50 | 32 | 2 | 162 | 114 | 72 | r |
| 063 | 51 | 33 | 3 | 163 | 115 | 73 | s |
| 064 | 52 | 34 | 4 | 164 | 116 | 74 | t |
| 065 | 53 | 35 | 5 | 165 | 117 | 75 | u |
| 066 | 54 | 36 | 6 | 166 | 118 | 76 | v |
| 067 | 55 | 37 | 7 | 167 | 119 | 77 | w |
| 070 | 56 | 38 | 8 | 170 | 120 | 78 | x |
| 071 | 57 | 39 | 9 | 171 | 121 | 79 | y |
| 072 | 58 | 3A | : | 172 | 122 | 7A | z |
| 073 | 59 | 3B | ; | 173 | 123 | 7B | { |
| 074 | 60 | 3C | < | 174 | 124 | 7C | \| |
| 075 | 61 | 3D | = | 175 | 125 | 7D | } |
| 076 | 62 | 3E | > | 176 | 126 | 7E | ~ |
| 077 | 63 | 3F | ? | 177 | 127 | 7F | DEL |

图 5-24　ASCII 码表

```
joy@ubuntu:~/chapter05$ sort ex0504_01_sort
 ant
*ant
1
25
39
Above all
America
^ant
ant
bear
cat
```

图 5-25　sort 的排序结果

（2）然后，执行 sort 命令来对文件内容进行排序。

```
sort ex0504_01_sort
```

结果如图 5-25 所示，可以看到文件中每行内容都按 ASCII 码的值升序排列。sort 首先比较每行第 1 个字符的 ASCII 码值，并进行排序；如果出现值相同的行，则接着比较那些行第 2 个字符的 ASCII 码值。以此类推得到最终的排序结果。

### 5.4.1　对文件内容按字典顺序排序

sort 中的-d 选项可以使 sort 忽略标点符号等特殊符号，而只对字母、数字和空格进行排序。请执行以下命令实现按字典顺序排序。

```
sort -d ex0504_01_sort
```

结果如图 5-26 所示，sort 忽略了特殊符号“*”和“^”，按字典顺序排序。

```
joy@ubuntu:~/chapter05$ sort -d ex0504_01_sort
 ant
1
25
39
Above all
America
*ant
^ant
ant
bear
cat
```

图 5-26　按字典顺序的排序结果

### 5.4.2　对文件内容不区分字母大小写排序

sort 中的-f 选项可以使 sort 在排序时忽略字母的大小写。请执行以下命令。

```
sort -f ex0504_01_sort
```

结果如图 5-27 所示，sort 忽略了字母的大小写，并进行排序。

### 5.4.3　对文件内容反向排序

sort 中的-r 选项可以使 sort 对文件内容反向排序(降序排序)。请执行以下命令。

```
sort -r ex0504_01_sort
```

结果如图 5-28 所示。

```
joy@ubuntu:~/chapter05$ sort -f ex0504_01_sort
 ant
*ant
1
25
39
Above all
America
ant
bear
cat
^ant
```

图 5-27　不区分字母大小写的排序结果

```
joy@ubuntu:~/chapter05$ sort -r ex0504_01_sort
cat
bear
ant
^ant
America
Above all
39
25
1
*ant
 ant
```

图 5-28　反向排序结果

### 5.4.4　对文件内容按数值大小排序

sort 中的-n 选项可以使 sort 在对文件中的数值进行排序时，按数值的大小排序，而不是按数值中数字的 ASCII 码值排序。请执行以下命令。

(1) 首先，创建 ex0504_02_sort 文件，内容如下。

```
9
56
2
100
7.4
43
```

(2) 然后，执行不加任何选项的 sort 命令。

```
sort ex0504_02_sort
```

结果如图 5-29 所示，文件内容按 ASCII 码值升序排列。

（3）最后，执行带-n 选项的 sort 命令。

```
sort -n ex0504_02_sort
```

结果如图 5-30 所示，文件内容按数值的大小升序排列。


图 5-29　不加任何选项的排序结果

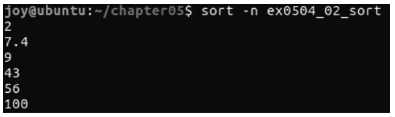


图 5-30　带-n 选项的排序结果

**注意**：ex0504_02_sort 文件中仅含有数字，因此排序结果如图 5-30 所示；若创建一个既含有数字又含有字符的文件，那么执行带-n 选项的 sort 命令后，数字将按数值大小排序，字符将按 ASCII 码值排序。

## 5.4.5　对文件内容按某一字段排序

sort 在对文件内容排序时，默认情况下是从每行的第 1 个字符开始比较。若第 1 个字符相同，则继续比较第 2 个字符、第 3 个字符……，以此类推得出最终排序结果。但实际上可以指定排序时的待比较字段，接下来将介绍根据某一字段进行的排序方法。

（1）首先，创建 ex0504_03_sort 文件，内容如下。

```
Amy mathematics June potato
Mike chemistry August tomato
Tom biology September carrot
John physics October bean
Linda physics May onion
```

可以看到以上内容第 1 列为人名，第 2 列为课程名称，第 3 列为月份，第 4 列为蔬菜名称，以上内容每行相邻字段均使用一个空格隔开。

（2）然后，执行以下命令。

```
sort -k 1 ex0504_03_sort
```

结果如图 5-31 所示，-k 选项用来指定待比较的字段，此条命令指示 sort 根据人名进行排序。

（3）如果指定字段中出现重复内容，那么 sort 便根据重复字段的下一字段继续排序。请执行以下命令。

```
sort -k 2 ex0504_03_sort
```

结果如图 5-32 所示，可以注意到第 2 列字段中出现重复的“physics”，sort 便根据第 3 列字段中的“May”和“October”继续排序。

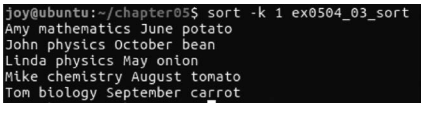


图 5-31　根据人名进行排序

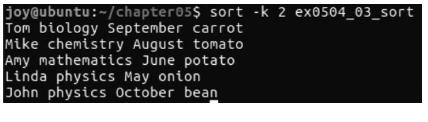


图 5-32　根据第 2 字段进行排序

## 5.4.6　对文件内容限定排序

在 5.4.5 节中，已经学习了如何使用 sort 指定某一字段进行排序，本节将介绍限定字

段的排序方法。

(1) 首先,执行以下命令。

```
sort +1 -3 ex0504_03_sort
```

结果如图 5-33 所示,其中,“+1”是指从第 1 列分隔符开始,“-3”是指到第 3 列分隔符结束,也就是将待比较内容限定在第 2、3 列字段。可以看到,图中包含“physics”的第 4、5 行是按第 3 列字段来进行排序的。

(2) 然后,执行以下命令。

```
sort +1 -2 +3 -4 ex0504_03_sort
```

结果如图 5-34 所示,其中,“+1 -2”限定了第 1 组待比较内容(通常称为主关键字段)为第 2 列字段,“+3 -4” 限定了第 2 组待比较内容(通常称为次关键字段)为第 4 列字段。可以看到,图中出现重复字段“physics”的第 4、5 行是按第 4 列字段来进行排序的。

```
joy@ubuntu:~/chapter05$ sort +1 -3 ex0504_03_sort
Tom biology September carrot
Mike chemistry August tomato
Amy mathematics June potato
Linda physics May onion
John physics October bean
```

图 5-33　限定在第 2、3 列字段的排序

```
joy@ubuntu:~/chapter05$ sort +1 -2 +3 -4 ex0504_03_sort
Tom biology September carrot
Mike chemistry August tomato
Amy mathematics June potato
John physics October bean
Linda physics May onion
```

图 5-34　带次关键字段的排序

(3) 最后,执行以下命令。

```
sort +1 -2 +3r -4 ex0504_03_sort
```

结果如图 5-35 所示,可以注意到“+3”后紧接着代表按降序排序的“r”,这条命令的意思是将主关键字段按默认的升序排序,次关键字段则按降序排序。因此,可以看到图中出现重复字段的第 4、5 行是按降序排序的。

```
joy@ubuntu:~/chapter05$ sort +1 -2 +3r -4 ex0504_03_sort
Tom biology September carrot
Mike chemistry August tomato
Amy mathematics June potato
Linda physics May onion
John physics October bean
```

图 5-35　次关键字段的降序排序

## 5.4.7　在不同字段分隔符下使用 sort

在 5.4.6 节中,使用 sort 排序的默认字段分隔符是空格,本节将学习在不同字段分隔符下使用 sort。

(1) 首先,创建 ex0504_04_sort 文件,内容如下。

```
Arm Eye:Leg:Foot
Leg Arm:Foot:Eye
Eye Foot:Arm:Leg
```

**注意**:第 1、2 列字符串之间为空格,其余列字符串之间为冒号。

(2) 然后,执行以下命令。

```
sort -k 2 ex0504_04_sort
```

结果如图 5-36 所示,由于默认分隔符是空格,因此在未指定其他分隔符的情况下,sort 根据第 2 列字符串(空格后)进行排序。

(3) 最后,执行以下命令。

```
sort -k 2 -t: ex0504_04_sort
```

结果如图 5-37 所示，-t 选项后紧接着的是用户指定的分隔符冒号。因此，可以看到 sort 按第 3 列字符串（第 1 列冒号后）进行排序。

```
joy@ubuntu:~/chapter05$ sort -k 2 ex0504_04_sort
Leg Arm:Foot:Eye
Arm Eye:Leg:Foot
Eye Foot:Arm:Leg
```

图 5-36　使用默认分隔符的排序

```
joy@ubuntu:~/chapter05$ sort -k 2 -t: ex0504_04_sort
Eye Foot:Arm:Leg
Leg Arm:Foot:Eye
Arm Eye:Leg:Foot
```

图 5-37　用户指定分隔符的排序

## 5.4.8　对文件排序后重写

到目前为止，sort 的排序结果都被输出到屏幕上，方便我们查看，但实际上我们还可以将 sort 的排序结果写入指定文件。若指定文件为待排序文件，则能实现对指定文件的重写。

（1）首先，将 ex0504_03_sort 文件的排序结果写入 ex0504_05_sort 新文件，并使用 more 查看新文件的内容。请执行以下命令。

```
sort ex0504_03_sort > ex0504_05_sort
more ex0504_05_sort
```

```
joy@ubuntu:~/chapter05$ sort ex0504_03_sort > ex0504_05_sort
joy@ubuntu:~/chapter05$ more ex0504_05_sort
Amy mathematics June potato
John physics October bean
Linda physics May onion
Mike chemistry August tomato
Tom biology September carrot
```

图 5-38　将排序结果写入指定文件

如图 5-38 所示，“>”为重定向符，指示 sort 将排序结果写入指定的文件 ex0504_05_sort。

（2）然后，将 ex0504_03_sort 文件内容复制进 ex0504_06_sort 新文件中，使用 more 确认新文件后，对新文件进行排序后重写，并再次使用 more 查看新文件。请依次执行以下命令。

```
cp ex0504_03_sort > ex0504_06_sort
more ex0504_06_sort
sort ex0504_06_sort > ex0504_06_sort
more ex0504_06_sort
```

如图 5-39 所示，新文件经过排序并重写后，其内容变为空；这是因为当 Shell 被指示将信息写入一个已存在的文件时，Shell 会事先将此文件清空，所以 ex0504_06_sort 文件的内容变为空。

（3）要想成功将被排序文件重写，只需要在 sort 命令中添加-o 选项。请执行以下命令。

```
cp ex0504_03_sort > ex0504_07_sort
more ex0504_07_sort
sort  -o ex0504_07_sort ex0504_07_sort
more ex0504_07_sort
```

图 5-40 显示了上述命令执行的全过程。读者可以注意到，ex0504_07_sort 文件排序后被成功重写。

```
joy@ubuntu:~/chapter05$ cp ex0504_03_sort ex0504_06_sort
joy@ubuntu:~/chapter05$ more ex0504_06_sort
Amy mathematics June potato
Mike chemistry August tomato
Tom biology September carrot
John physics October bean
Linda physics May onion
joy@ubuntu:~/chapter05$ sort ex0504_06_sort > ex0504_06_sort
joy@ubuntu:~/chapter05$ more ex0504_06_sort
joy@ubuntu:~/chapter05$
```

图 5-39　ex0504_06_sort 文件的内容变为空

```
joy@ubuntu:~/chapter05$ cp ex0504_03_sort ex0504_07_sort
joy@ubuntu:~/chapter05$ more ex0504_07_sort
Amy mathematics June potato
Mike chemistry August tomato
Tom biology September carrot
John physics October bean
Linda physics May onion
joy@ubuntu:~/chapter05$ sort -o ex0504_07_sort ex0504_07_sort
joy@ubuntu:~/chapter05$ more ex0504_07_sort
Amy mathematics June potato
John physics October bean
Linda physics May onion
Mike chemistry August tomato
Tom biology September carrot
```

图 5-40　对 ex0504_07_sort 文件进行排序后重写

# 5.5　文件内容比较

在处理文件时，有时需要对文件内容进行比较，或者删除内容中的重复行，本节将介绍如何使用 3 种不同实用程序对文件内容进行比较。

## 5.5.1　识别和删除重复行

实用程序 uniq 可以识别和删除文件内容中相邻的重复行，具体介绍如下。

### 1. 统计行的相邻重复次数

(1) 首先，创建 ex0505_01_uniq 文件，内容如下。

```
apple
cherry
banana
banana
banana
date
lemon
cherry
cherry
```

以上内容共有 9 行，其中 apple 为第 1 行，cherry 为第 2 行，banana 为第 3 行，……，以此类推。各行之间的关系可分为 4 种(相邻且重复，相邻但不重复，不相邻但重复，不相邻且不重复)，如表 5-2 所示。

表 5-2　行之间的关系

| 关系 | 相邻且重复 | 相邻但不重复 | 不相邻但重复 | 不相邻且不重复 |
|---|---|---|---|---|
| 示例数据 | 第 3 行的 banana 和第 4 行的 banana | 第 6 行的 date 和第 7 行的 lemon | 第 2 行的 cherry 和第 8 行的 cherry | 第 1 行的 apple 和第 6 行的 date |

(2) 然后，执行以下命令。

```
uniq -c ex0505_01_uniq
```

结果如图 5-41 所示，-c 选项可以将相邻且重复行的重复次数显示在行首，可以发现“banana”相邻且重复了 3 次，“cherry”相邻且重复了 2 次。

```
will0322@ubuntu:~/chapter05$ vi ex0505_01_uniq
will0322@ubuntu:~/chapter05$ uniq -c ex0505_01_uniq
      1 apple
      1 cherry
      3 banana
      1 date
      1 lemon
      2 cherry
```

图 5-41　统计行的相邻且重复次数

### 2. 查看相邻且重复行的其中一行

使用-d 选项可以查看文件中相邻且重复行的其中一行，请执行以下命令。

```
uniq -d ex0505_01_uniq
```

结果如图 5-42 所示。

### 3. 查看不相邻但重复的行

使用-u 选项可以查看文件中不相邻但重复的行。请执行以下命令。

```
uniq -u ex0505_01_uniq
```

结果如图 5-43 所示，可以看到不相邻但重复的行被输出了。

```
joy@ubuntu:~/chapter05$ uniq -d ex0505_01_uniq
banana
cherry
```

图 5-42 查看相邻且重复行的其中一行

```
will0322@ubuntu:~/chapter05$ uniq -u ex0505_01_uniq
apple
cherry
date
lemon
```

图 5-43 查看不相邻但重复的行

**注意**：如何理解不相邻且不重复？从上述结果来看，不相邻且不重复更偏向于不相邻。

### 4. 识别和删除所有的重复行

(1) 首先，执行以下命令。

```
uniq ex0505_01_uniq
```

结果如图 5-44 所示，可以看到 uniq 删除了所有相邻且重复行的重复部分，但是未识别出不相邻的重复行“cherry”。

(2) 然后，执行以下命令。

```
sort ex0505_01_uniq | uniq
```

结果如图 5-45 所示，文件中的所有重复行都被删除了。此条命令先使用 sort 对文件内容排序，经过排序后所有的重复行都彼此相邻，接着再使用 uniq 删除所有的重复行。

```
will0322@ubuntu:~/chapter05$ uniq ex0505_01_uniq
apple
cherry
banana
date
lemon
cherry
```

图 5-44 使用 uniq 后的结果

```
will0322@ubuntu:~/chapter05$ sort ex0505_01_uniq | uniq
apple
banana
cherry
date
lemon
```

图 5-45 识别和删除所有的重复行

## 5.5.2 按行比较两个文件

实用程序 comm 可以按行比较两个有序文件的异同，具体介绍如下。

(1) 首先，创建两个文件。

ex0505_01_comm 文件内容如下。

```
Venus
Earth
Mars
Saturn
```

ex0505_02_comm 文件内容如下。

```
Saturn
Uranus
Mars
Neptune
```

(2) 然后，执行以下命令使用 comm 按行比较两个文件的异同。

```
comm ex0505_01_comm ex0505_02_comm
```

结果如图 5-46 所示。可以看到，不仅输出结果混乱，还提示这两个待比较文件是无序的。这说明我们在使用 comm 之前，一定要保证待比较文件是有序的。

(3) 接下来，对两个文件排序重写，并执行 comm 命令。

```
sort -o ex0505_01_comm ex0505_01_comm
sort -o ex0505_02_comm ex0505_02_comm
comm ex0505_01_comm ex0505_02_comm
```

如图 5-47 所示，上述命令执行完毕后，可以看到出现了 3 列内容。

```
joy@ubuntu:~/chapter05$ comm ex0505_01_comm ex0505_02_comm
        Saturn
        Uranus
comm: file 2 is not in sorted order
        Mars
        Neptune
Venus
comm: file 1 is not in sorted order
Earth
Mars
Saturn
```

图 5-46　无序文件的比较结果

```
joy@ubuntu:~/chapter05$ sort -o ex0505_01_comm ex0505_01_comm
joy@ubuntu:~/chapter05$ sort -o ex0505_02_comm ex0505_02_comm
joy@ubuntu:~/chapter05$ comm ex0505_01_comm ex0505_02_comm
Earth
                Mars
        Neptune
                Saturn
        Uranus
Venus
```

图 5-47　有序文件的比较结果

上述 3 列内容所代表的含义如表 5-3 所示。

**表 5-3　执行 comm 的结果解释**

| 列号 | 含　义 | 内　容 |
|---|---|---|
| 1 | 只存在于 ex0505_01_comm 文件中的行 | Earth<br>Venus |
| 2 | 只存在于 ex0505_02_comm 文件中的行 | Neptune<br>Uranus |
| 3 | 两个文件中都存在的行 | Mars<br>Saturn |

(4) 最后，请执行以下命令。

```
comm -3 ex0505_01_comm ex0505_02_comm
```

结果如图 5-48 所示，“-3”表示不输出第 3 列(“两个文件中都存在的行”那一列内容)，可以看到 comm 只输出了第 1 和第 2 列。

```
joy@ubuntu:~/chapter05$ comm -3 ex0505_01_comm ex0505_02_comm
Earth
        Neptune
        Uranus
Venus
```

图 5-48　只输出第 1 和第 2 列的结果

## 5.5.3　查看文件不同之处

实用程序 diff 也可以用来比较两个文件的不同之处，但与 comm 不同的是，它的输出结果会告诉用户如何将前一文件内容转换为后一文件内容。

下面给出一个例子。

(1) 首先，创建两个文件。

ex0505_01_diff 文件内容如下。

```
pull
push
hit
lip
kiss
kick
```

ex0505_02_diff 文件内容如下。

```
abuse
push
lip
kiss
eat
beat
kick
```

(2) 然后，执行以下命令使用 cat 查看两个文件的内容，并输出行号。

```
cat -n ex0505_01_diff
cat -n ex0505_02_diff
```

结果分别如图 5-49 和图 5-50 所示。

```
joy@ubuntu:~/chapter05$ cat -n ex0505_01_diff
     1  pull
     2  push
     3  hit
     4  lip
     5  kiss
     6  kick
```

图 5-49　查看 ex0505_01_diff 文件内容

```
joy@ubuntu:~/chapter05$ cat -n ex0505_02_diff
     1  abuse
     2  push
     3  lip
     4  kiss
     5  eat
     6  beat
     7  kick
```

图 5-50　查看 ex0505_02_diff 文件内容

(3) 最后，使用 diff 对两个文件内容进行比较，请执行以下命令。

```
diff ex0505_01_diff ex0505_02_diff
```

结果如图 5-51 所示，可以发现 3 个两侧带数字的特殊字母“c”“d”“a”，它们分别代表修改、删除、添加这 3 个操作，并且字母左侧的数字是前一文件(ex0505_01_diff)的行号，字母右侧的数字是后一文件(ex0505_02_diff)的行号，行号之间用逗号隔开。

```
joy@ubuntu:~/chapter05$ diff ex0505_01_diff ex0505_02_diff
1c1
< pull
---
> abuse
3d2
< hit
5a5,6
> eat
> beat
```

图 5-51　使用 diff 比较两个文件内容

下面进一步解释 diff 的比较结果，如表 5-4 所示。

**表 5-4　进一步解释 diff 的比较结果**

| 内容 | 解　　释 |
| --- | --- |
| 1c1<br>< pull<br>---<br>> abuse | 前一文件中的行用“<”开头，后一文件中的行用“>”开头，以下两行内容同理；<br>将前一文件的第 1 行“pull”改为后一文件的第 1 行“abuse”；<br>“---”为分隔符 |

续表

| 内容 | 解　　释 |
| --- | --- |
| 3d2<br>< hit | 若将前一文件的第 3 行“hit”删除，则前一文件和后一文件的第 2 行便相同了 |
| 5a5，6<br>> eat<br>> beat | 前一文件的第 5 行“kiss”之后添加后一文件的第 5 行“eat”和第 6 行“beat” |

## 5.6　文件内容替换

实用程序 tr 可以实现对文件内容的替换，本节将介绍 tr 的几种不同用法。

### 5.6.1　替换指定字符

tr 可以替换文件内容中的指定字符，接下来给出一个例子。

(1) 首先，创建 ex0506_01_tr 文件，内容如下。

```
epoch
grape
pear
a-e
```

(2) 然后，使用 tr 将文件中的“r”全部替换成“R”，请执行以下命令。

```
tr 'r''R'< ex0506_01_tr
```

结果如图 5-52 所示，“< ex0506_01_tr”的意思是将 ex0506_01_tr 文件的内容作为 tr 的输入，以此来实现对文件内容的操作。可以发现，文件中所有“r”都被替换成了“R”。

(3) 最后，执行以下命令。

```
tr '\n'"< ex0506_01_tr
```

结果如图 5-53 所示，可以看到文件中的所有换行符(“\n”)都被替换成了空格。

```
joy@ubuntu:~/chapter05$ tr 'r' 'R' < ex0506_01_tr
epoch
gRape
peaR
a-e
```

图 5-52　替换某个字母

```
joy@ubuntu:~/chapter05$ tr '\n' ' ' < ex0506_01_tr
epoch grape pear a-e joy@ubuntu:~/chapter05$
```

图 5-53　替换某个字符

(4) 此外，tr 还可以替换多个字符。请执行以下命令。

```
tr 'pe-''PE,'< ex0506_01_tr
```

结果如图 5-54 所示，可以看到，文件中的所有“p”都被替换成了“P”，所有“e”都被替换成了“E”，“—”被替换成了“,”。

```
joy@ubuntu:~/chapter05$ tr 'pe-' 'PE,' < ex0506_01_tr
EPoch
graPE
PEar
a,E
```

图 5-54　替换多个字符

**注意**：待替换字符串“pe-”并不是一个整体，tr 会将它们分成 3 个不同字符来分别处理，这样一来它们的顺序也就不重要了，因此文件中的“ep”被替换成了“EP”。

## 5.6.2 按范围替换

5.6.1 节已经介绍了如何使用 tr 按指定字符进行替换，本节将学习如何按范围进行替换。

（1）首先，执行以下命令。

```
tr '[a-g]''[A-G]'< ex0506_01_tr
```

结果如图 5-55 所示，此命令的意思是将所有小写字母 a 至 g，都替换成对应的大写字母 A 至 G。

（2）字母不仅可以替换成字母，还可以替换成数字，接下来请执行以下命令。

```
tr '[a-g]''[1-7]'< ex0506_01_tr
```

结果如图 5-56 所示，可以发现“a”被替换成了“1”，“c”被替换成了“3”，“e”被替换成了“5”，“g”被替换成了“7”。

```
joy@ubuntu:~/chapter05$ tr '[a-g]' '[A-G]' < ex0506_01_tr
EpoCh
GrApE
pEAr
A-E
```

图 5-55 将小写字母替换成大写字母

```
joy@ubuntu:~/chapter05$ tr '[a-g]' '[1-7]' < ex0506_01_tr
5po3h
7r1p5
p51r
1-5
```

图 5-56 将字母替换成数字

## 5.6.3 删除指定字符

tr 不仅可以实现指定字符的替换，还可以删除指定字符，具体介绍如下。

（1）创建 ex0506_02_tr 文件，内容如下。

```
Use       commas to add clarity and emphasis.

Do not use       double parentheses in text.
Put other ideas into your own     words.
```

（2）通过-d 选项可以删除所有指定字符，请执行以下命令。

```
tr -d 'o'< ex0506_02_tr
```

结果如图 5-57 所示，可以看到文件中的字母“o”都被删除了。

```
joy@ubuntu:~/chapter05$ tr -d 'o' < ex0506_02_tr
Use      cmmas t add clarity and emphasis.

D nt use       duble parentheses in text.
Put ther ideas int yur wn         wrds.
```

图 5-57 删除字母

（3）还可以删除文件中所有的换行符（“\n”），请执行以下命令。

```
tr -d '\n'< ex0506_02_tr
```

结果如图 5-58 所示，可以发现文件中所有的换行符都被删除了。

```
joy@ubuntu:~/chapter05$ tr -d '\n' < ex0506_02_tr
Use      commas to add clarity and emphasis.Do not use       double parentheses in text
.Put other ideas into your own         words.joy@ubuntu:~/chapter05$
```

图 5-58 删除换行符

（4）添加-s 选项后，可以检测出文件中的连续重复字符，并将第 1 个连续重复字符之后的所有连续重复字符删除，接下来以删除文件中连续重复空格为例，演示 tr 的此功能。请

执行以下命令。

```
tr -s ''< ex0506_02_tr
```

结果如图 5-59 所示。

```
joy@ubuntu:~/chapter05$ tr -s ' ' < ex0506_02_tr
Use commas to add clarity and emphasis.

Do not use double parentheses in text.
Put other ideas into your own words.
```

图 5-59　删除空格

### 5.6.4　结合管道替换

tr 还可以结合管道来使用。“|”为管道符，其左侧命令的输出会作为右侧命令的输入，接下来给出几个例子。

(1) 结合管道，对文件每行进行连续编号，并输出其大写字母形式，请执以下命令。

```
cat -n ex0506_02_tr | tr '[a-z]' '[A-Z]'
```

结果如图 5-60 所示。

(2) 有时需要得到一个有序的文件，结合管道便可以对文件进行排序并删除重复空格，请执行以下命令。

```
sort ex0506_02_tr | tr -s ''
```

结果如图 5-61 所示，此命令先将 ex0506_tr 文件排序，再将排序结果作为 tr 的输入，经过处理后再输出。

```
joy@ubuntu:~/chapter05$ cat -n ex0506_02_tr | tr '[a-z]' '[A-Z]'
     1  USE      COMMAS TO ADD CLARITY AND EMPHASIS.
     2
     3  DO NOT USE       DOUBLE PARENTHESES IN TEXT.
     4  PUT OTHER IDEAS INTO YOUR OWN         WORDS.
```

图 5-60　结合管道使用 cat 和 tr

```
joy@ubuntu:~/chapter05$ sort ex0506_02_tr | tr -s ' '

Do not use double parentheses in text.
Put other ideas into your own words.
Use commas to add clarity and emphasis.
```

图 5-61　结合管道使用 sort 和 tr

(3) 还可以利用 tr 对文字内容进行过滤，请执行以下命令。

```
echo s4ync hr on i5za9ti0on | tr -d '[0-9]'
```

结果如图 5-62 所示，过滤了全部数字和空格后，得到了清晰的文字内容。

```
joy@ubuntu:~/chapter05$ echo s4ync  hr on  i5za9ti0on | tr -d '[0-9]'
synchronization
```

图 5-62　结合 echo 使用 tr

## 5.7　单行编辑数据

实用程序 sed 可以实现对文件的按行编辑，本节将介绍 sed 的几种不同用法。

### 5.7.1　修改指定单词

sed 可以修改指定的单词，接下来给出一个例子。

(1) 首先，创建 ex0507_sed 文件，内容如下。

```
court court legal
court legal legal
legal court court
reply legal
```

然后，执行以下命令。

```
sed 's/court/fee/' ex0507_sed
```

结果如图 5-63 所示，其中斜杠之前的“s”表示替换每行的第 1 个指定单词。可以看到，此命令将每行的第 1 个“court”替换成“fee”，而第 1、3 行中的第 2 个“court”都未被替换。

（2）接着，执行以下命令。

```
sed 's/court/fee/g' ex0507_sed
```

结果如图 5-64 所示，此命令中的“g”表示替换文件中所有的指定单词，因此可以看到文件中所有的“court”都被替换成了“fee”。

```
joy@ubuntu:~/chapter05$ sed 's/court/fee/' ex0507_sed
fee court legal
fee legal legal
legal fee court
reply legal
```

图 5-63　替换每行的第 1 个指定单词

```
joy@ubuntu:~/chapter05$ sed 's/court/fee/g' ex0507_sed
fee fee legal
fee legal legal
legal fee fee
reply legal
```

图 5-64　替换文件中所有的指定单词

（3）最后，执行以下命令。

```
sed '/court/s/legal/fee/g' ex0507_sed
```

结果如图 5-65 所示，此命令将所有包含“court”的行中的“legal”全部替换成“fee”，即包含“court”的前 3 行中的“legal”全部都被替换成了“fee”，而未包含“court”的第 4 行，其中的“legal”未被替换。

```
joy@ubuntu:~/chapter05$ sed '/court/s/legal/fee/g' ex0507_sed
court court fee
court fee fee
fee court court
reply legal
```

图 5-65　指定行替换

## 5.7.2　删除指定行

除了替换，sed 还可以删除文件中指定的行，下面给出一个例子。

（1）首先，执行以下命令。

```
sed '/court/d' ex0507_sed
```

结果如图 5-66 所示，“d”表示删除，此命令将所有包含“court”的行全部删除，即包含“court”的前 3 行都被删除了，而未包含“court”的第 4 行则被保留。

（2）然后，执行以下命令。

```
sed '4d' ex0507_sed
```

结果如图 5-67 所示，其中“4d”表示删除第 4 行，可以发现文件的前 3 行被输出了，而第 4 行被删除。

```
joy@ubuntu:~/chapter05$ sed '/court/d' ex0507_sed
reply legal
```

图 5-66　按指定单词删除行

```
joy@ubuntu:~/chapter05$ sed '4d' ex0507_sed
court court legal
court legal legal
legal court court
```

图 5-67　按行号删除行

## 5.7.3　结合正则表达式修改

跟 grep 一样，sed 也可以结合正则表达式来使用，接下来给出几个例子。

（1）我们可以指定待替换的行首字符，请执行以下命令。

```
sed '/^c/s/legal/fee/g' ex0507_sed
```

结果如图 5-68 所示，可以发现所有以“c”开头的行中的“legal”都被替换成了“fee”。

```
joy@ubuntu:~/chapter05$ sed '/^c/s/legal/fee/g' ex0507_sed
court court fee
court fee fee
legal court court
reply legal
```

图 5-68　按行开头字符修改

（2）还可以指定待替换的行尾字符，请执行以下命令。

```
sed '/l $ /d' ex0507_sed
```

（注意：“$”前是小写字母“l”。）

结果如图 5-69 所示，可以看到以“l”结尾的行都被删除了。

（3）不仅如此，正则表达式甚至可以实现对指定字符串的替换，请执行以下命令。

```
sed 's/legal/& high/g' ex0507_sed
```

结果如图 5-70 所示，其中的“&”指代了被替换的单词，可以发现所有的单词“legal”都被替换成了“legal high”。

```
joy@ubuntu:~/chapter05$ sed '/l$/d' ex0507_sed
legal court court
```

图 5-69　按行末尾字符修改

```
joy@ubuntu:~/chapter05$ sed 's/legal/& high/g' ex0507_sed
court court legal high
court legal high legal high
legal high court court
reply legal high
```

图 5-70　按指定单词修改

## 5.8　数据操作工具

实用程序 awk 主要用来对数据进行操作。它可对文件内容进行各种检索（这一点与之前介绍的 grep 类似），还可以直接使用预定义变量，甚至可以结合正则表达式和逻辑判断语句来使用，必要时还可与之前介绍的 bc 一样，进行简单的数学运算。接下来，本节将具体介绍实用程序 awk。

### 5.8.1　数据操作工具介绍

在使用 Ubuntu 时，经常需要对文件中的内容进行各种处理。例如：使用 grep 在某一文件中查找指定单词；使用 tr 将文件中的换行符都替换成空格；使用 sed 将文件中的某一单词替换成另一单词。上述操作在本质上都是针对数据所做的处理。

数据是对客观世界的一种抽象，是千百年来人类智慧的结晶。我们在前述章节中输入的命令、操作的文件均可被视为数据的一种。随着人们需要记载的信息越来越多，尤其是随着信息技术与人们日常生活的深度融合，计算机系统中的数据呈爆发式增长，以至于对数据的操作和管理变得极为复杂。因此，合理存储数据并对其进行全面管理和高效利用，对人们的日常生活具有十分重要的意义。

最初我们对数据的管理采用纯手工的方式，使用这一方式对数据进行管理，最大的弊端是人工成本高。某些工作理论上可行，实际上由于复杂度太高而难于实现。假定一个大学在校学生有 4 万人，如果一张纸上可以记录 40 条学生的基本信息，则需要 1000 张纸才可以

记录所有学生的基本信息。若该校名字为“张三”的学生共有 10 位，他们的信息分别随机出现在第 900 至第 1000 页，我们事先不可能知道这些信息，此时使用纯手工方式来统计这一信息将极为困难，并且容易出现错误。若需要统计全校所有姓名相同的学生则情况更为复杂，且手工统计结果的可信度不高。

为了解决类似问题，可先将上述数据整理成文件，然后由计算机进行存储和处理。仍以统计全校姓名相同的学生为例，如果我们先将学生基本信息组织成文件，然后使用实用程序 sort 去处理这一文件，那么立即可得到准确可信的结果。和上述纯手工管理数据的方式相比，这种将数据整理成文件并由计算机直接处理的方法可极大地提高工作效率。不过在实际使用时仍存在以下问题：不同文件之间的关系不够明确，导致文件中数据冗余度较高，一旦数据量很大，将会造成存储空间的极大浪费；文件内部的数据格式无法统一，因此在处理文件内容时会存在无法对齐的问题。

越来越多的程序员发现仅用文件来处理数据存在诸多弊端。20 世纪 70 年代，E. F. Code 便提出了关系数据库理论，试图解决这一问题。关系数据库中的所有数据都被排列成一张二维表。首行显示某类事物的不同属性名称，如学生基本信息表中的学号、姓名、性别、年龄等；之后的每一行便是相应的记录，它表示某个事物的具体属性，如某一学生的学号为“201526703014”，姓名为“李华”，性别为“男”，年龄为“20”岁等。

我们在学习过程中使用的文本文件通常也被认为是数据库的一种，其内容与关系数据库中的二维表相似。我们可以使用 grep、tr、sed、awk 等实用程序对其进行操作，但与 awk 相比，grep、tr、sed 的数据处理能力较为单一，而 awk 则具有极为强大的数据处理能力。它可以选择行并输出指定字段、修改字段分隔符、操作数据库和选择记录、使用预定义变量和正则表达式，以及在输出结果中插入空格、进行数学运算。接下来将逐一介绍 awk 的上述用法。

### 5.8.2 选择行并输出字段

awk 可以对文件内容实现按行检索，若检索成功则会输出相应的结果。这些结果为若干字段所对应的数据，它们来源于目标字符串所在的行。接下来给出一个例子。

（1）首先，在当前目录下将 ls 的结果保存到 ex0508_01_awk 新文件中，并使用 cat 查看新文件的内容。请执行下列命令以完成上述操作。

```
ls > ex0508_01_awk
cat ex0508_01_awk
```

结果如图 5-71 所示。

（2）然后，使用 tr 将 ex0508_01_awk 文件中的下画线都替换成空格，再将替换结果保存到 ex0508_02_awk 新文件中，并使用 cat 查看新文件内容。请执行下列命令以完成上述操作。

```
tr '_' ' ' < ex0508_01_awk > ex0508_02_awk
cat ex0508_02_awk
```

结果如图 5-72 所示。

（3）最后，请执行以下命令。

```
awk '/ex0505/ {print $3}' ex0508_02_awk
```

结果如图 5-73 所示。

```
joy@ubuntu:~/chapter05$ ls > ex0508_01_awk
joy@ubuntu:~/chapter05$ cat ex0508_01_awk
ex0501_column
ex0502_01_grep
ex0502_02_grep
ex0502_03_grep
ex0502_04_grep
ex0502_05_grep
ex0504_01_sort
ex0504_02_sort
ex0504_03_sort
ex0504_04_sort
ex0504_05_sort
ex0504_06_sort
ex0504_07_sort
ex0505_01_comm
ex0505_01_diff
ex0505_01_uniq
ex0505_02_comm
ex0505_02_diff
ex0505_02_uniq
ex0505_03_uniq
ex0506_01_tr
ex0506_02_tr
ex0507_sed
ex0508_01_awk
```

图 5-71　将 ls 结果保存到 ex0508_01_awk 文件中

```
joy@ubuntu:~/chapter05$ tr '_' ' ' < ex0508_01_awk > ex0508_02_awk
joy@ubuntu:~/chapter05$ cat ex0508_02_awk
ex0501 column
ex0502 01 grep
ex0502 02 grep
ex0502 03 grep
ex0502 04 grep
ex0502 05 grep
ex0504 01 sort
ex0504 02 sort
ex0504 03 sort
ex0504 04 sort
ex0504 05 sort
ex0504 06 sort
ex0504 07 sort
ex0505 01 comm
ex0505 01 diff
ex0505 01 uniq
ex0505 02 comm
ex0505 02 diff
ex0505 02 uniq
ex0505 03 uniq
ex0506 01 tr
ex0506 02 tr
ex0507 sed
ex0508 01 awk
```

图 5-72　将替换结果保存到 ex0508_02_awk 文件中

```
joy@ubuntu:~/chapter05$ awk '/ex0505/ {print $3}' ex0508_02_awk
comm
diff
uniq
comm
diff
uniq
uniq
```

图 5-73　输出符合检索条件的行的某一字段

awk 默认的字段分隔符是空格，它将 ex0508_02_awk 文件的内容分成了 3 列，每列对应一个字段。上述命令的参数释义如表 5-5 所示。

**表 5-5　命令的参数释义**

| 参数 | /ex0505/ | {print $3} | ex0508_02_awk |
|---|---|---|---|
| 释义 | 双斜杠中间的“ex0505”是待检索的字符串 | “print $3”指明了对符合检索条件的行所要执行的操作，其中“print”表示输出，“$3”表示第 3 个字段。这一参数表示输出目标文件中的第 3 个字段 | 待检索的文件 |

可以看到符合检索条件的行的第 3 个字段都被输出了。

(4) 此外，还可以指定输出多个字段，请执行以下命令。

```
awk '/ex05/ {print $2, $3}' ex0508_02_awk
```

结果如图 5-74 所示，花括号中的多个变量($2 和 $3)使用逗号隔开，可以看到输出了符合检索条件的行的第 2 个字段和第 3 个字段。

## 5.8.3　指定字段分隔符

使用 awk 命令时默认的字段分隔符是空格，在 5.8.2 节中使用的字段分隔符是空格，可以使用-F 选项指定某一符号为分隔符。接下来请执行以下命令。

```
awk -F_ '/ex/ {print $3, $1}' ex0508_01_awk
```

结果如图 5-75 所示，在上述命令中使用-F 指定下画线为字段分隔符，并在参数中设置

```
joy@ubuntu:~/chapter05$ awk '/ex05/ {print $2,$3}' ex0508_02_awk
column
01 grep
02 grep
03 grep
04 grep
05 grep
01 sort
02 sort
03 sort
04 sort
05 sort
06 sort
07 sort
01 comm
01 diff
01 uniq
02 comm
02 diff
02 uniq
03 uniq
01 tr
02 tr
sed
01 awk
```

图 5-74　输出符合检索条件的行的某些字段

了先输出第 3 个字段，再输出第 1 个字段，字段间使用空格分隔。

```
joy@ubuntu:~/chapter05$ awk -F_ '/ex/ {print $3,$1}' ex0508_01_awk
 ex0501
grep ex0502
grep ex0502
grep ex0502
grep ex0502
grep ex0502
sort ex0504
sort ex0504
sort ex0504
sort ex0504
sort ex0504
sort ex0504
sort ex0504
comm ex0505
diff ex0505
uniq ex0505
comm ex0505
diff ex0505
uniq ex0505
uniq ex0505
tr ex0506
tr ex0506
 ex0507
awk ex0508
```

图 5-75　分隔符为下画线

**注意**：花括号中的变量 $3 和变量 $1 以逗号分开，这使得输出结果中的所有字段均以空格分开。

此外，还可以使用 awk 删除指定的字符。接下来请执行以下命令。

```
awk -F0 '/ex/ {print $1 $2 $3 $4}' ex0508_01_awk
```

结果如图 5-76 所示，-F 指定字段分隔符为数字 0，使用这种方法成功删除了数字 0。

```
joy@ubuntu:~/chapter05$ awk -F0 '/ex/ {print $1 $2 $3 $4}' ex0508_01_awk
ex51_column
ex52_1_grep
ex52_2_grep
ex52_3_grep
ex52_4_grep
ex52_5_grep
ex54_1_sort
ex54_2_sort
ex54_3_sort
ex54_4_sort
ex54_5_sort
ex54_6_sort
ex54_7_sort
ex55_1_comm
ex55_1_diff
ex55_1_uniq
ex55_2_comm
ex55_2_diff
ex55_2_uniq
ex55_3_uniq
ex56_1_tr
ex56_2_tr
ex57_sed
ex58_1_awk
```

图 5-76　分隔符为数字 0

**注意**：与之前的 awk 命令不同，花括号中的所有变量均以空格分开，这使得输出结果中的所有字段均无分隔符。

### 5.8.4　awk 命令语法

我们在前几节中已经介绍了如何使用 awk 来操作文件，接下来学习 awk 命令的语法。此命令的语法如下。

```
awk 选项 '模式 {操作}'文件名
```

每个参数的释义如表 5-6 所示。

**表 5-6　awk 命令的参数的释义**

| 参数 | 选　　项 | 模　　式 | 操　　作 | 文件名 |
|---|---|---|---|---|
| 释义 | 能使 awk 完成某种特定功能的参数，比如用来指定字段分隔符的-F 选项 | 待检索的字符串，通常是正则表达式或者关系表达式 | 对检索成功的行执行相应的操作，默认的操作是按指定格式输出检索成功行中的若干字段 | 待检索文件的名称 |

在使用 awk 时，表 5-6 中所示的“模式”和“操作”参数是可选的，即可仅使用“模式”参数或仅使用“操作”参数。接下来给出其具体用法。

**1. 仅使用“操作”参数**

请执行以下命令。

```
awk '{print}' ex0508_02_awk
```

在仅使用“操作”参数时，使用 awk 命令对 ex0508_02_awk 文件执行的结果如图 5-77 所示。

**注意**：由于“{print}”中未指定输出的字段，awk 便输出了检索成功行的所有字段。

**2. 仅使用“模式”参数**

请执行以下命令。

```
awk '/sort/' ex0508_02_awk
```

在仅使用“模式”参数时，使用 awk 命令对 ex0508_02_awk 文件执行的结果如图 5-78 所示。

```
joy@ubuntu:~/chapter05$ awk '{print}' ex0508_02_awk
ex0501 column
ex0502 01 grep
ex0502 02 grep
ex0502 03 grep
ex0502 04 grep
ex0502 05 grep
ex0504 01 sort
ex0504 02 sort
ex0504 03 sort
ex0504 04 sort
ex0504 05 sort
ex0504 06 sort
ex0504 07 sort
ex0505 01 comm
ex0505 01 diff
ex0505 01 uniq
ex0505 02 comm
ex0505 02 diff
ex0505 02 uniq
ex0505 03 uniq
ex0506 01 tr
ex0506 02 tr
ex0507 sed
ex0508 01 awk
```

图 5-77　仅使用“操作”参数

```
joy@ubuntu:~/chapter05$ awk '/sort/' ex0508_02_awk
ex0504 01 sort
ex0504 02 sort
ex0504 03 sort
ex0504 04 sort
ex0504 05 sort
ex0504 06 sort
ex0504 07 sort
```

图 5-78　仅使用“模式”参数

### 5.8.5 使用 awk 操作数据库

在之前的各节中，使用 awk 时，其操作对象都是文件。从本节开始，将学习如何使用 awk 操作数据库。

（1）使用 vi 创建一个数据库 ex0508_03_awk，内容如下。

```
201501 Amy f 19 89.3
201502 Mike m 20 70.6
201503 John m 23 64.9
201504 Linda f 18 82.7
201505 Cindy f 20 0.0
201506 Frank m 21 75.0
201507 Gina f 20 95.1
201508 Herry m 19 96.0
201509 Nike m 20 85.0
201510 Kathy f 22 80.5
```

ex0508_03_awk 中共有 5 列数据，每列之间均使用一个空格隔开，与关系数据库中的二维表相似。我们可以将其看成某学校的学生基本信息表，给每列数据赋予一个属性名称，第 1 列是学号，第 2 列是姓名，第 3 列是性别，第 4 列是年龄，第 5 列是平均成绩。每一行数据都是一条记录，存储了某位学生的具体属性，比如，由第 2 行数据可知，姓名为“Mike”的学生，其学号为“201502”，性别为“男”，年龄为“20”岁，平均成绩为“70.6”分。

请执行以下命令使用 awk 操作数据库 ex0508_03_awk。

```
awk '/2015/ {print $1, $2, $4}' ex0508_03_awk
```

结果如图 5-79 所示。

（2）为了增强可读性，首先输出每一列的属性名称，然后再输出每一列的数据。接下来请执行下列命令。

```
awk '{print "Num Name Sex Age Score";exit}' ex0508_03_awk; awk '{print}' ex0508_03_awk
```

结果如图 5-80 所示，先在首行输出了每一列的属性名称，第 1 列是“Num”，第 2 列是“Name”，第 3 列是“Sex”，第 4 列是“Age”，第 5 列是“Score”，然后输出了每一列的数据。

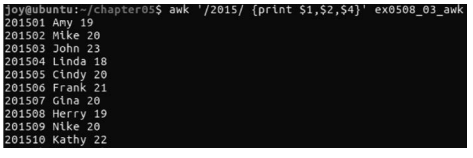


图 5-79 使用 awk 操作数据库

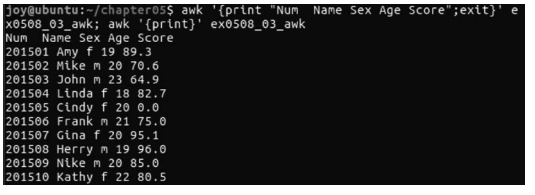


图 5-80 在输出结果中加入字符串

### 5.8.6 选择输出数据库的字段

使用 awk 时，在对数据库的字段进行输出时，可以根据需求选择输出数据库中的某一个字段，或是某一些字段，甚至全部字段。

**1. 输出数据库中的某个字段**

请执行以下命令输出数据库中的第 1 个字段。

```
awk '{print $1}' ex0508_03_awk
```

结果如图 5-81 所示。

**2. 输出数据库中的某些字段**

请执行以下命令输出数据库中的第 1 个、第 2 个和第 3 个字段。

```
awk '{print $1, $2, $3}' ex0508_03_awk
```

结果如图 5-82 所示。

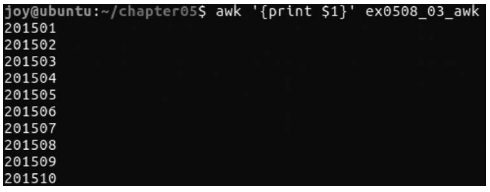


图 5-81　输出数据库中的某个字段

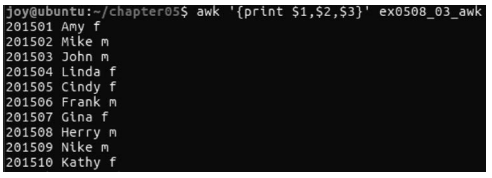


图 5-82　输出数据库中的某些字段

**3. 输出数据库中的全部字段**

请执行以下命令输出数据库中的全部字段。

```
awk '{print $0}' ex0508_03_awk
```

结果如图 5-83 所示，参数“$0”代表某一行的全部字段，可以看到输出了数据库中的全部字段。

执行以下命令也可达到上述同样的效果。

```
awk '{print}' ex0508_03_awk
```

结果如图 5-84 所示。

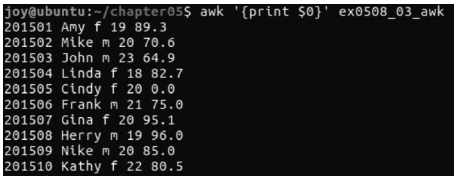


图 5-83　输出数据库中的全部字段

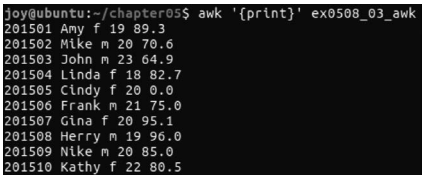


图 5-84　输出数据库中的全部字段

从上述被执行命令可看出“{print $0}”和“{print}”的输出结果是一致的。

## 5.8.7　使用 awk 的预定义变量

awk 中有一些事先定义的变量，它们被称为预定义变量，本节将介绍 3 个常用预定义变量的用法。

使用前，先来了解这些常用预定义变量的释义，如表 5-7 所示。

**表 5-7　预定义变量的释义**

| 预定义变量 | $n(某个或某些字段) | NF(Number of Fields in the Current Record，当前记录的字段总个数) | NR(Current Record Number in the Total Input Stream，当前记录的编号) |
|---|---|---|---|

续表

| | | | |
|---|---|---|---|
| 释义 | 当 $n>0$ 时，$n 表示待检索文件中的第 $n$ 个字段。<br>当 $n=0$ 时，$0 表示待检索文件中的所有字段 | 在 awk 检索文件时，表示当前被检索行的字段个数 | 在 awk 检索文件时，表示当前被检索行的行号 |

由于在之前的各节中已经介绍过预定义变量“$n”的用法，接下来就只介绍预定义变量“NF”和“NR”的用法。

**1. 当前记录的字段总个数(NF)**

(1) 可以使用“NF”得到文件中每行的字段个数。请执行以下命令完成上述操作。

```
awk '{print NF}' ex0508_03_awk
```

结果如图 5-85 所示，在 ex0508_03_awk 文件中，每一行都有 5 个字段。

(2) 还可以在“NF”前面加上“$”来输出每一行的最后一个字段。请执行以下命令。

```
awk '{print $NF, $5}' ex0508_03_awk
```

结果如图 5-86 所示，因为在 ex0508_03_awk 文件中，每一行都只有 5 个字段，所以“$NF”和“$5”相同，因此预定义变量“$NF”和“$5”的输出结果是一致的。

```
joy@ubuntu:~/chapter05$ awk '{print NF}' ex0508_03_awk
5
5
5
5
5
5
5
5
5
5
```

图 5-85　使用“NF”输出字段个数

```
joy@ubuntu:~/chapter05$ awk '{print $NF,$5}' ex0508_03_awk
89.3 89.3
70.6 70.6
64.9 64.9
82.7 82.7
0.0 0.0
75.0 75.0
95.1 95.1
96.0 96.0
85.0 85.0
80.5 80.5
```

图 5-86　使用“$NF”输出字段

(3) 通过“NF”可以进行某些数学运算。请执行以下命令。

```
awk '{print $(NF-1)}' ex0508_03_awk
```

如图 5-87 所示，因为此处“NF”先减 1 再进行取值运算，所以结果输出了每行的倒数第 2 个字段。

**注意**：若想让“NF”先进行数学运算再进行取值运算，必须在数学运算式的两侧使用小括号，以此来划分“NF”的数学运算范围。

**2. 当前记录的编号(NR)**

可以使用“NR”来输出文件中每一行的行号。请执行以下命令。

```
awk '{print NR, $0}' ex0508_03_awk
```

结果如图 5-88 所示。

```
joy@ubuntu:~/chapter05$ awk '{print $(NF-1)}' ex0508_03_awk
19
20
23
18
20
21
20
19
20
22
```

图 5-87　使用“$(NF-1)”输出字段

```
joy@ubuntu:~/chapter05$ awk '{print NR,$0}' ex0508_03_awk
1 201501 Amy f 19 89.3
2 201502 Mike m 20 70.6
3 201503 John m 23 64.9
4 201504 Linda f 18 82.7
5 201505 Cindy f 20 0.0
6 201506 Frank m 21 75.0
7 201507 Gina f 20 95.1
8 201508 Herry m 19 96.0
9 201509 Nike m 20 85.0
10 201510 Kathy f 22 80.5
```

图 5-88　使用“NR”输出行号

## 5.8.8　使用自定义变量、字符串和数字

在使用 awk 时，正确使用自定义变量、字符串和数字都是非常有必要的，本节将学习如何使用它们。

### 1. 使用自定义变量

(1) 需要先通过使用-v 选项设置一个自定义变量，然后才可以在进行操作时使用它。请执行以下命令完成上述操作。

```
awk -v test='student' '{print test}' ex0508_03_awk
```

结果如图 5-89 所示，接在-v 后面的“test”为一个自定义变量，将其赋值为“student”后，接着使用“{print test}”输出它的值。可以看到在检索文件 ex0508_03_awk 的每一行时都成功输出了“student”。

```
joy@ubuntu:~/chapter05$ awk -v test='student' '{print test}' ex0508_03_awk
student
student
student
student
student
student
student
student
student
student
```

图 5-89　使用自定义变量

(2) 在 awk 中还可以同时使用自定义变量和预定义变量，以下命令展示了这一用法。

```
awk -v test='student:' '{print test, $0}' ex0508_03_awk
```

如图 5-90 所示，先输出自定义变量“test”的值“student:”，接着输出预定义变量“$0”的值。

```
joy@ubuntu:~/chapter05$ awk -v test='student:' '{print test,$0}' ex0508_03_awk
student: 201501 Amy f 19 89.3
student: 201502 Mike m 20 70.6
student: 201503 John m 23 64.9
student: 201504 Linda f 18 82.7
student: 201505 Cindy f 20 0.0
student: 201506 Frank m 21 75.0
student: 201507 Gina f 20 95.1
student: 201508 Herry m 19 96.0
student: 201509 Nike m 20 85.0
student: 201510 Kathy f 22 80.5
```

图 5-90　自定义变量和预定义变量结合使用

(3) 若我们未使用-v 选项设置自定义变量“test”而直接使用它，输出时这一变量将会被忽略，无任何输出，以下命令的执行结果清楚地展示了这一点。

```
awk '{print test, $0}' ex0508_03_awk
```

结果如图 5-91 所示，仅输出了“$0”所指示的结果。

```
joy@ubuntu:~/chapter05$ awk '{print test,$0}' ex0508_03_awk
 201501 Amy f 19 89.3
 201502 Mike m 20 70.6
 201503 John m 23 64.9
 201504 Linda f 18 82.7
 201505 Cindy f 20 0.0
 201506 Frank m 21 75.0
 201507 Gina f 20 95.1
 201508 Herry m 19 96.0
 201509 Nike m 20 85.0
 201510 Kathy f 22 80.5
```

图 5-91　使用未定义的变量

### 2. 使用字符串

在使用 awk 时，可以通过使用双引号输出若干字符串。请执行以下命令完成上述

操作。

```
awk '{print "Num:" $1,"Name:" $2}' ex0508_03_awk
```

结果如图 5-92 所示，双引号中的字符串被原样输出。

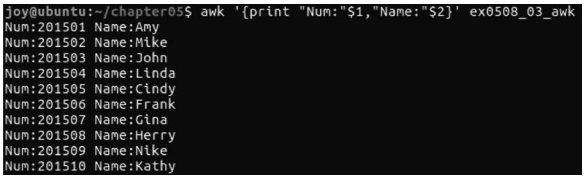


图 5-92　使用字符串

**3. 使用数字**

（1）在 awk 中，除了可以使用自定义变量和字符串，还可以使用数字。请执行以下命令。

```
awk '{print 78.9}' ex0508_03_awk
```

结果如图 5-93 所示。

（2）在 awk 中可执行以下命令进行数学运算。

```
awk '{print 78.9 - 1}' ex0508_03_awk
```

结果如图 5-94 所示。

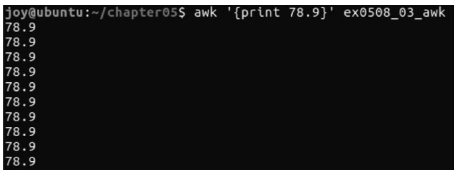


图 5-93　使用数字

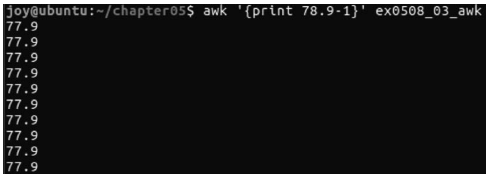


图 5-94　进行数学运算

（3）在 awk 中，数字的数学运算结果与普通字符串是不同的。

```
awk '{print "78.9 - 1",78.9 - 1}' ex0508_03_awk
```

结果如图 5-95 所示，两侧使用双引号的为普通字符串，可以发现 awk 对普通字符串原样输出，而对数字则是先进行数学运算再输出。


图 5-95　区别普通字符串和数字

## 5.8.9　使用正则表达式

正则表达式不仅可以在 grep 命令中使用，还可以在 awk 命令中使用。之前在学习 grep 命令时简要介绍了如何使用正则表达式，本小节将结合 awk 命令，进一步介绍正则表

达式的其他用法。

(1) 首先,创建数据库 ex0508_04_awk,内容如下。

```
bus  1
bike
Bus  2
plane  604
subway  5
ship  200
train  150
Train  100
taxi  10
```

(2) 然后,执行以下命令在数据库中查找并包含字母“T”或“t”的行。

```
awk '/[Tt]/ {print}' ex0508_04_awk
```

如图 5-96 所示,结果输出了数据库中“T”或“t”所在的行。

```
joy@ubuntu:~/chapter05$ awk '/[Tt]/ {print}' ex0508_04_awk
train   150
Train   100
taxi    10
```

图 5-96　查找包含字母“T”或“t”的行

**注意**:在 awk 中字母是区分大小写的。

(3) 再执行以下命令。

```
awk '/[p-z]/ {print}' ex0508_04_awk
```

上述命令中的“[p-z]”表示按字母表顺序逐行查找从“p”到“z”内的所有字母,因此只要某一行包含了“p”到“z”中的任意一个字符,该行就会被输出。如图 5-97 所示,输出结果的所有行中都包含了“p”到“z”的任意一个字符,而“bike”中的“b”“i”“k”“e”均不在“p”到“z”的范围内,因此只含“bike”的行未被输出。

(4) 接着执行以下命令。

```
awk '/^[a-s]/ {print}' ex0508_04_awk
```

“[ ]”外面的“^”指示了行的起始,“[a-s]”表示按字母表顺序逐行查找从“a”到“s”的所有字母,因此“^[a-s]”表示查找所有起始字符为“a”到“s”的任意一个字母的行。结果如图 5-98 所示,输出了起始字符为“b”“p”和“s”的行,而未输出起始字符为“B”“T”“t”的行。

```
joy@ubuntu:~/chapter05$ awk '/[p-z]/ {print}' ex0508_04_awk
bus     1
Bus     2
plane   604
subway  5
ship    200
train   150
Train   100
taxi    10
```

图 5-97　查找符合条件的结果

```
joy@ubuntu:~/chapter05$ awk '/^[a-s]/ {print}' ex0508_04_awk
bus     1
bike
plane   604
subway  5
ship    200
```

图 5-98　查找“^[a-s]”的结果

(5) 再执行以下命令。

```
awk '/[^a-z]/ {print}' ex0508_04_awk
```

与上一条命令不同的是,此处的“^”在“[ ]”里面,“[^a-z]”表示按字母表顺序逐行查找从“a”到“z”的所有字母,只要某一行包含了“a”到“z”以外的任意一个字符,这一行就会被输

出。结果如图 5-99 所示，只包含“bike”的那一行未被输出，而既包含字母又包含数字的剩余所有行都被输出了。

（6）最后，请执行以下命令。

```
awk '/[1-4]/ {print}' ex0508_04_awk
```

与之前的“[p-z]”类似，“[1-4]”表示逐行查找“1”到“4”的任意一个字符，只要某一行包含了从“1”到“4”的任意一个字符，这一行就会被输出。如图 5-100 所示，可以看到结果中的每一行均包含“1”到“4”的任意一个字符。

```
joy@ubuntu:~/chapter05$ awk '/[^a-z]/ {print}' ex0508_04_awk
bus     1
Bus     2
plane   604
subway  5
ship    200
train   150
Train   100
taxi    10
```

图 5-99　查找“[^a-z]”的结果

```
joy@ubuntu:~/chapter05$ awk '/[1-4]/ {print}' ex0508_04_awk
bus     1
Bus     2
plane   604
ship    200
train   150
Train   100
taxi    10
```

图 5-100　查找“[1-4]”的结果

## 5.8.10　使用指定的字段选择记录

在之前的小节中，是通过查找每一个字段来完成记录的选择，而在本节中将学习查找指定字段并使用一些逻辑运算符来选择记录。

首先学习如何通过查找指定字段来选择记录。

（1）首先，执行以下命令。

```
awk '$3 == "f" {print}' ex0508_03_awk
```

此条命令中的“$3==“f””是用来查找数据库中的第 3 个字段的值等于“f”的记录。结果如图 5-101 所示，可以看到输出了第 3 个字段的值等于“f”的所有记录。

**注意**：在比较字符串时，一定要在待比较的内容两侧加双引号。

（2）然后，执行以下命令。

```
awk '$4 > 19 {print}' ex0508_03_awk
```

此条命令中“$4>19”是用来查找数据库中的第 4 个字段的值大于 19 的记录。结果如图 5-102 所示，可以看到所有输出行的第 4 个字段值均大于 19。

```
joy@ubuntu:~/chapter05$ awk '$3=="f" {print}' ex0508_03_awk
201501 Amy f 19 89.3
201504 Linda f 18 82.7
201505 Cindy f 20 0.0
201507 Gina f 20 95.1
201510 Kathy f 22 80.5
```

图 5-101　输出第 3 个字段等于“f”的行

```
joy@ubuntu:~/chapter05$ awk '$4>19 {print}' ex0508_03_awk
201502 Mike m 20 70.6
201503 John m 23 64.9
201505 Cindy f 20 0.0
201506 Frank m 21 75.0
201507 Gina f 20 95.1
201509 Nike m 20 85.0
201510 Kathy f 22 80.5
```

图 5-102　输出第 4 个字段的值大于 19 的行

（3）接着，执行以下命令。

```
awk '$1~/[Bb]us/ {print}' ex0508_04_awk
```

此条命令中的“$1~/[Bb]us/”是用来查找数据库中的第 1 个字段匹配“/[Bb]us/”的记录。如图 5-103 所示，“Bus”和“bus”均符合检索条件。

```
joy@ubuntu:~/chapter05$ awk '$1~/[Bb]us/ {print}' ex0508_04_awk
bus     1
Bus     2
```

图 5-103　输出符合正则表示式“/[Bb]us/”的行

(4) 最后,执行以下命令。

```
awk '$1!~/[Tt]rain/ {print}' ex0508_04_awk
```

此条命令中的“$1!~/[Tt]rain/”是用来查找数据库中的第 1 个字段不匹配“/[Tt]rain/”的记录。结果如图 5-104 所示,输出的行均为不匹配的记录。

```
joy@ubuntu:~/chapter05$ awk '$1!~/[Tt]rain/ {print}' ex0508_04_awk
bus     1
bike
Bus     2
plane   604
subway  5
ship    200
taxi    10
```

图 5-104　输出不符合正则表示式“/[Tt]rain/”的行

接下来将学习使用与、或、非这 3 个逻辑运算符。

① 运算符与

运算符与“&&”用来连接多个不同的查找条件,仅当满足所有条件时整个表达式的值为真。

请执行以下命令。

```
awk '$3=="m" && $4>20 {print}' ex0508_03_awk
```

结果如图 5-105 所示,输出的行均满足第 3 个字段为“m”且第 4 个字段的值都大于 20。

```
joy@ubuntu:~/chapter05$ awk '$3=="m" && $4>20 {print}' ex0508_03_awk
201503 John m 23 64.9
201506 Frank m 21 75.0
```

图 5-105　使用与运算符

② 运算符或

运算符或“||”用来连接多个不同的查找条件,只要满足其中一个条件,整个表达式的值就为真。

请执行以下命令。

```
awk '/[Bb]us/ || $2>=150 {print}' ex0508_04_awk
```

结果如图 5-106 所示,前 2 行都满足了条件“/[Bb]us/”,后 3 行都满足了条件“$2>=150”。

```
joy@ubuntu:~/chapter05$ awk '/[Bb]us/ || $2>=150 {print}' ex0508_04_awk
bus     1
Bus     2
plane   604
ship    200
train   150
```

图 5-106　使用或运算符

③ 运算符非

运算符非“!”表示否定,通常放在查找条件的前面使用,使用时需要在条件的两侧加上小括号。

请执行以下命令。

```
awk '!(NF==1) {print}' ex0508_04_awk
```

“!(NF==1)”表示查找所有字段总数不等于 1 的行。结果如图 5-107 所示,可以看到所有输出行的字段总数都为 2,都不等于 1。

```
joy@ubuntu:~/chapter05$ awk '!(NF==1) {print}' ex0508_04_awk
bus     1
Bus     2
plane   604
subway  5
ship    200
train   150
Train   100
taxi    10
```

图 5-107　使用非运算符

### 5.8.11 使用 awk 命令文件

在之前的学习中，都是通过在命令行中输入命令来使用 awk 的，当需要使用更加复杂的 awk 语句时，临时输入命令既费时费力又容易出错，若把那些复杂的 awk 语句放在一个单独的 awk 命令文件中，使用 awk 命令直接调用这些文件内容，便可使操作更加简单。本节将介绍如何使用 awk 命令文件。

（1）首先，使用 vi 创建一个 awk 命令文件 ex0508_05_awk，内容如下。

```
/Amy/ || $4>=20 {print $2,$4}
```

（2）然后，执行以下命令来调用 ex0508_05_awk 文件的内容。

```
awk -f ex0508_05_awk ex0508_03_awk
```

这条命令使用-f 选项来调用 ex0508_05_awk 文件的内容，并对 ex0508_03_awk 文件执行 ex0508_05_awk 文件中的操作（即查找包含“Amy”或第 4 个字段大于等于 20 的记录，并输出这些记录的第 2 个字段和第 4 个字段）。该命令的执行结果如图 5-108 所示。

使用直接输入命令的方式也可以达到上述效果，请执行以下命令。

```
awk '/Amy/ || $4>=20 {print $2,$4}' ex0508_03_awk
```

结果如图 5-109 所示。

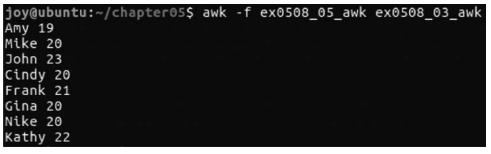


图 5-108 使用 awk 命令文件

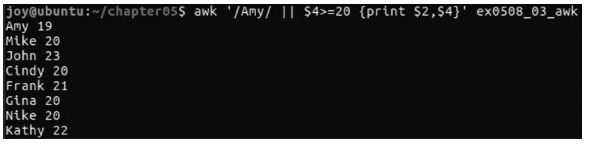


图 5-109 直接输入命令

可以看到上述 2 条命令的执行结果是相同的，但使用 awk 命令文件的方式可以实现复用，后续将会详细介绍。

在使用 awk 时，可以指定查找和输出时使用的分隔符，接下来将学习 4 个用来指定分隔符的预定义变量，它们的释义如表 5-8 所示。

**表 5-8 预定义变量的释义**

| 预定义变量 | FS（Splits Records Into Fields as a Regular Expression，查找时使用的字段分隔符） | RS（Input Record Separator，查找时使用的记录分隔符） | OFS（Inserted between Fields on Output，输出时使用的字段分隔符） | ORS（Terminates Each Record on Output，输出时使用的记录分隔符） |
|---|---|---|---|---|
| 释义 | 在 awk 对文件进行检索时使用的字段分隔符，默认为空格或制表符（键盘的 Tab 键） | 在 awk 对文件进行检索时使用的记录分隔符，默认为换行符 | 在 awk 输出检索结果时使用的字段分隔符，默认为空格 | 位于每一条记录的末尾，是在 awk 输出检索结果时使用的记录分隔符，默认为换行符 |

接下来将介绍上述预定义变量的用法。

（1）首先，创建 awk 命令文件 ex0508_06_awk，内容如下。

```
BEGIN{
```

```
FS = "_"
OFS = "～"
}
{
print $1, $2
}
```

其中的“FS＝"_""指定了检索时的字段分隔符为“_”,“OFS＝"～""指定了输出时的字段分隔符为“～”。

(2) 然后,执行以下命令。

```
awk -f ex0508_06_awk ex0508_01_awk
```

结果如图 5-110 所示,awk 在检索时使用“_”作为字段分隔符,输出时使用“～”将第 1 个字段和第 2 个字段分隔开。

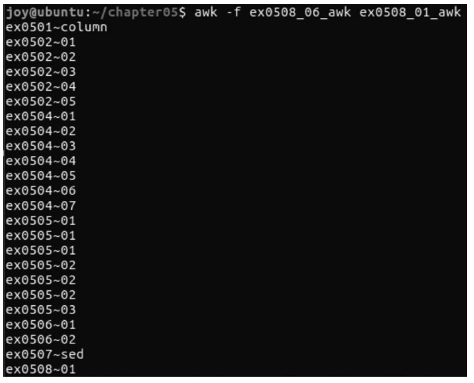


图 5-110　使用预定义变量“FS”和“OFS”

(3) 接下来创建 3 个文件。

ex0508_07_awk 文件内容为:

```
2017 - 01 - 10 * 2017 - 02 - 10 * 2017 - 03 - 10 *
2017 - 04 - 10
```

(4) 将 ex0508_06_awk 文件复制并重命名为 ex0508_08_awk 文件,再使用 vi 编辑器对其进行修改,修改后的内容如下所示。

```
BEGIN{
FS = " - "
RS = " * "
}
{
print $1, $2
}
```

ex0508_08_awk 文件中的“FS＝"-""指定了检索时的字段分隔符为“-”,“RS＝" * ""指定了检索时的记录分隔符为“ * ”。

(5) 将 ex0508_06_awk 文件复制并重命名为 ex0508_09_awk 文件,再使用 vi 编辑器对其进行修改,修改后的内容如下所示。

```
BEGIN{
FS = " - "
RS = " * "
ORS = "…… "
}
{
print $1, $2
}
```

与 ex0508_08_awk 文件不同的是,ex0508_09_awk 文件中新增的“ORS＝"……""指定了输出时的记录分隔符为“……”。

(6) 再执行以下命令。

```
awk -f ex0508_08_awk ex0508_07_awk
```

如前所述，此时 awk 在检索文件时不再以默认的换行符作为记录分隔符，而是以“*”作为记录分隔符，又因为“-”为检索时的字段分隔符，且默认输出时的记录分隔符为换行符。因此结果如图 5-111 所示，输出了每条记录的第 1 个字段和第 2 个字段，相邻记录之间使用换行符分开。

（7）接着，执行以下命令。

```
awk -f ex0508_09_awk ex0508_07_awk
```

与上一条命令不同的是，ex0508_09_awk 文件中新增的“ORS="……"”指定了输出时的记录分隔符为“……”。因此结果如图 5-112 所示，输出的相邻记录之间使用省略号分开。

```
joy@ubuntu:~/chapter05$ awk -f ex0508_08_awk ex0508_07_awk
2017 01
2017 02
2017 03
2017 04
```

图 5-111　使用预定义变量“FS”和“RS”

```
joy@ubuntu:~/chapter05$ awk -f ex0508_09_awk ex0508_07_awk
2017 01...2017 02...2017 03...2017 04...joy@ubuntu:~/chapter05$
```

图 5-112　使用预定义变量“FS”“RS”和“ORS”

## 5.8.12　awk 命令的拓展

在 5.8.11 节中，学习了如何使用 awk 命令文件，本小节将学习一些 awk 命令的拓展用法。接下来给出一些示例。

（1）首先，创建 awk 命令文件 ex0508_10_awk，内容如下。

```
/Mike/ || /Amy/ {
print $1
print $2, $4
}
```

注意以下两点。

① “{”必须跟在待检索内容的末尾，不能另起一行，比如此文件中的“{”必须跟在“/Mike/ || /Amy/”的末尾。

② 在“{ }”中，必须一条语句单独占一行，比如此文件中的两条输出语句“print $1”和“print $2, $4”之间必须换行。

（2）然后，依次执行下列命令。

```
awk '/Mike/ || /Amy/ {print $1"\n"$2, $4}' ex0508_03_awk
awk -f ex0508_10_awk ex0508_03_awk
```

这两条命令的执行结果如图 5-113 所示。虽然这两条命令的输出结果是相同的，但是第 1 条命令比第 2 条更加简洁，且可读性更强。

```
joy@ubuntu:~/chapter05$ awk '/Mike/ || /Amy/ {print $1"\n"$2,$4}' ex0508_03_awk
201501
Amy 19
201502
Mike 20
joy@ubuntu:~/chapter05$ awk -f ex0508_10_awk ex0508_03_awk
201501
Amy 19
201502
Mike 20
```

图 5-113　两条命令的执行结果

（3）接着，创建 awk 命令文件 ex0508_11_awk，内容如下。

```
/Mike/ || /Amy/ {
name = $2
```

```
age = $4
print name,age
}
```

与之前使用的比较符号“==”不同,“=”为赋值符号,它表示将其右侧变量的值赋给其左侧的变量,因此“name=$2”表示将其右侧变量“$2”的值赋给其左侧变量“name”,“age=$4”表示将其右侧变量“$4”的值赋给其左侧变量“age”。“print name,age”表示依次打印自定义变量“name”和“age”的值。

(4) 然后,执行以下命令。

```
awk -f ex0508_11_awk ex0508_03_awk
```

结果如图 5-114 所示。与直接使用“$2”和“$4”不同的是,此处先将上述预定义变量的值赋给具有特定含义的自定义变量,当我们想打印姓名时只需输入“name”,而不用去寻找姓名是在第几个字段。

```
joy@ubuntu:~/chapter05$ awk -f ex0508_11_awk ex0508_03_awk
Amy 19
Mike 20
```

图 5-114　使用自定义变量

(5) 最后,请执行以下命令。

```
awk '{print NR,"t\Num:"$1,"t\Name:"$2,"t\Score:"$5}' ex0508_03_awk
```

结果如图 5-115 所示。在 awk 中加入“Num:”“NR”等标注语句可以提高输出结果的可读性,制表符“\t”可以让输出结果排列整齐。

```
joy@ubuntu:~/chapter05$ awk '{print NR,"\tNum:"$1,"\tScore:"$5}' ex0508_03_awk
1       Num:201501      Score:89.3
2       Num:201502      Score:70.6
3       Num:201503      Score:64.9
4       Num:201504      Score:82.7
5       Num:201505      Score:0.0
6       Num:201506      Score:75.0
7       Num:201507      Score:95.1
8       Num:201508      Score:96.0
9       Num:201509      Score:85.0
10      Num:201510      Score:80.5
```

图 5-115　添加一些标注语句

## 5.8.13　在 awk 中进行数学运算

在之前已经学习了使用 bc 进行数学运算,本节将介绍如何在 awk 中对数据库中的各项元素进行数学运算。

请读者按以下步骤操作。

(1) 首先,创建一个数据库文件 ex0508_12_awk,内容如下。

```
potato      3.5     2       vegetable
pepper      4.2     1       vegetable
soap        5.6     1       daily necessities
towel       9.1     2       daily necessities
crisps      5.5     1       snacks
carrot      2.5     3       vegetable
raisin      4.0     1       snacks
bean        1.5     4       vegetable
clock       10.0    1       daily necessities
tomato      3.7     2       vegetable
```

我们可以将 ex0508_12_awk 文件视为一张购物小票，第 1 列为商品名称，第 2 列为商品单价，第 3 列为商品数量，第 4 列为商品类别。

（2）然后，创建 awk 命令文件 ex0508_13_awk，内容如下。

```
{
name = $1
unitprice = $2
quantity = $3
category = $4
price = unitprice * quantity
print name "\t" price
}
```

在此文件中，我们先将第 1 个字段的值赋给“name”，第 2 个字段的值赋给“unitprice”，第 3 个字段的值赋给“quantity”，第 4 个字段的值赋给“category”，再使用“price＝unitprice * quantity”来计算每种商品的总价，最后将计算结果打印成表。

（3）再将计算结果存储到 ex0508_14_awk 文件中并查看其中的内容，请执行下列命令完成上述操作。

```
awk -f ex0508_13_awk ex0508_12_awk > ex0508_14_awk
cat ex0508_14_awk
```

结果如图 5-116 所示。

```
joy@ubuntu:~/chapter05$ awk -f ex0508_13_awk ex0508_12_awk > ex0508_14_awk
joy@ubuntu:~/chapter05$ cat ex0508_14_awk
potato  7
pepper  4.2
soap    5.6
towel   18.2
crisps  5.5
carrot  7.5
raisin  4
bean    6
clock   10
tomato  7.4
```

图 5-116　计算并显示每种商品的总价

（4）接着，将 ex0508_13_awk 文件复制并重命名为 ex0508_15_awk 文件，再使用 vi 编辑器对其进行修改，修改后的内容如下所示。

```
{
name = $1
unitprice = $2
quantity = $3
category = $4
price = unitprice * quantity
totalprice = totalprice + price
print name "\t" price "\t" totalprice
}
```

在此文件中，我们使用“totalprice＝totalprice＋price”来计算所有商品的总价。

（5）再执行以下命令。

```
awk -f ex0508_15_awk ex0508_12_awk
```

“totalprice＝totalprice＋price”表示 awk 先将“totalprice”的值初始化为 0，在之后检索每一行时都会将当前行的“totalprice”和“price”的值相加再赋给“totalprice”，并将其显示在当前行的第 3 个字段，当检索完毕时便计算出所有商品的总价。如图 5-117 所示，在输出结

果中，最后一行的第 3 个字段为所有商品的总价。

```
joy@ubuntu:~/chapter05$ awk -f ex0508_15_awk ex0508_12_awk
potato  7       7
pepper  4.2     11.2
soap    5.6     16.8
towel   18.2    35
crisps  5.5     40.5
carrot  7.5     48
raisin  4       52
bean    6       58
clock   10      68
tomato  7.4     75.4
```

图 5-117　计算所有商品的总价

**注意**："totalprice＝totalprice＋price"可以简写成"totalprice＋＝price"。

（6）将 ex0508_15_awk 文件复制并重命名为 ex0508_16_awk 文件，再使用 vi 编辑器对其进行修改，修改后的内容如下。

```
$4 == "vegetable" {
name = $1
unitprice = $2
quantity = $3
price = unitprice * quantity
everytotalprice += price
print name "\t" price "\t" everytotalprice
}
```

此文件中的"$4＝＝"vegetable""将操作范围限制为类别是"vegetable"的商品，使用这种方法可以实现对商品的分类统计。

（7）最后，执行以下命令。

```
awk -f ex0508_16_awk ex0508_12_awk
```

如图 5-118 所示，在输出结果中，最后一行的第 3 个字段为所有蔬菜的总价。

```
joy@ubuntu:~/chapter05$ awk -f ex0508_16_awk ex0508_12_awk
potato  7       7
pepper  4.2     11.2
carrot  7.5     18.7
bean    6       24.7
tomato  7.4     32.1
```

图 5-118　计算所有蔬菜的总价

## 本章小结

在本章中，我们学习了多种进行文本操作的实用程序，包括使用 column 输出文本内容，使用 sort 进行排序，使用 comm、diff 和 uniq 进行文件内容的比较，使用 tr 和 sed 进行文件内容的修改，使用 grep 和 awk 进行文件内容的检索，并具体介绍了 awk 的使用方法；另外，还介绍了如何使用 bc 进行数学计算。

## 习题 5

### 1. 选择题

（1）执行（　　）命令在 practice050101 文件中查找字符串"turn on"。

A. grep turn on practice050101

B. grep 'turn on' practice050101

C. grep 'turnon' practice050101

D. grep 'turn'&'on' practice050101

(2) 使用(　　)选项让 grep 在查找时忽略字母的大小写。

A. -v　　B. -i　　C. -l　　D. -n

(3) 使用(　　)选项让 grep 输出不包含目标字符串的行。

A. -v　　B. -i　　C. -l　　D. -n

(4) 执行(　　)命令仅输出 practice050104 文件中的空行。

A. grep '^. $ ' practice050104　　B. grep ^ $ practice050104

C. grep '^ $ ' practice050104　　D. grep ' $ ' practice050104

(5) 在实用程序 bc 中，如果未事先设置 scale 的值，那么 10/3 的输出结果应该是(　　)。

A. 3.333　　B. 3.3　　C. 3.0　　D. 3

(6) 使用 sort 中的(　　)选项对文件内容按字典顺序排序。

A. -n　　B. -f　　C. -r　　D. -d

(7) 如果想要在对文件内容排序时忽略字母的大小写，那么需要使用 sort 中的(　　)项。

A. -n　　B. -f　　C. -r　　D. -d

(8) 使用 sort 中的(　　)选项对文件内容实现按数值大小排序。

A. -v　　B. -i　　C. -l　　D. -n

(9) 如果想要对文件内容按某一指定字段排序，那么需要使用 sort 中的(　　)选项。

A. -k　　B. -f　　C. -r　　D. -d

(10) 以下(　　)命令指定分隔符的选项是-t。

A. sort　　B. awk　　C. uniq　　D. diff

(11) 以下(　　)命令可以识别和删除文件中的相邻重复行。

A. diff　　B. comm　　C. uniq　　D. tr

(12) 执行 uniq -cd ex0505_01_uniq 命令的输出结果为(　　)。

A.
```
banana
cherry
```

B.
```
3 banana
2 cherry
```

C.
```
1 apple
1 cherry
1 date
```

D.
```
3 cherry
2 banana
```

(13) 执行(　　)命令可以得到如下结果：

```
Mars
Saturn
```

A. comm -3 ex0505_01_comm ex0505_02_comm

B. comm -2,3 ex0505_01_comm ex0505_02_comm

C. comm -1,2 ex0505_01_comm ex0505_02_comm

D. comm -12 ex0505_01_comm ex0505_02_comm

(14) 执行(　　)命令将 practice050114 文件中的“y”全部替换成“Y”。

A. tr 'Y' 'y' < practice050114　　B. tr 'y' 'Y' > practice050114

C. tr y Y > practice050114　　D. tr 'y' 'Y' < practice050114

(15) 在 tr 中可以使用(　　)选项删除指定的字符。

A. -d　　B. -f　　C. -r　　D. -n

(16) 在 practice050116 文件中,如果想要将所有包含“name”的行中的“age”全部替换成“score”,需要执行(　　)命令。

A. sed 's/name/age/score/g' practice050116

B. sed '/name/s/age/score/g' practice050116

C. sed '/name/s/age/score/' practice050116

D. sed '/name/age/score/g' practice050116

(17) 以下是 4 条 awk 命令,符合语法的是(　　)。

A. awk '^B' practice050117

B. awk {print $1} practice050117

C. awk '/Amy/' practice050117

D. awk '{print} practice050117'

(18) 设 practice050118 文件只有 1 行内容,执行 awk '{print "3-1",3-1,'3-1'}' practice050118 命令的输出结果是(　　)。

A. 3-1 2 2　　B. 3-1 2 3-1　　C. 3-1 3-1 2　　D. 2 2 2

(19) 执行以下(　　)命令,在 practice050119 文件中查找首字符为“1”到“5”中任意一个数字的记录。

A. awk '/[^1-5]/' practice050119　　B. awk '/^[1-5]/ practice050119'

C. awk '/^[1-5]/' practice050119　　D. awk /[^1-5]/ practice050119

(20) 如果想要打印在 practice050120 文件中字段总个数不为 2 的记录,那么需要执行(　　)命令。

A. awk '!(NF==2) {print}' practice050120

B. awk '!(FS==2) {print}' practice050120

C. awk '!(NR==2) {print}' practice050120

D. awk '!(OFS==2) {print}' practice050120

**2. 填空题**

(1) 执行_____命令将 practice050201 文件的内容按先行后列的顺序输出。

(2) 执行_____命令使用 grep 在当前目录下的所有文件中查找字符串“book”。

(3) 如果想要在多个文件中查找并输出包含目标字符串文件的文件名,那么需要使用 grep 中_____选项。

(4) 在正则表达式中,字符_____指示了行的末尾。

(5) 如果想在 Ubuntu 中进行一些简单的数学计算,那么可以使用实用程序_____。

(6) 在 bc 中输入_____来退出程序。

(7) 我们可以使用_____命令来查看 ASCII 码的相关信息。

(8) sort 中-r 选项的功能是_____。

(9) 在对 practice050209 文件使用 sort 时,如果想要将待比较内容限定在第 3、4 列字段,那么可以使用_____命令。

(10) 执行_____命令使用 sort 对 practice050210 文件进行排序,并指定逗号作为字

段分隔符。

（11）如果想要查看两个文件的不同之处并了解如何将其中一个文件转换为另一个文件，需要用到实用程序_____。

（12）执行_____命令使用 tr 将 practice050212 文件中的所有小写字母全部转换为大写字母。

（13）执行_____命令使用 sed 删除 practice050213 文件的第 3 行。

（14）在 awk 检索文件时，预定义变量_____表示当前被检索行的行号。

（15）在 awk 命令中可以使用_____选项来设置一个自定义变量。

（16）如果要在 practice050216 文件中查找第 1 个字段的首字母为“d”的记录，并输出符合条件记录的第 3 个字段，应该使用_____命令。

（17）在 awk 中，制表符属于预定义变量_____的默认值之一。

（18）在 awk 中，预定义变量 ORS 默认为_____。

（19）使用 awk 中的_____选项来调用 awk 命令文件。

（20）在 awk 命令文件中，等式“count-=price”的另一种写法是_____。

**3. 简答题**

（1）执行 grep '^' ex0502_04_grep 命令的输出结果是什么？

（2）practice050302 文件的内容如下：

```
grape
16
pepper
5
pear
32
apple
```

执行 sort -n practice050302 命令的输出结果是什么？

（3）怎样对 practice050303_01 文件进行排序，并将其重写到 practice050303_02 文件中？使用什么选项可以确保重写成功？

（4）在使用 comm practice050304_01 practice050304_02 命令来比较两个文件的异同时，输出的 3 列分别有什么含义？

（5）命令 sed '/name/d' practice050305 的含义是什么？

（6）在命令 sed 's/computer/& science/g' file 中，“&”指代的是什么？

（7）写出两种在 awk 中指定检索时使用逗号作为字段分隔符的方法。

（8）如果要使用 awk 输出 practice050308 文件的全部字段，应该使用哪两条命令？

（9）请写出执行 awk '{print name $3}' ex0508_03_awk 命令的结果。

（10）请编写一个 awk 命令文件，此文件需要符合以下要求：指定检索时的字段分隔符为“～”、记录分隔符为“-”，输出检索结果时的字段分隔符为“*”、记录分隔符为“$”，并输出第 1 个字段和第 2 个字段，字段间使用空格分开。