第5章 文本编辑

在第2章中使用Linux系统提供的众多命令对文件进行了各种操作，如显示文件内容、复制文件、删除文件，等等。那么，如何建立自己的文件呢？例如，编写C程序源文件。

其实，无论是一般文本文件、数据文件、数据库文件，还是程序源文件，对它们的建立和修改都要利用编辑器。在Windows系统中利用Word工具可以方便地编辑文本文件，而在UNIX/Linux系统上最受程序员喜爱的文本编辑器是vi。

UNIX/Linux系统中有多个编辑器，按功能它们分为两类：行编辑器（如ed，ex，edit）和屏幕编辑器（如vi）。vi是visual interface的简称，它汇集了行编辑和全屏幕编辑的特点，成为UNIX/Linux系统中最常用的编辑器，几乎每个UNIX/Linux系统都提供了vi。

在Linux系统中，还提供了vim（Vi IMproved）编辑器，它是vi的增强版本，与vi向上兼容。它支持多个窗口和缓冲、语法高亮显示、命令行编辑、联机帮助等功能。通常在Linux中用到的vi实际上是vim。

本章介绍如何使用vi建立、编辑、显示及加工处理文本文件。

5.1 进入和退出vi

只有进入vi编辑器之后才可以使用vi的命令；完成文本编辑以后，应退出vi，回到shell命令状态下。

5.1.1 进入vi

在系统提示符（设为$ ）下输入命令vi和想要编辑（建立）的文件名，便可进入vi。例如：

```
$ vi  example.c
```

如果example.c是一个新文件，在每一行开头都有一个“～”符号，表示空行，在最底行显示example.c［新文件］的信息，光标停在屏幕的左上角。如图5.1所示（注意：字符上带字符底纹的表示光标所在位置，下同）。

如果指定的文件已在系统中存在，那么输入上述形式的命令后，则在屏幕上显示出该文件的内容，光标停在左上角。在屏幕的最底行显示出一行信息，包括正在编辑的文件名、行数和字符个数，该行称做vi的状态行。

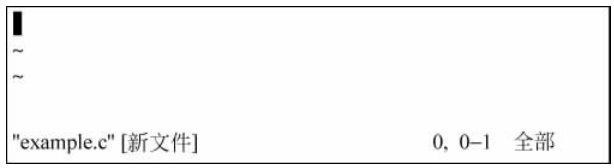

图 5.1 编辑(建立)新文件

5.1.2 退出 vi

当编辑完文件、准备返回到 shell 状态时，要执行退出 vi 的命令。在 vi 的命令方式下有几种方法可以退出 vi 编辑器：

(1) :wq 把编辑缓冲区的内容写到你编辑的文件中，退出编辑器，回到 shell 下。

(其操作过程是，先输入冒号“：”，再输入命令 wq。以下命令操作相同。)

(2) :ZZ (大写字母 ZZ)仅当做过修改时才将缓冲区内容写到文件上。

(3) :x 与:ZZ 相同。

(4) :q! 强行退出 vi。感叹号(!)告诉 vi，无条件退出，丢弃缓冲区内容。

应该强调一下，当利用 vi 编辑器编辑文本时，你所输入或修改的内容都存放在编辑缓冲区中，并没有存放在磁盘的文件中。如果你没有使用写盘命令而直接退出 vi，那么，编辑缓冲区中的内容就被丢弃了，你在此之前所做的编辑工作也就白费了。所以，在你退出 vi 时，应想清楚是否需要保存所编辑的内容，然后再执行合适的退出命令。

5.2 vi 的工作方式

vi 编辑器有三种工作方式：命令方式、插入方式和 ex 转义方式。通过相应的命令或操作，在这三种工作方式之间可以进行转换，如图 5.2 所示。

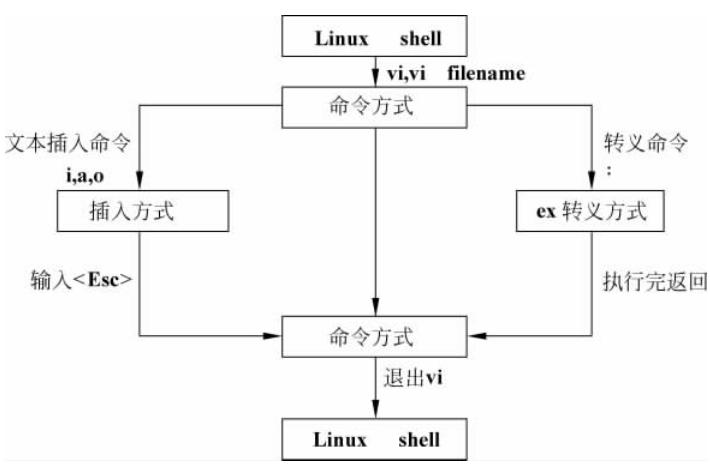

图 5.2 vi 编辑器三种工作方式

1. 命令方式

当输入命令 vi 进入编辑器时，就处于 vi 的命令方式。此时，从键盘上输入的任何字符都被当做编辑命令来解释，例如 a(append)表示附加命令，i (insert)表示插入命令，x 表示删除字符命令等。如果输入的字符不是 vi 的合法命令，则机器发出“报警声”，光标不移动。

另外，在命令方式下输入的表示命令的字符并不在屏幕上显示出来，例如输入 i，屏幕上并无什么变化，但通过执行 i 命令，编辑器的工作方式却发生变化——由命令方式变为插入方式。

2. 插入方式

通过输入 vi 的插入命令(i)、附加命令(a)、打开命令(o)、替换命令(s)、修改命令(c)或取代命令(r)可以从命令方式进入插入方式。在插入方式下，从键盘上输入的所有字符都被插入到正在编辑的缓冲区中，被当做该文件的正文。所以，进入插入方式后输入的可见字符都在屏幕上显示出来，而编辑命令不再起作用，仅作为普通字母出现。

由插入方式回到命令方式的办法是按 Esc 键(通常在键盘的左上角)。如果已在命令方式下，那么按 Esc 键就会发出“嘟嘟”声。为了确保你想执行的 vi 命令是在命令方式下输入的，不妨多按几下 Esc 键，听到嘟声后再输入命令。

3. ex 转义方式

vi 和 ex 编辑器功能是相同的，二者主要区别是用户界面。在 vi 中，命令通常是单个键击，例如 a，x，R 等。而在 ex 中，命令是以 Enter 键结束的正文行。vi 有一个专门的“转义”命令，可访问很多面向行的 ex 命令。为使用 ex 转义方式，可输入一个冒号(:)。冒号作为 ex 命令提示符出现在状态行(通常在屏幕最下一行)。按中断键(通常是 Del)可终止正在执行的命令。多数文件管理命令都是在 ex 转义方式下执行的(如读取文件，把编辑缓冲区的内容写到文件中)。例如：

```
:1, $ s/I/i/g  (按 Enter 键)
```

其功能是：从文件第 1 行至文件末尾所有的大写 I 字母都替换成小写字母 i。

转义命令执行后，自动回到命令方式。

5.3 文本输入命令

如果你想新建一个文件，或者想对已存文件进行添加或者要做较多的修改，那么你就要在插入方式下输入新的文本。文本插入命令总是把你带入插入方式。以下命令是纯粹的插入命令，使用时不会删除文本。

1. 插入命令

插入命令有两个：i (小写字母)和 I(大写字母)。

i 在该命令之后输入的内容都插在光标位置之前，光标后的文本相应向右移动。如输入<Enter>，就插入新的一行或者换行。

I 在光标所在行的行首插入新增文本，行首是该行的第一个非空白字符。当输入 I 命令时，光标就移到行首。例如，原来屏幕显示为：

```
/ * this is an example * /
int main ()
{
  a , b = 10 ;          （光标在等号 = 上）
  printf (" % d \ n", a = b * 2);
}
～
～
…
```

输入 I 命令后，显示为：

```
/ * this is an example * /
int main ()
{
  a, b = 10;            （光标移到该行的首字符 a 上）
  printf (" % d \ n", a = b * 2);
}
～
```

接着输入 int 和一个空格，显示为：

```
/ * this is an example * /
int main ()
{
  int a , b = 10;       （光标仍在该行的字符 a 上）
  printf (" % d \ n", a = b * 2);
}
～
～
```

2. 附加命令

附加命令有两个：a(小写字母)和 A(大写字母)。

a 在该命令之后输入的字符都插到光标之后，光标可在一行的任何位置。a 和 A 是把文本添加到行尾的唯一方法。

A 在光标所在行的行尾添加文本。当输入命令 A 后，光标自动移到该行的行尾。

例如，原来屏幕显示为：

```
/ * this is an example * /
int main ()
{
  int a;                (光标在该行的字符 a 上)
  printf (" % d \ n", a = b * 2);
}
~
~
```

输入命令 a 和字符串“,b=10”后,显示为:

```
/ * this is an example * /
int main ()
{
  int a,b = 10 ;        (光标停在该行的分号字符;上)
  printf (" % d \ n", a = b * 2);
}
~
~
```

3. 打开命令

打开命令有两个:o(小写字母)和 O(大写字母)。

o　在光标所在行的下面新开辟一行,随后输入的文本就插入在这一行上。

O　在光标所在行的上面新开辟一行,随后输入的文本就插入在这一行上。

在新行被打开之后,光标停在新行的行首,等待输入文本。例如,原来屏幕显示为:

```
/ * this is an example * /
int main()        (光标位于字母 m 的位置)
    printf (" OK ! ");
}
~
~
```

输入命令 o(小写字母)后,显示为:

```
/ * this is an example * /
int main()
                  (光标位于该行开头)
    printf (" OK ! ");
}
~
~
```

然后输入“{ int a,b=10;”显示为：

```
/* this is an example */
int main()
{  int a,b=10;        (光标位于该行末尾)
   printf (" OK ! ");
}
~
~
```

4. 插入方式下光标移动

在键盘的右下方有 4 个方向键，利用它们可以在插入方式下移动光标。每按一次上、下方向键，光标相应移动一行；每按一次左、右方向键，光标在当前行上相应移动一个字符位置。当光标位于行首（或行尾）时，按左向键（或右向键），系统会发出嘟嘟声。同样，当光标位于首行（或末行）时，按上向键（或下向键），系统也会发出嘟嘟声。

利用 Backspace（退格键）可将光标从当前行上回退一个字符，并且删除光标之前的一个字符。例如，屏幕显示的正文行（刚插入的）是：

```
int main(int argc,char **argv)        (光标位于该行行尾)
```

连续输入三次 Backspace 后，显示为：

```
int main(int argc,char **a)
```

5.4 光标移动命令

在命令方式下有很多命令可以在一个文件中移动光标位置。通常，除 4 个方向键外，还有 h、j、k、l 四个命令，以及按键 Space、Backspace、Ctrl+N 和 Ctrl+P 可以移动光标。

1. 向右（向前）移动一个字符

可以使用命令（键）l（小写字母）、Space、右向键将光标向右移动一个字符。

它们都可把光标向右移动一个字符。如果在相应命令的前面加上一个数字 n，那么，就把光标向右移动 n 个字符。例如：6l，则向右移 6 个字符；2+Space，则向右移 2 个字符。

但是，应注意：使用组合命令＜数字＞l 和＜数字＞右向键时，光标移动不能超过当前行的末尾，即，如果当前光标位置至行尾只有 6 个字符，那么 25l 也只能移到行尾。而＜数字＞+Space 组合命令则可以向下跨行移动光标。

2. 向左（向后）移动一个字符

可以使用命令（键）h（小写字母）、Backspace、左向键将光标向左移动一个字符。

如果在相应命令的前面加上一个数字 n，那么就把光标向左移动 n 个字符。例如，4h，则向左移动 4 个字符。但是，应注意：使用组合命令＜数字＞h 和＜数字＞左向键时，光标

移动不能超过当前行的行首，即，如果当前光标位置至行首只有 4 个字符，那么 10h 也只能移到该行行首。而＜数字＞＋Backspace 组合命令则可以向上跨行移动光标。

3. 移到下一行

可以使用命令(键)＋、Enter 将光标移到下一行的开头。如果在相应命令的前面先输入一个数字 n，那么光标就向下移 n 行。例如，3＋，则向下移 3 行，光标位于行首。2＋Enter，则向下移 2 行。

命令(键)j、Ctrl＋N 和下向键分别将光标向下移一行，但是光标所在列不变。若下一行比当前光标所在位置还短，则下移到行尾。如果在它们前面先输入一个数字 n，那么光标就向下移 n 行。例如，6j(或 6＋Ctrl＋N 或 6↓)向下移 6 行，而列数相同。

4. 移到上一行

可以使用命令(键)－、k(小写字母)、Ctrl＋P、上向键将光标上移一行。“－”命令把光标移到上行行首，而其余三个命令(键)把光标移到上一行的同一列上。可以在这些命令之前先输入一个数字 n，则光标就上移 n 行。例如，4－，则光标上移 4 行，位于行首。6k，则光标上移 6 行，列数不变。

以上四组基本移动光标的命令及其功能如图 5.3 所示。

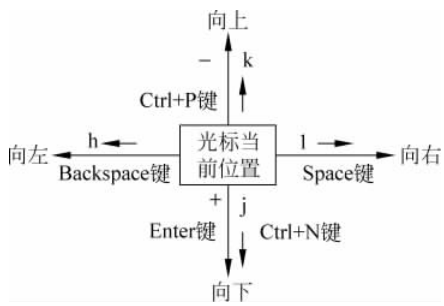

图 5.3　基本移动光标命令示意图

5. 移至行首

可以使用命令(键)^和 0(数字 0)将光标移到当前行的开头。

二者有些差别：命令 0 总是将光标移到当前行的第一个字符，不管它是否为空白符；而命令^将光标移到当前行的第一个非空白符(非制表符或非空格符)。例如，有如下文本行：

```
int main ()
{
    printf ("H e llo ! \ n ");     (光标位于字母 e 上)
}
```

输入^命令，光标移至字母 p。而随后输入 0 命令，光标左移至该行的第一列上，因为 p 的左边有若干空格。

6. 移至行尾

可以使用命令(键)$将光标移至当前行的行尾,停在最后一个字符上。如果在它前面先输入一个数字,例如6$,则光标移到当前行下面5行的行尾。

7. 移至指定行

可以使用命令(键)[行号]G(大写字母)将光标移至由行号所指定的行的开头。例如,输入3G,则光标移至第3行的开头。如果没有给出行号,则光标移至该文件最后一行的开头。

8. 移至指定列

可以使用命令(键)[列号]|(竖杠)将光标移至当前行指定的列上。如果没有指定列号,则移至当前行的第一列上。例如,9|,则移至第9列上。

5.5 文本修改命令

在命令方式下可使用有关命令对文本进行修改,用另外的文本取代当前文本。这意味着某些文本必须被删除。而删除的东西还可复原。

5.5.1 文本删除

1. 删除字符

可以使用命令x、X删除文本中的字符。命令x(小写字母)删除光标所在的字符。如果前面给出一个数值,例如,5x,则由光标所在字符开始向右删除5个字符。这是删除少量字符的快捷方法。

命令X(大写字母)删除光标前面的那个字符。如果前面给出数值,例如,8X,则由光标之前的那个字符开始、向左删除8个字符。

2. 删除文本对象

可以使用命令dd、D、d“光标移动命令组合”来删除文本对象。其中,命令dd删除光标所在的整行。命令D从光标位置开始删除到行尾。而字母d与光标移动命令组合而成的命令删除从光标位置开始至光标移动命令限定的位置之间的所有字符。向前(即向右)删除,会删除光标所在字符;而向后(即向左)删除并不包括光标所在字符。如果光标移动命令涉及多行,则删除操作从当前行开始至光标移动所限定的行为止。例如:

d0　从光标位置(不包括光标位)删至行首。

d3l　从光标位置(包括光标位)向右删3个字符。

d$　从光标位置(包括光标位)删至行尾。与D相同。

d5G 将光标所在行至第5行都删除。

5.5.2 复原命令

复原命令 u(undo)是很有用的命令,它的作用是取消刚执行过的命令。例如,屏幕显示为:

```
# include  <stdio.h>
int main()
{
  printf (" OK! ");
  printf("\n");
}
```

输入 d5G 后,显示为:

```
# include  <stdio.h>
}
```

接着输入 u,显示为:

```
# include  <stdio.h>
int main()
{
  printf (" OK! ");
  printf("\n");
}
```

表明:刚刚被 d5G 命令删除的 4 行正文又被恢复了。

复原命令有两种形式:u 和 U。它们都能取消刚才执行的插入或删除命令的效果,恢复到原来的情况。但二者在功能上又有所区别:小写 u 命令的功能是:如果插入后用 u 命令,就删除刚插入的正文;如果删除文本后使用它,就又恢复刚删除的正文。所有修改文本的命令都视为插入。而大写 U 命令则把当前行恢复成它被编辑之前的状态,不管你把光标移到该行后对它编辑了多少次。

5.5.3 重复命令

重复命令.(圆点)重复实现刚才的插入命令或删除命令。例如,屏幕显示为:

```
# include  < stdio.h >
int main()
{
}
```

输入 o 命令，并插入一行正文“printf（）；”，按 Esc 后，显示为：

```
# include  < stdio.h >
int main()
{
    printf () ;
}
```

连续输入两个“.”命令，显示为：

```
# include  < stdio.h >
int main()
{
    printf ();
    printf ();
    printf ();
}
```

输入一个 dd，再输入“.”命令，显示为：

```
# include  < stdio.h >
int main()
{
    printf ();
    printf ();
```

使用重复命令时应注意，它是在命令方式下起作用的。另外，它仅重复最新一次使用的插入命令或删除命令，而不能重复更早执行的命令。重复命令的执行与光标位置有关。例如，在第 1 行中插入字符串 abcd，回到命令方式下，然后将光标移至第 4 行的某一列，输入.命令后，则在该位置再一次插入 abcd，光标停在字母 d 上。

5.5.4 修改命令

命令 c、C 和 cc 的功能是修改文本对象，即用新输入的文本代替老的文本。它们等价于用删除命令删除老的文本，然后利用 i 命令插入新的文本。注意，输入修改命令后，就进入

到插入方式。所以,输入新文本后,还要按 Esc 键,才能回到命令方式。

1. 命令 c

命令 c(小写字母)的一般使用方式是:c 后面紧随光标移动命令(用来限定删除文本的范围),之后是新输入的文本,最后按 Esc 键。

例如,原来屏幕显示是:

```
/* thare are a C program */     (光标位于字母 a 上)
```

输入 c$后,屏幕显示为:

```
a C program */
```

接着输入"/* this is ",按 Esc 键,显示为:

```
/* this is  a C program */
```

表明:此操作是用后面输入的字符串"/* this is "代替了字符 a(不含)至行首的字符串"/* thare are "。由此可见,c 命令中修改文本的范围是由当前光标位置和光标移动命令二者共同限定的。

2. 命令 C

命令 C(大写字母)可以修改从光标位置到该行末尾的文本。它使用的一般方式是:C 后面紧接新输入的文本,最后按 Esc 键。它等价于 c$。例如,屏幕显示为:

```
/* this are a example */
```

输入 C,显示为:

```
/* this
```

接着输入字符串"is a program */",然后按 Esc 键,显示为:

```
/* this is a program */
```

这表明,原文本中的" are a example */ "被" is a program */ "所代替。

注意,C 命令除了可修改光标所在行的内容外,还可修改指定行数的文本内容。例如,3C,就把光标所在字符至本行行尾(不是整行)的字符和下面两个整行的内容都删除,由随后输入的文本内容代替。

3. 命令 cc

命令 cc 删除光标所在行整行(不是行的一部分),用随后输入的字符串替代。其余作用与 C 命令相同。例如,屏幕显示为:

```
/* this are test */
int main()
```

输入 cc 后,显示为:

```
int main()
```

接着输入字符串“/* this is a program */”，并按 Esc 键，显示为：

```
/* this is a program */
int main ()
```

cc 命令前可加上一个数字，表示要从当前行算起一共修改（删除）多少行。例如，5cc 表示先删除光标所在行及其下面 4 行，然后以新输入的文本代替。如果给定的行数太多，例如 999cc，超过光标所在行至文件末尾的总行数（例如仅为 5），则只是把光标所在行至文件末尾的这些行删除，然后等待输入新的文本内容。

5.5.5 取代命令

取代命令有两个：r 和 R。

1. 命令 r

命令 r 用随后输入的单个字符取代光标所在的字符。例如，屏幕显示为：

```
/* this as a program */        (光标在字母 a)
```

输入命令：ri，则显示为：

```
/* this is a program */
```

表明：字母 i 替代了原来光标所在的字符 a。

如果在 r 前面给出一个数字，例如 3，则从光标位置开始向右共有 3 个字符被新输入的字符替代。例如，屏幕显示为：

```
/* this is abcd */
```

输入：3rA，显示为：

```
/* this is AAAd */
```

2. 命令 R

命令 R 用随后输入的文本取代光标所在字符及其右面的若干字符，每输入一个字符就替代原有的一个字符。如新输入字符数超过原有对应字符数，则多出部分就附加在后面。例如，屏幕显示为：

```
/* this as a program */
main()
```

输入 R，接着输入 is a good example program */，按 Esc 键，显示为：

```
/* this is a good example program */
main()
```

如果在 R 命令之前给出一个数字，例如 5，则新输入的正文重复出现 5 次，依次覆盖当前行中光标及其后面的字符序列，而未被覆盖的内容仍保留下来。例如，屏幕显示为：

```
/* this is a good example program */
main()
```

输入 5RAB，然后按 Esc 键，显示为：

```
/* this is ABABABABA Bmple program */
main()
```

如果新输入的正文占多行，那么，也只有光标所在行的对应字符被覆盖，而其余各行的内容保留不变。例如，屏幕显示为：

```
/* this is program */
int main()
```

输入 R，接着依次输入：program ＜Enter＞ is good ＜Enter ＞ and nice */，按 Esc 键，显示为：

```
/* this program
is good
and nice */
int main()
```

5.5.6 替换命令

替换命令有两个：s 和 S。

1. 命令 s

命令 s(小写字母)用随后输入的正文替换光标所在的字符。其一般使用方式是：输入 s，随后输入替换正文，然后按 Esc 键。例如，屏幕显示为：

```
/* this is A example C program */
int main()
```

输入 s 后，显示变为：

```
/* this is A xample C program */
int main()
```

光标所在的 e 被删除。然后输入：good，并按 Esc 键，显示为：

```
/* this is A goo dxample C program */
int main()
```

如果只用一个新字符替换光标所在的字符，则命令 s 与 r 功能类似，如 sL 与 rL 的作用都是将光标所在的字符变为 L。但二者也有区别：r 命令仅完成置换，而 s 命令在完成置换同时，工作模式从命令方式转为插入方式。因此，使用命令 s 时，最后一定要按 Esc 键。

如果在 s 前面给出一个数字，例如 5，则光标所在字符以及其后的 4 个字符(共 5 个字

符）被新输入的字符序列替换。

2. 命令 S

命令 S（大写字母）用新输入的正文替换整个当前一行。例如，屏幕显示为：

```
/* this is a program */
int a;
int main()
```

输入 S 后，光标所在行成为空行，光标停在行的开头。接着输入：

```
# include < stdio.h > < Enter > # include < math.h > < Esc >
```

显示为：

```
/* this is a program */
# include < stdio.h >
# include < math.h >
int main()
```

可见执行 S 命令时，先删除当前行的内容，然后进入插入方式；输入完正文后，要用 Esc 键回到命令方式。因此，S 命令的一般使用方式是：输入 S，随后输入替换正文，最后是按 Esc 键。

如果在 S 之前给出一个数字，例如 3，则表示有 3 行（包括当前行及其下面的 2 行）要被新输入的正文替换。

5.6 字符串检索

编辑文本时，往往要根据给定的模式检索字符、词或者字符串。字符串检索既可以向前检索，也可以向后检索。

向前检索命令的基本格式是：

/模式< Enter >

在斜线之后给出要查找的模式（字符或字符串），然后按 Enter 键。系统在该文件中从光标所在行开始向前查找给定模式，并对所有与模式匹配的对象做上标记（高亮显示）。如果不存在与给定模式相匹配的字符串，则在状态行显示：Pattern not found（模式未找到）。

向后检索命令的基本格式是：

?模式< Enter >

这种形式的命令其功能与前一种相似，只是检索方向相反。

思考题

1. 进入和退出 vi 的方法有哪些？

2. vi 编辑器的工作方式有哪些？相互间如何转换？

3. 建立一个文本文件，如会议通知。

(1) 建立文件 notes，并统计其大小。

(2) 重新编辑文件 notes，加上一个适当的标题。

(3) 修改 notes 中开会的时间和地点。

(4) 删除文件中第 3 行，然后予以恢复。

4. 建立一个文本文件，将光标移至第 5 行，分别利用 c，C 和 cc 命令进行修改。

5. 在 vi 之下，把光标上、下、左、右移动的方式有哪些？

6. 解释下述 vi 命令有什么功能：

```
20G18 |x10cc3rk5s7S/this?abc? - 5
```

7. 如果希望进入 vi 后光标位于文件中的第 10 行上，应输入什么命令？

8. 不管文件中的某一行被编辑了多少次，总能把它恢复成被编辑之前的样子，应使用什么命令？

9. 要将编辑文件中所有的字符串 s1 全部用字符串 s2 替换，包括在一行中多次出现的字符串，应使用的命令格式是什么？