## 第10章 找回文本

在 GNU Emacs 中,无论你何时用"kill"命令从缓冲区中剪切了文本,你都能用一个"yank"命令将其重新找回。从缓冲区中剪切的文本被放在 kill 环中,yank 命令则将 kill 环中适当的内容重新插入到缓冲区中(不一定非得是原来的缓冲区)。

一个简单的 C-y(yank)命令,就从 kill 环中取出第一个元素插入到当前的缓冲区中。如果 C-y 命令后面紧跟一个 M-y 命令,则不是第一个元素而是第二个元素被插入到当前缓冲区中。连续的 M-y 命令则将使第三个元素或第四个元素等代替第二个元素而被插入到当前缓冲区中。当这样不断键人 M-y 而到达 kill 环的最后一个元素时,它就循环地将第一个元素插入到当前缓冲区中。(因此 kill 环被称为 "环",而不仅仅是列表。然面,保存文本的实际数据结构是一个列表。关于将一个列表作为环来处理的详细内容,参见附录B,"处理 kill 环"。)

## **10.1** kill 环总览

kill环是文本字符串的一个列表。它类似于下面的列表:

("some text" "a different piece of text" "yet more text")

如果 kill 环的内容就是这样的一个列表,当键人 C-y 时,字符串 "some text"将插入到当前缓冲区中光标当前所处的位置(即位点处)。

yank 命令也用于复制文本,这通过拷贝文本实现。被拷贝的文本不是从缓冲区中删除掉,但是这部分文本的一个拷贝被放到 kill 环中,并可以用 yank 命令将其找回来插入到当前缓冲区中。

能够将文本从 kill 环中找回的函数有三个: yank 函数,通常绑定到 C-y; yank-pop 函数,通常绑定到 M-y; rotate-yank-pointer 函数,这个函数被前面两个函数使用。

这些函数通过一个被称为 kill-ring-yank-pointer 的变量指向 kill 环。事实上, yank 函数和 yank-pop 函数中插入文本的代码都是下面这个表达式:

(insert (car kill-ring-yank-pointer))

在开始理解 yank 函数和 yank-pop 函数如何工作之前,分析一下 kill-ring-yank-pointer 变量和 rotate-yank-pointer 函数是必要的。

## 10.2 kill-ring-yank-pointer变量

就像 kill-ring 是一个变量一样,kill-ring-yank-pointer 也是一个变量。它通过被 绑定到相应的值来指向某些东西,这一点与其他 Lisp 变量没有什么两样。

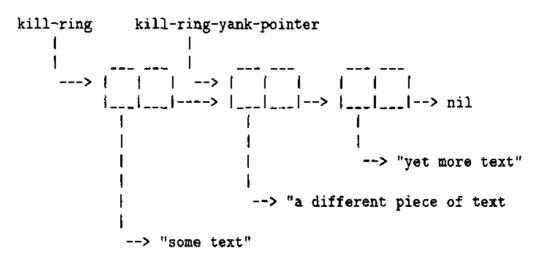
因而,如果 kill 环的值是:

("some text" "a different piece of text" "yet more text")

并且 kill-ring-yank-pointer 变量指向其中的第二个元素,则 kill-ring-yank-pointer 变量的值就是:

("a different piece of text" "yet more text")

就像在前面章节中解释的那样(参见第9章,"列表是如何实现的"), 计算机并不保存同时被 kill-ring 变量和 kill-ring-yank-pointer 变量指向的内容的两个拷贝。"a different piece of text"和 "yet more text"两个字符串都没有复制的副本。事实上,两个 Lisp 变量指向同样的一个文本块。用一个图来表示就是:



变量 kill-ring 和变量 kill-ring-yank-pointer 都是指针。但是在提到 kill 环本身时,常常就像是指其组成部分。在提到变量 kill-ring 时,就像它是一个列表而不是一个指向这个列表的指针。相反,在提到变量 kill-ring-yank-pointer 时,它确实是指向一个列表的指针。

这样两种讨论同样一件事情的方法,使人初看之下不免迷惑;但是仔细分析,就会发现这是有意义的。kill环一般被认为是保存从当前缓冲区中剪切的文本信息的完整结构。另一方面,变量 kill-ring-yank-pointer 则用于指示(即"指向")kill 环的某一部分,kill 环的这一部分的第一个元素将被插入到缓冲区中。

rotate-yank-pointer 函数改变 kill-ring-yank-pointer 变量指向的 kill 环中的元素。当这个指针指向的下一个元素超出了 kill 环的末尾,它自动将这个指针指向 kill 环的第一个元素。这就是将一个列表自动地转化为一个环的方法。rotate-yank-pointer 函数本身并不难理解,但是它包含了许多细节。在附录B"处现 kill 环"中,将描述这个函数以及更简单的 yank 函数和 yank-pop 函数。

## 10.3 练习: 使用 yank函数和 nthcdr 函数

- 使用 C-h v(describe-variable) 命令,看一看你的 kill 环的值。为你的kill环增加几个元素,然后再看一看它的值。使用 M-y(yank-pop) 命令在 kill环中移动。你的 kill 环中有多少个元素? 用 kill-ring-max 得到这个值。你的 kill 环是否已经满了,或者你能否可以继续往其中保存更多的文本块?
- 使用 nthcdr 和 car 函数构建一系列表达式分别来返回一个列表的第一个元素、第二个元素、第三个元素、第四个元素、等等。