# 11. CUT BUFFERS

When Elvis deletes text, it stores that text in a cut buffer. This happens in both visual mode and EX mode. There are 36 cut buffers: 26 named buffers ("a through "z), 9 anonymous buffers ("1 through "9), and 1 extra cut buffer (".). There is no practical limit to how much text a cut buffer can hold.

## 11.1 Putting text into a Cut Buffer

In visual mode, text is copied into a cut buffer when you use the d, y, c, C, s, or x commands. There are also a few others.

By default, the text goes into the "1 buffer. The text that used to be in "1 gets shifted into "2, "2 gets shifted into "3, and so on. The text that used to be in "9 is lost. This way, the last 9 things you deleted are still accessible.

You can also put the text into a named buffer -- "a through "z. To do this, you should type the buffer's name (two keystrokes: a double-quote and a lowercase letter) before the command that will cut the text. When you do this, "1 through "9 are not affected by the cut.

You can append text to one of the named buffers. To do this, type the buffer's name in uppercase (a double-quote and an uppercase letter) before the d/y/c/C/s/x command.

The ". buffer is special. It isn't affected by the d/y/c/C/s/x command. Instead, it stores the text that you typed in the last time you were in input mode. It is used to implement the . visual command, and ^A in input mode.

In EX mode, the :delete, :change, and :yank commands all copy text into a cut buffer. Like the visual commands, these EX commands normally use the "1 buffer, but you can use one of the named buffers by giving its name after the command. For example...

```
:20,30y a
```

... will copy lines 20 through 30 into cut buffer "a.

You can't directly put text into the ". buffer, or the "2 through "9 buffers.

## 11.2 Pasting from a Cut Buffer

There are two main styles of pasting: line-mode and character-mode. If a cut buffer contains whole lines (from a command like "dd") then line-mode pasting is used; if it contains partial lines (from a command like "dw") then character-mode pasting is used. The EX commands always cut whole lines.

Elvis also supports a limited form of rectangular cut and paste.

This is handy, for example, when you want to swap two columns in a table. The only way to put a rectangular area into a cut buffer is to select it via the visual ^V command, and then yank or delete it with a y or d command, respectively. When a cut buffer has been filled this way, it will be pasted using rectangle-mode pasting.

Character-mode pasting causes the text to be inserted into the line that the cursor is on.

Line-mode pasting inserts the text on a new line above or below the line that the cursor is on. It doesn't affect the cursor's line at all.

In visual mode, the p and P commands insert text from a cut buffer. Uppercase P will insert it before the cursor, and lowercase p will insert it after the cursor. Normally, these commands will paste from the "1 buffer, but you can specify any other buffer to paste from. Just type its name (a double-quote and another character) before you type the P or p.

In EX mode, the :put command pastes text after a given line. To paste from a buffer other that "1, enter its name after the command.

## 11.3 Macros

The contents of a named cut buffer can be executed as a series of ex/vi commands.

To put the instructions into the cut buffer, you must first insert them into the file, and then delete them into a named cut buffer.

To execute a cut buffer's contents as EX commands, you should give the EX command :@ and the name of the buffer. For example, :@z will execute "z as a series of EX commands.

To execute a cut buffer's contents as visual commands, you should give the visual command @ and the letter of the buffer's name. The visual @ command is different from the EX :@ command. They interpret the cut buffer's contents differently.

The visual @ command can be rather finicky. Each character in the buffer is interpreted as a keystroke. If you load the instructions into the cut buffer via a "zdd command, then the newline character at the end of the line will be executed just like any other character, so the cursor would be moved down 1 line. If you don't want the cursor to move down 1 line at the end of each @z command, then you should load the cut buffer by saying 0"zD instead.

One way to store keystrokes into a buffer for use with the visual @ command is via the [key and ]key commands. They record keystrokes into a cut buffer as you type them.

## 11.4 The Effect of Switching Files

Elvis 2.1 retains the contents of all cut buffers when you switch files, e.g. via a :next or :edit command. This differs from the traditional behavior of vi.

In the real vi and in elvis 1.X, the anonymous buffers ("1 through "9) were clobbered and the named buffers ("a through "z) were left intact. This made sense then, but since elvis 2.1 allows you to edit several files at the same time, the rules changed.

## 11.5 Cut & Paste Between Applications

There is a special cut buffer named "^ (doublequote-carat) which accesses the GUI's cut&paste feature. Each time you yank text into the "^ cut buffer, it is copied to the GUI's clipboard. Each time you paste text from the "^ cut buffer, elvis reads from the GUI's clipboard.

Not all GUIs have clipboards. For example, the plain old termcap interface doesn't have one. The "^ cut buffer still exists, but it resides inside elvis, just like any other cut buffer. (Exception: The Windows version of the termcap interface has been patched to access the Windows clipboard.)

Elvis' X11 interface does use X's clipboard. Clicking the middle mouse button causes the clipboard's contents to be inserted at the cursor position. When you select text via the mouse, the text is immediately copied to the clipboard. Text that you select via keyboard commands is *not* automatically copied because elvis has no way of knowing when you're through selecting it.

The Win32 version of elvis has the usual Cut/Copy/Paste toolbar buttons and menu items.