Mastering Linux by Paul S. Wang

Appendix: An Introduction to `vim`

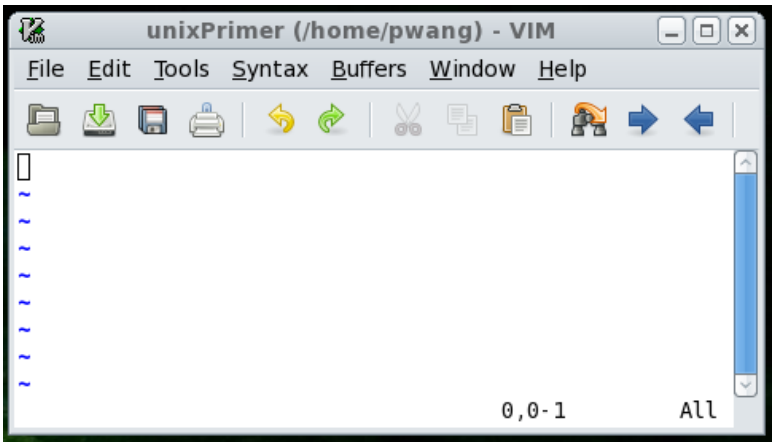
Creating and editing text files is basic to many tasks on the computer. There are many text editors for Linux including the GUI-based **gedit** and the terminal-based **nano**, **pico**, and **emacs**. However, **vim** (**vi** iMproved) remains the editor preferred by many. In any case, pick a text editor and learn it well. It can make life on Linux so much easier.

To invoke the editor **vim** for editing *file*, type from the Shell level

vim *file* (**vim** in terminal mode)
gvim *file* (**vim** in GUI mode)

If the file exists, **vim** displays it for editing. Otherwise, you are creating a new file by that name. In terminal mode, the terminal window and the keyboard are used for editing, whereas in GUI mode, a window (Figure 2) with a toolbar and menus is displayed and you can use both the keyboard and mouse for editing. Information in this section applies also to the **vi** command and to the

FIGURE 2: Gvim Editing Window



view command which is a read-only version of **vi**.

Once inside **vim**, you are working in a text-editing environment controlled by **vim** and you can create text, make changes, move text about, and so on. To exit from **vim** and save the file with the changes, type the **vim** command:

ZZ (save file and exit **vim**)

which makes the changes permanent in the file and terminates **vim**. If you want to quit **vim** without saving the changes, type the **vim** command:

:q! (exit **vim**, no save)

followed by RETURN.

Let us go through a quick editing session. Type:

vim `myprog`

to call up a file named `myprog`. Because `myprog` does not exist, it will be created. The screen will clear except for a column of tilde characters (~), a cursor, and perhaps a brief message on the last line. The **vim** editor has two input modes: the *command* mode and the *insert* mode. The command mode moves the cursor with the arrow keys, deletes text, moves text, and so on; the insert mode enters and changes text. The **vim** editor always begins in the command mode. You enter the insert mode simply by pressing **i** (for "insert", no RETURN is necessary) and you exit insert mode by pressing ESC. This returns you to command mode. Now press **i** and enter the following text:

```
echo  It is time for all
echo  good men to come to
echo  the aid of their country.
```

Type RETURN or ENTER at the end of a line. This puts a NEWLINE character at the end of the line, which is how Linux text files separate lines. Exit the insert mode by pressing ESC.

If you want to change `It is time` to `Now is the time`, the procedure is simple. Recall that we are in the command mode after pressing ESC, so we may use the arrow keys to move the cursor to the `I` of `It`, the beginning of the first word we want to change. Now press **dw**, and `It` is deleted (**dw** will be explained shortly). Note how the word disappears and the rest of the line slides over so that the `i` in `is` is now at the cursor. Next, press **i** to return to the insert mode and type the word `Now`, a space, and then ESC to exit the insert mode. We want to insert the word `the` before the word `time`, so position the cursor at the *t* in *time*. Now, press **i** for the insert mode and type `the` and a space. Hit ESC to return to the command mode. The correction is finished.

New users of **vim** sometimes find it difficult to always remember which mode they are using. Vim makes the mode obvious by displaying `-- INSERT --` on the bottom line as a reminder.

As you use **vim**, any changes you make are only to a buffer. To leave the editor and save myprog to disk, type **ZZ** (without hitting RETURN). The editor will report that it has saved the buffer in a new file named `myprog` and then return you to the Shell. You have just created and edited a file named `myprog` containing the following three lines:

```
echo  Now is the time for all
echo  good men to come to
echo  the aid of their country.
```

This file is an actual program consisting of Shell-level commands. Such programs are called *Shell scripts*. Shell-level programming is the subject of (Chapter 6). Now type:

chmod `+x myprog`

to make the file `myprog` executable. Then type:

`./myprog`

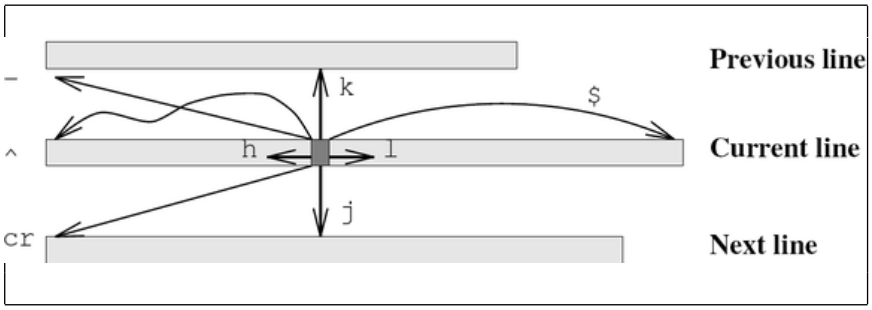
to execute the simple Shell script.

The **vim** editor is very powerful, and many different commands are available to handle all sorts of needs, many of which you may never use. The following is a list of basic **vim** commands that will serve most practical purposes:

Cursor Movement

h	(or LEFT ARROW) moves the cursor one position to the left (Figure 3)
j	(or DOWN ARROW) moves the cursor down one position
k	(or UP ARROW) moves the cursor up one position
l	(or RIGHT ARROW) moves the cursor right one position
+	(plus sign) moves the cursor to the beginning of the next line (same as RETURN or ENTER)
–	(minus sign) moves the cursor to the beginning of the previous line
^	moves the cursor to the beginning of the current line
$	moves the cursor to the end of the current line
G	moves the cursor to the end of the file
***n*G**	moves the cursor to the beginning of the *n*th line of the file

FIGURE 3: Cursor Movement



Deletion

x	erases the character at the cursor; *n***x** erases *n* characters from the cursor. The positive integer *n* is called a *repeat number* or *count*. Many of the commands given here can take a repeat number.

*n***dw** deletes *n* words. As noted previously, the *n* is a repeat number and may be omitted if the repeat number is one.

*n***dd** deletes *n* lines; **dd** for deleting one line. Where a line is deleted, an `@` is sometimes shown in its place to indicate its absence.

Insertion

i enters the insert mode. Subsequent keystrokes will be inserted immediately to the left of the cursor. When the insertion is completed, press ESC to mark the end of the insertion and to return to the command mode. An insertion may consist of more than one line.

a enters the insert mode. **a** is identical to **i**, except that the insertion occurs to the immediate right of the cursor.

Other

/*pattern*ESC searches from the current cursor position forward, or down, in the file to the first occurrence of *pattern*. For example, `/while`ESC searches for the text pattern `while`. The cursor will be left at the beginning of the pattern found. Some special characters can be used to specify patterns. The RETURN key can be used instead of ESC to terminate a search pattern.

?*pattern*ESC does the same search in the backward direction toward the beginning of the file.

ZZ saves the file being edited and exits from **vim**

:q! quits; exits **vim** without saving the file

The Linux *interrupt character* (usually CTRL+C) is used to send an interrupt to **vim**. The interrupt causes **vim** to abort the current command and return to the command mode.

Although this section is only an introduction, it should provide enough material for you to get started editing your files. For more information see the appendix on **vi**.