Text Editors – nano, vim

Prabhjeet Singh

Text Editors

1. Nano - Nano's another editor, inspired by Pico.

   DESCRIPTION

   nano is a small and friendly editor. It copies the look and feel of Pico, but is free software, and implements several features that Pico lacks, such as: opening multiple files, scrolling per line, undo/redo, syntax coloring, line numbering, and soft-wrapping overlong lines.

   When giving a filename on the command line, the cursor can be put on a specific line by adding the line number with a plus sign (+) before the filename, and even in a specific column by adding it with a comma. (Negative numbers count from the end of the file or line.) The cursor can be put on the first or last occurrence of a specific string by specifying that string after +/ or +? before the filename. The string can be made case sensitive and/or caused to be interpreted as a regular expression by inserting c and/or r after the + sign. These search modes can be explicitly disabled by using the uppercase variant of those letters: C and/or R. When the string contains spaces, it needs to be enclosed in quotes. To give an example: to open a file at the first occurrence of the word "Foo", one would do:

   ```
   nano +c/Foo file
   ```

   As a special case: if instead of a filename a dash (-) is given, nano will read data from standard input.

EDITING

Entering text and moving around in a file is straightforward: typing the letters and using the normal cursor movement keys. Commands are entered by using the Control (^) and the Alt or Meta (M-) keys. Typing ^K deletes the current line and puts it in the cutbuffer. Consecutive ^Ks will put all deleted lines together in the cutbuffer. Any cursor movement or executing any other command will cause the next ^K to overwrite the cutbuffer.

A ^U will paste the current contents of the cutbuffer at the current cursor position.

When a more precise piece of text needs to be cut or copied, one can mark its start with ^6, move the cursor to its end (the marked text will be highlighted), and then use ^K to cut it, or M-6 to copy it to the cutbuffer. One can also save the marked text to a file with ^O, or spell check it with ^T.

On some terminals, text can be selected also by holding down Shift while using the arrow keys. Holding down the Ctrl or Alt key too will increase the stride. Any cursor movement without Shift being held will cancel such a selection.

The two lines at the bottom of the screen show some important commands; the built-in help (^G) lists all the available ones. The default key bind-ings can be changed via a nanorc file -- see nanorc(5).

```
$ nano file1.txt
```

Alt+U → undo a change

Alt+e → redo the change

Page up and page down also can be used

Single line traversal using arrow keys.

Ctrl+c → find the

Alt+\ → move to the top/starting

Alt +/ → move to the end

To copy– select with alt+a and then move cursor and after selecting the text use

Alt+k

to cut → use ctrl+k

ctrl+w → to search a text

alt+w → to check next occurrence after search

alt+q → to check backward

ctrl+o → to save the file

ctrl+x → to exit the editor.

VIM editor → Vi IMproved, a programmer's text editor.

Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between Vim and Vi.

While running Vim a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below.

Most often Vim is started to edit a single file with the command vim file

More generally Vim is started with:

```
vim [options] [filelist]
```

If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited.

```
$ vim file1.txt
```

Click i for the insert mode
edit the file

:wq -> save and exit from vim mode

Esc → to exit edit mode.

/ → to search

n key → for next match

N key → previous match

l key → move left

k key – to move up

j Key → move down

h Key → move right

0 → beginning of the line – zero

$ -> end of the line’

Jump to the line with line number → :31 → go to the 31 line number.

U → undo

Ctrl+r → redo