Editing Text

Contents

Overview

In this module we will go over how `vim` works as well as writing and inserting with `vim`.

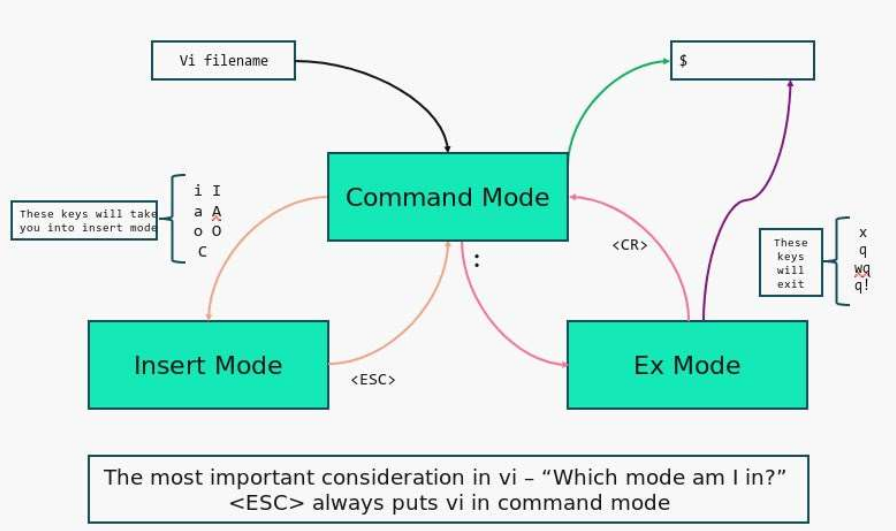
Linux Text Editors

`vi` is a visual display-presentation editor.
It is based on an underlying line editor (ex).
The interface is unusual and difficult for newcomers to grasp.

The GPL version, provided on Linux is called `vim`.
`vim` is an imporved version of `vi` with wider functionality.

It has three modes of operation:

- `Command` - for moving around the text
- `Insert` - Adding text
- `Ex` - Extended commands



Command Mode

The command mode is the default postion on entry.
The basic format for the `vi` commands are as follows:

```
[count][command][scope]
10dw
```

- `10` is the count
- `d` is the command
- `w` is the scope

This command deletes 10 words.

Some commands place `vi` in insert mode (Displayed above).
You need to press `<ESC>` to stop entering text.

Some commands require a text arguement.
In which case you enter the test (it will appear at the bottom of the screen) and press `<CR>` to execute the command.

Screen Navigation

- `0` - Start of line
- `$` - End of line
- `nG` - Go to line n (where n is a number)
- `G` - Go to last line
- `^F` - Forward one screen
- `^B` - Backwards one screen
- `^D` - Down half a screen
- `^U` - Up half a screen
- `b` - Beginning of a word
- `e` - End of a word
- `w` - Start of next word

Inserting

The different insert choices, do the following:

- `i` - Insert before cursor
- `a` - Append after cursor
- `I` - Insert at start of line
- `A` - Append at the end of the line
- `O` - Open new line above cursor
- `o` - Open new line below cursor

Deleting Commands

- `x` - Delete character under cursor
- `X` - Delete character to left of cursor
- `d<scope>` - Delete to buffer, specifying size of text to change
- `dd` - Delete complete line
- `dw` - Delete until start of next word
- `d3l` - Delete next three characters
- `d0` - Delete back to start of line
- `d1G` - Delete back to start of file
- `dG` - Delete until end of file
- `d$` - Delete until end of line
- `D` - Delete until end of line

Changing Text

- `r` - Replace character under cursor (type new character next)
- `R` - Enter replace mode (overtype text) until `<ESC>` is pressed
- `c<scope>` - Change text, specifying size of text to change
- `cc` - Change complete line
- `cw` - Change until start of next word
- `c3l` - Change next three characters
- `c0` - Change back to start of line
- `c1G` - Change back to start of file

- `cG` - Change until end of file
- `c$` - Change until end of line
- `C` - Change until end of line

Cut, Copy and Paste

Text can be copied using the 'yank' command.

- `y<scope>` - Yanks (copies) text specifying size of text to copy
- `yy` - Yank complete line
- `yw` - Yank to start of next word
- `y0` - Yank to start of line
- `yG` - Yank to end of file

The text can also be pasted back.

- `p` - Paste buffer after current position
- `P` - Paste buffer before current position

These commands manipulate the file itself, allowing you to save and exit `Vim`.

- `:r` - File read file and place contents after current line (merge)
- `:w` - Write file
- `:w!` - Write file, overriding protection mode
- `:w file` - File write to named file
- `:wq` - Write file and quit
- `:x` - Write file and quit (exit)
- `:q` - Quit without writing
- `:q!` - Force quit even if file not written
- `:!cmd` - Execute shell command
- `:r!cmd` - Insert contents of shell command
- `zz` - Write and exit vi from command mode

Alternative Editors

The standard Unix/Linux editor is `vi`.

`X-window environments` provide additional editors:

- Gedit
- Emacs
- Nano

Editing text files without editor

In a constrained Linux environments (for example in Docker containers with minimal dependencies) you often cannot find any popular editor (even Nano or Vi). If you need to edit text in such an environment, you can use a standard output redirection with `echo` and `cat` commands.

For example if you need to configure a DNS server in `/etc/resolv.conf` file without an editor, you can just redirect the content of that file using `echo` command:

```
echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

In order to write multiple-line content to the file, execute `cat` command with `-` argument. Dash argument tells cat to start reading content from the standard input:

```
$ cat - > /etc/resolv.conf
# This is my nameserver
nameserver 8.8.8.8
```

Then you are done writing your content, press Ctrl+D key combination to close the stream and write its content to the file. You can then use `cat` command as well to see the contents of the file you edited:

```
$ cat /etc/resolv.conf
# This is my nameserver
nameserver 8.8.8.8
```

Tutorial

There is no tutorial for this module.

Exercises

Now conduct your own research on the three alternative editors.

▶ Examples