An Introduction to Command-Line Text Editors

UNO CTF

Contents

1 Terminal-Based Text Editors

1.1 Emacs

Emacs is an extensible text editor that gives you a ton of tools and options right out of the box. While it is primarily a text editor, it can be used to read email, manage a calendar, debug code, access the filesystem, use a serial terminal for communication with outside hardware, and more.

While Emacs now has a GUI, it is still at its core a terminal editor. Because of this, it is designed well for touch-typing–you never need to use the mouse!

You do not need to worry about this now, but Emacs is built as a Lisp interpreter that has its own Lisp dialect: Emacs Lisp. If you choose to customize Emacs further than is allowed by its internal settings menus, you can even create your own add-ons in Emacs Lisp and customize your `.emacs` file (the user settings file) as you please. However, for now, let's proceed to commands.

1.1.1 Command Structures

In a command, a capitalized character (such as "C" for "Control" or "M" for "Meta") indicates a *modifier* key. When these keys are used, they are used in conjunction with another to modify that character in the form of "C-x". So to type the command "C-x", you would hold the `[Control]` key and hit the `x` key. For a command such as "C-h d", you would first hold the `[Control]` key and hit `h`. Then you would release both keys, and hit `d`. If you have two or more modified keys that use the same modifier in a row, you do not have to release the modifier key when typing the command. So for "C-x C-s" to save, you would hold `[Control]`, and while doing so you would press and release `x`, and then press and release `s`, and then release the modifier.

1.1.2 Modifier Keys

The two primary modifier keys used are Control and Meta. Control typically refers to the `[Control]` key, while Meta refers to either the `[Alt]` key or the `[Escape]` key. Occasionally your terminal program may use the `[Alt]` key to handle some keyboard shortcuts, so `[Escape]` may be more convenient to use.

1.1.3 Buffers

Emacs allows the user to maintain separate frames, known as *buffers*. These can be used to simultaneously open multiple files, or use multiple applications within Emacs. For example, a user could be working on a C program, and switch buffers to check a calendar.

1.1.4 Windows

A user can have multiple buffers visible at once. When Emacs is opened, typically one *window* is visible, but it can be split into multiple windows. A window in this context is not the same as those in a GUI windowing system, but rather, Emacs can make use of a single terminal session to represent multiple distinct documents or applications within that one terminal. A window in Emacs displays a single buffer. In each window, a user can swap out buffers as necessary.

1.1.5 Minibuffer

Emacs commands often make use of what is called the *minibuffer*. It is a small line at the bottom of the Emacs terminal session that takes user input for commands that require additional input. For example, when opening a file from within Emacs, Emacs will ask for the file name, and at this time, all user input will be entered into the minibuffer.

1.1.6 Common Commands

Task	Command
Emacs tutorial	`C-h t`
Open a file	`C-x C-f`
Save a file	`C-x C-s`
Exit Emacs	`C-x C-c`
Split window vertically	`C-x 3`
Split window horizontally	`C-x 2`
Switch buffer in current window	`C-x b`
Kill (close) a buffer	`C-x k`
Move cursor to next window	`C-x o`
Delete current window	`C-x 0`
Delete all but current window	`C-x 1`
Search current buffer	`C-s`
Continue current search	`C-s`

1.2 vi/Vim

Vim, the usual newer form of the vi text editor, is available on generally any Linux distribution by default either as `vi` or `vim`. You may find yourself in a situation with a basic Linux installation and no internet connection to download another editor, and so because of Vim's common availability

alone, it is worth knowing at least how to open, save, and close a file (even if you prefer another editor) in Vim.

1.2.1 Modes

Vim provides two primary modes: *command* mode and *insert* mode.

1. Command mode Command mode allows for navigation of pre-existing text. To move around, you can hit `h` to move left, `l` to move right, `k` to move up, and `j` to move down. This is done so movement in insert mode can be done purely by the right hand, and quickly. Here, you can delete text. To do so, move to over a character and hit `x`. To insert text (thus entering *insert* mode), type `i`.

2. Insert mode Hitting the `i` key in command mode enters *insert* mode. Here, text is inserted. Note that you can not use the typical movement keys here–you must re-enter command mode to do this. Hit `[Esc]` to do so. Vim has a number of additional modes; you can also hit `[Esc]` to exit any of these.

1.2.2 Common Command Mode Commands

Task	Command
Quit with no unsaved changes	`:q`
Force quit (ignore unsaved changes)	`:q!`
Save current file	`:w`
Save and Exit	`:wq`
Save to file	`:w [file]`
Open a file	`:e [file]`
Delete character under cursor (forwards)	`x`
Delete character under cursor (backwards)	`X`
Search forward	`/<pattern>`

1.3 nano

Nano is an editor that is much less-featured than Vim or Emacs–hence the name. Nano has no modes–if you type any key sequences unmodified by Control or Meta (like Emacs' meta), text will be entered. Common functions are visible at the bottom of the screen–most commands are Control sequences–Control is indicated with the caret (^). For example, to exit Nano, the command is `^X`, so you would hold `[Ctrl]` and hit `X`. A Meta command is prefixed with the string `"M-"`, and will be completed with some other character.

1.3.1 Common Commands

Task	Command
Open a file	`^R`
Save a file	`^O`
Exit nano	`^X`
Search	`^W`
Continue search	`M-W`