vi tutorial

vi Tutorial

For this tutorial you will first create and edit a file. The following conventions will be used:

- Instructions will be in: this normal typeface.
- Things which you will type will be in: `this typeface`.
- Things which will appear on the UNIX screen will be in: this *italics typeface.*

Part 1: Connect to the CSIF

It is unlikely that you have vi installed on your machine but it is available on the CSIF. If you are unsure of how to connect to the CSIF refer back to the Linux Tutorial.

Alternatively you can install vi on your own machine and complete the tutorial there but instructions on how to do so will not be provided in this tutorial.

Part 2: Creating and Editing a Poem

Starting VI

You start vi by typing vi [filename] at the command prompt. The filename is optional.

Type: `vi`

Text Entry Mode

When you start vi, you are in Command Mode. To enter text in the vi buffer you must change to Text Entry
Mode (also called Insert Mode). You change to Text Entry mode by pressing one of three keys

1. 'i': text is inserted in front of the cursor
2. 'a': text is added after the cursor;
3. 'R' = text is replaced starting with the character under the cursor.

Press `a` and enter the following poem:

```
I always remember standing in the rains,
On a cold and damp september,
```

```
Brown Autumn leaves were falling softly to the ground,
Like the dreams of a life they slide away.
```

Command Mode

Command Mode is used for all vi operations other than text entry. To change to Command Mode press the `ESC`
key. If you are unsure whether you are in Command Mode, you may always press the ESC key again without
any harm other than an annoying beep.

Cursor Movement

To move around in your file you can use the following keys while in Command Mode

- k = up one line
- j = down one line
- l = right one character
- h = left one character
- w = move forward one word
- ^ = start of line
- $ = end of line
- Ctrl-f = down one screen
- Ctrl-b = up one screen
- and the arrow keys work as expected.
  You may also move to a specific line number by typing ":nn" where nn is the line number to which you wish to move.

Again the above all work while in **Command Mode**.

Ensure you are in Command Mode by pressing the `ESC` key.

Move to the fourth line by typing `:4`

Now position the cursor at the beginning of the line enter Text Entry Mode by pressing `i`

Type in the word `Just` and then press `ESC` to return to Command Mode.

Deleting Text

There are four ways to delete items in the vi buffer:

1. x = delete the character under the cursor
2. dw = deletes from the cursor to the end of the word under the cursor

3. [number]dd = delete the line under the cursor, or if you precede dd with a number it will delete that many lines starting with the line under the cursor
4. :d = deletes the lines indicated in the range. We will describe the syntax of ranges shortly.

Now delete the word "always" from the first line by:

1. typing `:1` to move to the first line
2. typing `w` to move forward one word
3. typing `dw` to delete the word "always".

Now move the cursor over the "s" in "rains" on the first line and delete the "s" by typing `x`.

While in command mode you will occasionally need to indicate a range of line numbers. There are two special
symbols used in the range syntax:

1. $ = end of file
2. . = current line, inclusive.

Ranges are indicated by typing first the beginning line number (or special symbol), second a comma, and finally the ending line number
(or special symbol). For example, 3,5 is line numbers 3,4, and 5. 4,$ is from line 4 to the end of the file. . , 9 is
from the current line to line number 9. There two commands to toggle the display of line numbers:

1. :set number = turn on line numbers
2. :set nonumber = turn off line numbers

Type `:set number` to display line numbers.

To give you a few useless lines to delete do the following:

1. move to line 3 by typing :3;
2. change to Text Entry Mode by pressing i;
3. type each of the following letters followed by the enter key a b c d e f g h i j k. When done your screen should look like the following:

```
1 I remember standing in the rain,
2 On a cold and damp september,
3 a
4 b
5 c
6 d
7 e
```

```
8 f
9 g
10 h
11 i
12 j
13 k
14 Brown Autumn leaves were falling softly to the ground,
15 Just Like the dreams of a life they slide away.
16
```

Now move the cursor to line #4 and delete lines 4 and 5 by typing `2dd`

Now delete the new lines 6 through 8 by typing `:6,8d`

Now move to the line #6 which has an "i" on it and delete it and the next two lines by typing `:.,8d`

Move to the last useless line, line #5 which has an "e" on it and delete it and the two lines above it by typing
`:3,.d`

You buffer should now just contain the original four lines of the poem. You can eliminate the display of line
numbers by typing `:set nonumber`. If you like the look with the line numbers on you can
type `:set number` again.

Replacing Text

There are two main ways to replace text:

1. r = replace the character under the cursor;
2. R = enter text mode in overwrite mode.

Position the cursor over the "s" in "september" on the second line and type `r`, and then `S` to capitalize the month.

Position the cursor over "s" in "standing" on the first line and type `R`, then `walking`, and finally press the `ESC`
key. Then type `x` to delete the extra "g".

Continue and replace "damp" with "dark", "slide" with "slip", and the "L" of "like" by "l". Here is the final version of the poem.

```
I remember walking in the rain,
On a cold and dark September,
```

```
Brown Autumn leaves were falling softly to the ground,
Just like the dreams of a life they slip away.
```

Copying and Pasting Text

There are three operations involved in copying and pasting text:

1. :y which yanks lines into the paste buffer;
2. :pu which inserts the contents of the paste buffer after the current line
3. :nnpu which inserts the contents of the paste buffer after line nn

To copy the first two lines into the paste buffer and then paste them after the third line type `:1,2y` to yank the first
two lines into the paste buffer.

Next type `:3pu` to paste them after line 3. The poem should now look like this:

```
I remember walking in the rain,
On a cold and dark September,
Brown Autumn leaves were falling softly to the ground,
I remember walking in the rain,
On a cold and dark September,
Just like the dreams of a life they slip away.
```

To restore the poem type `:4,5d` to delete lines 4 and 5.

Searching

To search for a pattern in vi you simply type /pattern. To find the next match press n. To find the previous match
press N.

Position the cursor on the first line by typing `:1`.

Now search for the pattern "he" in the poem by typing `/he`

This should take you to the "h" in "the" on the first line.

Press `n`

The cursor should now be over the "h" in "the" on the third line.

Press `n` once more and the cursor should be over the "h" in the "the" on the last line.

Press `N` and the cursor should be back over the "h" of the "the" on the third line.

Search and Replace

To search and replace an old pattern with a new pattern you can type either

1. :s/old_pattern/new_pattern/ to replace the first occurrence of old_pattern with new_pattern
2. :s/old_pattern/new_pattern/g to replace every occurrence of old_pattern with new_pattern.

To replace every occurrence of the substring "re" by "XXX" type `:1,$s/re/XXX/g`

The resulting poem is

```
I XXXmember walking in the rain,
On a cold and dark September,
Brown Autumn leaves weXXX falling softly to the ground,
Just like the dXXXams of a life they slip away.
```

To revert to the original type `:1,$s/XXX/re/g`

Add Your Name

Create a new, empty line at the top of poem, and then insert your name on the new first line.

Saving/Loading Files

There are two ways to use the :w command to write the buffer out to a file:

1. type :w to write the buffer to a file named filename
2. type :w to write the buffer to the current filename.

Save your buffer in a file called rain.txt by typing `:w rain.txt`

Quitting vi

There are four methods of quitting vi:

1. :q = quit vi if the work has been saved since its last change
2. :q! = quit vi and discard unsaved work
3. :wq = save the buffer and then quit vi
4. ZZ = save the buffer and then quit vi (note that this method doesn't need the preceding colon).

Since you have just saved your buffer type `:q` to quit vi.

Customizing vi

vi is highly customizable which is one of the reasons that many people love it.

vim is "improved" vi. It is vi with more extensions added to it.

Let's try setting some options for vim.

Type `vim ~/.vimrc`

Type `i` to enter insert mode and type the following

```
:filetype on
:filetype plugin on
:filetype indent on
:set number
if has("syntax")
  syntax on
endif
```

Here's what the options do

- filetype on: turns on file type detection
- filetype plugin on: enable the plugin for this type of file
- filetype indent on: use the indentation standards for this type of file
- set number: turn on line numbers
- syntax on: turn on syntax highlighting

Press the `ESC` key to enter command mode

Type `:wq` to save the file and exit vim

Open rain.txt again by typing `vim rain.txt`. You should see that the line numbers are on by default.

Type `:q` to quit.

Type `vim`

Type `i` to enter insert mode then type the following. When doing the indentation press the TAB key.

```
def say_hi():
    print('Hello',end=' ')
    print('World')
```

Press the `ESC` key to enter command mode

Type `:w hello1.py` to save the file.
You should now see syntax highlighting for python appear since vim now knows this is a python file.

Type `:q` to quit.

Type `vim hello2.py`

Type `i` to enter insert mode then type the following again. Keep using the TAB key for indentation.

```
def say_hi():
    print('Hello',end=' ')
    print('World')
```

This time in addition to the line numbers and syntax highlighting being on at the beginning you should also notice a difference with the indentation. You should see

- That whenever you press TAB 4 spaces are inserted instead (the python standard)
- When you move to the next line, the line is automatically indented

Press the `ESC` key to enter Command Mode

Type `:wq` to save the file and quit

Personal Musings

Learning how to use vi, vim or any text based editor presents a large difficulty hurdle because

1. It is so different from what you are used to using
2. There are so many options

This difficulty hump leads many, including myself, to avoid learning how to use them.

Those that take the time to learn how they work are able to greatly speed up their efficiency when writing code and just manipulating files in general so it ends up being worth it.
One of my former students, Christina Phan, took the time to learn how to use vim
And she just flies around files. It actually embarrasses me a little to see how good she is at it. Here are her recommendations if you want to get good with vim

- She prefers neoVim
- It does take time to learn how to use vim so don't worry if you don't master it right out of the box
- You learn vim by using vim
- Google is your friend. Google "How do I do X in vim" will help out a lot

And here are some recommended resources from her

- Vim Basics in 8 Minutes
- Mastering the Vim language

- Your problem with Vim is that you don't grok vi
- Seven Habits of Effective Text Editing
- VimRC 2021 (Jan): How to setup your Vim RC
- Why I use Vim in 2022

Submitting

Submit rain.txt as proof that you have complete this tutorial. If you connected to the CSIF remotely use the techniques covered in the Linux Tutorial to transfer rain.txt to your computer so you can submit it.