**Operating Systems**

**Lab 1b- The Vi -Editor**

# Introducing the vi Text Editor

A text editor enables you to create and modify text documents that contain just the letters and characters that you can type in on your keyboard. A word processor, on the other hand, not only allows text editing but also allows changing the presentation of that text by selecting the font size and style. Simple text files are extremely important under UNIX. All UNIX system administration, with few exceptions, is ultimately done by editing text files that customize the system and control how it operates. Shell scripts and C programs are keyed in as text files by a programmer. Even Internet Web pages are simple text files with fonts, colours, and graphics indicated as textual directives to the Web browser.

**vi** is the one text editor that is available on **all** UNIX systems. Don't look for font choices in **vi** because it is not a word processor. Some UNIX users do not like **vi** because you have to memorize all the commands. There are no drop-down menus to help you cut and paste or replace text. There is no help facility inside.

Some types of UNIX systems do provide alternatives to **vi**, such as **emacs**, **pico**, **ed**, **vip**, **sed**, **gedit**, **kedit**, **gless** or **WordPerfect for UNIX**. Some of these are full screen editors in where you can move a cursor around the screen to edit anywhere. However for the purposes of creating and editing files in this course we are going to use the line editor **vi** since **vi** is available on all UNIX computers.

**Aim: Learning to use the vi editor**

Before you begin this session, logon to **aisling** and make sure that you are in the home directory (type **cd** to change to the home directory).

The vi editor is invoked by entering:

**vi *filename***

If you specify an existing file, the system displays the file contents in an editor window. It also displays the file name and the file size in lines and number of characters on the bottom line.

If you specify a file that does not exist, a cleared edit window containing several lines of tildes (~) is displayed. The ~ is used to represent an empty line. The new file name (without the file size) and an indicator that this is a new file are displayed at the bottom of the window.

Note that all of your changes are being made in vi's workspace, not in the file itself. If you don't save your work before you exit vi, all your editing will be lost. When you are finished editing, press Esc and then enter the following:

**:wq**  (to save your work and exit vi), or enter:

**:q!**  (to abort [that is, throw away] your changes and exit vi)

**vi modes**

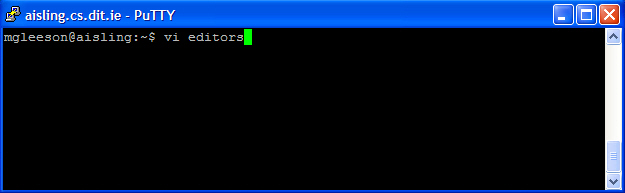
***Command*** mode - When you invoke vi, the edit session starts in command mode. From here you can exit the editor, enter vi commands, move the cursor, delete and alter text or move into another edit mode. How can you tell that you are in command mode? - if you press the ESC key when you are in command mode, you will hear a beep (this is not the case when using Linux). In Linux you will know that you are in command mode if you do not see the word “insert” or a colon at the bottom of your editing window.

***Text-Input* (Insert mode)** mode - Your keyboard behaves like a normal typewriter allowing you to enter text. You will know that you are in text-input mode when you see the word “insert” at the bottom of your editing window. Pressing ESC returns you to command mode.

***Last Line* Mode / *Colon*** mode - While in the command mode, if you enter a “: “ the cursor will move to the last line on the screen and you can enter global commands such as wq or q!. You will need to be in Last Line mode to carry out some complicated operations on your file which cannot be done with simple key strokes. You must hit “enter” after the command to have it executed. You know that you are in last line mode when you see the cursor at the last line. You can go to command mode from last line mode by hitting the enter key.

### Editing a File

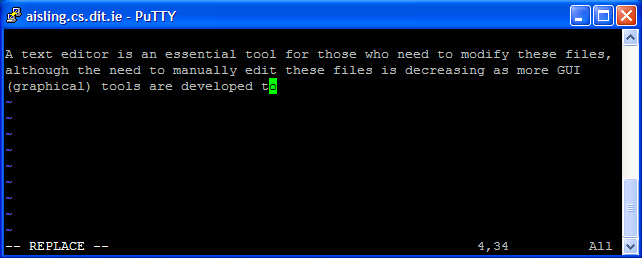
Create and edit a new file called ***editors*** (by typing vi editors) as in the image below



To enter **insert** mode type **i**, then enter the following text into the file and then typing the text: Press “enter” after each line to start the next line.

*A text editor is an essential tool for those who need to modify these files, although the need to manually edit these files is decreasing as more GUI (graphical) tools are developed to maintain Linux system files. Graphical (X-windows based) text editors include gedit, gless, kedit, and kwrite. Ed is a basic/crude line editor, for those who like pain! Emacs is a screen based (but not X-windows) editor. Peter Pico is a screen based (but not X-windows) editor. Sed is a stream editor (filter program), like ed, for use in shell scripts/programs. Vi - visual editor (screen based but not X-windows) editor.*

When you have finished entering text, press the ESC key - this brings you back to command mode. Press colon then type w (**:w**) to save what you have typed.



# Edit Anything with Just 8 vi Commands

There are more than 75 separate letter and punctuation commands to learn in vi. Where should you start? Here are the first 8 vi commands to learn, and they will enable you to edit anything.

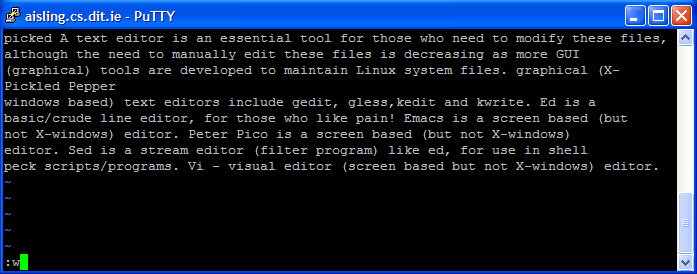
1. **Moving the cursor**

Practise moving the cursor around the file using the **arrow keys** on the keyboard. vi will not let you move to a position on the screen unless there is text in the workspace at that position. You cannot move past the end of a line. You cannot move down from the last line. If you edit an empty file, you will not be able to move at all until you enter some text.

1. **Using i to Insert**

Here is the second vi command to learn. Enter **i** to turn on **Insert** mode. In Insert mode, all the text that you type is entered at the current cursor position into the workspace. Press Esc to turn off Insert mode; this puts you back into vi **Command** mode. You can input multiple lines in one insert by pressing Enter after each line.

* Insert the word “picked” in the first line. Don’t forget to finish entering text by pressing ESC.
* Insert the word “peck” at the start of the last line.
* Insert the line “Pickled Pepper” between lines 3 and 4.
* Save the file



1. **Using x to Delete**

Here is the third vi command to learn. Type **x** to delete **one** character (where the cursor is), similarly 25x will delete 25 characters (from where the cursor is). Use this information to remove all the text you have added after saving your file.

1. **Using a Repeat Count**

You can put a repeat count directly before most vi commands. 50x will delete 50 characters. **vi** will reduce any repeat count larger than the rest of the line so that the rest of the current line is deleted but the following lines are not affected.

1. **Using dd to delete the current line**

Here is the fifth **vi** command to learn. Enter dd to delete the current line. You can be anywhere in the line when you enter dd. As with the repeat count above 25dd will delete 25 lines, including all of the current line.

* Delete an entire line from the file and then insert it again.

1. **Using J to Join Lines**

Here is the sixth vi command to learn. Unlike some editors, vi has no end-of-line character that you can delete to join two lines. The only way to join two lines is to use the J command. 5J will join 5 lines. J, 1J, and 2J all do the same thing: they join two lines. Any leading spaces in the second line will be reduced to a single space.

Practice splitting and joining lines. Splitting is done by pressing the ‘enter’ key.

### Using :wq to Save and Exit

Here is the seventh vi command to learn. So far all of your changes have modified the vi workspace, which is a temporary file. To save your work and exit, enter the following:

**:wq**

The leading colon will cause vi to enter Last Line Mode. The cursor will go down to the last line where the w and q will echo onscreen. Press Enter to complete this command and any other Last Line command.

### Using :q! to Abort Changes

Here is the 8th vi command to learn. If you don't like your changes, you can exit vi without saving them back to the original file by entering the following:

**:q!**

This returns you to the shell prompt and throws away all the editing you have done. Be aware that after you throw away your editing changes by :q!, you cannot change your mind later and get them back again.

This completes the 8 commands you need to use vi. You can now edit anything. Practice with these 8 for a while. Then read the following section to see the next group of vi commands you should master. When you have left the vi editor check the file contents by entering ‘cat editors’

### 

### Learn These Ten vi Commands Next

### Press Ctrl+L or Ctrl+R to Repaint the Screen

Sometimes over a modem line or slow network, garbage may appear on your screen or the cursor will end up on a different line than vi thinks it is. If the cursor is on the wrong line, you will see characters from a different line appear at the cursor when you space to the right. To clear up this problem, make sure you are in command mode (press escape if in doubt) press these two control characters, Ctrl+L Ctrl+R, to repaint your screen to match the workspace.

1. **Undo Changes Using u**

vi remembers the last insert, delete, or modification done. If you want to undo it, enter the **u** command in vi Command mode. You do not have to be near the line where the change was made when issuing the u command. Commands to move the cursor are ignored by the u command and cannot be undone. (Actually many cursor motion commands can be undone by entering back-quote backquote ["].) A second u command will undo the undo command, reinstating the last change again. A third u command will toggle the undo and remove the last change. A fourth u command will reinstate the last change, and so on.

If you join five lines with the 5J command, a u command will separate them into their original lines. A second u will join them again. You can watch the lines magically join together and then separate each time you enter the u command. This is a great way to learn about more complex vi commands.

### Using the u command if something goes wrong

* Sometimes an accidental control character or line noise will cause garbage on the vi screen. Press Ctrl+L Ctrl+R to repaint the current screen and remove the garbage.
* Enter the u command to undo the last change. The cursor will move to the location of that change.
* Enter u several times to watch the change toggle. If the change is an undesirable side-effect of the accident, enter u one or two more times to undo the bad change.

1. **Using A to Append Text at the End of Line**

The capital A command first moves the cursor to the end of the current line. It then enables you to start inserting text after the last character on the line. Press Esc to end Insert mode.

* Append the word “really” to the end of line 4 in your file.

1. **Using :w to Save Your Work Periodically**

If you are making a lot of changes in your vi session, it is prudent to save your work periodically. Enter **:w** to write the current workspace to the disk file. Then if you accidentally hit the wrong keystroke or your system connection dies, you can log back on and your disk file will have the last-saved contents.

1. **Using :w File to Save in a Different File**

You can save your work to a different file than you are editing by entering the following:

**:w** newfile

If I am editing a critical document, for example, I might enter the following:

**:w** doc1

After more editing, I might enter this:

**:w** doc2

In this way, I can save many versions of this one file. This gives me more chances to recover information in case I accidentally delete some text and then save the file without that text.

### Using r to Replace One Character

Enter **r** to replace one character. The next character typed will appear onscreen, replacing the character that was at that cursor position. Then you will be back in Command mode, ready to enter new vi commands.

* Change “Pico” to “Rico” in your file. Using the r command

1. **Using cw to Change Words**

Move the cursor to the start of the desired word or words to change. Type **cw** to change 1 word, if you want to change more than one word, enter the number of words to change (eg 2 words = 2cw). Do not press Enter. The word or words following the cursor will disappear and you will be in input mode. Start typing your replacement text. It can be shorter or longer than the words being changed. It can have fewer or more words. You may use the Enter key to input multiple lines. If the new text is longer, the remainder of the line will move to the right to accommodate it as you type. Press Esc after you have entered all the replacement text. If the new text is shorter than the words being replaced, the remainder of the line will move to the left to fill up the gap automatically.

* Change the word “Peter” to “Thomas” in your file.
* Change the three words “Linux system files” to “Windows files” with 3cw .

1. **Using 1G/G to Go to Start/End of File**

Enter a number and then capital G to go to that line number— for example, 1475G will move the cursor to line 1,475 of the workspace. If the file does not have that many lines, it will give an error beep. Moving quickly to a specific line number in a file can be useful for a C programmer because the compiler often reports the line number in the source where an error occurs.

The two most common uses of the G command are 1G to go to start of file and G to go to the end of the file.

1. **Using :f to See the Filename, Size, and Where You Are**

Enter: f to see the name of the file you are currently editing, what line you are on, and how many lines are in the file.

1. **Restoring the Whole Line Using U**

In vi Command mode, capital U will undo all changes on the current line since you moved to it. After you leave a line, you lose the ability to restore it back to its original state.

**Quick Reference Cheat Sheet**

