This laboratory explores the *vi* text editor.

History and Overview

The *vi* text editor was developed by Bill Joy while he was a graduate student at U.C. Berkeley in the late 1970's. It has survived the test of time. It has lots of useful features, combining WYSIWYG text editing with a sophisticated text processing language. It and its derivatives are still in heavy usage today. The vi editor may be the only editor available on a particular machine, so you should learn it as the common text editor on UNIX-like platforms.

Pronunciation

The word *vi* is always pronounced by saying the letters in the word separately, as if it were an acronym. Do not pronounce *vi* as “vye”.

Setup

Run the following commands:

```
% mkdir lab06
% cd lab06
% touch newFile
```

Starting and Stopping *vi*

Execute the following command to invoke *vi*:

```
% vi newFile
```

This invokes *vi* on the file you specified on the command line. This file is empty, so you will see a mostly blank screen with probably some confusing characters.

Don't press any keys yet or attempt to type yet except as instructed; we'll get to typing soon.

To exit the *vi* editor, press the colon key (shift-semicolon). The colon is the character **:** and the semicolon is the character **;**. Then, press the lower-case **q** key and hit the enter key. You will see this keystroke sequence notated as:

```
:q
```

You will now be back at the shell's command prompt.

Notice the use of the q key to indicate exiting out of a program presenting text to you. This is just like what you did to exit the *info* program or the *more* program. Program designers have been aware of the historical usage of various keys on the keyboard and attempt to use them in similar ways in new programs that perform related functions.

Invoking vi *in Read-Only Mode*

If you want to look at a file with *vi*, but you want to ensure that you don't accidentally change the contents of the file, then invoke *vi* as follows:

```
% view newFile
```

or:

```
% vi -R
```

Exit *vi* as you did previously.

Editing a File in *vi*

Start the *vi* editor again as follows:

```
% vi newFile
```

Your *vi* session starts out in what's called *command mode*. In command mode, the keys on the keyboard are used to immediately perform commands on the text of the file you are editing.

Once in the editor, press the lower-case **a** key.

This is one of the commands that sets the editor in *input mode*.

Now, type the following line (without hitting the enter key at the end of the line):

```
This is the first line of this file.
```

When you are done typing, tell vi to go back to command mode by hitting the escape key. The escape key is marked **Esc**, and appears at the upper left of the keyboard. The escape key is an important key in *vi*; if you get into a state in *vi* where you don't know what to do, you can get back to command mode by pressing the escape key one or more times.

Next, save the changes you made to the file by the following command within *vi*:

```
:w
```

Pressing the colon key while in command mode tells *vi* to provide a command line for your commands within *vi*. This is not the same command line as the one the shell presents to you. It is the command line that *vi* provides for you to run various *vi*-specific commands. These commands are not the same ones you would run on the shell's command line. The prompt for this command line is also the colon, confirming that *vi* has shifted into this command-line-based command mode, where you enter a command and hit enter to run the command, as opposed to the immediate command mode you were in before, where each key you type immediately executes a command.

In exercises in this paper, commands that you should run on *vi*'s command line will be notated as above.

Exit the editor by:

```
:q
```

You'll be back at the command line. Execute the command:

```
% cat newFile
```

You should see what you typed in vi's input mode.

Editing a Document

(I decided that having you copy and paste commands from this document wastes too much class time, so I'll provide scripts that you'll run to set up your environment. You'll have to copy and paste one line from this document per setup, instead of potentially many lines as in the past.)

Set up for the next part of this lab by running the following command:

```
% ~bmcdonald/class/lab06.sh
```

Then, start *vi* as follows:

```
% vi fullFile
```

Next, run the following *vi* command on *vi*'s command line:

```
:set showmode
```

This command tells *vi* to show you what mode you are in. After you run the above command, you should see a “command” indicator on the right side of the status line. You don't have to do this, but at this stage it may be useful to you to know what mode you are in.

Moving the Cursor

The cursor is the point in the file you are editing where editing commands take effect in command mode. This is the same role the cursor plays in graphical text editors like Microsoft Word (if you've heard the term insertion point applied to the cursor in graphical editors, that is the equivalent of the cursor in *vi*). You'll need to know how to move the cursor around in order to make the changes you want to your file.

Small Moves

Fortunately, some cursor movement operations are just like they are in other editors you are familiar with. The down/up/left/right arrow keys work just like you'd expect. There are also the following keys for moving the cursor around the file.

- h – move the cursor one space to the left
- j – move the cursor one row down
- k – move the cursor one row up
- l - move the cursor one space to the right
- spacebar – move the cursor one space to the right
- backspace – move the cursor one space to the left (this doesn't always work by default on all systems)

These keys are the equivalent of the arrow keys. The handy thing about them is that you don't have to take your hands off standard typing position in order to use them, whereas you must move your hands with the arrow keys (not to mention the mouse). The *vi* editor was developed before computer mice

were put into general use, and it saves users time by allowing them to keep their hands in the standard typing position.

In addition, there are the following cursor movement commands:

`w`, `W` – move the cursor to the beginning of the next word

`e`, `E` – move the cursor to the end of the current or next word

`b`, `B` – move the cursor to the beginning of the current or previous word

`(` – move the cursor to the beginning of the current or previous sentence

`)` – move the cursor to the beginning of the next sentence

`0` – move the cursor to the beginning (left side) of the current line

`$` – move the cursor to the end (right side) of the current line

`+` – move the cursor to the beginning of the next line

`-` – move the cursor to the beginning of the previous line

Try out these commands now.

Big Moves

You don't have to hold down the j key to navigate forward through the file. You can use other keys as follows:

control -F – move one screenful Forward in the file

control-B – move one screenful Backward in the file

control-D – move one-half screenful Down (forward) in the file

control-U – move one-half screenful Up (backward) in the file

H – move to the line at the top of the screen

M – move to the line in the middle of the screen

L – move to the line at the bottom of the screen

Try out these commands now.

More Complex Commands

On some commands in command mode, if you include a number in them, they will use that number to increase the action they take for the command. For example, if you type the following in command mode:

```
2w
```

the cursor moves forward by two words. All of the commands in the Small Moves and Big Moves sections above can be modified in this way. Try this out now with these commands.

By Line Number

The following commands relate to moving the cursor to specific line numbers:

`G` – move to the last line in the file

`#G` – move the cursor to the line specified by the # (replace the # with the number of the line to which you want to go)

You can check what line you are on (as well as the status of the file you are editing) by hitting control-G.

Try this out now.

Search

Press the slash key (`/`). This provides you with another command line. This one is for entering a set of characters to search for (called a *search string*) in the document. Searching is case-sensitive. Now enter:

```
on
```

and hit Enter. The cursor positions to the beginning of the search string found, or does not move if the search string is not found in the document.

Once you have entered a search string, pressing the `n` key will go to the next instance of that string, and the `N` key will go to the previous instance of that string.

Also, the slash key initiates searches forward in the document, where pressing the question mark key initiates searches backward in the document. Searches will *wrap* in either direction, for example, if a forward search encounters the end of the file, the search will resume at the beginning of the file.

Adding Text

The following commands allow you to add text to your documents:

`a` – append after the cursor

`A` – append after the end of the current line

`i` – insert before the cursor

`I` – insert at the beginning of the current line

`o` – open a new line below the cursor

`O` – open a new line above the cursor

All of these commands put the editor in input mode. After typing what you want, press the escape key to return to command mode.

Try these commands now.

Changing and Deleting Text

Various forms of the following commands allow you to change or delete text:

`c` – change text

`d` – delete text

`r` – replace the character at the cursor with the character typed immediately after the `r`

`s` – substitute text for the single character at the cursor (puts editor in input mode – press escape when done to return to command mode)

`x` – deletes the character at the cursor

In addition, there is:

`u` – undoes the last command-mode change

`J` – joins the current line and the next line

Some of these commands require modifiers to be useful to you:

`dd` – deletes the current line

`dw` – deletes the current or next word

`d2w` – deletes the next two words

Pressing `d` then pressing the spacebar will delete the character under the cursor; this has the same effect as the `x` command.

`cw`, `cW` – puts the editor in input mode to change the text of the current/next word (hit escape when done)

`c2w` – change the following two words (hit escape when done typing changes)

Try out these commands now.

Cut/Copy/Paste

Position the cursor on a non-blank line. Press:

```
yyp
```

The `yy` command yanks the current line into the paste buffer (equivalent to the clipboard). The `p` command puts the contents of the paste buffer before the cursor position. You've just done a copy and paste operation.

Text deleted with the `d` command goes into the paste buffer, which gives you the ability to cut and paste.

The `P` command puts the contents of the paste buffer after the cursor position.

Try out these commands now.

Saving

Hit the following to save your work:

`:w`

and press Enter.

Exiting

Hit the following to quit the editor after saving your work:

`:q`

and press Enter. You can combine writing and exiting by typing:

`:wq`

and pressing Enter.

If you want to exit no matter what, even if you lose unsaved changes, type:

`:q!`

and press Enter.

Working With Multiple Files

The *vi* editor lets you work with only one file at a time. You can load a different file to edit in *vi* using the following commands:

`:e filename`

The e command on *vi*'s command line tells *vi* to load the file you specify in the *filename* argument.

Try this with:

`:e editFile`

You should now see the contents of the file *editFile* in *vi*.

To get back to editing *fullFile*, you can use the following command:

`:e #`

This tells *vi* that you want to edit the file you had previously loaded into *vi*.

Try this now.

If you make some big mistakes editing, and you can't recover them using the undo command, and you haven't yet saved your work, you can reload the most recent version of the current file by:

`:e!`

The exclamation point indicates that you want to throw away any changes you have made.

You can read in the contents of another file using the command:

`:r filename`

Try this by running the command:

```
:r readFile
```

Escaping to the Shell

You can run shell commands from within *vi* using the following command:

```
:! command
```

Try the following:

```
:! ls -l
```

Press Enter to resume in *vi*'s command mode.

Escaping the Command-line Command Mode

If you get stuck in vi's command-line (with the colon prompt), and you want to return to command mode without running any command line commands, type:

```
vi
```

and hit Enter to return to command mode. In some versions the escape key will also get you back to command mode.