9.1 Introduction

The premier text editor for Linux and UNIX is a program called `vi`. While there are numerous editors available for Linux that range from the tiny editor `nano` to massive `emacs` editor, there are several advantages to the `vi` editor:

- The `vi` editor is available on every Linux distribution in the world. This is not true of any other editor.
- The `vi` editor can be executed both in a CLI interface and a GUI interface. While graphical editors, like `gedit` from the Gnome desktop environment or `kedit` from K desktop environment, are easier to use, they require a GUI, which servers won't always have running.
- While new features have been added to the `vi` editor, the core functions have been around for decades. This means if someone learned the `vi`editor in the 1970s, they could use a modern version without any problem. While that seems trivial, it may not seem to so trivial twenty years from now.

Consider This

The correct way to pronounce "the vi editor" is "the vee-eye editor". The letters vi stand for "**vi**sual", but it was never pronounced "vi" by the developers, but rather the letter "v" followed by the letter "i".

The original `vi` editor was written by Bill Joy, the co-founder of Sun Microsystems. Since `vi` is part of the Single UNIX Specification (SUS), it is required that conforming UNIX-based systems have it. Since the Linux Standards Base (LSB) mirrors the requirements of SUS, Linux systems that conform to LSB must also include the `vi` editor.

In reality, most Linux systems don't include the original `vi`, but an improved version of it known as `vim`, for **vi im**proved. This fact may be hidden by most Linux distributions. On some the `vi` file will link to `vim`, while on others an alias exists that will execute `vim` when the `vi` command is run:

```
[sysadmin@localhost ~]$ which vi
alias vi='vim'
        /usr/bin/vim
```

For the most part, `vim` works just like `vi`, but has additional features. For the topics that will be covered in this course, either `vi` or `vim` will work.

To get started using `vi`, simply type the command followed by the pathname to the file to edit or create:

```
sysadmin@localhost:~$ vi newfile
```

9.2 Command Mode Movement

There are three modes used in `vi`: command mode, insert mode, and ex mode. Initially, the program starts in command mode. Command mode is used to type commands, such as those used to move around a document, manipulate text, and access the other two modes. To return to command mode at any time, press the **ESC** key.

Once some text has been added into a document, to perform actions like moving the cursor, the **ESC** key needs to be pressed first to return to command mode. This seems like a lot of work, but remember that `vi` works in a terminal environment where a mouse is useless.

Movement commands in `vi` have two aspects, a motion and an optional number prefix, which indicates how many times to repeat that motion. The general format is as follows:

```
[count] motion
```

The following table summarizes the motion keys available:

Motion	Result
`h`	Left one character
`j`	Down one line
`k`	Up one line
`l`	Right one character
`w`	One word forward
`b`	One word back
`^`	Beginning of line
`$`	End of the line

Note: Since the upgrade to `vim` it is also possible to use the arrow keys `←↓↑→` instead of `hjkl` respectively.

These motions can be prefixed with a number to indicate how many times to perform the movement. For example, `5h` would move the cursor five characters to the left,`3w` would move the cursor three words to the right.

To move the cursor to a specific line number, type that line number followed by the `G` character. For example, to get to the fifth line of the file type `5G`. `1G` or `gg` can be used to go to the first line of the file, while a lone `G` will take you to the last line. To find out which line the cursor is currently on, use **CTRL-G**.

9.3 Command Mode Actions

The standard convention for editing content with word processors is to use copy, cut, and paste. The `vi` program has none of these, instead `vi` uses the following three commands:

Standard	Vi	Meaning
cut	`d`	delete
copy	`y`	yank

Standard	Vi	Meaning
paste	`P` | `p`	put

The motions learned from the previous page are used to specify where the action is to take place, always beginning with the present cursor location. Either of the following general formats for action commands is acceptable:

```
action [count] motion
[count] action motion
```

Delete

Delete removes the indicated text from the page and saves it into the buffer, the buffer being the equivalent of the "clipboard" used in Windows or Mac OSX. The following table provides some common usage examples:

Action	Result
`dd`	Delete current line
`3dd`	Delete the next three lines
`dw`	Delete the current word
`d3w`	Delete the next three words
`d4h`	Delete four characters to the left

Change

Change is very similar to delete, the text is removed and saved into the buffer, however the program is switched to insert mode to allow immediate changes to the text. The following table provides some common usage examples:

Action	Result
`cc`	Change current line
`cw`	Change current word

Action	Result
`c3w`	Change the next three words
`c5h`	Change five characters to the left

Yank

Yank places content into the buffer without deleting it. The following table provides some common usage examples:

Action	Result
`yy`	Yank current line
`3yy`	Yank the next three lines
`yw`	Yank the current word
`y$`	Yank to the end of the line

Put

Put places the text saved in the buffer either before or after the cursor position. Notice that these are the only two options, put does not use the motions like the previous action commands.

Action	Result
`p`	Put (paste) after cursor
`P`	Put before cursor

Searching in vi

Another standard function that word processors offer is find. Often, people use **CTRL+F** or look under the edit menu. The `vi` program uses search. Search is more powerful than find because it supports both literal text patterns and regular expressions.

To search forward from the current position of the cursor, use the `/` to start the search, type a search term, and then press the **Enter** key to begin the search. The cursor will move to the first match that is found.

To proceed to the next match using the same pattern, press the `n` key. To go back to a previous match, press the `N` key. If the end or the beginning of the document is reached, it will automatically wrap around to the other side of the document.

To start searching backwards from the cursor position, start by typing `?`, then type the pattern to search for matches and press the **Enter** key.

9.4 Insert Mode

Insert mode is used to add text to the document. There a few ways to enter insert mode from command mode, each differing by where the text insertion will begin. The following table covers the most common:

Input	Purpose
`a`	Enter insert mode right after the cursor
`A`	Enter insert mode at the end of the line
`i`	Enter insert mode right before the cursor
`I`	Enter insert mode at the beginning of the line
`o`	Enter insert mode on a blank line after the cursor
`O`	Enter insert mode on a blank line before the cursor

9.5 Ex Mode

Originally, the `vi` editor was called the `ex` editor. The name `vi` was the abbreviation of the **visual** command in the `ex` editor that switched the editor to "visual" mode.

In the original normal mode, the `ex` editor only allowed users to see and modify one line at a time. In the visual mode, users could see as much of the document that will fit on the screen. Since most users preferred the visual mode to the line editing mode, the `ex` program file was linked to a `vi` file, so that users could start `ex` directly in visual mode when they ran the `vi` link.

Eventually, the actual program file was renamed `vi` and the `ex` editor became a link that pointed the `vi` editor.

When the ex mode of the `vi` editor is being used, it is possible to view or change settings, as well as carry out file-related commands like opening, saving or aborting changes to a file. In order to get to the ex mode, type a `:` character in command mode. The following table lists some common actions performed in ex mode:

Input	Purpose

Input	Purpose
`:w`	Write the current file to the filesystem
`:w filename`	Save a copy of the current file as filename
`:w!`	Force writing to the current file
`:1`	Go to line number 1 or whatever number is given
`:e filename`	Open filename
`:q`	Quit if no changes made to file
`:q!`	Quit without saving changes to file

A quick analysis of the table above reveals if an exclamation mark `!` is added to a command, then attempts to force the operation. For example, imagine you make changes to a file in the `vi` editor and then try to quit with `:q`, only to discover that the "quit" command fails. The `vi` editor doesn't want to quit without saving the changes you made to a file, but you can force it to quit with the ex command `:q!`.

Consider This

While it may seem impossible, the `vi` editor can save changes to a read-only file. The command `:w!` will try to write to a file, even if it is read-only, by attempting to change the permissions on the file, perform the write to the file and then change the permissions back to what they were originally.

This means that the root user can make changes to almost any file in the `vi` editor, regardless of the permissions on the file. However, ordinary users will only be able to force writing to the files that they own. Using `vi` doesn't change the fact that regular users can't modify the permissions on file that they do not own.

Although the ex mode offers several ways to save and quit, there's also `ZZ` that is available in command mode; this is the equivalent of `:wq`. There are many more overlapping functions between ex mode and command mode. For example, ex mode can be used to navigate to any line in the document by typing `:` followed by the line number, while the `G` can be used in command mode as previously demonstrated.

Chapter 9: The vi Editor

This chapter will cover the following exam objectives:

103.8: Perform basic file editing operations using vi

Weight: 3

Description: Candidates should be able to edit text files using vi. This objective includes vi navigation, basic vi modes, inserting, editing, deleting, copying and finding text.

Key Knowledge Areas:

- Navigate a document using vi
  Section 9.2
- Use basic vi modes
  Section 9.2
- Insert, edit, delete, copy and find text
  Section 9.4

Chapter 9: The vi Editor

/, ?

This is used to search for text while in command mode. the / is used to start searching. Enter a key term and press enter to begin searching the file for the text entered. If the user would like to search backwards in the document, a ? can be used instead of the /.
Section 9.3

ZZ, :w!, :q!, :e!

These keys are used to exit the vi editor from command mode. ZZ is used to save and quit the file. It must be done for each file. :e! is used to restor the original file allow the user to start over. :w! will force the writing of the current file. :q! will exit the editor without saving changes to the current file.
Section 9.5

c, d, p, y, dd, yy

These are used to cut, copy, replace and paste text when in command mode. c is used to change a line from the the current cursor location to the end of the line with whatever the user types. d is used to cut one alphabetic word, where as dd is used to cut an entire line of text. y is used to copy one one alphabetic word, where as yy is used to copy and entire line at a time. If a number preceeds either dd or yy, this will copy that number of lines. For example if 3dd is typed this will cut 3 lines at a time.
Section 9.3

h, j, k, l

These keys are used for basic cursor movement in vi when in command mode. h moves left one character, j moves down one line, k moves up one line, and l moves right one character.
Section 9.2

i, o, a

i, o, and a are used to enter insert mode from command mode. i will allow a user to start inserting text at the current location of the cursor. o will allow a user to start inserting text a line below the current location of the cursor, and a will allow a user to insert text one postion after the current location of the cursor.
Section 9.4

vi

A screen-oriented text editor originally created for Unix operating systems. vi is also known as a modal editor in which the user must switch modes to create, edit, and search text in a file.
Section 9.1 | Section 9.2 | Section 9.3 | Section 9.5