

# Using the vim Editor

# Objectives

After completing this lesson, you should be able to:

- Access the `vim` editor
- Modify files with the `vim` editor



# Agenda

- Accessing the `vim` Editor
- Modifying Files with the `vim` Editor



# vim Editor: Introduction

- The `vim` editor is an interactive command-line editor that you can use to create and modify text files.
  - The `vim` editor is also the only text editor that you can use to edit certain system files without changing the permissions associated with the files.
- In Oracle Solaris and Oracle Linux, `vim` (`vi` improved) is the default editor.
  - The `vim` editor is an enhanced version of the `vi` editor and is accessed via an alias “`vi`” in Oracle Linux.
  - In Oracle Solaris, `vi` is a symbolic link that links to `vim`.

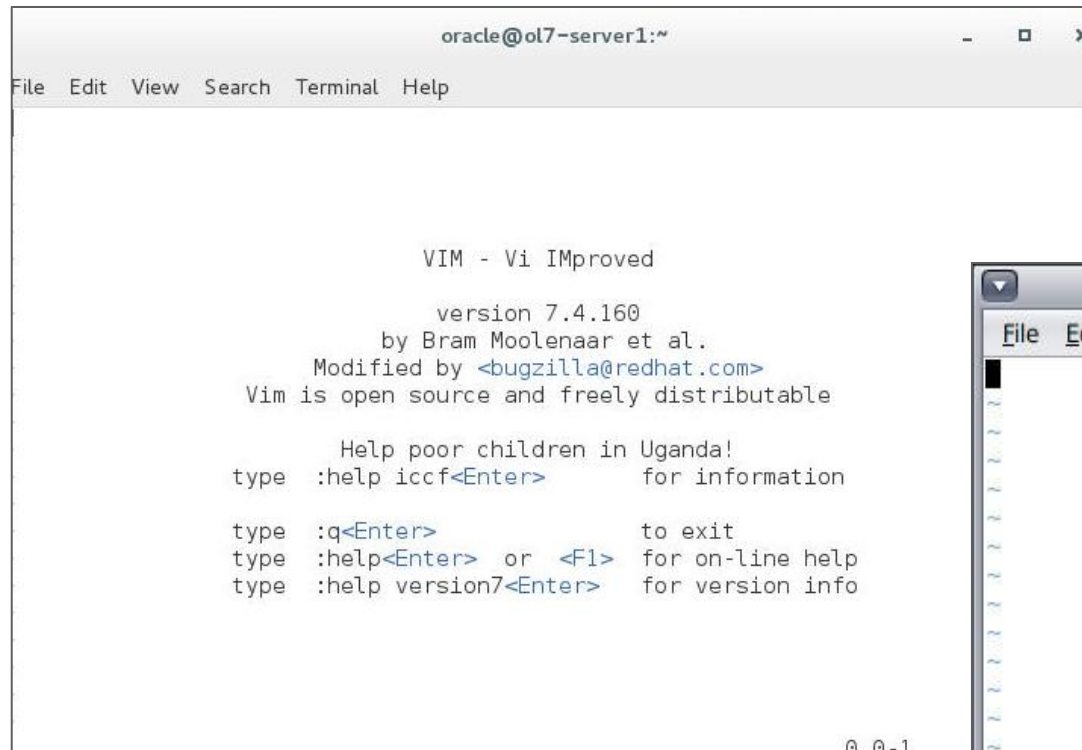
# Accessing the `vim` Editor

- To create, edit, and view files in the `vim` editor, use the `vi` command.
- The `vi` command includes the following three syntaxes:

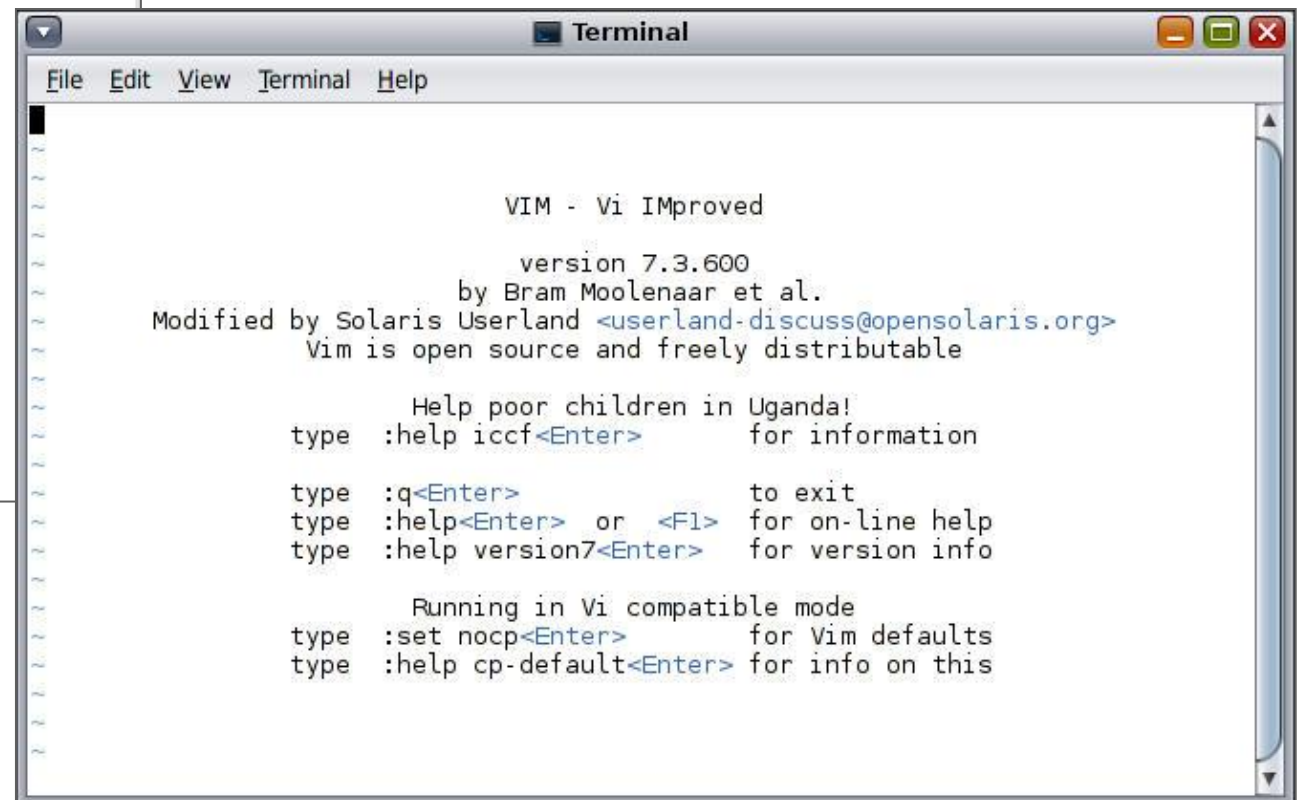
```
$ vi [options] filename  
$ vi  
$ vi filename
```

- For more information about `vim` command options, see the `vim` man page.

# vim Editor: Overview



```
oracle@ol7-server1:~  
File Edit View Search Terminal Help  
  
VIM - Vi IMproved  
version 7.4.160  
by Bram Moolenaar et al.  
Modified by <bugzilla@redhat.com>  
Vim is open source and freely distributable  
  
Help poor children in Uganda!  
type :help iccf<Enter> for information  
  
type :q<Enter> to exit  
type :help<Enter> or <F1> for on-line help  
type :help version7<Enter> for version info  
  
0.0-1
```



```
Terminal  
File Edit View Terminal Help  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
  
VIM - Vi IMproved  
version 7.3.600  
by Solaris Userland et al.  
Modified by Solaris Userland <userland-discuss@opensolaris.org>  
Vim is open source and freely distributable  
  
Help poor children in Uganda!  
type :help iccf<Enter> for information  
  
type :q<Enter> to exit  
type :help<Enter> or <F1> for on-line help  
type :help version7<Enter> for version info  
  
Running in Vi compatible mode  
type :set nocp<Enter> for Vim defaults  
type :help cp-default<Enter> for info on this
```

# vim Editor Modes

- The `vim` editor is a modal editor and provides six basic modes of operation:
  1. Command mode (normal): Typically the default mode when `vim` starts
  2. Insert (and replace) mode: Used when adding new text
  3. Visual mode: Changes how text is highlighted
  4. Select mode: Is the default mode for MS-Windows installations
  5. Command-line mode: Allows you enter `ex` commands
  6. Ex mode: Is optimized for batch processing
- The command and insert modes are the two most common modes and will be the ones used in this course.

**Note:** Originally the `vi` editor was the **visual interface** (`vi`) to the `ex` editor, which in turn is an **extended** version of the `ed` editor.

# Switching Between the Two Most Common Modes

- The default mode for the `vim` editor is **command mode**.
- To switch to **insert mode**, press `i`, `o`, `a` or `R`.
- To return to **command mode**, press the **Esc** key.
- In command mode, enter the `:wq` command that writes (saves changes to the file), quits the `vim` editor, and returns to the shell prompt.



# Agenda

- Accessing the `vim` Editor
- Modifying Files with the `vim` Editor



# Viewing Files in Read-Only Mode

- The `view` command enables you to view files in **read-only** mode.

```
$ view filename  
or  
$ vim -R filename
```

- The `view` command invokes the `vim` editor as read-only, which means you cannot save changes to the file.
- To view the `dante` file in read-only mode, enter the following command:

```
$ view dante
```

- The `dante` file appears. Enter the `:q` command to quit the `vim` editor, and return to the shell prompt.

# Moving the Cursor Within the `vim` Editor

Key Sequence	Cursor Movement
<b>H, left arrow, or Backspace</b>	Left one character
<b>J or down arrow</b>	Down one line
<b>K or up arrow</b>	Up one line
<b>L, right arrow, or space bar</b>	Right (forward) one character
<b>W</b>	Forward one word
<b>B</b>	Back one word
<b>E</b>	To the end of the current word
<b>\$</b>	To the end of the line
<b>0 (zero)</b>	To the beginning of the line
<b>^</b>	To the first non–white space character on the line

# Moving the Cursor Within the `vim` Editor

Key Sequence	Cursor Movement
<b>Return / Enter</b>	Down to the beginning of the next line
<b>G</b>	To the last line of the file
<b>1G</b>	To the first line of the file
<b>:n</b>	To Line n
<b>nG</b>	To Line n
<b>Control + F</b>	Pages forward one screen
<b>Control + D</b>	Scrolls down one half screen
<b>Control + B</b>	Pages back one screen
<b>Control + U</b>	Scrolls up one half screen
<b>Control + L</b>	Refreshes the screen
<b>Control + G</b>	Displays current buffer information

# Inserting and Appending Text

The table describes the commands to insert and append text to a new or existing file by using the `vim` editor.

Command	Function
<b>a</b>	Appends text after the cursor
<b>A</b>	Appends text at the end of the line
<b>i</b>	Inserts text before the cursor
<b>I</b>	Inserts text at the beginning of the line
<b>o</b>	Opens a new line below the cursor
<b>O</b>	Opens a new line above the cursor
<b>:r <i>filename</i></b>	Reads and inserts the contents of another file into the current file below the line containing the cursor

# Text-Deletion Commands

The table shows commands that delete text in the `vim` editor.

Command	Function
<b>R</b>	Overwrites or replaces characters on the line at and to the right of the cursor. To terminate this operation, press Escape.
<b>C</b>	Changes or overwrites characters from the cursor to the end of the line
<b>s</b>	Substitutes a string for a character at the cursor
<b>x</b>	Deletes a character at the cursor
<b>nx</b>	Deletes n characters beginning at the cursor
<b>dw</b>	Deletes a word or part of the word to the right of the cursor
<b>dd</b>	Deletes the line containing the cursor
<b>ndd</b>	Deletes n lines beginning with the line containing the cursor
<b>D</b>	Deletes the line from the cursor to the right end of the line
<b>:n,nd</b>	Deletes lines n–n. For example, <code>:5,10d</code> deletes lines 5–10.

# Edit Commands

The table describes the commands to change text, undo a change, and repeat an edit function in the `vim` editor.

Command	Function
<b><code>cw</code></b>	Changes or overwrites characters at the cursor location to the end of that word
<b><code>r</code></b>	Replaces the character at the cursor with one other character
<b><code>J</code></b>	Joins the current line and the line below
<b><code>xp</code></b>	Transposes the character at the cursor and the character to the right of the cursor
<b><code>~</code></b>	Changes letter casing to uppercase or lowercase, at the cursor
<b><code>u</code></b>	Undoes the previous command
<b><code>U</code></b>	Undoes all changes to the current line
<b><code>.</code></b>	Repeats the previous command

# Quiz



In which `vim` mode are commands normally initiated?

- a. ed mode
- b. ex mode
- c. Command mode
- d. Input mode





# Searching for and Substituting (Replacing) Text Within a File

The table shows the commands that search for and can substitute (replace) text using the `vim` editor.

Command	Function
<code>/pattern</code>	Searches forward for the pattern/string ( <i>regex_pattern</i> ).
<code>?pattern</code>	Searches backward for the pattern/string ( <i>regex_pattern</i> ).
<code>n</code>	Searches for the next occurrence of the pattern. Use this command after searching for a pattern.
<code>N</code>	Searches for the previous occurrence of the pattern. Use this command after searching for a pattern.
<code>: [%]s/oldstring/newstring /[g]</code>	Searches for the old string and substitutes (replaces) it with the new string globally. The “%” symbol searches the whole file and the “g” replaces every occurrence of oldstring with newstring globally.

# Copy-and-Paste Commands

The table shows the commands that copy (`yank`) and paste (`put`) text in the `vim` editor.

Command	Function
<code>yy</code>	Yanks a copy of a line containing the cursor
<code>n yy</code>	Yanks a copy of <code>n</code> lines, beginning with the line containing the cursor
<code>p</code>	Puts yanked ( <code>yy</code> ) or deleted ( <code>dd</code> ) text after the line containing the cursor
<code>P</code>	Puts yanked ( <code>yy</code> ) or deleted ( <code>dd</code> ) text before the line containing the cursor
<code>:n,n co n</code>	Copies lines <code>n–n</code> and puts them after line <code>n</code> . For example, <code>:1,3 co 5</code> copies lines 1–3 and puts them after line 5.
<code>:n,n m n</code>	Moves lines <code>n–n</code> to line <code>n</code> . For example, <code>:4,6 m 8</code> moves lines 4–6 to after line 8.

# Save and Quit Commands

The table describes the commands that save (write) the text file, quit the `vim` editor, and return to the shell prompt.

Command	Function
<code>:w</code>	Saves the file with changes by writing to the disk
<code>:w <i>new_filename</i></code>	Writes the contents of the buffer to <code>new_filename</code>
<code>:wq</code>	Saves the file with changes and quits the <code>vi</code> editor
<code>:x</code>	Saves the file with changes and quits the <code>vi</code> editor
<code>ZZ</code>	Saves the file with changes and quits the <code>vi</code> editor
<code>:q!</code>	Quits without saving changes
<code>ZQ</code>	Quits without saving changes

# Session Customization

- You can customize a `vim` session by setting options for the session.
- In VIM 7.*n*, there are more than 295 options.
- When you set an option, you enable a feature that is not activated by default.
- You can use the `:set` command to enable and disable options within `vim`.
- Two of the many `set` command options include displaying line numbers and invisible characters, such as the Tab and the end-of-line (^M) characters.

# Session Customization Commands

Command	Function
<code>:set nu</code>	Shows line numbers
<code>:set nonu</code>	Hides line numbers
<code>:set ic</code>	Instructs searches to ignore case
<code>:set noic</code>	Instructs searches to be case-sensitive
<code>:set list</code>	Displays invisible characters, such as <code>^I</code> for a Tab and <code>\$</code> for end-of-line characters
<code>:set nolist</code>	Turns off the display of invisible characters
<code>:set showmode</code>	Displays the current mode of operation
<code>:set noshowmode</code>	Turns off the mode of operation display
<code>:set</code>	Displays all the <code>vim</code> variables that are set
<code>:set all</code>	Displays all <code>vim</code> variables and their current values

# Quiz



Which three commands help save changes in your file and quit the `vim` editor?

a. `:wq`

b. `:wq!`

c. `ZZ`

d. `:q!`

e. `:w`



# Quiz



Which of the following commands searches backwards for the pattern?

- a. ?pattern
- b. /pattern
- c. !pattern
- d. ~pattern



# Summary

In this lesson, you should have learned how to:

- Access the `vim` editor
- Modify files with the `vim` editor





# Practice 4: Overview

- 4-1: Using the `vim` Editor

