RHSA1
Red Hat System Administration I
Day 3

Day 3 Contents

- Vi text editor.
- Initialization Files.
- Environment Variables.

Red Hat

The Vi Text Editor

- Vi editor (visual editor) is the default editor for Unix and Linux operating system.
- Vi is used to manage file content.
- Vi is an interactive editor that you can use to create and modify test files.
- Usually the only editor available in emergency mode.
- It is used when the desktop environment window system is not available.

The Vi Text Editor

- vi in Linux is usually vim (vi improved):
  - Syntax highlighting.
  - Arrow keys, Del, BS work in insert mode.
  - Mouse support.
- An advantages of this editor is that we can manipulate text without using a mouse. We can only need the keyboard.

Vi Operations

VI has three basic modes:

- **Command mode:**
  - **Default mode.**
  - **Perform commands to delete, copy, ...**
- **Edit (insert) mode:**
  - **Enter text into the file.**
- **Last line mode:**
  - **Advanced editing commands.**
  - **To access it, enter a colon (:) while in the command mode.**

Vi Operations

- The syntax of vi command:

  vi

  vi filename

  vi options filename

- To recover a file

  vi -r filename

- Viewing files in Read-only mode:

  view filename

  – Perform the :q command exit.

Vi Operations

- Inserting and appending text:
  - i Inserts text before the cursor.
  - o Opens a new blank line below the cursor.
  - a Appends text after the cursor.
  - A append text at the end of the line.
  - I insert text at the beginning of the line.
  - O opens a new line above the cursor.
- After editing Press esc to enter command mode.

Manipulating Files Within Vi

- Inserting and appending text:
  - h, left arrow, or backspace: left one character.
  - j or down arrow: down one line.
  - k or up arrow: up one line.
  - l, right arrow or space: right one character.

Manipulating Files Within Vi

- Moving the cursor within the vi (cont.):
  - w forward one word.
  - b back one word.
  - e to the end of the current word.
  - 0 to the beginning of the line.
  - Enter: down to the beginning of the next line.

Manipulating Files Within Vi

- Moving the cursor within the vi (cont.):
  - G Goes to the last line of the file.
  - nG Goes to Line n.
  - :n Goes to Line n.
  - Control-F Pages forward one screen.
  - Control-B Pages back one screen.
  - Control-L refresh the screen.

Manipulating Files Within Vi

- Substitute and delete text:
  - s Substitutes a string for a character at the cursor.
  - x Deletes a character at the cursor.
  - dw Deletes a word or part of the word to the right of the cursor.
  - dd Deletes the line containing the cursor.
  - D Deletes the line from the cursor to the right end of the line.
  - n,nd Deletes Lines n through n.

Manipulating Files Within Vi

- Search and replace:
  - /string Searches forward for the string.
  - ?string Searches backward for the string.
  - n Searches for the next occurrence of the string.
  - N Searches for the previous occurrence of the string.
  - %s/old/new/g Searches for the old string and replaces it with the new string globally.

Manipulating Files Within Vi

- Copy and paste:
  - yy Yank a copy of a line.
  - p Put yanked text under the line containing the cursor.
  - P Put yanked text before the line containing the cursor.
  - n,n co n Copy Lines n though n and puts them after Line n.
  - n,n m n Move Lines n through n to Line n.

Manipulating Files Within Vi

- Save and quit:
  - :w save the file.
  - :w new_file save as new file.
  - :wq, :x, ZZ save and quit.
  - :q! quit without saving.

Manipulating Files Within Vi

- Customizing vi session:
  - :set nu, :set nonu show and hide line numbers.
  - :set ic, :set noic ignore or be case sensitive.
  - :set showmode, :set noshowmode display or turn off mode.

Editing Files With Gedit

- The gedit text editor is a graphical tool for editing text files.
- The gedit window is launched by selecting:

  Search menu → gedit

Environment Variables

- $HOME
  - Complete path of the user home directory.

    Example:

    – mkdir $HOME/file1
- $PWD
  - The user current working directory.
- $SHELL
  - Path name of the login shell.

Environment Variables

- $USER
  - Currently logged in user.
- $HOSTNAME
  - Name of the computer.

Environment Variables

- $PATH
  - A colon-separated list of directories used by the shell to look for executable program names.

    Example:

    – echo $PATH

    /home/fatma/.local/bin:/home/fatma/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin

Viewing Variable Contents

- The shell assumes whatever follows the dollar sign ($) in the command line is a variable and substitutes its value.
  - echo $HOME

    /home/user1
- To display the current Environment variables with values use the env or printenv command.

Creating a User Environment

- When a user logs in, an environment is created.
- The environment consists of some variables that determine how the user is working.
- One such variable is $PATH, which defines a list of directories that should be searched when a user types a command.

Creating a User Environment

- To construct the user environment, a few file play a role:
  - Global initialization file: /etc/profile and /etc/bashrc
  - Initialization file: ~/.profile
  - Startup files: ~/.bashrc
- When logging in, the files are read in this order, and variables and other settings that are defined in these files are applied.
- If a variable or setting occurs in more than one file, the last one wins.

Creating a User Environment

- /etc/profile: Used for default settings for all users when starting a login shell.
- /etc/bashrc: Used to define defaults for all users when starting a subshell.
- ~/.profile: Specific setting for one user applied when starting a login shell.
- ~/.bashrc:Specific setting for one user applied when starting a subshell.

Command Alias

- The purpose of the linux shell is to provide an environment in which commands can be executed.
- The shell takes care of interpreting the command that a user has entered correctly.
- To do this, the shell makes a distinction between three kinds of commands.

  - Aliases.
  - Internal commands.
  - External commands.

Command Alias

- Alias is a command that a user can define as needed.
- alias newcommand = ‘oldcommand’ alias ll=’ls -l’
- Alias are executed before anything else.
- An internal command is a command that is a part of the shell itself and, as such, doesn’t have to be loaded from disk separately.
- An external command is a command that exists as an executable file on the disk of the computer.

Command Alias

- To find out whether a command is a Bash internal or an executable file on disk, you can use the type command.
- To find out which exact command the shell will be using, you can use the which command.
- Type alias at the terminal to see all set aliases.
- To remove aliases, you can use unalias command.

Command History

- Bash stores a history of commands you have entered so that you can recall them later.
- The history is stored in the user's home directory and is called .bash_history by default.
- You can recall commands by pressing the up arrow key.

!!: Repeats the last command.

!string: Repeats the last command that started with string.

!n: Repeats a command by its number in history output.

!-n:Repeats a command entered n commands back.