RHSA1

Red Hat System Administration I

Information Technology Institute

COURSE MATERIALS

You can access the course materials via this link

https://goo.gl/ezCT7j

DAY 3 CONTENTS

- Vi text editor
- Initialization Files.
- Environment Variables

THE VI TEXT EDITOR

- Default editor in all UNIX operating systems.
- Usually the only editor available in emergencies.
- Relatively hard to learn, but really powerful
- vi in Linux is usually vim (vi improved)
  - Syntax highlighting
  - Arrow keys, Del, BS work in insert mode
  - Mouse support
- VI editor is an interactive editor that you can use to create and modify test files.
- It is used when the desktop environment window system is not available.

VI OPERATIONS

- VI has three basic modes
  - **Command mode**
    - Default mode
    - Perform commands to delete, copy, …
  - **Edit 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

- **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

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

MANIPULATING FILES WITHIN VI

- **Moving the cursor within the vi**
  - **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

GLOBAL INITIALIZATION FILES

- `/etc/profile`
  - This file gets executed whenever a bash login shell is entered as well as by DisplayManager when the desktop session loads.
- `/etc/bash.bashrc`
  - This is the system-wide version of the `~/.bashrc` file. By default this file is executed whenever a user enters a shell or the desktop environment.

INITIALIZATION FILES

- `~/.profile`
  - It gets executed automatically by DisplayManager during startup process desktop session as well as by the login shell when on logs-in from the textual console.
- `~/.bash_profile` or `~/.bash_login`
  - If one of these file exits, bash executes it rather than "~/.profile" when it is started as a login shell. (Bash will prefer "~/.bash_profile" to "~/.bash_login"). However, these files won't influence a graphical session by default.

Only `.profile` works in the GUI but the others in the command line

STARTUP FILES

- `~/.bashrc`
  - By default this file will be executed in each and every invocation of bash as well as while logging in to the graphical environment.

ENVIRONMENT VARIABLES

- **$HOME**
  - Complete path of the user home directory
  - Example
    - `mkdir $HOME/file1`
- **$PATH**
  - A colon-separated list of directories used by the shell to look for executable program names
  - Example
    - `echo $PATH`

      /usr/bin:/bin:/usr/local/java/bin

ENVIRONMENT VARIABLES

- `$PWD`
  - The user current working directory
- `$SHELL`
  - Path name of the login shell
- `$USER`
  - Currently logged in user
- `$HOSTNAME`
  - Name of the computer

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/user
```

- To view the contents of all variables by running the `set` command

COMMAND ALIAS

```
alias l.='ls .* `
alias ll='ls -l `
alias ls='ls `
```

- Type `alias` at the terminal to see all set aliases
- **Remove aliases**

`unalias` command

- **Bypass aliases**

```
alias ls='ls -AF'
/usr/bin/ls
\ls
```

COMMANDS 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.

COMMANDS HISTORY

- ^old^new to repeat the last command with old changed to new. For example,
  - `$ cp file1  /usr/local/src/project`
  - `$ ^file1^file2`
- You will get the output:
  - `$ cp file2 /usr/local/src/project`