Text Editors and Text Processing

HUAWEI

Foreword

- Text processing is a basic OS operation used to manage files. A text editor, a type of computer software, is mainly used to write and view text files. Different text editors have different auxiliary functions. This document describes several common text editors and basic text processing operations.

Objectives

- After completing this course, you will be familiar with:
  - Common Linux text editors
  - Vi and Vim text editor modes
  - Common and quick Vim operations

Contents

Introduction to Linux Text Editors

- A text editor is one of the basic pieces of software in an operating system. There are many types of Linux text editors. Some accompany specific environments and others can be installed as needed.
- Some common Linux text editors are:
  - Emacs
  - Nano
  - gedit
  - KEDIT
  - Vi
  - Vim

Linux Text Editor - Emacs

- Emacs is more like an operating system than an editor. It comes with a built-in web browser, IRC client, calculator, and even Tetris. Emacs is used on a Linux GUI.
- Advantages:
  - Customizable and extensible
  - Powerful functions
  - Can be integrated with many free software programming tools
- Disadvantages:
  - Difficult to get started for common users

Linux Text Editor - Nano

- Nano is a simple text editor with a command line interface. Developed to replace the closed-source Pico text editor, the first version was released, licensed under the GNU General Public License (GPL), in 1999. This free software is also a part of the GNU Project. Nano has many user-friendly features such as syntax highlighting, regular expression searching and replacing, smooth scrolling, multiple buffers, custom shortcut keys, undo, and repeat editing.
- Advantages:
  - Easy to use, very suitable for simple text editing
- Disadvantages:
  - Complex text editing is time-consuming, with no powerful commands available for complex operations, for example, no support for macros, editing multiple files at a time, window splitting, vertical block/rectangle selection and editing, and automatic completion.

Linux Text Editor - gedit

- gedit is a text editor in the GNOME desktop environment and is compatible with UTF-8. gedit is free software which is easy to use and provides syntax highlighting, multiple character encodings including GB2312 and GBK, and GUI tabs for editing multiple files. It also supports a full undo and redo system, searching and replacing, and editing remote files with the GNOME VFS library.
- Advantages:
  - Easy to use on a GUI, featuring a Windows-like experience, for example, the same shortcut keys for operations like copy and paste
- Disadvantages:
  - Requires an installed GUI

Linux Text Editor - KEDIT

- Similar to gedit in the GNOME environment, KEDIT is a text editor in the K desktop environment (KDE). It is a small editor, especially suitable for browsing text and configuration files.
- Advantages:
  - Easy to use on a GUI, featuring a Windows-like experience, for example, the same shortcut keys for operations like copy and paste
- Disadvantages:
  - Requires an installed GUI

Linux Text Editor - Vi

- Vi, the oldest text editor, is a standard Unix text editor and one of the most popular text editors. All Linux and Unix operating systems have the Vi text editor by default. Although Vi operations are different from those of other text editors (such as gedit), Vi is still used frequently because it only needs a character interface and can be used in all Unix and Linux environments.
- Three command modes of the Vi editor:
  - **Command** Type ESC
  - **Insert** Type i
  - **Visual** Default mode Upon start
- Advantages: universal. Almost all Unix and Linux OSs have their own Vi editor.
- Disadvantages: lack of advanced function and display options.

Linux Text Editor - Vim

- Vim is a text editor developed from Vi. Equipped with many convenient programming functions such as code completion, compilation, and error redirection, Vim is frequently used by programmers, and is also one of Unix-like system users' favorite editors alongside Emacs.
- The first version of Vim was released in 1991 by Bram Moolenaar. The initial name was Vi IMitation. After it developed more functions, the name was changed to Vi IMproved. It is now free and open-source software.
- Vim has multiple modes:
  - Basic modes: **Normal**, **Insert**, **Visual**, **Select**, **Command-line**, and **Ex**
  - Derivative modes: **Operator-pending**, **Insert Normal**, **Insert Visual**, **Insert Select**, **Replace**
  - Others: **eVim**
- After installing openEuler 20.03 LTS, you need to manually install Vim.

X

Contents

Command Syntax of the Vim Editor

- Syntax:

```
vim  [options]  [file]...              Edit specified files.
vim  [options]  -                      Read text from standard input (stdin).
vim  [options]  -t  tag                Edit the file where the tag is defined.
vim  [options]  -q [errorfile]         Edit the file where the first error occurs.
```

- Common options:
  - **-c** runs a specified command before opening a file.
  - **-R** opens a file in read-only mode but allows you to forcibly save the file.
  - **-M** opens a file in read-only mode and does not allow you to forcibly save the file.
  - **-r** recovers a crashed session. X
  - **+***num* starts at line *num*.

Basic Vim Operations – Opening a File

```
[root@openEuler ~]# vim  filename
```

- If the *filename* file exists, it is opened and its content is displayed.
- If the *filename* file does not exist, Vim displays **[New File]** at the bottom of the screen and creates the file when saving the file for the first time.

```
[root@openEuler ~]# vim  test.txt
~
~
~
~
~
"test.txt"  [New File]
```

Basic Vim Operations – Moving the Cursor

- Cursor control !!!!
  - Arrow keys or **k**, **j**, **h**, and **l** keys move the cursor up, down, left, and right, respectively.
  - **0** moves the cursor to the beginning of the current line.
  - **g0** moves the cursor to the leftmost character of the current line that is on the screen.
  - **:***n* moves the cursor to line *n*.
  - **gg** moves the cursor to the first line of the file.
  - **G** moves the cursor to the last line of the file.
- Data operations
  - **yy** or **Y** copies an entire line of text.
  - **y***[n]***w** copies 1 or *n* words.
  - **d***[n]***w** deletes (cuts) 1 or *n* words.
  - *[n]* **dd** deletes (cuts) 1 or *n* lines.

Basic Vim Operations – Data Operations

- Copy
  - **yy** or **Y** copies an entire line of text.
  - **y** *[n]* **w** copies 1 or *n* words.
- Paste
  - Line-oriented data:
    - **p** places data below the current line.
    - **P** places data above the current line.
  - Character-oriented data:
    - **p** places data behind the cursor.
    - **P** places data before the cursor.
- Deletion
  - **d** *[n]* **w** deletes (cuts) 1 or *n* words.
  - *[n]* **dd** deletes (cuts) 1 or *n* lines.

Basic Vim Operations – Displaying and Hiding a Line Number

- Displaying a line number
  - :set nu

```
    1  hello
    2  openEuler
~
~
~
~
~
:set nu
```

- Hiding a line number
  - :set nonu

Basic Vim Operations – Finding and Replacing

- Find
  - **:/***word* searches forwards for the *word* string after the cursor. Press **n** to match the next *word* string and press **N** to match the previous *word* string.
  - **:?***word* searches backwards for the *word* string before the cursor. Press **n** to match the next *word* string and press **N** to match the previous *word* string.
- Replace

g global (all occurrences)

  - **:1,5s/***word1***/***word2***/g** replaces all occurrences of *word1* in lines 1 to 5 with *word2*. If **g** is not specified, only the first occurrence of *word1* in each line is replaced.
  - **%s/***word1***/***word2***/gi** replaces all occurrences of *word1* with *word2*. **i** indicates case-insensitive matches.

Basic Vim Operations – Highlighting Search Results

- Run the following command in **Command** mode for a temporary setting:
  - :set hlsearch

```
hello
openEuler
hello
world
~
~
:set nu
```

- Add **set hlsearch** to the **/etc/vimrc** file and update variables for a permanent setting:

Basic Vim Operations – Modifying a File

- After you run the **vim** *filename* command to open a file, the system enters **Normal** mode. To modify the file, press **i** to enter **Insert** mode. The system displays a message at the bottom, indicating that the current mode is **Insert**. You can press **ecs** to exit **Insert** mode and return to **Normal** mode.

```
[root@openEuler ~]# vim  test.txt
# Press i to enter Insert mode.
~
~
~
~
~
~
~
-- INSERT --
```

Basic Vim Operations – Undoing or Redoing

- **u** undoes the latest change.
- **U** undoes all changes on the current line since the cursor has been positioned on the line.
- **Ctrl+R** redoes the last undo operation.

Basic Vim Operations – Saving a File and Exiting

- Exit **Insert** mode:
  - Press **ecs** to exit **Insert** mode.
- The involved commands are as follows:
  - **:w** saves a file.
  - **:q** exits the editor.
  - **:wq!** saves a file and exits the editor.
  - **:q!** forcibly exits the editor.
  - **:wq!** forcibly saves a file and exits the editor.

Contents

Viewing a File – cat (1)

- **cat** is a tool for viewing and connecting text files. It has the following functions:
  - **cat filename** displays file content.
  - **cat > filename** edits a file.
  - **cat file1 file2 > file3** combines several files into one.
- Main options:
  - **-n** numbers all lines starting from 1 and displays the number at the beginning of each line.
  - **-b** numbers non-empty lines starting from 1 and displays the number at the beginning of each line.
  - **-s** outputs only one blank line when multiple blank lines exist.
  - **-E** adds **$** at the end of each line.
  - **--help** displays the help information.

λ

Viewing a File – cat (2)

- Example

```
[root@openEuler ~]# cat /etc/profile                # View the /etc/profile file content.
[root@openEuler ~]# cat -b /etc/profile             # View the /etc/profile file content and number non-blank
lines starting from 1.
[root@openEuler ~]# cat -n /etc/profile             # View the /etc/profile file content and display the line
number at the beginning of each line.
[root@openEuler ~]# cat -E /etc/profile             # View the /etc/profile file content and add $ at the end of
each line.
[root@openEuler ~]# cat -s /etc/profile             # View the /etc/profile file content. If multiple blank lines exist,
only one is displayed.
```

Viewing a File – more (1)

- **more** can be used to view one file at a time or a standard input page. Unlike **cat**, **more** can be used to view the file content by page or to directly jump to another line.
- Command syntax: **more [options] <file>...**
- Common options:
  - **+***n* displays from the *n*th line.
  - **-***n* defines the screen size to *n* lines.
  - **-c** clears a screen from the top and displays information.
  - **-s** displays multiple consecutive blank lines in one line.

Viewing a File – more (2)

- Common operations:
  - **Enter** scrolls down one line by default.
  - **Ctrl+F** scrolls down to the next page.
  - **Space key** scrolls down to the next page.
  - **Ctrl+B** scrolls up one page.
  - **b** scrolls up one page.
  - **=**outputs the current line number.
  - **:f** outputs file name and current line number.
  - **q** exits **more**.

Viewing a File – less (1)

- **less** can be used to view one file at a time or a standard input page, and is a more flexible command than **more**.
- Command syntax: **less** *[option] file*
- Common options:

x

  - **-f** forcibly opens a special file, such as peripheral device code, directory, or binary file.
  - **-g** specifies only the last found keyword.
  - **-i** ignores case sensitivity during search.
  - **-N** displays the line number of each line.
  - **-s** outputs only one blank line when multiple blank lines exist.
  - **-o** <*file name*> saves the **less** output to a specified file.

Viewing a File – less (2)

- Main operations:
  - **b** turns to the previous page.
  - **d** turns to the next half page.
  - **h** displays the help information.
  - **q** exits **less**.
  - **u** turns to the previous half page.
  - **y** turns to the previous line.
  - **Space key** turns to the next line.
  - **Enter** turns to the next page.
  - Up or down arrow key: turns to the previous or next line, respectively.

Extracting a File – head

- **head** is used to display the beginning of a file to the standard output. By default, the first 10 lines of a file are displayed.
- Command syntax: **head** *[option]*... *[file]*...
- Common options:
  - **-q** hides a file name during output. By default, the file name is not displayed.
  - **-v** displays a file name during output.
  - **-c** *num* displays the first *num* bytes.
  - **-n** *num* displays the first *num* lines.

X

Extracting a File – tail

- **tail** is used to display the end of a file to the standard output. By default, the last 10 lines of a file are displayed.
- Command syntax: **tail** *[option]*... *[file]*...
- Common options:
  - **-f** reads log files cyclically for log management
  - **-q** hides a file name. File names are not displayed by default.
  - **-v** displays a file name.
  - **-c** *num* displays the last *num* bytes of a file.
  - **-n** *num* displays the last *num* lines of a file.

Extracting a Column or Field – cut

- **cut** is used to display a specific column of a file or standard input. For example:

```
[root@openEuler ~]# cut  -d: -f1 /etc/passwd                    # Display the first column of the /etc/passwd file separated
by colons (:).
```

- Command syntax: **cut** *[option]*... *[file]*
- Common options:
  - **-b** [*range*] displays only the content within the specified range in a line.
  - **-c** [*range*] displays only characters within the specified range in a line.
  - **-d** specifies a field separator. The default field separator is **TAB**.
  - **-f** [*range*] displays the content of the specified *num*th field. Multiple fields can be separated by commas (,).

Extracting a Column or Field – awk

- **awk** is a powerful text analysis tool. It reads files or standard input by line, uses spaces as the default separator to slice each line, and performs analysis on each slice.

```
[root@openEuler ~]# last -n 5 | awk '{print $1}' # Display the last five accounts that have logged in to the system.
```

- Command syntax: **awk** *action filename*, for example, **awk '{print $0}' test.txt**

Extracting Keywords - grep

- grep is a powerful text search tool that uses regular expressions to search for text in one or more files, and then print matched lines. To find a pattern more than one word long, enclose the text string with single or double quotation marks, otherwise search terms after the first word will be misinterpreted as file names to search in. The search result is sent to standard output, leaving the original file(s) unaltered.
- Command syntax: **grep** *[option] [file]*...
- Common options:
  - **-c** prints a count of matching lines.
  - **-i** ignores case sensitivity.
  - **-w** prints whole word matches.
  - **-x** prints only results matching the whole line.

4

Text Statistics - wc

- **wc** is used to calculate the number of words, bytes, or columns of a file. If the file name is not specified or the given file name is **-**, **wc** reads data from the standard input device.
- Command syntax: **wc** [*option*]... [*file*]...
- Common options:
  - **-c**, **--bytes**, or **--chars** displays only the number of bytes.
  - **-l** or **--lines** displays only the number of lines.
  - **-w** or **--words** displays only the number of words.

Text Sorting - sort

- **sort** is used to sort files and output the sorting result in standard mode. You can use **sort** to obtain input from a specific file or standard input (stdin).
- Command syntax: **sort** [*option*]... *[file]*...
- Common options:
  - **-b** ignores the space character at the beginning of each line.
  - **-c** checks whether files are sorted in sequence.
  - **-d** processes only letters, digits, and spaces during sorting.
  - **-f** regards lowercase letters as uppercase letters during sorting.
  - **-n** sorts by value.
  - **-r** sorts in reverse order.
  - **-o** <*file*> saves the sorted result to a specified file.
  - **-u** ignores repeated lines.

x

Text Comparison - diff

- **diff** compares the similarities and differences between text files line by line. If a directory is specified, **diff** compares files with the same file name in the directory but does not compare subdirectories.
- Command syntax: **diff** [*option*]... *file*
- Common options:
  - **-B** does not check blank lines.
  - **-c** displays all content and marks the differences.
  - **-i** ignores case differences.
  - **-r** compares files in subdirectories.
  - **-w** ignores all space characters.

x

Text Operating Tool - tr

- **tr** reads data from a standard input device, converts character strings, and outputs the result to the standard output device. It is used to convert or delete characters in a file.
- Command syntax: tr [*option*]... set1 [set2]

```
[root@openEuler ~]# cat text.txt | tr a-z A-Z          # Convert lowercase letters to uppercase letters.
```

- Main options:
  - **-c** inverts the selection of characters. That is, characters that meet **set1** are not processed, and the remaining characters are converted.
  - **-d** deletes characters.
  - **-s** reduces consecutive repeated characters to a specified single character.
  - **-t** reduces the specified range of **set1** to the length of **set2**.

Text Operating Tool - sed

- **sed**, a streamlined, noninteractive editor, can edit standard input and text from files. Compared with **tr** (which performs character-based transformation), **sed** can modify character strings. When a command is executed, **sed** reads a line from the file or standard input and copies it to buffers. **sed** continues to read each next line until all lines have been edited. As such, sed only changes the copied text stored in buffers. To edit the original file directly, use the **-i** option. You can also redirect results to a new file.
- Command syntax: sed [*option*]... [*option*] {**script-only-if-no-other-script**} [input-file]...
- Common options:
  - **-n** cancels the default output.
  - **-e** specifies multipoint editing. Multiple subcommands can be executed.
  - **-f** reads commands from a script file.
  - **-i** directly edits the original file.
  - **-l** specifies the line length.
  - **-r** uses an extended expression in a script.

Summary

- This document describes the development of the Linux OS and introduces the openEuler OS, as well as its installation methods and shortcut operations.

More Information

- For more information about Linux shortcut keys, visit https://linuxtoy.org/archives/bash-shortcuts.html.

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI