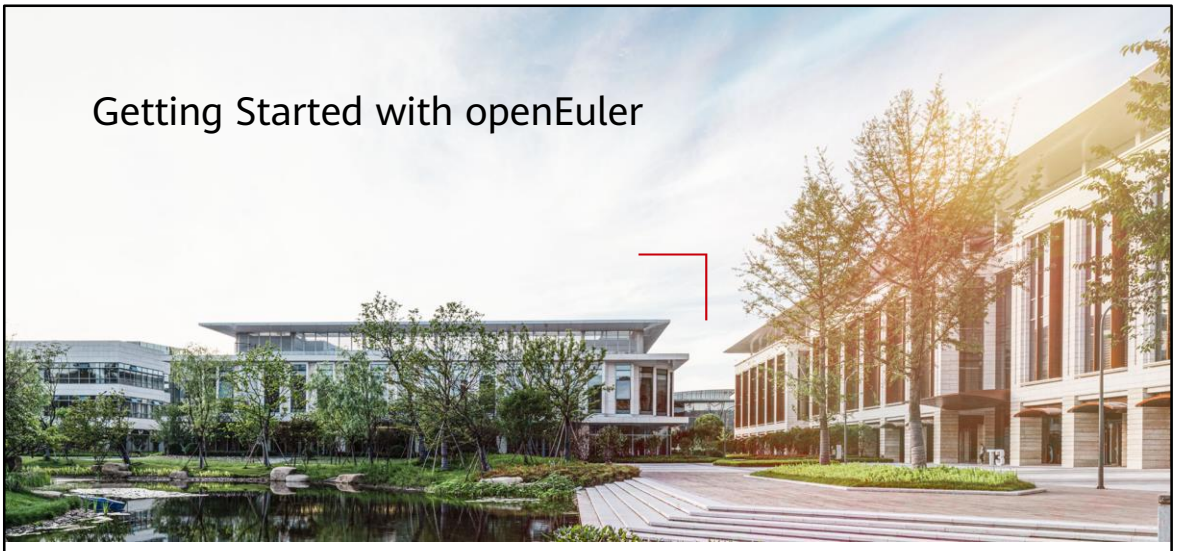
Getting Started with openEuler



HUAWEI

Foreword

- This document describes the following:
  - GNU Free Software Foundation
  - Origin of Linux
  - openEuler operating system (OS)
  - How to install and log in to the openEuler OS

Objectives

After completing this course, you will understand:

- Open source and GNU
- Origin of Linux
- Linux principles
- How to install openEuler
- How to log in to openEuler

Contents

OS Overview

- An OS is a set of programs that controls and manages the hardware and software resources of an entire computer system, providing a convenient interface and environment for users and other software. A computer's OS is its most basic system software, and it gradually evolves in line with the development of computer research and application.

Unix Development History

- Unix development
  - In the 1960s, Bell Labs, MIT, and GE jointly developed a multiprogramming information computing system named Multics.
  - In 1970, Ken Thompson developed Unix.
  - In 1974, Bell Labs released Unix, which became widely used in universities.
  - After the breakup of AT&T in 1982, Unix began to charge for commercial use.
  - Some large hardware companies have developed different versions of Unix based on their own computer systems.
    - AIX
    - HP-UX
    - Digital Unix
    - ...

- Back in the 1960s, computers were not widely used, and only a small number of individuals could even access them. At that time, computer systems were used for batch processing, whereby multiple tasks were submitted to the computer at once, which required users to wait for the results. No other interactions were supported during this process, resulting in a waste of computer resources. In 1965, Bell Labs, MIT, and GE worked together to change this situation by developing a time-sharing multitasking system. Simply put, they realized the vision of multiple people using computers at the same time. The resulting computer system was named Multics, which is short for Multiplexed Information and Computing Service. MULTICS might have been too far ahead of its time, and in 1969, unhappy with the slow pace of development, Bell Labs eventually pulled out of the project.
- After quitting the Multics program, the scientists at Bell Labs did not immediately work on a new project. One man, Ken Thompson, was in the process of developing a game called Space Travel while working on Multics. When Bell Labs pulled out, Thompson was no longer able to use the Multics environment. He spent a month writing a lightweight OS that was capable of running Space Travel. However, he found that his friends were far more interested in his system than in his game.
- Because Multics is short for Multiplexed information and Computing Service, they named this small system Uniplexed Information and Computing Service, as a play on words of MULTICS. These days we call it "Unix" for short.
- By then is 1970, which is referred to as the Unix epoch. As a result, the time in computer systems is measured from 1970.
- Unix is a multi-tasking and multi-user OS developed by AT&T Bell Labs in 1969. It was initially free of charge, and its security, efficiency, and portability made it ideal for servers. Later, it was commercialized. Many high-end applications in large-scale data centers use the Unix system.

GNU and Open Source

- In 1984, Richard Stallman launched the Free Software Campaign, established the Free Software Foundation, and achieved the following:
  - Created an open source version of the Unix utility.
  - Released the general public license (GPL).
  - Open source, also known as open source code, laid the foundations for the rapid development of IT technologies.
- There are many open source licenses, each with different rules. Some common open source licenses are as follows:
  - Mulan.
  - GPL.
  - Lesser GPL (LGPL).
  - Berkeley Source Distribution (BSD).

- The Free Software Foundation provides freedom in four aspects:
  - Freedom to run programs for any purpose
  - Freedom to learn and modify the source code
  - Freedom to redistribute programs
  - Freedom to create derivatives
- Mulan is China's first open source license. It involves five major aspects: copyright license, patent license, non-trademark license, distribution restriction, and disclaimer and liability limitation. It grants "every contributor" a permanent, global, free, non-exclusive, and irrevocable copyright license. Under the license, you can copy, use, modify, and distribute your "contributions" regardless of whether they are modified or not.
- GPL applies two measures to protect the rights of programmers: (1) copyright protection for software; (2) license for programmers, which gives them the legal permission to copy, distribute, and modify the software. In terms of copying and distribution, the GPL states that "You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program. You may charge any price for each copy that you convey, and you may offer support or warranty protection for a fee." Any software that uses ("use" refers to class library reference, modified code, or derivative code) from the GPL must use the GPL, which must be open source and free of charge.

Birth of Linux

- Birth of Minix
  - In 1987, Andrew S. Tanenbaum, a professor of Vrije University in Amsterdam, wrote Minix. Similar to Unix, this OS was instead dedicated to teaching.
- Birth of Linux
  - On September 17, 1991, Linus Torvalds released his own Linux OS on the Internet and claimed that it was free of charge. In addition, he hoped to improve the Linux OS through the efforts of developers.
  - In 1994, Linux kernel 1.0 was officially released.
  - More accurately, the full name of Linux is GNU/Linux.
- Today's Linux
  - Today, Linux has many derivatives, such as Red Hat, openSUSE, Ubuntu, and Deepin.
  - Linux distribution = Linux kernel + utility software

- Many universities could no longer use Unix due to its expensive licensing fees. In 1987, Andrew S. Tanenbaum, a professor of Vrije University in Amsterdam, wrote a Unix-like OS, Minix, for educational purposes. The spread of Minix attracts many developers around the world to use and improve the system. However, Andrew refused to incorporate these improvements into Minix, as he wanted to keep the system dedicated for teaching only.

Introduction to Linux Releases

- Kernel versions
  - You can visit kernel.org to view or download all Linux kernel versions.
  - The Linux kernel version number is composed of three digits:
    - The first digit indicates the current major release.
    - An even second digit indicates a stable version, while an odd second digit indicates a version under development.
    - The third digit indicates the number of revisions.
  - The kernel version of openEuler 20.03 LTS is 4.19.90.
- Linux distributions
  - Commercial distros: maintained by companies and provides charged services, such as patch upgrades.
  - Community distros: maintained by the community organization and free of charge.

openEuler OS

- openEuler is a free, open source OS operated by the openEuler community. The current openEuler kernel is based on Linux and supports Kunpeng and other processors, fully unleashing the potential of computing chips. As an efficient, stable, and sustainable open source OS built by global open source contributors, openEuler applies to database, big data, cloud computing, and artificial intelligence (AI) scenarios.
- openEuler has two kinds of releases:
  - Innovation release
    - Supports the technical innovation and content innovation of Linux enthusiasts, such as openEuler 20.09.
    - Generally, a new version is released every half a year.
  - Long-term support (LTS) release
    - A stable version of openEuler, such as openEuler LTS 20.03.
    - Usually a new version is released every two years.

- This course takes openEuler 20.03 LTS as an example to illustrate all operations involved.

Contents

openEuler OS Installation Process

Prepare the installation environment.

- openEuler can be installed on the Arm- and x86-based computing platforms. The ISO files of the two platforms are incompatible.
- You can obtain the ISO image from the openEuler community .

Select an installation mode.

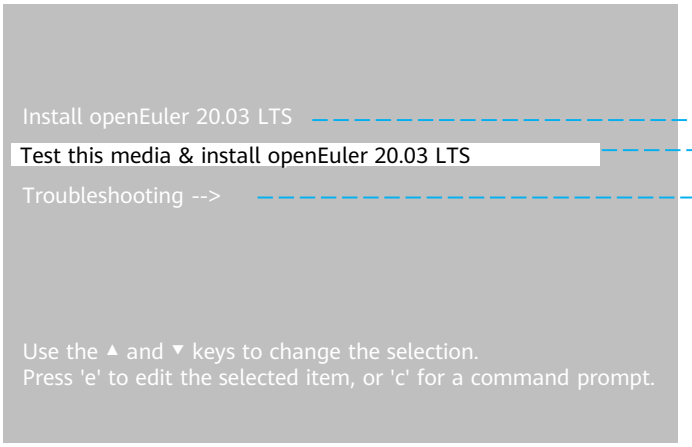
- openEuler supports various installation modes. If only a few devices need to be installed, use a USB flash drive, CD-ROM, or virtual CD-ROM drive. During batch installation, use the PXE boot mode.

Configure system settings.

- When installing openEuler, you need to set the system parameters, such as the installation language, installation location, software version, host name, and network configuration.

Installation complete.

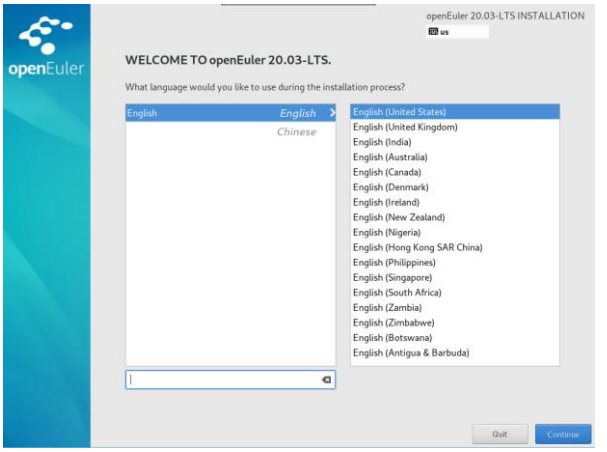
openEuler Installation and Configuration - Selecting Installation Options



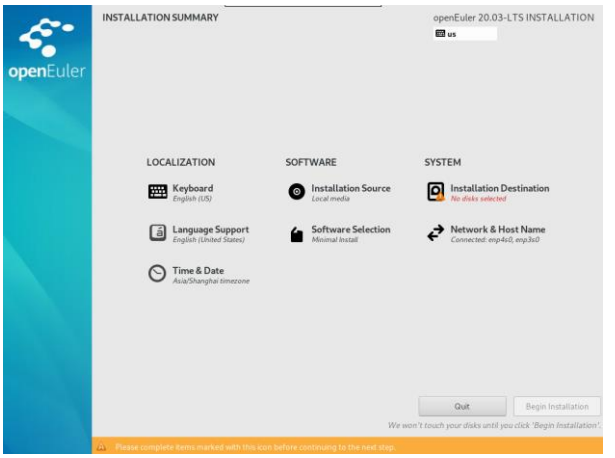
HUAWEI

The following uses the virtual drive as an example to describe how to install the OS.

openEuler Installation and Configuration - Selecting an Installation Language



openEuler Installation and Configuration - System Settings



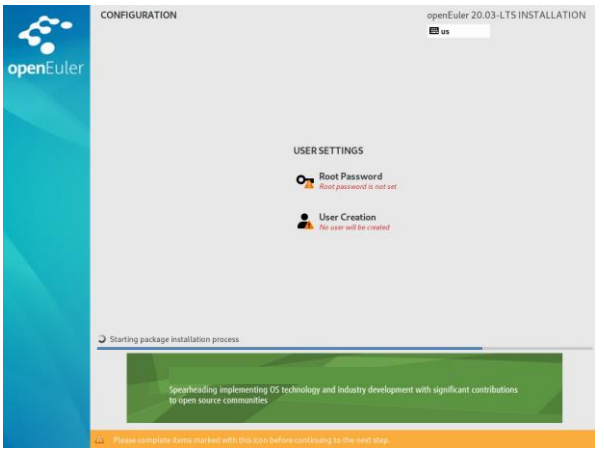
openEuler Installation and Configuration - Setting the Installation Location

- Set the system installation location and system partitions.
  - Select the disk where the OS is to be installed.
  - Choose how to partition your disk.
    - In manual mode, you can set partitions, including common partitions, logical volumes, and thin-provisioning logical volumes.
    - You are advised to set the following partitions for the openEuler system startup:
      - swap: swap partition, which is used to swap dirty data in the memory when space is insufficient. If the memory is small, you are advised to set the swap partition size to twice the memory size. If the memory is large, you can reduce the swap partition size.
      - /boot: booting
      - /boot/efi: boot device and application program to be started by the extensible firmware interface (EFI).
      - /: root partition. In Linux, everything starts from the root partition.

openEuler Installation and Configuration - Selecting Software to Be Installed

- openEuler 20.03 LTS supports the following software installation options:
  - Minimum installation
    - Minimum Linux installation: Most software is not installed. This mode is applicable to scholars who have background in Linux and want to further understand the Linux architecture. Other software is also available on the right.
  - Server
    - Install the software involved in the server scenario. Other software is also available on the right.
  - Hypervisor
    - Install the software involved in the virtualization scenario. Other software is also available on the right.

openEuler Installation and Configuration - Setting the Password of the Root User and Creating a User



HUAWEI

During openEuler installation, you need to set the password of the **root** user. You can create a common user as required. The **root** user is the super administrator of the system, who has the highest permissions. Usually, the Linux administrator cannot use the **root** user to manage the system.

Contents

Linux GUI and CLI

- GUI stands for graphical user interface. All elements of this user interface are graphical. The mouse is used as the input tool, and buttons, menus, and dialog boxes are used for interaction, enhancing ease of use.
- CLI is short for command line interface. All elements of a CLI are character-based. The keyboard is used as the input tool to enter commands, options, and parameters to execute programs, achieving high efficiency.
- No GUI is available for openEuler 20.03 LTS.

- Example:
  - Calculator of the Windows system. Open **Calculator** on the Windows GUI: Choose **Start Menu** > **Programs** > **Accessories** > **Calculator**. Click buttons on the calculator to enter expressions. The Windows calculator looks like physical one, which is user-friendly. Or, on-screen keyboard displayed when some programs ask you to enter the passwords by clicking the numbers on the on-screen keyboard. Banks also have on-screen keyboard for entering password. However, it enables the user to interact using hands, rather than using a mouse.
  - To use calculator via CLI, enter **bc** to open the calculator, enter expressions, such as **1 + 1**, and press **Enter**. Result **2** is obtained.

How to Log In to the Linux OS

- You can log in to the Linux OS through either of the following methods:
  - Local login
    - This method is similar to starting up your own computer or directly connecting the server to the monitor.
    - A typical Linux system runs six virtual consoles and one graphical console. Currently, openEuler does not support the GUI.
    - You can press Ctrl+Alt+F[1,6] to switch between the six virtual consoles.
  - Remote login
    - The openEuler OS supports remote login by default. You can also change the login mode.
    - You can use PuTTY or Xshell to remotely log in to the openEuler OS.

Introduction to Shell

- Shell, a program compiled in C language, serves as a bridge for those who wish to use Linux. Users control the Linux system through shells, which are also used by the Linux system to display system information.
- Common shells include bash, sh, csh, and ksh. You can specify a login shell when creating a user, or enter a shell name to open a shell. For example:

```
[root@openEuler ~]# sh
sh: openEuler_history: command not find
sh-5.0#                                  # The interaction modes varies with different shells.
sh-5.0# exit                             # Enter exit to exit the current shell.
[root@openEuler ~]#
```

- The default login shell for openEuler users is bash.
- The default system prompt is [Current login user@Host name Current location]$.
  - Generally, the last prompt of the root user and common users is #.

- Unless otherwise specified, this course uses the **bash** shell by default.

Changing Passwords

- The password is directly related to the security of the system and its data.
- To ensure system security, perform the following operations:
  - Change the password upon the first login.
  - Change passwords periodically.
  - Set a password with high complexity. For example, set a password containing more than eight characters and three or more of the following types of characters: uppercase letters, lowercase letters, digits, and special characters.
- You can run the passwd command to change your password.

```
[root@openEuler ~]# passwd
Changing password for user root.
New password:                                  # Enter the new password.
Retype new password:                           # Enter the new password again.
passwd: all authentication tokens updated successfully
```


- For security purposes, the openEuler OS does not display the password by default when you enter it, and does not use any characters to indicate the number of digits.

Linux Users

- The root user is a special administrator in the Linux OS.
  - This super administrator is similar to the administrator in the Windows OS.
  - The root user has the highest permissions, and can cause infinite damage.
  - Do not use the root user unless necessary.
- You can run the su - username command to switch users.
- You can check whether the current user is the root user or a common user through the command prompt. In the Unix or Linux OS, the command prompt of the root user generally ends with #, while that of a common user generally ends with $.
- You can also run the id command to view the current username.

Shortcut Operations with the Bash Shell

- tab
  - You can use the tab key to supplement functions and quickly enter commands or parameters.
- history
  - The history tool records historical commands. You can run the history command to view historical commands, or run the history n command to execute the historical command numbered n.
- ↑ and ↓
  - You can press the ↑ and ↓ arrow keys to quickly view historical commands.
- home and end
  - To move the cursor to the beginning or end of the current line, press home or end, respectively.
- clear and Ctrl+L
  - When the page is full of characters, you can enter clear or press Ctrl+L to quickly clear the screen.

Quiz

1. (True or false)You do not need to specify the password of the root user during the openEuler OS installation. However, you must specify the password of the root user when logging in to the OS after the installation.
   A. True
   B. False
2. (Single-answer)Which of the following keys can be used to quickly supplement commands and parameters?
   A. Ctrl+L
   B. ↑ and ↓
   C. tab
   D. Space

- Answer:
  1. A
  2. C

Summary

- This document describes the development of the Linux OS, introduces the openEuler OS, and describes how to install openEuler and related shortcut operations.

More Information

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

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.

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.

CLI Basics

HUAWEI

Foreword

- This document describes the basics of command line operations, including the command line interfaces (CLIs) and CLI-based file management.

Objectives

After completing this course, you will be familiar with:

- Basic Linux commands
- Linux login commands
- Linux system power management commands
- Linux file management commands

Contents

Linux GUI and CLI

- GUI stands for graphical user interface. All elements of this user interface are graphical. The mouse is used as the input tool, and buttons, menus, and dialog boxes are used for interaction, enhancing ease of use.
- CLI is short for command line interface. All elements of a CLI are character-based. The keyboard is used as the input tool to enter commands, options, and parameters to execute programs, achieving high efficiency.

- Example:
  - Start the calculator on the Windows GUI. Choose **Start** > **Programs** > **Windows Accessories** > **Calculator**. In the calculator, click the button on the top to enter an expression. Alternatively, a small keyboard is displayed when certain programs require you to enter the password, asking you to click on the numbers above. This user-friendly method provides a calculator which looks similar to the input device used at bank ATMs. The difference here is that you click it using a mouse, rather than using your own hands.
  - On the CLI, enter **bc** to start the calculator. Enter the calculation **1** + **1** and press **Enter**. Result **2** is obtained.

Why We Use the Linux Command Line?

- More efficient
  - The Linux system allows rapid operations using a keyboard, rather than a mouse.
  - The GUI is fixed, while CLIs in a script can be compiled to complete all required tasks. For example: deleting outdated log files.
- Low overhead (compared with GUI)
  - Running a GUI requires a large number of system resources, whereas a CLI is far more efficient. As a result, system resources can be allocated to other operations.
- CLIs are often the only choice
  - Most OSs for running servers do not utilize a GUI.
  - Tools for maintaining and managing connected devices do not provide a GUI.

Syntax of Linux Commands

- Syntax: *command* [-*option*] [*parameter*]
- Example: **ls -la /etc**
- Note:
  - Some commands do not comply with this format. The [] symbol indicates an option.
  - If there are multiple options, you can write them together.
  - Options can be short by following one hyphen (-) or long by following two hyphens (--). For example, **ls -a** equals **ls --all**.

- The above is an example of a common command format in the Linux system. Almost all commands (unless otherwise stated) comply with this syntax format.

Linux Command Line Keyboard Shortcuts

- **Tab**: automatically supplements commands or file names, saving time and improving accuracy.
  - If no command is entered, press **Tab** twice to list all available commands.
  - If you have entered a part of the command name or file name, pressing **Tab** will supplement them automatically.
- Cursor
  - ↑: Press ↑ to display recently executed commands, enabling you to quickly select and run them.
  - ↓: Used together with ↑, facilitating command selection.
  - **Home**: Press **Home** to move the cursor to the beginning of the current line.
  - **Ctrl**+**A**: Press **Ctrl**+**A** to move the cursor to the beginning of the line.
  - **Ctrl**+**E**: Press **Ctrl**+**E** to move the cursor to the end of the line.
  - **Ctrl**+**C**: Press **Ctrl**+**C** to stop the current program.
  - **Ctrl**+**L**: Press **Ctrl**+**L** to clear the screen.

Classification of Linux Commands

Category	Example Commands
Login and power management	login, shutdown, halt, reboot, install, exit, and last
File processing	file, mkdir, grep, dd, find, mv, ls, diff, cat, and ln
System management	df, top, free, quota, at, ip, kill, and crontab
Network operation	ifconfig, ip, ping, netstat, telnet, ftp, route, rlogin, rcp, finger, mail, and nslookup
System security	passwd, su, umask, chgrp, chmod, chown, chattr, sudo ps, and who
Others	tar, unzip, gunzip, unarj, mtools, and man

Contents

Login Command 1 - login (1)

- **login** is used to log in to the system, which is applicable to all users.
- If you choose to log in to the Linux OS in command line mode, the first command required is **login**.

```
Authorized users only. All activities may be monitored and reported.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Jul 29 14:15:56 2020 from 172.19.130.204

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time:  Wed Jul 29 14:25:33 CST 2020

System load:    0.00
Processes:      185
Memory used:    20.0%
Swap used:      0.0%
Usage On:       13%
IP address:     192.168.110.245
Users online:   2

[root@localhost ~]#
```


- Enter the username in the second line, press **Enter**, and enter the password after **Password** to log in to the system. For security purposes, characters are not displayed on the screen and the cursor does not move when you enter the password.

Login Command 1 - login (2)

- Linux is a multi-user OS that allows multiple users to log in at the same time and a single user to log in multiple times.
- This is because Linux, like many versions of Unix, provides a virtual console access mode that allows users to log in to the console (a monitor and keyboard that are directly connected to the system) multiple times simultaneously.
- Each virtual console can be regarded as an independent workstation and you can switch between workstations.
- You can press **Alt** and a function key (typically **F1** to **F6**) to switch between virtual consoles.

- For example, pressing **Alt**+**F2** after login will display the **login:** prompt, indicating the second virtual console. You can also press **Alt**+**F1** to return to the first virtual console. A newly installed Linux system allows users to access the first six virtual consoles using the **Alt**+**F1** to **Alt**+**F6** keys.
- The virtual console is especially useful when a program goes wrong and causes a system deadlock, as it enables the user to switch to another virtual console and close the program.

Login Command 2 - last

- **last** is used to display the recent logins of users or terminals, and is applicable to all users. Run the **last** command to view a program's log. The user will know who used, or attempted to connect to, the system.
- Main options:
  - **-n**: specifies the number of output records.
  - **-t tty**: displays the login status of the specified virtual console.
  - **-y**: displays the year, month, and day of the record.
  - **-ID**: displays the username.
  - **-x**: displays the history of system shutdowns, user logins, and user logouts.

- Other options of the **last** command:
  - **-f** *file* : specifies a log file for query.
  - **-h** *node* : displays the login information only on the specified node.
  - **-i** *IP* : displays the login information of the specified IP address.
  - **-1**: displays the IP address of a remote device.

Login Command 3 - exit

- **exit** is used to log out of the system, and is applicable to all users.
- The **exit** command has no options. After this command is executed, the system exits and the login page is displayed.

Contents

Power Management Command 1 - shutdown (1)

- **shutdown** is used to shut down the computer, and is only applicable to the superuser.
- Main options:
  - **-h**: powers off the server after it is shut down.
  - **-r**: powers on the server after it is shut down. (This operation is equivalent to restarting the server.)
  - **-t**: indicates the time after which the **init** program is shut down before changing to another run level.
  - **-k**: sends a warning signal to each user. It does not shut down the computer.
  - **-F**: forcibly performs file system consistency check (fsck) when restarting the computer.
  - **-time**: specifies the time before the shutdown.

- For a computer system, a superuser is a special user involved with system management. Compared with other common users, the superuser has the highest permissions and can configure and maintain the entire system. Standard users are only granted a subset of the superuser's permissions.

Power Management Command 1 - shutdown (2)

- The **shutdown** command safely shuts down the system. It is dangerous to shut down a Linux system by directly powering it off.
- Unlike the Windows OS, Linux runs many processes in the background. As such, forcible shutdown may result in data loss, leading to system instability or even damaging hardware in some cases.
- If you run the **shutdown** command to shut down the system, the system administrator notifies all login users that the system will be shut down and the **login** command will be frozen. As a result, no further users can log in to the system.

Power Management Command 2 - halt

- **halt** is used to shut down the system, and is only applicable to the superuser.
- Main options:
  - **-n**: prevents synchronization of system calls. It is used after the root partition is repaired using fsck, and prevents the kernel from overwriting the repaired superblock with that of an earlier version.
  - **-w**: writes the **wtmp** file in **/var/log/wtmp** instead of restarting or shutting down the system.
  - **-f**: forcibly shuts down or restarts the system without calling the **shutdown** command.
  - **-i**: shuts down all network interfaces before shutting down or restarting the system.
  - **-f**: forcibly shuts down the system without calling the **shutdown** command.
  - **-d**: shuts down the system without making a record.

- When **halt** is executed, the application process is terminated. System calls are synchronized to forcibly write the data in the buffer to the disk. After the write operation of the file system is complete, the kernel is stopped. If the system run level is 0 or 6, shut down the system. Otherwise, use the **shutdown** command (with the **-h** option) instead.
- The **sync** command is used to forcibly write the data in the buffer to the disk immediately.
- The **fsck** command is used to check and attempt to rectify faults occurred in the FAT32 file system.
- A superblock is located at the beginning of a block group. It describes the data structure of the overall information about a file system, including the static distribution of directories and files, and the size and quantity of each structure.
- The **/var/log/wtmp** file is a binary file that records the number of login times, as well as login durations, for each user.

Power Management Command 3 - reboot

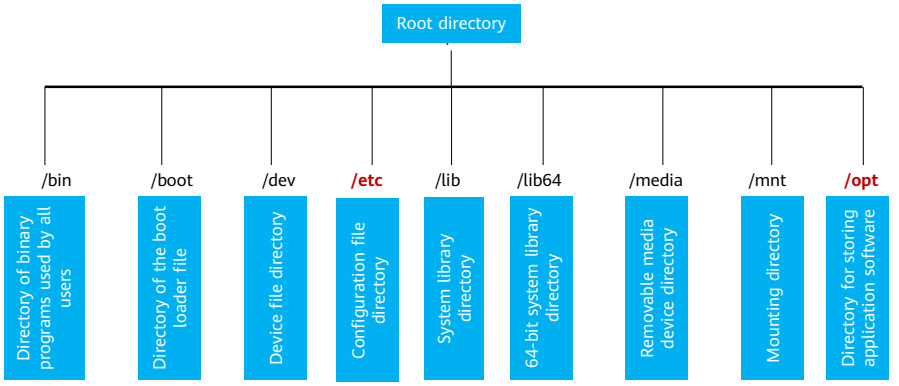
- **reboot** is used to restart the computer, and is applicable to the system administrator.
- Main options:
  - **-n**: saves the data and restarts the system.
  - **-w**: writes records to the **/var/log/wtmp** file. It does not restart the system.
  - **-d**: does not write records to the **/var/log/wtmp** file. (The **-n** option contains **-d**.)
  - **-i**: restarts the system after disabling the network settings.

Contents

Linux File Directory Structure (1)

- On the Linux OS, everything is a file.
- The file directory utilizes a tree structure, with / as the root directory.

```
[root@localhost ~]# ls /
bin   dev  home  lib64       media  opt   root  sbin  sys  usr
boot  etc  lib   lost+found  mnt    proc  run   srv   tmp  var
```



HUAWEI

- The core philosophy of Linux is that everything is a file, , which means that all operations on files, including directories, character devices, block devices, sockets, printers, processes, threads, and pipes, can be read and written by using functions such as fopen(), fclose(), fwrite(), and fread().
- After logging in to the system, enter the **ls /** command in the current command window. You may find the following directories:
- **/bin**: short for binary. This directory stores the frequently used commands.
- **/boot**: stores some core files used for starting the Linux OS, including some connection files and image files.
- **/dev**: short for device. This directory stores Linux external device files. The method used to access devices on Linux is the same as that for accessing files on Linux.
- **/etc**: stores all configuration files and subdirectories required for system management.
- **/lib**: stores the system's most basic dynamic link libraries (DLLs). The function of this directory is similar to the of storing DLL files on Windows. Almost all applications need to use these shared libraries.
- **/media**: The Linux system automatically identifies some devices, such as USB flash drives and CD-ROM drives. After identifying devices, the Linux system mounts the devices to this directory.
- **/mnt**: temporary mount point for other file systems. You can mount the CD-ROM drive to **/mnt** and then go to this directory to view the contents in the CD-ROM drive.
- **/opt**: stores additional software that is installed on the host. For example, if you install an Oracle database, you can save the installation package to this directory. By default, this directory is empty.

- **/proc**: a virtual directory which is the mapping of the system memory. You can obtain system information by directly accessing this directory.
- **/root**: the home directory of the system administrator, also called the superuser.
- **/run**: a temporary file system that stores the information generated after the system is started. When the system is restarted, the files in this directory should be deleted or cleared. If the **/var/run** directory exists on your system, it should be pointed to **/run**.
- **/sbin**: **s** indicates the superuser. This directory stores the system management program used by the system administrator.
- **/srv**: stores the data that needs to be extracted after a service is started.
- **/sys**: the file system directory, which is a major change to the Linux 2.6 kernel. A new file system, **sysfs** in the Linux 2.6 kernel, is installed in this directory.
- **/tmp**: stores temporary files.
- **usr**: Many user applications and files are stored in this important directory, which is similar to the **program files** directory in Windows.
- **/var**: stores the contents that are constantly expanded, for example, log files. You are advised to place place frequently modified directories here.
- **/home**: the home directory of a user. On Linux, each user has a directory which is

generally named after the username.

Linux Directory Usage (1)

Directory	Main Files and Their Functions
/bin	**bin** is short for binary. This directory stores the most frequently used commands.
/boot	Stores core files used for starting the Linux OS, including some connection files and image files.
/dev	**dev** is short for device, and this directory stores Linux external device files. The method used to access devices on Linux is the same as that for accessing files.
/etc	Stores all configuration files and subdirectories required for system management.
/lib	Stores the system's most basic dynamic link libraries (DLLs). The function of this directory is similar to the storing of DLL files on Windows. Almost all applications need to use these shared libraries.
/mnt	Temporarily mounts other file systems.
/opt	Stores additional software that is installed on the host.
/proc	A virtual directory, which is the mapping of the system memory. You can obtain the system information by directly accessing this directory.

- 按英文字母顺序讲解目录用途

Linux Directory Usage (2)

Directory	Main Files and Their Functions
/root	This directory is the home directory of the system administrator, who is also called the superuser.
/sbin	**s** indicates the superuser. This directory stores the system management program used by the system administrator.
/srv	Stores the data that needs to be extracted after a service is started.
/tmp	Stores temporary files.
/usr	Many user applications and files are stored in this important directory, which is similar to the **program files** directory on Windows. **/usr/bin** is the application used by system users. **/usr/sbin** is an advanced management program and system daemon used by the superuser. **/usr/src** is the default directory for storing the kernel source code.
/var	Stores content that is constantly expanded, such as log files. You are advised to place frequently modified directories here.
/run	A temporary file system that stores the information generated after the system is started. The information is cleared or deleted when the system is restarted.

Linux File Paths

- When using the shell or invoking an application, specify the path of the invoked program.
- The path can be an absolute or relative path.
  - Absolute path: On Linux, an absolute path starts from **/** (also called the root directory). If a path starts from **/**, it must be an absolute path.
  - Relative path: The relative path is relative to the current directory.

File Command 1 - pwd

- The **pwd** command is used to print the current working directory.
- The **pwd** command has two options: **-L** and **–P**. The functions are similar to those of the **cd** command.
- **-L**: outputs the connection path when the directory is linked.
- **-P**: outputs the physical path.

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /var/log
[root@localhost log]# pwd
/var/log
[root@localhost log]#
```


- **pwd** stands for "print working directory". The following lists a comparison between the **-L** and **-P** options of the **pwd** command.
- $ cd /tmp
- $ ls -ld d1 link1 link2 link3
- drwxr-xr-x. 2 root root 6 Oct 17 15:56 d1
- lrwxrwxrwx. 1 root root 2 Oct 17 15:56 link1 -> d1
- lrwxrwxrwx. 1 root root 5 Oct 17 15:56 link2 -> link1
- lrwxrwxrwx. 1 root root 5 Oct 17 15:56 link3 -> link2
- $ cd /tmp/link3
- $ pwd
- /tmp/link3
- $ pwd -L
- /tmp/link3
- $ pwd -P
- /tmp/d1

File Command 2 - cd

- The **cd** command is used to change the current working directory.
- Syntax: **cd** *[dir]*
  - **cd /usr**: accesses the **/usr** directory.
  - **cd ..**: accesses the upper-level directory. Two dots indicate the parent directory.
  - **cd .**: accesses the current directory.
  - **cd**: accesses the home directory by default if no parameter is added.
  - **cd -**: accesses the previous directory. This command is used to quickly switch between two directories.
  - **cd ~**: accesses the home directory.

File Command 2 - Example

- Change the current working directory.

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /var/log
[root@localhost log]# pwd
/var/log
[root@localhost log]# cd ..
[root@localhost var]# pwd
/var
[root@localhost var]# cd ~
[root@localhost ~]# pwd
/root
```

File Command 3 - ls

- The **ls** command is one of the most frequently used Linux commands, and lists the content of a directory or file information. The output of this command is sorted by file name by default. If no target is specified, the content of the current directory is listed.
- Syntax: **ls** [*OPTION*]... [*FILE*]...
  - **-a**: displays all files and directories, including hidden files or directories whose names start with a dot (.).
  - **-l**: lists information such as the file type, permission, owner, and file size, in addition to the file name.
  - **-t**: lists files by creation time.
  - **-R**: lists any files in the current directory in sequence.

HUAWEI

- Usage:
  - ls — Lists all files and directories (except hidden files) in the current directory.
  - ls -l — Displays detailed information.
  - ls -A — Displays hidden files and directories, excluding those whose names start with a dot (.) or two dots (..).
  - ls -a — Displays hidden files and directories, including those whose names start with a dot (.) or two dots (..).
  - ls -t — Sorts by creation time.
  - ls -S — Sorts by size.
  - ls -X — Sorts by extension name.
  - ls -r — Sorts in reverse.
  - ls -1 — Displays one file per line.
  - ls -lh — Displays size information with the **-h** option in a human-readable format.
  - ls -x — Displays by row instead of by column by default.
  - ls -l /bin/bash — Lists information about a specified file.
  - ls -l / — Lists the contents of a specified directory.
  - ls -ld / — Displays directory information with the **-d** option, rather than the contents of the directory.
  - ls -l file1 file2 — Lists multiple files at the same time.

File Command 3 - Example

- Run the following commands to list all files (including hidden files) in the **/usr/local** directory, and sort them by creation time.

```
[root@localhost ~]# ls /usr/local/ -ahlt
total 48K
drwxr-xr-x. 12 root root 4.0K Jul 28 14:00 ..
drwxr-xr-x. 12 root root 4.0K Jul 28 14:00 .
drwxr-xr-x.  5 root root 4.0K Jul 28 14:00 share
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 bin
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 etc
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 games
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 include
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 lib
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 lib64
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 libexec
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 sbin
drwxr-xr-x.  2 root root 4.0K Mar 24 05:34 src
```

File Command 4 - mkdir

- The **mkdir** command is used to create a directory or folder.
- Syntax: **mkdir** [*OPTION*]... *DIRECTORY*...

```
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# mkdir my_dir_01
[root@localhost ~]# ls
anaconda-ks.cfg  my_dir_01
[root@localhost ~]# mkdir my_dir_02 my_dir_03
[root@localhost ~]# ls
anaconda-ks.cfg  my_dir_01  my_dir_02  my_dir_03
[root@localhost ~]# mkdir my_dir_04/sub_dir
mkdir: cannot create directory 'my_dir_04/sub_dir': No such file or directory
[root@localhost ~]# mkdir -p my_dir_04/sub_dir
[root@localhost ~]# ls
anaconda-ks.cfg  my_dir_01  my_dir_02  my_dir_03  my_dir_04
[root@localhost ~]#
```


- **mkdir** stands for "make directory". This command is used to create multiple directories at a time. If a directory already exists, an error is reported by default. The **-p** option can be used to enable the **mkdir** command to report no error in this case, which can also be used to automatically create a parent directory that does not exist.
- Usage:
  - mkdir dir1 Creates a directory named **dir1**.
  - mkdir dir1 dir2 Creates multiple directories.
  - mkdir -p dir1/dir2/dir3 Creates **dir1** and **dir2** if they do not exist.
  - mkdir -pv dir1/dir2 Displays the creation process with the **-v** option.

File Command 5 - touch

- The **touch** command is used to create an empty file.
- It can also be used to change the timestamp of a file.

```
[root@localhost ~]# ls
[root@localhost ~]# touch test01.log test02.log
[root@localhost ~]# ls -lt
total 0
-rw-------. 1 root root 0 Jul 29 15:06 test01.log
-rw-------. 1 root root 0 Jul 29 15:06 test02.log
[root@localhost ~]# touch -t 202001020304.05 test01.log
[root@localhost ~]# ls -lt
total 0
-rw-------. 1 root root 0 Jul 29 15:06 test02.log
-rw-------. 1 root root 0 Jan  2  2020 test01.log
[root@localhost ~]#
```


- The **touch** command can also be used to change the timestamp of a file.
  - Run the following command to change the timestamp of the file to the current time:
  - $touch file
  - Run the following command to change only the file access time:
  - $touch -a file
  - Run the following command to change the file content change time:
  - $touch -m file
  - Run the following command to set the timestamp of the file to the specified time:
  - $ touch -d "2020-01-17 17:14:10" file

File Command 6 - cp

- The **cp** command is used to copy files or directories. You can copy a single file or multiple files at a time. Exercise caution when running this command, as there is a risk of data loss.
- Syntax: **cp** *[OPTION]... SOURCE... DIRECTORY*
  - **-a**: copies the files of a directory while retaining the links and file attributes.
  - **-p**: copies the file content, modification time, and access permissions to the new file.
  - **-r**: copies all subdirectories and files in the source directory file.
  - **-l**: generates a link file but does not copy the file.

- **cp** [*OPTION*]... *SOURCE... DIRECTORY*: When multiple files are copied, the destination must be a directory.
- **cp** [*OPTION*]... **[-T]** *SOURCE DEST*: When a single file is copied, the destination can be a file or a directory. If the destination is a file, you can rename the new file. If the destination is a directory, the new file is placed in the directory and the name is the same as that of the source file. *DEST* can be a file or a directory. To specify *DEST* as a file, use the **-T** option. To specify *DEST* as a directory, add a slash (/) after *DEST*, that is, *DEST/*.
- **cp** [*OPTION*]... **-t** *DIRECTORY SOURCE...*: This command is rarely used, and is usually used together with the **xargs** command. If you want to copy all HTML files in multiple subdirectories of a directory to the **/data/html** directory, you can use either of the following methods:
  - find -name "*.html" -exec cp {} /data/html/ \;
  - find -name "*.html" | xargs cp -t /data/html/
  - The second method uses **xargs** and the **-t** option of the **cp** command to greatly reduce the number of **cp** processes to be started.

File Command 6 - Example

```
[root@localhost ~]# ls
test01.log  test02.log
[root@localhost ~]# cp /etc/passwd passwd.back
[root@localhost ~]# cp -r /var/log/audit ./
[root@localhost ~]# ls
audit  passwd.back  test01.log  test02.log
[root@localhost ~]# cp -s /etc/passwd passwd_link
[root@localhost ~]# ls
audit  passwd.back  passwd_link  test01.log  test02.log
[root@localhost ~]# ls -l
total 8
drwx------. 2 root root 4096 Jul 29 15:24 audit
-rw-------. 1 root root 2546 Jul 29 15:24 passwd.back
lrwxrwxrwx. 1 root root   11 Jul 29 15:25 passwd_link -> /etc/passwd
-rw-------. 1 root root    0 Jan  2  2020 test01.log
-rw-------. 1 root root    0 Jul 29 15:06 test02.log
[root@localhost ~]#
```


- cp f1 f2 Copies file **f1** and names the new file **f2**.
- cp f1 d1/ Copies **f1** to the **d1** directory. The name of the new file remains unchanged.
- cp f1 f2 f3 d1/ Copies multiple files to the same directory.
- cp -i f1 f2 If **f2** already exists, overwrites it and waits for user confirmation.
- cp -r d1 d2 Copies a directory with the **-r** option.
- cp -rv d1 d2 Displays the copy process.
- cp -rf d1 d2 Enables the **cp** command to delete the target file and try again, in cases where the existing target file cannot be opened.
- cp -a f1 f2 Copies block devices, character devices, and pipe files while retaining the attributes of the original file.

- As the **cp** command does not ask the user before overwriting files by default, Shell has made an alias for the **cp** command and added the **-i** option. The **-f** option in the **cp** command does not indicate forcible overwriting.

File Command 7 - mv

- The **mv** command is used to move a file or directory. Exercise caution when running this command, as there is a risk of data loss.
- If the source file and target file are in the same parent directory, the **mv** command is used to rename the file.
- Syntax: **mv** *[option] source file or directory target file or directory*
  - **-b**: backs up a file before overwriting it.
  - **-f**: forcibly overwrites the target file without asking the user.
  - **-i**: overwrites the target file at the destination after obtaining the user's consent.
  - **-u**: updates the target file only when the source file is newer than the target.

- The syntax of the **mv** command is the same as that of the **cp** command. The functions of the **-T** and **-t** options are also the same.
- The following options also have the same function as the **cp** command:
  - -i
  - -v
- The **-f** option enables the **mv** command to overwrite the target file without asking the user, while the **-i** option does the opposite.
- The **mv** command is similar to the **cp -a src dst** command. You need to copy the file to the destination and then delete the original file.
- As the **mv** command does not ask the user before overwriting files by default, Shell has made an alias for the **mv** command and added the **-i** option. If both **-i** and **-f** are provided, the option on the right takes effect.

File Command 7 - Example

- Change the name of the **test02.log** file to **test03.log**.
- Move the **statistics** file in the **mail** directory to the current directory.

```
[root@localhost ~]# ls
audit  passwd.back  passwd_link  test01.log  test02.log
[root@localhost ~]# mv test02.log test03.log
[root@localhost ~]# mv audit/audit.log ./
[root@localhost ~]# ls
audit  audit.log  passwd.back  passwd_link  test01.log  test03.log
[root@localhost ~]# ls audit/
audit.log.1
[root@localhost ~]# mv audit/ audit_back
[root@localhost ~]# ls
audit_back  audit.log  passwd.back  passwd_link  test01.log  test03.log
[root@localhost ~]#
```

File Command 8 - rm

- The rm command is used to delete a file or directory.
- Exercise caution when running this command, as it is not possible to completely restore files deleted in this manner. As the rm command does not move files to a place from which they can be restored, such as a "recycle bin", the deletion operation cannot be revoked.
- Syntax: rm [OPTION] file_or_dir
  - -f or --force: ignores the files that do not exist and does not display any message.
  - -I or --interactive: performs interactive deletion.
  - -r, -R, or --recursive: instructs rm to recursively delete all directories and subdirectories listed in the parameter.
  - -v or --verbose: displays the detailed procedure.

- As the **rm** command does not ask the user before overwriting files by default, Shell has made an alias for the **rm** command and added the **-i** option. If both **-i** and **-f** are provided, the option on the right takes effect.

File Command 8 - Example

- Delete the **test01.log** file after obtaining the user's consent.
- Forcibly delete the **test03.log** file.
- Delete the **mail.bak** directory and all files and directories within.

```
[root@localhost ~]# ls
audit_back  audit.log  passwd.back  passwd_link  test01.log  test03.log
[root@localhost ~]# rm test01.log
rm: remove regular empty file 'test01.log'? yes
[root@localhost ~]# rm -rf test03.log
[root@localhost ~]# rm -rf audit_back/
[root@localhost ~]# ls
audit.log  passwd.back  passwd_link
[root@localhost ~]#
```


- Common Options
  - rm f1 Deletes the **f1** file.
  - rm f1 f2 f3 Deletes multiple files.
  - rm -i f1 f2 The **-i** option enables the **rm** command to ask the user before deleting files.
  - rm -f f1 f2 The **-f** option enables the **rm** command to delete files without asking the user.
  - rm -r d1 Deletes the directory and all of its contents.
  - rm -rf d1 Deletes the directory and all of its contents.
  - rm -rv d1 Displays the deletion process.
- As the **rm** command does not ask the user before overwriting files by default, Shell has made an alias for the **rm** command and added the **-i** option. If both **-i** and **-f** are provided, the option on the right takes effect.

File Command 9 - cat

- The **cat** command is used to read the entire content of a file, or to combine multiple files into one.
- Syntax: **cat** [*OPTION*] [*FILE*]
  - **-A** or **--show-all**: equivalent to **-vET**.
  - **-b** or **--number-nonblank**: numbers a non-blank output line.
  - **-E** or **--show-ends**: displays **$** at the end of each line.
  - **-n** or **--number**: numbers all output lines. The value starts from **1**.

HUAWEI

- The **cat** command is used to read all contents of a file and write the contents to the standard output. Option **-A** is frequently used to display non-printable characters.
- Usage:
  - cat file — Reads all of a specified file's contents.
  - cat -A file — Displays non-printable characters.

File Command 9 - Example

- View the content of the **test01.log** and **test02.log** files, and combine the content of both into the **test03.log** file.

```
[root@localhost ~]# ls
audit.log  passwd.back  passwd_link  test01.log  test02.log
[root@localhost ~]# cat test01.log
This is a test!
[root@localhost ~]# cat -b test02.log
     1  This is a test too!
[root@localhost ~]# cat test01.log test02.log > test03.log
[root@localhost ~]# cat test03.log
This is a test!
This is a test too!
[root@localhost ~]#
```


HUAWEI

- Usage:
  - head -n 3 file — Reads the first three lines of the file.
  - head -c 3 file — Reads the first three bytes of a file.
  - head -n -3 file — Reads all lines except the last three in the file.
  - head -c -3 file — Reads all contents of a file except the last three bytes.

File Command 10 - head

- The **head** command is used to display the beginning of a file (the first 10 lines, by default).
- Syntax: **head** [*OPTION*] [*FILE*]
- Main options:
  - **-q**: hides the file name.
  - **-v**: displays the file name.
  - **-c**<*byte*>: displays the number of bytes.

File Command 10 - Example

- Display the first three lines of the **/etc/passwd** file.
- Display the content of the **/etc/passwd** file, excluding the last 20 lines.

```
[root@localhost ~]# head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
[root@localhost ~]# head -n -40 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
[root@localhost ~]#
```

File Command 11 - tail

- The **tail** command is used to read the tail of a file.
- Syntax: **tail** [*OPTION*]... [*FILE*]...
- Main options:
  - **-f**: reads data cyclically.
  - **-q**: does not display processing information.
  - **-v**: displays detailed processing information.
  - **-c**<*number*>: displays the number of bytes.
  - **-n**<*number*>: displays the number of lines.

File Command 11 - Example

- Read the last three lines of the **/etc/passwd** file and display the output of a ping operation in real time.

```
[root@localhost ~]# tail -n 3 /etc/passwd
tcpdump:x:72:72::/:/sbin/nologin
dbus:x:978:978:System Message Bus:/:/usr/sbin/nologin
openeuler:x:1000:1000:openEuler:/home/openeuler:/bin/bash
[root@localhost ~]# ping 192.168.110.245 > ping.log &
[1] 12865
[root@localhost ~]# tail -f ping.log
PING 192.168.110.245 (192.168.110.245) 56(84) bytes of data.
64 bytes from 192.168.110.245: icmp_seq=1 ttl=64 time=0.099 ms
64 bytes from 192.168.110.245: icmp_seq=2 ttl=64 time=0.113 ms
64 bytes from 192.168.110.245: icmp_seq=3 ttl=64 time=0.113 ms
64 bytes from 192.168.110.245: icmp_seq=4 ttl=64 time=0.114 ms
64 bytes from 192.168.110.245: icmp_seq=5 ttl=64 time=0.107 ms
64 bytes from 192.168.110.245: icmp_seq=6 ttl=64 time=0.117 ms
[root@localhost ~]#
```


- Usage:
  - tail -n 3 file Reads the last three lines of a file.
  - tail -c 3 file Reads the last three bytes of a file.
  - tail -n +3 file Reads data from the third line to the end of the file.
  - tail -c +3 file Reads data from the third byte to the end of the file.
  - tail -f file Traces the change of the file tail.

File Command 12 - more

- The **more** command displays further information, page by page, for users to read. Basically, you can press the space bar to go to the next page, press **b** to go back to the previous page, and search for strings. As **more** reads files from the front to the back, the entire file is loaded from the very start.
- Syntax: **more** [*OPTION*]... [*FILE*]...
  - **+n**: displays the information from the first *n* lines.
  - **-n**: defines the screen size as *n* lines.
  - **+/pattern**: searches for the character string **pattern** before the file is displayed, and then displays the character string from the first two lines.
  - **-c**: clears the screen from the top.

HUAWEI

- Control commands:
  - SPACE Next screen
  - RETURN Next line
  - q Exit

File Command 12 - Common Operation Commands

- You can perform interactive operations when reading file information by running the **more** command.
  - **Enter**: moves to the next *n* lines. This operation needs to be defined. By default, *n* is set to **1**.
  - **Ctrl**+**F**: scrolls down to the next screen.
  - Space bar: scrolls down to the next screen.
  - **Ctrl**+**B**: returns to the previous screen.
  - =: outputs the number of the current line.
  - V: invokes the vi editor.
  - **!** command: invokes and executes a shell.
  - q: exits the **more** command.

File Command 13 - less

- The **less** command is used to read content and display it on multiple screens. The **less** command is similar to the **more** command. Unlike **more**, which only moves downwards, **less** can move both upwards and downwards and does not load the entire file before displaying the its content.
- Syntax: **less** [*OPTION*]... [*FILE*]...
- Common operations:
  - /*Character string*: searches downwards for character strings.
  - ?*Character string*: searches upwards for character strings.
  - **Q**: exits the **less** command.
  - Space bar: scrolls to the next page.
  - **Enter**: scrolls to the next line.

HUAWEI

- Compared with **more**, **less** has more control functions. It can move upwards and downwards, and has a Vim-like UI. In many scenarios, it is more convenient to use the **less** command.
- Other operations:
  - **n**: repeats the previous search (related to **/** or **?**).
  - **N**: reversely repeats the previous search (related to **/** or **?**).
  - **b**: scrolls down to the next page.
  - **d**: scrolls down half a page.
  - **h**: displays the help information.
  - **Q**: exits the **less** command.
  - **u**: scrolls up half a page.
  - **y**: scrolls up to the previous line.
  - **pagedown**: scrolls down to the next page.
  - **pageup**: scrolls up to the previous page.

File Command 14 - find

- The **find** command is used to search for files in a specified directory.
- You can specify search conditions, such as file name, file type, user, and even timestamp.
- Syntax: **find** [*path*...] [*expression*]
  - **-name**: searches for files by file name.
  - **-perm**: searches for files by file permission.
  - **-user**: searches for files by file owner.
  - **-mtime -n +n**: searches for files by modification time.

- The **find** command has many options, which can be combined to implement complex and powerful search functions. For details about the options, see find(1) in the **man** document.
- Example:
  - find -name "*book*" Searches for files with **book** in the file names.
  - find -user mysql Searches for files where the UID is **mysql**.
  - find -size 0 Searches for files with a size of 0.
  - find -type l Searches for files with a file type of soft link.
  - find /etc -name "*passwd" Searches for information in the **/etc** directory.
  - find -empty Searches for empty files (including empty directories).
  - find -empty -delete Finds and deletes empty files.

File Command 14 - Example (1)

- Search for files by file name.

```
[root@localhost ~]# find /etc -name passwd
/etc/pam.d/passwd
/etc/raddb/mods-enabled/passwd
/etc/raddb/mods-available/passwd
/etc/passwd
[root@localhost ~]# find . -name "*.log"
./test01.log
./ping.log
./test02.log
./test03.log
./audit.log
[root@localhost ~]#
```

File Command 14 - Example (2)

- Search the **/var/log/anaconda** directory for the common files with a modification time earlier than the last seven days.

```
[root@localhost ~]# find /var/log/anaconda/ -type f -mtime +7
/var/log/anaconda/dnf.librepo.log
/var/log/anaconda/syslog
/var/log/anaconda/dbus.log
/var/log/anaconda/ks-script-cdcy5u0e.log
/var/log/anaconda/packaging.log
/var/log/anaconda/ifcfg.log
/var/log/anaconda/lvm.log
/var/log/anaconda/program.log
/var/log/anaconda/journal.log
/var/log/anaconda/hawkey.log
/var/log/anaconda/anaconda.log
/var/log/anaconda/storage.log
/var/log/anaconda/X.log
[root@localhost ~]#
```


- File types:
  - **d**: directory
  - **c**: font device file
  - **b**: block device file
  - **p**: named storage
  - **f**: general file
  - **l**: symbolic link

File Command 15 - gzip

- **gzip** is a command used to compress and decompress files on Linux.
- Specifically, **gzip** can be used to compress large, rarely-used files to save disk space.
- Syntax: **gzip** [*option*] [*file or directory*]
  - **-d**, **--decompress**, or **----uncompress**: decompresses a package.
  - **-f** or **--force**: forcibly compresses a file, regardless of whether the file name exists or whether the file is a symbolic link.
  - **-l** or **--list**: lists information about compressed files.
  - **-r** or **--recursive**: recursively processes all files and subdirectories in a specified directory.
  - **-v** or **--verbose**: displays the command execution process.

- According to statistics, the **gzip** command has a compression ratio of 60% to 70% for text files. After a file is compressed using **gzip**, the file name is suffixed with **.gz**.

File Command 15 - Example

- Compress, view, and decompress files.

```
[root@localhost ~]# ls
audit.log    passwd_link  test01.log  test03.log
passwd.back  ping.log     test02.log
[root@localhost ~]# gzip *.log
[root@localhost ~]# ls
audit.log.gz  passwd_link  test01.log.gz  test03.log.gz
passwd.back   ping.log.gz  test02.log.gz
[root@localhost ~]# gzip -l test01.log.gz
         compressed        uncompressed  ratio uncompressed_name
                 45                  16   0.0% test01.log
[root@localhost ~]# gzip -dv test01.log.gz
test01.log.gz:    0.0% -- replaced with test01.log
[root@localhost ~]# ls
audit.log.gz  passwd_link  test01.log     test03.log.gz
passwd.back   ping.log.gz  test02.log.gz
[root@localhost ~]#
```

File Command 16 - tar

- The **tar** command is used to pack files. You can pack multiple files into a package to facilitate data transfers.
- Syntax: **tar** [*OPTION*...] [*FILE*]
  - **-c**: creates a compressed file.
  - **-x**: extracts files from a compressed file.
  - **-t**: displays the content of a compressed file.
  - **-z**: supports **gzip** decompression.
  - **-j**: supports **bzip2** file decompression.
  - **-v**: displays the operation process.

- The **-f** option is followed by the name of a **tar** package. If a minus sign (-) is used, it indicates the standard output (when creating a package) or standard input (when decompressing or viewing a package).
- The **tar** command is usually used together with options **-z**, **-j**, and **-J**, which corresponds to the **gzip**, **bzip2**, and **xz** compression tools, respectively. After a compression option is specified, the **tar** command starts the corresponding compression tool to compress or decompress data, and then transmits data through the pipe and compression tool. The **tar cf - dir1 | gzip > dir1.tar.gz** command is equivalent to the **tar czf dir1.tar.gz dir1** command.
- Usage:
  - tar cf ball.tar dir1 Packs the **dir1** directory and all of its contents.
  - tar tf ball.tar Lists the contents in the package.
  - tar xf ball.tar Decompresses the package to the current directory.
  - tar czf ball.tar.gz dir1 Compresses files using the **gzip** tool.
  - tar cjf ball.tar.bz2 dir1 Compresses files using the **bzip2** tool.
  - tar cJf ball.tar.xz dir1 Compresses files using the **xz** tool.
  - tar xf ball.tar -C /tmp Decompresses to the **/tmp** directory (the current directory by default).
  - tar xvf ball.tar Displays the process with the **-v** option.

File Command 16 - Example

- Compress files.
- Query the files in a package and decompress the package to the specified directory.

```
[root@localhost ~]# ls
passwd  test01.log  test02.log
[root@localhost ~]# tar -cf log.tar *.log
[root@localhost ~]# tar -zcf log.tar.gz *.log
[root@localhost ~]# ls
log.tar  log.tar.gz  passwd  test01.log  test02.log
[root@localhost ~]# tar -ztvf log.tar.gz
-rw------- root/root         0 2020-07-29 17:47 test01.log
-rw------- root/root         0 2020-07-29 17:47 test02.log
[root@localhost ~]# mkdir log
[root@localhost ~]# tar -zxf log.tar.gz -C ./log/
[root@localhost ~]# ls
log  log.tar  log.tar.gz  passwd  test01.log  test02.log
[root@localhost ~]# ls log
test01.log  test02.log
[root@localhost ~]#
```

File Command 17 - ln (1)

- The **ln** command is used to create a link file.
- There are two types of links on Linux: soft link (also known as symbolic link) and hard link.

Soft Link	Hard Link
A path, similar to a Windows shortcut	A file copy, which does not occupy the actual space
A link, which becomes invalid after the source file is deleted	A link, which has no impact on the source file after being deleted
Linking to a directory is supported	Linking to a directory is not supported
Cross-file system linking is supported	Cross-file system linking is not supported

- **ln** creates a synchronous link for a file in another location. If the same file is required in different directories, it is not necessary to place it in each one. Instead, you can place the file in a fixed directory and run the **ln** command to link to it in other directories, thereby conserving disk space.

File Command 17 - ln (2)

- If the **ln** command does not contain any option, a hard link is created by default.
- Syntax: **ln** [*-f*] [*-n*] [*-s*] *SourceFile* [*TargetFile*]
  - **-b**: deletes and overwrites the existing link.
  - **-d**: allows the superuser to create hard links to directories.
  - **-f**: forcibly executes the command.
  - **-i**: indicates the interactive mode. If the file exists, the system prompts you to overwrite it.
  - **-n**: regards symbolic links as common directories.
  - **-s**: soft link (symbolic link).

File Command 17 - Example

- Create a link, delete the source file, and restore the source file. Then, check the link status.

```
[root@localhost ~]# ls
passwd
[root@localhost ~]# ln passwd link_h_password
[root@localhost ~]# ln -s passwd link_s_password
[root@localhost ~]# ls -l
total 8
-rw-------. 2 root root 2546 Jul 29 15:24 link_h_password
lrwxrwxrwx. 1 root root    6 Jul 29 17:41 link_s_password -> passwd
-rw-------. 2 root root 2546 Jul 29 15:24 passwd
[root@localhost ~]# rm -f passwd
[root@localhost ~]# ls -l
total 4
-rw-------. 1 root root 2546 Jul 29 15:24 link_h_password
lrwxrwxrwx. 1 root root    6 Jul 29 17:41 link_s_password -> passwd
[root@localhost ~]# cp /etc/passwd passwd
[root@localhost ~]# ls -l
total 8
-rw-------. 1 root root 2546 Jul 29 15:24 link_h_password
lrwxrwxrwx. 1 root root    6 Jul 29 17:41 link_s_password -> passwd
-rw-------. 1 root root 2546 Jul 29 17:41 passwd
[root@localhost ~]#
```

Contents

Help Command - man

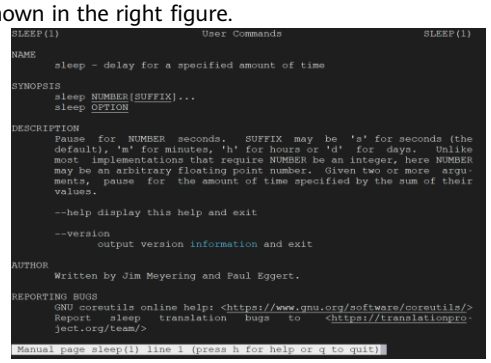
- **man** is used to view the manual, which is classified into the following nine types:

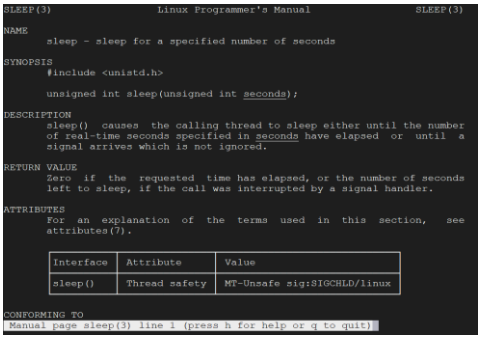
No.	Description
1	Commands or programs that can be operated by users in the shell
2	Functions and tools that can be invoked by the system kernel
3	Common functions and function libraries
4	Device document description, which is usually in the **/dev** directory
5	File format and conventions
6	Games
7	Miscellaneous (including macros and conventions)
8	System management commands (applicable only to the **root** user)
9	Kernel routines (non-standard)

- Linux provides various documents, which can be accessed by executing commands such as **man**, **info**, and **txt**.
- Documents that are accessed by running **man** are classified into eight types according to the standard classification method. The four common types are numbered 1, 4, 5, and 8, respectively. If different types of documents use the same name, specify the file type. For example:
  - man 1 passwd
  - man 5 passwd
- Typically, we can run the following commands to search for **man** documents:
  - man -k KEYWORD
  - find /usr/share/man -iname "*KEYWORD*"

Help Command - Example

- Search results by chapter number in the manual. For example, enter **man sleep**, as shown in the left figure.
- By default, only command manuals are displayed. To view library functions, enter **man 3 sleep**, as shown in the right figure.





HUAWEI

- Linux provides various documents, which can be accessed by executing commands such as **man**, **info**, and **txt**.
- Documents that are accessed by running **man** are classified into eight types according to the standard classification method. The four common types are numbered 1, 4, 5, and 8, respectively. If different types of documents use the same name, specify the file type. For example:
  - man 1 passwd
  - man 5 passwd
- Typically, we can run the following commands to search for **man** documents:
  - man -k KEYWORD
  - find /usr/share/man -iname "*KEYWORD*"

Help Command - help

- Due to the vast number of commands in the Linux system, it is almost impossible to remember them all. However, you can run the **help** command to obtain detailed information.
- Syntax:
  - **help** *[option] [command]*
- The options are as follows:
  - **-d**: displays the brief description of the command.
  - **-s**: displays the brief description of the command syntax.
- See the following example:

```
[root@localhost ~]# help pwd
pwd: pwd [-LP]
    Print the name of the current working directory.
    Options:
      -L        print the value of $PWD if it names the current working directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory cannot be read.
```

Quiz

1. （True or false）When a user wants to use mv command to overwrite the target file at the destination after obtaining the his consent, he should add “-i” option into the command.

   A. True

   B. False

- Answer:

  1. A

Summary

- This document describes the basic operations of the Linux command line, and the basic commands for logging in to the Linux system, starting and shutting down the system, and operating files.

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

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**
  - **Insert**
  - **Visual**
- 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.

- Basic modes:
  - **Normal**: This mode allows the use of editor commands, such as those for moving a cursor and deleting text. It is the default mode after Vim is started, which is the opposite of what many new users expect (most editors are in **Insert** mode by default). Vim's powerful editing capabilities come from its **Normal**-mode commands, each of which requires an operator at the end. For example, the **dd** command is used to delete the current line, and you can replace the second **d** with another move command (for example, **j**) to move one line down and delete both the current and next lines. In addition, you can specify the number of times that a command can be repeated. For example, **2dd** indicates that the command is repeated twice, and the effect is the same as that of **dj**. After learning various move and jump commands and Normal-mode edit commands, and understanding how to flexibly combine these commands, you can edit text more efficiently compared with using modeless editors. There are many ways to enter **Insert** mode from **Normal** mode. For example, press **a** (append) or **i** (insert).
  - **Insert**: In this mode, most keys are used to insert text into the text buffer. New users often want the text editor to remain in this mode throughout the editing process. You can press **ESC** to enter **Normal** mode from **Insert** mode.
  - **Visual**: This mode is similar to **Normal** mode. The main difference is that using a move command will enlarge the highlighted text area, which can be a character, a line, or a piece of text. A non-move command is executed on the highlighted area. Vim's text object can also be used in this mode, similar to the move command.

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.
  - **+***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.

- For more Vim shortcut operations, visit https://blog.csdn.net/u011365893/article/details/17139361.

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.

- 更多vim的快捷操作可以参考：
  https://blog.csdn.net/u011365893/article/details/17139361

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
  - **: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.

- Modifying a file is a high-risk operation. If you have to modify a file, especially a configuration file, you are advised to save a copy before the modification. If you do not need to modify a file, do not run the **wq** or **wq!** command to exit.

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.

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:
  - **-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.

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 (,).

- Specified characters or character ranges of **cut**:
  - *N*-: starts from the *N*th byte, character, or field to the end.
  - *N*-*M*: starts from the *N*th byte, character, or field to the *M*th byte, character, or field (inclusive).
  - -*M*: starts from the first byte, character, or field to the *M*th byte, character, or field (inclusive).

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.

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.

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.

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.

Quiz

1. (Single-answer)How many lines of a file are displayed by default when using the head command ?

   A. 5

   B. 10

   C. 15

   D. 20

- Answer:
  - B

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.

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

User and Permission Management

HUAWEI

Foreword

This document describes user and permission management on openEuler. After learning this document, you will understand the basic concepts of users and user groups on openEuler, and be familiar with commands for creating, deleting, and modifying files and directories. You will also learn about file permission configuration and specific operation commands.

Objectives

- After completing this course, you will understand:
  - Basic concepts about users and user groups
  - Command line operations related to files and directories
  - File permission configuration and related command operations
  - Special control methods for file access

Contents

Basic Concepts About Users

- Linux is a multi-user operating system.
  - All users who wish to use system resources must apply for an account from a system administrator, and then use it to log in.
  - Multiple users can be created on the system, and can log in to the same system at the same time to perform different tasks without affecting each other.
- User
  - A user can be viewed as a set of permissions to obtain system resources.
  - Each user is assigned a unique user ID (UID).

- A user group is a logical set of users with the same features. Sometimes, multiple users need to have the same permissions, for example, the permission to view and modify a file. One way is to grant individual permission to each user, but if there are say 10 users, this is not reasonable. Instead, you can create a group and grant it the permission to view and modify a file. Then, add all users who need to access the file to the group, who will gain all the same permissions. User groups help manage users and control access permissions on the Linux operating system, greatly simplifying user management.

UID

- UID is a unique user identification that can also identify the user type. When a user logs in to a system, the UID (not the user name) is used to identify the user type.
  - Superuser: also called a **root** user whose UID is **0**. A superuser has full control over a system, and can modify and delete files, as well as running commands. As such, exercise caution when delegating the **root** user.
  - Common user: also called a normal or regular user whose UID ranges from 1000 to 60000. Common users can access and modify files in their own directories, and access authorized files.
  - Virtual user: also called a system user whose UID ranges from 1 to 999. Virtual users can log in to a system without passwords, facilitating system management.

- A common user's ID is assigned from 1000 onwards by default, meaning that common users' UIDs are generally above 1000. The UID of the **root** user is **0**. The **root** user has full control over the system, and can modify or delete any file, and run any command. Logging in as a root user can be risky for the system, as the root user can delete all file systems even when a system is running properly, causing irreparable damage. Therefore, exercise caution when logging in to the system as the root user.
- The **root** user can read, modify, or delete files or directories beyond the scope of permissions of any other user or user group (within the normal scope of system permissions); execute and terminate executable programs; add, create, and remove hardware devices; change the owner and permission of files and directories to meet system management requirements (because the **root** user has the highest privileges on the system).

Distinguishing User Types

- You can determine whether a user is the superuser, a common user, or a virtual user by viewing the UID.
- Run the **id** [*option*] [*user_name*] command to view a UID.
- Main options:
  - **-u**, **-user**: displays only a valid UID.
  - **-n**, **-name**: displays a name instead of a number for **-ugG**.
  - **-r**, **-real**: displays a real ID instead of a valid ID for **-ugG**.
- UID **0** indicates the superuser (**root** user). UIDs from 1000 to 60000 indicate common users. UIDs from 1 to 999 indicate virtual users (system users).

- Users in Linux user groups are classified into the following types:
  - User **root:** has all system permissions. The UID is **0**.
  - Common user: has limited permissions to use a system. The UID ranges from 500 to 60000.
  - System user: ensures proper system running. Generally, system users do not need a password to log in to the system. The UID ranges from 1 to 499.
- User group classification:
  - Common user group: Multiple users can be added to a common user group.
  - System group: Some system users are added to the system group.
  - Private group (also called basic group): When a user is created, if a group to which the user belongs is not specified, a user private group is defined for the user. The group name is the same as the user name. Note that a private group can be changed to a common group by adding another user.
  - User groups help manage users and control access permissions on the Linux operating system, greatly simplifying user management.

Contents

Managing Users

- On Linux, each common user has an account containing information such as a user name, password, and home directory. In addition, there are some special users created by the system. The most important user has the administrator account, and is the superuser **root** by default.
- Users can run commands to create, modify, and delete user files, as well as changing passwords.

Creating a User - useradd

- The **useradd** command is used to create a user account and save it in the **/etc/passwd** file.
- Syntax: **useradd** [*options*] *user_name*

  Main options:

  - **-u** specifies the user's UID.
  - **-o** is used together with **-u** to allow duplicate UIDs.
  - **-g** specifies a basic group to which the user belongs. It can be a user group name or a group ID (GID) if the group exists.
  - **-d** specifies the home directory for the user and automatically creates the home directory.
  - **-s** specifies the default shell program of a user.
  - **-D** displays or changes the default configuration.

- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Creating a User - Example

- Create a user named **user**.
  - The command is **useradd user**.

```
[root@localhost ~]# useradd user
```

  - Run the **cat /etc/passwd** command to check whether a user has been successfully created. The command output shows that the user has been created.

```
[root@localhost ~]# cat /etc/passwd

dbus:x:980:980:System Message Bus:/:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
test02:x:1001:1001::/home/test02:/bin/bash
user:x:1002:1004::/home/user:/bin/bash
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Modifying a User - usermod

- The **usermod** command is used to modify the information about a user account.
- Syntax: **usermod** [*options*] *user_name*

  Main options:

  - **-u** modifies the user's UID.
  - **-g** modifies the user group to which the user belongs.
  - **-l** modifies the user account name.
  - **-d** modifies the user's home directory.
  - **-s** modifies the user's default shell program.

HUAWEI

- Basic information about the user is stored in the database **/etc/passwd**. The record consists of seven fields separated by colons:
  - name:password:uid:gid:comment:home:shell
  - You can run the **usermod** command to modify six fields.
- Other options:
  - **-c**<*Remarks*> Modifies remarks about a user account.
  - **-d** <*Login directory*> Modifies the user's login directory.
  - **-e**<*Validity period*> Modifies the validity period of an account.
  - **-f**<*Grace period*> Modifies the number of days before an account is disabled once the password expires.
  - **-g**<*Group*> Modifies the group to which a user belongs.
  - **-G**<*Group*> Modifies any additional groups to which a user belongs.
  - **-l**<*Account name*> Modifies the user account name.
  - **-L** Locks the user's password to invalidate it.
  - **-s**<*shell*> Modifies the shell used by a user after login.
  - **-u**<*UID*> Modifies the user ID.
  - **-U** Unlocks the password.

Modifying a User - Example

- Modify a user named **user**:
  - Run the **id user** command to view the user's UID before modification.

```
[root@localhost ~]# id user

uid=1002(user) gid=1004(user) groups=1004(user)
```

  - Run the **usermod -u 1003 user** command to modify the UID to **1003**. The command output shows that the UID has been modified.

```
[root@localhost ~]# usermod -u 1003 user

[root@localhost ~]# id user
uid=1003(user) gid=1004(user) groups=1004(user)
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Deleting a User - userdel

- The **userdel** command is used to delete a specified user and related files.
- Syntax: **userdel** [*options*] *user_name*

  Main options:

  - **-f** forcibly deletes a user account even if the user is logged in.
  - **-r** deletes a user and all related files.
  - **-h** displays the help information about a command.

  (Using the **userdel** command to delete a specified user and user-related files modifies the user account system files.)

- If you need to delete a user home directory and all contents in the directory, run the **userdel** command with the **-r** option to delete them recursively.
- You are not advised to directly delete a user who has logged in to the system. To forcibly delete a user, run the **userdel -f** *Test* command.
- If you do not add any options when entering the deletion command, only the user account is deleted, but not the related files.

Deleting a User - Example

- Delete a test user:
  - Run the **cat /etc/passwd** command to view a user before deletion.

```
[root@localhost ~]# cat /etc/passwd

dbus:x:980:980:System Message Bus:/:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
test02:x:1001:1001::/home/test02:/bin/bash
user:x:1002:1004::/home/user:/bin/bash
```

  - Run the **userdel user** command to delete a test user. The command output shows that the user has been deleted.

```
[root@localhost ~]# userdel user

[root@localhost ~]# cat /etc/passwd
dbus:x:980:980:System Message Bus:/:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
test02:x:1001:1001::/home/test02:/bin/bash
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Changing a User Password - passwd

- The **passwd** command is used to change a user password.
- Syntax: **passwd** [*OPTION*...] *user_name*

  Main options:

  - **-n** sets the minimum number of days between password change.
  - **-x** sets the maximum number of days between password change.
  - **-w** sets the number of days of warning before password expires.
  - **-i** sets the number of days before an account is disabled once a password expires.
  - **-d** deletes the user password.
  - **-S** displays the user password information.

  (The **root** user can change any user's password. Common users can change only their own passwords.)

- Note: Only the **root** user can execute the **passwd** command.
- Other options:
  - **-f** Forcibly executes a command.
  - **-k** Retains a user who is about to expire. After the validity period expires, the user can still use the system.
  - **-g** Changes a group password.
  - **-e** Forces a password to expire.
  - **-l** Locks a user password. The locked user cannot log in to the system.
  - **-u** Unlocks a user password.

Changing a User Password - Example

- Change the password of a test user:
  - Run the **/etc/shadow** command to check the user password before the change. The output shows that the password has not set and is displayed as **!**.

```
[root@localhost ~]# cat /etc/shadow

user:!:18421:0:99999:7:::
```

  - Run the **passwd user** command to change the user password.

```
[root@localhost ~]# passwd user
Changing password for user user.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

  - Run the **/etc/shadow** command again to check whether the password has been changed successfully. The command output shows that the password has been changed successfully.

```
[root@localhost ~]# cat /etc/shadow

user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyaSYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAdbEqQcnQuDoWT7ocu3Ss.zzWSrEb6
cZ6Ae6b2EN/:18421:0:99999:7:::
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Contents

Basic Concepts About User Groups

- User group:
  - A user group is a logical set of users with the same features. These users in one group can have the same permissions, which facilitates management.
  - Each user has a private group.
  - All users in the same group can share files in the group.
  - Each user group is assigned a unique group ID (GID).

- A user group is a logical set of users with the same features. Sometimes, multiple users need to have the same permissions, for example, the permission to view and modify a file. One way is to grant individual permission to each user, but if there are say 10 users, this is not reasonable. Instead, you can create a group and grant it the permission to view and modify a file. Then, add all users who need to access the file to the group, who will gain all the same permissions. User groups help manage users and control access permissions on the Linux operating system, greatly simplifying user management.

GID

- A GID is similar to a UID, and is a unique identifier of a user group in a system.
  - When an account is added, a group with the same name as the user is created by default. The UID and GID are the same.
  - UID 0 is assigned to the superuser and GID 0 is assigned to a user group that has the superuser (that is, the **root** user group).
  - The system reserves some small GIDs for virtual users (also called system users).
- You can run the **id** [*option*] [*user_name*] command to view the GID and the number of users in a group.

- Each system reserves a different number of GIDs. For example, Fedora reserves 500 GIDs.
- When directories and files are created, the default user group is used.

User Group Classification

- Common user group: Multiple users can be added to a common user group.
- System group: Some system users are added to the system group.
- Private group (also called basic group): When a user is created, if a group to which the user belongs is not specified, a user private group is defined for the user. The group name is the same as the user name.

- Users in Linux user groups are classified into the following types:
  - Administrator **root**: has all system permissions. The UID is **0**.
  - Common user: has limited permissions to use a system. The UID ranges from 500 to 60000.
  - System user: ensures proper system running. Generally, system users do not need a password to log in to the system. The UID ranges from 1 to 499.
- Note that a private group can be changed to a common group by adding another user.
- User groups help manage users and control access permissions on the Linux operating system, greatly simplifying user management.

Relationships Between Users and User Groups

- One-to-one: A user is the only member in one group.
- One-to-many: A user belongs to multiple user groups and has their permissions.
- Many-to-one: Multiple users belong to a user group, and have that group's permissions.
- Many-to-many: Multiple users belong to multiple user groups, and have relevant permissions based on the above rules.

- One-to-one: A user is the only member in one group.
- Many-to-one: Multiple users belong to one group and do not belong to any other user groups. For example, the users **beinan** and **linuxsir** belong only to the **beinan** user group.
- One-to-many: A user is a member of multiple user groups. For example, **beinan** is a member of the **root** group, the **linuxsir** user group, and the **adm** user group.
- Many-to-many: Multiple users belong to multiple user groups (and can belong to the same group). This relationship is based on the other three, and follows the same concepts.

Contents

Managing User Groups

- As the number of users increases, user permission control and system security management are becoming more complex. But having user groups means each user is associated with at least one user group, which facilitates permission management.
- User and user group management is an important part of system security management. You can run commands to create, modify, and delete user group files, and associate users with user groups.

Creating a Group - groupadd

- The **groupadd** command is used to create a user group and add the new group information to a system file.
- Syntax: **groupadd** [*options*] *group_name*

  Main options:

  - **-f** exits if the group already exists.
  - **-g** assigns a GID to the new user group.
  - **-h** shows help information and exits.
  - **-o** allows duplicate GIDs.
  - **-p** assigns an encrypted password for the new user group.
  - **-r** creates a system account.

- Other options:
  - **-R**, **--root CHROOT_DIR** Directory specified by **chroot**
  - **-P**, **--prefix PREFIX_DIR** Directory prefix
  - **-K**, **--key KEY=VALUE** The default value in **/etc/login.defs** is not used.

Creating a Group - Example

- Create a test user group:
  - Run the **groupadd usergroup** command to create a test user group.

```
[root@localhost ~]# groupadd usergroup
```

  - Run the **cat /etc/group** command to check whether the user group has been created successfully. The command output shows that a new test group has been successfully created.

```
[root@localhost ~]# cat /etc/group

test04:x:1003:
user:x:1004:
usergroup:x:1005:
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Modifying a Group - groupmod

- The **groupmod** command is used to change a group identifier or group name.
- Syntax: **groupmod** [*options*] *group_name*
- Main options:
  - **-g** sets a new GID.
  - **-h** shows help information and exits.
  - **-n** sets a new group name.
  - **-o** allows duplicate GIDs.
  - **-p** modifies the encrypted password.

Modifying a Group - Example

- Modify a test user group GID:
  - Run the **id user** command to view the original test user GID.

```
[root@localhost ~]# id user

uid=1002(user) gid=1004(user) groups=1004(user)
```

  - Run the **groupmod –g 1006 user** command to modify the GID of the test user group to **1002**. The command output shows that the GID has been successfully modified.

```
[root@localhost ~]# groupmod -g 1006 user

[root@localhost ~]# id user
uid=1002(user) gid=1006(user) groups=1006(user)
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Deleting a Group - groupdel

- The **groupdel** command is used to delete a user group from the system. If a user group contains users, delete users before deleting the group.
- Syntax: **groupdel** [*options*] *group_name*
- Main options:
  - **-f** deletes a group even if it is the primary group of a user.
  - **-h** shows help information and exits.

- Function: This command is used to delete a specified user group.

Deleting a Group - Example

- Delete a test user group:
  - Run the **cat /etc/group** command to view a test user group.

```
[root@localhost ~]# cat /etc/group

test04:x:1003:
user:x:1006:
usergroup:x:1005:
```

  - Run the **groupdel usergroup** command to delete the test user group. Then, run the **cat /etc/group** command to check whether the test user group has been deleted.

```
[root@localhost ~]# groupdel usergroup

[root@localhost ~]# cat /etc/group
test04:x:1003:
user:x:1006:
```


HUAWEI

- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

Associating a User with a Group - gpasswd

- The **gpasswd** command is used to add or delete a user from a group.
- Syntax: **gpasswd** [*option*] *group_name*
- Main options:
  - **-a** adds a user to a group.
  - **-d** deletes a user from a group.
  - **-M** sets a group member list.
  - **-A** sets a group administrator list.
  - **-r** removes the group password.
  - **-R** restricts group access to members only.
  - **-Q** indicates a directory specified by **chroot**.

HUAWEI

- You can run the **usermod -G** *group_name user_name* command to add a user to a specified group. However, this will remove the user from any previous groups.
- If you want to add a user to a group and retain the previous groups, run the **gpasswd** command to add users.
- **chroot** changes the root directory. On the Linux system, the default directory structure starts from the **/root** directory. **chroot** changes the system directory structure to a specified directory.
- Advantages of **chroot**:
  - Enhances system security, and restricts user permissions.
  - Establishes a system directory structure that is isolated from the original system to facilitate user development.
  - Switches the root directory of a system, and provides Linux bootloader and system recovery.

Associating a User with a Group - Example

- Associate a test user with a test group.
  - Run the **gpasswd –a user usergroup** command to add a user to a user group.

```
[root@localhost ~]# gpasswd -a user usergroup

Adding user user to group usergroup
```

  - Run the **groups user** command. The command output shows that the user has been added to the group.

```
[root@localhost ~]# groups user

user : user usergroup
```


- Usage:
  - **useradd -D** Displays the default value used to create a user.
  - **useradd -D -g 500** Changes the default value used to create a user.
  - **useradd user1** Creates a user named **user1**.
  - **useradd -m user1** Creates a home directory for a user.
  - **useradd -M user1** Does not create a home directory for a user.
  - **useradd -d / user1** Specifies a home directory for a new user as a root directory.
  - **useradd -u 501 user1** Specifies the UID for a new user.
  - **useradd -g g1 user1** Specifies the GID for a new user. The group must already exist.
  - **useradd -G g1,g2,g3 user1** Adds users to additional groups g1, g2, and g3.
  - **useradd -o -u 100 user1** **-o** allows duplicate UIDs.
  - **useradd -s /bin/python user1** Uses the specified shell program.
- The configuration file **/etc/login.defs** also affects parameters for creating a user.

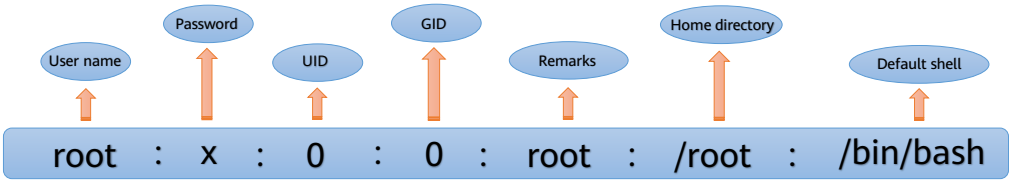
Contents

Files Associated with Users on openEuler

- In the openEuler directory, there are two types of files related to user information management:
- **/etc/passwd**: user account information file.
  - This file saves the main information about all users in a system. Each line represents a record.
  - Each record defines various user attributes.
- **/etc/shadow**: encrypted user account information file (also called shadow file).
  - This file stores user passwords in a system.
  - All users can read the **/etc/passwd** file, which may cause password leakage. Therefore, the password information is separated from the **/etc/passwd** file and stored in the **/etc/shadow** file.

/etc/passwd File

- Each line in the **/etc/passwd** file consists of seven fields separated by colons (:).



- The Linux shell is a kernel-based command line interpreter (CLI) for users, which is presented as an interface for user login. The Linux shells include sh, csh, ksh, tcsh, and bash.

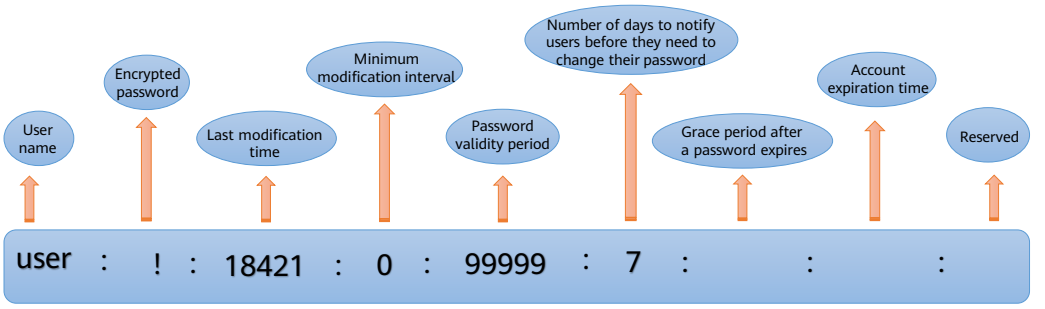
/etc/passwd File-related Parameters

No.	Description
1	A username, which can contain uppercase letters, lowercase letters, digits, minus signs (–), periods (.), and underscores (_). Other characters are invalid.
2	An encrypted password, which is represented by **X** in this field. For details about the password, see the **/etc/shadow** file.
3	A UID, which is used to identify a user and determine the user type.
4	A GID, which corresponds to the group information in **/etc/group** and is used to manage users by group.
5	Remarks, which contain attributes such as the user information.
6	Home directory, where a user is located during login. This field can be customized.
7	Default shell, which is used to transfer instructions issued by users to a kernel.

- **/etc/passwd has** seven fields:
  - Field 1 specifies a **username**. For example, **root** is a username. A username contains only uppercase letters, lowercase letters, digits, minus signs (cannot be the first character), periods, and underscores. You are not advised to use periods or minus (-) signs in usernames, especially as the first character.
  - Field 2 stores an encrypted user password. In earlier versions, the Lunix system password was stored in this directory. However, for security purposes, the password is now stored in the **/etc/shadow** file. Here, **x** corresponds to the password in the **/etc/shadow** file.
  - Field 3 specifies the UID. The system identifies users based on this number. Generally, the value of UID ranges from 0 to 65535, even though the value can be 4294967294. **0** indicates the **root** user. You can change the test user ID to **0**, in which case the system considers that the **root** and test users belong to the same account. Values 1 to 499 are reserved for system users. A UID of a common user starts from 500, so if you create a common user, you will see that their UID is greater than or equal to 500.

/etc/shadow File

- Only the superuser (**root**) has read permission on the **/etc/shadow** file, which ensures user password security.
- A password protected by **/etc/shadow** is displayed as **X** in a user record of the **/etc/passwd** file.
- In the **/etc/shadow** file, each line of record also indicates a user and there are 9 fields separated by colons (:).



HUAWEI

- When a system is installed, shadow protection is enabled by default. If it is not enabled, you can run the **pwconv** command to enable shadow protection. To disable it, run the **pwunconv** command. (Only the **root** user can run these commands.)

/etc/shadow File-related Parameters

No.	Description
1	A user name, which means the same as the user name in the **/etc/passwd** file.
2	An encrypted password. If the value of this field is ***** or **!**, the account cannot be used to log in to the system.
3	Time when the password was last changed.
4	Minimum modification interval. If this field is set to **0**, a password can be changed at any time.
5	Validity period of a password. Users must periodically change their passwords to improve system security.
6	Number of days to notify users before they need to change their password. When a user password is about to expire, a warning message is displayed to prompt the user to change the password.
7	Grace period after a password expires. If a password is not changed within the grace period, the user is disabled.
8	User expiration date. After a password expires, the user is no longer a valid user and cannot log in to the system.
9	This field is reserved and is left blank for future use.

(Note: The **/etc/shadow** file is automatically generated by the **pwconv** command based on data in the **/etc/passwd** file.)

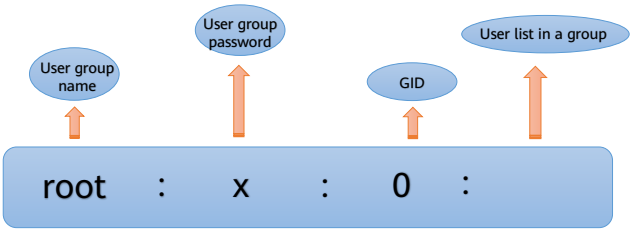
- What should I do if I forget my account and password? Can I use this account again? There is a simple solution. You can use the **root** account to reset a specified account password without knowing its password. (Use the **passwd** command as the **root** user.)
- If you forget the password of the **root user**, restart the system to enter single-user mode. The system provides the corresponding bash interface. In this case, you can run the **passwd** command to change the password of the **root** user. You can also mount the root directory to modify the **/etc/shadow** file, thereby clearing the root user's password. In this way, the **root** user can log in to the system without a password. You are advised to run the **passwd** command to configure the **root** user's password.
- There are three types of encrypted passwords: character string, asterisk (*), and double exclamation marks (!!). A character string is an encrypted password file. An asterisk (*) indicates that an account is locked. Double exclamation marks (!!) indicate that the password has expired. A string may start with **6,**
- **1, 2, or 5,** indicating that the string is encrypted using SHA-512, MD5, Blowfish, or SHA-256, respectively.

Files Associated with User Groups on openEuler

- In the openEuler directory, there are two types of files related to user group information management:
- **/etc/group** : group information file
  - This file saves all information about user groups. Each line represents a user group.
  - Grouping users is a method of managing users and controlling access permissions. Each user belongs to one or more user groups and a group can contain multiple users.
- **/etc/gshadow**: encrypted group information file
  - This file saves the encryption information of a user group. For example, this file contains the user group management password (similar to the **/etc/shadow** file).
  - This file is complementary to the **/etc/group** file. For a large-scale server, there are many users and user groups, generating complex permission models. Therefore, it is important to set and manage passwords.

/etc/group File

- Each line in the **/etc/group** file consists of four fields separated by colons (:).
- The **root** user is used as an example to describe fields in a user group record in the **/etc/group** file.



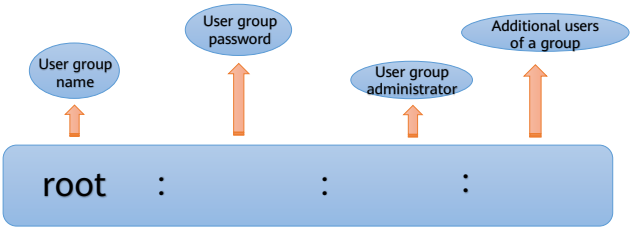
/etc/group File - related Parameters

No.	Description
1	A user group name, which has the same meaning as that in the **/etc/passwd** file.
2	A user group password. To ensure security, this field is represented by **X**. The user group password is stored in the **/etc/gshadow** file.
3	A GID, which is used to identify a user group and determine a user group type.
4	A list of users in a user group. This field lists all users in a user group.

- A user group name consists of letters or digits. Similar to a user name in **/etc/passwd**, a group name must be unique.
- A group password is the same as that in the **/etc/passwd** file. The **x** is only a password identifier, and the encrypted group password is saved in the **/etc/gshadow** file by default.
- A user group password is used to specify a group administrator. There may be a large number of accounts in a system, and the **root** user may not have time to adjust user groups. In this case, you can specify a group administrator. If a user needs to be added to or removed from a user group, the group administrator can manage this for the **root** user. However, as this function is rarely used, a group password is not often set. To grant a user permission to adjust a user group, run the **sudo** command.
- Similar to user names, the GID (not the group name) is used to distinguish user groups on Linux. Here, the GID corresponds to the fourth field in the **/etc/passwd** file. In fact, the group name corresponding to the GID in the **/etc/passwd** file is obtained through this file.
- A list of users in a group. This field lists all users in each group. Note that if the group is the user's initial group, the user is not written into this field. In this case, users displayed in this field are additional users of the user group.

/etc/gshadow File

- Each line in the **/etc/gshadow** file consists of four fields separated by colons (:).
- The **root** user is used as an example to describe fields in the **/etc/gshadow** file.



/etc/gshadow File - related Parameters

No.	Description
1	A user group name, which has the same meaning as the user name in the **/etc/group** file.
2	A user group password. Generally, a user group password is not set, and this field can be left empty or set to an exclamation mark (!).
3	A user group administrator. This field can be left empty. If there are multiple user administrators, separate them with commas (,).
4	Additional user group users. This field lists additional users in a user group.

- A group name, which is the same as that in the **/etc/group** file.
- A group password. As the group password is not set for most users, this field is usually left empty. Sometimes, it is displayed as **!**, which indicates that the group does not have a group password or a group administrator.
- A group administrator. From the perspective of a system administrator, the most important function of this file is to create a group administrator. In cases where many accounts exist on a Linux system, the super administrator **root** may be too busy to quickly respond when a user wishes to join a group. If a group administrator exists, the administrator can manage and add users to group. In this way, the **root** user does not need to perform any operations. As tools such as sudo are available for this purpose, this function is rarely used.
- Additional users in a user group. The value of this field is the same as additional users in the **/etc/group** file.

Contents

Permission Overview

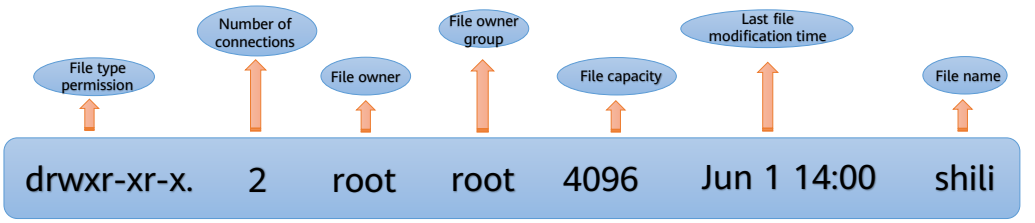
- Permissions are used by an operating system to control resource access, and are classified into read, write, and execute permissions.
- Different users have different levels of permissions on a Linux system. To ensure system security, a Linux system defines rules for each user permission.
- Each file or directory has a specific access permission, owner user, and owner group. These rules can be used to specify which user and which group can perform what operations on a specific file.

- The following three access modes can be used to restrict user access permissions:
  - Allow access by the current user only
  - Allow access by users in a specified user group
  - Allow access by any user in a system

Permission Example

- You can run the **ls –l** command to display detailed information about file permissions.
  - The file information of the **root** user is used here as an example.

drwxr-xr-x. 2 root root 4096 Jun 1 14:00 shili



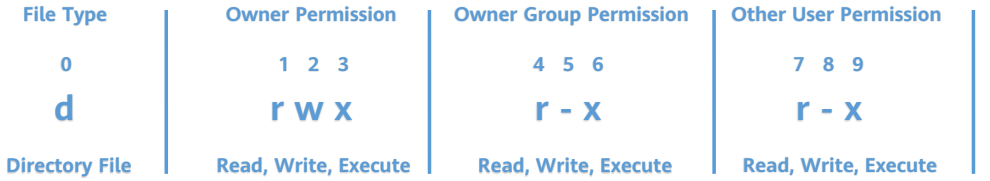
Permission Example - File Types

- The first character in each line indicates a file type. There are seven file types on Linux:

File Type	Description
-	Files excluding the other six file types
d	Directory
b	Block device file, which is a random access device
c	Character device file, which is a one-time read device such as keyboard and mouse
l	Symbolic link file, which points to another file
p	Named pipe file
s	Socket file

Permission Bits

- Linux file or directory access permission is controlled by nine permission bits. Every three bits form a group, which is a combination of read (r), write (w), and execute (x) permissions. The order of the three permission bits remain unchanged in a file or directory. A hyphen (-) indicates that a user does not have the permission.



Bit 0 indicates a file type. Bits 1 to 3 indicate the permissions of a file owner. Bits 4 to 6 indicate the permissions of other users in the same group of the file owner. Bits 7 to 9 indicate the permissions of other users on the file.

- **r**: the read permission, corresponding to digit 4.
- **w**: the write permission, corresponding to digit 2.
- **x**: the execute permission, corresponding to digit 1.
- **-**: specifies no permissions, corresponding to digit 0.
- An attribute of each file consists of 0 to 9 from left to right.

Permission Example - Access

- The following is displayed in the file information:
  - **-r** allows users to read files or all contents in a directory.
  - **-w** allows users to write files, or to create or delete files in a directory.
  - **-x** allows users to execute files or go to a directory.
  - **-** specifies no permissions and is displayed in the **r**, **w**, and **x** positions.
- The following uses the **usertxt** file as an example: **drwxr-xr-x. 2 root root 4096 Jun 1 14:00 usertxt**

Position	Permission	Binary	Decimal	Permission Details
Bits 1, 2, and 3	**rwx**	111	4+2+1=7	The file owner has the read, write, and execute permissions.
Bits 4, 5, and 6	**r-x**	101	4+1=5	Users in the same group have the read and execute permissions, but do not have the write permission.
Bits 7, 8, and 9	**r-x**	101	4+1=5	Other users have the read and execute permissions, but do not have the write permission.

- Read
  - Permission to read the actual content of a file.
  - Permission to read the directory structure list.
- Write
  - Permission to edit, add, or modify file content.
  - Permission to modify, delete, or move files in a directory.
- Execute
  - Permission to execute files.
  - Permission to access a directory.

Contents

Main Permission Setting Commands

- **chmod**: modifies a file permission.
  - The visitors to a Linux file is divided into three categories: file owner, group, and others. The **chmod** command specifies who can visit a file.
  - Usage permission: file owner
- **chown**: modifies a file owner and owner group.
  - Linux is a multi-user and multi-task system, where all files have owners. You can run the **chown** command to change an owner of a specific file to a specified user or group.
  - Usage permission: administrator (**root**)
- **chgrp**: modifies a file owner group.
  - You can run the **chgrp** command to change a group to which a file or directory belongs.
  - Usage permission: administrator (**root**)
- **umask**: sets the mask code.
  - You can run the **umask** command to preset the permission mask when creating a file
  - Usage permissions: administrator and common user

Modifying File Permissions - chmod

- The **chmod** command specifies who (file owner, group, and others) can visit a file.
- Syntax: **chmod** [*OPTION*]... **MODE**[,*MODE*]... *FILE*...
- Main options:
  - Operation objects:
    - **u**: user, indicating an owner of a file or directory
    - **g**: user group, indicating a group to which a file or directory belongs
    - **o**: other users
    - **a**: all users
  - Operators:
    - **+**: adds permissions.
    - **-**: reduces permissions.
    - **=**: provides specified permissions.
  - Permissions:
    - **r**: read
    - **w**: write
    - **x**: execute
- Based on the configuration scenario, you can modify a group of permissions on a file at the same time or modify a single permission on the file.

Modifying File Permissions - Example

- Modify all permissions of the test file **usertxt**:
  - Run the **ls –l** command to check the permissions of the **usertxt** file before modification.

```
[root@localhost ~]# ls -l

drwx------. 2 root   root      4096 Jun  8 11:10 usertxt
```

  - Run the **chmod 644 usertxt** command to modify permissions. Run the **ls –l** command to check whether permissions have been modified.

```
[root@localhost ~]# chmod 644 usertxt

[root@localhost ~]# ls –l
drw-r--r--. 2 root   root      4096 Jun  8 11:10 usertxt
```

Modifying File Permissions - chown

- The **chown** command is used to change the owner of a specified file to a particular user or group.
- Syntax: **chown** [*OPTION*]... [*OWNER*][:[*GROUP*]] *FILE*...
- Main options:
  - **-c**: displays the modified information.
  - **-f**: ignores error information.
  - **-h**: restores a symbolic link.
  - **-v**: displays detailed processing information.
  - **-R**: processes all files in a specified directory and its subdirectories.
- Based on the configuration scenario, you can change an owner or owner group, or change both at the same time.

- chown [-cfhvR] [--help] [--version] user[:group] file...
- **-c**: displays the modified information.
- **-f**: ignores error information.
- **-h**: restores a symbolic link.
- **-v**: displays detailed processing information.
- **-R**: processes all files in a specified directory and its subdirectories.
- **-R**: processes all files in a specified directory and its subdirectories.
- **--version**: displays a version.

Modifying File Permissions - Example

- Modify both owner and owner group:
  - Run the **ls –l** command to check the owner and owner group of the **usertxt** file before modification.

```
[root@localhost ~]# ls –l

drw-r--r--. 2 root   root      4096 Jun  8 11:10 usertxt
```

  - Run the **chown user: usergroup usertxt** command to modify permissions, and run the **ls –l** command to check whether permissions have been modified.

```
[root@localhost ~]# chown user:usergroup usertxt

[root@localhost ~]# ls -l
drw-r--r--. 2 user   usergroup    4096 Jun  8 11:10 usertxt
```

Modifying File Permissions - chgrp

- The **chgrp** command is used to modify a group to which a file or directory belongs.
- Syntax: **chgrp** [*OPTION*]... *GROUP FILE*...
- Main options:
  - **-v**: displays the command execution process.
  - **-c**: similar to the **-v** option, but returns only the modified part.
  - **-f**: does not display error information.
  - **-h**: modifies only symbolic link files.
  - **-R**: recursively processes all files and subdirectories in a specified directory.
- Modify a group to which the file belongs based on the configuration scenario.

HUAWEI

- **-c** or **--changes**: similar to the **-v** option, but returns only the modified part.
- **-f**, **--quiet**, or **--silent**: does not display error information.
- **-h** or **--no-dereference**: modifies only symbolic link files.
- **-R, --recursive**: recursively processes all files and subdirectories in a specified directory.
- **-v, --verbose**: displays the command execution process.
- **--help**: provides online help.
- **--reference=**<*Reference file or directory*>: sets an owner group of a specified file or directory to be the same as the owner group of a reference file or directory.
- **--version**: shows version number.

Modifying File Permissions - Example

- Modify a file owner group:
  - Run the **ls –l** command to check the owner group of the **usertxt** file before modification.

```
[root@localhost ~]# ls –l

drw-r--r--. 2 user   usergroup    4096 Jun  8 11:10 usertxt
```

  - Run the **chgrp usergroup01 usertxt** command to modify permissions, and run the **ls –l** command to check whether permissions have been modified.

```
[root@localhost ~]# chgrp usergroup01 usertxt

[root@localhost ~]# ls –l
drw-r--r--. 2 user   usergroup01    4096 Jun  8 11:10 usertxt
```

Preset Permission Mask - umask

- The **umask** command is used to specify a preset permission mask when a file or directory is created.
- Syntax: **umask**: **umask** [**-p**] [**-S**] [*mode*]
- Main options:
  - **-p**: displays a command name.
  - **-S**: specifies a permission mask in text format.
- Main umask values and corresponding file or directory permissions are as follows:

umask Value	File Permission	Directory Permission
022	644	755
027	640	750
002	664	775
006	660	771
007	660	770

- **-c** or **--changes**: similar to the **-v** option, but returns only the modified part.
- **-f**, **--quiet**, or **--silent**: does not display error information.
- **-h** or **--no-dereference**: modifies only symbolic link files.
- **-R, --recursive**: recursively processes all files and subdirectories in a specified directory.
- **-v, --verbose**: displays the command execution process.
- **--help**: provides online help.
- **--reference=**<*Reference file or directory*>: sets an owner group of a specified file or directory to be the same as the owner group of a reference file or directory.
- **--version**: shows version number.

Preset Permission Mask - Example

- Modify a file permission mask:
  - Run the **umask** command to check a umask value before modification.

```
[root@localhost ~]# umask
0077
```

  - Run the **umask 022** command to modify a permission, and run the **umask** command to check whether a permission mask has been changed.

```
[root@localhost ~]# umask 022
[root@localhost ~]# umask
0022
```

Contents

Access Control List (ACL)

- The **chmod**, **chown**, **chgrp**, and **umask** commands for common permissions can be used to modify file permissions. Why do we use the Access Control List (ACL)?
  - Without using ACL, the visitors to the Linux system are divided into three categories: file owner, user group, and other users. However, as technology develops, traditional file permission control cannot meet the requirements of complex scenarios. For example, there are multiple employees (such as **user01** and **user02**) in a department (a user group), and different permissions are assigned to employees with different responsibilities. For example, the read and write permissions are assigned to **user01**, the read-only permission is assigned to **user02**, and no permissions are assigned to **user03**. As these employees belong to the same department, employee permissions cannot be further refined. The access control list (ACL) is developed to solve this problem. ACL provides permission settings in addition to common permissions (such as **rwx** and **ugo**), and you can set specific permissions for a single user or a user group.
- Main types:
  - Assign permissions to a file owner.
  - Assign permissions to a user group of a file.
  - Assign permissions to other users.

- ACL is currently supported by most file systems.
  - Ext3\ext4
  - Xfs\zfs
- Assign permissions to a file owner.
- Assign permissions to a file's owner group.
- Assign permissions to additional users.
- Assign permissions to additional user groups.
- Assign permissions to other users.
- Maximum access permission.

ACL - Related Commands

- On Linux, we can use ACL to manage a file and its specific user and user group permissions. Simply put, an ACL requires only three commands: **setfacl**, **getfacl**, and **chacl**.
- **setfacl**: sets a file ACL.
  - The **chmod** command assigns permissions based on a file owner, owner group, and other users. The **setfacl** command assigns permissions more precisely for each file or directory.
- **getfacl**: obtains a file ACL.
- **chacl**: changes an ACL of a file or directory.
  - The **chacl** command is similar to the **chmod** command, but is more powerful and precise. While the **chmod** command specifies who can invoke a file, if a file of a user needs to be viewed only by a specific user, the **chacl** command must be executed to meet the user's requirements.

- ACL is currently supported by most file systems.
  - Ext3\ext4
  - Xfs\zfs

Obtaining a File ACL - getfacl

- The **getfacl** command is used to obtain a file or directory ACL.
- Syntax: **getfacl** [**-aceEsRLPtpndvh**] *file*...
- Main options:
  - **-a**: displays only a file ACL.
  - **-d**: displays only the default ACL.
  - **-c**: does not display comment headers.
  - **-e**: displays all valid permissions.
  - **-E**: displays invalid permissions.
  - **-s**: skips files that contain only base entries.

HUAWEI

- Main options:
  - **-a, --access**: displays only a file ACL.
  - **-d, --default**: displays only the default ACL.
  - **-c, --omit-header**: does not display comment headers.
  - **-e, --all-effective**: displays all valid permissions.
  - **-E, --no-effective**: displays invalid permissions.
  - **-s, --skip-base**: skips files that contain only base entries.
  - **-R, --recursive**: recursively displays subdirectories.
  - **-L, --logical**: logic traversal (followed by symbolic links).
  - **-P, --physical**: physical traversal (not following symbolic links).
  - **-t, --tabular**: tab-delimited output format
  - **-n, --numeric**: displays numeric user/group IDs.
  - **-p, --absolute-names**: does not remove the slash (/) at the beginning of a path.
  - **-v, --version**: displays a version and exits.
  - **-h, --help**: shows help information.

Obtaining a File ACL - Example

- Run the **getfacl** command to view all valid permissions of the **usertxt** file.
  - Command: **getfacl –e usertxt**

```
[root@localhost ~]# getfacl -e usertxt

# file: usertxt
# owner: user
# group: usergroup01
user::rw-
group::r--
other::r--
```

Setting a File ACL - setfacl

- The **setfacl** command is used to set a file ACL.
- Syntax: **setfacl** [**-bkndRLP**] {**-m|-M|-x|-X**... } *file* ...
- Main options:
  - **-m**: modifies the ACL of a specified file. This parameter cannot be used together with **-x**.
  - **-x**: deletes subsequent parameters.
  - **-b**: deletes all ACL parameters.
  - **-k**: removes preset ACL parameters.
  - **-R**: recursively sets ACL parameters.
  - **-d**: presets the ACL parameters of a directory.

Setting a File ACL - Example

- Grant a user's read and write permissions.
  - Run the **getfacl usertxt** command to view permissions on the **usertxt** file.

```
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
group::r--
other::r--
```

  - Run the **setfacl –m u:user:rw usertxt** command to add read and write permissions to the user, and then run the **getfacl** command to check permissions. The command output shows that permissions have been added.

```
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
user:user:rw-                   #effective:rw-
group::r--                      #effective:r--
mask::rw-
other::r--
```

Changing a File or Directory ACL - chacl

- The **chacl** command is used to set control permissions on files or directories.
- Syntax: **chacl** [**acl/R/D/B/l/r**] *pathname*... **/ chacl -b acl dacl** *pathname*... **/ chacl -d dacl** *pathname*...
- Main options:
  - **-b**: modifies file permissions and default directory permissions at the same time.
  - **-d**: sets default permissions on a directory.
  - **-R**: deletes only file permissions.
  - **-D**: deletes only directory permissions.
  - **-B**: deletes all permissions.
  - **-l**: lists all file and directory permissions.
  - **-r**: sets permissions on all directories and subdirectories.

Changing a File or Directory ACL - Example

- Clear ACL settings from the **usertxt** file:
  - Command: **chacl –B usertxt**

```
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
user:user:rw-                   #effective:rw-
group::r--                      #effective:r--
mask::rw-
other::r--

[root@localhost ~]# chacl -B usertxt
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
group::r--
other::r--
```

Contents

Other Management Permissions

- In the Linux operating system, the default account is a common user. However, only the **root** user can modify system files or run certain commands.
- The following commands are commonly used to switch to the **root** user:
  - **su**: switches a user identity to the **root** user. The user is still a common user in the shell environment.
  - **su –**: switches a user to the **root** user for both the user identity and shell environment.
  - **sudo**: allows common users to execute commands that can be executed only by administrator accounts.

Commands - su and su-

- The **su** command is used to change a user identity, but does not change the shell environment.
- Syntax: **su** [*options*] [-] [<*user*> [<*argument*>...]]
- Main options:
  - **-m, -p**: specifies that environment variables are not changed when the **su** command is executed.
  - **-s**: specifies the shell to be executed (such as **bash**, **csh**, and **tcsh**).
  - **-c**: switches to the account *user* and restores to the original user after the command is executed.
  - **-f**: does not need to read the boot file. This parameter is used only for **csh** or **tcsh**.

Example

- Change the user identity to **root** and display detailed information.
  - Command: **su –c ls root**

```
[root@localhost ~]# su -c ls root

anaconda-ks.cfg  createVM.sh    deleteVM.sh                    revertsnap.sh  test    usertxt
createsnap.sh    deletesnap.sh  httpd-2.4.34-15.oe1.aarch64.rpm  shili          test01
```

Command - sudo

- The **sudo** command allows common users to execute tasks that can be executed only by the **root** user.
- Syntax: **sudo –h | -K | -k | -V**
- Main options:
  - **-h**: displays the version number and instructions.
  - **-k**: asks the user for a password upon the next run of the **sudo** command.
  - **-V**: displays the version number.
  - **-l**: displays user permissions.
  - **-L**: displays the **sudo** settings.

HUAWEI

- Option description:
  - **-V**: displays the version number.
  - **-h**: displays a version number and command instructions.
  - **-l**: displays the permissions of the user who executes the **sudo** command.
  - **-v**: confirms a password after **sudo** is executed for the first time, or if it is not executed within *N* minutes (*N* is set to **5** by default). If the execution duration exceeds *N* minutes, password confirmation is also required.
  - **-k**: forces a user to enter a password when the **sudo** command is next executed (regardless of whether the user has run the **sudo** command within *N* minutes).
  - **-b**: executes a command in the background.
  - **-p prompt**: changes the password request prompt. *%u* is replaced with the user's account name. **%h** is replaced with a host name.
  - **-u username/#uid**: specifies that if this option is not added, a command is executed as the **root** user. If this option is added, a command is executed as the **username** user. **#uid** indicates the user ID of **username**.
  - **-s**: executes a shell specified by the environment variable *SHELL,* or a shell specified in **/etc/passwd**.
  - **-H**: specifies the environment variable *HOME* to a home directory of the target user. (If the **-u** is not added, the user is the **root** system administrator.)
- Commands are executed as the system administrator or another user specified by **-u**.

Example

- Run the **sudo** command to view the version number.
  - Command: **sudo -V**

```
[root@localhost ~]# sudo –V

Sudo version 1.8.27
Configure options: --build=aarch64-openEuler-linux-gnu --host=aarch64-openEuler-linux-gnu --program-prefix= --disable-dependency-tracking --prefix=/usr --exec-prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --sysconfdir=/etc --datadir=/usr/share --includedir=/usr/include --libdir=/usr/lib64 --libexecdir=/usr/libexec --localstatedir=/var --sharedstatedir=/var/lib --mandir=/usr/share/man --infodir=/usr/share/info --prefix=/usr --sbindir=/usr/sbin --libdir=/usr/lib64 --docdir=/usr/share/doc/sudo --disable-root-mailer --with-logging=syslog --with-logfac=authpriv --with-pam --with-pam-login --with-editor=/bin/vi --with-env-editor --with-ignore-dot --with-tty-tickets --with-ldap --with-selinux --with-passprompt=[sudo] password for %p:  --with-linux-audit --with-sssd
Sudoers policy plugin version 1.8.27
Sudoers file grammar version 46
```

Quiz

1. (Singel-answer)On openEuler, which of the following UIDs belongs to a common user by default?
   A. 0
   B. 200
   C. 800
   D. 1200
2. (Single-answer)Which of the following commands can be used to view information in files associated with users and groups?
   A. cat
   B. chmod
   C. clear
   D. chage

- Answers:
  1. D
  2. A

Summary

- This document describes basic concepts related to users and user groups on openEuler, and provides commands and methods for adding users and user groups. It describes common file permissions, such as read, write, and execute, and special file permissions, such as **setfacl**, **getfacl**, and **chacl**, as well as the access control list (ACL). It shows how to modify file permissions by using related command parameters and examples, such as how to modify read, write, and execute permissions on files and directories.

More Information

openEuler open source community:

https://openeuler.org/en/docs/20.03_LTS/docs/Releasenotes/release_notes.html

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

Software Installation and Service Management

HUAWEI

Foreword

- This chapter describes the concepts and operation commands of three software package installation methods on openEuler: RPM, source code, and YUM, as well as the concept and operation manner of the systemd management service.

Objectives

- On completion of this course, you will understand the concepts and operation commands of:
  - RPM
  - Source code
  - YUM
  - systemd

Contents

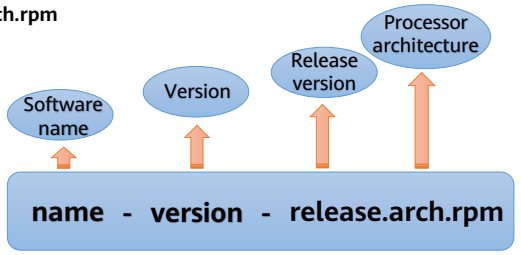
Overview: Linux Software Package Management

- Linux software packages are classified into source code and binary packages, which have different formats, for example, RPM and TGZ. RPM and compilation are two primary software installation methods.
- RPM is the primary tool for installing software and is easy to use.
- Different distributions of OSs such as openLinux, SUSE, and CentOS, also use RPM to manage software packages.
- The software packaging format and tool varies according to the related platform. Debian and Ubuntu use the DEB package and apt-get source installation methods to manage software packages. FreeBSD uses the ports and TXZ packaging formats and the make and pkg tools.

- DEB is the file name extension for Debian software packages and is a standard archive of Unixar. It contains the file information and content of a package, which are compressed using gzip and tar. dpkg is a typical program that handles these packages, which often runs through apt-get of Debian. The DEB format is a dedicated installation package format for Debian systems, which includes Debian and Ubuntu. It works with the APT software management system and has become a popular installation package on Linux.

RPM Software Package Management

- RPM is a packaging and automatic installation tool that can download packages from the Internet. It generates .rpm files to install, uninstall, and maintain applications.
- Name format:
  - **name-version-release.arch.rpm**



- Common RPM commands:
  - Installing a package: **rpm -ivh**
  - Upgrading a package: **rpm -Uvh**
  - Removing a package: **rpm -e**
- Installation options:
  - **--force** forcibly installs a package even if the package overwrites files in another package.
  - **--nodeps** forcibly installs an RPM package even if its installation depends on other packages that are not installed.
- Querying whether a package is installed: **rpm -q < rpm package name>**
- Obtaining information about an installed package: **rpm -qi < rpm package name>**
- Listing files in a package: # **rpm -ql < rpm package name>**
- Listing the RPM package that housing a server file: **rpm -qf**
- Combining multiple options: **rpm -qil < rpm package name>**
- Listing all installed RPM packages: **rpm -qa**
- Listing files contained in an RPM package that is not installed in the system: **rpm -qilp < rpm package name>**

RPM: Advantages and Disadvantages

- Advantages:
  - Simple, convenient, and compatible
  - Parameter information is recorded in a database for easy software query, upgrade, or uninstallation.
- Disadvantages:
  - The installation environment must be the same as the packaging environment.
  - Dependencies must be processed before you uninstall the software to prevent other software being unusable.

Contents

Common RPM Options

- RPM is usually used for installation, deletion, upgrade, refresh, and query.
- Syntax: **rpm** ***[OPTION...]***
- Main options:
  - **-i**: specifies the software package to be installed.
  - **-h**: uses a number sign (#) to display the process and progress of RPM installation.
  - **-v**: displays details of the installation process.
  - **-U**: upgrades a specified software package.
  - **-q**: queries whether a specified software package has been installed in the system or queries the content of a specified RPM package.
  - **-a**: views all software packages that have been installed in the system.
  - **-V**: queries the version of an installed software package.
  - **-c**: displays all configuration files.
  - **-p**: queries or validates files in a software package.

- Other RPM options:
  - **-vh**: displays the installation progress.
  - **-qpl**: lists the files in an RPM software package.
  - **-qpi**: displays the descriptions of an RPM software package.
  - **-qf**: searches for the RPM software package to which a specified file belongs.
  - **-Va**: validates all RPM software packages to search for lost files.
  - **-qa**: searches for a file, for example, **rpm -qa mysql**.

RPM Commands - Install

- Syntax:

```
rpm –i example.rpm

rpm –iv example.rpm

rpm –ivh example.rpm
```

- Main options:
  - **-i**: installs a package.
  - **iv**: installs a package and displays information about the file being installed.
  - **ivh**: installs a package, and displays information about the file being installed (including the progress).

- **-i**: specifies the software package to be installed.
- **-h**: uses a number sign (#) to display the process and progress of RPM installation.
- **-v**: displays details of the installation process.
- **-U**: upgrades a specified software package.
- **-q**: queries whether a specified software package has been installed in the system or queries the content of a specified RPM package.
- **-a**: views all software packages that have been installed in the system.
- **-V**: queries the version of an installed software package.
- **-c**: displays all configuration files.
- **-p**: queries or validates files in a software package.

RPM Commands - Uninstall

- Syntax:

```
rpm –e example.rpm
rpm –e -nodeps example.rpm
rpm –e -allmatches example.rpm
```

- Main options:
  - Dependency between packages needs to be considered when you uninstall an RPM software package.
  - If the package dependency is not considered during uninstallation, run the **nodeps** command to forcibly uninstall a package without checking its dependency.
  - If a software package has multiple versions, run the **allmatches** command to uninstall them in batches.

- If you run the **nodeps** command to forcibly delete a software package, other software may become unavailable. Therefore, you are not advised to use this command.

RPM Commands - Upgrade

- Syntax:

```
rpm -U example.rpm

rpm -Uvh example.rpm

rpm -F example.rpm

rpm -Fvh example.rpm
```

- Main options:
  - **-Uvh**: replaces an existing package with a new package.
  - **-Fvh**: upgrades an existing package.

RPM Commands - Query

- Syntax:

```
rpm -q example.rpm

rpm -qa
```

- Main options:
  - **-q**: queries whether a software package is installed.
  - **-qa**: queries all installed software packages.
  - **-qf**: queries all software packages that have once been installed.
  - **-qp**: queries uninstalled software packages.
  - **-ql**: queries the file list and full path in an installed software package.
  - **-qi**: queries details about a software package.
  - **-qc**: queries the configuration file in an installed software package.
  - **-qd**: queries the help document in an installed software package.

RPM Commands - Main Options

- **-qa**: queries all installed software packages.

```
[root@localhost ~]# rpm -qa
openvswitch-2.12.0-5.oe1.aarch64
tk-8.6.8-4.oe1.aarch64
scap-security-guide-0.1.39-4.oe1.noarch
libtar-1.2.20-17.oe1.aarch64
libwbclient-4.11.6-5.oe1.aarch64
```

- **-ql**: queries the file list and full path in an installed software package.

```
[root@localhost ~]# rpm -ql python3-libxml2-2.9.8-
9.oe1.aarch64
/usr/lib64/python3.7/site-
packages/__pycache__/drv_libxml2.cpython-37.opt-1.pyc
/usr/lib64/python3.7/site-
packages/__pycache__/drv_libxml2.cpython-37.pyc
/usr/lib64/python3.7/site-
packages/__pycache__/libxml2.cpython-37.opt-1.pyc
```

- **-qi**: queries details about a software package.

```
[root@localhost ~]# rpm -qi python3-libxml2-2.9.8-
9.oe1.aarch64
Name        : python3-libxml2
Version     : 2.9.8
Release     : 9.oe1
Architecture: aarch64
Install Date: Wed 22 Apr 2020 04:42:38 PM CST
Group       : Development/Libraries
Size        : 1411076
License     : MIT
```

Contents

Generation of the DNF Tool

- YUM is a Linux software management tool that automatically downloads RPM packages from a specified server and installs them. It serves as a software repository that manages software packages, and resolves the dependency problem between software packages, achieving a higher efficiency. Further, it needs the DNF tool.
  - DNF can be used because the YUM performance is poor, the memory usage is too high, and the dependency parsing speed is low. In addition, YUM is very reliant on YUM repo files. If such a file is faulty, YUM-related operations may fail. In this case, the DNF tool is used to overcome the YUM bottlenecks and improve memory usage, dependency analysis efficiency, and running speed for better use experience.

Overview: DNF Tool

- Dandified YUM (DNF) is a Linux software package management tool that manages RPM software packages.
- DNF can query software package information, obtain required software packages from a specified software repository, and install, uninstall, and update the software packages by automatically processing dependencies.
- DNF is fully compatible with YUM and provides YUM-compatible command lines and APIs for extensions and plugins.
- Only user **root** can use DNF.

Contents

Managing Software Packages with DNF

Configuring DNF	Managing Software Packages or Package Groups	Update
Modify DNF configuration parameters.	Search for software packages.	Check for updates.
Create a local software repository.	List required software packages or package groups.	Upgrade
Add, enable, or disable a software source.	Display RPM and software package information.	Update all packages and their dependencies.
	Install RPM packages and download software packages. Install package groups.	
	Delete software packages or package groups.	

DNF - Software Source Service

- Software sources are free repositories used for installing Linux applications.
- A software source can be a network server, a CD-ROM, or even a hard drive directory.
- Advantages of Linux software sources:
  - Certain software can be automatically downloaded and installed by using a tool.
  - A software source allows you to receive critical security updates and handle security risks.
  - The complex software dependency problem can be eliminated, boosting the installation speed.

DNF Configuration File - **/etc/dnf/dnf.conf**

- The main DNF configuration file is **/etc/dnf/dnf.conf**. The **main** part in this file stores the global DNF configuration.
- Run the **cat /etc/dnf/dnf.conf** command to view parameters of **main**.
- Parameters:
  - **cachedir**: cache directory, which is used to store RPM packages and database files.
  - **best**: always tries to install the latest version during package upgrade. If the latest version cannot be installed, the system displays a cause of the failure and stops installation. The default value is **True**.
  - **installonly_limit**: specifies the number of packages that can be synchronously installed and are listed in the **installonlypkgs** command. The default value is **3**. You are advised not to decrease the value.
  - **clean_requirements_on_remove**: deletes dependencies that are no longer used during **dnf remove**. If a software package is installed using DNF instead of an explicit user request, the software package can only be deleted through **clean_requirements_on_remove**. That is, the software package is introduced as a dependency. The default value is **True**.

Configuring DNF - Modifying Configuration Parameters

- You can customize software repositories, but each repository name must be unique to avoid conflicts. You can add one or more repositories to change the location of the software source to be installed.
- Run the **vim /etc/dnf/dnf.conf** command to add one or more repositories to a file.
- Parameter description:
  - **name=repository_name**: specifies a string that describes a software repository.
  - **baseurl=repository_url**: specifies a software repository URL, such as the location where the HTTP protocol is used (http://path/to/repo), the location where the FTP protocol is used (ftp://path/to/repo), or the local location (file:///path/to/local/repo).

Creating Local Software Repositories

- To create a software repository, perform the following steps:
  - Run the following command to install **createrepo** as user **root**:

    ```
    dnf install createrepo
    ```

  - Place the required software package in a directory, for example, **/mnt/local_repo/**.
  - Run the following command to create a software repository:

    ```
    createrepo --database /mnt/local_repo
    ```

Adding Software Repositories

- To customize a software repository, add one or more repositories to the **/etc/dnf/dnf.conf** file or add the .repo file to the **/etc/yum.repos.d/** directory.
- Only user **root** can add the .repo file.
  - After the following command is successfully executed, the corresponding repository file is generated in the **/etc/yum.repos.d/** directory.

```
dnf config-manager --add-repo repository_url
```

Enabling and Disabling Software Repositories

- After a software repository is added, run the following command to enable it as user **root** (the value of **repository** is the repository ID of the new .repo file):

```
dnf repolist # View the repository ID of the new .repo file.
dnf config-manager --set-enable repository
```

- If a software repository is no longer used, run the following command to disable it as user **root**:

```
dnf repolist # View the repository ID of the new .repo file.
dnf config-manager --set-disable repository
```

Managing Software Packages

- DNF can be used to install, query, and delete software packages.
  - Run the following command to search for the required RPM software package by its name, abbreviation, or description:

```
dnf search term
```

```
[root@localhost ~]# dnf search term
created by dnf config-manager from https://repo.openeuler.org/openEuler-20.03-LTS/OS/aar
Last metadata expiration check: 0:00:06 ago on Tue 02 Jun
================================================== Name & Summary Matched: term
xterm-help.aarch64 : Doc files for xterm
perl-Term-Cap.noarch : Perl termcap interface
gnome-terminal.aarch64 : A terminal emulator for GNOME
texlive-termcal-doc.noarch : Documentation for termcal
perl-Term-Cap-help.noarch : Documents for perl-Term-Cap
texlive-acroterm-doc.noarch : Documentation for acroterm
texlive-termlist-doc.noarch : Documentation for termlist
texlive-termmenu-doc.noarch : Documentation for termmenu
texlive-termcal-de.noarch : German localization for termcal
texlive-xwatermark-doc.noarch : Documentation for xwatermark
perl-TermReadKey-help.noarch : Documents for perl-TermReadKey
```

Listing Software Packages

- Run the following DNF commands to list all installed and available RPM software packages:

```
dnf list all

dnf list glob_expression... # Display information about a specified RPM package.
```

- Example:

Displaying RPM Package Information

- Run the following DNF command to view information about an RPM package:

```
dnf info package_name...
```

- Example:

```
[root@localhost ~]# dnf info httpd
Last metadata expiration check: 0:55:43 ago on Tue 02 Jun
Installed Packages
Name         : httpd
Version      : 2.4.34
Release      : 15.oe1
Architecture : aarch64
Size         : 8.8 M
Source       : httpd-2.4.34-15.oe1.src.rpm
Repository   : @System
From repo    : anaconda
Summary      : Apache HTTP Server
URL          : https://httpd.apache.org/
License      : ASL 2.0
Description  : Apache HTTP Server is a powerful and flexible HTTP/1.1 compliant web
             : server.
```

Downloading, Installing, and Deleting RPM Packages

- Run the following DNF commands to download, install, and delete RPM packages.

```
dnf download package_name
dnf install package_name
dnf remove package_name...
```

- Example:

```
[root@localhost ~]# dnf download httpd
Last metadata expiration check: 0:10:52 ago on Tue 02 Jun
httpd-2.4.34-15.oe1.aarch64.rpm                    409 kB/s | 1.2 MB     00:03
```

```
[root@localhost ~]# dnf install httpd
Last metadata expiration check: 0:08:27 ago on Tue 02 Jun
Package httpd-2.4.34-15.oe1.aarch64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

```
[root@localhost ~]# dnf remove totem
No match for argument: totem
No packages marked for removal.
Dependencies resolved.
Nothing to do.
Complete!
```


- You can also install multiple packages simultaneously by appending their names as arguments. Add **strict=False** to the **/etc/dnf/dnf.conf** configuration file and run the DNF command to add **--setopt=strict=0** to the file. Run the following command as user **root**: **dnf install package_name package_name... --setopt=strict=0**
- If you need to download a dependency that is not installed, add **--resolve**. Then run the **dnf download --resolve package_name** command.

Managing Software Package Groups

- A package group is a collection of associated software packages, such as system tools.
  - Run the following DNF commands to view all installed and available software package groups:

```
dnf groups summary

dnf group list# List the software package group and the corresponding group ID.
```

```
[root@localhost ~]# dnf groups summary
Last metadata expiration check: 0:20:45 ago on Tue 02 Jun
Installed Groups: 9
```

```
[root@localhost ~]# dnf group list
Last metadata expiration check: 0:21:08 ago on Tue 02 Jun
Available Environment Groups:
   Minimal Install
   Server
Installed Environment Groups:
   Virtualization Host
Installed Groups:
   Container Management
   Development Tools
   Headless Management
   Legacy UNIX Compatibility
   Network Servers
   Scientific Support
   Security Tools
   System Tools
   Smart Card Support
```

Displaying Package Group Information

- Run the following DNF command to list the mandatory and optional software packages in a package group:

```
dnf group info glob_expression...
```

- Example:

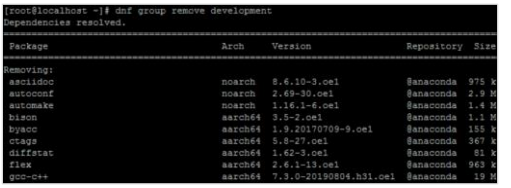
```
[root@localhost ~]# dnf group info "Development Tools"
Last metadata expiration check: 0:26:26 ago on Tue 02 Jun

Group: Development Tools
 Description: A basic development environment.
 Mandatory Packages:
   autoconf
   automake
   binutils
   bison
   flex
   gcc
   gcc-c++
   gdb
   gettext
   glibc-devel
   huaweijdk-8
   libtool
   make
```

Installing and Deleting Software Package Groups

- Each software package group has a name and an ID. You can install or delete a software package group by searching its name or ID.

```
dnf group install group_name/groupid

dnf group remove group_name/groupid
```

- Example:



HUAWEI

- You can also install multiple packages simultaneously by appending their names as arguments. Add **strict=False** to the **/etc/dnf/dnf.conf** configuration file and run the DNF command to add **--setopt=strict=0** to the file. Run the following command as user **root**: **dnf install package_name package_name... --setopt=strict=0**
- If you need to download a dependency that is not installed, add **--resolve**. Then run the **dnf download --resolve package_name** command.

Checking and Updating Software Packages

- Use DNF commands to check whether software packages or package groups in a system need to be updated, and update them as required.
  - Run the following commands to search for the required RPM software package by its name, abbreviation, or description:

```
dnf check-update
dnf update package_name / dnf group update group_name
```

- Example:

```
[root@localhost ~]# dnf check-update
Last metadata expiration check: 0:40:55 ago on Tue 02 Jun
Obsoleting Packages
linux-firmware.noarch                          20190815-4.oe1                  @anaconda
    linux-firmware.noarch                      20190815-4.oe1                  @anaconda
linux-firmware.noarch                          20190815-4.oe1                  repository
    linux-firmware.noarch                      20190815-4.oe1                  @anaconda
linux-firmware.noarch                          20190815-4.oe1                  openEuler
    linux-firmware.noarch                      20190815-4.oe1                  @anaconda
linux-firmware.noarch                          20190815-4.oe1                  repo.openeuler.org_openEuler-20.03-LT
    linux-firmware.noarch                      20190815-4.oe1                  @anaconda
```

Contents

Overview: Source Code-based Software Installation

- Source code offers an alternative way to install software. On Linux, most software is released as source packages, which are more portable than binary software packages. Only one source package needs to be released for each software and can be executed by different users after compilation. However, the configuration and compilation processes are complex.
- RPM is preferred for software installation on openEuler, but source code may also be required in the following scenarios:
  - The RPM software package version is outdated, and compilation parameters do not apply to the current service.
  - No RPM software package is available for the software to be installed.
  - RPM software packages lack certain features.
  - Compilation parameters are optimized to boost performance.

Source Code-based Software Installation: Advantages and Disadvantages

- Advantages:
  - During a compilation process, you can flexibly set parameters as required to install the software.
  - After local compilation, the software installed using source code is fully compatible with a local host.
- Disadvantages:
  - The configuration and compilation processes are complex.
  - The dependency package may not exist following a new software installation (or other problems). Consequently, software upgrade is complex and risky.

Contents

Procedure for Installing Software Through Source Code

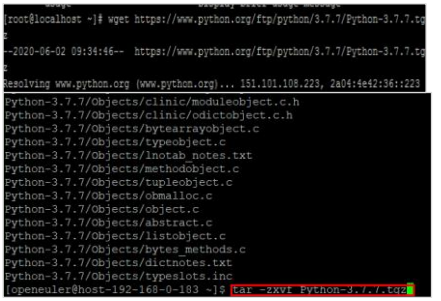
- Procedure:
  - Download and decompress a source package and verify its integrity.
  - View the **README** and **INSTALL** files, which record software installation methods and precautions.
  - Create a makefile by running the **./configure** script.
  - Run the **make** command to automatically compile the source code into a binary file.
  - Run the **make install** command to install the binary file compiled in the previous step to the corresponding directory. The default installation path is **/usr/local/**, and the corresponding configuration file is located in **/usr/local/etc** or **/usr/local/***/etc**.

- **./configure** --This command can be followed by various parameters which indicate functions to be added.
- When running the **make** command, you can add a parameter to form **make clean** to clear target files generated in the previous compilation. This ensures that the current compilation is not affected by files generated in the previous compilation.
- You can run the **./configure** command to change the default installation path for the final step.

Downloading and Decompressing a Source Package - Example

- The following uses the Python software as an example to describe the source code–based installation:
  - Download and decompress a source package and verify its integrity.

```
wget https://www.python.org/ftp/python/3.7.7/Python-3.7.7.tgz
tar -zxvf Python-3.7.7.tgz
```



Viewing the **README** File - Example

- The following uses the Python software as an example to describe the source code-based installation:
  - Go to the source code directory and view the **README** file.

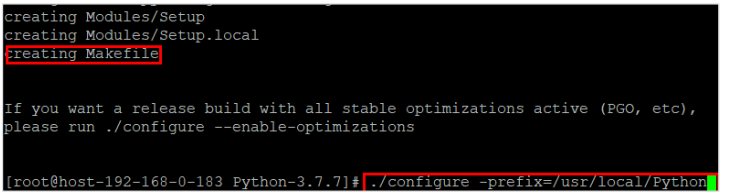
```
cat Python-3.7.7/README.rst
```

```
[openeuler@host-192-168-0-183 Python-3.7.7]$ cat README.rst
This is Python version 3.7.7
============================

.. image:: https://travis-ci.org/python/cpython.svg?branch=3.7
   :alt: CPython build status on Travis CI
   :target: https://travis-ci.org/python/cpython/branches

.. image:: https://dev.azure.com/python/cpython/_apis/build/status/Azure%20Pipelines%20CI?branch
Name=3.7
   :alt: CPython build status on Azure Pipelines
   :target: https://dev.azure.com/python/cpython/_build/latest?definitionId=4&branchName=3.7

.. image:: https://codecov.io/gh/python/cpython/branch/3.7/graph/badge.svg
   :alt: CPython code coverage on Codecov
   :target: https://codecov.io/gh/python/cpython/branch/3.7

Copyright (c) 2001-2020 Python Software Foundation.  All rights reserved.

See the end of this file for further copyright and license information.
```

Generating a Makefile - Example

- The following uses the Python software as an example to describe the source code-based installation:
  - Run the **./configure** command to generate a makefile.

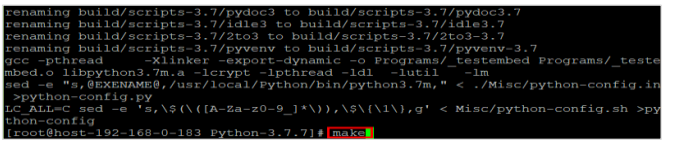
```
./configure --prefix=/usr/local/Python
```



Software Installation Through Source Code - Example

- The following uses the Python software as an example to describe the source code-based installation:
  - Run the **make** command to start compilation.

    ```
    make/make clean
    ```

  - Run the **make install** command to install the software.

    ```
    make install
    ```

  (Note: Related environment components may be missing during installation. You can use the YUM tool to download and install these components.)

Contents

Introduction to systemd

- On Linux, systemd is a system and service manager compatible with SysV and LSB initialization scripts. Enabling systemd provides on-demand activation policies based on daemon processes.
- The systemd service supports snapshot and system status restoration, and maintains mount and self-mount points. In this way, finer logical control can be implemented between services based on the dependency relationship, providing higher parallel performance.

systemd Unit Concepts

- The activation and monitoring system of systemd is based on units. A unit consists of a name and a type corresponding to a configuration file. The following are main unit types:
  - Service unit: system service
  - Target unit: a group of systemd units
  - Automount unit: file system mount point
  - Device unit: device file identified by a kernel
  - Mount unit: file system mount point
  - Path unit: a file or directory in a file system
  - Scope unit: externally created process
  - Snapshot unit: saving status of the systemd manager

HUAWEI

- Other types:
  - Slice unit: a group of hierarchically organized units that manage system processes
  - Socket unit: a socket for inter-process communication
  - Swap unit: a swap device or swap file
  - Timer unit: systemd timer

systemd Features

- systemd has the following features:
  - Fast activation
  - On-demand activation
  - Service lifecycle management by cgroups
  - Mount and automount point management
  - Transactional dependency management
  - Compatibility with SysVinit scripts
  - System status snapshots and system restoration

Contents

Managing System Services

- A systemd user can run, stop, restart, display, and enable or disable system services by running the **systemctl** command.
- The **systemctl** command functions similar to the **sysvinit** command. However, you are advised to use the **systemctl** command to manage system services.
- Differences between **systemctl** and **sysvinit** are as follows:

sysvinit	systemd	Remarks
service NetworkManager start	systemctl start NetworkManager	Starts a service (without restarting an existing one).
service NetworkManager stop	systemctl stop NetworkManager	Stops a service (without restarting an existing one).
service NetworkManager reload	systemctl reload NetworkManager	Reloads a configuration file without interrupting the wait operation.

systemctl - Displaying Services

- To view a running service, run the following command:

```
systemctl list-units --type service # To view all services, add –all to the end of the command line.
```

systemctl - Displaying Service Statuses

- To view the status of a service, run the following command:

```
systemctl status name.service
```

- Parameter description:
  - **Loaded**: determines whether a service is loaded and whether the corresponding absolute path is enabled.
  - **Active**: determines whether a service is running and displays a time stamp.
  - **Main PID**: specifies a PID of the corresponding system service.
  - **CGroup**: specifies other information about the cgroup.

systemctl - Related Operations

- Run the following systemctl commands to run, stop, restart, display, enable, or disable system services:
  - Run a service.

```
systemctl start name.service
```

  - Stop a service.

```
systemctl stop name.service
```

  - Restart a service.

```
systemctl restart name.service
```

  - Enable a service.

```
systemctl enable name.service
```

  - Disable a service.

```
systemctl disable name.service
```

systemctl - Other Operations

- Run the following systemctl commands to shut down, restart, or hibernate a system:
  - Shut down a system.

```
systemctl poweroff # Shut down a system and power it off.
systemctl halt # Shut down a system but do not power it off.
```

  - Restart a system.

```
systemctl reboot
```

  - Suspend a system.

```
systemctl suspend
```

  - Hibernate a system.

```
systemctl hibernate

systemctl hybrid-sleep # Suspend and hibernate a system.
```


HUAWEI

- When you perform these operations, a notification is sent to all login users. If you do not want the systemd to notify users, add the **--no-wall** option.

Quiz

1. (Single-answer)Which of the following commands is not a Linux file operating command?

   A. cat

   B. chmod

   C. clear

   D. more

- Answer:

  1. C

Summary

- This chapter describes the concepts and operations of three installation methods of openEuler (RPM, source code, and YUM), in addition to the concept and operations of the systemd management service.

More Information

openEuler open-source community:

https://openeuler.org/en/docs/20.03_LTS/docs/Releasenotes/release_notes.html

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

File System and Storage Management

HUAWEI

Foreword

- This course describes the basic concepts of file systems, drive storage, and logical volume storage, how to manage and use the file system and storage, and some common operation commands.

Objectives

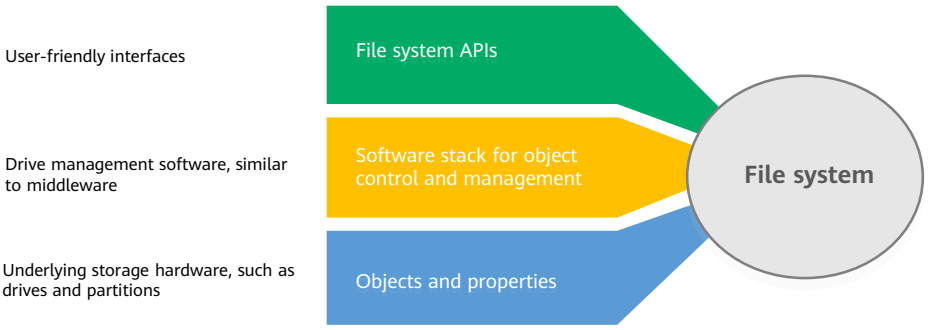
Upon completion of this course, you will understand:

- Basic concepts of file systems and storage
- How to mount and use drive storage
- Methods of managing logical volumes

Contents

File System Overview

- A file system is a method and a data structure used by an operating system (OS) to identify files on a storage device or a partition, that is, a method of organizing files on a storage device. In an OS, a software structure that manages and stores file data is referred to as a file management system, or file system for short.



File System Types and Application Scenarios

Common File System	Application Scenario
FAT	FAT, including the FAT16 and FAT32 variants, is used by Windows 9X OSs.
NTFS	NTFS is a security-based file system. It is a unique file system structure used by Windows NT. Windows 2000 adopts the updated NTFS 5.0.
NFS	Network file system (NFS) is used for file sharing between UNIX systems over the network.
RAW	RAW file system indicates that the drive is not processed or formatted.
ext	As the standard file system on GNU/Linux, extended file system (ext) features excellent file access performance and is more effective against small- and medium-sized files. ext variants include ext2, ext3, and ext4.
XFS	A high-performance log file system developed for the IRIX OS by Silicon Graphics in 1993. Later ported to the Linux kernel, it excels in large-file processing and provides smooth data transfer.

Contents

File Systems on openEuler

- The openEuler kernel is derived from Linux. The Linux kernel supports more than 10 types of file systems, such as Btrfs, JFS, ReiserFS, ext, ext2, ext3, ext4, ISO 9660, XFS, Minix, MSDOS, UMSDOS, VFAT, NTFS, HPFS, NFS, SMB, SysV and PROC. The following table describes the common file systems.
- The default file system on openEuler is ext4.

Common File System	Description
ext	File system specially designed for Linux. The latest version is ext4.
XFS	A high-performance log file system developed for the IRIX OS by Silicon Graphics in 1993. Later ported to the Linux kernel, it excels in large-file processing and provides smooth data transfer.
VFAT	On Linux, VFAT is the name of the FAT (including FAT16 and FAT32) file systems of DOS and Windows.
NFS	Network file system (NFS) is used for file sharing between UNIX systems over the network.
ISO 9600	The standard file system for optical disc media. Linux supports this file system, allowing the system to read optical discs and ISO image files, and burn optical discs.

Contents

Swap Space

- The swap space on Linux is a partition or a file on the drive. When the physical memory is insufficient, the resources that are not frequently accessed in the memory are saved to the preset swap space so that the memory occupied by the resources is released. In this way, the system has more memory to serve each process. When the content stored in the swap space is needed, the system loads the data in the swap space to the memory.
- The sum of the physical memory and swap space size is the total virtual memory provided by the system.
- Why is the swap space required?
  - More system memory: When the physical memory is insufficient, increasing the swap space size is more cost-effective than adding physical memory.
  - Overall system performance improvement: By moving infrequently used data to the swap space, the system has more memory for caching, accelerating the system I/O.
  - Enabling Linux hibernation: In many Linux distributions (such as Ubuntu), when the system hibernates, memory data is saved to the swap space. The saved data will be loaded to the memory upon next startup.

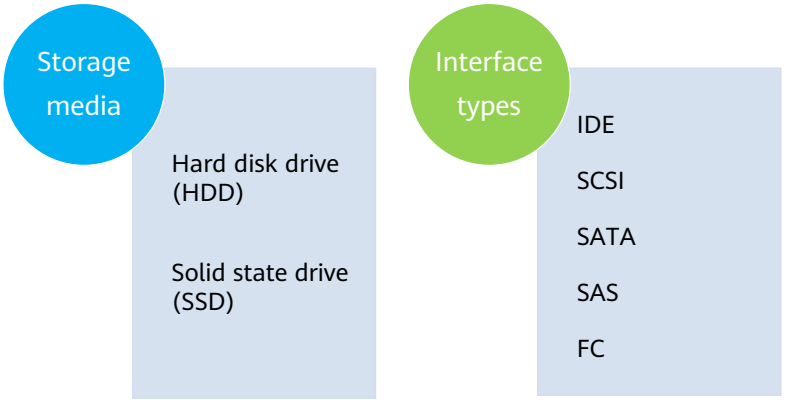
Swap Space Configuration

- Linux has two forms of swap space: the swap partition and swap file. The swap partition is an independent drive that has no other file or content. The swap file is a special file in the file system and is independent from the system and data files.
  - Creating a swap partition: Run **fdisk** to create a partition, **mkswap** to create a swap partition, and **swapon** to enable the swap partition.
  - Creating a swap file: Create a file, run **mkswap** to format the file, and then run **swapon** to enable the swap file.
- Recommended swap sizes

RAM Size	Recommended Swap Size
≤ 2 GB	2x RAM size
2 GB to 8 GB	Same as RAM size
> 8 GB	8 GB

Contents

Drive Types



Drive Interface Description

Drive Interface	Description
Integrated Drive Electronics (IDE)	The earliest general standard for HDDs. All electronic integrated drives, including SCSI, adopt the IDE standard.
Serial ATA (SATA)	SATA is distinct from IDE, which is also known as Parallel ATA (PATA). Therefore, the IDE interface is often called the parallel port and the SATA interface is called the serial port.
Small Computer System Interface (SCSI)	Drives that adopt the SCSI interface are SCSI drives. Serial Attached SCSI (SAS) is a serial implementation of the SCSI interface. SCSI drives are usually used on servers. Compared with PATA and SATA drives, SCSI drives provide higher performance, stronger stability, and support hot swap. However, SCSI drives have disadvantages such as small capacity, loud noise, and high cost.
Fibre Channel (FC)	FC interfaces enable drives to connect directly using optical fibers, improving the I/O performance of high-throughput performance-intensive systems.

Viewing Drive Information on Linux (1)

- **fdisk -l** is used to view information about all drives in the system, including mounted and unmounted drives.

```
[root@openEuler ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: AA321D82-C833-4D3E-885C-52FC0ADF3860

Device       Start      End  Sectors  Size Type
/dev/vda1     2048   411647   409600  200M EFI System
/dev/vda2   411648  2508799  2097152    1G Linux filesystem
/dev/vda3  2508800 83884031 81375232 38.8G Linux LVM

Disk /dev/mapper/openeuler-root: 34.82 GiB, 37367054336 bytes, 72982528 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Viewing Drive Information on Linux (2)

- **df -h** is used to check the mounting status, spaces, and usage of the drives.

```
[root@openEuler ~]# df -h
Filesystem                    Size  Used  Avail  Use% Mounted on
devtmpfs                      1.2G     0   1.2G    0%  /dev
tmpfs                         1.5G     0   1.5G    0%  /dev/shm
tmpfs                         1.5G   18M   1.5G    2%  /run
tmpfs                         1.5G     0   1.5G    0%  /sys/fs/cgroup
/dev/mapper/openeuler-root     35G   4.4G   28G   14%  /
tmpfs                         1.5G    64K  1.5G    1%  /tmp
/dev/vda2                     976M  125M  785M   14%  /boot
/dev/vda1                     200M  5.8M  195M   3%   /boot/efi
tmpfs                         298M     0   298M  0%   /run/user/0
tmpfs                         298M     0   298M  0%   /run/user/993
```

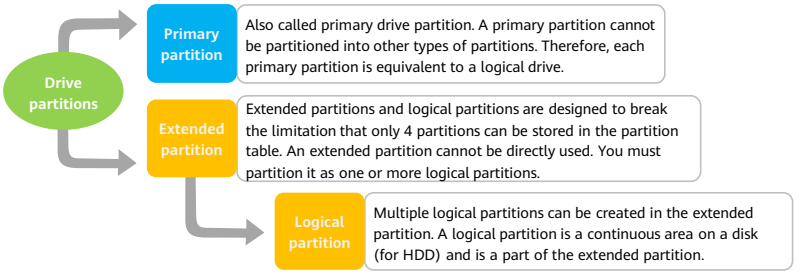
Contents

Drive Partitioning

- Through drive partitioning, a drive is divided into multiple logical storage units called partitions. The system administrator can use different partitions for different functions.
- Advantages of drive partitioning:
  - The available space of applications or users can be restricted.
  - The machine can boot into multiple OSs from different partitions on the same drive.
  - OS files are separated from program and user files.
  - A separate area can be created for OS virtual memory swapping.
  - Drive space usage can be restricted to improve the performance of diagnosis tools and image backups.

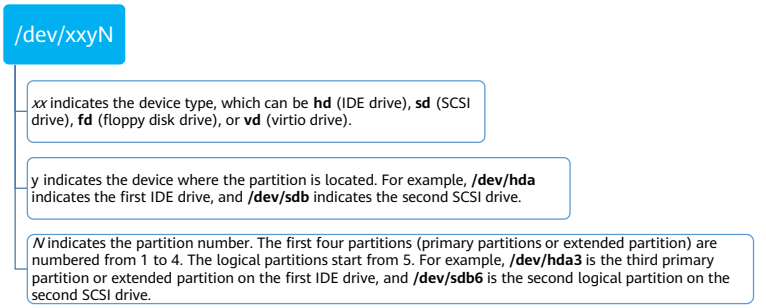
Drive Partition Types

- By partitioning the drive, you are modifying the drive partition table. Pay attention to the following while partitioning:
  - For the continuity of data, you are advised to place the extended partition in the last cylinders.
  - A drive has only one extended partition. Except the primary partition space, the rest space is allocated to the extended partition.
  - Drive capacity = primary partition capacity + extended partition capacity; Extended partition capacity = Sum of the capacities of all logical partitions.



Drive Partition Naming

- There are no drive letters in Linux. You can access a device through the device name. The device names are stored in the **/dev** directory.
- The devices are named as follows:



- Note: On Linux, SAS drives, SATA drives, and SSDs are identified by **sd**, and IDE drives are identified by **hd**.

Drive Partitioning Scheme - MBR

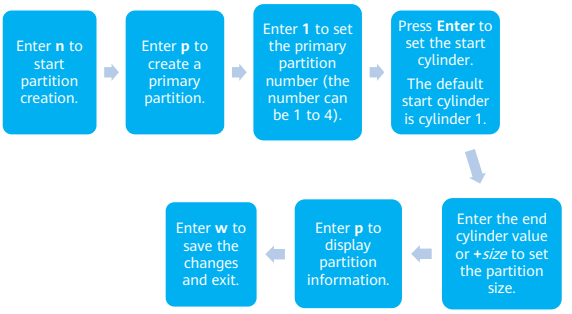
- Master Boot Record (MBR)
  - The MBR partitioning scheme specifies how the drive of a system running BIOS is partitioned. The MBR is a special boot sector at the beginning of the drive.
  - A SCSI drive partitioned using the MBR scheme can have a maximum of 15 partitions, among which a maximum of 4 primary partitions or 12 logical partitions can be created. Extended partitions cannot be directly used and therefore are not counted. An IDE drive partitioned using the MBR scheme can have a maximum of 63 partitions, among which a maximum of 4 primary partitions or 60 logical partitions can be created. Extended partitions cannot be directly used and therefore are not counted.
  - Because partition size data is stored as 32-bit values, the maximum size of a drive or partition cannot exceed 2 TB when the MBR scheme is used.

fdisk Drive Partitioning Utility

- fdisk is a commonly used traditional Linux drive partitioning utility. fdisk does not support partitions larger than 2 TB.
- The command syntax is as follows:
  - **fdisk [options]** *parameter*
  - Common options are as follows:
    - **-b <partition size>**: specifies the size of each partition.
    - **-l**: lists the partition tables of the specified peripheral devices.
    - **-s <partition number >**: prints the specified partition size in sectors to the standard output.
    - **-u**: used with **-l** to display the start address of each partition in the unit of sectors instead of cylinders.
    - **-v**: displays version information.

fdisk Interactive Mode

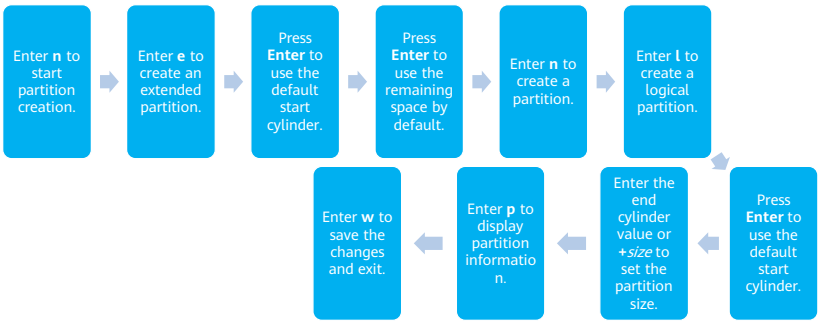
- Select a drive to enter the interactive mode for drive partitioning.
- For example: **fdisk /dev/sdb**. The following table lists the interactive mode commands.
- The process of creating a primary partition is as follows:



Command	Description
a	Set a bootable flag.
b	Edit the BSD disklabel.
c	Set the DOS compatibility flag.
d	Delete a partition.
l	Lists known file system types. 82 indicates a Linux swap partition, and 83 indicates a Linux partition.
m	Displays help menu.
n	Create a partition.
o	Create an empty DOS partition table.
p	List the partitions.
q	Quit without saving changes.
s	Creating an empty SUN partition table.
t	Change the system ID of a partition.
u	Change the unit of the displayed records.
v	Verify the partition table.
w	Save and exit.

fdisk Interactive Mode

- The process of creating an extended partition is as follows:



- Note: The extended partition cannot be directly used after creation. You must create a logical partition.

Drive Partitioning Scheme - GPT

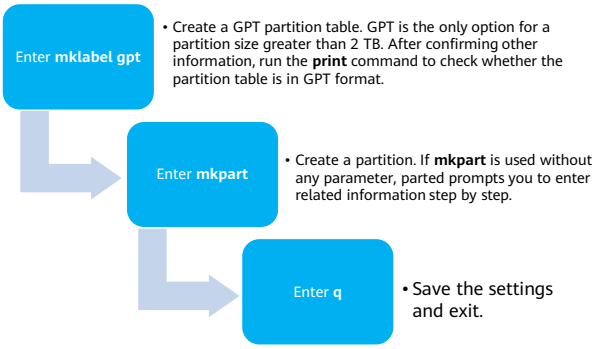
- GUID Partition Table (GPT)
  - As drives continue to increase in capacity, the 2 TB drive and partition size restriction of the old MBR partitioning is no longer a theoretical limit, but an actual problem that is frequently encountered in the production environment. As a result, GPT is replacing the traditional MBR solution for drive partitioning.
  - GPT assigns a globally unique identifier (GUID) to each partition on a drive. For systems that run Unified Extensible Firmware Interface (UEFI) firmware, GPT is the standard for arranging partition tables on physical drives.
  - There are no primary or logical partitions with GPT, and each drive can have a maximum of 128 partitions. As GPT uses 64 bits for logical block addresses, a maximum partition size of 18 EB is supported.

parted Drive Partitioning Utility

- parted is a partitioning utility commonly used in Linux to create drive partitions larger than 2 TB. Compared with fdisk, parted is more convenient and enables the partition size to be dynamically adjusted. The command syntax is as follows:
- **parted [options] [*device* [*command* [options...]...]]**
  - The command options are as follows:
    - **-h**: displays help information.
    - **-i**: enters the interactive mode.
    - **-s**: enters the script mode.
    - **-v**: displays parted version information.
    - *device*: indicates the drive device name, for example, **/dev/sda**.
    - *command*: indicates a parted command. If no command is specified, parted enters the interactive mode.

parted Interactive Mode

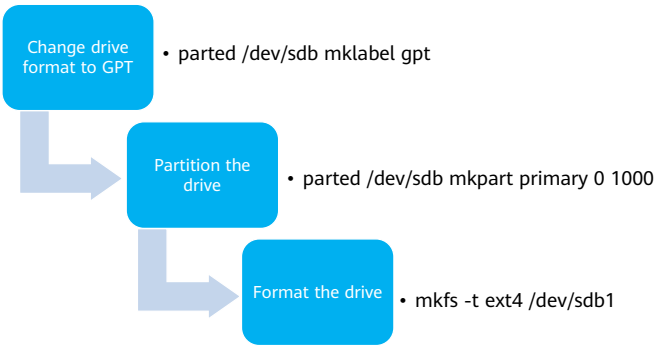
- Select a drive to enter the interactive mode for drive partitioning.
- For example: **parted /dev/sdb**.
- The table lists the interactive mode commands.



Command	Description
align-check	Check if partition **N** satisfies the alignment constraint of type **min** or **opt**.
mklabel	Create a disklabel (partition table).
name	Name a specified partition.
print	Display partition the table or partitions.
rescue	Rescue a lost partition.
resizepart	Resize a partition.
rm	Delete a partition.
select	Select the device to be edited. By default, only the specified device is operated. You can change the specified device here.
disk_set	Changes the flag on the selected device.
disk_toggle	Switch the state of the flag on the selected device.
quit	Exit.
set	Change the flag on a partition.
toggle	Set or unset the flag on a partition.
unit	Set the default unit.
version	Display version information.

parted Non - Interactive Mode

- Configure a selected drive using the non-interactive mode, that is, the command line mode.
- For example: **parted /dev/sdb**



Contents

Drive Formatting

- Formatting involves initializing a drive or a partition in a drive, and formatting a partition into different file systems. This operation usually causes all files in an existing drive or partition to be deleted.

For details about the file types supported by Linux, enter **mkfs** and press **Tab** twice to view commands for supported file types. Select a required command to format the drive.

Run **mkfs.ext4 /dev/sdb2** to format the sdb2 partition under the root into ext4.

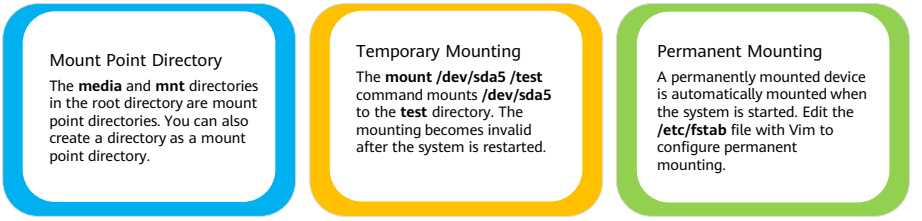
After the formatting is completed, run the **ll** command to view drive information, for example, **ll /dev/sdb2**.

mkfs Drive Formatting Command

- Short for "make file system", **mkfs** is used to create a Linux file system in a specified partition.
- Syntax: **mkfs [-V] [-t fstype] [fs-options]** ***filesys*** **[*blocks*]**
  - The command options are as follows:
    - ***filesys***: the drive partition to be checked. For example, **/dev/sda1**.
    - **-V**: produces verbose output.
    - **-t**: specifies the type of file system. The default value for Linux is **ext2**.
    - **-c**: checks the partition for bad blocks before building the file system.
    - **-l** ***bad_blocks_file***: reads information about bad blocks from the **bad_blocks_file** file.
    - ***blocks***: specifies the number of blocks to be used.

Drive Mounting

- A formatted drive has to be mounted before you can use it. The reasons are as follows:
  - Linux adopts "Everything is a file" design. To use the drive, you must set up a link between the drive and a file (directory). The process of setting up a link is called mounting.
  - When you access the directory linked to the sdb2 drive, you are accessing the **sdb2** device file. This directory is equivalent to an entry or interface for accessing sdb2. The drive can be accessed only when this interface is available.



Introduction to fstab

- Functions of the **/etc/fstab** file
  - This file stores static information about a file system. Upon startup, the OS automatically reads information from the file and mounts the specified file systems to directories accordingly. By writing mounting information to the file, you do not need to manually mount the drives after each startup.
- Format of **fstab**

```
<file system>   <dir>   <type>    <options>          <dump> <pass>
 tmpfs          /tmp     tmpfs    nodev,nosuid         0      0
/dev/sda1       /        ext4     defaults,noatime     0      1
/dev/sda2       none     swap     defaults             0      0
```

Key Parameters in fstab

Field	Parameter	Function
options	auto	Automatically mounts at startup or after the **mount -a** command is entered.
	ro	Mounts a file system as read-only.
	rw	Mounts a file system as read/write.
	user	Allows any user to mount the file system.
	nouser	Allows only the **root** user to mount the file system.
	dev/nodev	Does or does not interpret block special devices on the file system.
	noatime/ nodiratime	Does not update directory inode access records on this file system or directory. This option improves the performance.
	defaults	Uses the default mounting parameters for the file system.
	sync/async	Specifies synchronous/asynchronous I/O.
	suid/nosuid	Does or does not honor the SUID and SGID bits. This option is typically used for certain special tasks so that common users can temporarily escalate their privileges when running programs.
dump	0 /1	Specifies whether to back up the file system (**1**) or not (**0**) As most users do not install dump, the value of <dump> should be set to 0.
pass	0, 1, 2	Sets the fsck check priority for a file system. This value should be set to 1 for the root partition and 2 for other devices that need to be checked. 0 indicates that the device will not be checked by fsck.

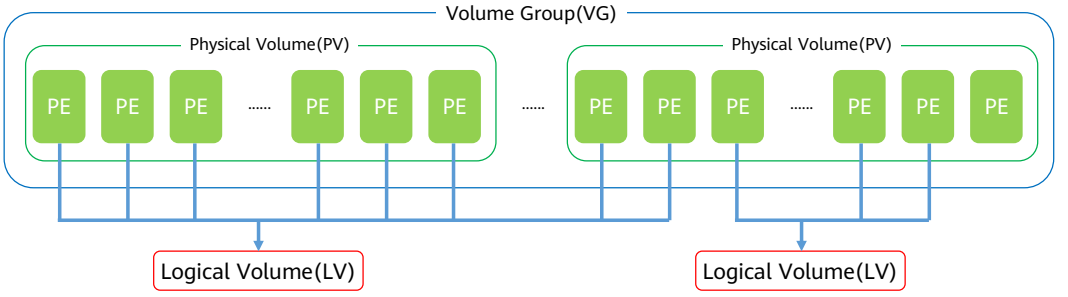
Contents

Logical Volume

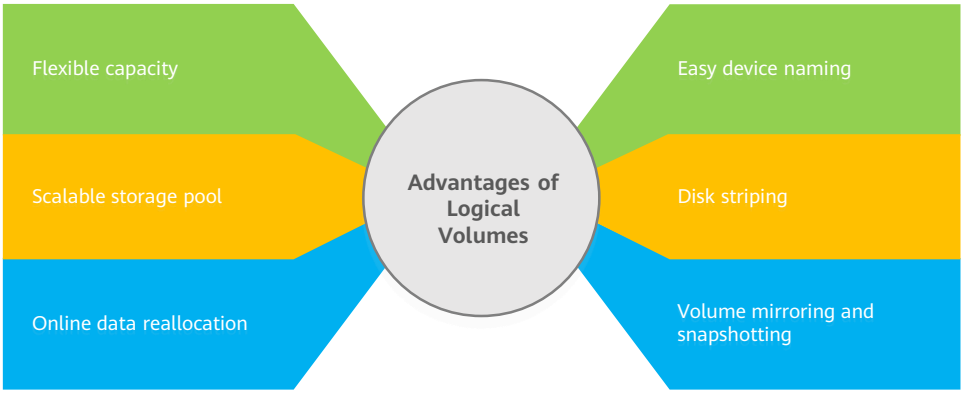
- Logical volume management (LVM) is a mechanism for managing drive partitions in Linux. It is a logical layer built above drives and partitions and below the file systems, which improves the flexibility of drive partition management.
  - Physical extents (PE): a PE with a unique number is the smallest unit that can be addressed by LVM. The sizes of PEs can be specified, with 4 MB being the default. Once the PE size is determined, it cannot be changed. The PE sizes of all physical volumes in the same volume group are the same.
  - Logical extents (LE): the minimum storage unit that can be allocated in LVM. The size of LE depends on the size of PE in the volume group (VG) where the logical volume (LV) is located. Within the same VG, the size of the LE is the same as that of the PE. Generally, the LE and PE are in one-to-one correspondence.
  - Physical volume (PV): an underlying device that provides capacity and stores data. A PV can be an entire drive or a partition on a drive.
  - Volume group (VG): a VG is created based on PVs and consists of one or more of them. Specifically, PVs are combined to provide capacity allocation. An LVM system can contain one or more VGs.
  - Logical volume (LV): an LV is created based on a VG. It is a logical device used by end users, and its size can be expanded or reduced.

Logical Volume Principles

- An LV is a large extended partition (VG) consisting of several drive partitions or block devices (PVs). Note that the PVs can be located in different drive partitions and their sizes can be different, and a VG must contain at least one PV. The extended partition cannot be used directly, and the LV can only be used after it is divided into LVs. The LVs can be formatted into different file systems and can be directly used after being mounted.

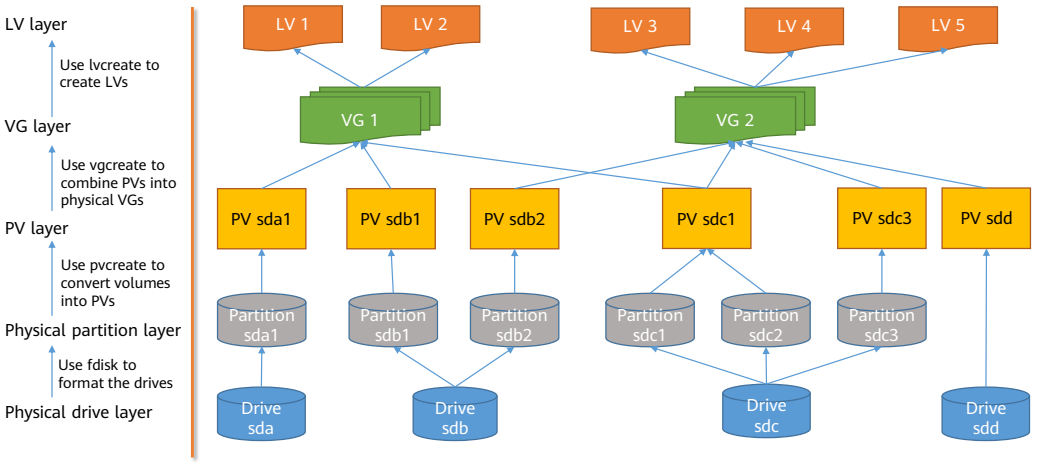


Advantages of Logical Volumes



Contents

Process of Creating a Logical Volume



HUAWEI

Logical Volume Management - Using PVs

- The **pvcreate** command is used to create a PV using a physical drive or drive partition.
- Syntax: **pvcreate [option]** ***device_file_name***
  - The command options are as follows:
    - **-f**: forcibly creates a PV without user confirmation.
    - **-u**: specifies the UUID of the device.
    - **-y**: answers yes to all questions.
  - The command parameter is as follows:
    - ***device_file_name***: specifies the device file name for the PV to be created.

  Note: After a partition is created, the partition ID is 83 by default. You need to change the ID to 8e to create a PV based on the partition. You can run the **fdisk** command to change the ID.

Logical Volume Management - PV Usage Example

- Create PVs based on drive partitions 6 to 9. Note the use of the braces.
  - Run the following command:

```
[root@openEuler ~]#pvcreate /dev/hda{6,7,8,9}
```

- Run the **pvdisplay**, **pvscan**, or **pvs** command to view PV information.
  - For example, run the **pvs** command:

```
[root@openEuler ~]#pvs # View the general information about the PVs
```

  - Output:

```
  PV         VG     fmt  Attr  PSize   PFree
/dev/sdb1  vg1000 lvm2  --   100.00M 100.00M
/dev/sdb2         lvm2  --   101.98M 101.98M
```

Logical Volume Management - Using VGs

- The **vgcreate** command is used to create a VG of LVM. A VG combines multiple PVs into a whole, shielding the details of the underlying PVs. When creating an LV on a VG, you do not need to consider the PVs.
- Syntax: **vgcreate [option]** ***VG_name PV_list***
  - The command options are as follows:
    - **-l**: specifies the maximum number of LVs that can be created on the VG.
    - **-p**: specifies the maximum number of PVs that can be added to the VG.
    - **-s**: specifies the PE size of a PV in the VG.
  - The command parameter is as follows:
    - ***VG_name***: name of the VG to be created.
    - ***PV_list***: list of PVs to be added to the VG.

Logical Volume Management - VG Usage Example

- Run the **vgcreate** command to create the **vg1000** VG, and then add the **/dev/sdb1** and **/dev/sdb2** PVs.
  - Run the following command:

```
[root@openEuler ~]#vgcreate vg1000 /dev/sdb1 /dev/sdb2
```

- Run the **vgdisplay** or **vgscan** command to view VG information.
  - For example, run the **vgdisplay** command:

```
[root@openEuler ~]#vgdisplay vg1000
```

Note: If **vg1000** is not specified, information about all VGs is displayed.

Logical Volume Management - Using LVs

- The **lvcreate** command is used to create an LV of LVM. LVs are created on VGs.
- Syntax: **lvcreate [option]** ***LV_name***
  - The command options are as follows:
    - **-L** specifies the size of the LV. The unit can be K, M, G, or T (case-insensitive).
    - **-l**: specifies the size of the LV (number of LEs).
  - The command parameter is as follows:
    - ***LV_name***: specifies the name of the LV to be created.

    Note: A created LV has to be formatted and mounted before use. The method is the same as that described in section 2.4. That is, use **mkfs** to format the LV and use **mount** to mount the LV to a directory.

Logical Volume Management - LV Usage Example

- Run the **lvcreate** command to create a 200 MB LV on the **vg1000** VG.
  - Run the following command:

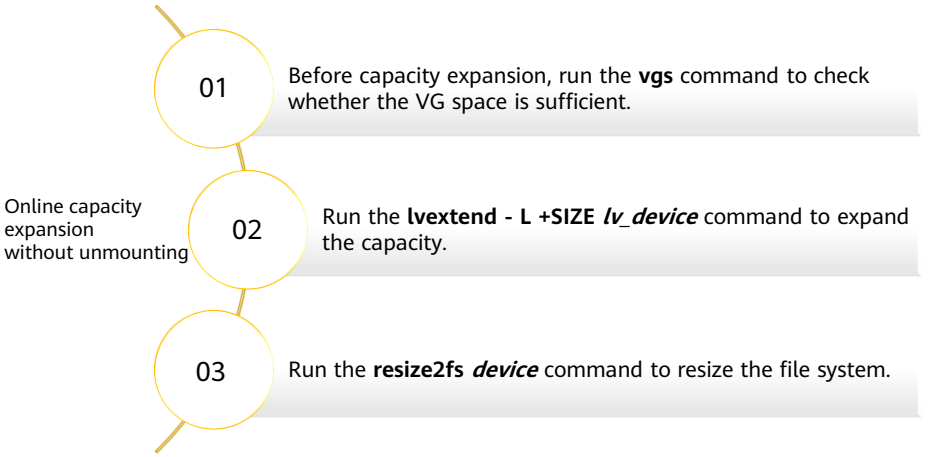
```
[root@openEuler ~]#lvcreate -L 200M vg1000
```

- Run the **lvdisplay** or **lvscan** command to view LV information.
  - For example, run the **lvscan** command:

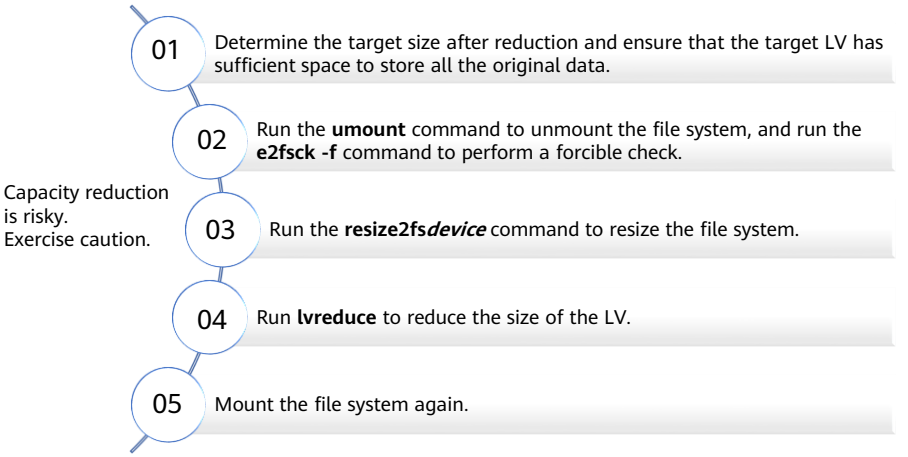
```
[root@openEuler ~]#lvscan # Scan all LVs.
```

Contents

Expanding Logical Volume Capacity



Reducing Logical Volume Capacity



Changing Logical Volume Capacity

- The **lvresize** command is used to adjust (increase or decrease) the size of an LV. As the **lvresize** command provides the **lvextend** and **lvreduce** functions, the procedures for expanding and reducing the capacity of an LV are the same.
- Syntax: **lvresize [option]** ***LV_name***
  - The command options are as follows:
    - **-L** specifies the size of the LV. The unit can be K, M, G, or T (case-insensitive).
    - **-l**: specifies the size of the LV (number of LEs).
  - The command parameter is as follows:
    - ***LV_name***: specifies the name of the LV to be created.
- For example, run the **lvresize** command to increase the capacity:

```
[root@openEuler ~]# lvresize -L +200M /dev/vg1000/lvol0 # Increase the LV space by 200 MB.
```

Quiz

1. (Single-answer) Which of the following commands can be used to format the **/dev/hdb6** partition?
   A. mkfs -t ext4 /dev/hdb6
   B. format -t ext4 /dev/hdb6
   C. format /dev/hdb6
   D. makefile -t ext4 /dev/hdb6
2. (True or false) Because reducing LV capacity is risky, you need to unmount and forcibly check the file system.
   A. True
   B. False

HUAWEI

- Answer:
  - A
  - A

Summary

- This course describes the concept of the file system, how to mount and use drives, and how to manage logical volumes.

Acronyms

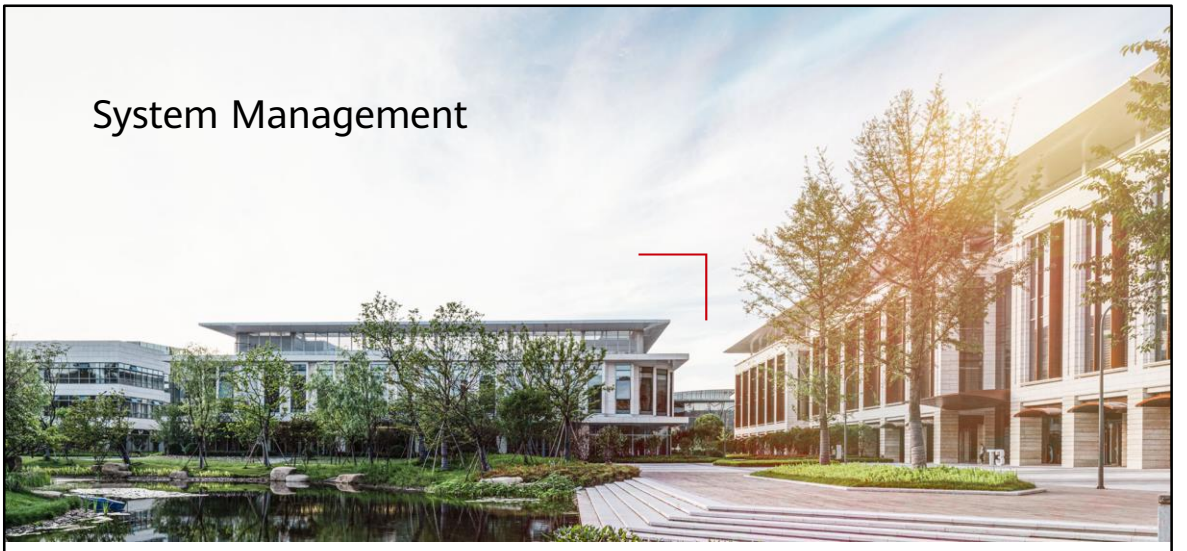
Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

System Management



HUAWEI

Foreword

- This chapter focuses on openEuler task, network, and process management, including scheduling.

Objectives

- Upon completion of this course, you will be familiar with:
  - One-off and scheduled task configuration
  - openEuler network configuration, ensuring hosts can communicate
  - openEuler process viewing and management

Contents

What Is Task Management?

- During system O&M, a specific task may need to be executed at a specific time.
- For example, sending periodic emails or backing up and clearing log files.
- The content of a task can be regarded as a series of commands, or a script, which needs to be executed at a specific time.

08:00 send email
15:00 reset log.txt
...

at, **cron**, **anacron**

send-email.sh, **reset.sh**...

Task Management Types

- **at**: scheduled task executed once, at a specified time.
- **cron**: task executed periodically for multiple times.

Using the at Command for Scheduled Execution

- The **at** command can be used to specify the time Linux will run scripts.
- The atd process is the daemon process of **at**. It runs in the background when the system is started.
- The atd process periodically checks the **/var/spool/at** directory to get the jobs written by running the **at** command.

```
[root@openEuler ~]# systemctl status atd
● atd.service - Deferred execution scheduler
   Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-07-08 11:20:10 CST; 2 weeks 6 days ago
     Docs: man:atd(8)
 Main PID: 2276 (atd)
    Tasks: 1
   Memory: 2.1M
   CGroup: /system.slice/atd.service
           └─2276 /usr/sbin/atd -f
Jul 08 11:20:10 localhost.localdomain systemd[1]: Started Deferred execution scheduler.
```


- The **at** command may not be installed in the openEuler OS. Run the following commands to install and start the **at** command:
  - **yum install -y at**
  - **systemctl start at**

at Command

- The **at** command is used to schedule commands to be executed once only.
- This command requires at least one command and one execution time.
- The **at** command can specify only the time or both time and date.
- **at** command syntax:

```
at [-V] [-q queue] [-f file_name] [-mldbv] time
at -c job [job...]
```


- The concept of "jobs" is the same as that of "tasks", "a series of commands", and "command sequences".

Setting an Absolute Time

- The scheduled time can be written in any of the following formats:
  - *hh:mm* (hour:minute) on the current day. If the time has already passed, the job will be executed the following day.
  - **midnight**, **noon**, **teatime** (16:00)
  - 12-hour time followed by **am** or **pm**
  - Time + date (*month day*, *mm/dd/yy*, or *dd.mm.yy*)

Setting a Relative Time

- Relative timing is convenient for commands which need to be executed soon.
  - **now+***N* **minutes**, **hours**, **days**, or **weeks**
  - *N* indicates the number of minutes, hours, days, or weeks. Another method is to use **today** and **tomorrow** to specify the time to complete commands.
  - Assume that the current time is 12:30 on June 7, 2015, and you want to run a command at 4:30 pm.

at 4:30pm at 16:30 at 16:30 today at now+4 hours	at now+240 minutes at 16:30 7.6.15 at 16:30 6/7/15 at 16:30 Jun 7

  - These eight commands all give the same result.

- Input **at** *time*. The interactive interface is displayed. Press **Ctrl** + **d** to end the command input.
- Input **at -f** *file-name***.sh** *time* to execute a specific script file.
- Run the **atq** command to view the job queue.
- Run the **atrm** command to delete a job from the job queue. The format is **atrm** *job ID*. The job ID is displayed in the first column of the **atq** command output.

Execution Permission

- The **at** command can only schedule commands from standard input or from the file specified by the **-f** option.
- If user A switches to user B using the **su** command and then runs **at**, the **at** command's execution result will be sent to user B.
- For other users, running a command or script depends on the **/etc/at.allow** and **/etc/at.deny** files.

```
[root@openEuler ~]# at -f example.sh 15:00
warning: commands will be executed using /bin/sh
job 1 at Wed Jul 29 15:00:00 2020
[root@openEuler ~]# atq
1       Wed Jul 29 15:00:00 2020 a root
```

Using the cron Command to Run Periodic Commands

- The **at** command can run commands at a scheduled time, but only once.
- If you need to run commands more than once, **cron** is a good helper.

Cron Running Mechanism

- The cron service searches the **/var/spool/cron** directory for the crontab file named after the username in the **/etc/passwd** file, and then loads the crontab file into the memory.
- It also searches for the **/etc/crontab** (timetable) file, which is written in a specific format.
- If no crontab files are found, the cron service enters sleep mode and releases system resources. The cron service then wakes up every minute to check whether there are commands to be executed.

- The cron service searches the **/var/spool/cron** directory for the crontab file named after the username in the **/etc/passwd** file, and then loads the crontab file into the memory. Each user has a crontab file, with the same name as the user name. For example, the crontab file of the **globus** user is **/var/spool/cron/globus**.
- The **cron** command also searches for the **/etc/crontab** file, which is written in a different format from files in **/var/spool/cron**. If no crontab files are found, the cron service enters sleep mode and releases system resources. The cron service then wakes up every minute to check whether there are commands to be executed.
- Command execution results are then mailed to users specified by the environment variable **MAILTO** in the **/etc/crontab** file. The cron service, once started, does not require manual intervention except when you need to replace periodic commands with new ones.

Crontab Command

- The **crontab** command is used to install, edit, remove, and list crontab files.
- Users can place the sequence of commands to be executed in crontab files. Each user can have their own crontab file.
- Common **crontab** commands:

```
crontab -u  //Set a user's cron service. This option is required only when the crontab command is run by the root user.
crontab -l  //List details about a user's cron service.
crontab -r  //Remove a user's cron service.
crontab -e  //Edit a user's cron service.
```

Crontab Files

- A crontab file contains a list of commands meant to be run at certain times.
- Each crontab line has five time-and-date fields, followed by a command.
- The fields are separated by spaces and tabs. The format is as follows:

```
minute hour day-of-month month-of-year day-of-week commands
```

Crontab File Parameters

minute hour day-of-month month-of-year day-of-week commands

Parameter	Description
minute	Minute (0–59)
hour	Hour (0–23)
day-of-month	Day of a month (1-31)
month-of-year	Month of a year (1-12)
day-of-week	Day of a week (0–6). The value **0** indicates Sunday.
commands	Commands to be executed

For example, run a series of commands every Monday at 08:00 a.m.

00 08 * * 1 *commands*

Supplementary Description of Crontab File Parameters

- The parameters on the previous slide cannot be left blank. They must be set.
- An absolute path to the commands shall be provided.
- In addition to numerical values, the following special symbols are allowed:

Parameter	Description
*	All numbers within the value range
/	Step value. For example, ***/5** indicates every five steps.
-	Value range
,	Several discrete numbers

For example, add **sleepy** to the **/tmp/test.txt** file every two hours from 11:00 p.m. to 8:00 a.m. The corresponding line in the crontab file is as follows:

```
* 23-8/2 * * * echo"sleepy" >> /tmp/test.txt
```


- Each time the cron service settings of a user are edited, the cron service generates in the **/var/spool/cron** directory a crontab file named after the user. The crontab file can be edited only using the **crontab -e** command. Alternatively, the user can create a file and run the **crontab** *filename* command to import the cron settings to the new file.
- Do not restart the cron service after a crontab file is modified, because the cron service, once started, reads the crontab file every minute to check whether there are commands that need to be executed periodically.

Editing the Configuration File

- The cron service reads, every minute, **/etc/crontab** and all files in **/var/spool/cron**.
- A crontab file contains user-specific commands. The **/etc/crontab** file contains system-wide commands.
- The file format is as follows:

```
SHELL=/bin/sh
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root          //If an error occurs or data is output, the data is sent to this account by email.
HOME=/
# run-parts01
* * * * root run-parts /etc/cron.hourly  // Execute the scripts in /etc/cron.hourly every hour.
02 4 * * * root run-parts /etc/cron.daily          // Execute the scripts in /etc/cron.daily once a day.
22 4 * * 0 root run-parts /etc/cron.weekly         // Execute the scripts in /etc/cron.weekly once a week.
42 4 1 * * root run-parts /etc/cron.monthly        // Execute the scripts in /etc/cron.monthly once a month.
```


- If **run-parts** parameter is deleted, the following text, for example **/etc/cron.hourly**, is treated as a script name instead of a directory name.

Contents

Important openEuler Network Concepts (1)

- Device
  - NIC on the host
- Broadcast address
  - Address used to send packets to all hosts on the network segment
- Interface
  - Created by drivers on the devices so that the system can use the devices

Important openEuler Network Concepts (2)

- Subnet mask
  - Number that distinguishes the network address and the host address within an IP address
- Route
  - Next-hop IP address when IP packets are transmitted across network segments
- Link
  - Device-to-network connection

Example of openEuler Device Information

```
[root@openEuler ~]# ip addr          // Check devices, including unconfigured network devices.
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 28:6e:d4:89:70:de brd ff:ff:ff:ff:ff:ff
    inet 192.168.110.246/24 brd 192.168.110.255 scope global dynamic noprefixroute enp4s0
       valid_lft 552233sec preferred_lft 552233sec
    inet6 fe80::fc7e:f7ba:e0d0:2f1f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:c7:04:f6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:c7:04:f6 brd ff:ff:ff:ff:ff:ff
```


- Consistent network device naming: https://www.freedesktop.org/software/systemd/man/systemd.net-naming-scheme.html
- **ip a** is equivalent to **ip addr**.

Configuration File of the openEuler NIC

- Default configuration path of network devices: **/etc/sysconfig/network-scripts/**
- Configuration file: **ifcfg-***

```
[root@openEuler ~]# ls -l /etc/sysconfig/network-scripts/
total 4
-rw-r--r--. 1 root root 308 Jul  8 11:07 ifcfg-enp4s0
```

NIC Configuration File Parameters

- Parameter values are case insensitive and quotation marks are not required.

Parameter	Description
TYPE	Interface type
BOOTPROTO	Boot-time protocol
ONBOOT	Whether to activate the device at boot-time
IPADDR	IP address
NETMASK	Subnet mask
GATEWAY	Gateway address
BROADCAST	Broadcast address
HWADDR/MACADDR	MAC address. You only need to set one MAC address. MAC addresses cannot conflict with each other.
PEERDNS	Whether to specify the DNS server address. If the DHCP protocol is used, the default value is **yes**.
DNS{1, 2}	DNS address
USERCTL	User permission control
NAME	Network connection name
DEVICE	Physical interface name

Modifying the Configuration File

- Use an editor to modify the configuration file. Back up the file before modifying it.
- Modification does not take effect immediately. For the modification to take effect, restart either the NetworkManager service process or the system.

```
Back up the configuration file.
[root@openEuler ~]# cp ifcfg-eth1 ifcfg-eth1.bak

Restart the NetworkManager service process.
[root@openEuler ~]# systemctl reload NetworkManager
```

Example of a Minimal NIC Configuration File

- An available NIC configuration file does not need to list all configuration options. This configuration, for example, is valid:

```
TYPE=Ethernet
BOOTPROTO=static
NAME=enp0s3
DEVICE=enp0s3
ONBOOT=yes
IPADDR=192.168.56.100
NETMASK=255.255.255.0
```


- Sometimes, too many configuration items increase the difficulty of network troubleshooting.

Viewing the IP Address

- Run the **ip** command to view device and address information.

```
[root@openEuler ~]# ip addr show enp4s0
2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
1000
    link/ether 28:6e:d4:89:70:de brd ff:ff:ff:ff:ff:ff
    inet 192.168.110.246/24 brd 192.168.110.255 scope global dynamic noprefixroute enp4s0
       valid_lft 552070sec preferred_lft 552070sec
    inet6 fe80::fc7e:f7ba:e0d0:2f1f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

No.	Description
1	Status of the current interface. **UP**: The current interface is enabled. **DOWN**: The current interface is disabled.
2	Link information: MAC address of the device
3	inet information: IPv4 address and subnet mask
4	Broadcast address, scope, and device name
5	inet6 information: IPv6 information

- To view the current IP addresses and subnet masks of all interfaces, run the **ip addr** command.

Viewing Interface Statistics

- Run the **ip** command to view network performance statistics.
- **RX** indicates received data packet information.
- **TX** indicates transmitted data packet information.

```
[root@openEuler ~]# ip -s link show enp4s0
2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 28:6e:d4:89:70:de brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    20274318   204775   0       15      0       0
    TX: bytes  packets  errors  dropped carrier collsns
    1057639    7024     0       0       0       0
```

Viewing Socket Information

- Run the **ss** command to view socket statistics. You can add parameters to filter sockets of different types. For example, to view information about all TCP sockets, run the following command:

```
[root@openEuler ~]# ss -ta
State       Recv-Q      Send-Q           Local Address:Port           Peer Address:Port      Process
LISTEN      0           128                    0.0.0.0:sunrpc              0.0.0.0:*
LISTEN      0           32               192.168.122.1:domain              0.0.0.0:*
LISTEN      0           128                    0.0.0.0:ssh                 0.0.0.0:*
LISTEN      0           5                      0.0.0.0:44321               0.0.0.0:*
LISTEN      0           5                      0.0.0.0:dey-sapi            0.0.0.0:*
ESTAB       0           0              192.168.110.246:ssh          172.19.130.180:56950
LISTEN      0           128                       [::]:sunrpc                 [::]:*
LISTEN      0           128                       [::]:ssh                    [::]:*
LISTEN      0           5                         [::]:44321                  [::]:*
LISTEN      0           5                         [::]:dey-sapi               [::]:*
```

NetworkManager

- NetworkManager is a daemon process for dynamically controlling and configuring networks in the system. It is used to keep the current network devices and connections working.
- The nmcli command line tool can be used to control NetworkManager. nmcli is comprehensive, powerful, and sophisticated.
- Key concepts
  - Device: On a network, a device is a network interface.
  - Connection: A connection is a configuration used by a device.
  - A device may have multiple connections, but only one connection at a time can be active.

Viewing Network Information Using nmcli

- con indicates a connection. The **--active** option can be used to filter active connections.

```
[root@openEuler ~]# nmcli connection show --active
NAME    UUID                                  TYPE      DEVICE
enp4s0  1d859d5a-b0c0-3b49-b153-269e0d4a85ce  ethernet  enp4s0
virbr0  52137c43-98c9-4f34-8954-d3d1ea89b946  bridge    virbr0
```


- Run the **ncmcli con show** *connection_name* command to display detailed information about a single connection. This command is similar to the **ip link** command.
- Run the **ncmcli dev status** command to display the device status.
- Run the **ncmcli dev show [device_name]** command to display the device information.

Creating a Connection Using nmcli

1. Create a connection named **Demo** and connect it to the **enp4s0** network port in DHCP mode.
2. Display all connections.
3. Start the created **Demo** connection.
4. Check the status of the connection.

```
[root@openEuler ~]# nmcli connection show --active
NAME    UUID                                  TYPE      DEVICE
enp4s0  1d859d5a-b0c0-3b49-b153-269e0d4a85ce  ethernet  enp4s0
virbr0  52137c43-98c9-4f34-8954-d3d1ea89b946  bridge    virbr0
[root@openEuler ~]# nmcli connection show
NAME    UUID                                  TYPE      DEVICE
enp4s0  1d859d5a-b0c0-3b49-b153-269e0d4a85ce  ethernet  enp4s0
virbr0  52137c43-98c9-4f34-8954-d3d1ea89b946  bridge    virbr0
[root@openEuler ~]# nmcli connection add con-name 'Demo' type Ethernet ifname enp4s0
Connection 'Demo' (789c723b-cd60-4515-99fc-5074b6f38498) successfully added.
[root@openEuler ~]# nmcli connection show
NAME    UUID                                  TYPE      DEVICE
enp4s0  1d859d5a-b0c0-3b49-b153-269e0d4a85ce  ethernet  enp4s0
virbr0  52137c43-98c9-4f34-8954-d3d1ea89b946  bridge    virbr0
Demo    789c723b-cd60-4515-99fc-5074b6f38498  ethernet  --
[root@openEuler ~]# nmcli connection up 'Demo'
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/16)
[root@openEuler ~]# nmcli connection show
NAME    UUID                                  TYPE      DEVICE
Demo    789c723b-cd60-4515-99fc-5074b6f38498  ethernet  enp4s0
virbr0  52137c43-98c9-4f34-8954-d3d1ea89b946  bridge    virbr0
enp4s0  1d859d5a-b0c0-3b49-b153-269e0d4a85ce  ethernet  --
```


- **nmcli con add con-name "Euler" type Ethernet ifname enp3s0**
- **nmcli con show**
- **nmcli con up "Euler"**
- **nmcli con show**
- There are many optional types for **nmcli con add**, such as **bridge**, **bind**, **vpn**, and **vxlan**. To view all options, run the **nmcli con add help** command.

Modifying a Connection Using nmcli (1)

- Run the **nmcli con mod** command to modify a connection. The input parameters are key-value pairs.
- The key is the attribute name, which can be queried by running the **nmcli con show [***connection_name***]** command. For example:

```
[root@openEuler ~]# nmcli connection show Demo
connection.id:                          Demo
connection.uuid:                        789c723b-cd60-4515-99fc-5074b6f38498
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              enp4s0
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1595988280
```


- This slide only lists certain attribute names.

Modifying a Connection Using nmcli (2)

- The following uses the domain name resolution (DNS) server as an example:

```
[root@openEuler ~]# nmcli connection modify 'Demo' ipv4.dns 192.168.100.250
[root@openEuler ~]# nmcli connection show Demo | grep ipv4.dns
ipv4.dns:                               192.168.100.250
ipv4.dns-search:                        --
ipv4.dns-options:                       ""
ipv4.dns-priority:                      0
```


- You can use a plus (+) or minus (-) sign before a parameter to add or delete values from the parameter. For example:
  - Adding a secondary DNS server:
  - **nmcli con mod "Euler" +ipv4.dns 114.114.114.114**
- After the modification, run the following command to bring up the active connection for the modification to take effect:
  - **nmcli con up "Euler"**
- nmcli also has an interactive connection editor.
- For more information, visit https://wiki.archlinux.org/index.php/NetworkManager.

Routing

- To enable two hosts in different subnets to communicate with each other, a mechanism is required to describe the traffic path between one host and another. This mechanism is called routing. Routing is described using routing entries.
- A routing entry is a pair of predefined addresses: destination and gateway.
- A routing entry allows communication with the destination to be completed through the gateway. Routing entries are listed in a routing table.

- Detailed description of routes: https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-routing

openEuler Route Management and Configuration

- In openEuler, the **route** command is used to view, configure, and manage local routes.
- Routes can also be managed using the **ip** and **nmcli** commands.
- These commands modify the routing table of the system. When the system is started, the routing table is loaded into the memory and maintained by the kernel.

- The **route**, **ip**, and **nmcli** commands can be used to manage routes. The following uses the **route** command as an example.

Using the route Command to View the Routing Table

- Run the **route** command to view the routing table.

```
[root@openEuler ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.110.254 0.0.0.0         UG    100    0        0 enp4s0
192.168.110.0   0.0.0.0         255.255.255.0   U     100    0        0 enp4s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```


- When the **-n** option is used to display routes, the values in the **Destination** column are IP addresses.
- Eight fields are displayed when you run the **route** command to view routes. The possible values of the **Flags** field include:
  - **U** (up) indicates that the route is up.
  - **H** (host) indicates that the gateway is a host.
  - **G** (gateway) indicates that the gateway is a router.
  - **R** (reinstate route) indicates that the route is reinstated for dynamic routing.
  - **D** (dynamically) indicates that the route is dynamically written.
  - **M** (modified) indicates that the route is dynamically modified by the routing daemon or redirect.
  - **!** indicates that the route is closed.

Using the route Command to Add a Route

- Add a temporary route to the network segment or host.

```
route [-f] [-p] [Command [Destination] [mask Netmask] [Gateway] [metric Metric]] [if Interface]]
```

- Example:

```
[root@openEuler ~]# route add -net 192.168.101.0 netmask 255.255.255.0 dev enp4s0
[root@openEuler ~]# route add -host 192.168.100.10 dev enp4s0
[root@openEuler ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp4s0
192.168.100.10  0.0.0.0         255.255.255.255 UH    0      0        0 enp4s0
192.168.101.0   0.0.0.0         255.255.255.0   U     0      0        0 enp4s0
192.168.110.0   0.0.0.0         255.255.255.0   U     100    0        0 enp4s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```


- You can use the **route** command to add routes. The added routes are stored in the memory and become invalid after the system is restarted.
- The **route add -net 192.168.101.0 netmask 255.255.255.0 dev enp3s0** command adds a route to the 192.168.101.0/24 subnet through the enp3s0 device.
- The **route add -host 192.168.101.100 dev enp3s0** command adds a route to the 192.168.101.100 host through the enp3s0 device.
- The output of the **route** command shows that routes to hosts have a higher priority than routes to subnets.

Using the route Command to Delete a Route

- Run the **route del** command to delete a route to a network segment or host.
- Syntax:

```
route del [-net|-host] [netmask Nm] [gw Gw] [[dev] If]
```

- Example:

```
[root@openEuler ~]# route del -host 192.168.100.10 dev enp4s0
[root@openEuler ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp4s0
192.168.101.0   0.0.0.0         255.255.255.0   U     0      0        0 enp4s0
192.168.110.0   0.0.0.0         255.255.255.0   U     100    0        0 enp4s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```


- The **route del -net 192.168.101.0 netmask 255.255.255.0 dev enp3s0** command deletes the route to the 192.168.101.0/24 subnet. To delete a route to a subnet, the subnet and mask parameters are mandatory, while the device parameter is optional.
- The **route del -host 192.168.101.100 dev enp3s0** command deletes the route to the 192.168.101.100 host. The device parameter is optional.
- To delete routes in the route file, use the vi editor to edit the file and restart the network.

Using the nmcli Command to Configure a Static Route

- Run the **nmcli** command to configure a static route for a network connection.

```
# nmcli connection modify enp3s0 +ipv4.routes "192.168.122.0/24 10.10.10.1"
```

- Run the following interactive commands to configure a static route using the editor:

```
# nmcli con edit type ethernet con-name enp3s0
===| nmcli interactive connection editor |===
Adding a new '802-3-ethernet' connection
Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.
You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-1x, ipv4, ipv6, dcb
nmcli> set ipv4.routes 192.168.122.0/24 10.10.10.1
nmcli>
nmcli> save persistent
Saving the connection with 'autoconnect=yes'. That might result in an immediate activation of the
connection.
Do you still want to save? [yes] yes
Connection 'enp3s0' (1464ddb4-102a-4e79-874a-0a42e15cc3c0) successfully saved.
nmcli> quit
```

Host Name

- A host name uniquely identifies a device in a LAN.
- The device can be a physical or a virtual machine.
- The host name is stored in the **/etc/hostname** file.

```
[root@openEuler ~]# cat /etc/hostname
openEuler
```

Setting the Host Name

- Run the **hostname** *new-name* command to temporarily set a host name (valid until next restart).
- Setting a permanent host name: **hostnamectl set-hostname** *new-name*
- Setting a host name by modifying the **hostname** file: write *new-name* to the **/etc/hostname** file.

```
[root@openEuler ~]# hostname
openEuler
[root@openEuler ~]# hostname huawei
[root@openEuler ~]# hostname
huawei
[root@openEuler ~]# hostnamectl set-hostname openEuler01
[root@openEuler ~]# hostname
openEuler01
[root@openEuler ~]# echo "HCIA-openEuler" > /etc/hostname
[root@openEuler ~]# hostname
openEuler01
```


- To make the setting take effect, log in again or run the **source .bashrc** command.
- Run the **hostname** command to view the host name of the current system.

hosts File

- Hosts in a LAN can be accessed through IP addresses.
- When there are a large number of hosts in the LAN, IP addresses are difficult to remember. Therefore, we want to access hosts directly through their host names.
- To do this, we can find the hosts we need in a table that records mappings between host names and IP addresses. This table is **hosts**.

```
[root@openEuler ~]# cat /etc/hosts
127.0.0.1   localhost       localhost.localdomain     localhost4        localhost4.localdomain4
::1         localhost       localhost.localdomain     localhost6        localhost4.localdomain6
```


- The **hosts** file is a system file without a file name extension. Its basic function is to establish a "database" of frequently used domain names and their corresponding IP addresses.
- When a user enters a website URL in the web browser, the system searches for the corresponding IP address in the **hosts** file. Once the IP address is found, the system opens the corresponding web page.
- If the URL is not found, the system sends the URL to the DNS server for IP address resolution.
- Run the **cat /etc/hosts** command to view the **hosts** file.

Modifying the hosts File

- You can use an editor to modify the **hosts** file in the following format:

```
ip<TAB>domain.com
192.168.10.20    www.example.com
```

- To delete an entry, add **#** to comment it out. For example:

```
#ip<TAB>domain.com
#192.168.10.20    www.example.com
```

Hosts and DNS

- As the number of hosts on a network increases, it becomes difficult for a single **hosts** file to carry a large number of mappings.
- If the IP address mapped to a domain name cannot be found in the **hosts** file, the host submits the domain name to the DNS server, after which the DNS server returns the IP address to the host. This process is called domain name resolution.
- The DNS server is like a public **hosts** file or distributed database.

Querying a DNS Record (1)

- In openEuler, run the **nslookup** command to query records on the DNS server.
- You can run the **nslookup** command to check whether domain name resolution is normal and diagnose network problems.
- The format of the **nslookup** command is as follows:

```
nslookup domain [dns-server]
In the preceding command, domain indicates the domain name to be queried.
[dns-server] specifies the DNS server. This parameter is optional. Common values are 8.8.8.8 and 114.114.114.114.
```


- nslookup is not installed in the openEuler OS. Run the following command to install it:
- **yum install -y bind-utils**

Querying a DNS Record (2)

- For example, to query the IP address mapped to **exam.openEuler.com**, run the following command:

```
[root@openEuler ~]# nslookup exam.openEuler.com
Server:         192.168.100.250
Address:        192.168.100.250#53

Name:   exam.openEuler.com
Address: 192.168.100.250
```


- dig 命令能够实现类似功能，详见：https://www.linuxcool.com/dig

DNS Resolution Records

- In addition to resolving a domain name to an IP address, the DNS server also supports other types of resolution records.

Record	Description
A	Maps a domain name to an IPv4 address.
CNAME	Maps one domain name to another domain name, to achieve the same domain access using both names.
MX	Sets up an email service that points to the email server address.
NS	Specifies a DNS server to resolve a subdomain name.
TXT	This parameter can be set to any value or left blank. Generally, it is used for verification.
AAAA	Maps a host name (or a domain name) to an IPv6 address.
SRV	Identifies computers hosting specific services.
SOA	Indicates the start of authority. This record includes administrative information about its zone, including the master server among multiple NS records.
PTR	Indicates the reverse of the A record, resolving an IP address to a domain name.
Explicit/Implicit URL forwarding	Maps a domain name to an HTTP/HTTPS address. When a user accesses the domain name, the user is automatically redirected to the target address. The real address is displayed in explicit forwarding and hidden in implicit forwarding.

- The **dig** command can implement similar functions. For details, see https://phoenixnap.com/kb/linux-dig-command-examples.

Querying Other Records

- Run the **nslookup** command to query various types of records. For example:

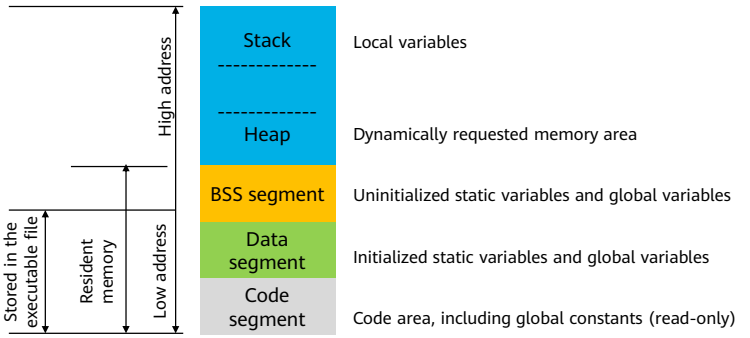
```
[root@openEuler ~]# nslookup
> set type=A
> exam.openEuler.com
Server:         192.168.100.250
Address:        192.168.100.250#53

Name:   exam.openEuler.com
Address: 192.168.100.250
```


- The command is used for interactive query.

Contents

Introduction to Processes

- A process is an entity of a running program in a computer. It is a specific implementation of that program.
- When a Linux process is created, the system allocates a segment of memory space (a certain logical address space) to the process.

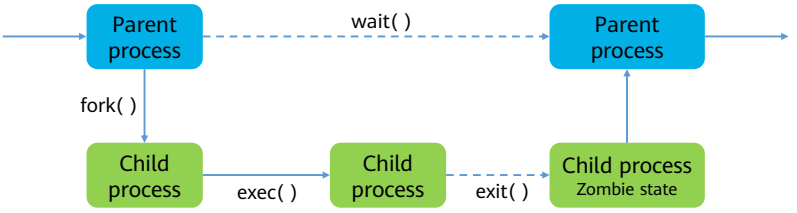


HUAWEI

- Each program can see a complete and continuous address space, which is not directly associated with the physical memory. Instead, the OS provides an abstract concept of the memory so that each process has a continuous and complete address space. During the running of a program, the virtual address is mapped into the physical address. We also know that the address space of a process is segmented, and there are so-called data segments, code segments, BBS segments, heaps, stacks, and so on.

Process Life Cycle

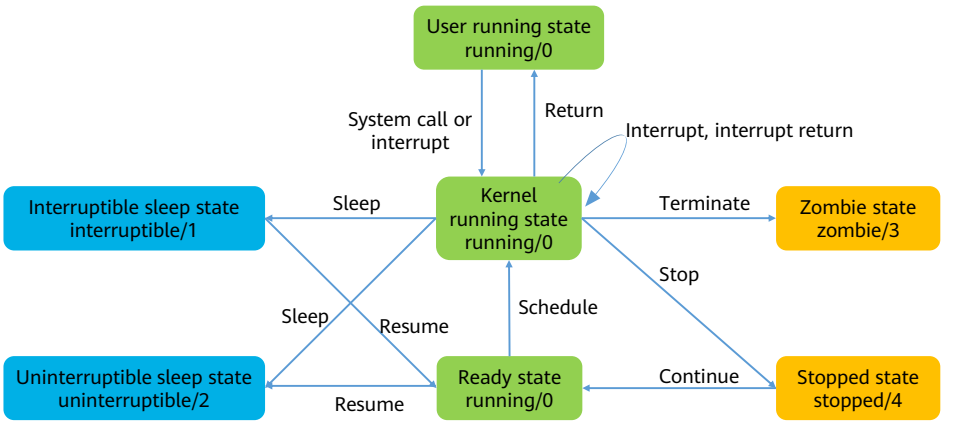
- Each process has a unique process ID (PID), which is used for tracing the process.
- Any process can create a child process by copying its own address space (fork). The parent process ID (PPID) is recorded in the child process.
- The first system process is systemd, and all the other processes are its descendants.



HUAWEI

- Each process has its own lifecycle, covering creation, execution, termination, and deletion. During system running, these phases are repeatedly executed for thousands of times.
- A process (parent process) can use the fork() system call to create a process (child process). fork() creates a process descriptor for the child process, sets a new process ID, copies the process descriptors of the parent process to the child process. The address space of the parent process will not be copied. Instead, the parent and child processes share the same address space.
- The exec() system call copies the new program to the address space of the child process. Because the same address space is shared, writing data to the child process will cause a page error exception. In this case, the kernel allocates a new physical page to the child process.
- This delayed operation is called copy-on-write (COW). A child process and a parent process usually execute their own programs. This operation avoids unnecessary overhead because copying the entire address space is slow and inefficient and consumes a lot of processor time and resources.
- When the program execution is complete, the child process is terminated by the exit() system call. exit() releases most of the data structures of the process and notifies the parent process of the termination message. Then the child process becomes a zombie process.
- The child process will not be completely cleared until the parent process knows that the child process is terminated through the wait() system call. Once the parent process knows that the child process is terminated, it clears all the data structures and process descriptors of the child process.

Process State



HUAWEI

- The running state is the combination of the running state and the ready state, indicating that a process is running or ready to run. Linux uses the TASK_RUNNING macro to indicate this state.
- The interruptible sleep state indicates a process is sleeping (blocked) and will be resumed when there are resources. The process can also be resumed by other process signals or clock interrupts and enters the run queue. Linux uses the TASK_INTERRUPTIBLE macro to indicate this state.
- The uninterruptible sleep state is similar to the interruptible sleep state, but the process cannot be resumed by other process signals or clock interrupts. Linux uses the TASK_UNINTERRUPTIBLE macro to indicate this state.
- The stopped state indicates process is stopped to accept certain processing. For example, a process being debugged is in this state. Linux uses the TASK_STOPPED macro to indicate this state.
- The zombie state indicates a process is ended but the process control block is not released. Linux uses the TASK_ZOMBIE macro to indicate this state.

Process Priority (1)

- A process's CPU resource (time slice) allocation refers to the priority of that process.
- Processes with a higher priority will be executed first.
- Configuring the priority of a process can improve system performance in a multitasking Linux environment.

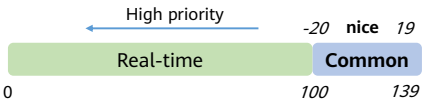
```
[root@openEuler ~]# ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  686110  686109  0  80   0 -  3383 do_wai pts/0    00:00:00 bash
0 S     0  686217  686110  0  80   0 -  3386 do_wai pts/0    00:00:00 bash
0 R     0  688767  686217  0  80   0 -  3399 -      pts/0    00:00:00 ps
```


- Run the **ps** command to view processes.
- The CFS scheduler is used to schedule processes in openEuler.
  - UID indicates the user ID. PID indicates the process ID.
  - PPID indicates the parent process ID.
  - PRI indicates the process priority (a smaller value indicates a higher priority).
  - NI Indicates the nice value of a process.
- The process priority is determined by the dynamic priority (PRI) and static priority (PR). It is a number that determines the execution sequence of a process in the CPU. A process with a higher priority has a higher probability of being executed by the processor.
- Depending on the behavior of a process, the kernel uses a heuristic algorithm to determine whether to enable or disable dynamic priority (PRI). The static priority (PR) of a process can be directly modified by the nice value. A process with a higher static priority (PR) obtains more time slices (the time slice is the execution time of the process in the processor).
- The nice ranges from **-20** (highest priority value) to **19** (lowest priority value) and the default is **0**. Only the **root** user can change the nice value of a process to a negative value (to make the process have a higher priority).

Process Priority (2)

- PRI: process priority, indicating the sequence in which a program will be executed by the CPU. A smaller value indicates higher process priority.
- NI: nice value, indicating a corrected process priority value. It can be understood as "concession".
- The nice value of a process is not the priority of that process. You can, however, adjust the nice value to affect the priority value.

Priority (PRI) Range	Description
0—99	Real-time process
100—139	Non-real-time process



HUAWEI

- The smaller the PRI value, the faster the execution. After the nice value is added, the PRI changes according to the following formula: PRI(new)=PRI(old)+nice.
- If the nice value is negative, the PRI value of a program decreases. That is, the priority of the program increases, and the program is executed faster.
- The kernel uses a simple value range from 0 to 139 to indicate the internal priority. A smaller value indicates a higher priority. The value range from 0 to 99 is for real-time processes. The nice value [-20,19] is mapped to the value range from 100 to 139.

Adjusting the Priority of a Process

- In openEuler, **nice** and **renice** are used to adjust the nice value of a process, which affects the process priority.
  - The syntax of the **nice** command is as follows (the **adjustment** value range is -19 to 20):

    ```
    nice [-n adjustment] [command [arg...]]
    ```

  - To set vi priority to **-18**, for example, run the following command:

    ```
    nice -n -18 vi &
    ```

  - The **adjustment** range of the **renice** command is the same as for **nice**. Objects include the program group **-g**, process **-p**, and user **-u**. The syntax is as follows:

    ```
    renice [-n adjustment] [-] <pid>
    ```

  - To set vi priority to -**12**, for example, run the following command:

    ```
    # renice -n 12 -p 9700
    9700(process ID) old priority -18, new priority 12
    ```


- Run the **nice –help** command to view the **nice** description.
- Run the **renice –h** command to view the **renice** description.
- You can also run the **top** command to change the **NI** value of a process.
  - Enter **top**, press **r**, enter the PID of the process, and enter the nice value.

Foreground and Background Processes

- Background processes barely interact with users, so they do not need high priority.
  - A daemon process of Linux is a special background process that is independent of the terminal. It periodically executes tasks, or waits to resume executing tasks.
- Foreground processes interact with users, requiring a high response speed and high priority.
  - A foreground process is a process used by a user to control a terminal.

- A daemon is independent of control from all terminals.
- Most Linux servers are implemented using daemons, for example, the Internet server inetd and the web server httpd. In addition, daemons typically perform system tasks, for example, the job planning process crond and the printing process lpd.
- A daemon process typically performs system services. You can use crontab to submit, edit, or delete tasks.
- Most Linux service processes are implemented through the daemon processes, for example, process 0 (scheduling process) and process 1 (init process). A daemon runs after a machine is started and stops after the machine is shut down.

Controlling Foreground and Background Processes

- The following are common commands used in openEuler to manage processes, including starting, stopping, and switching between foreground and background:
  - **&**: Place an ampersand (&) at the end of a command to execute it in the background.
  - **Ctrl+z**: Moves a running foreground process to the background and suspends it.
  - **Ctrl+c**: Interrupts a running process.
  - **jobs**: Lists processes running in the background.
  - **fg**: Moves a background process to the foreground.
  - **bg**: Resumes a background process.

Viewing Processes

- When troubleshooting the system, it is important to know the status of processes.
- In openEuler, run the **ps** or **top** command to view the details of different processes.

Using the ps and top Commands to View Processes

- Run the **ps** command to list currently running processes and their accompanying details. The table below describes the fields.
- Run the **ps aux** or **ps -ef** command to view process information.
- The **ps** command lists processes at a specific point in time. The **top** command shows processes in real time.

Field	Description
UID	User identifier
PID	Unique process identifier
%CPU	Process CPU usage
%MEM	Process memory usage
NI	Nice value (process concession)
PRI	Process priority
TTY	Terminal associated with the process
STAT	Process state

- Run the **ps aux** or **ps –ef** command to query process information.
- **ps** statically displays process information, while **top** dynamically displays process information.

Using Signals to Manage Processes

- In openEuler, processes communicate with each other through signals. The table below lists common signals.
- A process signal is a predefined message that a process recognizes and chooses to act upon or ignore.
- The system administrator needs to know what kind of signal to send to a process, and when.

Signal	Field	Description
1	HUP	Hangup
2	INT	Terminal interrupt
3	QUIT	Terminal quit
9	KILL	Unconditional termination
11	SEGV	Segment error
15	TERM	Termination
17	STOP	Stop executing unconditionally (but do not terminate)
18	TSTP	Terminal stop (but continue to run in the background)
19	CONT	Continue executing, if stopped (STOP or TSTP)

Managing Processes

- In openEuler, the **kill** and **killall** commands send signals to processes.
- To send signals to a process, the current user must be the owner of that process, or the **root** user.
- **kill** *PID* can send a signal to a process. The TERM signal is sent by default. Use the **–s** option to specify other signals. For example:

```
# kill 3389
# kill -s HUP 3389
```

- **killall** *process_name* can send a signal to a process matching the specified process name. Supporting wildcard characters, **killall** can also send a signal to multiple processes. The following command, for example, will send a signal to all processes starting with 'python'

```
# killall python*
```

Quiz

1. (Single-answer) Which of the following commands can dynamically check host resource usage?
   A. ps
   B. top
   C. free
   D. lscpu
2. (True or false) The **route** command can view or configure host route information. Configured route information will remain valid permanently.
   A. True
   B. False

- Answer:
  - B
  - B

Summary

This chapter described common management operations in openEuler, including task, network, and process management.

- Task management: how to schedule tasks in the system.
- Network management: basic network knowledge and how to manage network elements in the system.
- Process management: how to view and manage system processes.

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

Shell Scripts

HUAWEI

Foreword

- This course is based on openEuler and includes:
- Basic shell knowledge
- Shell script compilation best practices based on actual cases

Objectives

- Upon completion of this course, you will be familiar with:
  - Basic shell knowledge
  - Shell programming basics
  - Compiling common shell scripts

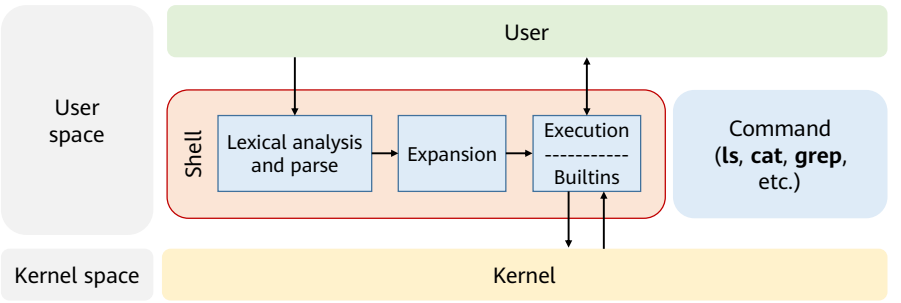
Contents

Overview and Objectives

- Know the role of shell in openEuler.
- Master basic shell script operations.

Shell Overview

- The shell is the user interface (UI) provided by the operating system for users to interact with the kernel.
- The shell sends commands entered by users to the kernel for execution, and returns the execution result.
- The shell is programmable, allowing users to write programs which include shell commands.



Shell Development History

- 1971: Ken Thompson of Bell Labs develops the first Unix shell, the V6 shell. It was an independent user program executed outside the kernel.
- 1977: Steven Bourne creates the Bourne shell for V7 in AT&T Bell Labs.
- 1978: Bill Joy develops the C shell (csh) for the BSD during his postgraduate studies at University of California, Berkeley.
- 1983: Ken Greer develops tcsh at Carnegie Mellon University by introducing some functions of the Tenex system, such as command line editing and automatic file name and command completion, to the C shell.
- 1983: David Korn creates the KornShell (ksh) at AT&T Bell Labs. It provided more powerful functions as well as associative array expression calculation.
- 1989: Brian Fox develops the Bourne again shell (Bash) for the open source GNU project, as a free software replacement for the Bourne shell. Bash has since then become the most popular shell in the world. Compatible with sh, csh, and ksh, it is the default shell of the Linux system.

In Linux, multiple shell programs are available.
For example, dash, csh, and zsh.
You can view the default shell in **/bin/sh** and modify it in **/etc/passwd**.

Viewing the Shell

- Log in to openEuler and check the default shell installed in the system.

  ```
  cat /etc/shells
  ```

- Check the default shell of the current login user.

  ```
  echo $SHELL
  ```

- View the current shell.

  ```
  echo $0
  ```


- Command for reference: **# cat /etc/shells**
- Command for reference: **# echo $SHELL**
- Command for reference: **# echo $0**

Contents

Basic Shell Script Knowledge

- In Unix/Linux, a program/command performs only one operation.
- Combining multiple commands can solve complex problems.
- The most simple shell script is an executable file made up of a series of commands. These commands can be reused in other scripts.

> Writing a shell script with a good style, which is easily readable, improves the automation and accuracy of routine tasks.

Shell Script Constraints

- There is no "silver bullet."
  - Shell scripts can complete many tasks, but they are not applicable to every case.
  - Shell scripts are a good choice for tasks that can be done by invoking other command-line utilities.
  - Shell scripts can be used as a glue language to integrate other programming languages.

> If a shell script solution to a problem is complex and/or inefficient, you can consider using other programming languages.

- Silver bullet: No tool is going to solve all your problems.

Shell Script Development Environment

- To create a shell script, you can open a new file in any text editor.
- Advanced editors such as Vim and Emacs can highlight, check, and supplement syntax after identifying .sh files.

```
[root@openEuler ~]# vim demo.sh # Create a script file and write the following content into the file:
#!/bin/bash
echo "Hello World"
[root@openEuler ~]# sh demo.sh
Hello World
```

Specifying an Interpreter for Shell Scripts

- The shell script is static code. To output the result, an interpreter is required.
- Generally, the interpreter that executes a script is specified in the first line of the script.
- If no interpreter is specified, the script can run properly in the default interpreter. For clearer specifications and better security, however, it is recommended to specify an interpreter as in the two examples below:

```
#!/bin/bash

#!/bin/csh
```

  - What is the difference between these two paths?

Running Shell Scripts (1)

- There are two execution modes for script files:
  - **sh script_name.sh**
  - **./script_name.sh**
- In Linux, everything is a file (an object, with permission attributes). A script is a file, as is a command or a program.
- When executing scripts sent by others or downloaded from the Internet, you may encounter permission problems. To solve such problems, you can grant the execute permission.
  - **chmod +x script_name.sh**

Running Shell Scripts (2)

- If a shell script is executable, you can invoke it by entering its name in the command line.
  - To successfully invoke a script, its path must be contained in the **$PATH** variable.
    - Check the **$PATH** variable.
    - Modify the **$PATH** variable.
    - Search for the path of a command or a script.

```
[root@openEuler ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@openEuler ~]# PATH=$PATH:/New/path
[root@openEuler ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/New/path
```


- To change **$PATH** permanently, add **export PATH="$PATH:/new/path"** to **.bashrc** or **.bash_profile**.
- For details about variables, see the following sections.

Running Shell Scripts in the Background

- The execution of some scripts takes a long time, occupying the CLI. You can run such scripts in the background.
  - **./my_script.sh &**
- Using this way, the script process will stop after you exit the shell. To ensure that the script keeps running, run the following command:
  - **nohup ./my_script.sh &**
  - Using this command, the script's standard output and standard error streams will be redirected to the **nohup.out** file.
- To view processes running in the background, you can run the **jobs** command.

Contents

Overview and Objectives

- Understand the input, output, and redirection of text streams in Linux.
- Master the usage of pipes in Linux.
- Master shell variables and operators.
- Master shell logic judgment, selection structure, and loop structure.

Text Stream in Linux

- The text stream exists in every Linux process.
- When a Linux process is started, three text stream interfaces are enabled: standard input, standard output, and standard error.
- These three interfaces correspond to a program's input, output, and exception throwing.

```
[root@openEuler ~]# date
Wed Jul 29 10:39:17 CST 2020
[root@openEuler ~]# datee
bash: datee: command not found
```

Example:
After a string of characters is entered in bash, the standard input interface in the bash process captures the input in the command line. After being processed, the data is output from the standard output interface and displayed on the screen. If an exception occurs during the processing, the exception is displayed on the screen through the standard error interface.

Output Redirection

- Sometimes, you need to save the output of a program. You can do this by saving it to a file through redirection.
  1. Direct the standard output to a file:
     - **ls > dir_log**
     - Using this command, the standard output of the program will overwrite the file's previous contents.
  2. Append the standard output to a file:
     - **ls >> dir_log**
     - Using this command, the standard output of the program will be appended to the end of the file.
  3. After the command executes, you can run the **cat dir_log** command to view the contents of the saved file.

Input Redirection

- Input redirection, similar to output redirection, redirects a program's standard input.
- Input redirection:
  - Format: *command* < *inputfile*
  - Use the file in the gray box as the standard input, then pass the command as in the format above.
  - Example: **wc -l < /dev/null**
- Inline input redirection:
  - Format: *command* << *maker*
  - Output redirection requires files, but inline input redirection can use instant input text as standard input.
  - *maker* on the right side of the format indicates the standard input's start and end. It is not included in the standard input.
  - In the code snippet in the gray box, **EOF** is the maker.

```
[root@openEuler ~]# less << EOF
> item 1
> item 2
> item 3
> EOF
item 1
item 2
item 3
(END)
```


- **command << maker >file** will save the input content as a file.

Pipe

- Sometimes, you need to connect the output of one command to the input of another command. This may be done with redirection, but is complex.
- A pipe (|) can connect the input and output of two commands, or even connect multiple commands in series.
- Format: *command1* **|** *command2* **|** *command3*

```
[root@openEuler ~] ls /bin/ | grep python | less
```

- **Pipes are actually a way of inter-process communication (IPC).**

Contents

Characters in the Shell

- Like other programming languages, the shell also has some special characters. When writing scripts, pay attention to these special characters.

Special Character	Description
#	Remarks
'	String reference
\	Escape
/	Path separator
!	Reverse logic

- Comments [Hashtag
  - A shebang in the very first line of a script consists of a hashtag and an exclamation mark (#!), for example, **#! /bin/bash**.
  - Lines beginning with a hashtag (#) are comments and will not be executed.
  - However, when a hashtag (#) is enclosed by single or double quotes, it is used as the number sign (#) and does not have the comment function.
- Full quoting [Single quote]
  - A pair of single quotes (') identify a single character string. Enclosing characters in single quotes (') preserves the literal value of each character within the quotes. No character in the single quotes has special meaning. Single quotes must appear in pairs.
- Escape [Backslash]
  - Escape character (\) before a special character: An escape character (\) is used to remove the special meaning from a single character. It indicates the special character itself and is commonly used in character strings.
  - Escape character at the end of a line of instruction: An escape character (\), when used at the end of a line, will direct you to the next line. The followed carriage return is invalid (that is, the carriage return is escaped), and the input of a new line is still a part of the current instruction.
- Path separator [Forward slash]
  - Path separator: A forward slash (/) is used to represent the root directory, while a path starting with a forward slash (/) indicates the path is read from the root directory.
  - Arithmetic operator: A forward slash (/) denotes division. Example: a=4/2
- Reverse (or negate) [bang],[exclamation mark]
  - Reverse (or negate) the sense of a test or exit status.
  - Change the sense of equal (=) to not-equal (!=).
  - Negation. For example, **ls a[!0-9]** lists files whose names start with an **a** that is followed by a non-digit character.

Variables

- Any language has variables.
- The shell is quite different from other strongly typed programming languages such as C, Java, and C++. Shell variables are typeless.
- A variable can be used to reference the value of a memory area. The variable name is the label attached to the memory area.
- Variable value assignment: *variable*=*value*

```
[root@openEuler ~]# a='Hello World'
[root@openEuler ~]# echo a
a
[root@openEuler ~]# echo $a
Hello World
```

Variable Types

- In the Linux shell, there are two types of variables:
  - Environment variables
  - User-defined variables
- Variables of each type are classified into either global variables or local variables according to their scope.
  - Global variables apply to the entire shell session and its subshells.
  - Local variables act in the process that defines them and its subprocesses.
- Viewing variables
  - Run the **printenv** command to view global variables.
  - Run the **set** command to view all variables in a specific process, including local variables, global variables, and user-defined variables.
- Modifying variables
  - Add the export statement to the **.bash_profile** or **.bashrc** file to permanently modify variables.

- Compare the difference between **printenv** and **set**: **diff -yw <(printenv) <(set)**
- The **set** command is a built-in shell command that displays both shell and user-defined environment variables, regardless of whether the variables are exported. The **set** command allows you to change the value of a shell option and set positional parameters, or display the name and value of a shell variable. Without any options or parameters, the **set** command lists the names and values of all shell variables and functions (sorted by the current language environment), and the output format can be used as the input for setting or resetting the current variables.
- **printenv** and **env** are similar in terms of environment variable printing. In terms of functions, **env** is used to set environment variables and run specified commands, while **printenv** is used to print environment variables.
- The **set** command is a built-in shell command that is used to set the attributes of the shell.

Using Variables

- Shell variable naming rules:
  - A variable name consists of digits, letters, and underscores (_).
  - It must start with a letter or underscore (_).
  - Keywords in the shell cannot be used.

```
[root@openEuler ~]# help | grep for
 break [n]                                  printf [-v var] format [arguments]
 for NAME [in WORDS ... ] ; do COMMANDS; done   unset [-f] [-v] [-n] [name ...]
 for (( exp1; exp2; exp3 )); do COMMANDS; done  until COMMANDS; do COMMANDS; done
```

- Format of a variable:
  - variable=value
  - variable='value'
  - variable="value"

➢ What is the difference between a single quotation mark (') and a double quotation mark (") in a shell script?

- No space is allowed on the left and right of the assignment operator.
- If necessary, you can also use the **declare** keyword to explicitly define the type of a variable.
- To query keywords, run the **help | grep <keywords>** command.
- If the value contains whitespace characters, use quotation marks to enclose the value.
  - Content enclosed in single quotation marks: The shell does not expand commands and variables, that is, the original output is displayed.
  - Double quotation marks: The shell expands the commands and variables enclosed in them.

Expanding Variables

- In the following example, if braces are not used, bash interprets **$FIRST_$LAST** as the variable **$FIRST**_ followed by the variable **$LAST** instead of the variables **$FIRST** and **$LAST** separated by an underscore (_).
- To make variable expansion work correctly, you must enclose the variable in braces.

```
[root@openEuler ~]# FIRST_=Jane
[root@openEuler ~]# FIRST=john
[root@openEuler ~]# LAST=Doe
[root@openEuler ~]# echo $FIRST_$LAST
Doe
[root@openEuler ~]# echo ${FIRST}_$LAST
john_Doe
```

Value Assignment and Variable Output

- Define and assign values to the following variables: **name** and **time**.
- Output the **name** and **time** variables.
- Read the input and output the variables.

```
name=Euler
time='2020202'
echo "My name is $name, today is $time"
read name
echo "Hello, $name, welcome!"
```


- The **read** command waits for user input and assigns the input content to a variable.

Arithmetic Extensions of the Shell

- Arithmetic extensions can be used to perform simple arithmetic operations of integers.
- Syntax: $[expression]
- Example:

```
[root@openEuler ~]# echo $[1+1]
2
[root@openEuler ~]# echo $[2*2]
4
[root@openEuler ~]# COUNT=1; echo $[$[$COUNT+1]*2]
4
```

Calculating Time in the Shell

- Define the 24-hour, 60-minute, and 60-second variables.
- Define the number of seconds per day and assign a value to the variable.
- Output the variables.

```
[root@openEuler ~]# SEC_PER_MIN=60
[root@openEuler ~]# MIN_PER_HR=60
[root@openEuler ~]# HR_PER_DAY=24
[root@openEuler ~]# SEC_PER_DAY=$[ $SEC_PER_MIN * $MIN_PER_HR * $HR_PER_DAY
]
[root@openEuler ~]# echo "There are $SEC_PER_DAY seconds in a day"
There are 86400 seconds in a day
```

Common Expressions for Arithmetic Operations

Operator	Function
<VARIABLE>++	Post-increment
<VARIABLE>--	Post-decrement
++<VARIABLE>	Pre-increment
--<VARIABLE>	Pre-decrement
-	Unary subtraction
+	Unary addition
**	Exponentiation
*	Multiplication
/	Division
%	Remainder
+	Addition
-	Subtraction

Arithmetic Operation Priorities

- The priorities, in descending order, are as follows:

Operator	Function
<VARIABLE>++, <VARIABLE>--	Post-increment and post-decrement
++<VARIABLE>, --<VARIABLE>	Pre-increment and pre-decrement
-, +	Unary subtraction and addition
**	Exponentiation
*, /, %	Multiplication, division, and remainder
+, -	Addition and subtraction

Difference Between Pre-increment and Post-increment Variables

- Is the output of **echo $[++i]** the same as that of **echo $[i++]**?
- What are the precautions for triggering boundary conditions?

```
[root@openEuler ~]# i=3
[root@openEuler ~]# echo $[i]
3
[root@openEuler ~]# echo $[++i]
4
[root@openEuler ~]# echo $[i++]
4
[root@openEuler ~]# echo $i
5
```


- **i++** creates a variable to save the value of **i** and returns the value, and then increments the value of **i** by 1. **++i** directly adds 1 to the space of the **i** variable and returns the value in the space. No temporary space is created, and the performance is higher.
- The variable returned in the code snippet is **i**. That's the difference.

Contents

Structured Commands in the Shell

- In addition to sequential execution, shell scripts require some extra logic control processes.
- Similar to other programming languages, structured commands in shell scripts include conditions and loops.

Conditional Statements

- if-then statement
- Syntax:

```
if command
   then
      commands
fi
```

- The bash shell executes the command following **if** first. If the exit status code is **0**, the bash shell continues to execute the commands in **then**. Otherwise, the bash shell executes the next command in the script.

- The **if** judgment conditions in the shell are different from other **if** judgment conditions.

Multi-branch Judgment Statements

- When a multi-condition judgment is required, there may be a complex **if-then-else** statement, **elif**, which frequently checks the value of a variable. In this scenario, the **case** statement may be a more suitable solution.
- Syntax:

```
case variable in
    pattern1 | pattern2) commands1;;
    pattern3) commands2;;
    *) default commands;;
esac
```


- The **case** statement compares the specified variable with different patterns. If they match, the corresponding command is executed.
- Multiple patterns can be separated by vertical bars (|).
- The asterisk (*) indicates all values that do not match the known patterns.

Loop Statements

- Shell scripts often contain some repeated tasks, which are the same as cyclically executing a group of commands until a specific condition is met.
- There are three common loop statements: **for**, **while**, and **until**.
- There are two types of loop control characters: **break** and **continue**. They are used to control the direction of the loop process.

For Loop

- Syntax (shell style)

```
for var in list
   do
      commands
   done
```

Example:

```
for i in {1..10}
do
        printf "$i\n"
done
```

- Syntax (C language style)

```
for ((var assignment ; condition ; iteration process))
do
        commands
done
```

Example:

```
for (( i = 1; i < 10 ; i++))
do
   echo "Hello"
done
```


HUAWEI

- The commands between **do** and **done** is called a loop body. The number of execution times is the same as the number of constants or character strings in the list. The **for** loop assigns the first constant or character string in the list after **in** to the loop variable, executes the loop body to traverse the list, and executes the command sequence after the **done** command.

List in For Loop

```
[root@openeuler ~]# for HOST in host1 host2 host3; do echo $HOST; done
host1
host2
host3
[root@openeuler ~]# for HOST in host{1..3}; do echo $HOST; done
host1
host2
host3
[root@openeuler ~]# for EVEN in $(seq 2 2 8); do echo "$EVEN"; done;
2
4
6
8
```

Odd Accumulator in For Loop

- Define the odd accumulator file and execute it.
- Reference script:

```
#!/bin/bash

sum=0
for i in {1..100..2}
do
    let "sum+=i"
done
echo "sum=$sum"
```

File Display in For Loop

- Display all files in the directory.
- Reference script:

```
#!/bin/bash

for file in $(ls)
do
   echo "file: $file"
done
```

While Loop

- The **while** loop is a pre-test loop. It is used to perform some repetitive tasks that may be executed one or more times according to specific conditions.
- To avoid infinite loops, ensure that the loop body contains the exit condition.

```
#!/bin/bash

sum=0;i=1
while(( i <= 100 ))
do
    let "sum+=i"
    let "i += 2"
done
     echo "sum=$sum"
```


HUAWEI

- The example here is equivalent to the previous odd accumulator. Question: What is the difference between the **while** and **for** loops?

Until Loop

- The until loop is similar to the while loop, but the conditions for exiting the loop are different.

```
#!/bin/bash
# odd accumulator
sum=0;i=1
until(( i > 100 ))
do
    let "sum+=i"
    let "i += 2"
done
    echo "sum=$sum"
```


- The example script here is equivalent to the previous odd accumulator.
- Question: What are the differences between the **until** loop and the **for** and **while** loops?

Printing a Multiplication Table Using a Loop

- Reference script:

```
#!/bin/bash

for (( i = 1; i <=9; i++ ))do
    for (( j=1; j <= i; j++ ))do
    let "temp = i * j"
    echo -n "$i*$j=$temp  "
    done
printf "\n"
done
```

```
[root@openEuler ~]# sh demo.sh
1*1=1
2*1=2  2*2=4
3*1=3  3*2=6  3*3=9
4*1=4  4*2=8  4*3=12  4*4=16
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25
6*1=6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```


- Try to use the **while** or **until** loop.

Section Summary

- This section described the elements involved in shell script programming, including characters, variables, and logic control structures.
- It demonstrated, with examples, how to compile common scripts.

Contents

Overview and Objectives

- Master bash debugging.
- Master basic shell script debugging methods.
- Understand the importance of a good style when compiling shell scripts.
- Master common script compilation methods.

Troubleshooting Shell Script Errors

- Administrators who write, use, or maintain shell scripts inevitably encounter script errors.
  - Errors are usually caused by input errors, syntax errors, or incorrect script logic.
  - When writing scripts, using a text editor in conjunction with the bash syntax highlighter helps make errors more obvious.
  - The most direct way to find and correct errors in a script is to debug.
  - Another simple way to avoid introducing errors into a script is to follow a good style during script creation.

Debugging Mode (1)

- To activate the debugging mode on a script, add the **-x** option to the command interpreter in the first line of the script.
- Example:

```
#!/bin/bash -x
```

```
[root@euler ~]# cat adder.sh
#!/bin/bash -x
```

Debugging Mode (2)

- The debugging mode of bash prints the commands executed by the script before the script is executed.
- The results of all shell extensions that have been executed are displayed in the print output. The state of all variable data is printed in real time for tracing.

```
[root@openEuler tmp]# bash -x adder.sh
```

Evaluation Exit Code (1)

- Every command returns an exit state, also known as a return state or an exit code.
- The exit code can be used for script debugging.
- The following examples show the execution and exit state retrieval of several common commands.

```
[root@openEuler tmp]# ls /etc/hosts
/etc/hosts
[root@openEuler tmp]# echo $?
0
----------------------------------------------------------------
[root@openEuler tmp]# ls /etc/nofile
ls: cannot access /etc/nofile: No such file or directory
[root@openEuler tmp]# echo $?
2
```

Evaluation Exit Code (2)

- Use an exit code in a script.
- A script, once executed, will exit after it has processed everything. It may sometimes exit earlier, for example after encountering an error condition.
- Exiting earlier can also be achieved using the **exit** command. When a script encounters the **exit** command, it exits immediately, skipping the remaining content.

```
[root@openEuler tmp]# cat hello.sh
#!/bin/bash
echo "Hello World"
exit 1
[root@openEuler tmp]# ./hello.sh
Hello World
[root@openEuler tmp]# echo $?
1
```

Contents

Good Style (1)

- The following are some specific practices which can be followed:
  - Divide a long command into multiple lines of smaller code blocks. Shorter code segments are easier for readers to understand.
  - Arrange the beginning and end of multiple statements so you can see where the control structures start and end and whether they are closed correctly.
  - Indent lines containing multiple statement lines to show code logic hierarchy and control structure flow.
  - Use line spacing to separate command blocks, to clarify the start and end of code segments.
  - Use the same format throughout a script.

Good Style (2)

- Adding comments and spaces can greatly improve script readability.
- These simple methods make it much easier for users to find errors during compilation and improve the readability of scripts for future readers.
- Example of code with poor readability:

```
#!/bin/bash

for PACKAGE in $(rpm -qa | grep kernel); do echo "$PACKAGE was installed on $(date -d
@$(rpm -q --qf "%{INSTALLTIME}\n" $PACKAGE))"; done
```

Good Style (3)

- Script after modification:

```
#!/bin/bash
# This script is used to read kernel-related software package information and query the software package installation time from the RPM database.
PACKAGETYPE=kernel
PACKAGES=$(rpm -qa | grep $PACKAGETYPE)
# Loop information
for PACKAGE in $PACKAGES; do
    # Query the installation time of each software package.
    INSTALLEPOCH=$(rpm -q --qf "%{INSTALLTIME}\n" $PACKAGE)
    # Convert the time to a common date and time.
    INSTALLDATETIME=$(date -d @$INSTALLEPOCH)
    # Print information.
    echo "$PACKAGE was installed on $INSTALLDATETIME"
done
```

Contents

Monitoring Log Alarms and Sending Email Notifications (1)

- Scenario:
  - A carrier requires long-term monitoring of the current environment's **error.log** file, with an alarm email sent if the keyword **danger** is found.
- Requirements:
  - Create a log file and use a script to simulate log file writing.
  - Run the **mail** command to send emails. In a lab environment, however, you can write the **mail** character string to the log file instead.
  - The following figure shows the process.

```
[root@euleros bin]# touch try.log
[root@euleros bin]# ./test.sh&
[5] 18177
[root@euleros bin]# cat try.log
[root@euleros bin]# ./addlog.sh
[root@euleros bin]# cat try.log
danger
mail
[root@euleros bin]#
```

Run the monitoring script.

The initial log content is empty. Enter the keyword **danger**.

Automatically send an email (replaced by writing **mail**).

Monitoring Log Alarms and Sending Email Notifications (2)

- Run the monitoring script in the background.
  - **./test.sh &**
- Simulate the error reporting.
  - **./addlog.sh**
- View the **error.log** file.
  - **cat /var/log/error.log**

```
#!/bin/bash
# Log detection script test.sh

tail -f /var/log/error.log | while read danger;
do echo 'mail' >> /var/log/error.log;
sleep 1m;
done
```

```
#!/bin/bash
# Error reporting simulation script addlog.sh

echo 'danger' >> /var/log/error.log
```

Quiz

1. (Single-answer)Which one in the following commands can work successfully ?
   A. 1myname=openEuler1
   B. myname = openEuler
   C. myname=openEuler
   D. myname =openEuler

HUAWEI

- Answer:
  1. C

Summary

This chapter described shell script elements including execution, compilation, and debugging, using actual cases and sample code to help readers easily apply the material in their own work.

- Execution: Common foreground and background running modes as well as corresponding operations were introduced.
- Compilation: Variables and statement structures required for compiling shell scripts were introduced.
- Debugging: Common debugging methods and the importance of a good style when writing were introduced.

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.

Samba File Sharing Server Management

HUAWEI

Foreword

- After completing the courses on basic openEuler commands and operations, you can set up openEuler-based various services and apply them to IT systems. In this course, you will set up a file sharing server using openEuler and Samba.

Objectives

Upon completing this course, you will understand:

- openEuler-based File sharing server configuration
- File sharing server user and permission configuration
- File sharing server basic O&M and troubleshooting

Contents

Samba File Sharing Service

- Samba is a piece of software which enables the Linux OS to use the Server Message Block (SMB) network communication protocol.
- Samba can be used for file sharing and print sharing between Linux and Windows systems, as well as between Linux systems.
- Samba adopts the client/server structure, so a Samba server can function as:
  - A file sharing server
  - A Samba client
- On Windows, Samba uses the NetBIOS protocol. To use files shared from Linux on Windows, ensure that this protocol is installed.

- SMB is the core startup service of Samba. It is responsible for setting up a session between the Linux Samba server and Samba client, authenticating user identities, and providing access to files and printing systems. Only when SMB is started, files can be shared and TCP port 139 can be listened. The NMB service is used for parsing, which is similar to the function implemented by the DNS. The NMB service can map the name of the workgroup shared by the Linux system to the IP address of the workgroup. If the NetBIOS message block daemon (NMBD) service is not started, the NMBD service can access shared files only through the IP address and listen on UDP ports 137 and 138.

Samba Suites

- To install a Samba server, at least three suites are required: samba, samba-common, and samba-client.
  - samba: contains the smbd and nmbd daemons, Samba documents, log rotation settings, and preset startup options
  - samba-common: provides the main configuration file of the Samba **smb.conf** and the testparm program for checking the syntax of the **smb.conf** file
  - samba-client: provides the tool commands required when running a Samba client on Linux, such as the **smbmount** command for mounting a Samba file system

Samba Configuration Files (1)

- Samba configuration files are stored in **/etc/samba**. The main configuration files are **smb.conf**, **lmhost**, and **smbpasswd**.
  - **smb.conf**: the most important Samba configuration file—also the only basic settings configuration file. Its main sections are:
    - **[global]**: sets server functions
    - **[sharedir]**: sets the attributes of each shared directory
  - **lmhosts**: maps the NetBIOS names and IP addresses of the hosts. Samba is constantly improving and can now, at startup, obtain the IP addresses which correspond to hosts' NetBIOS names in the local area network (LAN). Generally, you do not need to configure this file.
  - **smbpasswd**: If the Samba server requires users to log in with passwords, user passwords are stored in this file.

Samba Configuration Files (2)

- **smb.conf** file example

```
[global]                                # Set the overall environment of the Samba server.
    workgroup = MYGROUP                 # Name of a work group
    server string = Samba Server Version        # Description of the host
    netbios name = MYSERVER             # Default host name of the Samba server
    hosts allow = 127.  192.168.12.  192.168.13.        # IP address range that is allowed to access the
                                                        Samba server. By default, all IP addresses are
                                                        allowed.
    security = user                     # Set the security mode. user mode requires user
                                        authentication while share mode does not.

[homes]
    valid users =           # Specify users who are allowed to access the Samba server.
    invalid users =         # Specify users who are not allowed to access the Samba server.
    write list =            # Specify users who are allowed to write data.
    read list =             # Specify users who are allowed to read data only.
    public = yes            # Specify whether anonymous access is allowed.
```

Common Samba Commands

- Common commands of the Samba server are as follows:
  - **smbpasswd**: sets Samba users and passwords. Option **-a** is used to add a Samba user and set that user's password.
  - **smbclient**: views the directories and devices shared from a host. Option **-L** is followed by the IP address of the host to be viewed, and **-U** is followed by the user name used for login.
  - **smbmount**: similar to the **mount** command, this mounts a shared directory from a remote host to the Linux host.
  - **testparm**: tests whether the Samba configuration is correct by checking the **smb.conf** file. If the file passes the test, the Samba service will be able to properly load the configuration.

Installing Samba from a Specific Source using DNF

- Configure the software source for the DNF software package management tool, then obtain and install the Samba suites, including samba, samba-common, and samba-client along with their dependency packages.
  - Add a DNF software source:
    - dnf config-manager --add-repo repository_url
  - Enable the added DNF software source:
    - dnf config-manager --set-enable repository
  - Download and install the software using DNF:
    - dnf install samba samba-common samba-client

Managing the Samba Service, Listening Port, and Startup Settings

- After Samba is installed, enable the Samba service to start upon system startup, start the Samba service, and check the status of its listening ports.
  - Enable the Samba service to start upon system startup:
    - systemctl enable smb
  - Start the Samba service:
    - systemctl start smb
  - View the Samba service running status:
    - systemctl status smb
  - Check the port listening status:
    - netstat -lantp |grep 139
    - netstat -lantp |grep 445

Contents

Adding Users

- If you need the Samba server to control access to resources, you need to create a user on openEuler and only allow the user to log in through the Samba service (not log in to the system through a shell).
  - Create user **smb**. Do not create a home directory for the user or grant the user shell login permission.

```
[root@openEuler ~]#  useradd smb –M –s  /sbin/nologin
```

  - Allow the **smb** user to log in through the Samba server and set the user's password.

```
[root@openEuler ~]#  smbpasswd –a smb
```

Preparing Shared Files and Setting File Access Permissions

- Provide shared directories for the Samba server. Create the shared files on openEuler and set the permissions on the shared directories.
  - Create shared directories **share** and **smb**.

```
[root@openEuler ~]# mkdir  /var/share /var/smb
```

  - Grant all users the read, write, and execute permissions on the **share** and **smb** shared directories :

```
[root@openEuler ~]# chmod 777 /var/share /var/smb
```

  - Change the owner of the **smb** shared directory to the **smb** user:

```
[root@openEuler ~]#  chown smb:smb /var/smb
```

Configuring Samba Sharing (1)

- Edit the Samba configuration file **smb.conf** to allow the client to anonymously read and write to the **share** directory, and to allow authenticated users to read and write to the **smb** directory.
  - Add **map to guest = Bad User** to the **[global]** section to enable anonymous access.
  - Add a **[share]** section for the **share** directory and set its permissions:

```
[share]
        comment = share
        path = /var/share
        guest ok = yes
        browseable = yes
        writeable = yes
```

Configuring Samba Sharing (2)

- Add an **[smb]** section for the **smb** directory and set its permissions:

```
[smb]
        comment = smb
        path = /var/smb
        write list = smb
        browseable  = yes
        writable = yes
        read list = smb
        valid users = smb
        create mask = 0777
```

Verifying that the Samba Server Can Be Accessed

- On Windows, access the Samba server in file sharing mode. After connecting to the file sharing server, you should be able to create files.
  - Access the directory through *****Samba_IP***\share**. You should be able to access the directory without login authentication and create or delete folders or files.
  - Access the **smb** directory through *****Samba_IP***\smb**. You should be able to open files and create folders or files in the **smb** directory after providing the authentication information of the **smb** user.

Contents

Using a Scheduled Task to Manage Files (1)

- You can schedule a task to back up the shared data of a Samba server every day. Use a shell script and a crontab scheduled task to archive and back up the data in the **share** directory to the **smb** directory.
  - Create the **backup.sh** script in **/root**.

```
#! /bin/sh
mkdir /var/backup                          # Create a temporary backup directory.
cp -r /var/share/ /var/backup/             # Copy the data in the share directory to the backup directory.
tar -zcPvf /var/smb/backup$(date +%Y%m%d).tar.gz /var/backup    # Compress the data in the share
                                                                 directory to /var/backup.
rm -rf /var/backup/                        # Delete the temporary backup directory.
find /var/smb/ -mtime +30 -name "*.tar.gz" -exec rm -rf {} \;    # Delete backup data that is stored
                                                                 more than 30 days.
```

  - Set the execute permission on the script.

```
[root@openEuler ~]# chmod +x backup.sh
```

Using a Scheduled Task to Manage Files (2)

- Set and edit a periodic task by running the **crontab** command. The **backup.sh** script is executed at 22:00 every day. Data is backed up to the **smb** directory as a file named after the time the backup is run.

```
[root@openEuler ~]# crontab –e
0 22 * * * /root/backup.sh
```

- Run the **crontab –l** command to view the backup task.

System and Service Logs

- To view the status or error information of the Samba server through logs, you can run **ls** to find the log file of the Samba service in the **/var/log** directory. Then, you can run a command to view the latest 20 log entries.
  - Run **ls –l /var/log/samba** to find the log file in the **/var/log/samba** directory, for example, **log.smbd**.
  - Run **tail /var/log/samba/log.smbd –n 20** to view the latest 20 log entries.
  - Run **tail /var/log/messages** to view the latest 20 log entries of the openEuler system.

File Sharing Fault Locating and Troubleshooting

- These errors can make the file sharing impossible when you use Samba. Solutions are provided:
  - The file sharing server cannot be accessed or the network connection is faulty.
    - If the firewall is enabled, run **systemctl stop firewalld** to disable it.
    - If the Samba service is not started, run **systemctl restart smb** to restart it.
  - The file sharing server cannot be started due to an incorrect configuration.
    - Run **cd /etc/samba; testparm** to check whether the configuration file is configured correctly, and modify the configuration according to the prompt.
  - The user does not have the permission to access or create files. Permissions on the shared directory are configured incorrectly.
    - If the file owner is incorrect, run **chown smb:smb /var/smb** to change the file owner to the correct user.
    - If the file permissions are incorrect, run **chmod 777 smb:smb /var/smb** to change the file permissions.

Quiz

1. (Multiple-answer)To configure the Samba file sharing server, which software needs to be installed?
   A. samba
   B. samba-common
   C. samba-client
   D. samba-user
2. (True or false)Samba's main configuration file is smb.conf
   A. Ture
   B. False

HUAWEI

- Answer:
  1. ABC
  2. A

Summary

- This course describes how to deploy, configure, and maintain the file sharing server as well as how to use openEuler and how to configure services. It also presents simple service troubleshooting and basic Linux O&M capabilities.

Acronyms

Acronym	Full Name	Description
POSIX	Portable Operating System Interface	Portable Operating System Interface (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
GNU	GNU's Not Unix	GNU (prononunced GAH-noo with a hard "G") is an ambitious project started by Richard Stallman to create a completely free operating system based upon the design of Unix.
AT&T	American Telephone and Telegraph Co.	It was an American telecommunications company founded in 1877
KDE	K Desktop Environment	One of the popular desktop environments for Linux. Kubuntu uses KDE by default.
ksh	Korn shell	An interactive command interpreter and a command programming language. 2. A command interpreter developed for UNIX, which forms the basis for the z/OS shell.
csh	C shell	A command line processor for UNIX that provides interactive features such as job control and command history.
GUI	graphical user interface	A visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes, on the screen.
CLI	command-line interface	A means of communication between a program and its user, based solely on textual input and output.

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.