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.

```
9:44:~/Downloads/people$ ccat bin/smart_thumbnail.sh
#!/bin/bash

for image in $@; do
  echo "Processing $image"
  output_file="smart_thumbs/$(basename ${image%.*}).png"
  algo run opencv/SmartThumbnail -B "$image" -s -o "$output_file"
done
9:45:~/Downloads/people$ bin/smart_thumbnail.sh *.jpg
Processing boy-on-beach.jpg
Processing celebrate.jpg
Processing dancing_stars.jpg
Processing Fareisa.jpg
Processing formal_couple.jpg
Processing kobe.jpg
Processing obama.jpg
Processing smiling-woman.jpg
Processing ultimatums.jpg
Processing walk-on-beach.jpg
Processing water-ski.jpg
Processing woman.jpg
9:46:~/Downloads/people$
```

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 For example: deleting outdated log files.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**.

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 ~]#
```

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.

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.

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.

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.

Power Management Command 3 - reboot

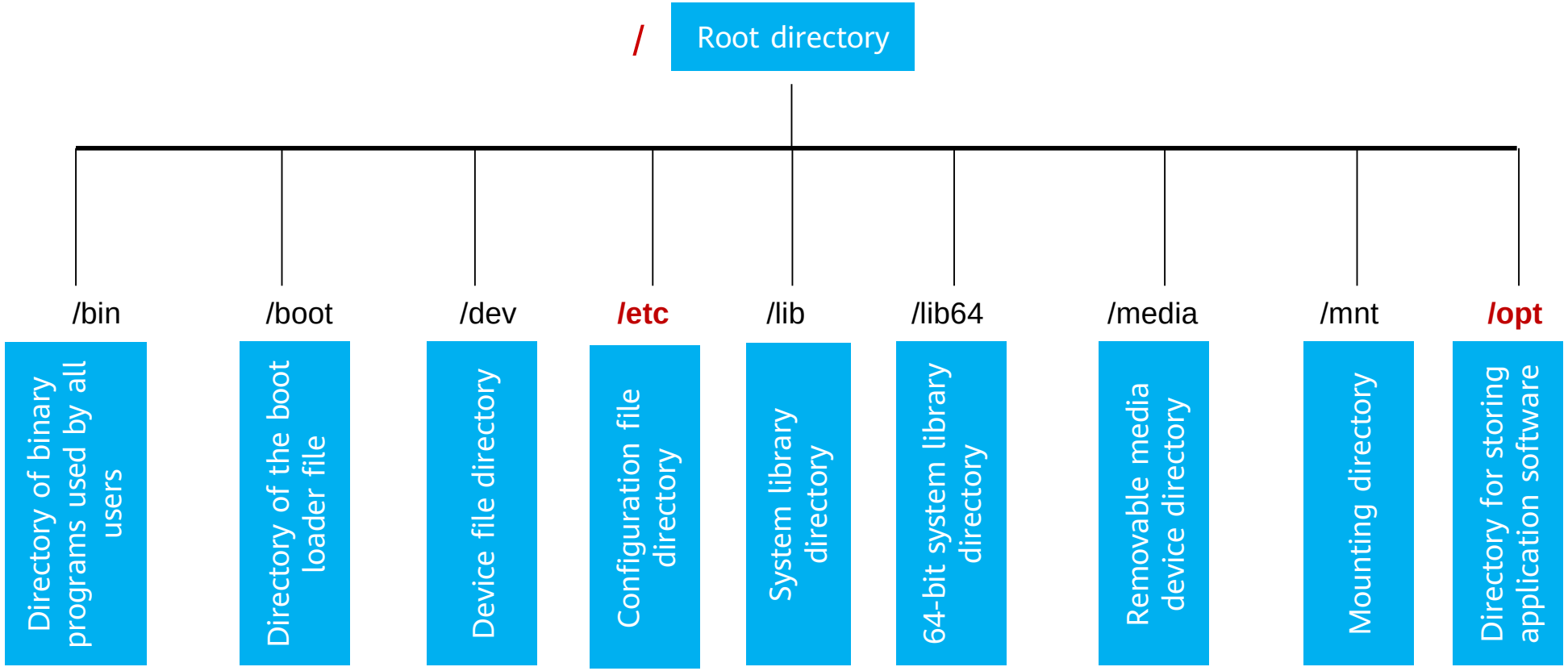
- **reboot** is used to restart the computer, and is applicable to the system administrator.
- Main options:
  - **-n**: saves the data and restarts the system.
  - **-w**: writes records to the **/var/log/wtmp** file. It does not restart the system.
  - **-d**: does not write records to the **/var/log/wtmp** 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

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.

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.

Exemple Absolute Path: /var/log/wtmp

Current Directory : /opt

Relative Path (relative to opt directory) : ../var/log/wtmp

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.

cd : change directory

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

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. Files or directories whose names start with a dot (.) are hidden.
  - **-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.

  -h : show all hidden files and directories

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*...

-p : create directories recursively

```
[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 ~]#
```

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.

-t [date]: change file timestamp

```
[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 ~]#
```

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.

    -s: same as -l

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 ~]#
```

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.

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.

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 ~]#
```

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**.

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 ~]#
```

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.

-n : Specify the number of lines to display

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 ~]#
```

total number of lines minus 40

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 ~]#
```

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.

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.

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.

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 Command 15 - gzip

Very Important !!!

- **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.

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.

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

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 - 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.
```

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.

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

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

HUAWEI