Experimental Instruction Manual

Linux operating system related commands

The command format in Linux is: command [options] [arguments] Square brackets indicate optional, that is, some commands do not require options or parameters, but some commands require multiple options or parameters when running.

- options： options are switches that adjust the execution behavior of commands. Different options determine different display results of commands.
- arguments： The arguments refer to the object of the command.

1. vi/vim

Text editor. Edit the file if it exists, create and edit the text if it doesn't.

Command syntax：

```
vim [arguments]
```

Parameter description: Editable file name.

Command example:

- Edit the xml text named clusterconfig:

```
vim clusterconfig.xml
```

Note：

The vim editor has the following three modes:

- Normal mode: In other modes, press Esc or Ctrl+[to enter, the lower left corner displays the file name or is empty.
- Insert mode: Press i key to enter in normal mode, --INSERT-- is displayed in the lower left corner.
- Visual mode: Press v key to enter in normal mode, and display --VISUAL-- in the lower left corner.

Exit command (in normal mode)：

- :wq to save and exit.
- :q! Force quits and ignores all changes.
- :e! Abandons all changes and opens the original file.

2. cd

Display the name of the current directory, or switch the current directory (open the specified directory).

Command syntax：

```
cd [arguments]
```

Parameter description:

- No parameter: switch the user's current directory.

- . : indicates the current directory;
- .. : indicates the upper level directory;
- ~ : indicates the home directory;
- / : Indicates the root directory.

Command example:

- Switch to the bin directory under the usr directory：

```
cd /usr/bin
```

- Switch to the user's home directory：

```
cd
```

- Switch to the current directory (cd followed by a.) ：

```
cd .
```

- Switch to the directory one level above the current directory (the cd followed by two.) ：

```
cd ..
```

- Switch to the user home directory:

```
cd ~
```

- Switch to the root directory:

```
cd /
```

Note: Switching directories requires understanding the concepts of absolute paths and relative paths.

- Absolute path: In Linux, the absolute path starts from / (ie the root directory), such as /opt/software, /etc/profile, if the directory starts with /, it is an absolute directory.
- Relative path: a directory starting with . or ... . indicates the location where the user is currently operating, and .. indicates the parent directory. For example, ./gs_om represents a file or directory in the current directory.

3. mv

Moving (rename) files or moving files or directories to other locations is often used to back up files or directories.

Command syntax：

```
mv [options] argument1    argument2
```

Common options：

- -b: If the file needs to be overwritten, backup it before overwriting.

Parameter description：

- Parameter 1: source file or directory.
- Parameter 2: target file or directory.

Command example：

- Rename the file python to python.bak:

```
mv python python.bak
```

- Move all files and directories in the /physical/backup directory to the /data/dbn1 directory:

```
mv  /physical/backup/*  /data/dbn1
```

4. curl

In Linux curl is a file transfer tool that works from the command line using URL rules. It supports uploading and downloading of files and is a comprehensive transfer tool.

Command syntax：

```
curl [options] [URL]
```

Common options：

- -A/--user-agent <string>：Set the user agent to send to the server;
- -C/--continue-at <offset>：Continue at breakpoint;
- -D/--dump-header <file>：Write the header information to the file;
- -e/--referer：source URL；
- -o/--output：write the output to this file;
- -O/--remote-name：Write the output to this file, keeping the filename of the remote file;
- -s/--silent：silent mode. do not output anything;
- -T/--upload-file <file>：upload files;
- -u/--user <user[:password]>：Set the server user and password;
- -x/--proxy <host[:port]>：use an HTTP proxy on the given port;
- -#/--progress-bar：A progress bar shows the current delivery status.

Parameter description：

- URL：The specified file transfer URL address.

Command example：

- Save the content of the url (https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo) to the file /etc/yum.repos.d/CentOS-Base.repo

```
curl -o /etc/yum.repos.d/CentOS-Base.repo https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo
```

- If the connection is dropped during transmission, you can use -C to resume the transmission.

```
curl -C -O https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo
```

5. yum

Shell front-end package manager. Based on RPM package management, it can

automatically download and install RPM packages from a specified server, handle dependencies automatically, and install all dependent software packages at one time, without having to download and install tediously again and again.

Command syntax：

```
yum [options] [command] [package ...]
```

Common options：

- -h：view help;
- -y：When the installation process prompts to select all "yes";
- -q：The installation process is not displayed.

Parameter description：

- command：action to be performed.
- package：Installed package name.

Command example：

- List all updatable software inventory commands:

```
yum check-update
```

- Update all software commands:

```
yum update
```

- List all installable software inventory commands:

```
yum list
```

- Install the specified software:

```
yum install -y libaio-devel flex bison ncurses-devel glibc.devel patch lsb_release wget python3
```

6. wget

wget is the most common command for downloading files under Linux. wget supports HTTP, HTTPS and FTP protocols, and supports automatic download, that is, it can be executed in the background after the user exits the system until the download ends.

Command syntax：

```
wget [options] [URL]
```

Common options:

- -c：Then download the unfinished file;
- -b：Transfer to background execution after startup;
- -P：Specify the download directory;
- -O：Change the download file name;
- --ftp-user --ftp-password：Download using FTP user authentication.

Parameter description:

- The specified file download URL.

Command example：

- Download the openGauss database installation file to the current folder:

```
wget https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/x86/openGauss-1.0.0-CentOS-64bit.tar.gz
```

- Use wget to resume the upload:

```
wget –c https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/x86/openGauss-1.0.0-CentOS-64bit.tar.gz
```

7. ln

Create a synchronous link for a file in another location (soft and hard links, no option is a hard link).

When the same file needs to be used in different directories, there is no need to put a file that must be the same in each required directory. We only need to put the file in a fixed directory, and then put it in other directories. You can use the ln command to link (link) it in the directory of the directory, and it is not necessary to repeatedly occupy the disk space.

Command syntax：

```
ln [options] argument1 argument2
```

Common options：

- -b --Delete, overwrite previously established links;
- -d -- Allow superusers to make hard links to directories;
- -s -- Soft links (symbolic links).

Parameter description：

- Parameter 1: source file or directory.
- Parameter 2: The linked file or directory.

Command example：

- Create a soft link /usr/bin/python for python3 files. If python3 is missing, /usr/bin/python will be invalid:

```
ln -s python3 /usr/bin/python
```

- Create a hard link /usr/bin/python for python3, python3 has the same attributes as /usr/bin/python:

```
ln python3 /usr/bin/python
```

8. mkdir

To create a directory with the specified name, the user who creates the directory is required to have write permission in the current directory, and the specified directory name cannot be an existing directory in the current directory.

Command syntax：

```
mkdir [options] [argument]
```

Common options：

- -p -- can be a pathname. At this time, if some directories in the path do not yet exist, after adding this option, the system will automatically create those directories that do not exist, that is, multiple directories can be created at one time (recursive);
- -v --Display information every time a new directory is created;
- -m --Set permissions <mode> (similar to chmod) instead of rwxrwxrwx minus umask.

Parameter description：

- The directory to be created.

Command example：

- Create an empty directory:

```
mkdir test
```

- Create multiple directories recursively:

```
mkdir -p /opt/software/openGauss
```

- Create a directory with a permission of 777 (the permission of the directory is rwxrwxrwx):

```
mkdir –m 777 test
```

9. chmod

Change file permissions.

Command syntax：

```
chmod [options] <mode> <file...>
```

Common options：

- -R, --Recursively make the same permission changes to all files and subdirectories in the current directory.

Parameter description：

- mode：Permission setting string, the detailed format is as follows：

```
[ugoa...][[+-=][rwxX]...][,...],
```

Among them, [ugoa...]: u means the owner of the file, g means those who belong to the same group as the owner of the file, o means other people, a means all (including the above three) ;[+-=]: + means increase permission, - means cancel permission, = means only set permission; [rwxX]: r means readable, w means writable, x means executable, X means only when The file is a subdirectory or the file has been made executable.

- file：List of files (single or multiple files, folders).

Command example：

- Set the file cluterconfig.xml to be readable by all users：

```
chmod ugo+r cluterconfig.xml
或
```

```
chmod a+r    cluterconfig.xml
```

- Set all files and subdirectories in the current directory to be readable and writable by anyone：

```
chmod -R a+rw *
```

Digital rights use format:

- In this usage mode, the numbers 4, 2 and 1 are specified to indicate read, write and execute permissions, that is, r=4, w=2, x=1.
- Example： rwx = 7 （4+2+1）； rw = 6 （4+2）； r-x = 5 （4+0+1）； r-- = 4 （4+0+0）； --x = 1 （0+0+1）；

Each file can be set with different rwx (read, write and execute) permissions for three granularities. That is, we can use three octal numbers to represent the permission details of owner , group , and other groups (u , g , o), and use chmod to directly add three octal numbers to directly change the file permissions. The syntax format is:

```
chmod <abc> file...
```

Among them, a, b, c are each a number, representing the permissions of User, Group, and Other respectively, which is equivalent to the simplified version of chmod u= permission, g= permission, o= permission file..., and the permission here The read, write, and execute permissions of User, Group, and Other will be represented by octal numbers.

Command example：

- Give the cluterconfig.xml file readable, writable and executable permissions (all permissions)：

```
chmod 777 cluterconfig.xml
```

- Give all files and subdirectories in the /opt/software/openGauss directory readable and executable permissions for all users, and readable and executable permissions for other users:

```
chmod    R 755 /opt/software/openGauss
```

10. chown

Use chown to change the owner of the specified file to the specified user or group, the user can be the user name or user ID; the group can be the group name or group ID; the file is a space-separated list of files whose permissions are to be changed, and wildcards are supported. Only the system administrator (root) has such privileges. Permission to use: **root**。

Command syntax：

```
chown [options] user[:group] file...
```

Common options:

- -c : information showing the changed part;
- -f : ignore error messages;
- -R : Process all files in the specified directory and its subdirectories.

Parameter description:

- user : User ID of the new document owner.
- group : The user group of the new file owner.

- flie：File.

Command example：

- Set the owner of the file file1.txt to omm, and the group user dbgrp:

```
chown omm:dbgrp /opt/software/openGauss/clusterconfig.xml
```

- Set the owner of all files and subdirectories in the current directory to omm, and the group user dbgrp:

```
chown -R omm:dbgrp *
```

11. ls

List the contents of files and directories.

Command syntax：

```
ls [options] [argument]
```

Common options：

- -l --Display in long format, listing the details of the file, such as creator, creation time, list of read and write permissions of the file, etc.;
- -a --List all the files under the file, including hidden files starting with "." and ".." (the hidden files of files under Linux start with ., if there is .. it means there is a parent directory);
- -d --List the directory itself rather than the files in the directory, usually used with -l;
- -R--List all subdirectory layers at the same time, similar to -l, except that the owner of the file is not displayed, which is equivalent to the "recursive" implementation in programming;
- -t -- Sort files by time, Time (time);
- -s--Print out the size of the file after each file, size (size);
- -S -- Sort by file size.

Parameter description：

- Directory or file.

Command example：

- List the files and directories in the current directory in long format:

```
ls -l
```

12. cp

Copy files or directories.

Command syntax：

```
cp [options] argument1 argument2
```

Common options：

- -f -- If the target file cannot be opened, remove it and try again (not necessary when the -n option is present);

- -n -- Do not overwrite existing files (invalidate the previous -i option);
- -I -- Ask before overwriting (disables the previous -n option);
- -p -- Keep the specified attributes (default: mode, ownership, timestamp), if possible additional attributes: environment, link, xattr, etc.;
- -R,-r -- Copy the directory and all items within the directory.

Parameter description:

- Parameter 1: source file.
- Parameter 2: target file.

Command example:

- Copy the abc file in the home directory to the opt directory:

```
cp /home/abc /opt
```

Note: When the target file exists, it will ask whether to overwrite. This is because cp is an alias for cp -i. When the object file exists, even if the -f flag is added, it will ask whether to overwrite.

13. rm

Delete one or more files or directories in a directory, it can also delete a directory and all files and subdirectories under it. For linked files, only the link is deleted, and the original files remain unchanged.

rm is a dangerous command, be careful when using it, otherwise the whole system will be destroyed by this command (for example, execute rm * rf in / (root directory)). Therefore, before executing rm, it is best to confirm which directory it is in and what to delete, and maintain a high level of sobriety during the operation.

Command syntax:

```
rm [options] documents
```

Common options:

- -f -- Ignore non-existing files, never prompt;
- -r -- Instructs rm to recursively delete all directories and subdirectories listed in the argument.

Parameter description:

- The file or directory to be deleted.

Command example:

- To delete files:

```
rm qwe
```

Note: After entering the rm qwe command, the system will ask whether to delete the file. After entering y, the file will be deleted. If you do not want to delete the file, enter n.

- Forcibly delete a file:

```
rm-rf clusterconfig.log
```

14. cat

Concatenate files and output on standard output. This command is often used to display the contents of a file, or to concatenate several files for display, or to read the contents from standard input and display it. It is often used in conjunction with the redirection symbol.

Command syntax：

```
cat [options] [arguments]
```

Common options：

- -E -- Display $ at the end of each line;
- -n –Number all output lines starting from 1;
- -b or --number-nonblank： Similar to -n, except that blank lines are not numbered;
- -v -- Use ^ and M- notation, except for LFD and TAB.

Parameter description：

- Actionable file name.

Command example:

- Display the content of the testfile file:

```
cat textfile
```

- After adding the line numbers (no blank lines) to the document contents of textfile1 and textfile2, append the contents to the textfile3 document:

```
cat -b textfile1 textfile2 >> textfile3
```

- Append content to /etc/profile (enter EOF to end appending):

```
cat >>/etc/profile<<EOF
>export LD_LIBRARY_PATH=$packagePath/script/gspylib/clib:$LD_LIBRARY_PATH
>EOF
```

Note：

- EOF is the abbreviation of end of file, which means the end of "text stream" (stream). The "text stream" can be a file or standard input (stdin). In Linux systems, EOF is a signal value (ie -1) returned when the system reads the end of the file.