**LS command:**

The ls command is the most commonly used command in linux, and the ls command is an acronym for list.

Ls is used to print a list of current directories. If ls specifies a different directory, then the list of files and folders in the specified directory will be displayed.

By using the ls command, you can view not only the files contained in the linux folder, but also file permissions (including directories, folders, and file permissions) and directory information, and so on.

**(1) Command Format**

ls [OPTION]

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -a | -all List all files in the directory, including hidden files beginning with . |
| -l | In addition to the file name, the file permissions, owner, file size, etc. are listed in detail. |
| -d | -directory displays the directory as a file instead of the file in it |
| -h | –human-readable List file sizes in an easy-to-understand format (eg 1K 234M 2G) |
| -t | Sort by file modification time |

**(3) Common Examples**

**Example 1:** List the details of all the files and directories in the /home folder. You can use the following command:

ls -a -l /home

ls -al /home

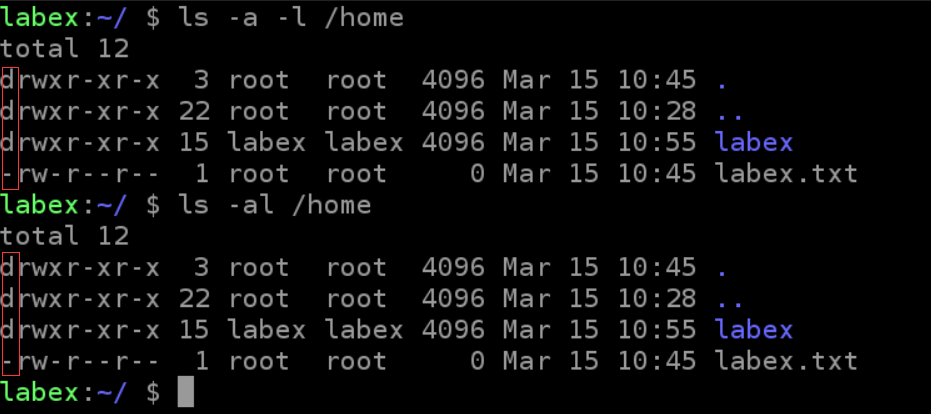
To make it easier to see the test results, we first create a test file. Here we will use the cd and touch commands. We will learn later so that We are not explain it here. Create commands are as follows:

cd /home

sudo touch labex.txt

image desc

The two commands above perform the same result. The result is as follows:



Note the red box above, **d\*\* stands for directory, \*\*-** indicates that files, files, and directories display different colors.

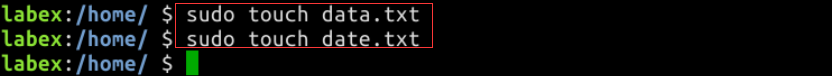
**Example 2:** List the contents of all the file directories starting with "d" in the current directory. You can use the following command:

ls -l d\*

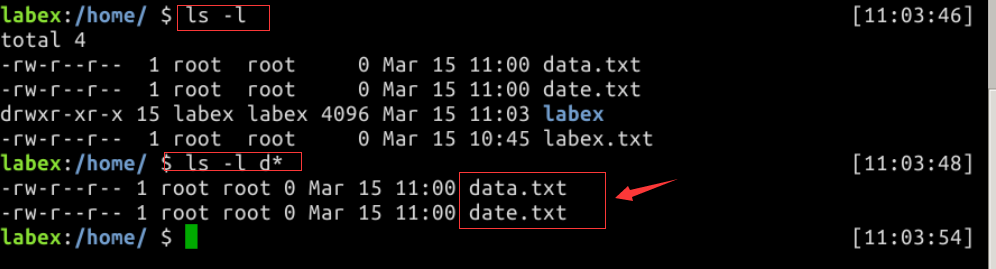
Here for the sake of showcase, we create 2 test files. touch command is used here which will be explained later:

sudo touch data.txt

sudo touch date.txt

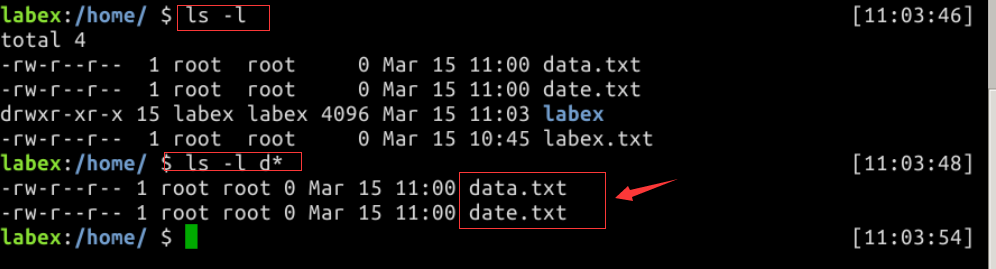


output：



**Example 3:** List the size of all the file directories in the /home directory in an easy-to-understand format. You can use the following command:

ls -alh /home



**（4）Questions**

1. List the size of all file directories beginning with "d" in the /home directory in an easy-to-understand format.
2. List all directories in the /home directory that begin with "s".

**CD Command:**

The cd command can be said to be the most basic command statement in Linux. Other command statements are to be operated on using the cd command. The cd command is an abbreviation of change directory, which switches the current directory to the specified directory.

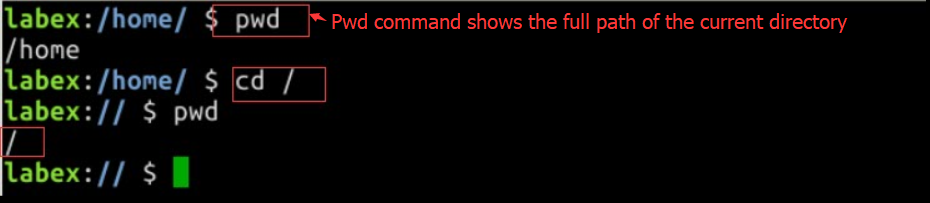
**(1) Command Format**

cd [Directory ]

**(2) Common Examples**

**Example 1:** From the current directory enter into the system root directory, you can use the following command:

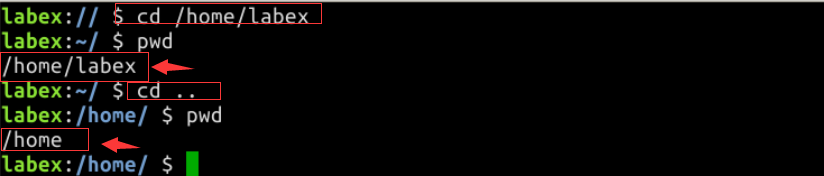
cd /



**Example 2:** From the current directory enter into the parent directory, you can use the following command:

cd ..

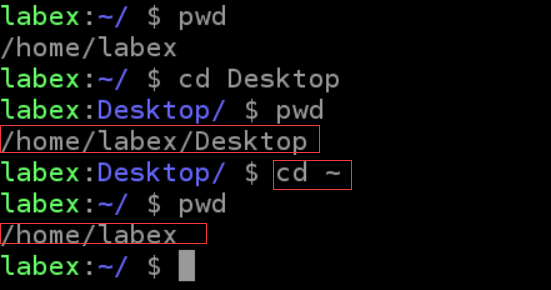
.. represents the parent directory



**Example 3:** Entering the current user's home directory from the current directory, you can use the following command:

cd ~

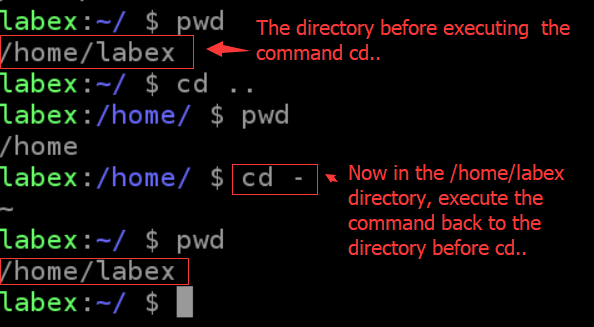
~ Indicates the current user's home directory, not the same concept as the system root directory



**Example 4:** From the current directory into the previous directory, you can use the following command:

cd -

* Indicates last entered directory



**PWD Command:**

We use the pwd command in Linux to see the full path to the "current working directory". Simply put, every time you operate in the terminal, you will have a current working directory. When you are not sure about the current location, pwd is used to determine the exact location of the current directory within the file system.

The pwd command is an acronym for Print Working Directory.

**(1) Command Format**

pwd [OPTION]

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -P | Display actual physical path instead of using link path |
| -L | When the directory is a connection path, the connection path is displayed |

**(3) Common Examples**

**Example 1:** To display the path of the current directory, use the following command:

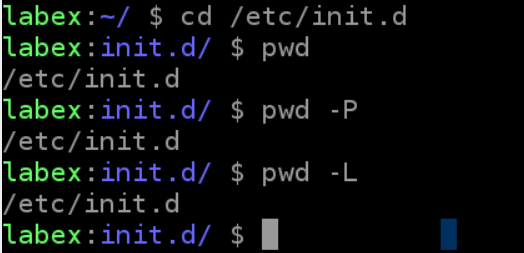
pwd

**Example 2:** To display the physical path of the current directory, use the following command:

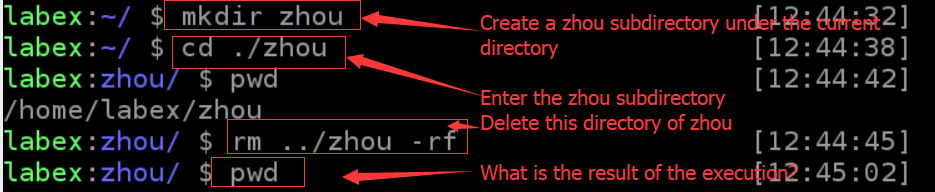
pwd -P

**Example 3:** To display the connection path of the current directory, use the following command:

pwd -L



**(4) Questions**



**Mk dir:**

**(1) Command Format**

mkdir [options] directory

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -m --mode=mode | Set permissions |
| -p --parents | It can be a path name. If some of the directories in the path do not exist yet, this option will automatically create directories that do not yet exist, that is, multiple directories can be created at one time. |
| -v –verbose | Each time a new directory is created, information is displayed |

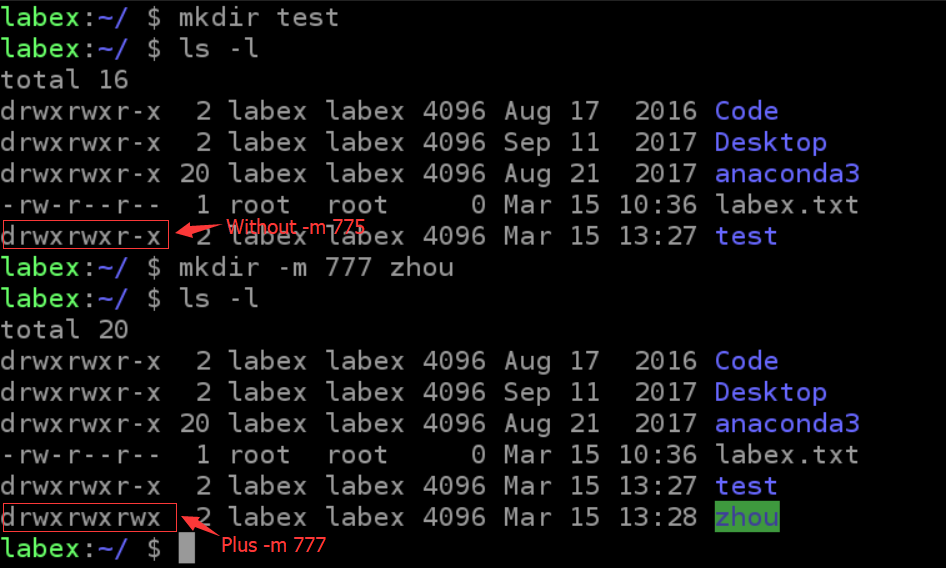
**(3) Common Examples**

**Example 1:** Recursively create multiple directories, you can use the following command:

mkdir -p zhou/test

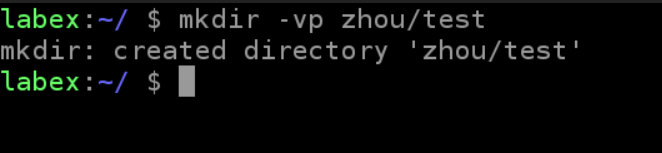
**Example 2:** To create a directory with privilege 777, use the following command:

mkdir -m 777 zhou



**Example 3:** To create a directory display message, use the following command:

mkdir -vp zhou/test



**(4) Questions**

Try the following command and see if anything magic happens

Tree needs to be installed separately

sudo apt-get update

sudo apt-get install tree

mkdir -vp shiyanlou/{lib/,bin/,doc/{info,product}}

tree shiyanlou/

**RM command:**

Rm is a commonly used command. The function of this command is to delete one or more files or directories in a directory. It can also delete a directory and all files and subdirectories under it. For the link file, only the link is deleted and the original file remains unchanged.

Rm is a dangerous command. Be careful when using it. Especially for novices, the entire system will be destroyed in this command (for example, rm \* -rf under / (root directory)). Therefore, before we implement rm, it is better to confirm which directory we are in, and what we want to delete.

The rm command is an abbreviation for remove.

**(1) Command Format**

rm [options] file or directory

**(2) Common Parameters**

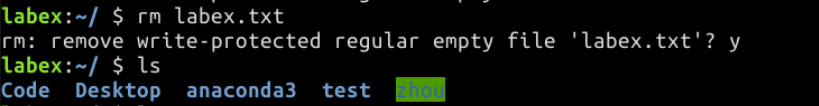
| **Parameter** | **Description** |
| --- | --- |
| -f --force | Ignore nonexistent files and never give prompts |
| -i --interactive | Make an interactive deletion |
| -r --recursive | Instructs rm to recursively delete all directories and subdirectories listed in the parameter |
| -v --verbose | Detailed display of the steps performed |

**(3) Common Examples**

**Example 1:** delete the file, the system will first ask whether to delete, you can use the following command:

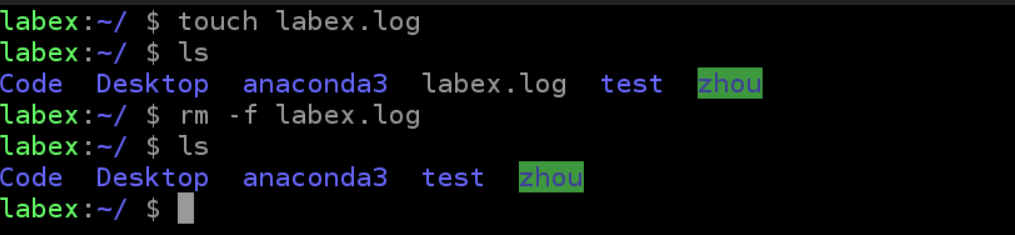
rm labex.txt

image desc



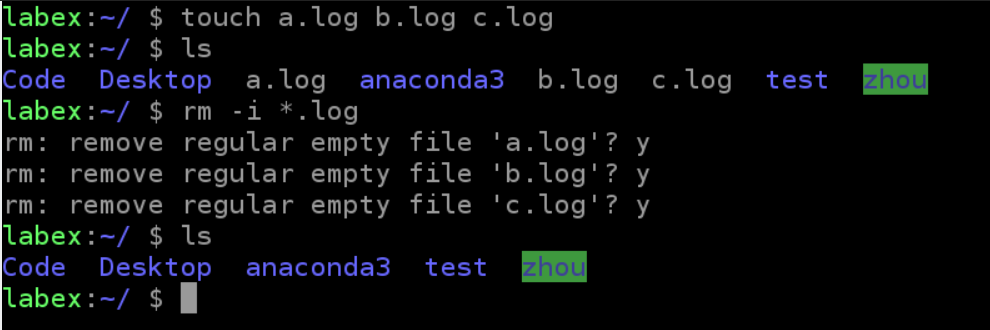
**Example 2:** Forcibly deleting files, the system no longer prompts, you can use the following command:

rm -f labex.log



**Example 3:** Delete all the .log files with suffixes. Before deleting them, you can use the following command:

rm \*.log 或 rm -i \*.log



**(4)Questions**

Consider what the following command does.

myrm(){ D=/tmp/$(date +%Y%m%d%H%M%S); mkdir -p $D; mv "$@" $D && echo "moved to $D ok"; }

**MV command:**

The mv command's function is used to move files or change file names. It is a commonly used command on Linux systems and is often used to back up files or directories.

The mv command decides to either rename the file or move it to a new directory based on the second parameter type (either the target file or the target directory). When the second parameter type is a file, the mv command completes the file renaming. At this time, the source file can only have one (it can be the source directory name). It renames the given source file or directory to a given file. Target file name. When the second parameter is an existing directory name, there can be multiple source file or directory parameters. The mv command moves the source files specified by each parameter to the target directory.

The mv command is an abbreviation of move.

**(1) Command Format**

mv [options] source file or directory target file or directory

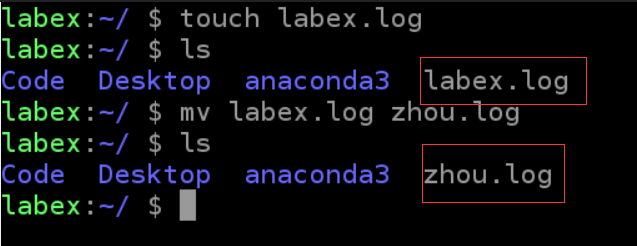
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -b --back | If you need to overwrite the file, you will need to back it up before overwriting. |
| -f --force | If the target file already exists, it will be directly overwritten without asking |
| -i --interactive | If the target file already exists, it will ask whether to cover |
| -u --update | If the target file already exists and the source file is new, it will be updated |
| -t --target | This option is suitable for moving multiple source files to a directory. At this time, the target directory is first and the source file is later. |

**(3) Common Examples**

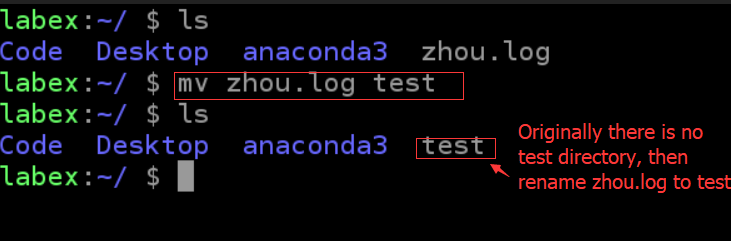
**Example 1:** Rename the file labex.log to zhou.log. You can use the following command:

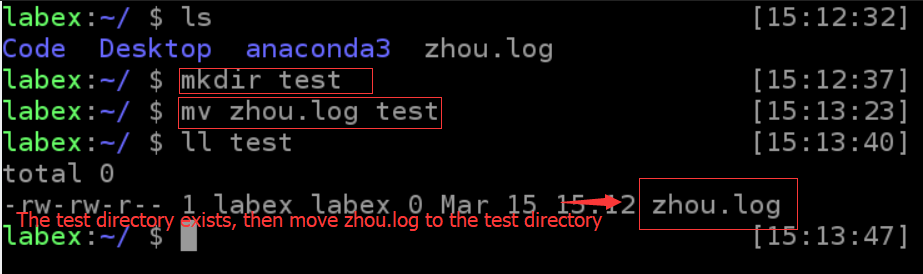
mv labex.log zhou.log



**Example 2:** Move the file zhou.log to the test directory (the test directory must already exist, otherwise perform the rename), you can use the following command:

mv zhou.log test





**Example 3:** Move the file a.txt to the test1 directory. If the file exists, you will be asked whether to overwrite it before overwriting. You can use the following command:

mv -i a.txt test1

**CP Command:**

The cp command is used to copy files or directories and is one of the most commonly used commands on Linux systems. Under normal circumstances, the shell will set an alias. When the file is copied from the command line, if the target file already exists, it will ask whether to overwrite, whether you use the -i parameter or not. However, if you execute cp in a shell script, there is no -i parameter that will not ask for overwriting. This shows that the command line and the shell script perform differently.

The cp command is an abbreviation for copy.

**(1) Command Format**

cp [options] source file directory

cp [options] -t directory source files

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -t --target-directory | Specify the target directory |
| -i --interactive | Ask before overwriting (make the previous -n option invalid) |
| -n --no-clobber | Do not overwrite existing files (disable previous -i option) |
| -s --symbolic-link | Create symbolic links to source files instead of copying files |
| -f --force | Forcibly copy files or directories, regardless of whether the destination file or directory already exists |
| -u --update | After using this parameter, the file will be copied only when the modification time of the source file is more than the destination file, or the corresponding destination file does not exist. |

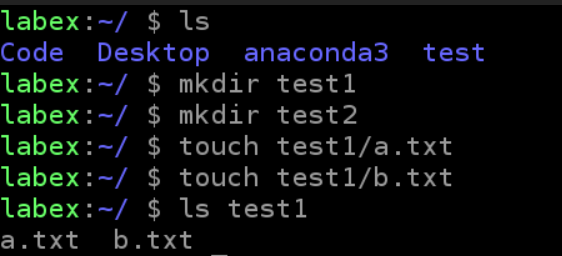
**(3) Common Examples**

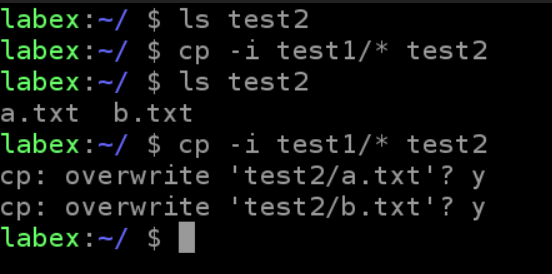
**Example 1:** To create a symbolic link to the file labex.log zhou.log, you can use the following command:

cp -s labex.log zhou.log

**Example 2:** Copy all the files in the test1 directory to the test2 directory. Before overwriting, you can use the following command:

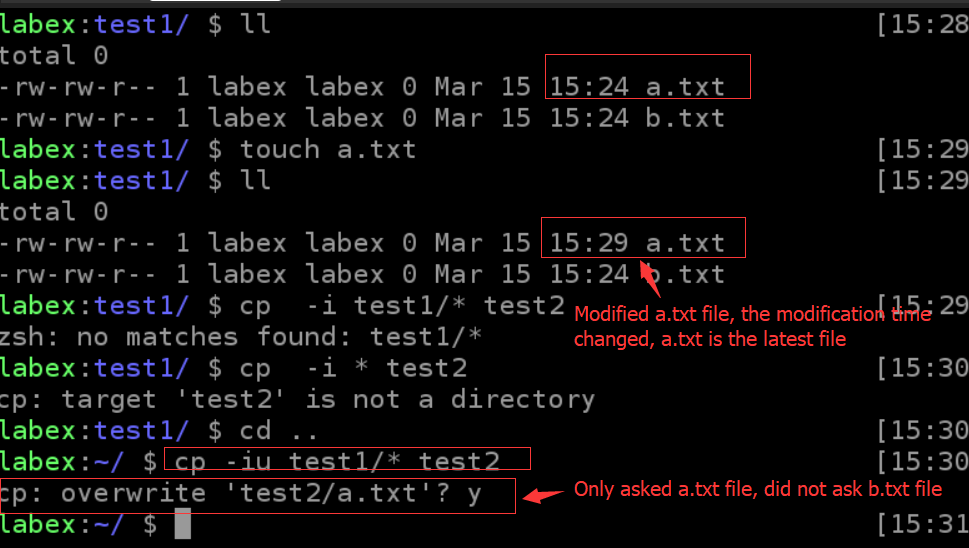
cp -i test1/\* test2





**Example 3:** Copy the most recently updated file from the test1 directory to the test2 directory. Before overwriting, use the following command:

cp -iu test1/\* test2



**Cat Command:**

The function of the cat command is to output a file or standard input combination to standard output. This command is often used to display the contents of a file, or to link several files together, or to read and display content from standard input. It is often used in conjunction with redirection symbols.

The cat command is an abbreviation for concatenate.

**(1) Command Format**

cat [option](https://labex.io/lab/file)

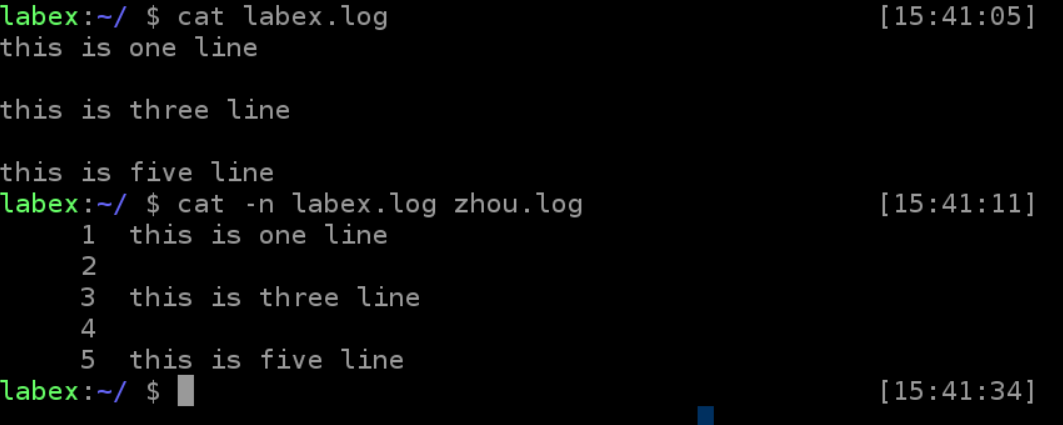
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -A --show-all | Equivalent to -vET |
| -b --number-nonblank | Number of non-empty output lines |
| -e | Equivalent to -vE |
| -E --show-ends | Display at the end of each line |
| -n --number | For all the output line numbers, starting from 1 for all output line numbers |
| -s --squeeze-blank | If there are more than two blank lines, replace it with a blank line |
| -t | Is equivalent to -vT |
| -T --show-tabs | Display tab characters as ^I |
| -u | (be ignored) |
| -v --show-nonprinting | Use ^ and M-references, except LFD and TAB |

**(3) Common Examples**

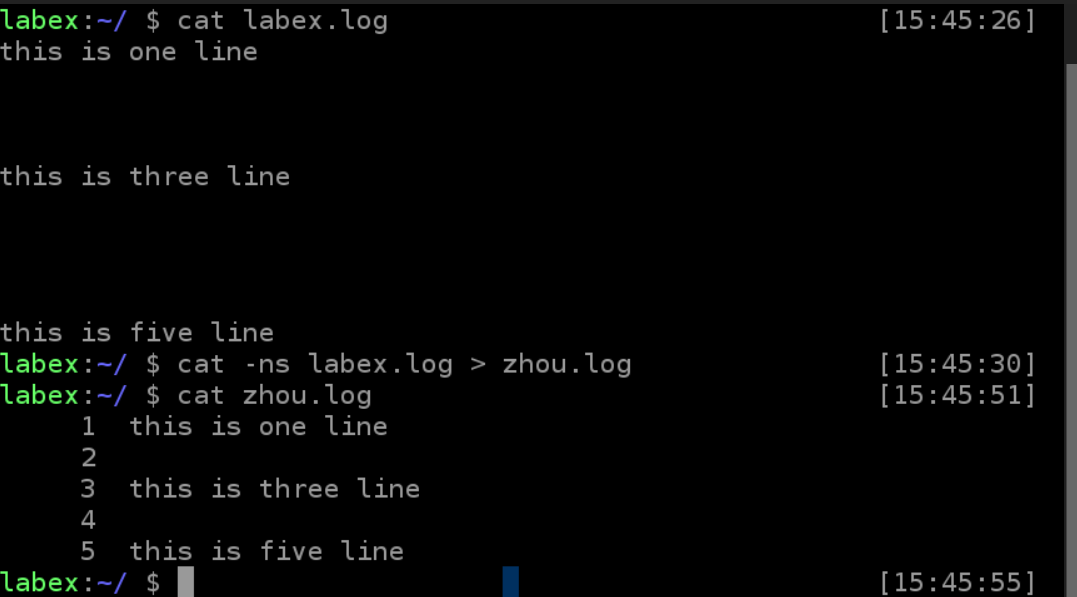
**Example 1:** Input labex.log file with line number and input zhou.log file. You can use the following command:

cat -n labex.log > zhou.log



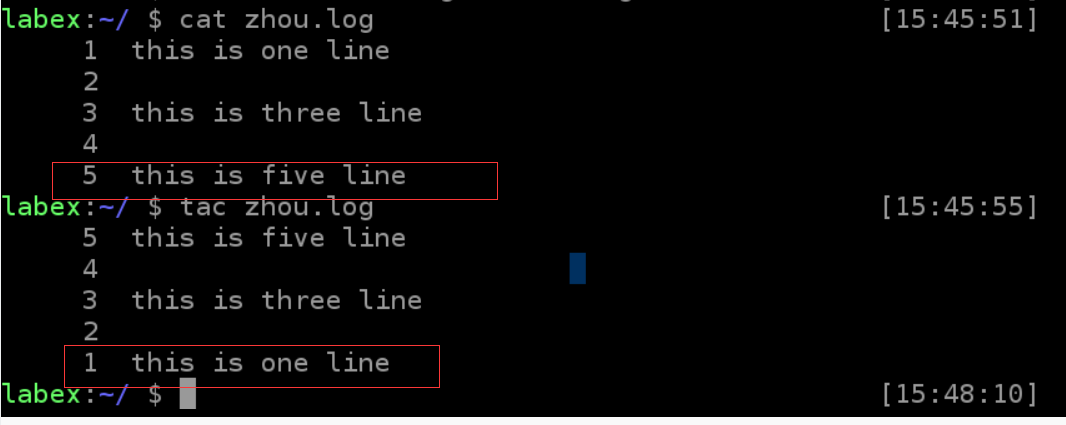
**Example 2:** Input labex.log file with line number and input zhou.log file, replace multi-line blank line with one line output. You can use the following command:

cat -ns labex.log > zhou.log



**Example 3:** To reverse the contents of the zhou.log file, use the following command:

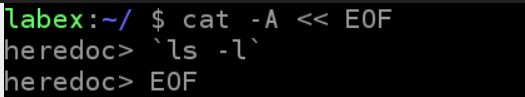
tac zhou.log



Description: tac is to overwrite cat, so its function is the opposite of cat, cat is continuously displayed on the screen from the first line to the last line, and tac is the reverse from the last line to the first line on the screen display.

**(4) Questions**

Execute the following command will output what.



**Nl command:**

The nl command is used in the linux system to calculate the line number in the file. Nl can automatically add the line number to the output file content. The default result is slightly different from cat -n. nl can display the line number more than the display design, including the number of bits and whether it automatically fills 0 and so on.

The nl command is an abbreviation for number of lines.

**(1) Command Format**

nl [option](https://labex.io/lab/file)

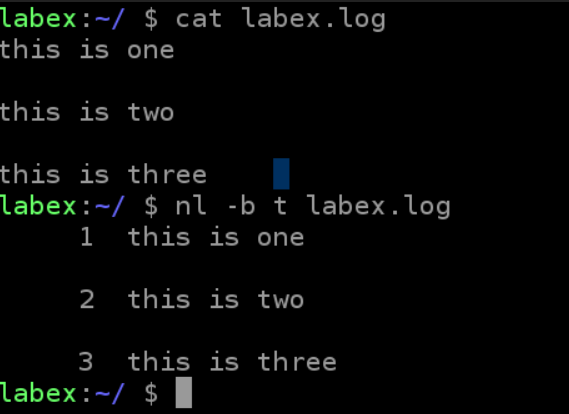
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -b | There are two main ways to specify the line number: |
| -b a | Indicates that the line number is also listed whether it is a blank line or not (similar to cat -n) |
| -b t | If there is an empty line, do not list the line number in the empty line (default) |
| -n | There are three main ways to list line numbers: |
| -n ln | The line number is displayed at the far left of the screen |
| -n rn | The line number is displayed on the far right of its own field, without adding 0 |
| -n rz | The line number is displayed on the far right of its own field, plus 0 |
| -w | The number of digits occupied by the row number field |

**(3) Common Examples**

**Example 1:** After the labex.log file is added with the line number, the blank line without the line number can be used as follows:

nl -b t labex.log

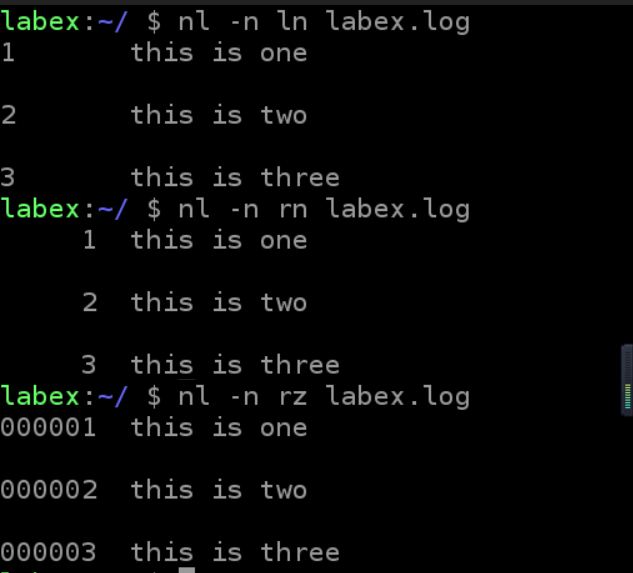


**Example 2:** After the labex.log file is added with the line number, the line number is displayed at the leftmost, rightmost, and rightmost 0 of the screen. You can use the following command:

nl -n ln labex.log

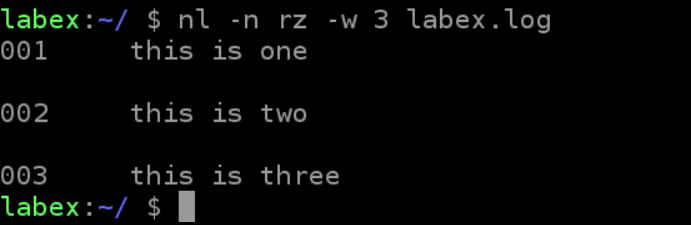
nl -n rn labex.log

nl -n rz labex.log



**Example 3:** After adding the line number to the labex.log file contents, the line number is displayed at the far right of the screen plus 0. The line number column has 3 bits. You can use the following command:

nl -n rz -w 3 labex.log



**More command:**

The more command, which is similar in function to cat , is to display the contents of the entire file from top to bottom on the screen. The more command displays page by page, allowing users to read page by page. The most basic command is to press the space bar to display on the next page, and press the b button to go back to the next page. There is also a search string function. The more command reads the file from front to back, so the entire file is loaded at startup.

**(1) Command Format**

more [option] file

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| +n | Start from n line |
| -n | Define the screen size as n lines |
| +/pattern | Search for the pattern before each file is displayed, then start from the first two lines of the string |
| -c | Clear the screen from the top then show |
| -d | Prompt "Press space to continue, 'q' to quiet", disable the ring function |
| -p | Foil the file by clearing the window instead of scrolling, similar to the -c option |
| -s | Displaying consecutive blank lines as one line |
| -u | Remove the lower line from the contents of the file |

**(3) Common operations**

| **symbol** | **Description** |
| --- | --- |
| = | Output the current line number |
| q | Exit more |
| space bar | Scroll down one screen |
| b | Return to previous screen |

**(4) Common Examples**

Please create a file labex.log with the following contents:

2014-11-5 a

2014-11-5 b

2014-11-5 c

2014-11-5 d

2014-11-5 e

2014-11-5 f

2014-11-5 g

2014-11-5 h

2014-11-5 e

2014-11-5 a

2014-11-5 b

2014-11-5 c

2014-11-5 d

2014-11-5 e

2014-11-5 f

2014-11-5 g

2014-11-5 h

2014-11-5 a

2014-11-5 b

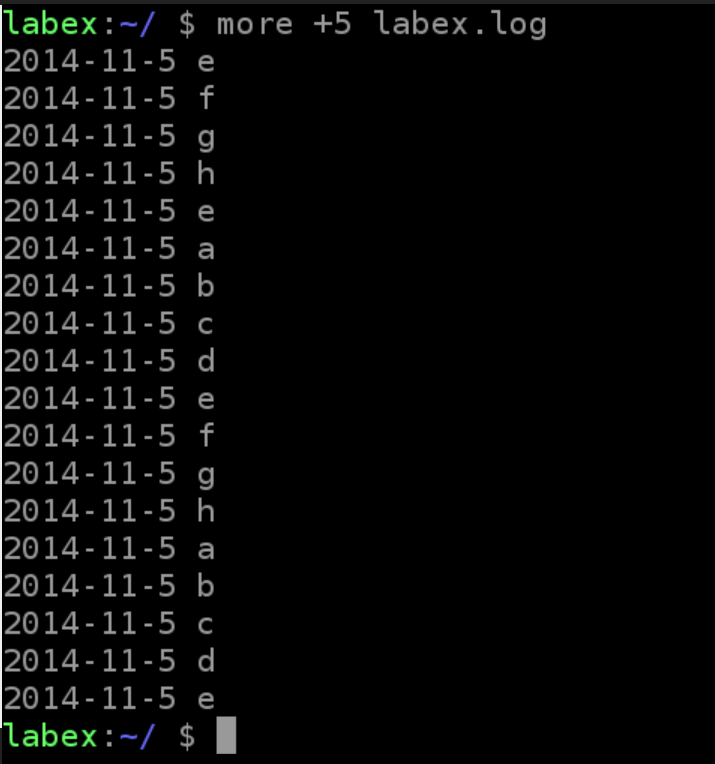
2014-11-5 c

2014-11-5 d

2014-11-5 e

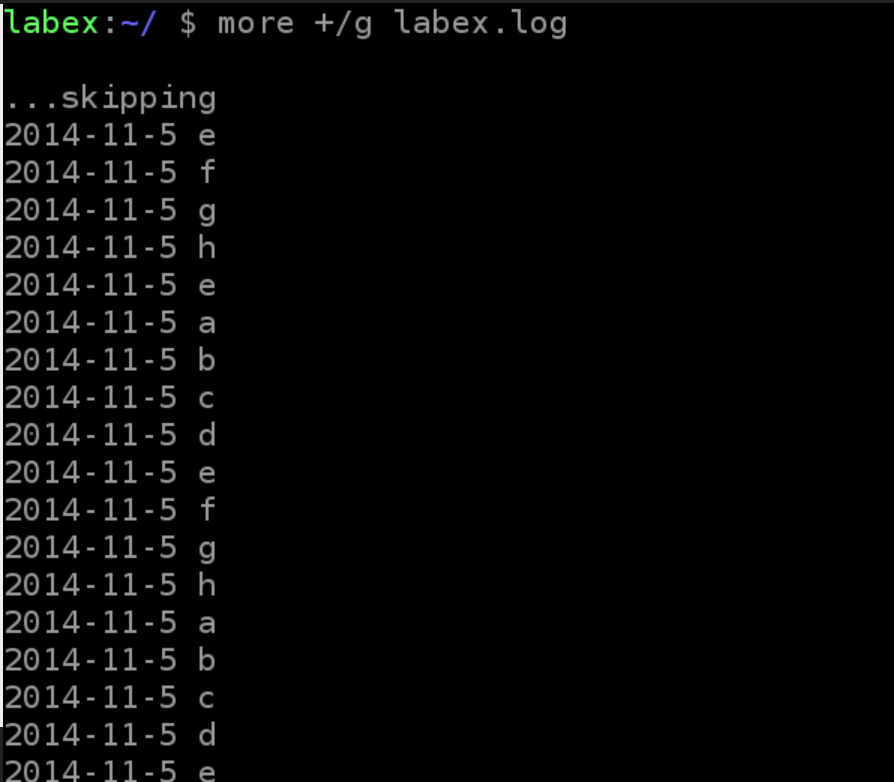
**Example 1:** To display the contents of the labex.log file from the fifth line, use the following command:

more +5 labex.log



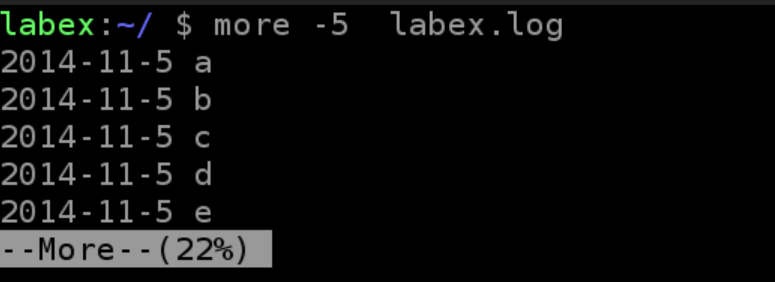
**Example 2:** Find the first line of the "g" string from the labex.log file, and display the output starting from the first two lines. Use the following command:

more +/g labex.log



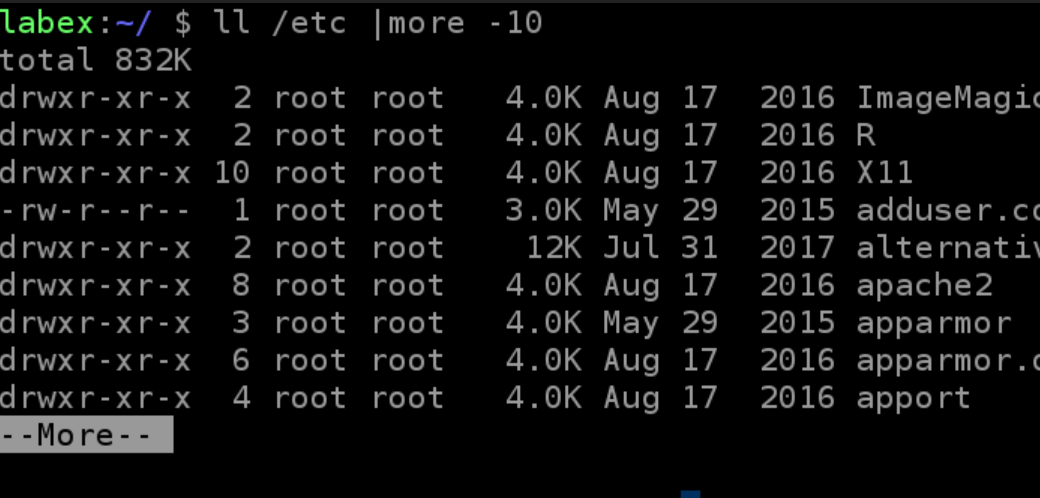
**Example 3:** To set the number of lines per screen to 5, use the following command:

more -5 labex.log



**Example 4:** Using the ll and more commands to display the /etc directory information, you can use the following command:

ll /etc | more -10



Each page shows 10 file information. Pressing Ctrl+F or Space will display the next 10 file information.

**Less:**

The less tool is also a tool for pagination of files or other output. It should be said that linux is an orthodox tool for viewing the content of files and is extremely powerful.

**(1) Command Format**

less [option] file

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -e | When the file is displayed, it leaves automatically |
| -f | Forces the opening of special files, such as peripheral code numbers, directories, and binary files. When the file is displayed, it leaves automatically |
| -i | Ignore case when searching |
| -m | Shows a percentage like the more command |
| -N | Display the line number of each line |
| -s | Display continuous empty behavior |

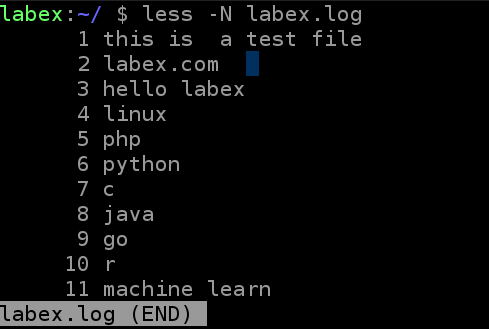
**(3) Common operations**

| **symbol** | **Description** |
| --- | --- |
| /string | Search for "string" down |
| ?string | Search for "string" upwards |
| n | Repeat previous search (related to / or ?) |
| N | Reverse the previous search (related to / or ?) |
| b | Turn a page forward |
| d | Turn half a page backward |
| q | Exit less command |
| space bar | Turn back one page |
| Up arrow | Flip up one line |
| Down arrows | Flip down one line |

**(4) Common Examples**

**Example 1:** Display the contents of the labex.log file and display the line number. You can use the following command:

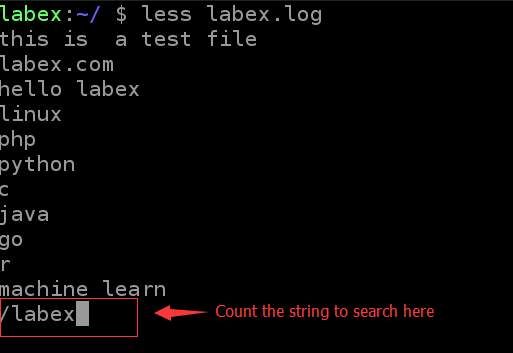
less -N labex.log



**Example 2:** Display the contents of the labex.log file, the search string "labex", using the following command:

less labex.log

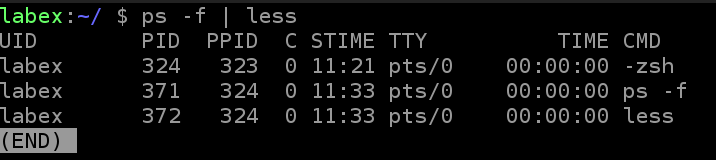
/labex





**Example 3:** ps checks the process information and displays it through less pages. You can use the following command:

ps -f | less



**Difference between less,cat and more:**

Cat command function: used to display the contents of the entire file, used alone without page turning function. Therefore, it is often used with the more command. The cat command also has the function of combining several files into one file.

More Command Function: Allows the screen to pause when a full page is displayed. Press the space bar to continue to display the next screen, or press the q button to stop the display.

The less command function: The use of the less command is similar to the more command. It can also be used to browse more than one page of files. The difference is that the less command can use the up and down keys to scroll through files, in addition to pressing the space bar to display files down. When you want to end browsing, just press q under the ":" prompt of the less command.

In fact, the three commands except the cat command have the function of merging files, the rest of the functions are similar, but they are different from the browsing habits and display methods.

**Head Command:**

The head command is as easy to understand as its name is. It is mainly used to display the beginning of the file to the standard output. The default head command prints the first 10 lines of its corresponding file.

**(1) Command Format**

head [option](https://labex.io/lab/file)

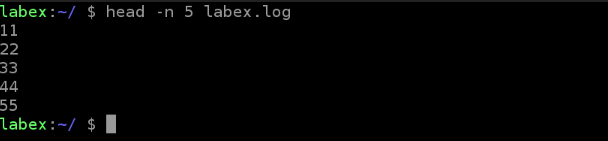
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -q | Hidden file name |
| -v | Display file name |
| -c | Display bytes |
| -n | The number of rows displayed |

**(3) Common Examples**

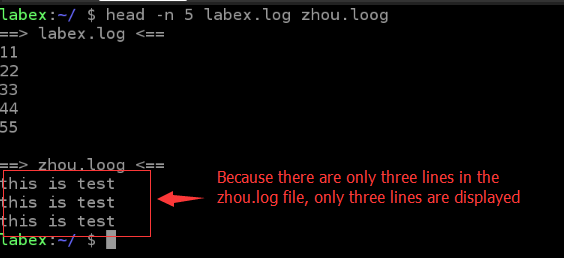
**Example 1:** To display the first 5 lines of the lqbex.log file, use the following command:

head -n 5 labex.log



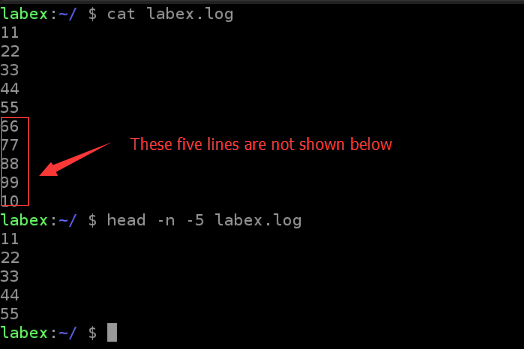
**Example 2:** To display the first 5 lines of the labex.log and zhou.log files, use the following command:

head -n 5 labex.log zhou.log



**Example 3:** To display the contents of the last 5 lines in the labex.log file, use the following command:

head -n -5 labex.log



**Tail command:**

The tail command is mainly used to display the contents of the specified file. Commonly used to view log files.

**(1) Command Format**

tail [option](https://labex.io/lab/file)

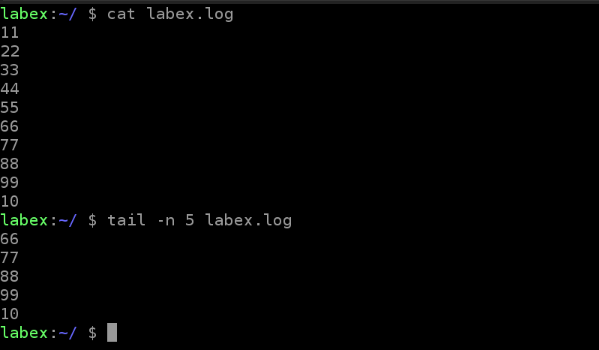
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -f | Loop reading |
| -q | Do not display processing information |
| -v | Display detailed processing information |
| -c | Display bytes |
| -n | The number of rows displayed |

**(3) Common Examples**

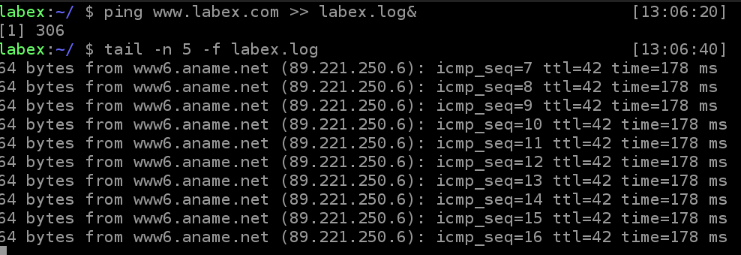
**Example 1:** To display the last 5 lines of the labex.log file, use the following command:

tail -n 5 labex.log

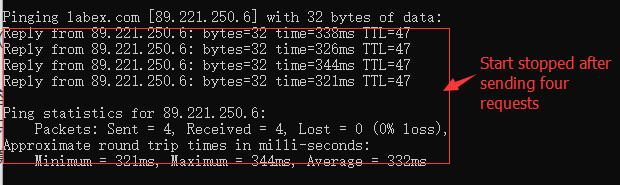


**Example 2:** The last 5 lines of the labex.log file are displayed. When new content is added to the labex.log file, it is updated automatically. You can use the following command:

tail -n 5 -f labex.log



Ping [www.labex.com](http://www.labex.com/) >> labex.log This command pings the remote host and appends the information to the labex.log file. The effect of & is to place this command in the background so that the labex.log file will always have more content. Note that the ping command under Linux will always be executed and must be stopped manually. When the ping command is executed under Windows, it will stop automatically after sending a certain request.



Use the -f option of the tail command to output the contents of the file immediately after the file changes. tail -f filename will display the contents of the end of filename on the screen, and not only refresh, allowing you to see the latest file content.Also by the way, how do you view the tasks that are running in the background and how to stop them?The jobs command can see the tasks that are running in the background. The kill command can kill a task but use the task's id. The id of the task can be obtained by viewing the ps command.

**Which Command:**

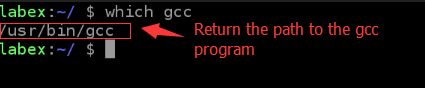
The role of the which command is to search for the location of the executable file in the path specified by the PATH variable. It is generally used to confirm whether the specified software is installed in the system.

**(1) Command Format**

which executable file name

**(2) Common Examples** **Example 1:** To confirm whether gcc is installed, use the following command:

which gcc



**Example 2:** To view the path of the cd command, use the following command:

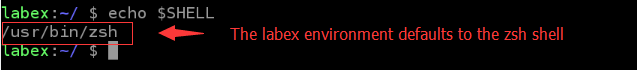
which cd

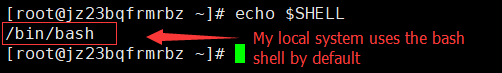
From the above display information you can see in the labex system cd is a shell function.

Then execute the which cd command on my local linux system. The result is as follows:

There is no cd command displayed in the path specified by the PATH variable. Why is it not displayed? This is because cd is a shell built-in command, and which by default finds the directory specified in PATH. The built-in command is not in its specified directory, so of course it must not be found.

Then there is a problem, why cd is a shell function in the labex environment, and cd is a built-in command in other linux environment? This is related to the bash used.





The echo $SHELL command looks at the currently used shell.

**What exactly is an order?**

The command can be one of the following four forms:

1. Is an executable program, just like the file we have seen in the directory /usr/bin. Programs that fall into this category can be compiled into binary files, such as those written in C and C++, or programs written in scripting languages, such as shell, perl, python, ruby, and so on.
2. Is a command built into the shell itself. Bash supports several commands, internally called builtins. For example, the cd command in my local environment above is a shell internal command.
3. Is a shell function. These are small-scale shell scripts that are mixed into environment variables. For example, the cd command mentioned above is a shell function in the lab environment.
4. Is a command alias. We can define our own commands and build on other commands. The echo $SHELL command looks at the currently used shell.

**Whereis Command:**

The whereis command is mainly used to locate the executable file, source code file, and help file in the file system. The whereis command also has the ability to search for source code, specify an alternate search path, and search for unusual items.

The whereis command looks up very quickly because it isn't just looking around randomly on disk, but in a database (/var/lib/mlocate/). This database is automatically created by the Linux system, contains information on all local files, and is updated once every day by automatically executing the updatedb command. It is precisely because this database is updated once a day, it will make the search results of whereis command sometimes inaccurate, such as just added files may not be found.

**(1) Command Format**

whereis [options] file

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -b | Targeting executables |
| -m | Locate help file |
| -s | Locating source code files |
| -u | Searching for files other than executables, source files, and help files in the default path |
| -B | Specify the path to the search executable |
| -M | Specify the path to the search help file |
| -S | Specify the path to the search source file |

**(3) Common Examples**

**Example** 1: Search for the path to the gcc executable. You can use the following command:

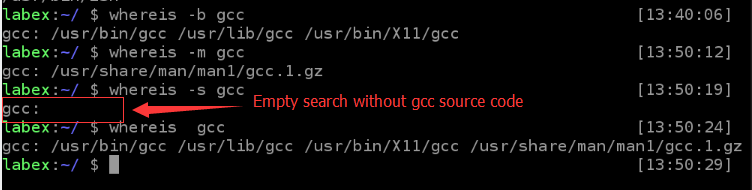
whereis -b gcc

Example\*\* 2: Search for the path to the gcc help file. You can use the following command:

whereis -m gcc

**Example 3:** Search for the path to the gcc source code. You can use the following command:

whereis -s gcc



**Locate Command:**

The locate command is similar to the whereis command and they use the same database. However, the whereis command can only search for executable files, online help files, and source code files. If you want more comprehensive search results, you can use the locate command.

The locate command uses a very complex matching syntax that uses special characters (such as '\*' and '?') to specify which samples to look for.

**(1) Command Format**

locate [option](Search string)

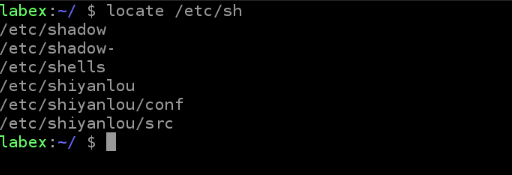
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -q | Quiet mode, no error message is displayed |
| -n | Display at most n outputs |
| -r | Use regular expressions to do the search |
| -V | Display version message |

**(3) Common Examples**

**Example 1:** To search for all files beginning with sh in the etc directory, use the following command:

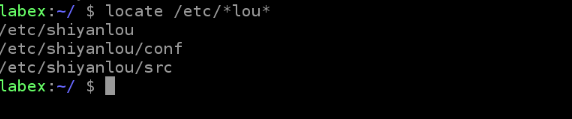
locate /etc/sh



Note that executing this command in the labex environment may result in an error: locate cannot be executed stat() ‘/var/lib/mlocate/mlocate.db’: There is no such file or directory. The solution is to execute the sudo updatedb command.

**Example** 2: Search for the file named lou in the etc directory. You can use the following command:

locate /etc/**\*lab\***



**Find Command:**

The main function of the find command is to traverse down the file hierarchy, match the files that meet the conditions, and perform the corresponding operations. Under the Linux find command provides a lot of search conditions, the function is very powerful, the corresponding learning is also more difficult.

**(1) Command Format**

find option [expression]

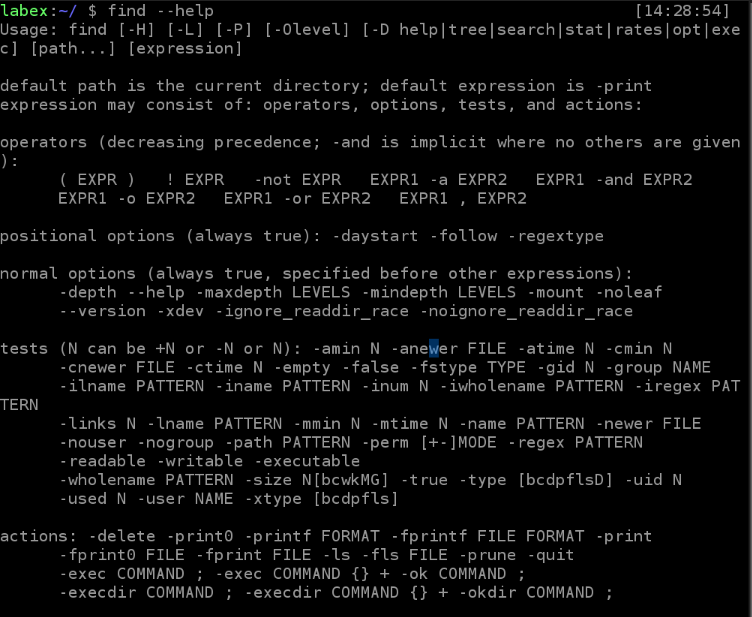
The default path is the current directory and the default expression is -print.

The expression may consist of the following components: operators, options, test expressions, and actions.

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -print | Find command outputs matching files to standard output |
| -exec | The find command executes the shell command given for the matching file |
| -name | Find files by file name |
| -type | Finding a type of file |
| -prune | Using this option to make the find command not lookup in the currently specified directory. If you use the -depth option, -prune will be ignored by the find command. |
| -user | Find files by owner of the file |
| -group | Find files by the group they belong to |
| -mtime -n +n | Find the file according to the file's change time, -n means the file change time is less than n days, +n means the file change time is now greater than n days, the find command also has -atime and -ctime options |

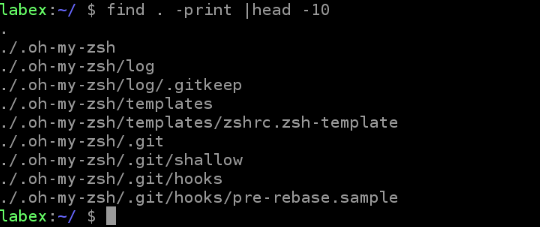
Other commands Command options and expressions can be seen in the figure below.



**(3) Common Examples**

**Example 1:** To print a list of file directories in the current directory, use the following command(Only the first 10 rows are displayed):

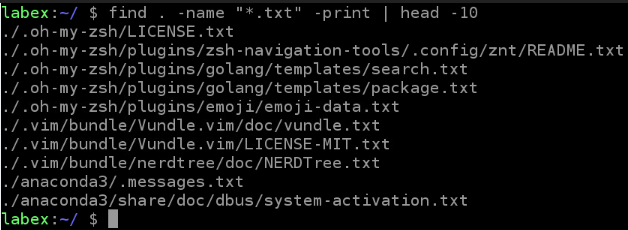
find . -print | head -10



**Example 2:** Print all the file names ending in .txt in the current directory, you can use the following command (the screenshot shows only part):

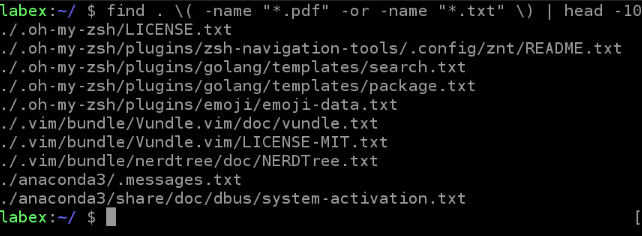
find . -name "\*.txt" -print | head -10

The -iname option works the same as the -name option except that -iname ignores case letters.



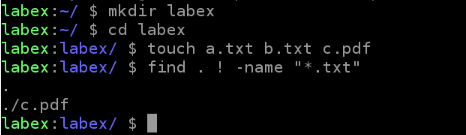
**Example 3:** To print all file names ending with .txt or .pdf in the current directory, use the following command (screenshot shows only part):

find . \( -name "\*.pdf" -or -name "\*.txt" \)



**Example 4:** Print all file names in the current directory that do not end with .txt. You can use the following command:

find . ! -name "\*.txt"



[N](javascript:;)

Here's how to find matching files based on file type, permissions, owner, and operator.

According to the file type to find the file, use the -type option, the common find file type is shown in the following table.

| **File type** | **Description** |
| --- | --- |
| b | Block device file |
| c | Character device file |
| d | Directory |
| f | Ordinary file |
| l | Symbolic link |

Find files based on file permissions using the -perm option. The owner uses the -user option.

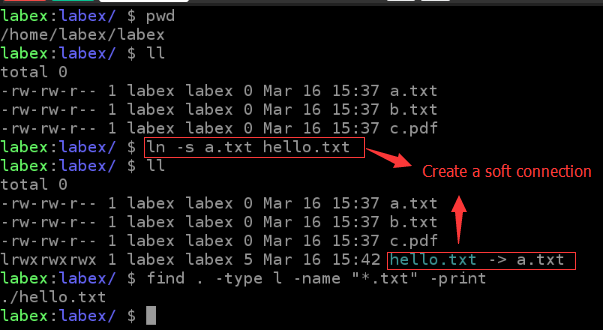
In addition, the find command can create more complex logical relations through logical operators. For example, the operator -or is used in Example 3 of the find command (1). The logical operators for the find command are shown in the following table.

| **Operators** | **Description** |
| --- | --- |
| -and | Matches if the test files on both sides of the operator are true. Can be abbreviated as -a. Note that -and is used by default if no operator is used |
| -or | Match if any of the test conditions on both sides of the operator are true. Can be abbreviated as -o |
| -not | Match if the test condition following the operator is false. Can be abbreviated as an exclamation point (!) |
| () | Combining test conditions and operators to form larger expressions. This will always control the logic calculation priority. By default, find commands are evaluated from left to right. It is often necessary to override the default job search order to get the desired result. Even if not necessary, sometimes including the combined characters is helpful for improving the readability of the command. Note that because parentheses have special meaning to the shell, when using them on the command line, they must be enclosed in parentheses before they can be passed as arguments to the find command. Usually backslash characters are used to escape parentheses |

**(1) Common Examples**

**Example 1:** To print all symbolic links ending in .txt in the current directory, use the following command:

find . -type l -name "\*.txt" -print



**Example 2:** Print all php files with permissions of 777 in the current directory (php files on web servers generally need execute permission). You can use the following command:

find . -type f -name "\*.php" -perm 777

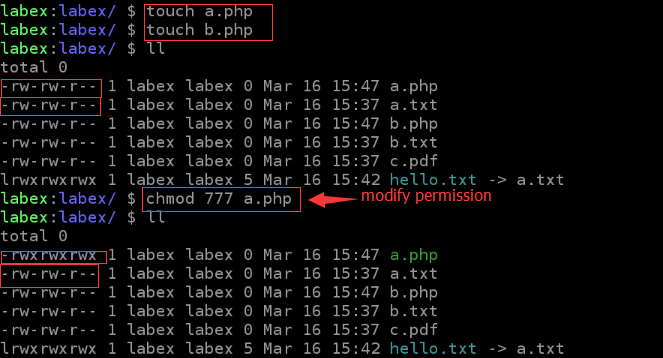
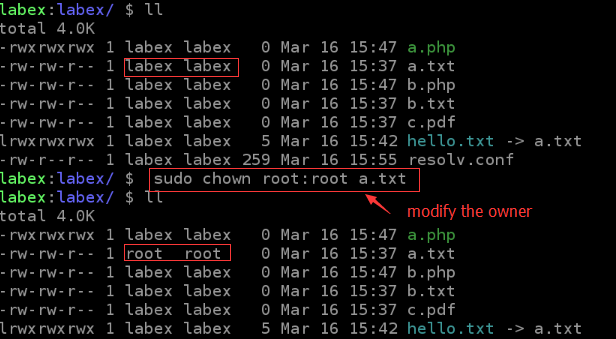
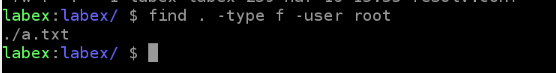


image desc

**Example** 3: To print all files owned by the root user in the current directory, use the following command:

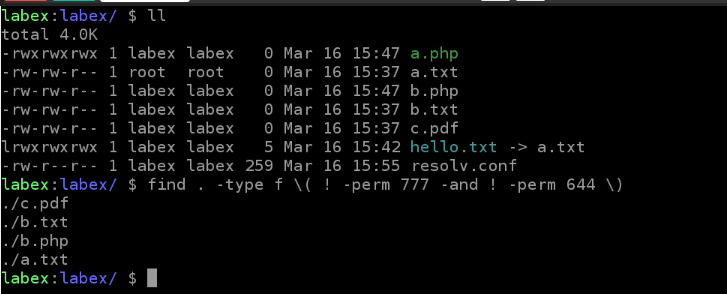
find . -type f -user root





**Example 4:** Print all files whose permissions under the current directory are not 777 and 664. You can use the following command:

find . -type f \( ! -perm 777 -and ! -perm 644 \)



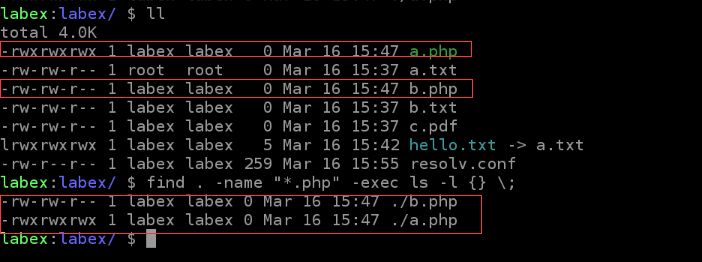
[N](javascript:;)

The following describes how to use the -exec option of the find command to perform the specified action command on the found file.

**(1) Common Examples**

**Example 1:** To find all the php files in the current directory and display their detailed information, you can use the following command:

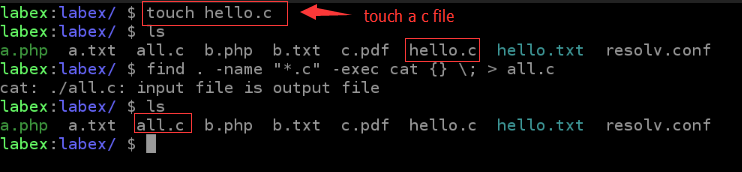
find . -name "\*.php" -exec ls -l {} \;



-exec is the action that the find command performs on the found file. This action is to execute the command on the found file. The above command is ls -l {}. Here to explain {} and ;, {} It is actually a placeholder, in the implementation of the find command will continue to be replaced by the currently found file, which is equivalent to "ls -l found file". And ; is the end of the -exec command, because the command behind -exec must end with ; but; has a special meaning in the shell and must be escaped, so written as ;.

**Example 2:** In the labex environment, there are many c language code files written by ourselves. Now we want to download all the c language code files. If one or the other is very troublesome, we can find all the c language code files first, and then Write the contents of these files to a file, download the file, you can use the following command:

find . -name "\*.c" -exec cat {} \; > all.c



**Example 3:** By default, only one command can be used after -exec. If you want multiple commands, you can write multiple commands to the script file. Then use this script in -exec. You can use the following command:

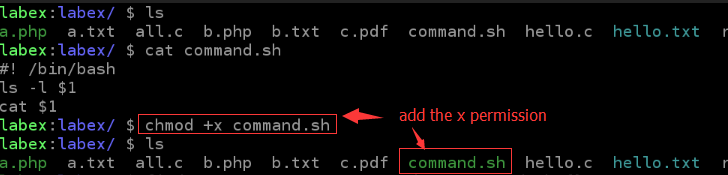
find . -name "\*.c" -exec ./command.sh {} \;

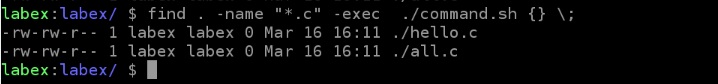
Script content:

#！ /bin/bash

ls -l $1 # $1 Indicates the first parameter

cat $1





[N](javascript:;)

**Xargs Command:**

We can use pipe to redirect stdout (standard output) of one command to stdin (standard input) of another command. However, some commands can only receive data in the form of command line parameters and cannot receive data streams through stdin. In this case, data cannot be redirected to these commands through pipes.

In this case, xargs can play its role. The xargs command can receive input from standard input and convert the input to a specific parameter list.

**(1) Command Format**

command | xargs [Options](https://labex.io/lab/command)

The xargs command should immediately follow the pipeline operator because it uses the standard input as the primary source data stream.

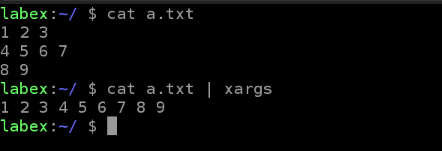
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -n | Specify the maximum number of parameters per row |
| -d | Specify separator |

**(3) Common Examples**

**Example 1:** Converting multi-line input to single-line output can use the following command:

cat a.txt | xargs



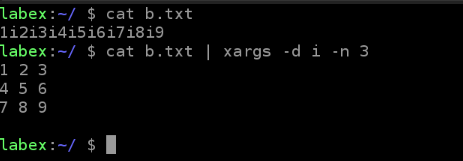
**Example 2:** To convert a single-line input to multiple lines of output, use the following command:

echo "1 2 3 4 5 6 7" | xargs -n 3



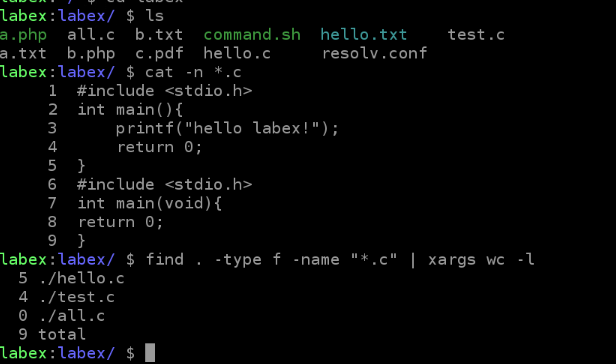
**Example 3:** Convert single-line input to multi-line output. Specify the separator as i. You can use the following command:

cat b.txt | xargs -d i -n 3



**Example 4:** Find all c code files in the current directory and count the total number of lines. You can use the following command:

find . -type f -name "\*.c" | xargs wc -l



**WC Command:**

The wc command is a statistical tool that is used to display the number of lines, words, and bytes contained in a file.

The wc command is an abbreviation for word count.

**(1) Command Format**

wc [option](https://labex.io/lab/file)

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -c | Statistics bytes |
| -l | Statistics rows |
| -m | Number of characters counted. This flag cannot be used with the -c flag |
| -w | The number of statistical words, a word is defined as a string separated by blanks, tabs or newline characters |
| -L | Print the length of the longest line |

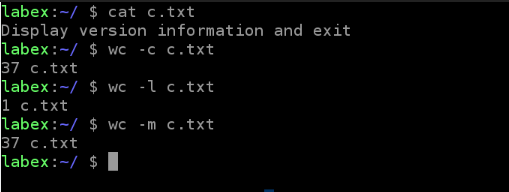
**(3) Common Examples**

**Example 1:** Count the number of bytes, lines, and characters of a file. You can use the following command:

wc -c c.txt

wc -l c.txt

wc -m c.txt



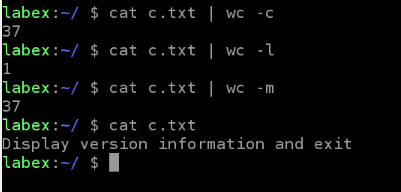
One line break per line here is also a character, the word (32) + space (4) + line break (1) = 37.

**Example 2:** Count the number of bytes, lines, and characters of a file, print only numbers, and do not print the file name. You can use the following command:

cat c.txt | wc -c

cat c.txt | wc -l

cat c.txt | wc -m



**Example** 3: To count the number of commands in the /bin directory, use the following command:

ls /bin | wc -l

**Grep command:**

Grep is a very powerful command for finding matching text in a file, accepting regular expressions and wildcards, and using multiple grep command options to generate output in various formats.

Grep works like this, searching for a string template in one or more files. If the template includes spaces, it must be referenced and all strings after the template are treated as file names. The search results are sent to the standard output without affecting the original file content.

Grep can be used for shell scripts because grep states the status of the search by returning a status value, 0 if the template search succeeded, 1 if the search was unsuccessful, and 2 if the searched file did not exist. We can use these return values to perform some automated text processing.

**(1) Command Format**

grep [options] pattern [file]

**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -c | Calculate the number of 'search string' (ie pattern) found |
| -i | Ignore case differences, so case is considered the same |
| -n | Output line number |
| -v | Reverse selection, print mismatched lines |
| -r | Recursive search |
| --color=anto | Add a color display to the found keyword section |

**(3) Common Examples**

**Example 1:** Take out the root line in the /etc/passwd file, and add the color to the keyword. You can use the following command:

grep "root" /etc/passwd --color=auto

cat /etc/passwd | grep "root" --color=auto



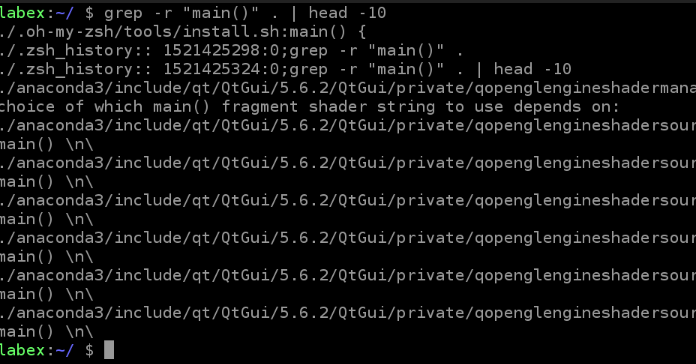
**Example 2:** To remove the root and nologin lines from the /etc/passwd file, use the following command:

grep -v "root" /etc/passwd | grep -v "nologin"



**Example 3:** In the current directory recursive search file contains main () file, often used to find some function in which source code file, you can use the following command:

grep -r "main()". head -10



**Regular expressions and grep commands:**

A regular expression is a symbolic representation that is used to identify text patterns. To some extent, they are similar to shell wildcards that match file and path names, but they are larger. Many command-line tools and most programming languages support regular expressions to help solve text manipulation problems.

Regular expression metacharacters consist of the following characters:

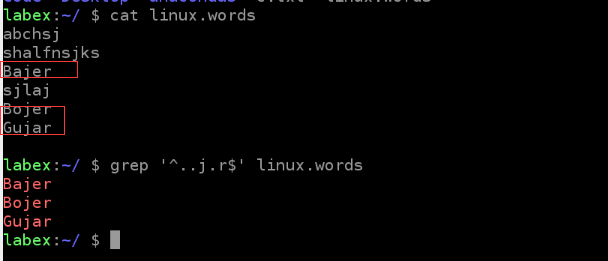
^ $ . [ ] { } - ? \* + ( ) | \

| **Regular expression** | **Description** | **Example** |
| --- | --- | --- |
| ^ | Line start marker | ^labex matches rows starting with labex |
| $ | End of line marker | $labex matches rows ending with labex |
| . | Match any character | a.c matches abc, aac but does not match abbc |
| [] | Match any character contained in [Character] | ab[cd] matches abc or abd |
| [^] | Match any character except [ ^ character] | 1[^01] matches 12,13 but does not match 10,11 |
| [-] | Match any character within the specified range | [1-5] Match any number from 1 to 5 |
| {n} | Match the previous item n times | [0-9]{2} matches any two-digit number, which is equivalent to [0-9][0-9] |
| {n,} | The previous item needs to match at least n times | [0-9]{2,} matches a digit with two or more digits |
| {n,m} | Specify the minimum and maximum number of times the previous item needs to match | [0-9]{2,5} matches any number from two digits to five digits |
| ? | Match the previous item 1 or 0 times | lab?ex matches labex, laex |
| \* | Match the previous item 0 or more times | lab\*ex matches laex, labex, labeex |
| + | Match the previous item one or more times | lab+ex matches labex, labbex |
| () | Create a substring for matching | Ma(in)? matches ma or main |
| | | Match | Any of the two sides | Dec(1st | 2nd) matches Dec 1st or Dec 2nd |
| \ | Escape special characters above | a\+b matches a+b |

**(1) Common Examples**

**Example 1:** Use the Linux system's own dictionary to find a five-letter word, the third letter is j, the last letter is r, /usr/share/dict directory to store the dictionary file,If you do not have a dictionary, you can create your own test file, you can use the following command:

grep '^..j.r$' linux.words



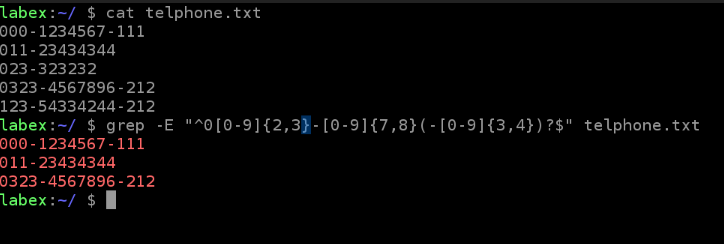
Example 2: Verify the fixed phone and print the eligible phone. The format of the fixed phone is basically the area code with 0 + connector "-" + phone number, and there may be an extension number, and the area code has 3, 4 digits. The phone number has 7 and 8 digits, you can use the following command:

grep "^0[0-9]{2,3}-[0-9]{7,8}(-[0-9]{3,4})?$" telphone.txt

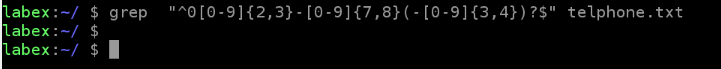
Area code: preceding 0, followed by 2-3 digits: 0[0-9]{2,3}

Phone number: 7-8 digits: [0-9]{7,8}

Extension number: Generally 3-4 digits: [0-9]{3,4}



Note that there is no matching output when executing the following command. This is because without the -E option, why isn't this case? Why? This is because grep treats it as a shell wildcard, not a regular expression metacharacter.



**Cut command:**

The cut command is a gadget that splits the text by columns, and it can specify the delimiter that separates each column.

If a row of data contains more than one field (multiple columns) and you now want to extract one or more of them, this is the cut command.

**(1) Command Format**

cut [option](file name)

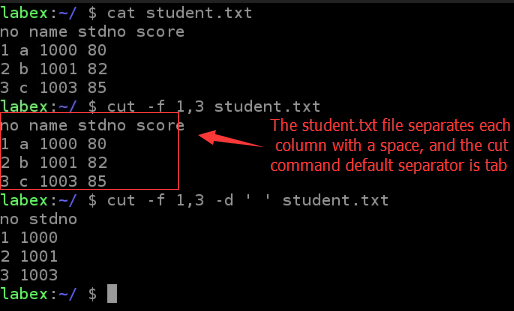
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -b | Split in bytes |
| -c | Split by character |
| -d | Custom separator, defaults to tabs |
| -f | Custom fields |
| --complement | Extract entire text lines except those specified by the -c or -f options |

**(3) Common Examples**

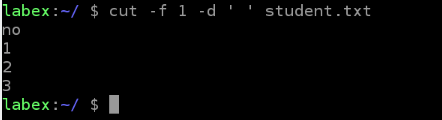
**Example 1:** To remove the first and third columns in the student.txt file, use the following command:

cut -f 1,3 -d ' ' student.txt



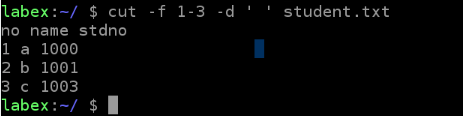
**Example 2:** To remove the first column in the student.txt file, use the following command:

cut -f 1 -d ' ' student.txt



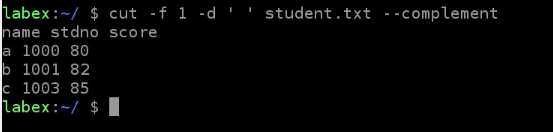
**Example 3:** To remove the first three columns in the student.txt file, use the following command:

cut -f 1-3 -d ' ' student.txt



**Example 4:** Remove the other columns in the student.txt file except for the first column, you can use the following command:

cut -f 1 -d ' ' student.txt --complement



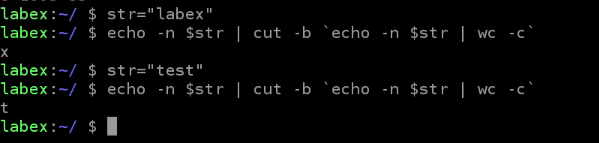
**Example 5:** Give any string str, remove its last character, you can use the following command:

num=$(echo -n $str | wc -c)

echo -n $str | cut -b $num

# or

echo -n $str | cut -b `echo -n $str | wc -c`



**Paste command:**

The function of the paste command is exactly the opposite of cut. Instead of extracting text columns from the file, it adds one or more text columns to the file. It reads multiple files and then combines the fields in each file into a single text stream that is input to the standard output.

(**1) Command Format**

paste [option](file name)

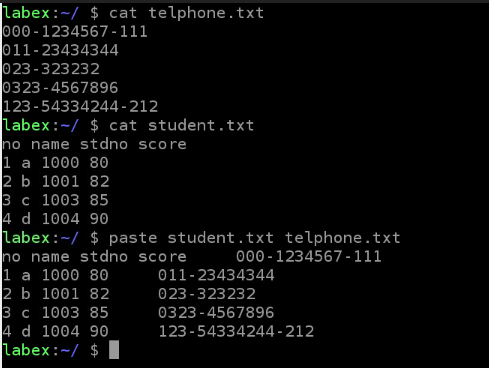
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -s | Merge each file into rows instead of lines |
| -d | Custom separator, defaults to tabs |

**(3) Common Examples**

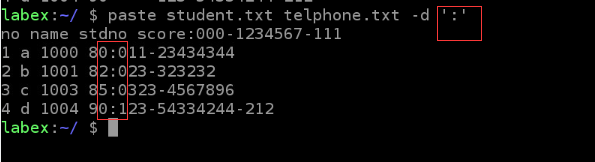
**Example 1:** Splicing the contents of student.txt and telphone.txt files by column, you can use the following command:

paste student.txt telphone.txt



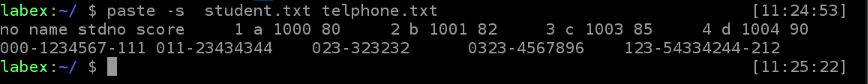
**Example 2:** Splicing the contents of the student.txt and telphone.txt files by columns. Specify the separator as ':'. You can use the following command:

paste student.txt telphone.txt -d ':'



**Example 3:** Splicing the contents of the student.txt and telphone.txt files into a single line, you can use the following command:

paste -s student.txt telphone.txt



**TR command:**

The tr command is often used to change characters. We can think of it as a character-based search and replace operation. Word change is the process of converting characters from one letter to another. Tr Replaces, reduces, and/or deletes characters from standard input and writes the result to standard output.

Tr can only pass stdin (standard input) and cannot accept input via command line arguments.

Tr is the abbreviation for translate.

**(1) Command Format**

Tr [options] SET1 SET2

Map input characters from stdin from SET1 to SET2 and write their output to stdout (standard output). SET1 and SET2 are character classes or character sets. If the lengths of the two character sets are not equal, then SET2 will continue to repeat its last character until it has the same length as SET1. If the length of SET2 is greater than SET1, all the characters that exceed SET1 in SET2 are ignored.

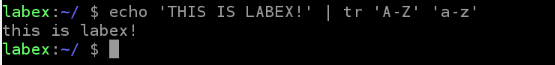
**(2) Common Parameters**

| **Parameter** | **Description** |
| --- | --- |
| -d | Delete content that matches SET1 and do not replace |

**(3) Common Examples**

**Example 1:** To convert the input characters to lowercase, use the following command:

echo 'THIS IS SHIYANLOU!' | tr 'A-Z' 'a-z'



**Example 2:** To delete the numbers in the input characters, use the following command:

echo 'THIS 123 IS S1HIY5ANLOU!' | tr -d '0-9'

image desc

**Example 3:** An interesting use of the tr command is to perform ROT13 text encoding. ROT13 is a trivial type of encryption based on a simple replacement password. Calling ROT13 "encryption" is generous, and "text blurring" is more accurate. Sometimes it is used to hide potential attack content in text. This method is simply to move each character forward by 13 in the alphabet. Because the number of bits moved is half of the possible 26 characters, executing this algorithm again on the text returns to its original form. You can use the following command:

#encryption

echo 'shiyanlou' | tr 'a-zA-Z' 'n-za-mN-ZA-M'

fuvlnaybh

#Decryption

echo 'fuvlnaybh' | tr 'a-zA-Z' 'n-za-mN-ZA-M'

shiyanlou

