“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної та програмної інженерії

ЗВІТ ПО ВИКОНАННЮ

ЛАБОРАТОРНОЇ РОБОТИ №6

з дисципліни: «Операційні системи»

Тема: “Команди Linux для архівування та стиснення даних. Робота з текстом”

Виконали

студенти

групи РПЗ-03

Команда 6: Sichkar Maxim,

Brytyuk Bohdan

Kyiv 2023

**The goal of the work:**

1. Getting practical skills for working with the Bash command shell.

2. Acquaintance with basic commands for data archiving and compression.

3. Getting to know the basic actions when working with text in the terminal.

**Material support to occupy:**

1. IBM PC type computer.

2. OS family Windows (Windows 7).

3. Virtual machine - Virtual Box (Oracle).

4. GNU/Linux operating system - CentOS.

5. Cisco network academy site netacad.com and its online Linux courses

**Tasks for preliminary preparation.**

1. Read the short theoretical information for the laboratory work and make a small dictionary

basic English terms for command assignments and their parameters.

2. Study the materials of the online course of the Cisco Academy “NDG Linux Essentials”:

- Chapter 09 - Archiving and Compression

- Chapter 10 - Working With Text

3. Take the test in the NDG Linux Essentials course on the following topics:

- Chapter 09 Exam

- Midterm Exam (Modules 1 - 9) will be a separate task in the Google class

- Chapter 10 Exam

4. On the basis of the considered material, answer the following questions:

**4.1. What is the purpose of tar, xz, zip, bzip, gzip commands? Make a brief description of each team and highlight their main parameters. How to install them.**

Here's a brief description of each command and their main parameters:

tar: This command is used for creating and extracting archive files. The main parameters used with this command are:

c: Create a new archive file.

x: Extract an archive file.

v: Verbose mode - display the progress of the command.

f: Specify the file name of the archive.

To install tar, you can use the following command:

For Ubuntu and Debian: sudo apt-get install tar

For CentOS and Fedora: sudo yum install tar

xz: This command is used for compressing and decompressing files. It uses the LZMA compression algorithm to achieve a high compression ratio. The main parameters used with this command are:

c: Compress the file.

d: Decompress the file.

v: Verbose mode - display the progress of the command.

To install xz, you can use the following command:

For Ubuntu and Debian: sudo apt-get install xz-utils

For CentOS and Fedora: sudo yum install xz

zip: This command is used for compressing and archiving files. It uses the ZIP compression algorithm and is widely used on Windows systems. The main parameters used with this command are:

r: Recursively compress files in a directory.

v: Verbose mode - display the progress of the command.

To install zip, you can use the following command:

For Ubuntu and Debian: sudo apt-get install zip

For CentOS and Fedora: sudo yum install zip

bzip: This command is used for compressing and decompressing files. It uses the Burrows-Wheeler algorithm to achieve a high compression ratio. The main parameters used with this command are:

c: Compress the file.

d: Decompress the file.

v: Verbose mode - display the progress of the command.

To install bzip, you can use the following command:

For Ubuntu and Debian: sudo apt-get install bzip2

For CentOS and Fedora: sudo yum install bzip2

gzip: This command is used for compressing and decompressing files. It uses the DEFLATE algorithm to achieve a high compression ratio. The main parameters used with this command are:

c: Compress the file.

d: Decompress the file.

v: Verbose mode - display the progress of the command.

To install gzip, you can use the following command:

For Ubuntu and Debian: sudo apt-get install gzip

For CentOS and Fedora: sudo yum install gzip

**4.2. Give three examples of implementing data archiving and compression using different commands.**

Using tar and gzip to create a compressed archive:

To create a compressed archive of a directory, we can use the tar command along with the gzip command. Here's an example:

tar -czvf archive.tar.gz directory\_to\_archive

This will create an archive named "archive.tar.gz" that contains all the files and directories in the "directory\_to\_archive" directory. The "-c" flag is used to create a new archive, "-z" flag tells tar to use gzip for compression, "-v" flag displays the progress of the command, and "-f" flag specifies the file name of the archive.

Using zip to create a compressed archive:

To create a compressed archive of a directory using zip, we can use the following command:

zip -r archive.zip directory\_to\_archive

This will create an archive named "archive.zip" that contains all the files and directories in the "directory\_to\_archive" directory. The "-r" flag tells zip to recursively include all files and subdirectories in the archive.

Using bzip to create a compressed file:

To create a compressed file using bzip, we can use the following command:

bzip2 filename

This will create a compressed file named "filename.bz2". The compression ratio achieved by bzip2 is generally higher than gzip, but it takes more time to compress and decompress. To decompress the file, we can use the following command:

bzip2 -d filename.bz2

This will decompress the file and create a new file named "filename".

**4.3. What is the purpose of the commands cat, less, more, head and tail? Make a brief description of each command and highlight their main parameters. How to install them**

Here's a brief description of each command and their main parameters:

cat: This command is used to display the contents of a file. The main parameters used with this command are:

n: Display line numbers.

E: Display the end of each line with a dollar sign.

A: Display all characters, including control characters.

To install cat, you don't need to do anything, as it's a basic command that comes pre-installed in most Unix-like operating systems.

less: This command is used to display the contents of a file, but it allows you to navigate through the file using arrow keys. The main parameters used with this command are:

N: Display line numbers.

S: Disable line wrapping.

F: Follow the end of the file.

To install less, you can use the following command:

For Ubuntu and Debian: sudo apt-get install less

For CentOS and Fedora: sudo yum install less

more: This command is similar to less, but it displays the contents of the file one screen at a time, and you have to press the spacebar to display the next screen. The main parameters used with this command are:

N: Display line numbers.

S: Disable line wrapping.

To install more, you can use the following command:

For Ubuntu and Debian: sudo apt-get install more

For CentOS and Fedora: sudo yum install more

head: This command is used to display the first few lines of a file. The main parameters used with this command are:

n: Display a specific number of lines.

To install head, you don't need to do anything, as it's a basic command that comes pre-installed in most Unix-like operating systems.

tail: This command is used to display the last few lines of a file. The main parameters used with this command are:

n: Display a specific number of lines.

f: Follow the end of the file and display new lines as they are added.

To install tail, you don't need to do anything, as it's a basic command that comes pre-installed in most Unix-like operating systems.

**4.4. Explain how the command shell works with channels, streams, and filters**

The command shell is a program that provides a user interface for interacting with the operating system. It interprets the commands entered by the user and executes them by launching other programs or scripts. The shell uses three main concepts - channels, streams, and filters - to manipulate data.

Channels: Channels are pipelines that allow the output of one command to be used as the input of another command. The most common channel is the pipe symbol "|", which is used to connect two commands together. For example, the following command connects the output of the "ls" command to the input of the "grep" command:

ls | grep keyword

This will display a list of files and directories that contain the keyword "keyword".

Streams: Streams are channels of data that are used to communicate between programs. There are three streams: standard input (stdin), standard output (stdout), and standard error (stderr). Standard input is where the program receives input from the user, standard output is where the program sends its output, and standard error is where the program sends error messages. By default, all three streams are connected to the terminal.

Filters: Filters are commands that transform the data that passes through them. They take input from the standard input stream and produce output on the standard output stream. Some common filters include "grep" for searching for patterns in text, "sort" for sorting lines of text, "sed" for editing text, and "awk" for manipulating text data.

Using channels, streams, and filters together, we can construct complex commands that perform sophisticated operations on data. For example, the following command lists all the files in the current directory, sorts them alphabetically, removes duplicates, and then displays the result:

ls | sort | uniq | less

This command uses the pipe symbol to connect the output of one command to the input of the next command. The "ls" command lists all the files in the current directory and sends its output to the "sort" command, which sorts the files alphabetically. The "uniq" command removes duplicates from the list of files, and the "less" command displays the result on the screen one page at a time.

**4.5. What is the purpose of the grep command?**

The grep command is a very useful command-line tool that is used to search for specific text patterns in files or input streams. It stands for "Global Regular Expression Print". The main purpose of the grep command is to search for a particular pattern or regular expression (regex) in a given file or input stream and display the matching lines.

Here's a brief overview of the main features of the grep command:

Search for a specific pattern or regex in a file or input stream.

Display the matching lines that contain the pattern or regex.

Ignore case sensitivity with the -i option.

Use regular expressions to specify complex patterns to search for.

Display line numbers for matching lines with the -n option.

Search recursively through directories with the -r option.

Search for patterns in compressed files with the -z option.

The basic syntax of the grep command is:

grep [options] pattern [file...]

Here, "pattern" refers to the regular expression or text pattern to search for, and "file..." refers to the files to search in. If no files are specified, grep reads from the standard input.

For example, the following command searches for the word "example" in the file "file.txt":

grep example file.txt

The above command will display all lines in "file.txt" that contain the word "example".

The grep command is an extremely powerful tool that can be used for a variety of purposes such as searching for specific log messages, finding files that contain specific code snippets, and more.

5. Prepare the initial version of the report electronically:

- Title page, topic and purpose of the work

- Glossary of terms

- Answers to p. 4.1 and p. 4.5 from tasks for preliminary training

**Progress.**

1. Initial work in CLI mode in Linux OS of the Linux family:

1.1. Start the VirtualBox virtual machine, select CentOS and run it. Log in

under user: CentOS, password for login: reverse (if you run LR in 401 aud.) and run

terminal.

1.2. Start the Ubuntu\_PC virtual machine (if you are doing the LR tasks through the netacad academy)

1.3. Start your Linux family operating system (if you are working on your own PC and its

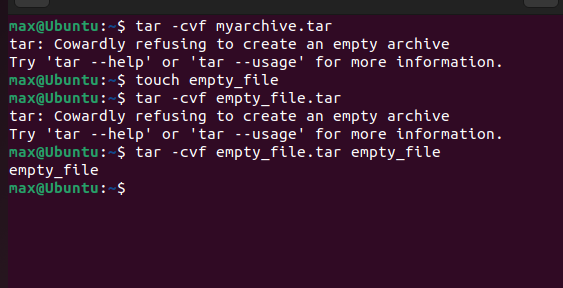
installed) and launch the terminal.

2. Work through all the command examples presented in the labs of the NDG Linux Essentials course -

Lab 9: Archiving and Compression and Lab 10: Working With Text. Create a table to describe these teams\*\*\*

|  |  |
| --- | --- |
| mkdir mybackups | Створення нової директорії mybackups у домашньому каталозі користувача |
| tar -cvf mybackups/udev.tar /etc/udev | Команда tar використовується для об’єднання кількох файлів в  /etc/udev один файл. В даному випадку вміст директорії /etc/udev буде  збережено в архів udev.tar у директорії mybackups. Параметр -c  повідомляє команді tar створити файл tar. Параметр -v означає  “verbose”, який наказує команді tar продемонструвати, що вона  робить. Параметр -f використовується для вказівки назви файлу tar. |

3. Familiarize yourself with the tar command and use it to perform the following actions in the terminal:

- create a file with the extension .tar; ****

- create a file with the .tar extension consisting of several files and directories at the same time;

****

- viewing the contents of the file; ****

- extract the contents of the tar file;

****

- create a tar archive compressed with bzip; ****

- extract the contents of the tar bzip file; ****

- create an archive tar file compressed with gzip; ****

- extract the contents of the tar gzip file. ****

4. How will the redirection of output streams in bash for subsequent operations with commands (marked as cmd) and files (marked as file):

|  |  |
| --- | --- |
| cmd 1> file | In Bash, the redirection of output streams can be used to redirect the output of a command or script to a file instead of printing it to the terminal. There are three types of output streams that can be redirected:  Standard output (stdout): This is the default output stream for a command or script. It contains the regular output of the program.  To redirect standard output to a file, use the following syntax:  cmd 1> file  or  cmd > file  This will execute "cmd" and redirect its standard output to the file named "file". If the file does not exist, it will be created. If it already exists, the existing contents will be overwritten.  Standard error (stderr): This stream contains error messages and warnings generated by a command or script.  To redirect standard error to a file, use the following syntax:  cmd 2> file  This will execute "cmd" and redirect its standard error to the file named "file". If the file does not exist, it will be created. If it already exists, the existing contents will be overwritten.  Standard input (stdin): This is the stream that provides input to a command or script.  Here are some examples to illustrate the redirection of output streams:  To execute the command "ls" and save its output to a file named "list.txt", use the following command:  ls > list.txt  To execute a command and save both the output and error messages to separate files, use the following command:  cmd > output.txt 2> errors.txt  To execute a command and discard its output, use the following command:  cmd > /dev/null |
| cmd > file |  |
| cmd 2> file |  |
| cmd >>file | cmd >>file: This command redirects the standard output of cmd to the end of file. If file does not exist, it is created. If file already exists, the output is appended to the end of the file instead of overwriting it. The >> operator is used for appending to a file. |
| cmd &>file | cmd &>file: This command redirects both standard output and standard error of cmd to the same file. If file does not exist, it is created. If file already exists, its content will be overwritten with the output of the command. The &> operator is used for redirecting both streams to a file. |
| cmd >file 2>&1 | cmd >file 2>&1: This command redirects the standard output of cmd to file, and then redirects the standard error stream to the same destination as the standard output, which is file. The > operator is used to redirect standard output to a file, and 2>&1 is used to redirect standard error to the same destination as standard output. |
| cmd >>file 2>&1 | cmd >>file 2>&1: This command redirects the standard output (stdout) of cmd to be appended to file, while also redirecting the standard error (stderr) stream to the same location. The >> operator is used to append the output to the file instead of overwriting it. The 2>&1 redirects the stderr stream to the same location as the stdout stream, which in this case is the file. |
| cmd 2 > /dev/null | cmd 2> /dev/null: This command redirects the standard error (stderr) stream of cmd to /dev/null, which is a special device file that discards all output sent to it. This is often used to silence error messages or other unwanted output that may appear in the stderr stream. |
| cmd 2 > /dev/null | cmd 2> /dev/null: This command is similar to the previous example, but it redirects both the stdout and stderr streams to /dev/null. This effectively silences all output from cmd. |
| cmd1 | cmd2 |  |
| cmd1 2>1 | cmd2 | The pipe symbol ("|") is used to redirect the output of one command to the input of another command. For example, the command "cmd1 | cmd2" pipes the output of "cmd1" as input to "cmd2". This means that the output of "cmd1" will not be displayed on the terminal, but instead, it will be sent to the input of "cmd2".  The "2>1" syntax is used to redirect the error output (stderr) of a command to the same destination as the standard output (stdout). For example, the command "cmd1 2>1 | cmd2" runs "cmd1", and redirects any error messages to the same destination as the standard output, which is then piped to "cmd2". This means that any error messages produced by "cmd1" will also be sent to the input of "cmd2" along with the standard output.  On the other hand, if we want to redirect the output of a command to a file, we can use the ">" symbol. For example, the command "cmd1 > file" redirects the output of "cmd1" to the file "file". This means that the output of "cmd1" will be saved to the file "file" instead of being displayed on the terminal.  Similarly, we can redirect the error output of a command to a file using the "2>" syntax. For example, the command "cmd1 2> file" redirects the error output of "cmd1" to the file "file". |

5. Consider the following examples and explain what these commands do and what type of thread redirection they use:

|  |  |
| --- | --- |
| $echo «It is a new story.”>story | The command "echo «It is a new story.”>story" redirects the output of the "echo" command to a file named "story". Specifically, the "echo" command outputs the string "It is a new story." to the standard output stream, which is then redirected to the file "story" using the ">" operator. This is an example of output redirection. |
| $ date > date.txt | The command "date > date.txt" redirects the output of the "date" command to a file named "date.txt". Specifically, the "date" command outputs the current date and time to the standard output stream, which is then redirected to the file "date.txt" using the ">" operator. This is also an example of output redirection. |
| $ cat file1 file2 file3 > bigfile | The command "cat file1 file2 file3 > bigfile" concatenates the contents of files "file1", "file2", and "file3" and redirects the output to a file named "bigfile". Specifically, the "cat" command outputs the contents of each file to the standard output stream, which is then redirected to the file "bigfile" using the ">" operator. This is an example of output redirection and file concatenation. |
| $ls -l >> directory | The command "ls -l >> directory" lists the contents of the current directory in long format and appends the output to a file named "directory". Specifically, the "ls" command outputs the contents of the current directory in long format to the standard output stream, which is then appended to the file "directory" using the ">>" operator. This is an example of output redirection using the append operator, which adds the output to the end of the file instead of overwriting it. |
| $ sort < file1\_unsorted > file2\_sorted | This command sorts the contents of file1\_unsorted and writes the sorted output to file2\_sorted. It uses input redirection (<) to read the contents of file1\_unsorted as input to the sort command, and output redirection (>) to write the sorted output to file2\_sorted. |
| $ find -name ‘\*.txt’> file.txt 2> /dev/null | This command searches for files in the current directory and its subdirectories with names that end in .txt, and writes the list of matching files to file.txt. It redirects the standard output of the find command to file.txt using output redirection (>), and redirects the standard error output of find to /dev/null (a special file that discards all data written to it) using the 2> notation. |
| $ cat file1\_unsorted | sort > file2\_sorted | This command reads the contents of file1\_unsorted using cat, pipes the output to the sort command, which sorts the input and writes the sorted output to file2\_sorted using output redirection (>). This command uses pipeline redirection (|) to pass the output of one command as input to the next command in the pipeline. |
| $ cat myfile | grep student | wc -l | This command reads the contents of myfile using cat, pipes the output to the grep command, which searches for lines that contain the word "student", and pipes the output of grep to the wc -l command, which counts the number of lines in the output. The command returns the number of lines that contain the word "student". This command uses pipeline redirection (|) to pass the output of one command as input to the next command in the pipeline. |

**Control questions**

1. **Give a comparative description of compression and archiving processes.**

Compression and archiving are two related but distinct processes that are often used together to manage large amounts of data.

Compression involves reducing the size of a file or a set of files by encoding the information in a more compact format that takes up less space. This is done by removing redundant information, encoding common patterns more efficiently, or using a more efficient encoding scheme. Compression is commonly used for files that need to be transmitted over the internet or stored on a limited amount of storage space, such as hard drives or flash drives.

Archiving, on the other hand, involves combining multiple files or directories into a single file, known as an archive, for the purpose of organizing and storing them. Archiving is commonly used for files that need to be backed up, transferred to another system, or saved for long-term storage. The archive file typically contains all of the files in their original format, along with additional information about the files such as timestamps, permissions, and directory structures.

Compression and archiving can be used together to create a compressed archive, which combines the benefits of both processes. A compressed archive is a single file that contains multiple files or directories that have been compressed to reduce their size. This makes it easier to transfer or store large amounts of data, while still retaining the original files and directory structures.

1. **What programs, in addition to those listed in the work, can be used for compression and archiving files and directories in the Linux OS? Give examples and their brief description.**

7-Zip: 7-Zip is a free and open-source file archiver that supports a wide range of compression formats including 7z, ZIP, RAR, and TAR.

WinRAR: WinRAR is a popular file archiver for Windows that supports a wide range of compression formats including RAR, ZIP, and TAR.

WinZip: WinZip is a popular file archiver for Windows that supports a wide range of compression formats including ZIP, RAR, and TAR.

PeaZip: PeaZip is a free and open-source file archiver that supports a wide range of compression formats including 7z, ZIP, RAR, and TAR.

gzip: gzip is a free and open-source compression utility for Unix-based systems that is commonly used to compress and decompress files in the gzip format.

bzip2: bzip2 is a free and open-source compression utility for Unix-based systems that is commonly used to compress and decompress files in the bzip2 format.

1. **Compare the compression algorithms used in the commands (programs) used in Linux. Which of the algorithms can be considered the fastest and most efficient?**

gzip: gzip uses the DEFLATE compression algorithm, which is a combination of LZ77 and Huffman coding. It is fast and efficient, with a good compression ratio for text-based files. However, it may not be as efficient for compressing binary files or files with already compressed data.

bzip2: bzip2 uses the Burrows-Wheeler block sorting algorithm and the move-to-front transform, followed by Huffman coding. It has a better compression ratio than gzip, but is slower and uses more CPU resources.

xz: xz uses the LZMA2 compression algorithm, which is a variant of LZ77 with improved compression ratios. It is slower than gzip and bzip2, but provides much better compression ratios and is more efficient in compressing large files.

7zip: 7zip uses the LZMA and LZMA2 compression algorithms, which are similar to the LZ77 algorithm used by xz. It provides better compression ratios than xz, but is slower and uses more memory and CPU resources.

zip: zip uses the DEFLATE compression algorithm, similar to gzip. It is fast and efficient for text-based files, but may not be as efficient for compressing binary files.

In terms of speed and efficiency, gzip can be considered the fastest and most efficient algorithm for compressing text-based files. However, for large files or files with complex data, xz provides the best compression ratios and efficiency. Ultimately, the choice of compression algorithm depends on the specific requirements of the task at hand, such as file size, compression ratio, and processing time.

**4. Describe the compression and archiving software tools that might be used in yours**

**mobile phone.**

WinZip: WinZip is a popular compression and archiving tool that is available for both Android and iOS devices. It supports a variety of compression formats, including ZIP, RAR, 7Z, and GZIP. It also provides encryption and file sharing capabilities.

ZArchiver: ZArchiver is a free and open-source archiving and compression tool for Android devices. It supports a variety of compression formats, including ZIP, RAR, 7Z, TAR, and GZIP. It also allows users to create and extract password-protected archives.

iZip: iZip is a compression and archiving tool for iOS devices. It supports a variety of compression formats, including ZIP, RAR, 7Z, and GZIP. It also provides encryption and file sharing capabilities.

RAR: RAR is a popular compression and archiving tool that is available for both Android and iOS devices. It supports a variety of compression formats, including its own RAR format, as well as ZIP, GZIP, and TAR. It provides strong encryption and can split archives into multiple volumes.

AndroZip: AndroZip is a free and open-source archiving and compression tool for Android devices. It supports a variety of compression formats, including ZIP, RAR, 7Z, TAR, and GZIP. It also allows users to create and extract password-protected archives.

1. **Describe and compare software tools for data compression and (de)archiving in the OS family Windows.**

WinRAR: WinRAR is a powerful and popular compression and archiving tool for Windows. It supports a variety of compression formats, including its own RAR format, as well as ZIP, TAR, GZIP, and 7Z. It provides strong encryption and can split archives into multiple volumes.

7-Zip: 7-Zip is a free and open-source compression and archiving tool for Windows. It supports a variety of compression formats, including its own 7Z format, as well as ZIP, TAR, GZIP, and RAR. It provides strong encryption and can also extract ISO files.

WinZip: WinZip is a popular compression and archiving tool for Windows. It supports a variety of compression formats, including ZIP, RAR, 7Z, and GZIP. It also provides encryption and file sharing capabilities.

PeaZip: PeaZip is a free and open-source compression and archiving tool for Windows. It supports a variety of compression formats, including its own PEA format, as well as ZIP, TAR, GZIP, and RAR. It provides strong encryption and can also extract ISO files.

Bandizip: Bandizip is a lightweight and fast compression and archiving tool for Windows. It supports a variety of compression formats, including ZIP, RAR, 7Z, and GZIP. It provides strong encryption and can also extract ISO files.

When it comes to comparing these software tools, some factors to consider are the compression ratio, compression and decompression speed, ease of use, and file format support. WinRAR is known for providing high compression ratios, while 7-Zip and Bandizip are known for their fast compression and decompression speeds. PeaZip and WinZip provide a balance between ease of use and advanced features. All of these tools support a variety of file formats, but some may have more limited support than others.

**6. Explain how data compression and archiving can be used for data backup.**

**In what other system administration tasks can it be used.**

Data compression and archiving can be used as a part of a data backup strategy to reduce the amount of storage space required for backups. By compressing and archiving data, it is possible to reduce the size of the backup files, which can be particularly beneficial for larger datasets.

For example, if a system administrator needs to back up a large directory containing thousands of small files, they can use a compression tool such as gzip, bzip2 or zip to compress the directory into a single file. This compressed file can then be easily backed up to an external hard drive, cloud storage or another backup medium. In the event of data loss, the compressed and archived data can be restored from the backup to the original location.

In addition to data backup, compression and archiving can be used in several other system administration tasks. For example:

Transferring files: Compressing and archiving files can be used to reduce the time and bandwidth required to transfer files over a network or between systems.

Storage management: Archiving older or less frequently used files can help to free up disk space and improve overall storage management.

Version control: Compressing and archiving previous versions of files can help to maintain a version control system and reduce storage requirements for older versions of files.

Log management: Compression and archiving can be used to manage system logs and reduce storage requirements for logs that are no longer required for day-to-day operations.

**7. What is the purpose of the /dev/null file directory?**In Unix-like operating systems such as Linux, /dev/null is a special device file that discards all data written to it and returns no data to any process that reads from it. In other words, it is a black hole for data.

The purpose of /dev/null is to provide a way to dispose of unwanted output or input from a program. For example, if a program generates a lot of output that is not necessary, it can be redirected to /dev/null to prevent it from being displayed or saved to a file. Similarly, if a program expects input that is not required, it can read from /dev/null to ignore the input.

For example, the command command > /dev/null redirects the output of command to /dev/null, effectively discarding it. Similarly, the command command < /dev/null redirects input from /dev/null, which means that command does not receive any input.

In summary, /dev/null provides a convenient way to discard data in Unix-like operating systems, and it is often used in scripts and command-line operations to manage input and output streams.

**Conclusion:** we get practical skills for working with the Bash command shell, acquaintance with basic commands for data archiving and compression, get to know the basic actions when working with text in the terminal.