“Київський фаховий коледж зв’язку”

Циклова комісія комп’ютерної 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-13А та РПЗ-13Б

Eleven Two Zeroes:

Vlad Sapozhnyk

Max Karpenko

Dmytro Onufriiev

Київ 2024

Робота студентів групи РПЗ-13А та РПЗ-13Б Eleven Two Zeroes: Vlad Sapozhnyk, Max Karpenko and Dmytro Onufriiev.

**Мета роботи:**

1. Отримання практичних навиків роботи з командною оболонкою Bash.
2. Знайомство з базовими командами для архівування та стиснення даних.
3. Знайомство з базовими діями при роботі з текстом у терміналі.

**Матеріальне забезпечення занять:**

1. ЕОМ типу IBM PC.
2. ОС сімейства Windows та віртуальна машина Virtual Box (Oracle).
3. ОС GNU/Linux (будь-який дистрибутив).
4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

**Завдання для попередньої підготовки: *Created by Dmytro Onufriiev***

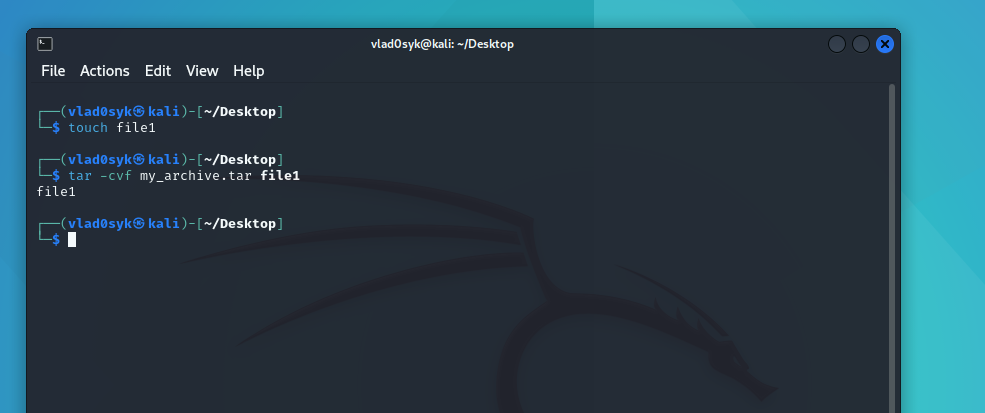
**Хід роботи. *Created by Vlad Sapozhnyk***

1. Опрацюйте всі приклади команд, що представлені у лабораторних роботах курсу ***NDG Linux Essentials - Lab 9: Archiving and Compression*** та ***Lab 10: Working With Text.*** Створіть таблицю для опису цих команд.

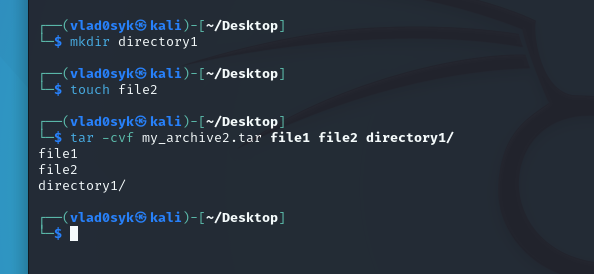
| Name of the command | Its purpose and functionality |
| --- | --- |
| mkdir mybackups | Create a new mybackups directory in the user's home directory. |
| tar -cvf mybackups/udev.tar /etc/udev | The tar command is used to combine several files into a single file. In this case, the contents of the /etc/udev directory will be saved to the udev.tar archive in the mybackups directory. The -c option tells the tar command to create a tar file. The -v option stands for "verbose," which tells the tar command to show what it is doing. The -f option is used to specify the name of the tar file. |
| tar –tvf mybackups/udev.tar | Display the contents of a tar file by using the available options (t = list contents, v = verbose, f = filename). |
| tar –zcvf mybackups/udev.tar.gz /etc/udev | To create a tar file that is compressed use -z option: The -z option makes use of the gzip utility to perform compression. |
| xz words | Using **xz** and **unxz** to compress and uncompress a file is also very similar to using **gzip** and **gunzip**. The compressed file is created with a **.xz** extension. The extension is removed when uncompressed. Execute the following commands to compress a copy of the words file. |
| unxz words.xz | Execute the following commands to uncompress the words.xz file. |
| zip words.zip words | Use the zip command to compress the words file. |
| zip -r udev.zip /etc/udev | Compress the /etc/udev directory and its contents with zip compression. |
| unzip -l udev.zip | To view the contents of a zip archive, use with the -l option with the unzip command. |
| rm -r etc  unzip udev.zip | To extract the zip archive, use the unzip command without any options. In this example we first need to delete the files that were created in the earlier tar example. |
| echo "Hello World"  echo "Hello World" > mymessage  cat mymessage | Use the redirection symbol > along with the echo command to redirect the output from the normal output of stdout (to the terminal) to a file. The cat command can be used to display file contents and will be used in this example to verify redirected output to the file. Type the following. |
| echo "Greetings" > mymessage  cat mymessage | When you use the > symbol to redirect stdout, the contents of the file are first destroyed. Type the following commands to see a demonstration. |
| cat mymessage  echo "How are you?" >> mymessage  cat mymessage | You can avoid clobbering a file by using >> instead of >. By using >> you append to a file. Execute the following commands to see a demonstration of this. |
| find ~ -name "\*bash\*" | The find command is a good command to demonstrate how stderr works. This very flexible command allows searching with a host of options such as filename, size, date, type and permission.  The find command will begin the search in the directory specified and recursively search all of the subdirectories. For example, to search for files beginning in your home directory containing the name bash. |
| find /etc -name hosts 2> err.txt  cat err.txt | To redirect stderr (error messages) to a file, issue the following command. |
| find /etc -name hosts > std.out 2> std.err  cat std.err  cat std.out | You can also redirect stdout and stderr into two separate files. |
| find /etc -name hosts > find.out 2>&1  cat find.out | To redirect both standard output (stdout) and standard error (stderr) to one file, first redirect stdout to a file and then redirect stderr to that same file by using the notation 2>&1. |
| tr a-z A-Z  this is interesting  how do I stop this?  ^D | Standard input (stdin) can also be redirected. Normally stdin comes from the keyboard, but sometimes you want it to come from a file instead. For example, the tr command translates characters, but it only accepts data from stdin, never from a file name given as an argument. This is great when you want to do something like capitalize data that is inputted from the keyboard (Note: Press **Control**+**d**, to signal the tr command to stop processing standard input). |
| tr A-Z a-z > myfile  Wow, I SEE NOW  This WORKS! | The tr command accepts keyboard input (stdin), translates the characters and then redirects the output to stdout. To create a file of all lower-case characters, execute the following. |
| cat myfile  tr a-z A-Z < myfile | Execute the following commands to use the tr command by redirecting stdin from a file. |
| ls -l /etc | more | Another popular form of redirection is to take the output of one command and send it into another command as input. For example, the output of some commands can be massive, resulting in the output scrolling off the screen too quickly to read. Execute the following command to take the output of the ls command and send it into the more command, which displays one page of data at a time. |
| cut -d: -f1 /etc/passwd | In the following example, you will use a command called cut to extract all of the usernames from a database called /etc/passwd (a file that contains user account information). First, try running the cut command by itself. |
| cut -d: -f1 /etc/passwd | sort | The output in the previous example was unordered and scrolled off the screen. In the next step you are going to take the output of the cut command and send it into the sort command to provide some order to the output. |
| cut -d: -f1 /etc/passwd | sort | more | Now the output is sorted, but it still scrolls off the screen. Send the output of the sort command to the more command to solve this problem. |
| cat /etc/passwd | The /etc/passwd is likely too large to be displayed on the screen without scrolling the screen. To see a demonstration of this, use the cat command to display the entire contents of the /etc/passwdfile. |
| more /etc/passwd | Use the more command to display the entire contents of the /etc/passwd file. |
| h | While you are in the more command, you can view the help screen by pressing the **h** key. |
| <SPACE> | Press the **Spacebar** to view the rest of the document. |
| less /etc/passwd  /bin  nnnNNNq | Use the less command to display the entire contents of the /etc/passwd file. Then search for the word bin, use **n** to move forward, and **N** to move backwards. Finally, quit the less pager by typing the letter **q.** |
| head /etc/passwd | You can use the head command to display the top part of a file. By default, the head command will display the first ten lines of the file. |
| tail /etc/passwd | Use the tail command to display the last ten lines of the /etc/passwd file. |
| head -2 /etc/passwd | Use the head command to display the first two lines of the /etc/passwd file. |
| ls /etc | tail -5 | Execute the following command line to pipe the output of the ls command to the tail command, displaying the last five file names in the /etc directory. |
| head -n -20 /etc/passwd | Another way to specify how many lines to output with the head command is to use the option -n -#, where # is the number of lines counted from the bottom of the output to exclude. Notice the minus symbol - in front of the #. For example, if the /etc/passwd contains 27 lines, the following command will display lines 1-7, excluding the last twenty lines. |
| cd /etc  grep sshd passwd | The use of grep in its simplest form is to search for a given string of characters, such as sshd in the /etc/passwd file. The grep command will print the entire line containing the match. |
| grep root passwd | Regular expressions are "greedy" in the sense that they will match every single instance of the specified pattern. |
| grep '^root' passwd | To limit the output, you can use regular expressions to specify a more precise pattern. For example, the caret ^ character can be used to match a pattern at the beginning of a line; so, when you execute the following command line, only lines that begin with root should be matched and displayed. |
| grep 'sync' passwd | Match the pattern sync anywhere on a line. |
| grep 'sync$' passwd | Use the $ symbol to match the pattern sync at the end of a line. |
| grep '.y' passwd | Use the period character . to match any single character. For example, execute the following command to match any character followed by a 'y'. |
| grep 'sshd|root|operator' passwd | The pipe character, |, or "alternation operator", acts as an "or" operator. For example, execute the following to attempt to match either sshd, root or operator. |
| grep -E 'sshd|root|operator' passwd | Use the -E switch to allow grep to operate in extended mode in order to recognize the alternation operator. |
| egrep 'no(b|n)' passwd | Use another extended regular expression, this time with egrep with alternation in a group to match a pattern. The strings nob and non will match. |
| head passwd | grep '[0-9]' | The [ ] characters can also be used to match a single character. However, unlike the period character ., the [ ] characters are used to specify exactly what character you want to match. For example, if you want to match a numeric character, you can specify [0-9]. Execute the following command for a demonstration. |
| grep -E '[0-9]{3}' passwd | Suppose you want to search for a pattern containing a sequence of three digits. You can use { } characters with a number to express that you want to repeat a pattern a specific number of times; for example: {3}. The use of the numeric qualifier requires the extended mode of grep. |

1. Ознайомтесь з командою tar та за її допомогою виконати у терміналі наступні дії:

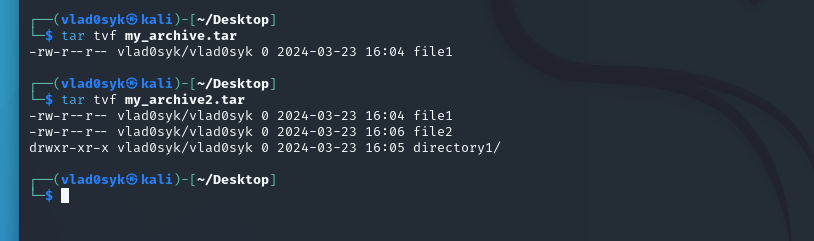
* створити файл з розширенням .tar;



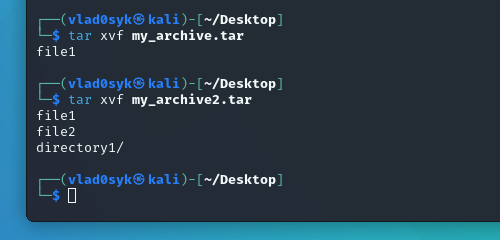
* створити файл з розширенням .tar, що складається з декількох файлів і каталогів одночасно;



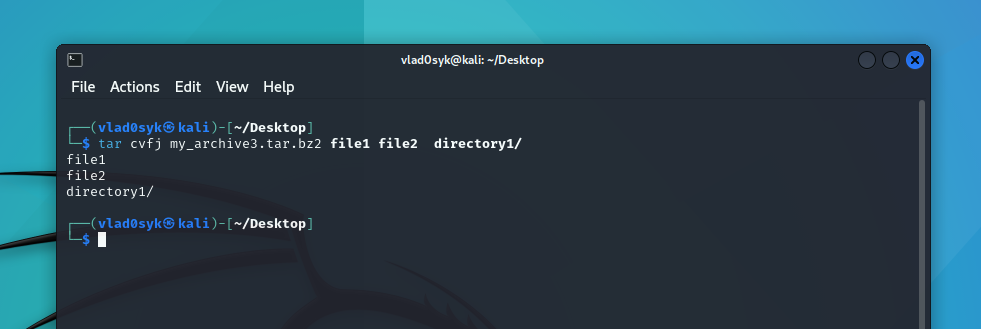
* перегляду вмісту файлу;



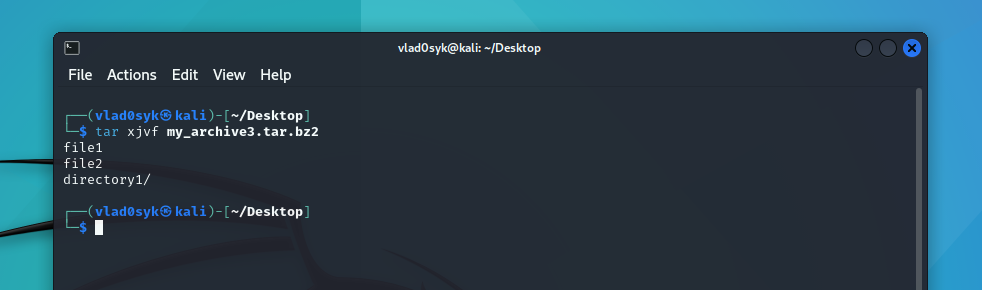
* витягти вміст файлу tar;



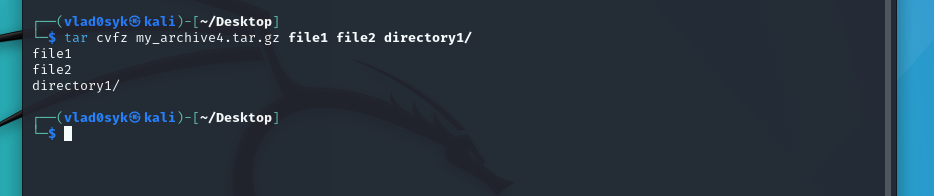
* створити архівний файл tar, стиснений за допомогою bzip;



* витягти вміст файлу tar bzip;



* створити архівний tar файл, стисненого за допомогою gzip;



* витягти вміст файлу tar gzip.



1. \*Як буде відбуватись перенаправлення потоків виведення в bash для наступних дій з командами (позначено як cmd) та файлами (позначено як file):

| Command | What does command do? |
| --- | --- |
| cmd 1> file | The command cmd 1> file attempts to redirect the output of the command cmd to a file named file. |
| cmd > file | The command cmd > file attempts to redirect the standard output of the command cmd to a file named file. |
| cmd 2> file | The command cmd 2> file attempts to redirect the standard error of the command cmd to a file named file. |
| cmd >> file | The command cmd >> file attempts to append the standard output of the command cmd to a file named file. |
| cmd &> file | The command cmd &> file redirects both the standard output **and** standard error of the command cmd to a file named file. |
| cmd > file 2>&1 | The command cmd > file 2>&1 combines two redirection operators to control the output of the command cmd (assuming cmd is a valid command) |
| cmd >> file 2>&1 | The command cmd >> file 2>&1 combines redirection techniques to achieve a specific outcome. |
| cmd 2>&1 > /dev/null | The command cmd 2>&1 > /dev/null silences the output of the command cmd, regardless of whether it's successful or produces errors. |
| cmd 2> /dev/null | The command cmd 2> /dev/null aims to suppress any error messages that might be generated during the execution of the command cmd. |
| cmd1 | cmd2 | The command cmd1 | cmd2 uses a concept called piping (|) to connect the output of one command (cmd1) to the input of another command (cmd2). |
| cmd1 2>&1 | cmd2 | The command cmd1 2>&1 | cmd2 combines redirection and piping to achieve a specific outcome. |

1. \*\*Розгляньте наведені нижче приклади та поясніть, що виконують дані команди та який тип перенаправлення потоків вони використовують:

| Command  (command container) | What does the command do? | What is the redirection flow? |
| --- | --- | --- |
| $echo "It is a new story." > story |  |  |
| $ date > date.txt |  |  |
| $ cat file1 file2 file3 > bigfile |  |  |
| $ls -l >> directory |  |  |
| $ sort < file1\_unsorted > file2\_sorted |  |  |
| $ find -name '\*.txt' > file.txt 2> /dev/null |  |  |
| $ cat file1\_unsorted | sort > file2\_sorted |  |  |
| $ cat myfile | grep student | wc -l |  |  |

**Контрольнi запитання: *Created by Max Karpenko***

**Висновок:**