“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної та програмної інженерії

ЗВІТ ПО ВИКОНАННЮ

ЛАБОРАТОРНОЇ РОБОТИ №3

з дисципліни: «Операційні системи»

Тема: “Знайомство з базовими командами CLI-режиму в Linux”

Виконали

студенти

групи РПЗ-03

Команда 6: Sichkar Maxim,

Brytyuk Bohdan

Kyiv 2023

**Мета роботи:**

1. Знайомство з базовими командами CLI-режиму в Linux.

2. Знайомство з базовими текстовими командами в термінальному режимі роботи в різних ОС.

**Матеріальне забезпечення занять:**

ЕОМ типу IBM PC.

2. ОС сімейства Windows (Windows 7).

3. Віртуальна машина – Virtual Box (Oracle).

4. Операційна система GNU/Linux – CentOS.

**Завдання для попередньої підготовки.**

Прочитайте короткі теоретичні відомості до лабораторної роботи та зробіть невеликий словник

базових англійських термінів з питань призначення команд та їх параметрів.

2. Вивчіть матеріали онлайн-курсу академії Cisco “NDG Linux Essentials”:

- Chapter 5 - Command Line Skills

- Chapter 6 - Getting Help

3. Пройдіть тестування у курсі NDG Linux Essentials за такими темами:

- Chapter 05 Exam

- Chapter 06 Exam

**4. Дайте визначення наступним поняттям:**

**- Command prompt.**

A command prompt is software that provides a user interface for interacting with an operating system through the command line or scripted files. The user can execute various commands or run programs by entering their names and parameters at the command line, the interpreter converts the commands into instructions that the operating system can understand and execute. In addition, an interpreter may have the ability to work with the file system, change settings, start background processes, perform paging, and other operations. The most commonly used shells in UNIX-like operating systems are Bash, Zsh, and Fish.

**- Shell**

A shell refers to software that provides users with access to an operating system through a command line or graphical interface. The shell is an intermediate layer between the user and the operating system, allowing the user to interact with the operating system by executing commands and running programs.

Shells can have various functions and capabilities, such as command history, auto-complete, the ability to redirect input and output, pipeline commands, run background processes, and other operations. In UNIX-like operating systems, the most common shells are Bash, Zsh, and Fish. In the Windows operating system, the standard shells are the Command Prompt and PowerShell.

**- Command**

A command refers to software that provides users with access to an operating system through a command line or graphical interface. A shell is an intermediate layer between the user and the operating system that allows the user to interact with the operating system by executing commands and running programs.

Shells can have various functions and capabilities, such as command history, auto-complete, the ability to redirect input and output, pipeline commands, run background processes, and other operations. In UNIX-like operating systems, the most common shells are Bash, Zsh, and Fish. In the Windows operating system, the standard shells are the Command Prompt and PowerShell.

**5. Дайте відповіді на наступні питання:**

**- What basic information does the prompt prompt provide?**

The prompt prompt is the main command line interface of an operating system and provides the user with basic information about the current state of the system and possible actions that he or she can perform.

- Typically, the prompt line includes the following information:

- The username used to log in to the system.

- Host name or the name of the computer where the command prompt is running.

- The path to the current working directory in which the user is currently located.

- A character or sign that prompts the user to enter a command, for example, usually the "$" character in UNIX-like operating systems or the ">" character in Windows.

For example, the prompt line in a UNIX-like operating system might look like "user@hostname:/current/directory/$", where "user" is the user name, "hostname" is the host name, "/current/directory/" is the current path to the working directory, and "$" is the command prompt character.

**- Why does the command need parameters and arguments?**

In UNIX-like operating systems, parameters and arguments are needed to change the behavior of a command and affect its output.

Parameters are usually options for a command that change its behavior or output. For example, the "ls" command without parameters displays a list of files and folders in the current directory, but with the additional parameter "-l" displays more detailed information about each file and folder.

Arguments are data that a command processes or that is used by its execution. For example, the "cp" command copies files from one directory to another, the arguments of this command specify the paths to the files and folders to be copied and the destination for them.

There are many commands with different parameters and arguments in UNIX-like operating systems that allow the user to manage and control the system. The user can find documentation for each command and its parameters and arguments by using the help or the "man" or "info" commands.

**- What is the purpose of the ls command, what parameters and arguments can it have? Give 3 examples.**

The ls command is used to display a list of files and folders in the current directory or in a specified directory. The main purpose of this command is to get information about files and folders on a file system.

The ls command has the following options:

-a - displays all files and folders, including those starting with a dot (hidden files)

-l - displays more detailed information about files and folders, such as permissions, owner, size, date and time of modification, etc.

h - displays file sizes in an easy-to-read format

Below are 3 examples of how to use the ls command with parameters and arguments:

ls -a - lists all files and folders, including hidden files and folders in the current directory.

ls -l /home/user/documents - display more detailed information about the files and folders in the /home/user/documents directory.

ls -lh /var/log - lists the files and folders in the /var/log directory, including their sizes, in an easy-to-read format.

**- How can I use command history, what are the benefits?**

Using command history is an opportunity to quickly repeat previously entered commands without having to re-enter them from the keyboard. In Linux, command history is stored in the terminal's memory and can be recalled and used at any time.

To use the command history in Linux, you can use the following methods:

Using the up arrow key on the keyboard, which allows you to view previous commands entered from the keyboard.

Using the history command, which displays a list of the last commands you entered, with command numbers, allowing you to reuse any of them with the !command number command.

Use additional options, such as grep, to quickly find the command you need in the command history.

Setting up the saving of command history to a file. This allows you to save the history between terminal sessions and conveniently view and repeat previously entered commands.

Advantages of using command history in Linux:

Quick and convenient repetition of previously entered commands without the need to re-enter them from the keyboard.

The ability to view the last commands you entered, which allows you to detect and correct errors.

Ability to store command history between terminal sessions, which allows you to conveniently view and repeat previously entered commands in the future.

**- What is the purpose of the echo command?**

The echo command in Linux is designed to output text to the standard output. This command allows you to display any text string on the screen, and it can also be used in scripts to display messages to users.

For example, the echo "Hello World" command will display the string "Hello World" on the screen. The echo command can also be used with additional options and arguments, such as:

-n - allows you to print text without a newline character, which allows you to print messages on the same line.

-e - allows you to use escape sequences such as \n to insert a newline character.

$ - allows you to use the values of environment variables in the output text.

For example, the command echo -n "Enter your name: " will display the text "Enter your name: " on the screen without a newline character so that the user can enter their name on the same line.

Thus, the echo command is a powerful tool for displaying text and messages on the screen in Linux.

**- Describe the concept of a variable in the Bash shell, what types of variables does it support?**

Variables in the Bash shell are a tool for storing named values that can be used in commands and scripts. Variables can be used to store strings, numbers, and Boolean values.

Bash supports the following types of variables:

String variables are variables that contain strings of characters. To define string variables, you use the = assignment operator. For example, name="John" defines the variable name with the value "John". String variables can also be called with $, for example, echo $name will print the string "John".

Integer variables are variables that contain integers. To define integer variables, use the = assignment operator. For example, count=10 defines the variable count with a value of 10. Integer variables can also be called with $, for example, echo $count will print the integer 10.

Floating point variables are variables that contain fractional numbers. To define floating point variables, use the = assignment operator. For example, pi=3.14 defines the variable pi with a value of 3.14. Floating point variables can also be called with $, for example, echo $pi will print the fractional number 3.14.

Variables of Boolean type are variables that contain a value of true or false. To define variables of Boolean type, the assignment operator = is used. For example, is\_valid=true defines

**- What is the purpose of the env, export, and unset commands?**

The env command in Linux is used to display the current values of environment variables. In addition, the command can be used to start a program in a new environment where other values of environment variables can be set. For example, the env python command will start the Python interpreter in the current environment.

The export command is used to set the value of an environment variable. When a variable is exported, its value becomes available in all process descendants, including other commands and programs. For example, the command export MY\_VAR="Hello" sets the value "Hello" for the environment variable MY\_VAR.

The unset command is used to delete an environment variable. If the variable does not exist, the command does not perform any action. For example, the unset MY\_VAR command will delete the environment variable MY\_VAR.

All of these commands are related to managing environment variables in Linux, where the value of these variables can be used in programs and scripts running on the operating system.

**- What commands do you know for terminal help?**

There are several commands that you can use to get help for commands in the Linux terminal:

man: The man command provides complete help for any command, including parameter descriptions, usage, and examples. To use this command, type man followed by the command name, for example, man ls.

--help: Most commands in Linux support the --help option, which provides a brief description of the command and examples of how to use it. To use this option, you must enter the command name and --help, for example, ls --help.

info: The info command provides information about a program or command similar to man, but in a more extended form. To use this command, you must enter info and the command name, for example, info ls.

whatis: The whatis command provides a brief description of a command, including its purpose. To use this command, you must enter whatis and the command name, for example, whatis ls.

apropos: The apropos command allows you to search for commands by keywords. It displays a list of commands that contain the entered keyword in their name or description. To use this command, you must type apropos and a keyword, such as apropos file.

Each of these commands provides a different level of detail and information about commands in Linux, so the choice of which command to use for help depends on the user's specific needs.

6. Підготувати в електронному вигляді початковий варіант звіту:

- Титульний аркуш, тема та мета роботи

- Відповіді на п.4 та п.5 з завдань для попередньої підготовки

**Хід роботи.**

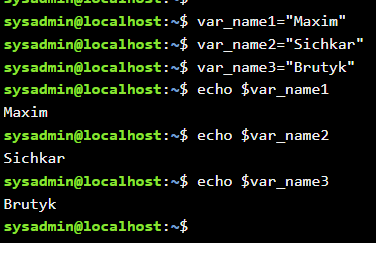
**Опрацюйте всі приклади команд, що представлені у лабораторній роботі курсу NDG Linux Essentials - Lab 5: Command Line Skills та Lab 6: Getting Help. Створіть таблицю для опису цих команд.**

|  |  |
| --- | --- |
| **whoami** | The output of the whoami command, sysadmin, displays the user name of the current user. Although in this case your username is displayed in the prompt, this command could be used to obtain this information in a situation when the prompt did not contain this information. |
| **uname** | The next command displays information about the current system. To be able to see the name of the kernel you are using |
| **uname -n** | Run unamecomandu again in a terminal, once with the -ni option again with the --nodename option. This will display the hostname of the network node, also found in the prompt. |
| **pwd** | The pwd command is used to display your current "location" or current "working" directory. |
| **!** | To execute a command again, type the exclamation point and the history list number |
| **echo** | The echo command can be used to print text and the value of a variable, and to show how the shell environment expands metacharacters |

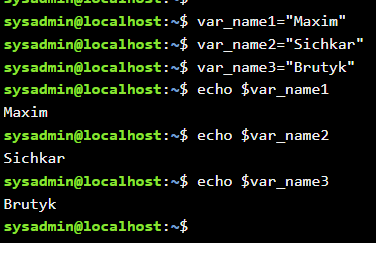
**2. Work in the terminal (consolidation of practical skills) be sure to submit your screenshots:**

**2.1. Work with variables and aliases in the terminal:**

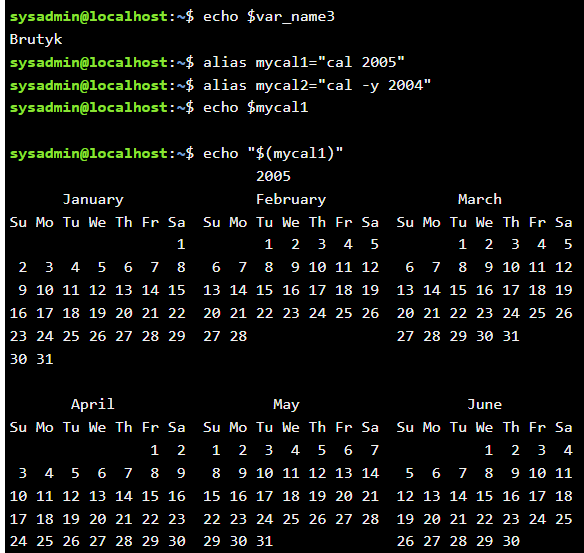
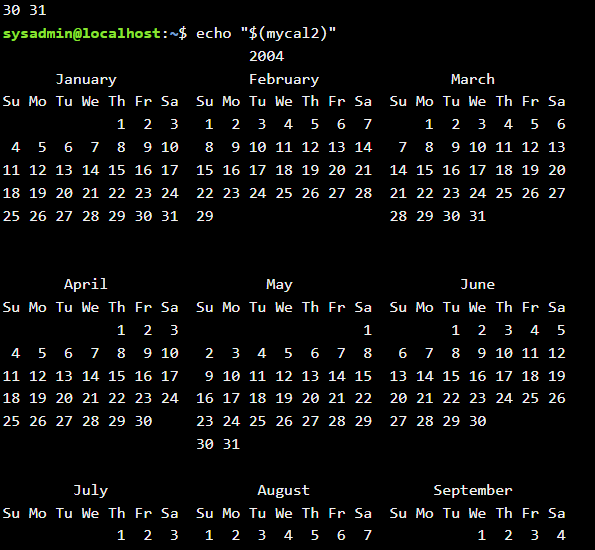
**- Create variables that will contain your first and last names $var\_name1, $var\_name2, $var\_name3**

****

**- Use the echo command to print the names of the students in your team**

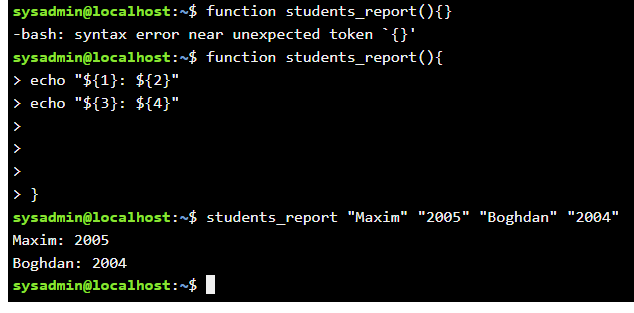
****

**- Create aliases mycal1, mycal2, mycal3 for the cal command to automatically display the calendar of your year of birth**

****

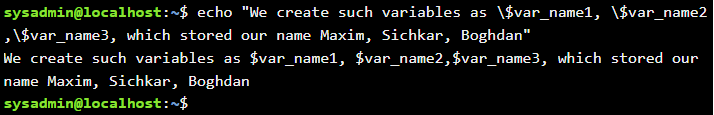
**2.2. Work with functions in the terminal:**

**- Create a students\_report function that will display the names of your team's students in order, followed by their years of birth**

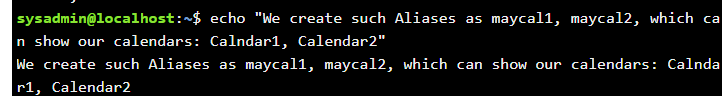
****

**2.3. Working with quotes (Quoting) in the terminal. Print the following sentences on the command line:**

- "We create such variables as $var\_name1, $var\_name2, $var\_name3, which stored our names Name1, Name2, Name3" (in the sentence, first print the names of the variables, and then their contents)

****

- "We create such Aliases as mycal1, mycal2, mycal3, which can show our calendars: Calendar1, Calendar2, Calendar3" (in the sentence, first display the name of the alias command, then display the output of these commands).

****

**2.4. Working with Control Statements in the terminal:.**

- Is it possible to perform tasks 2.1 and 2.2 of the workflow using control statements without writing a separate function, how will it look like?

Yes, you can. This is how the code will look like without the "function"

var\_name1="Maxim"

var\_name2="Sichkar"

var\_name3="Brutyk"

echo "Maxim 2005"

echo "Boghdan 2004"

**2.5. Working with Man Pages commands in the terminal:.**

- Using the uname command as an example, demonstrate how to get help. Based on the additional information obtained, provide 5 different options for displaying the result of the information on this command using 5 different options

To get help about the uname command, you can use the man uname or uname --help command.







**Контрольні запитання:**

1. **Які типи команд існують в оболонці Bash?**

There are several types of commands in the Bash shell:

Built-in commands: these are commands that are built into the Bash shell itself, so their execution is faster than running separate programs. Examples of built-in commands include cd, echo, alias, and so on.

System programs: these are commands that are represented in the system as separate executable programs in the file system. Examples of system programs include ls, grep, find, and so on.

Functions: these are small pieces of code that can be used as commands. Functions can contain parameters and shell interpreter variables.

Shell scripts: these are sets of commands that are stored in a text file and executed as a single block of code. Shell scripts can contain conditional statements, loops, and other functions, making them useful for automating tasks.

Aliases: these are short names that can be used to replace long command names or to create custom combinations of commands. Aliases are created using the alias command.

Each of these command types has its own peculiarities and can be used to perform different tasks in the Linux system.

1. **Що таке змінні оточення? Які вони бувають. Як їх можна переглянути в терміналі?**

Environment variables are dynamic named values that can affect the behavior of processes running on a computer. They are used to store information such as system paths, user preferences, and configuration settings.

In Linux, environment variables are set and managed by the shell. When a new shell is opened, it inherits the environment variables of its parent shell.

Environment variables can be viewed in the terminal using the env command. This command prints a list of all currently defined environment variables and their values. Alternatively, the printenv command can also be used to view environment variables. To view the value of a specific environment variable, you can use the echo command followed by the name of the variable enclosed in dollar signs, for example: echo $PATH.

Environment variables can be set and modified using the export command. For example, to set the MY\_VAR variable to a value of hello, you would use the command export MY\_VAR=hello. The unset command can be used to remove an environment variable, for example: unset MY\_VAR.

Overall, environment variables provide a powerful and flexible way to customize the behavior of the system and applications running on it.

1. **Опишіть змінну $PS1. Як в терміналі переглянути її вміст?**

The $PS1 variable is an environment variable that contains the definition of the primary prompt for the Bash shell. The prompt is the text that appears in the terminal to indicate that the shell is waiting for input. By default, the $PS1 variable is set to '\s-\v\$ ' which displays the name and version of the shell followed by a dollar sign and a space.

The $PS1 variable can be customized to display any information that the user finds useful or convenient. It can include text, special characters, and escape sequences that represent various properties such as the current working directory, username, hostname, and more.

To view the contents of the $PS1 variable in the terminal, you can use the echo command followed by the variable name enclosed in dollar signs. For example, to display the contents of the $PS1 variable, you would enter echo $PS1 in the terminal.

Overall, customizing the $PS1 variable can be a helpful way to make the terminal more personalized and easier to use.

1. **Як можна змінити значення змінної $PS1? Що при цьому відбудеться в рядку запрошенні в bash (рядок запрошення перед початком кожної команди). Як змінити значення цієї змінної не на поточний сеанс, а за замовчуванням?**

To change the value of the $PS1 variable, you can use the export command followed by the new value of the variable. For example, to change the prompt to display the current working directory followed by a dollar sign, you could use the following command:

export PS1="\w\$ "

This sets the value of the $PS1 variable to \w\$ , where \w represents the current working directory and \$ represents the prompt character, which is a dollar sign.

Once you have changed the value of $PS1, the new prompt will be displayed in the terminal before each command you enter.

To change the value of $PS1 permanently, you can add the export command to your shell startup file, such as ~/.bashrc. This ensures that the new value of $PS1 is applied each time you start a new shell session.

To restore the default value of $PS1, you can either manually set it to the default value or simply unset it using the unset command. The default value of $PS1 can vary depending on the distribution and version of Linux you are using, but it is often set to '\s-\v\$ '. For example, to restore the default value of $PS1, you could use the following command:

unset PS1

This removes the value of $PS1 and restores the default prompt.

1. **Для чого використовують лапки в оболонці Bash?**

In the Bash shell, quotation marks are used to specify how the shell should treat strings and variables that are passed as arguments to commands. There are three types of quotation marks that can be used in Bash:

Single quotes ('): Any characters within single quotes are treated as literal strings, and special characters within them lose their special meaning. This means that variables and escape sequences within single quotes are not expanded or evaluated by the shell. For example, 'hello $USER' would be treated as a literal string, and the variable $USER would not be replaced with its value.

Double quotes ("): Any characters within double quotes are treated as literal strings, except for certain characters that are treated as special. Variables within double quotes are expanded to their values, and some escape sequences are recognized and evaluated by the shell. For example, "hello $USER" would be expanded to "hello username", where username is the value of the $USER variable.

Backticks (`): Backticks are used to execute a command and substitute its output as part of another command or expression. For example, echo `date` would execute the date command and substitute its output into the echo command.

1. **Для чого використовують інструкції керування, які їх види Ви знаєте?**

In Bash, control instructions are used to alter the flow of execution of a script or command. They allow you to specify conditions that determine which commands should be executed and in what order. Some of the most commonly used control instructions in Bash include:

if/then/else: This instruction allows you to specify a condition that should be tested, and a set of commands to be executed if the condition is true and another set of commands to be executed if the condition is false.

for/while/until: These instructions allow you to execute a set of commands repeatedly, based on a specified condition. for loops iterate over a set of values, while while and until loops execute as long as a specified condition is true or false, respectively.

case/esac: This instruction allows you to test a variable against a set of possible values and execute a set of commands based on the value of the variable.

break/continue: These instructions allow you to alter the behavior of loops. break exits a loop prematurely, while continue skips the remainder of the current iteration of the loop and moves on to the next one.

function: This instruction allows you to define a reusable block of code that can be called multiple times within a script.

1. **В чому різниця якщо в кінці рядку запрошення bash стоїть символ $ чи #? Наприклад на екрані ми бачимо наступні записи**

**[centos@localhost Desktop]$**

**[root@localhost Desktop]#**

The Bash prompt line typically ends with either a $ or # character, and the difference between them depends on the user's permissions and the current working directory.

A prompt ending with a $ character indicates that the user is logged in as a regular non-root user, and the commands are being executed with regular user permissions. For example, the prompt [centos@localhost Desktop]$ indicates that the user centos is logged in as a regular user and currently in the Desktop directory.

A prompt ending with a # character indicates that the user is logged in as the root user, and the commands are being executed with root user permissions. The root user has complete control over the system and can execute any command with unrestricted access. For example, the prompt [root@localhost Desktop]# indicates that the user root is currently in the Desktop directory and has root user permissions.

The difference between a prompt ending with a $ or # character is that the $ character indicates a regular user with limited permissions, while the # character indicates the root user with complete system control.

1. **Яке призначення команд whereis та locate? Яка між ними відмінність?**

The whereis and locate commands are used to find the location of files or binaries in a Linux system, but they differ in their approach and scope.

The whereis command is used to locate the binary, source, and man page files for a command. When executed with a specific command as an argument, whereis searches the standard binary directories, such as /usr/bin, /bin, and /usr/local/bin, for the binary file, the source code directory, and the man page file. For example, whereis python will search for the python binary, source, and man page files in the default locations. The whereis command provides a quick way to locate the most important files related to a command.

The locate command, on the other hand, searches a pre-built database of file names and paths on the system, which allows it to locate files more quickly and efficiently than whereis. By default, locate searches the entire file system, but you can restrict the search to specific directories or file types using options. For example, locate \*.txt will search the entire file system for files with a .txt extension. The locate command is useful for finding files or directories quickly and efficiently, especially when working with a large file system.

**Conclusion:** I have familiarized myself with the basic commands of the CLI mode in Linux, and I have familiarized myself with the basic text commands in terminal mode in different OSes. I improved my skills in using the console and the Linux operating system, collaborated with the team