Київський фаховий коледж зв’язку

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №3**

з дисципліни: «Операційні системи»

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

Виконали: студенти 3 курсу,

групи КСМ-13А

Засенко Олександр      
Дзюбенко Дмитро    
Сторожук Костянтин

Київ 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**

1. ЕОМ типу IBM PC.

2. ОС сімейства Windows (Windows 7).

3. Віртуальна машина – Virtual Box (Oracle).

4. Операційна система GNU/Linux – CentOS.

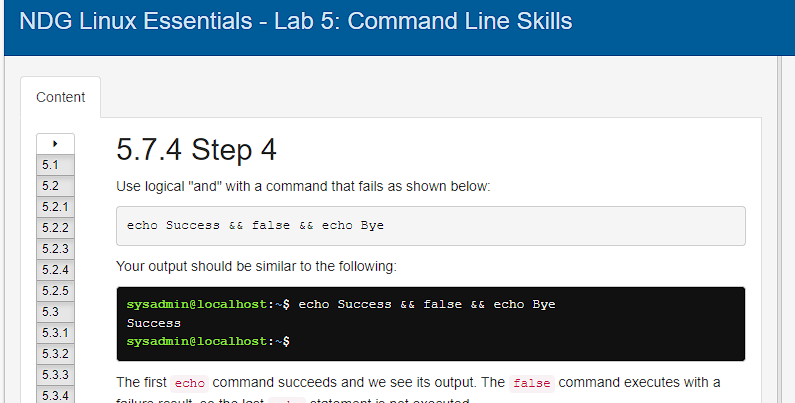
**Завдання для попередньої підготовки**

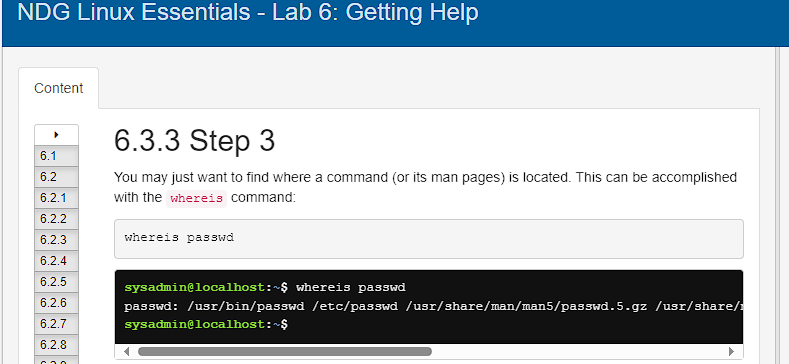
***Готував матеріал студент Zasenko***

Невеликий словник базових англійських термінів з питань призначення команд та їх параметрів.

|  |  |
| --- | --- |
| **Команди** | **Пояснення українською** |
| **ls** | **Вивести список файлів і каталогів у поточному каталозі.** |
| cd | **Змінити поточний каталог.** |
| pwd | **Показати поточний каталог (шлях)** |
| mkdir | **Створити новий каталог.** |
| rmdir | **Видалити порожній каталог.** |
| touch | **Створити новий порожній файл.** |
| cp | **Копіювати файли або каталоги.** |
| **mv** | Перемістити (перейменувати) файли або каталоги. |
| **rm** | Видалити файли або каталоги. |
| **cat** | Вивести вміст файлу на екран |

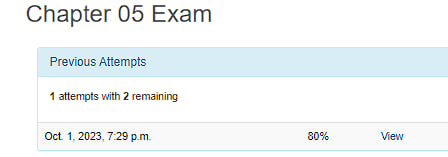
**Вивчить матеріали онлайн-курсу академії Cisco “NDG Linux Essentials”:**



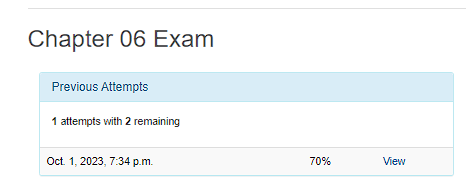


**Пройдіть тестування у курсі NDG Linux Essentials за такими темами:**

**- Chapter 05 Exam**



**- Chapter 06 Exam**



**Дайте визначення наступним поняттям:**

***Готував матеріал Zasenko***

**Командний інтерпретатор (shell)** **-** це програма в операційній системі, що дозволяє вам взаємодіяти з комп'ютером через текстовий інтерфейс. Він виконує команди, працює з файловою системою, управляє процесами та дозволяє створювати скрипти для автоматизації завдань. Bash є стандартним інтерпретатором для багатьох Linux-систем.

**Оболонка (shell)** - це інтерфейс між користувачем та операційною системою, який дозволяє користувачеві взаємодіяти з комп'ютером через команди і скрипти. Оболонка приймає команди від користувача, виконує їх та повертає результати. Вона також надає доступ до файлової системи, управляє процесами і виконує інші системні завдання.

**Команда** - це інструкція або вираз, який ви вводите в командний рядок або командний інтерпретатор (shell) операційної системи для виконання певної дії або завдання. Команди можуть включати запуск програм, взаємодію з файлами, керування процесами, налаштування системи та багато інших дій.

Наприклад, ось декілька прикладів команд:

1. ls - команда для відображення списку файлів і каталогів у поточному каталозі.
2. cd - команда для зміни поточного каталогу.
3. mkdir - команда для створення нового каталогу.
4. cp - команда для копіювання файлів або каталогів.
5. mv - команда для переміщення або перейменування файлів або каталогів.
6. rm - команда для видалення файлів або каталогів.
7. ps - команда для відображення списку активних процесів.
8. kill - команда для завершення (вбивання) процесів.

**Дайте відповіді на наступні питання:**

1. The prompt line in the shell provides basic information that helps the user determine the current state of the system and whether it is ready to enter commands. This information includes:

1. User name: The prompt bar often contains a username that indicates who you are entering commands as.

2. Computer name (host): This can include the name or address of the computer on which you are running the commands, especially if you are working on a network or remotely.

3. Current directory: Usually shows the path to the current working directory you are working in. This helps you identify where you are in the file system.

4. Command delimiter: Usually the $ symbol is used for users with limited rights and # for users with administrator (root) rights. It indicates that the system is ready to enter a new command.

Examples of invitation strings:

* user@hostname:~/myfolder$ - a regular user.
* root@localhost:/var/www# - administrator (root).

The invitation string can be customized and customized by the user depending on their own requirements and preferences.

2. Commands in the operating system use parameters and arguments to provide additional information and control their behavior. Here's the difference between parameters and arguments:

1. Parameters (options):

* Parameters are specified on the command line after the command name and are usually preceded by one or two hyphens (e.g., -f or --verbose).
* They allow you to customize various aspects of the command, such as operating modes, level of output detail, file paths, etc.
* The options are optional and can be passed in any order.

Example: 

In this example, -l and --all are parameters of the ls command that set the output mode to display a more detailed list of files and include hidden files.

2. Arguments:

* Arguments are data or values that are passed to the command after the parameters.
* They are specified without hyphens and are used for specific operations or processing.
* Arguments can be mandatory or optional, depending on the command.

Example: 

In this example, source.txt and destination/ are arguments to the cp command that specify which file to copy and where to copy it to.

The use of parameters and arguments allows commands to be more flexible and simplifies the user's interaction with the operating system by providing the ability to customize and control their operation depending on the needs.

3. The ls command in Linux and Unix operating systems is used to display a list of files and directories in the current directory. It allows the user to browse the contents of a directory. Here are some of its possible parameters and arguments:

1. **Parameter.** -l **(long format)**:

* The -l option allows you to display a detailed list of files and directories, including information about permissions, owner, group, size, last modified time, and file or directory name.

Example: 

2**. The -a (all) option:**

* The -a option allows you to output all files, including hidden files, that begin with the dot ..

Example: 

3. **Arguments (paths to directories or files):**

* You can pass arguments that are paths to specific directories or files, and ls will list the contents of those directories or information about the specified files.

Example: 

These are just a few examples of how to use the ls command. ls has many other options and features for customizing output, and you can combine them to achieve the desired result.

4. Using command history is a useful feature of the shell that allows users to save and use previous commands they have entered at the command prompt. To use command history, you typically use the Up Arrow and Down Arrow key combination on your keyboard.

Here's how you can use team history and what benefits it provides:

1. Repeating commands: Pressing the Up Arrow allows you to view previous commands you have entered. This allows you to easily repeat commands you have already executed, which saves time and prevents you from having to enter a command again from scratch.

2. Edit commands: After you display a previous command, you can edit it before executing it. This allows you to correct mistakes or make changes to the command before executing it again.

3. Search for commands: You can quickly find previous commands by using Ctrl + R and searching for keywords or phrases stored in the history. This is especially useful when working with a large volume of commands.

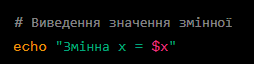
4. Training: Command history can also serve as a learning tool. You can review your previous commands to learn how to use operating system commands more efficiently and effectively.

5. Automation: You can create scripts and batch files that use command history to automatically perform a sequence of actions.

In general, using command history makes working at the command line more convenient, faster, and more productive by simplifying access to previously entered commands and providing a user-friendly interface for editing and re-executing commands.

5. The echo command in Linux and Unix operating systems is used to display textual content on a console or terminal screen. The main purpose of the echo command is to display text messages, variables, or other content on the user's screen. It can be useful in a variety of scenarios, including:

**1.** Debugging scripts: When debugging scripts or batch files, the echo command allows you to output variable values, intermediate results, and messages to verify that the code is executing correctly.

Example: 

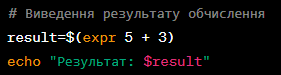
**2.** Display messages for the user: The echo command can be used to output informational messages or prompts to the user while a script or command is executing.

Example:

**3.** Create text files: You can use the echo command to create text files or add text to existing files.

Example: 

**4.** Display variable values: The echo command can be used to output variable values or other data that you want to display.

Example:

In general, the echo command is an important tool for outputting textual content on the command line, and it is used in many scripts and commands to communicate with the user and convey information.

6. A variable in the Bash shell is a named object that stores a value. It can be used to store text information, numbers, or other data that you can use in scripts, commands, and expressions in the shell. The main characteristics of variables in the Bash shell include:

1. **Variable names:** Variable names must begin with a letter or underscore (\_) and can contain letters, numbers, and underscores. The case of the letters is important (the difference between uppercase and lowercase letters is considered different variables).

2. **Assigning values:** A value can be assigned to a variable using the = operator. For example, x=10 assigns the value 10 to the variable x.

3. **Reading values:** You can read the values of variables by accessing them by name, for example, echo $x will display the value of the variable x.

4. **Environment variables:** Bash also supports environment variables, which are available in all processes running with the shell. Typically, environment variable names are capitalized (e.g., PATH).

5. **Local and global variables:** Variables can be local (available only within the current process or script) or global (available to all processes).

6. **Special variables:** Bash has some special variables, such as $0 (name of the running program), $1, $2, ... (command line arguments), $# (number of command line arguments), $? (exit code of the last command), and others.

Variable types in the Bash shell are dynamic, meaning that the data type of a variable is not predefined and can change at runtime. Typically, variables in Bash are used to store string data, but they can also contain numbers and other data if they are appropriate for the operations and context. Variables are very useful for passing data between commands, controlling program logic, and interacting with the user.

7. The env, export, and unset commands in the Bash shell are used to control environment variables and shell variables. The main tasks of these commands are as follows:

1. env: The env command is used to display environment variables or execute a command in a specified environment. It displays a list of all environment variables available for the current process and their values.

An example of displaying all environment variables:

An example of executing a command in a specified environment:

2. export: The export command is used to create global (exported) variables that will be available to all processes running with the current shell. It is intended to export local variables to the global environment.

An example of exporting a variable:

After executing this command, the VAR variable will be available for all subsequent processes running with the current shell.

3. unset: The unset command is used to remove variables that have been created in the current process or globally. It is used to clean up variables so that they are no longer available.

An example of deleting a variable:

This command will delete the VAR variable from the current environment.

In general, the env command helps you view and manage environment variables, export is used to create global variables, and unset is used to delete variables. Using these commands, you can effectively manage variables in the Bash shell and control the behavior of your system.

8. In a Linux or Unix terminal, there are several ways to get help for commands. The main commands for this are:

1. man (manual): The man command is used to display the section of the help manual for a particular command.

For example, to get help for the ls command, you can run:

This command displays a detailed description of the ls command with information about its options and usage.

1. --help or -h: Many commands support the --help or -h options, which display a brief description of the command and a list of available options.

For example:

3. info: The info command provides access to documentation in Info format, which is similar to man pages but can be more detailed and interactive.

To get help for a command, you can run:

For example: 

After entering this command, you will see a detailed description of the ls command in Info format.

4. whatis: The whatis command displays a brief description of the command and its purpose.

For example:

1. apropos: The apropos command is used to search for commands related to the keyword or phrase you entered.

For example:

This command displays a list of commands associated with the "file" keyword.

These commands help users get detailed help and information about the available commands in the system.

**Хід роботи**

***The material was prepared by a student Dziubenko***

1.

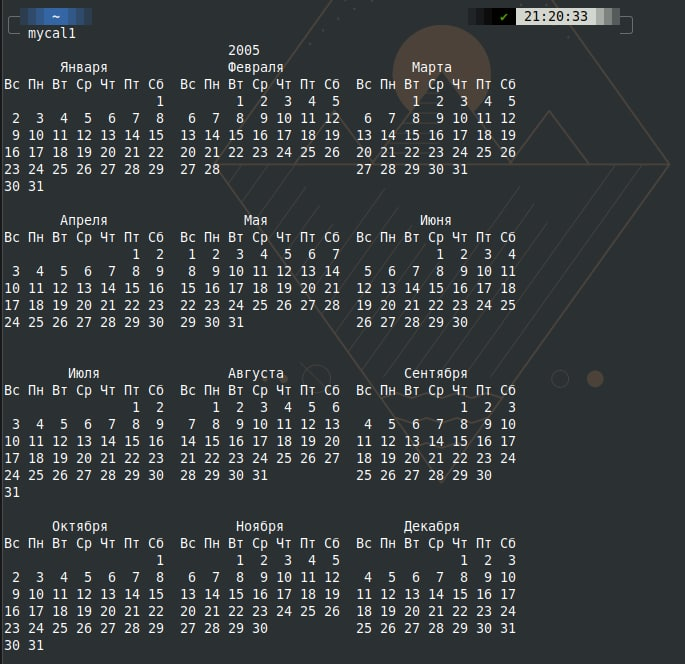
|  |  |
| --- | --- |
| uname | Displays system information. |
| man | Displays manual or help pages. |
| clear | Clears the terminal screen. |
| ls | Lists directory contents. |
| nano | A text editor for the terminal. |
| echo | Prints text or variables to the terminal. |
| bash | Executes the Bash shell. |
| alias | Creates command aliases. |
| source | Executes commands from a file. |
| rm | Removes files or directories. |
| touch | Creates empty files. |

2. Working in the terminal

2.1.



Fig 1. Working with variables



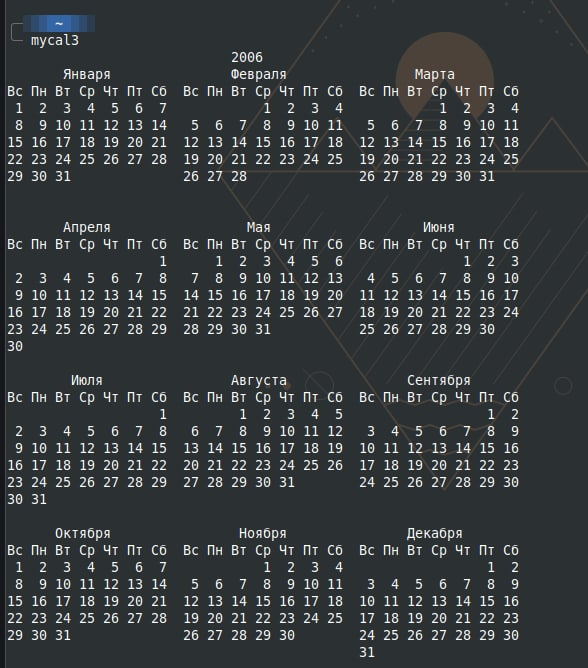
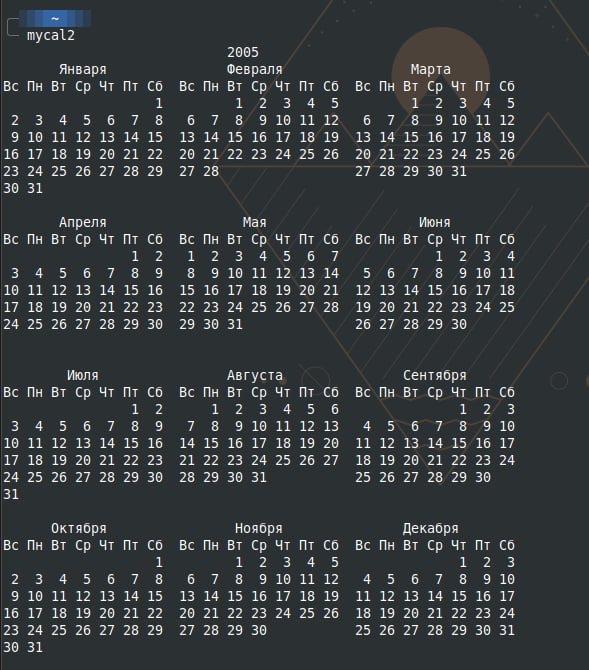


Fig 2. Working with variables

2.2.

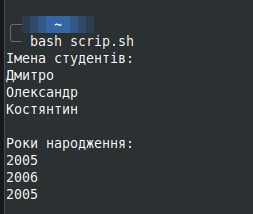


Fig 3. Working with functions in the terminal

2.3.

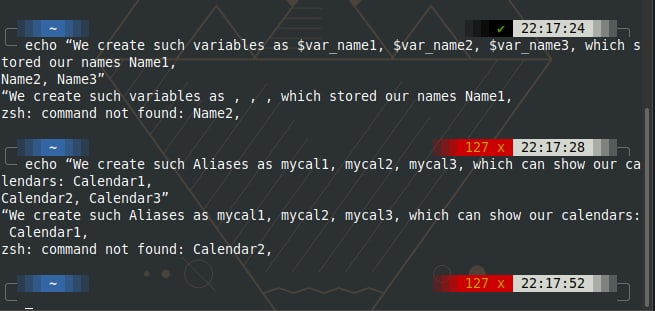


Fig 4. Working with quotation marks

2.4. For example, tasks 2.1 and 2.2 can be performed through control instructions without writing a separate function.

Here is an example of such code:

var\_name1="Дмитро"

var\_name2="Костянтин"

var\_name3='Олександр"

echo "$var\_name1"

echo "$var\_name2"

echo "$var\_name3"

alias mycal1="cal 2005"

alias mycal2="cal 2005"

alias mycal3="cal 2006"

mycal1

mycal2

mycal3

2.5.

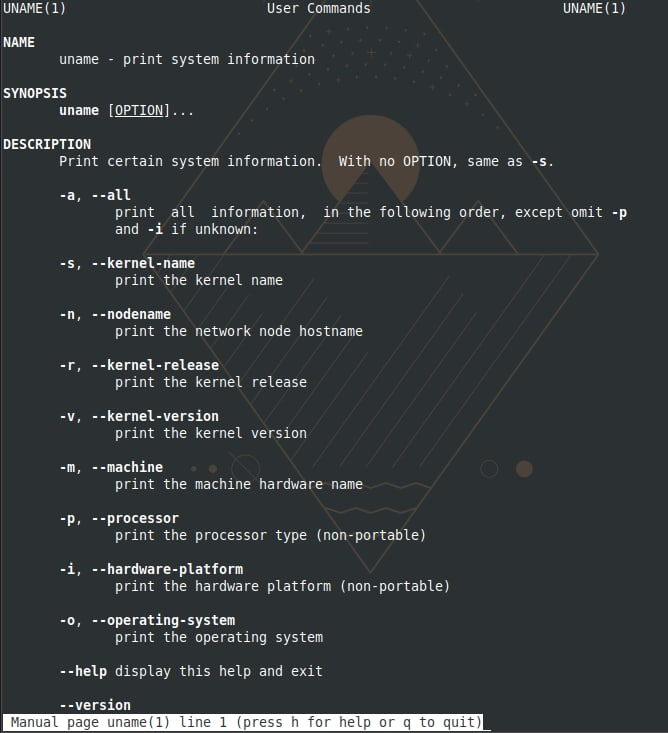
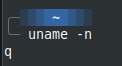
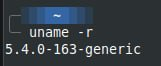
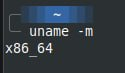


Fig 5. Work with help commands in the terminal



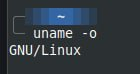
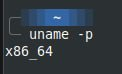


Fig 6. Work with help commands in the terminal exemples

**Контрольні запитання**

***The material was prepared by a student Storozhuk***

1. The following types of commands exist in the Bash shell:

* **Built-in commands:** Built into the shell itself, such as cd, echo.
* **External commands:** Found in executable files, e.g., ls, grep.
* **Scripts:** Shell language scripts that contain sequences of commands.
* **Alias:** Aliases for existing commands.
* **Functions:** Blocks of code that can be defined and called in the shell.

1. Environment variables are system variables that contain settings and information. You can view them in the terminal with the command echo $VARIABLE, where VARIABLE is the name of the variable. You can also print all environment variables with the env or printenv command.

3. The $PS1 variable in the Bash shell defines the format of the prompt string that is displayed before each command in the terminal. It can contain special sequences and variables that represent various information, such as username, hostname, current directory, and much more.

To view the contents of the $PS1 variable in the terminal, you can use the echo command:

bash

echo $PS1

This will display a string that defines the current format of the prompt string in your Bash shell.

1. To change the value of $PS1 for the current session, type:

bash

PS1="NEW\_VALUE"

To change the default value of $PS1, add this line to your shell's configuration file, for example, .bashrc for Bash.

1. Quotation marks in the Bash shell are used to:

* Protect strings with special characters and spaces.
* Protect variable values from being separated and substituted.

6. Control statements in the Bash shell are used to control the flow of command execution. They include:

* **Conditional statements:** Execute commands based on conditions, such as if, else, elif.
* **Loops:** Allow you to repeat commands, such as for and while.
* **Exit statements:** Control the exit from the current function or script, such as return and exit.
* **I/O redirection statements:** Control input and output streams, such as > and <.
* **Process instructions:** Allow you to control processes, such as kill and wait.
* **Shell instructions:** Run commands in new shells, such as ( ) and { }.

7. The difference between the $ and # characters at the end of a Bash shell query line is the identification of the current user and their role in the system:

* **$ (dollar):** If you see the $ symbol in the query line, it means that you are logged in to the shell as a regular (unregistered) user. Such a prompt shows that you have limited access rights and capabilities, and you do not have full control over the system.
* **# (hash character):** If the prompt contains the # character, it means that you are logged in as a superuser (usually called "root" on Unix/Linux systems) or you are running commands with elevated privileges (for example, using the sudo command). The # symbol indicates that you have full control over the system and the ability to perform dangerous actions.

8. whereis is designed to search for binary and reference files in standard system directories.

**locate** uses an index to quickly search for files and directories on the system in any directory.

**Висновки**

***The material was prepared by a student Storozhuk***

In this work, we have familiarized ourselves with basic CLI commands in linux, and basic text commands in terminal mode in different operating systems. We encountered a few errors, namely an error in task 2.3, because of this error, the data could not be displayed in the terminal, but we managed to solve it.