**“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 КСМ-13Б**

1. **Міньков Ілля**

**2. Колотуша Микола**

**Робота студентів групи КСМ-13б**

**Команда Defense of the Ancients: Міньков І. ,Колотуша М.**

**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.

**Хід роботи**

**Готував матеріал студент: Міньков І.**

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

In the Bash shell (Bourne Again Shell), there are several types of commands. Here are the main types of commands in the Bash shell:

Built-in Commands: These are commands that are built directly into the Bash shell and are executed without launching separate programs. Some examples of built-in commands include cd (change directory), echo (output text), pwd (print working directory), and export (set environment variables).

External Commands: These are commands that are represented as standalone executable files or programs located in system directories or other locations specified in the PATH variable. Examples of external commands include ls (list files), grep (search for text in files), and cat (concatenate and display file content).

Pseudo-Commands: These are special commands that are not executable programs but are used to control the shell or change its behavior. Examples of pseudo-commands include exit (exit the shell), source or . (execute a script within the current shell context), and history (display command history).

Variable Commands: The Bash shell allows the creation of variables that can be used within commands. For example, you can use a variable to store a directory path and then use that variable within a command.

Aliases: Bash permits the creation of aliases, which are short and custom command shortcuts for longer or complex commands. For instance, you can create an alias like alias ll='ls -l' to abbreviate the ls -l command.

These types of commands collectively provide a powerful interface for interacting with the Bash shell and the Linux or Unix operating system.

**Готував матеріал студент: Міньков І.**

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

Environment variables are special variables that store information about the configuration and settings of an operating system and its working environment. They are used to store information that can be utilized by programs and processes during their execution. Environment variables can define various aspects, including paths to system directories, usernames, language settings, shell configurations, and more.

Some common environment variables include:

PATH: Specifies the paths to directories where the operating system looks for executable program files. When you enter a command in the terminal, the system checks the paths defined in the PATH variable to find the corresponding executable file.

HOME: Indicates the path to the home directory of the current user.

USER: Contains the username of the current user.

LANG: Defines the language localization for program text output.

SHELL: Specifies the path to the shell used by the user.

TERM: Determines the type of terminal being used and affects terminal settings and text display.

To view the values of environment variables in the terminal, you can use various commands. For example:

To view all environment variables, you can use the printenv or env command:

To view the value of a specific environment variable, you can use the echo command followed by the variable name, either with or without the dollar sign:

These commands help you inspect information about environment variables that define the configuration and parameters of your shell and operating system.

**Готував матеріал студент: Міньков І.**

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

The $PS1 variable in the Bash shell defines the command prompt, which is displayed before each user-entered command in the terminal. It determines how the terminal prompt looks and can be customized to display various information, such as the username, hostname, current directory, and more.

Typically, the $PS1 variable is set in shell configuration files like .bashrc or .bash\_profile and can appear as a string containing various escape sequences and special characters. For example, a typical $PS1 value might look like this:

\u@\h:\w\$

Where:

\u displays the username.

\h displays the hostname (computer name).

\w displays the current working directory.

\$ displays the $ character (indicating a regular user) or # (indicating a superuser or administrator, if you are in a system with administrator privileges).

To view the contents of the $PS1 variable in the terminal, you can use the echo command with the variable name, either with or without the dollar sign:

echo $PS1

This command will output the value of the $PS1 variable, which defines how the terminal prompt is displayed.

**Готував матеріал студент: Міньков І.**

**4. 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 in the Bash shell, you can set a new prompt string using the PS1 variable. This can be done directly in the terminal for the current session or by making changes in one of the shell configuration files (e.g., .bashrc or .bash\_profile) to apply the changes by default every time you start a new shell session.

To change the $PS1 value for the current session:

PS1="new\_prompt\_string"

For example:

PS1="\u@\h:\w\$ "

This command will set a new prompt string that displays the username, hostname, current working directory, and the $ symbol at the end. The changes will immediately reflect in your current session.

To make changes to $PS1 by default, you should add the appropriate command to your shell's configuration file (e.g., .bashrc or .bash\_profile). This command will be executed every time you start a new shell session. For example, add the following line to your .bashrc file:

export PS1="\u@\h:\w\$ "

After making these changes and either restarting your shell or opening a new terminal, the new prompt string will be used by default in every new shell session.

**Готував матеріал студент: Колотуша М.**

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

In Bash shell scripting, both single quotes (' ') and double quotes (" ") are used to enclose strings, and they serve different purposes depending on which type you use:

**1.Single Quotes** (' '): Anything enclosed within single quotes is treated as a literal string. Variables and special characters within single quotes are not interpreted or expanded. For example:

bash

echo 'This is a literal string $HOME'

The result will be:

csharp

This is a literal string $HOME

**2.** **Double Quotes (" "):** Strings enclosed within double quotes allow variable substitution and command substitution. This means that variables and special characters within double quotes are expanded and their values are used. For example:

bash

echo "The value of HOME variable is: $HOME"

The result will be:

Javascript

The value of HOME variable is: /home/user

In summary, single quotes are used for creating literal strings, where variables and special characters are not interpreted, while double quotes are used for creating strings where variable and command substitutions are performed.

**Готував матеріал студент: Колотуша М.**

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

**Control statements in programming are used to control the flow of execution in a program. They allow you to make decisions, specify conditions for executing certain code blocks, and perform iterations (repetitions) of specific operations. The main types of control statements include:**

**1.Conditional Statements:**

**if**: Executes a specific code block if a given condition is true.

**else**: Executes a specific code block if the condition in the if statement is false.

**elif** (else if): Allows you to check additional conditions if the previous conditions are false.

**switch** (case): Used to handle different cases or choices based on the value of an expression.

**2.Looping Statements (Iteration):**

**for**: Used to iterate through a list of items.

**while**: Executes a code block as long as a specified condition is true.

**do-while**: First executes a code block and then checks the condition.

**3.Control Statements for Breaking and Exiting Loops:**

**break**: Used to exit a loop prematurely.

**continue**: Skips the current iteration of a loop and moves to the next iteration.

**4.Function and Subroutine Call Statements:**

**return**: Used to return a value from a function.

**function**: Defines a new function for later use.

**5.Exception Handling Statements:**

**try**, **catch**: Used to handle exceptions or errors in code.

These control statements allow programmers to manage the program's logic, make decisions based on conditions, perform actions in loops, and handle exceptional situations. They are fundamental for building more complex and functional programs.

**Готував матеріал студент: Колотуша М.**

**7. В чому різниця якщо в кінці рядку запрошення bash стоїть символ $ чи #?**

In the context of the Bash shell, the symbols $ and # at the end of the command prompt (prompt string) have different meanings and indicate different user modes:

**1.$ (Dollar) Prompt**:

When the $ symbol appears at the end of the command prompt, it typically indicates that you have entered the Bash shell as a regular user.

A regular user has permissions to execute most commands but does not have the authority to make changes to system settings or perform actions that could impact the system's stability.

**2.# (Hash) Prompt**:

When the # symbol appears at the end of the command prompt, it usually indicates that you have entered the Bash shell as a superuser or system administrator (root).

A superuser has full access to the system and can execute any commands, make changes to system files and settings, and perform administrative tasks.

Therefore, the difference between $ and # at the end of the prompt string lies in the level of user privileges. $ indicates a regular user, while # indicates a superuser or system administrator with full privileges.

**Готував матеріал студент: Колотуша М.**

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

The **whereis** and **locate** commands are used for searching files and programs on a Linux system, but they have different methods of searching and are used for different purposes:

**1.whereis** **Command:**

The **whereis** command is used to locate executable files, manuals, and related documentation associated with a command or program on the system.

It uses a database created by the system for quick searching and returns paths to executable files, manual pages, and source code.

Example usage of **whereis**:

bash

whereis ls

The result may look like:

bash

ls: /bin/ls /usr/share/man/man1/ls.1.gz

**2. locate Command:**

The **locate** command uses an indexed database for fast searching of files and directories on the system.

It searches for file and directory paths based on keywords or regular expressions and can be faster than **whereis** for searching large numbers of files.

Example usage of **locate**:

bash

locate myfile.txt

The result may look like:

bash

/home/user/myfile.txt

/var/www/myfile.txt

The main difference between them lies in the search method and speed. **whereis** uses the system's database and searches for executable files and documentation, whereas **locate** uses an indexed database for quick searching of files and directories. **locate** can be faster for broad searches but may require updating its database, while **whereis** uses up-to-date system information.

Висновок:

The objective of this work was to introduce students to basic command-line interface (CLI) commands in Linux and familiarize them with basic text commands in terminal mode in different operating systems. The following materials and tools were used for the lessons:

IBM PC-type computer.

Windows-based operating system (Windows 7).

Virtual machine - VirtualBox (Oracle).

GNU/Linux operating system - CentOS.

The goal was to provide students with practical experience in using the command-line interface in the Linux operating system and to teach them fundamental commands for file system navigation, file and directory creation, copying, moving, and working with text files. Additionally, students had the opportunity to compare the use of the command line in different operating systems and learn how to work with the terminal in a virtual machine.

Through this work, students acquired important skills for working in command-line mode, which can be valuable in their future work with Linux operating systems and other text-based interfaces.