“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №3**

з дисципліни: «Операційні системи»

**Тема: «Знайомство з базовими командами CLI-режиму в Linux»**

Виконали

Студенти групи КСМ-03Б

Команда: Кучмій-Зікеєв

М.О, Зінченко І.О. та

Звєрьков А.Г.

Перевірив викладач

Сушанова В.С.

Київ 2022

**Завдання для попередньої підготовки.**

***Виконав студент Зінченко І.***

4. Дайте визначення наступним поняттям:

***Command interpreter***

A command interpreter is a computer program, a part of the operating system, which provides basic computer management capabilities by means of interactive input of commands through the command line interface or sequential execution of batch command files. As a rule, its functions are limited to providing the user with the opportunity to run other programs, it can also contain some basic input-output commands and its own simple script programming language. The COMMAND.COM command interpreter is included in the MS-DOS and Windows 95 operating systems, in Windows NT it is cmd.exe, in OS/2 the command interpreter is also called cmd.exe, the most common command interpreter in Linux and FreeBSD is bash, in addition to which there is a large family of others.

***Shell***

In computing, the term shell is used to refer to those programs that provide a user interface for accessing operating system services. This can be graphics or plain text, depending on the type of interface they use. Shells are designed to facilitate the way to call or run the various programs available on the computer. It should be noted that there are 2 types of shell and these are: Common text shells like bash, emacs, windows command line, among others. Common charting shells like GNome, KDE, XFCE, LXDE, Unity, MacOS Desktop Environment, Windows Desktop, etc.

***Command***

A command is an order to a computer program to act as an interpreter to solve a problem. In the more general case, a command is an indication of a command line to some interface (such as a command shell). In particular, the term command is used in imperative programming languages. These languages ​​are named so because their operators are usually written like the imperative verb used in many natural languages. If we look at the imperative language operator as a natural language sentence, then the command, in general, is similar to a verb.

5. Дайте відповіді на наступні питання:

**Prompt** - specifies what text will be visible in the prompt line and changes the prompt in the Windows command line.

The command is broken down into an array of strings with arguments. Argument 0 is (usually) the name of the command, argument 1 is the first element following the command, and so on. These arguments are sometimes called positional parameters. The parameter is a documented 1 type of argument that changes the behavior of the command, for example, -lobychno means "long", -vpodrobny. -lv two options combined into one argument. There are also long parameters, for example --verbose (see also Using getopts for processing long and short parameters of the command line). As their name suggests, parameters are usually optional. However, there are several commands with paradoxical "required parameters".

**ls** — List directory contents. ls is probably the most common command. A lot of times, you'll be working in a directory and you'll need to know what files are located there. The ls command allows you to quickly view all files within the specified directory. Syntax: ls [option(s)] [file(s)]. Common options: -a, -l

It tracks all actions in the command line mode, which allows us to easily call previous commands. We don't need to retype all the long commands we typed before. We can simply go back to the history of the command line and use the "ex" commands in the command line. To view bash history, use the history command. This prints out our last commands, one command at a time. This should output no more than the number of strings you selected for the HISTSIZE variable. At the moment, it will probably be less. To find all history commands that contain a particular string, you can send the results to a grep command that looks for the specified string in each string. There are many situations where being able to get a list of commands you've previously run can be useful. If you want to run one of these commands again, you can copy one of the commands from your output and paste it into your prompt. This works, but bash comes with several shortcuts that allow you to extract and then automatically execute commands from your history.

**echo -** Outputs text to the terminal window. Echo outputs text to a terminal window and is commonly used in shell scripts and batch files to print status text to the screen or computer file. Echo is also especially useful for displaying the values of environment variables that tell the shell how to behave when the user is working on the command line or in scripts. Syntax: echo[options] [strings] General options: -e, -n

As we've already seen, Bash understands many types of variables and parameters. Until now, we didn't care about the types of variables we were operating on, so our variables could hold any values we assigned to them. However, there may be situations where this behavior of variables is undesirable, for example when processing telephone or other numbers. In addition to numbers and variables, you can declare constant variables. This is often done at the beginning of the script, when the constants are assigned their values. Further, we see only a reference to the name of the constant, that is, if the constant needs to be changed, it can be done only once. Also, a variable can consist of a set of variables of any type, these are so-called arrays of variables. As we mentioned above, a variable is a temporary store of information. The following are two things you can do when using a variable in a bash script: Set a variable to a specific value. Reading the value of a variable. You can evaluate variables in different ways. The most common is to set a value directly to a variable, or you can set its value as a result of command or program processing. When reading a variable, we need to place its name with a $ sign at the beginning of the variable you might want to read. Before executing each line of a Bash script, it first checks for variable names. It finds each variable and replaces its value with the variable name. It then starts executing a particular line of code and repeats the process for the next line. There are a few important syntax points to keep in mind when reading a variable: Do not use special characters or the $ sign when setting a variable value. When reading a variable, put a $ sign at the beginning of the variable name. Some programmers capitalize variable names, but we can assign names as we like. All of them can be large, small or a mixture of them. You can set and read data from a variable through the terminal as follows: First, you need to open a terminal program on your system. Open the terminal using the keyboard shortcut "Ctrl + Alt + t". Or you can open it using the application launcher search bar. Click on "Activities" and then type "terminal" in the search bar that will appear on the desktop and press "Enter". You will see the following output on your terminal screen. Click on the terminal icon and launch it.

The ***env*** command allows you to change the environment in which programs are running,passing a set of variable values to the command: env VAR1="blahblah" command\_to\_run command\_options. ***Export*** is one of the built-in commands of the bash user shell and is intended for exporting variables and functions of the current process to a child process. In practice, the export command is used as the main means of determining the settings of specific applications. ***Unset*** is a built-in bash command that unsets all variables or functions in the name list. The unset command cannot be applied to the PATH, PS1, PS2, MAILCHECK, and IFS variables.

**Хід роботи.**

**1.**

|  |  |
| --- | --- |
| Назва команди | Її призначення та функціональність |
| ls | Is used to list information about directories and files. |
| ls -1 | Displays information from command ls in the long format. |
| ls -l /tmp | Adding the location of a specific directory to the ls command will list information for that directory. |
| whoami | The following command will display the same information that you see in the first part of the prompt. The output of the whoami command displays the user name of the current user. |
| uname | Displays information about current system. |
| pwd | This command is used to display current location or current working directory. (print working directory). |
| history | This command is used to view numbered command history list. |
| clear | Clears the terminal. |
| echo | Can be used to print text and the value of a variable. |
| which | Used to to determine if there is and executable file. |
| type | Can be used to determine information about command type. |
| man | Opens access to manual page to learn more about commands. |
| date | Shows current date and time |
| locate | An easy way to search for a file. |
| whereis | Shows where the command is located. |

**2.**

2.1

2.2

2.3

2.4

2.5

**Контрольні запитання**

***Виконав студент Зінченко І.***

***1)*ls — List directory contents**. ls is probably the most common command.

**echo — Prints text to the terminal window**. echo prints text to the terminal window and is typically used in shell scripts and batch files to output status text to the screen or a computer file.

**touch — Creates a file**. touch is going to be the easiest way to create new files, but it can also be used to change timestamps on files and/or directories.

**mkdir — Create a directory**. mkdir is a useful command you can use to create directories.

**grep — search**. grep is used to search text for patterns specified by the user.

**man — Print manual or get help for a command**. The man command is your manual and is very useful when you need to figure out what a command does.

**pwd — Print working directory**. pwd is used to print the current directory you’re in.

**cd — Change directory**. cd will change the directory you’re in so that you can get info, manipulate, read, etc.

**mv — Move or rename directory**. mv is used to move or rename directories.

**rmdir — Remove directory**. rmdir will remove empty directories. This can help clean up space on your computer and keep files and folders organized.

**locate — Locate a specific file or directory**. This is by far the simplest way to find a file or directory.

**less — view the contents of a text file**. The less command allows you to view files without opening an editor.

***2)*** Sharpening changes in operating systems based on the Linux kernel are those changes that should replace textual information beaten by other programs at the time of launch. They seem to include basic system options, both graphical and command shell, music settings data, overwriting song files, and more. The values ​​of such variables are specified, for example, by numbers, symbols, paths for current or files. Zavdyakov, which helpless addons automatically remove access to the main settings, and also have the ability to change or create new options for the koristuvach. Each user has his own home directory where all important objects are stored, including configuration files for user variables. It is already clear from their name that they are applied to a specific user at the time when he is authorized through the local "Terminal". They also work when connected remotely. You can easily view all system and user variables present in Linux and their definitions with just one command that will display the list. For this, you will need to perform just a few simple actions through the standard console.

***3)*** PS1 is the main prompt variable that contains bash special characters \u@\h \W\\$. This is the default bash prompt structure that is displayed every time a user logs in via a terminal. These default values are set in the /etc/bashrc file.

***4)*** The prompt is managed through a special shell variable. You need to set the variable PS1, PS2, PS3 and PS4. If set, the value is executed as a command before issuing each main prompt. PS0 – The value of this parameter is expanded (see PROMPTING below) and displayed by interactive shells after reading the command and before its execution. PS1 – The value of this parameter is expanded (see PROMPTING below) and is used as the main prompt string. The default value is \s-\v\$ . PS2 – The value of this parameter is expanded, as in the case of PS1, and is used as an additional prompt string. By default > PS3– The value of this parameter is used as a hint for the selection command. PS4 – The value of this parameter is expanded, as in the case of PS1, and the value is printed before displaying each bash command during the execution trace. If necessary, the first PS4 symbol is replicated several times to indicate several levels of indirection. The default is +. $PS1 is an environment variable that tells your shell how to format the invitation. Changing the value using export, like you, applies only to this session; so you can just open a new shell (not a nested shell) and see the familiar prompt.

***5)*** Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to disable special treatment for special characters, to prevent reserved words from being recognized as such, and to prevent parameter expansion. Each of the shell metacharacters has special meaning to the shell and must be quoted if it is to represent itself. When the command history expansion facilities are being used, the history expansion character, usually ‘!’.

***6)*** Operating instructions are used to better familiarize the user with the product or technique. Also, the instructions are used when problems arise during use, or in order to learn about the greater functionality of the device.Depending on the nature and time of the briefings, they are divided into types: introductory; primary; second; unplanned; targeted

***7)*** In short, if you see a dollar sign ($) or a grid (#) on the screen to the left of the blinking cursor, you are in a command line environment. The $, #, % symbols indicate the type of user account you are logged into. The dollar sign ($) means that you are a regular user. a grid (#) means that you are a system administrator (root). In shell C, the invitation ends with a percent sign ( %). Different Unix or GNU/Linux distributions have differences in prompts due to their default settings. For example, the invitation for Debian/Ubuntu is guest@linux:~$, for Fedora/CentOS/RedHat — [guest@linux ~]$, and for SuSE Linux/OpenSUSE — guest@linux:~>. As a general rule, the prompt usually shows the login user name, the hostname of the computer, and the current working directory, and ends with a dollar sign ($), a percent (%) or a hash (#

***8)*** The whereis command searches for and returns the location of binaries, source code, and help pages for a program or command. On the other hand, the locate command is used to scan the entire system for a file with a given name. It reads one or more databases prepared by the updatedb command to find files with the specified name. The updatedb command is periodically launched to index files in the system. Because the locate command uses a database, the results will not include files added after the last database update, but may display files deleted after the event.