“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №2**

з дисципліни: «Операційні системи»

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

Виконавли студенти

групи РПЗ-03

Команда: Губенко Є.О.,

Заїка С.В. та Кресан Р.А.

Перевірив викладач

Сушанова В.С.

Київ 2022

**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 Заїка С.В.***

1. Дайте визначення наступним поняттям:

- Command prompt

A command prompt is a program that allows a user to interact with an operating system using a text-based interface. It allows the user to execute various commands, such as creating, copying, moving or deleting files and directories, changing system settings, and much more.

- Shell

A shell is a computer program that provides a user (a person or another program) with access to the services of an operating system.

- Team

A command is a program name that a user enters in the terminal to perform a specific task. Commands can have additional data and parameters

1. Дайте відповіді на наступні питання:

- What basic information does a prompt line provide?

The prompt bar provides the user with a text message that they must read and then enter data into the input field. The prompt string may contain hints for the user, such as the type of data they should enter.

- Why does the command need parameters and arguments?

Parameters and arguments are used to give a command additional parameters that will be used to perform an action. For example, the "ls" command can take parameters and arguments such as "-l" and "/tmp" to display detailed information about the files in the /tmp directory.

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

`ls' is a terminal command to display a list of objects in the current directory. It can have parameters and arguments that allow users to display detailed information about the objects in the directory.

*Examples:*

*1. `ls -l` - display detailed information about all objects in the current directory in the form of a list.*

*2. `ls -a` - list all objects in the current directory, including hidden objects.*

*3. `ls /home/user` - display a list of objects in the /home/user directory.*

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

Command history allows users to reuse commands they have previously used to save time. This allows users to use the system more efficiently. In addition, command history allows users to reuse commands that they have forgotten. This allows users to use the system more efficiently and save time.

- What is the purpose of the echo command?

The echo command is used to display text strings or variables on the screen. It can be used to display messages when a script is run, or to display the results of executing commands.

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

Variables in the Bash shell are special concepts that allow you to store data for later access. A variable can be defined or assigned a value. The Bash shell supports such types of variables as logical variables, string variables, integer variables, arrays, and environment variables.

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

The env command is used to display environment variables and their values. The export command is used to add environment variables to the list of available variables for the program. The unset command is used to remove environment variables from the list of available variables for the program.

- What commands do you know to get help on commands in the terminal?

1. man - used to get help on any command in the terminal.

2. help - used to get help for any command in the terminal.

3. info - used to get help for any command in the terminal.

4. apropos - used to find help for commands that match the specified search query.

**Хід роботи**

***Готував матеріал студент Заїка С.В.***

1. Work through all the sample commands presented in the NDG Linux Essentials labs - Lab 5: Command Line Skills and Lab 6: Getting Help.

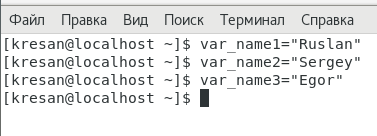
|  |  |
| --- | --- |
| Command name | Purpose and functionality |
| ls | Displays information about directories and files. By default, without arguments, displays information for the current directory |
| ls -l | Using the -l option in the ls command allows you to display information about files located in the current working directory in a long format that provides more extensive additional information |
| ls -l /tmp | Using the /tmp argument in conjunction with the -l option in the ls command allows you to display detailed information about the files in the /tmp directory. |
| whoami | It helps to define user access rights to files and directories, as well as to solve problems with access to resources and ensure security in the operating system. |
| uname | It displays the name of the operating system, its version, and the type of processor it is running on. This command is useful for diagnosing problems with the operating system and determining the compatibility of applications with different OS versions. |
| uname -n | Displays the name of a node or computer on the network. It allows users to identify computer identifiers on the network and helps in establishing network connections and connections to remote computers. |
| uname --nodename | Displays the name of a node or computer on the network. This command is an alternative to the shortened "uname -n" command. |
| pwd | Displays the current working directory or the path to the current working directory. It helps users to determine the current location in the file system and navigate between different directories. This command is useful for performing routine tasks in the terminal and automating work with files and directories. |
| echo | Prints a text string to the screen or to another output stream. |
| history | Displays the history of commands entered in the current terminal session. It allows you to view previous commands and their parameters that were entered in previous sessions and repeat them again without having to enter them again. |
| history 5 | Displays a list of the last five commands that were entered in the current terminal session. It allows you to view only the most recent commands from the input history, instead of the full list of all entered commands. |
| !9 | Executes the ninth command in the command history. It allows you to re-execute previous commands without having to enter them again. |
| echo $HISTSIZE‌ | Displays the current value of the HISTSIZE environment variable. The HISTSIZE variable determines the maximum number of commands that will be stored in the command history. |
| echo $PATH | Displays a list of paths to directories containing executable files. The PATH variable contains colon-separated paths to directories where the operating system searches for executable files. |
| which date | Displays the path to the executable file for the date command. The command is useful for determining the location of the executable file for a specific command in the system. |
| type command | It is used to determine the type of command (built-in, external, or alias) specified in the command line arguments. It allows you to find out exactly how the command is executed in the terminal and helps to solve problems related to unknown commands or name conflicts. |
| which ls | Displays the path to the executable file for the ls command. The command is useful for determining the location of the executable file for a specific command in the system. |
| alias | Used to create a shortened version of a command that consists of a more complex sequence of commands or parameters. After using the "alias" command, the user can create an abbreviation for any command or sequence of commands and call them using the new abbreviation. |
| echo Today is `date` | Displays the current date and time on the screen. |
| echo D\* | Displays a list of all files and folders in the current directory whose names begin with the letter "D". The "\*" character is used to match any string after the letter "D", so any file or folder name beginning with "D" can be listed. |
| false | Always returns the value 1, which means that the command was not executed successfully or that an error occurred. |
| echo Start && echo Going && echo Gone | Outputs three messages "Start", "Going", and "Gone" sequentially as standard output. |
| echo Success && false && echo Bye | Prints the message "Success" to the standard output, then executes the command "false", which always returns "false", that is, it is not executed successfully, so the next command "echo Bye" is not executed. |
| false || echo Fail Or  true || echo Nothing to see here | The `false || echo Fail Or` command is intended to print the string "Fail Or" only if the previous `false` command fails.  The command `true || echo Nothing to see here` is intended to print the string "Nothing to see here" only if the previous command `true` is successful. |
| date | Displays the current date and time in a format that can be customized using command line parameters. |
| man date | is used to display the man page for the `date` command on Unix-like operating systems. This page contains detailed information about the syntax and options for the `date` command, as well as examples of how to use it. |
| man -k password | is designed to search the Linux manual by the keyword "password". The functionality is that it allows you to find all documents (including user manuals) that contain the word "password" in their description. |
| man -f passwd | displays a brief description of the help page associated with the `passwd` command. The `-f` option specifies that only a short description is displayed instead of the full help page. The `passwd` command is used to change a user's password on Linux systems. |
| man 5 passwd | is designed to display information from Chapter 5 (File Formats and Standards) of the Linux User's Guide about the `/etc/passwd` file This file contains information about system users, including their names, home directories, and other account details. |
| info date | Відображає інформацію про команду "date" в форматі Info. Інформація включає опис опцій, синтаксис та приклади використання команди "date" в Unix-подібних операційних системах. |
| date --help | displays a brief summary of the options and formats supported by the `date` command Its purpose is to provide the user with a brief description of the available options and formats of the `date` command |
| ls /usr/share/doc | displays a list of files and folders containing documentation for installed packages on the system in the "/usr/share/doc" directory. Its purpose is to view information about the programs that are installed on the computer. |
| locate crontab | The "locate" command searches for paths to files and directories using a database that stores information about the file system. In this case, the "locate crontab" query searches for the path to any file or directory with the word "crontab" in its name or path. |
| locate -b "\crontab" | searches for a quick database connection with the specified `crontab` expression and returns the path to any file that contains this expression in its name. The `-b` parameter provides an exact match only to the base file name (without the path). |
| whereis passwd | is designed to search for executable files, binaries and help files associated with a given file name. In this case, `whereis passwd` searches for the location of the executable, data and help file of the `passwd` command |

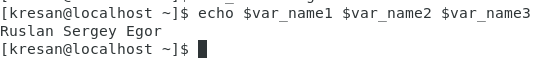
***Готував матеріал студент*** ***Кресан Р.А.***

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

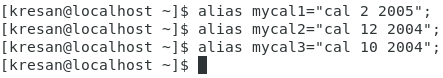
2.1. Working with variables and aliases in the terminal:

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



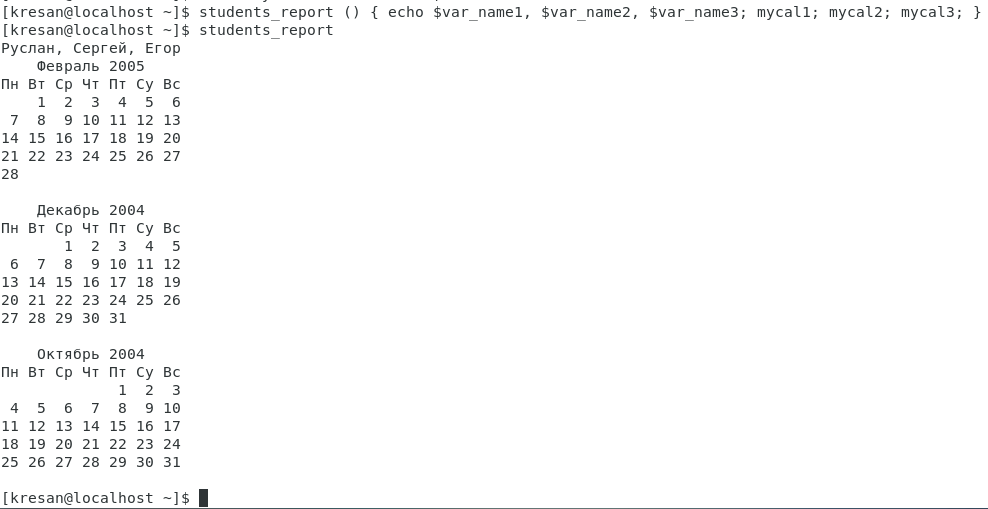
- Use the echo command to print the names of the students in your team

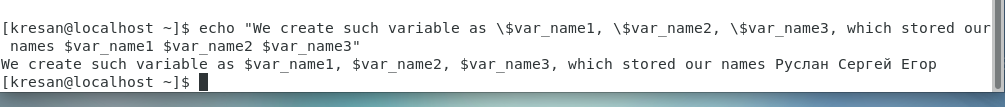
- Create aliases mycal1, mycal2, mycal3 for the cal command to automatically display the calendar of your year of birth



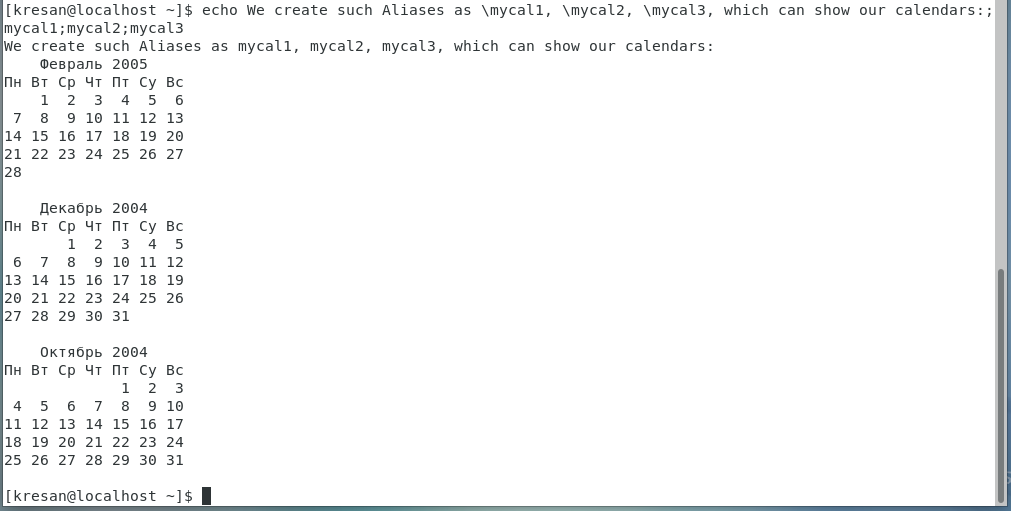
2.2. Робота з функціями (Functions) в терміналі:

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



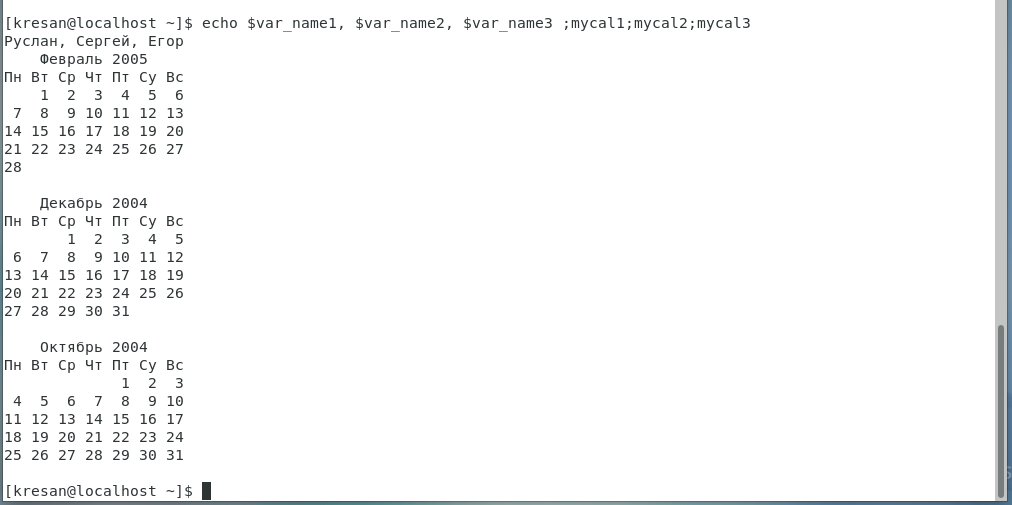
2.3. Work with quotation marks (Quoting) in the terminal. Print the following sentences on the command line:  
- "We create such variables as $var\_name1, $var\_name2, $var\_name3, which stored our names Name1, Name2, Name3" (in the sentence, first display the names of the variables, and then their contents)

- “ We create such Aliases as mycal1, mycal2, mycal3, which can show our calendars: Calendar1, Calendar2, Calendar3" (in the sentence, first print the name of the alias command, then print 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 through control instructions without writing a separate function, how will it look like?



2.5. Робота з командами довідки (Man Pages) в терміналі:.

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

|  |
| --- |
|  |
| Using the command:  -s, (-kernel-name) - prints the kernel name.  -n, (-nodename) - prints the name of the system node (hostname). This is the name that the system uses when communicating over the network.  -v, (-kernel-version) prints the kernel version.  -p, (-processor) prints the processor architecture.  -o, (-operating-system) - prints the name of the operating system.  -a, (-all) - when using the -a option, uname behaves the same as if the -snrvmo option were specified. |

**Відповіді на контрольні запитання**

1. What types of commands exist in the Bash shell?

*The Bash shell supports various types of commands, including the following:*

*Built-in commands: these are commands that are built into the Bash shell itself, such as cd, echo, pwd, exit, etc.*

*External commands: These are commands that are not built into the Bash shell, but are found in separate programs that can be run from the command line. Examples include ls, cat, grep, etc.*

*Alias: short names that you can create for long commands to make them easier to use in the future.*

*Functions: These are code snippets that you can write yourself and use instead of full commands. Functions are typically used to perform complex tasks that require multiple steps.*

*Aliases: Another form of aliases that allow you to create short names for commands that can be entered from anywhere in the shell. They are valid no matter where the user is in the shell.*

*These types of commands can be used individually or in combination to create complex commands and scripts that perform different tasks in the Bash shell.*2. What are environment variables? What are they? How can they be viewed in the terminal?

*Environment variables are variables that are set in the operating system and are available to all processes running in the system. Environment variables contain information about the system, shell settings, paths to executable files, and more.*

*There are several environment variables that are standard on most Unix-like systems, such as:*

*HOME: the path to the user's home directory*

*PATH: paths to executable files*

*USER: the name of the user under which the process is running*

*SHELL: the path to the shell used by the user*

*TERM: the type of terminal used for output*

*Environment variables can be viewed in the terminal using the "printenv" or "env" command. For example, the "printenv PATH" command displays the paths to executable files that are available in the current environment. You can also use the "echo" command to display the values of individual variables, for example, "echo $HOME".*

3. Describe the variable $PS1. How to view its contents in the terminal?

*The $PS1 variable is one of the environment variables in the Bash shell that is responsible for the contents of the command line prompt. This variable is used to customize the appearance of the command line in the terminal and can contain various special characters and control sequences that display information about the current state of the shell, such as the current path, username, host system, etc.*

*An example of the value of the $PS1 variable:*

*export PS1='\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '*

*In this example, the content of $PS1 consists of the following components:*

*\u - user name*

*\h - the name of the host system*

*\w - current path*

*$ - command line prompt character ($ for a regular user, # for a user with root privileges)*

*control sequences with colors (in this case, green for the username and host, and blue for the current path)*

*To view the contents of the $PS1 variable in the terminal, you can use the command "echo $PS1". This will display the current value of the $PS1 variable in the terminal.*

4. How can I change the value of the $PS1 variable? What will happen in the bash prompt line (the prompt line before each command)? How to change the value of this variable not for the current session, but by default?

*You can change the value of the $PS1 variable using the export command. For example, if you want to change the value of the $PS1 variable to a new value, you can enter the following command in the Bash terminal:*

*export PS1="new value"*

*Changing the value of the $PS1 variable will change the display of the prompt line before each command in the Bash terminal. The new value can contain any special characters and control sequences that are responsible for the appearance of the command prompt.*

*To change the default value of the $PS1 variable, you can add the command export PS1="value" to the shell configuration file (for example, .bashrc). This will save the new value of the $PS1 variable persistently, meaning that it will be used by default every time the shell is launched in the future.*

5. What are quotation marks used for in the Bash shell?

*Quotation marks (single and double) in the Bash shell are used to delimit namespaces and to pass strings of special characters as a single argument.*

*Single quotes (') are used to delimit namespaces. If you use single quotes, variables in the middle of a string will not be interpreted, i.e., the value of the variables will not be substituted into the string, but will be considered plain text. For example:*

*echo '$USER'*

*The string $USER will be displayed, not the username, as in the case of double quotes.*

*Double quotes (") are used to pass strings with special characters as a single argument. In the middle of the string, the values of variables will be interpreted and substituted into the string. For example:*

*echo "Hello, $USER"*

*This will display the string Hello, username.*

*It is important to note that if the string contains characters that have a special meaning for the shell (for example, a space, semicolon, etc.), the string must be enclosed in double quotes. It is important to keep in mind that the double quote character also has a special meaning, so it should be escaped with a backslash (\).*6. What are control instructions used for and what types of instructions do you know?

*Control instructions are used in programming to change the behavior of a program. They allow you to control the execution of a program, including conditional statements, loops, and jumping to other parts of the program.*

*The main types of control statements include:*

*Conditional statements:*

*if-else: allows you to execute one block of code if the condition is true and another block of code if the condition is false.*

*switch: allows you to check whether the value of a variable is equal to one of a list of specified values.*

*Loops:*

*for: allows you to repeat the execution of a code block a specified number of times.*

*while: allows you to repeat the execution of a code block as long as the condition is true.*

*do-while: allows you to repeat the execution of a block of code at least once, and then continue to repeat as long as the condition is true.*

*Jump instructions:*

*break: allows you to stop the execution of the loop or go to the next statement after the switch code block.*

*continue: allows you to go to the next iteration of the loop.*

*goto: allows you to jump to a label in the code that is located somewhere else in the same function.*

*Control statements can greatly facilitate the programming process and make the code more understandable and structured. However, their use should be limited and justified, as excessive use can make the code more complex and less understandable.*

7. What's the difference if the bash prompt line ends with a $ or # character? For example, on the screen we see the following entries  
  




*The entry [centos@localhost Desktop]$ means that a user named "centos" is running a bash shell with normal access, since the "$" character is at the end of the prompt.*

*The entry [root@localhost Desktop]# means that the user named "root" is working in the bash shell with elevated privileges, since the "#" character is at the end of the prompt.*

*So, the difference between the $ and # characters at the end of the prompt line is the user's access level in the bash shell: the $ character indicates a normal access level, while the # character indicates an elevated access level.*

8. What is the purpose of the whereis and locate commands? What is the difference between them?

*The whereis and locate commands are used to search for files and programs in the operating system.*

*The whereis command searches for program files in the specified system directories and returns the path to the executable, documentation, and program code. For example, running the whereis bash command will return the path to the bash executable, the documentation file, and the program code file.*

*The locate command, on the other hand, searches the system's file database for files rather than specific directories, which can speed up the search considerably. The search is fast, but the database must be updated regularly to contain up-to-date information about system files.*

*So, the difference between whereis and locate is that whereis searches for program files in the specified directories, while locate searches for files in the system file database. The second command can be faster, but it requires updating the database, which can take some time.*

**Conclusions.**In the course of the lab, we learned the basic commands of CLI mode in Linux and basic text commands in terminal mode in different operating systems.