“Kyiv specialized College of Communications”

Commission of computer engineering

**REPORT ON THE IMPLEMENTATION**

**LABORATORY WORK №3**

From the discipline: "Operating systems"

**Topic: "Introduction to basic CLI commands**

**in Linux"**

The students

performed Groups RPZ-03

Team 3:

Kanavets K.S.,

Kryvenko A.I.,

Kulikovska M.V.

Checked by the teacher

Sushanova V.S.

Кyiv 2023

***The material was prepared by student Kanavets Kateryna (kanavetsk)***

1. Work through all of the sample commands presented in the NDG Linux Essentials labs - Lab 5: Command Line Skills and Lab 6: Getting Help. Create a table to describe these commands\*\*\*.

|  |  |
| --- | --- |
| Command name | Its 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 | Returns information about the current user executing the command. The functionality of "whoami" is that it returns the username that is displayed in the operating system. |
| uname | Returns information about the operating system you are running on. The functionality of "uname" is that it returns a variety of information about the system, including the operating system name, kernel version, and processor architecture. |
| uname -n | Displays only the host name of the computer on which the command is executed. |
| uname --nodename | The "uname --nodename" command is an alternative version of the "uname -n" command and displays the name of the computer node on which the command is being executed in the terminal. The "uname --nodename" command can be useful for obtaining the name of a computer node in scripts or programs running in the terminal. |
| pwd | Displays the path (absolute path) to the current working directory in the terminal. |
| history | The command displays a list of the last commands used in the current terminal or command line session. |
| echo | The command is used to output a specified text string or variable value to the terminal or command line. |
| echo $PATH | The "echo $PATH" command displays a list of paths containing executable files that can be executed from anywhere on the command line. |
| which date | The "which" command is used to find the path to an executable file in the operating system. The "which" command takes an argument - the name of the executable file and returns the full path to this file. |
| type command | The "type" command is used to determine the type of command depending on how it is interpreted by the system. The functionality of the "type" command is that it can specify how the operating system interprets a given input string (command). |
| which ls | The "which" command is used to determine the path to the executable file for the specified command in the operating system. For example, executing the "which ls" command may return "/bin/ls", which means that the executable for the "ls" command is in the "/bin" directory. |
| type vi | The "type" command is used to determine the type of command in the operating system. Executing the "type vi" command will show that "vi" is a built-in command that is a text editor. |
| echo Today is `date` | The "echo Today is date" command displays the current date and time in an easy-to-read format. The "date" command is used to display the current date and time in the operating system, and the "echo" command is used to display a text message in the terminal. |
| echo Today is $(date) | The command "echo Today is $(date)" performs the same operation as "echo Today is date". Using "$(date)" instead of "date" allows you to execute the "date" command in parentheses without using the apostrophe character. |
| echo This is the command | The "echo This is the command 'date'" command displays a text string containing the command "date" in apostrophes. The "echo" command is used to display a text message in the terminal. |
| echo This is the command \`date\` | The command "echo This is the command `date`" also displays a text string containing the command "date" in apostrophes. |
| echo This is the command "`date`" | The command "echo This is the command "date"" also prints a text string containing the command "date" in apostrophes. But in this case, double quotes are used to insert the command. |
| echo D\* | The "echo D\*" command displays all file names beginning with the letter "D" in the current directory (i.e., the directory in which the command is executed). |
| echo "D\*" | The "echo" command in a Linux terminal or similar system displays the arguments passed to it on the screen. In this case, when you use the "echo" command with the argument "D\*", it will display all file names in the current directory that begin with the letter "D". |
| man date | When you run the "man date" command in the terminal, the man page for the "date" command opens, providing information on how to use the command and its various parameters and arguments. |
| man -k password | The "man -k password" command searches the manual database (man pages) for all descriptions containing the word "password". This command helps the user find all available tools that are related to the topic "password". |
| apropos password | This command helps the user to find all available guides that are related to the topic "password". After executing the command, a list of all found manuals containing the word "password" is displayed. |
| man -f passwd | The "man -f passwd" command is intended to search for brief information about the specified file or command in the manual database (man pages). |
| man 5 passwd | To display the help page for a different chapter, specify the chapter number as the first argument to the man command. |
| whatis passwd | Instead of using man -f to display all sections of the help page for a name, you can also use the whatis command. |
| info date | The "info date" command opens a help file (info page) about the "date" command in Linux. |
| date --help | The "date --help" command runs the "date" command with the "--help" option, which allows you to display a brief help on the "date" command with a list of available options and their purposes. |
| ls /usr/share/doc | The "ls /usr/share/doc" command displays a list of files and directories that are located in the "/usr/share/doc" directory on the Linux system. |
| locate crontab | The "locate crontab" command searches for files named "crontab" in all available directories on the system using the file name database stored on the system. |
| locate -b "\crontab" | The "locate -b '\crontab'" command performs the same search for files named "crontab" in all available directories on the system as the "locate crontab" command. However, it adds the "-b" option, which allows you to perform a precise search for files named "crontab", including only full paths to files that contain the word "crontab" as a single word. |
| whereis passwd | The "whereis passwd" command is used to search for the binary, help files, and source files for the "passwd" program in standard locations on the Linux system. |

***The material was prepared by student Kryvenko Andrew (AndrewKryvenko)***

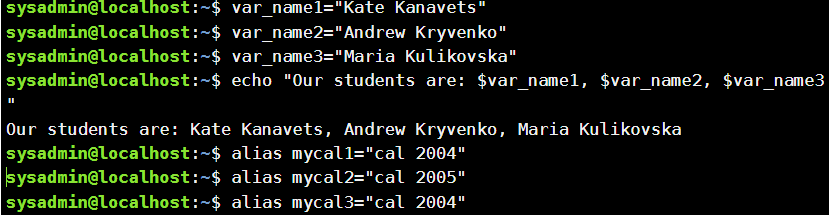
2. Work in the terminal (consolidation of practical skills) be sure to submit your screenshots:

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

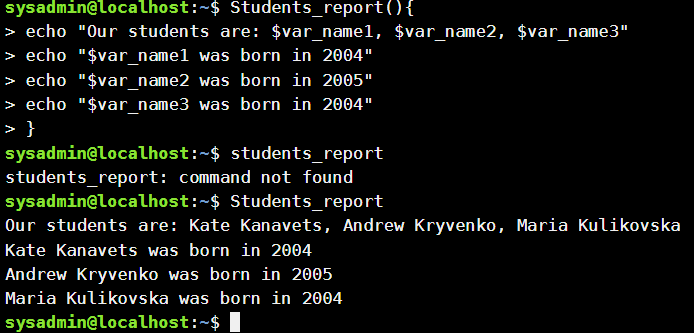
- Use the echo command to print the names of the students in your team

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



2.2. Working with functions in the terminal:

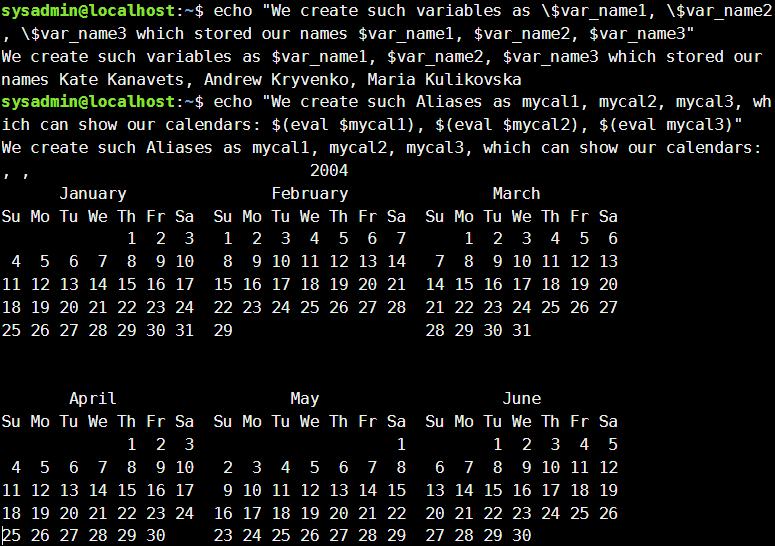
- Create a function called students\_report that will display the names of the students on your team, and then their years of birth



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 display the names of the variables, and then their contents)

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



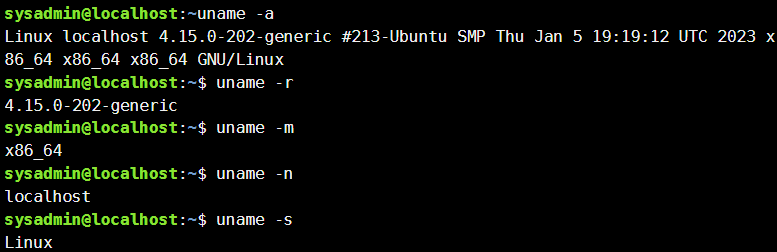
2.4. Working with Control Statements in the terminal:.

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

We could accomplish tasks 2.1 and 2.2 using terminal statements, but this would be more complicated and less straightforward than using functions. For example, we could use loops and conditional statements to display the variables and calendars in the appropriate order, but this would require more code.

2.5. Working with help commands (Man Pages) in the terminal:.

- Using the uname command as an example, demonstrate how to get help. On the basis of the received additional information, give 5 different options for displaying the result of the information on this command using 5 different options (Options)



***The material was prepared by student Kulikovska Maria (@Smith5004)***

4. Define the following terms:

- Command prompt

The command line is a part of a computer operating system designed to read lines of text entered by the user, thus implementing a command line interface. The command line is also known as a shell.

- Shell

The shell is the command line interpreter that translates commands entered by a user into actions to be performed by the operating system.

- Command

A command is a software program that when executed on the CLI, performs an action on the computer. To execute a command, the first step is to type the name of the command.

5.Answer the following questions:

- What basic information does the prompt prompt line provide?

The request contains information about the user and the system. Typical structure of the request: sysadmin@localhost:~$.

This typical request contains the following information:

* User name (sysadmin)
* System name (localhost)
* Current directory (~)

- Why do commands need parameters and arguments?

The argument is needed to specify how the command should act, and is used with commands to extend or change the conduct of the command.

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

* Using the -l option of the ls command leads to a long list with additional

information about the listed files, such as permissions, file size, and other information.

* Using the -r option causes results to be printed in reverse alphabetical order.
* Using the -h option causes file sizes to be displayed in a human-readable format.

- How can you use team history and what are the benefits?

All commands that you enter while working are saved and you can find and view them at any time by using the history command.

You can also use this command to go back a few commands so that you don't have to type a recently executed command all over again.

- What is the purpose of the echo command?

The echo command is usually used in shell scripts to display a message or to output the results of other commands to the terminal.

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

A variable is a function that allows the user or shell to store data.

Variables are named and stored temporarily in memory.

Two types of variables are used in the Bash shell: local variables and environment variables.

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

The env command allows you to modify the list of environment variables before executing a user command. On the other hand, env allows to change the environment in which programs are executed by passing a set of variable definitions in the following command

The export command is one of the built-in bash shell commands, so it is an integral part of your command line. It marks an environment variable for export with any new child process and this allows the child process to inherit all the marked variables.

The unset command is used to remove the variable from your shell environment.

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

Commands for getting help: help, info, man, apropos, whatis.

Glossary of terms

In-line editing (Внутрішнє редагування)

In-line editing allows you to quickly make changes without going to a new page.

Aliases (псевдонім)

A shortcut that refers to a command. Used to replace long commands, increase efficiency, and prevent possible spelling errors.

Variables (змінна)

A feature that allows the user or the shell to save data.

Current Directory (поточний каталог)

The directory where all commands are executed.

Крапка з комою (крапка з комою)

Separates commands on a single line. All arguments before the semicolon are treated as separate comm

***The material was prepared by student Kanavets Kateryna (kanavetsk)***

Control questions

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

There are several types of commands in the Bash shell:

1) Inline commands: these are commands that are implemented directly in the Bash shell. They are faster than external commands because they do not require starting a new process. Some examples of built-in commands are: cd, echo, pwd, export, alias.

2) External commands: these are commands that are executed in a separate process created by the shell. They are usually placed in directories contained in the PATH variable, which contains a list of paths to executable files on your system. Some examples of external commands are: ls, grep, awk, sed, sort.

3) Functions: these are blocks of code that can be called from the command line or from another function in the shell. Functions are useful for organizing repetitive code and ensuring its reuse.

4) Alias: This is shorthand for a command or sequence of commands that can be invoked from the command line. Aliases are useful for shortening long or complex commands and for preserving personalized commands.

5) Special characters: These are characters used to control shell behavior and perform various operations on the command line. Some examples of special characters are: >, <, |, &, ;.

2. What are environmental variables? What are they? How can they be viewed in the terminal?

Environment variables are variables that contain information about the environment in which the process is running in the operating system. Each process has its own list of environment variables that can be customized to provide different functionality and configuration options.

There are several types of environment variables:

System environment variables: These are variables that are set by the system and reflect system parameters. For example, PATH stores a list of paths to executable files in the operating system.

User environment variables: These are variables that can be set by the user or used in various scripts or programs. For example, MY\_VAR can be used to store certain values in a program.

To view the environment variables in the terminal, you can use the following command:

printenv

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

$PS1 is an environment variable that is responsible for displaying the initial command line prompt in the terminal. It contains special characters that can be used to display information about the current working directory, user, host, current date, and other information.

The value of the $PS1 variable contains the character formatting that determines the appearance and content of the initial command line prompt. Typically, the value of this variable consists of a sequence of special characters, such as \u, \h, \w, and others, which are interpreted by the terminal and replaced with the required information when the initial command prompt is displayed.

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 in Bash by setting a new value for this variable. As for the question of what happens to the invitation line after changing the $PS1 variable, it should be noted that it will reflect the new value of the $PS1 variable if it has been changed to something else.

To change the default value of the $PS1 variable, you need to make changes to the Bash shell configuration file. Usually, this is the ~/.bashrc file. To change the value of the $PS1 variable in this file, you need to add a line containing the command to change the value of the $PS1 variable

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

Quotation marks (single or double) in the Bash shell are used to keep lines of text as a single element. In particular, the use of quotes allows you to:

Set the value of variables if they contain spaces or special characters such as "\*", "?", "[", "{", "}", "$", etc.

Specify arguments for commands if they contain spaces or special characters.

Use special characters that have special meanings in Bash as regular characters.

6. What are control statements used for, what types of them do you know?

Control statements are used to control the behavior of the shell and execute commands depending on certain conditions. The main types of control statements in the Bash shell are as follows:

Conditional statements:

if-then-else: used to check a condition, if the condition is true, the first block of commands is executed, if the condition is false, the second block of commands is executed.

case: used to replace a sequence of if-else statements. It checks the value of a variable and executes the corresponding branch of commands.

Cyclic instructions:

for: used to execute a specific set of commands for each item in the list.

while: used to execute commands as long as the condition remains true.

until: used to execute commands until the condition becomes true.

Other instructions:

break: used to exit the loop.

continue: used to move to the next iteration of the loop.

exit: used to terminate the program.

function: used to define functions.

source: used to read and execute commands from a file.

Each type of control instruction has its own syntax and features of use.

7. What is the difference if there is a $ or # at the end of the bash prompt line? For example, on the screen we see the following entries

The $ or # character at the end of the bash prompt line (PS1) indicates which user is running the command shell. Typically, the $ symbol indicates that the shell is running as a regular user, and the # symbol indicates that it is running as a superuser (root).

So, if you see the $ symbol at the end of the prompt, it means that the shell is running as a regular user. If the # character is at the end of the prompt line, it means that the shell is running as the superuser (root).

Changing the character at the end of the prompt line from # to $ can indicate a change in the rights of the user with whom the shell is running. For example, if you change the user from a regular user to a superuser (root), the # character can change to the $ character.

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

The whereis and locate commands are used in Unix-like operating systems to search for files in the system.

Whereis searches for executable files, object files, and help files for a given command specified as an argument. It returns a list of paths to these files.

locate searches for a file in a database that contains the names of all files on the system. It returns a list of files matching the specified pattern.

The main difference between whereis and locate is that whereis searches for a file by its type, while locate searches for a file by its name. Also, locate is faster because it searches the database, while whereis searches the entire file system. However, updating the database for locate can take some time, while whereis will always return the most current result.

Conclusion

Therefore, in this lab, we learned how to use basic commands in CLI mode.