“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”**

Виконав(ла/ли) студент(ка/и)

групи: КСМ-23а

The Awkward Turtles:

Когут Б.М., Михайленко О.О.,

Трощинський Я.П.

Перевірила викладач

Сушанова В.С.

Київ 2024

**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 та віртуальна машина Virtual Box (Oracle).

3. ОС GNU/Linux (будь-який дистрибутив).

4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

***Виконав Михайленко Олексій***

1.  **Command interpreter** – a program that receives text commands from the user, interprets them, and performs the corresponding actions in the operating system. An example of a command interpreter is the command line interface (CLI) or terminal. It translates user commands into system calls to manage the computer’s resources.

 **Shell** – an interface between the user and the operating system. It allows the user to input commands that the operating system will execute. The shell can be textual (command line) or graphical (graphical user interface). Software shells may also include additional features like script handling and automation.

 **Command** – an individual instruction or request that a user or program sends to the operating system via a command interpreter or shell to perform a specific task. Commands can carry out various operations such as launching programs, working with files, managing processes, etc.

2.  **What basic information does the prompt line provide?**  
The prompt line in a command shell provides basic information about the system's current state. Typically, this includes the username, hostname, the current directory, and a symbol indicating readiness to accept a command (e.g., $ for a regular user or # for the superuser). For example:

user@hostname:~$

 **Why do commands need parameters and arguments?**  
Parameters and arguments modify a command’s behavior and define what the command should operate on. Parameters (also called "options" or "flags") specify how the command should perform the task, while arguments indicate the objects (such as files or directories) on which the command will operate. For example, in the command ls -l /home:

* -l is a parameter that changes the output format to detailed;
* /home is an argument specifying the directory to list.

 **What is the purpose of the ls command, and what parameters and arguments can it have? Give 3 examples.**  
The ls command is used to display a list of files and directories in the current or specified directory. Some parameters and examples:

* ls -l: detailed list of files with permissions, owner, size, and last modification time;
* ls -a: displays all files, including hidden ones (files that start with a dot);
* ls -lh: detailed file list with sizes shown in human-readable units (KB, MB, etc.).

 **How can the command history be used, and what advantages does it offer?**  
Command history allows users to view and reuse previously entered commands. This improves efficiency by eliminating the need to retype long or complex commands. The history command lists previous commands, or the arrow keys can be used to navigate through them. A specific command can be executed again by its number using !<number>. For example, !45 will execute command number 45 from the history.

 **What is the purpose of the echo command?**  
The echo command is used to display text or the values of variables in the terminal. It’s useful for outputting information or checking the values of environment variables. For example:

echo "Hello, World!"

or to display the value of a variable:

echo $HOME

 **Define the concept of a variable in the Bash shell. What types of variables does it support?**  
A variable in Bash is an object that stores data for use in scripts or commands. Bash supports two types of variables:

* **Local variables** – available only within the current session or script;
* **Environment variables** – available to all processes spawned from the current environment.

Variables can store various data types: strings, numbers, or the results of commands.

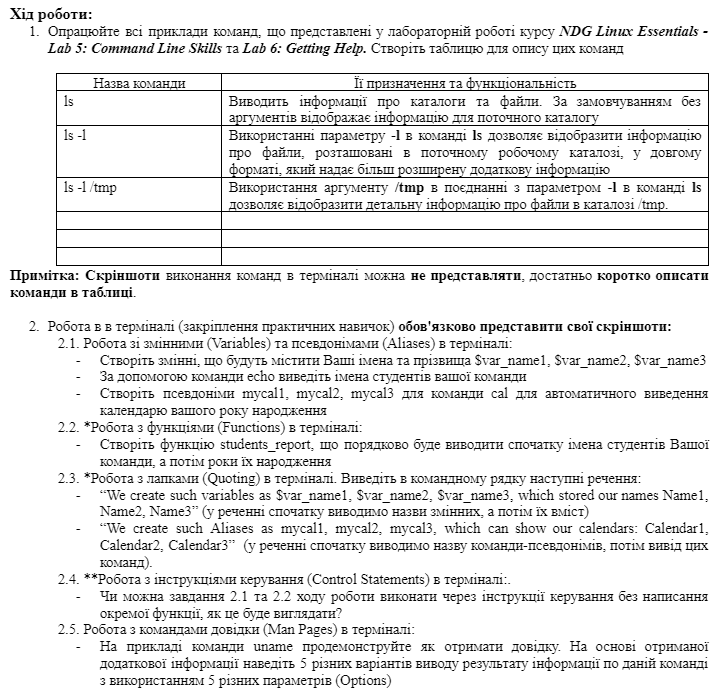
 **What is the purpose of the env, export, and unset commands?**

* env: displays a list of all environment variables or runs a program with specified environment variables;
* export: declares a variable as an environment variable, making it available to child processes;
* unset: deletes a variable, freeing up memory and making it inaccessible.

 **What commands for obtaining help on terminal commands do you know?**  
To get help on terminal commands, you can use:

* man <command> – displays a detailed manual for the command;
* --help – many commands support this option for a quick help summary (e.g., ls --help);
* info <command> – another way to get structured help documentation.

***Виконав Трощинський Ярослав***



**Work progress:**

1. Work through all the command line examples presented in the NDG Linux Essentials course lab Lab 5: Command Line Skills and Lab 6: Getting Help. Create a table to describe these commands
2. Working in the terminal (consolidation of practical skills), you must present your screenshots:

2.1. Working with variables (Variables) and aliases (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 display the names of the students on your team

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

2.2. \*Working with functions (Functions) in the terminal:

Create the students\_report function, which will first display the names of the students in your team, and then their birth years in order

2.3. \*Working 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, we 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, we first output the name of the alias command, then the output of these commands).

2.4. \*\*Working with control instructions (Control Statements) in the terminal:.

Is it possible to do tasks 2.1 and 2.2 of the workflow through control instructions without writing a separate function, what would it look like?

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

Use the uname command as an example to demonstrate how to get help. Based on the received additional information, specify 5 different options for outputting the result of information on this command using 5 different parameters (Options)

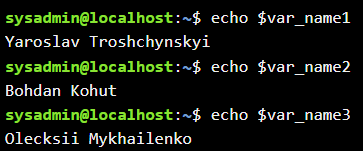
| Name of command | Its function |
| --- | --- |
| ls | Lists directories and files in selected directory |
| ls -l | Gives the same information as “ls” in long format |
| whoami | Displays the user name of the current user |
| uname | Displays information about the current system |
| history | Displays last entered commands |
| ! -*number* | Executes command №*number* from history |
| *variable*=*value* | Initialize and give a value to a variable |
| echo *$variable* | Prints text or value of variable |
| type *command* | Used to determine information about command type |
| which *command* | Determines if there is an executable file named *command* |
| alias *name=command* | Maps longer commands to shorter ones |
| date | Displays today's date |
| man *command* | Gives access to the manual page of *command* |
| apropos *keyword* | A way of viewing man page summaries with a keyword |
| whatis *command* | Displays all man page sections for a name |
| locate *file* | A way of searching for a file |
| whereis *command* | A way to find where a command located |

2. 1. I created 3 variables: var\_name1=”Yaroslav Troshchynskyi”, var\_name2=”Bohdan Kohut”, var\_name3=”Olecksii Mykhailenko”. Next step is to do a 3 aliases:

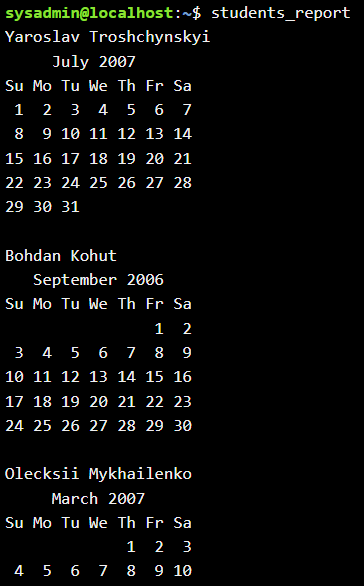
alias mycal1=”cal 07 2007”

alias mycal2=”cal 09 2006”

alias mycal3=”cal 03 2007”



2. 2. After creating variables and aliases I wrote a function



function students\_report

{

echo $var\_name1

mycal1

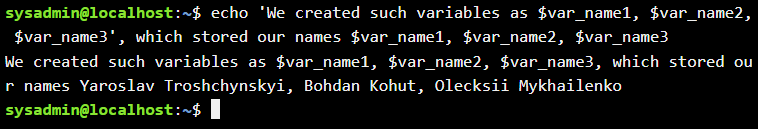
echo $var\_name2

mycal2

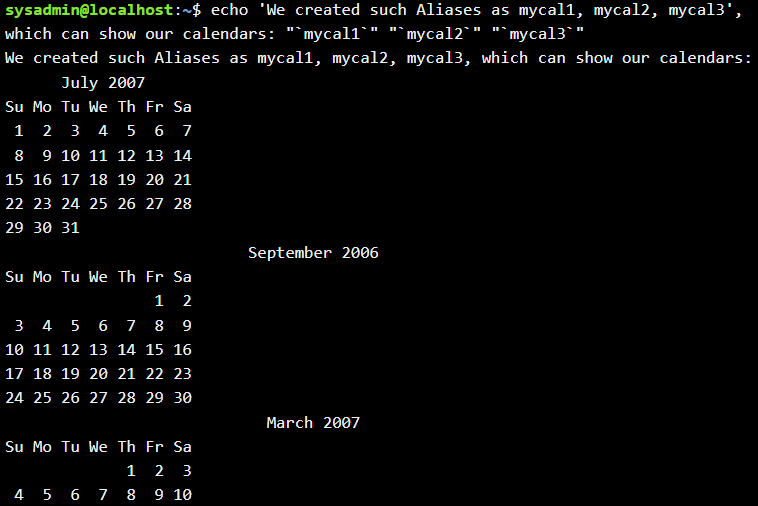
echo $var\_name3

mycal3

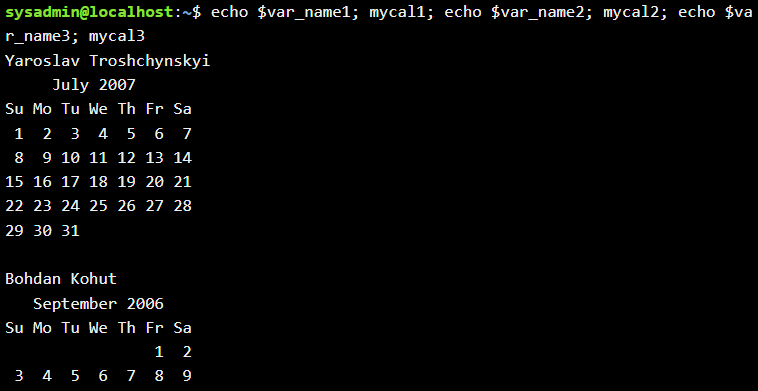
}



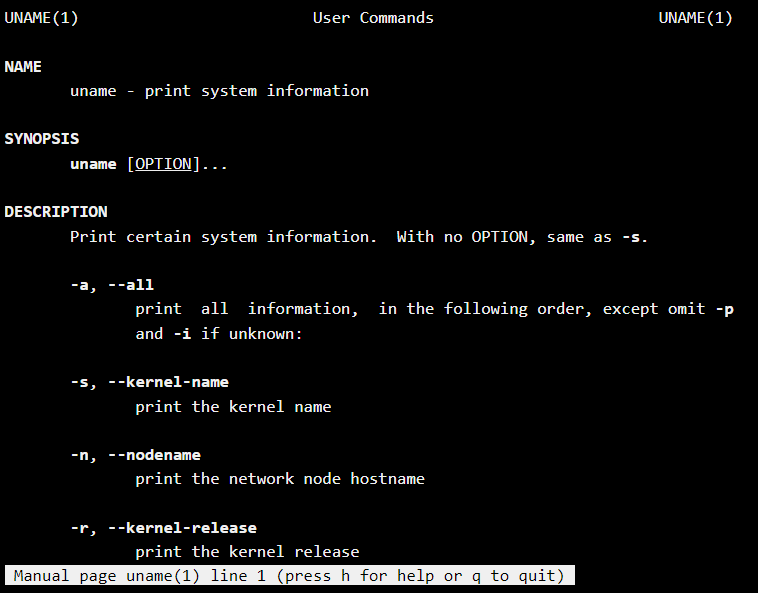
2. 3.



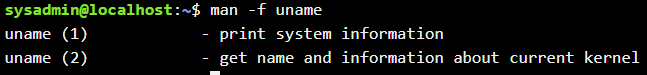
2. 4. Yes, for this you would need to write all the commands you had written in function in 1 line using semicolon(;) between them. Result is the same:



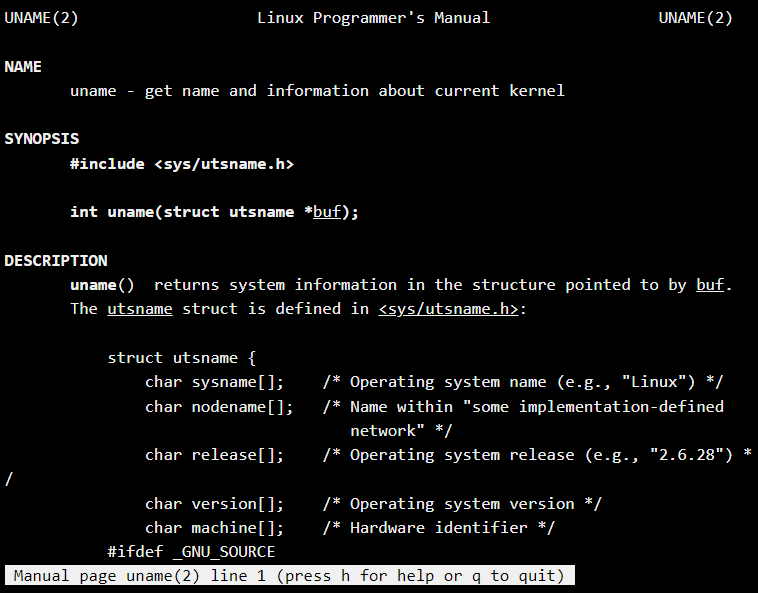
2. 5. Using command “man uname” I got full manual about this command:



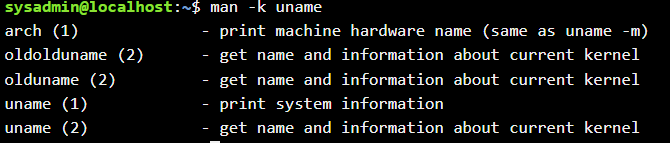
Printing “man -f uname” I got a list of sentences in which we have a word “uname”



Command “man 2 uname” opens a second page of uname`s manual



Command “man -k uname” searches for a keyword “uname” from all of man manuals



Command “man -a uname” searches for all of pages of uname manual, and opens first of them, after you close it you can tap Enter to enter to the next one, Ctrl+D to skip this page or Ctrl+C to quit reading uname manuals

