МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп’ютерних наук та інформаційних систем

**Лабораторна робота № 3**

з курсу «Програмування та підтримка веб-застосувань»

на тему:

«Написання Bash-скриптів для управління Linux-сервером»

**Виконав:**

студент групи КН-42

Постоловський Я. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Перевірив:**

ст. викладач кафедри КНІС

к.т.н. Ізмайлов А. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис)

м. Івано-Франківськ

2024

**Лабораторна робота №3**

**Написання Bash-скриптів для управління Linux-сервером**

***Мета роботи*:** Навчитись створювати Bash-скрипти для управління Linux-базованим сервером.

* 1. **Теоретичні відомості**

У Linux існує кілька командних оболонок (shell), основними з яких є: Bourne Shell (sh), C Shell (csh), Korn Shell (ksh), TC Shell (tcsh) та Bourne Again Shell (bash). Найбільш популярною є оболонка bash, що поєднує корисні функції Korn та C shell, забезпечуючи сумісність із sh. Bash підтримує стандарт IEEE POSIX та додає зручності як для програмування, так і для інтерактивного використання.

**Структура bash-скриптів**

На початку скрипту у bash прийнято вказувати:

#!/bin/bash

Цей запис дає системі зрозуміти, що текст файлу є скриптом. Синтаксис bash підтримує літери латинської абетки, цифри, знаки пунктуації та ключові слова.

**Робота зі змінними та умовами**

Змінні в bash автоматично створюються при присвоєнні значення. Усі значення зберігаються у вигляді рядків, але за потреби можуть бути використані як числа. Bash не підтримує типізацію змінних, тому значення може бути як числовим, так і символьним.

Основна форма умовного оператора if виглядає так:

if [ expression ]; then

statements

elif [ expression ]; then

statements

else

statements

fi

Секції elif та else є опціональними. Пробіли після [ та перед ] є обов’язковими.

**Цикли в bash**

Bash підтримує кілька типів циклів:

* while-do: виконує команду, поки умова є істинною:

while [ expression ]; do

command-list

done

* for ... in: перебирає значення у списку:

for item in list; do

command-list

done

* until-do: виконує ітерації до настання певної умови.

Bash забезпечує основні можливості для автоматизації завдань та керування даними, що дозволяє ефективно використовувати оболонку у повсякденній роботі.

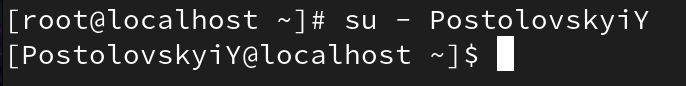
**1.2 Хід роботи**

**1.2.1**

Усі подальші завдання слід виконувати на віртуальній машині та з-під

користувача, які були створені у попередній лабораторній роботі. У

якості інтерфейсу можна використовувати довільний ssh-клієнт.



**1.2.2**

Додайте до стартового сценарію bash привітання користувача при вході

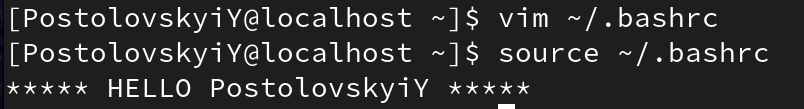
до режиму командного рядка у вигляді:

\*\*\*\*\* HELLO username \*\*\*\*\*

де username – ім’я користувача.

Включіть до звіту знімок екрану, який міститиме появу цього

привітання при вході користувача до системи.



**1.2.3.**

Змініть послідовність символів, які показуються перед запитом на ввід

у командному рядку так, щоб перед запитом відображались (у заданому

порядку):

• дата

• година

• ім’я користувача

• поточна директорія

Усі наведені компоненти мають розділятись принаймні одним

пропуском. Включіть до звіту знімок екрану, який міститиме

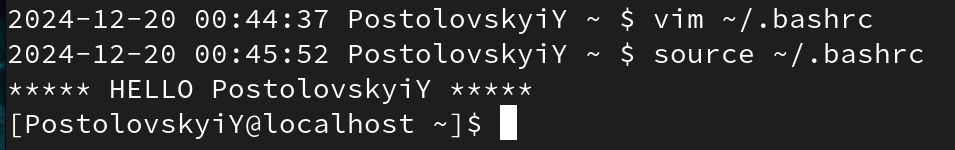
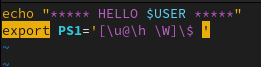
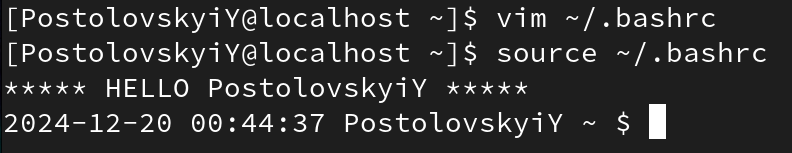
результати виконаного завдання.

Після виконання завдання, поверніть значення послідовності символів

за замовчуванням. Не зберегли чи не запам’ятали, як було за

замовчуванням? Наступного разу перед тим, як починати виконувати

завдання – дочитайте його до кінця ☺



**1.2.4**

Напишіть bash-скрипт, який

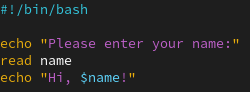
1. просить користувача увести своє ім’я;

2. виводить користувачеві іменне привітання у консоль, наприклад

«Hi, Bill!».

Включіть до звіту лістинг створеного скрипту та знімок (-ки) екрану,

які демонструють результати його виконання.



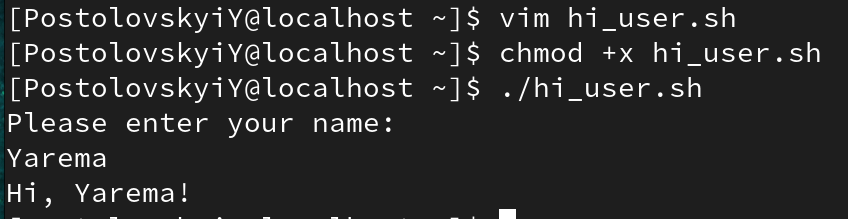
**Код:**

#!/bin/bash

echo “Please enter your name:”

read name

echo “Hi, $name!”



**1.2.5**

Напишіть bash-скрипт, який очікує на позиційний параметр у вигляді

рядка з адресою текстового файлу і:

• у випадку відсутності позиційних параметрів виводить у консоль

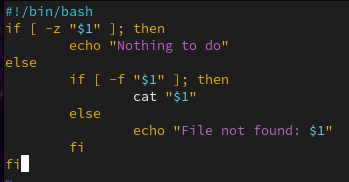
повідомлення «Nothing to do»;

• у випадку наявності одного позиційного параметра виводить у

консоль вміст відповідного текстового файлу.

Включіть до звіту лістинг створеного скрипту та знімок (-ки) екрану,

які демонструють результати його виконання.



**Код:**

#!/bin/bash

if [ -z “$1” ]; then

echo “Nothing to do”

else

if [ -f “$1” ]; then

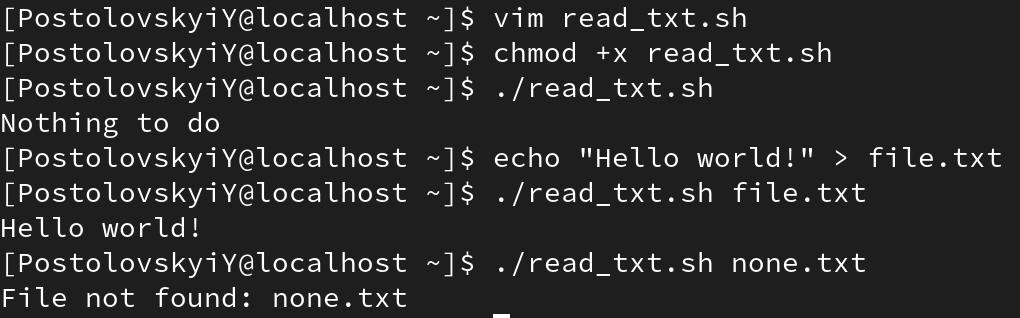
cat “$1”

else

echo “File not found: $1”

fi

fi



**1.2.6**

Напишіть bash-скрипт, який очікує на числові позиційні параметри і:

• у випадку відсутності позиційних параметрів виводить у консоль

повідомлення «Nothing to do»;

• у випадку наявності одного позиційного параметра виводить у

консоль його значення та повідомлення «Only one parameter

detected»;

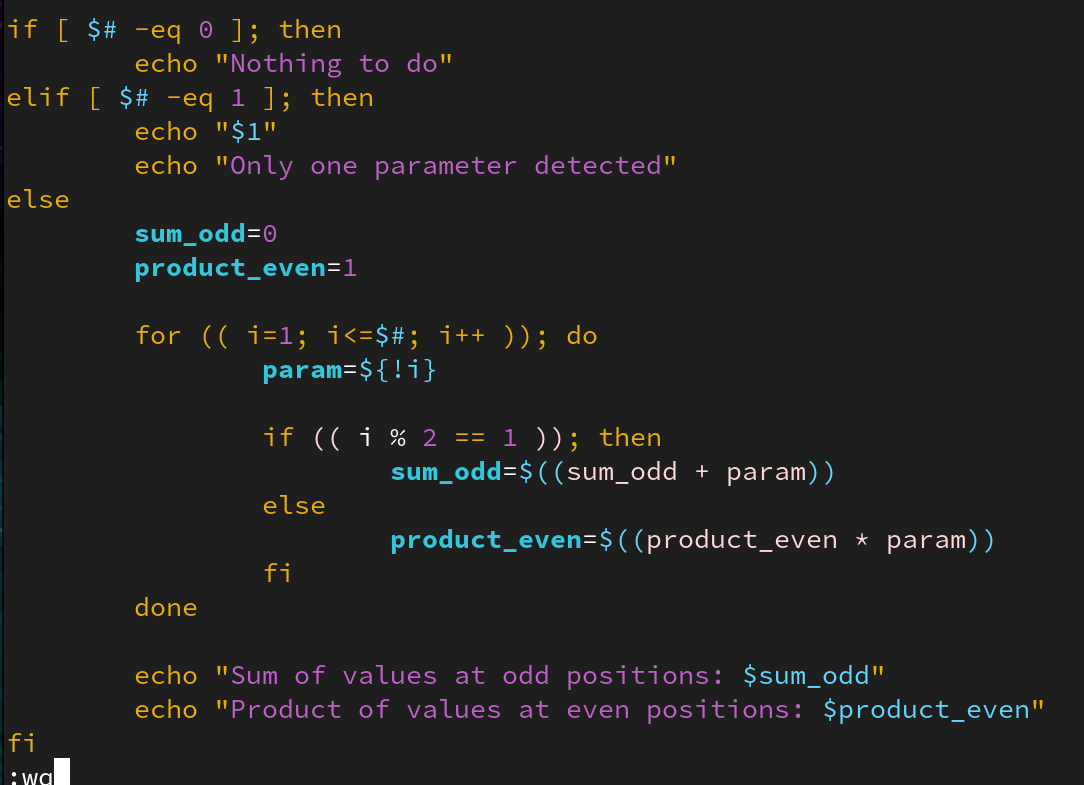
• у випадку наявності двох або більше параметрів виводить у

консоль суму значень параметрів з непарними номерами та

добуток значень параметрів з парними номерами.

Включіть до звіту лістинг створеного скрипту та знімок (-ки) екрану,

які демонструють результати його виконання.



#!/bin/bash

if [ $# -eq 0 ]; then

echo "Nothing to do"

elif [ $# -eq 1 ]; then

echo "$1"

echo "Only one parameter detected"

else

sum\_odd=0

product\_even=1

for (( i=1; i<=$#; i++ )); do

param=${!i}

if (( i % 2 == 1)); then

sum\_odd=$((sum\_odd + param))

else

product\_even=$((product\_even \* param))

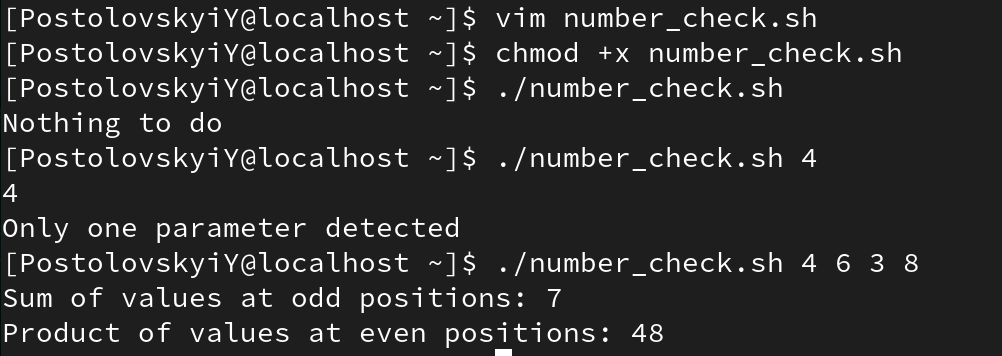
fi

done

echo "Sum of values at odd positions: $sum\_odd"

echo "Product of values at even positions: $product\_even"

fi



**1.2.7**

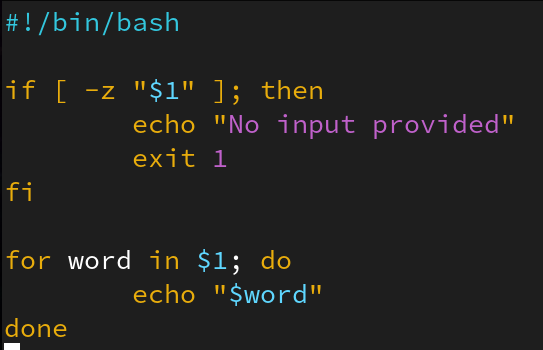
Напишіть bash-скрипт, який приймає параметри у вигляді рядка, який

складається зі слів, які розділені пробілом, і виводить їх у консоль по

одному в рядку (у стовпчик).

Включіть до звіту лістинг створеного скрипту та знімок (-ки) екрану,

які демонструють результати його виконання.



#!/bin/bash

if [ -z "$1" ]; then

echo "No input provided"

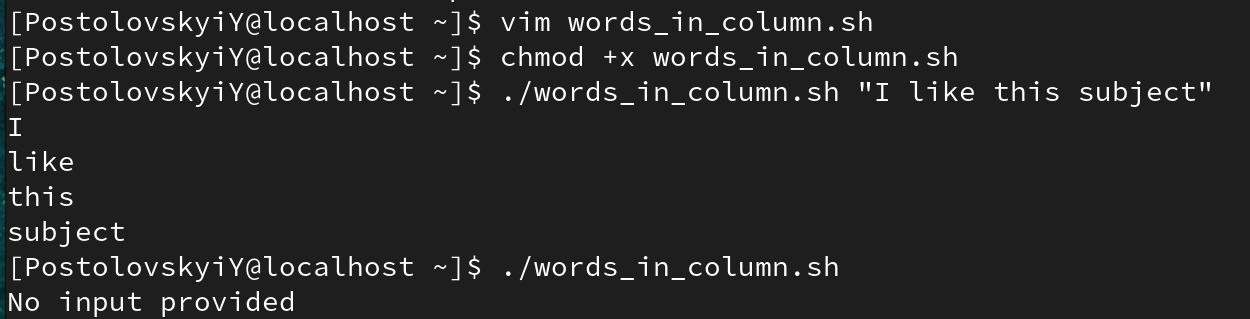
exit 1

fi

for word in $1; do

echo "$word"

done



**1.2.8**

Напишіть bash-скрипт, який очікує у якості першого параметра одне зі

слів «dir» або «file», а у якості другого параметра – рядок із шляхом до

певної папки. Визначте поведінку скрипту у наступному вигляді:

• у випадку відсутності позиційних параметрів скрипт виводить у

консоль повідомлення «Nothing to do»;

• у випадку, якщо за переданою адресою папки не існує, вивести у

консоль повідомлення: «No such directory» і завершити роботу;

• у випадку наявності одного позиційного параметра, який

дорівнює «dir» скрипт виводить у консоль список директорій, які

містяться у поточній папці;

• у випадку наявності одного позиційного параметра, який

дорівнює «file» скрипт виводить у консоль список файлів (але не

директорій), які містяться у поточній папці;

• у випадку наявності одного позиційного параметра, який не

дорівнює «dir» або «file» скрипт виводить у консоль список

файлів і директорій, які містяться у папці за адресою, яка

передана у якості параметра, якщо така існує;

• у випадку наявності двох або більше позиційних параметрів,

перший з яких дорівнює «dir» скрипт виводить у консоль список

директорій, які містяться у папці за адресою, яка передана у

якості другого параметра, якщо така існує;

• у випадку наявності двох або більше позиційних параметрів,

перший з яких дорівнює «file» скрипт виводить у консоль список

файлів (але не директорій, які містяться у папці за адресою, яка

передана у якості другого параметра, якщо така існує;

• у випадку наявності двох або більше позиційних параметрів,

перший з яких не дорівнює «dir» або «file» скрипт виводить у

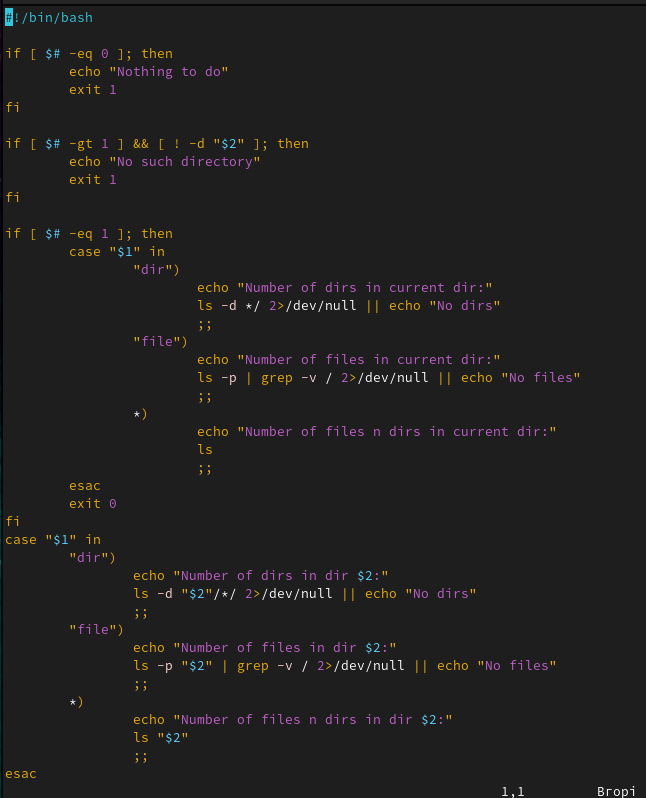
консоль список файлів і директорій, які містяться у папці за

адресою, яка передана у якості другого параметра, якщо така

існує.

Включіть до звіту лістинг створеного скрипту та знімок (-ки) екрану,

які демонструють результати його виконання.



#!/bin/bash

if [ $# -eq 0 ]; then

echo "Nothing to do"

exit 1

fi

if [ $# -gt 1 ] && [ ! -d "$2" ]; then

echo "No such directory"

exit 1

fi

if [ $# -eq 1 ]; then

case "$1" in

"dir")

echo "Number of dirs in current dir:"

ls -d \*/ 2>/dev/null || echo "No dirs"

;;

"file")

echo "Number of files in current dir:"

ls -p | grep -v / 2>/dev/null || echo "No files"

;;

\*)

echo "Number of files n dirs in current dir:"

ls

;;

esac

exit 0

fi

case "$1" in

"dir")

echo "Number of dirs in dir $2:"

ls -d "$2"/\*/ 2>/dev/null || echo "No dirs"

"file")

echo "Number of files in dir $2:"

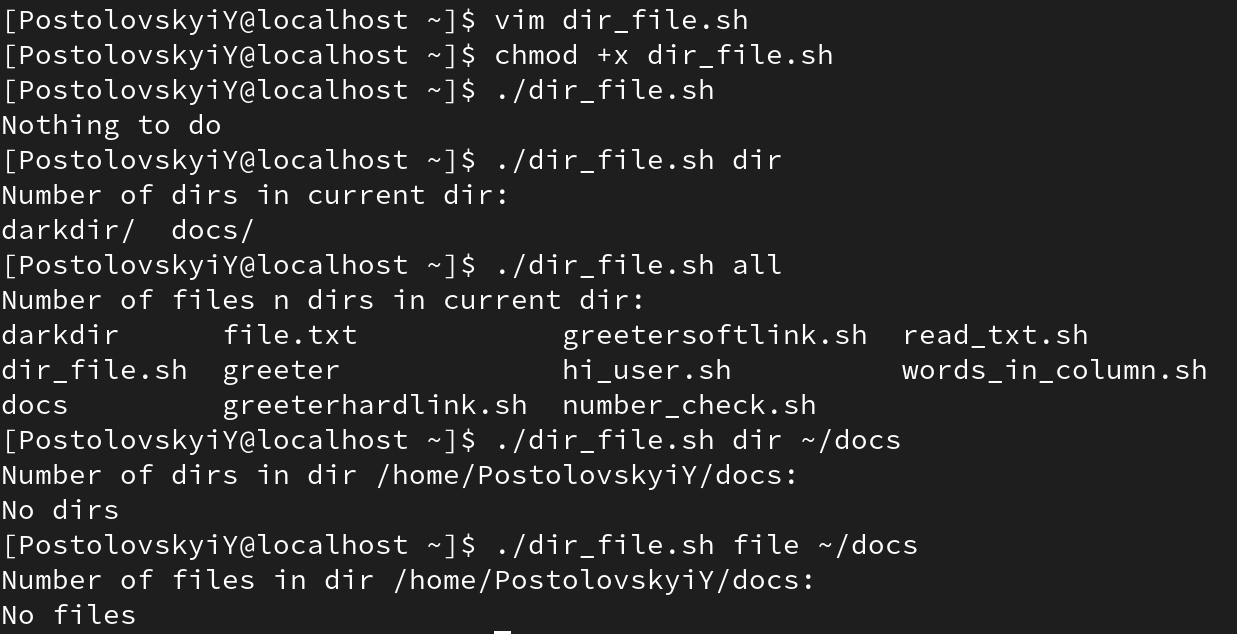
ls -p "$2" | grep -v / 2>/dev/null || echo "No files"

\*)

echo "Number of files n dirs in dir $2:"

ls "$2"

esac



**1.2.9**

Напишіть bash-скрипт, який очікує на числові позиційні параметри і:

• у випадку відсутності позиційних параметрів виводить у консоль

повідомлення «Nothing to do»;

• містить у собі декларацію функції пошуку максимального

елемента у переліку числових значень, які передаються у якості

аргументів;

• у випадку наявності принаймні одного позиційного параметра

знаходить за допомогою відповідної функції максимальний

елемент у переліку чисел, утвореному позиційними параметрами

і виводить його у консоль.

Локальні змінні функції не повинні бути видимі у скрипті ззовні

контексту виконання функції. Максимальне значення, знайдене

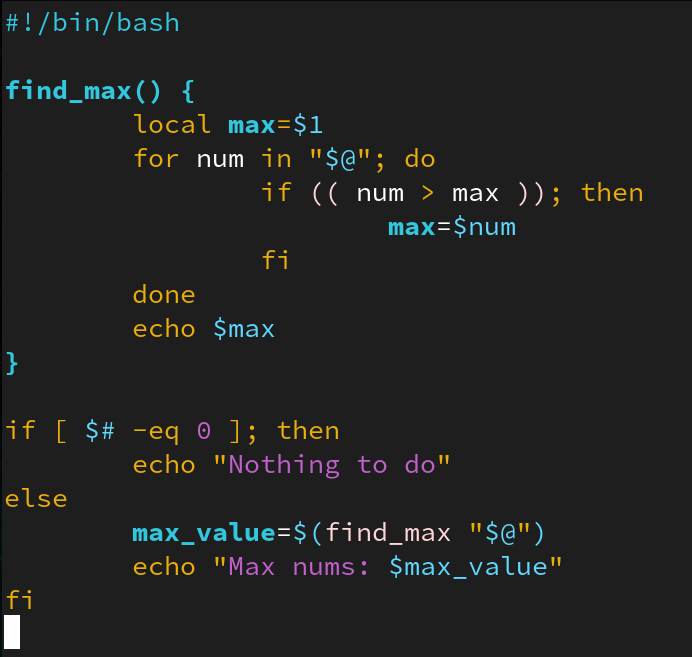
функцією не повинне виводитися у консоль з її контексту виконання, а

повинне бути «повернене» у скрипт і бути виведене у консоль з його

контексту.

Включіть до звіту лістинг створеного скрипту та знімок (-ки) екрану,

які демонструють результати його виконання.



#!/bin/bash

find\_max() {

local max=$1

for num in "$@"; do

if (( num > max )); then

max=$num

fi

done

echo $max

}

if [ $# -eq 0 ]; then

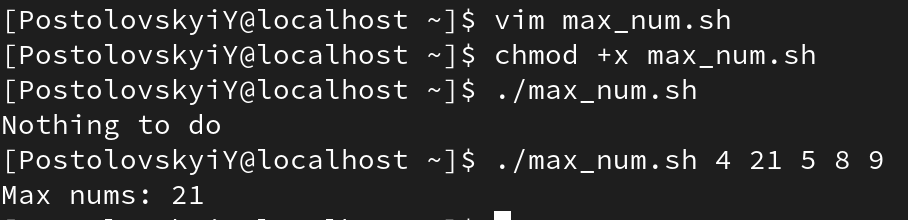
echo "Nothing to do"

else

max\_value=$(find\_max "$@")

echo "Max nums: $max\_value"

fi



**Висновок**

Під час виконання лабораторної роботи я заглибився у вивчення командної оболонки bash, яка є основним інструментом для роботи з Linux-системами завдяки своїй гнучкості та підтримці стандартів POSIX. Я освоїв не лише основи синтаксису bash, а й продовжив вивчати більш складні концепції програмування, по суті продовження другої лабораторної яке дає більш широке сприйняття і знання в Linux.

Реалізовані практичні завдання включали пошук максимальних значень у масивах даних, обробку структур файлової системи та розробку скриптів для зручного виведення результатів. Це дало змогу значно поглибити мої суперські знання в області bash-програмування та автоматизації операцій у Linux, що є важливим інструментом для ефективного адміністрування серверів та систем.