Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №5

з дисципліни «Операційна система UNIX»

«Створення сценаріїв в оболонці Bash»

Виконав:

студент 2-го курсу, НН ІАТЕ

групи ТР-23

Ровний Григорій Олександрович

Перевірила:

проф. Левченко Л.О.

КИЇВ 2023

**Мета роботи:**

* Набуття навичок створення bash-скриптів в ОС Linux.

**Теоретичні відомості**

Bourne-again shell (GNU Bash) - це реалізація Unix shell, написана на C в 1987 році Брайаном Фоксом для GNU Project. Синтаксис мови Bash є надбудовою синтаксису Bourne shell. Більшість скриптів для Bourne shell можуть бути виконані інтерпретатором Bash без змін, за винятком скриптів, які використовують спеціальні змінні або вбудовані команди Bourne shell. Також синтаксис мови Bash включає ідеї, запозичені з Korn shell (ksh) і C shell (csh): редагування командного рядка, історія команд, стек директорій, змінні $ RANDOM і $ PPID, синтаксис POSIX для підстановки команд: $ (...).

Командна оболонка BASH дозволяє створювати скрипти за допомогою групування кількох команд. Скрипт - звичайний текстовий файл, що містить системні або вбудовані команди оболонки. Такий файл може бути запущений на виконання наступним чином: $bash ім’я\_файла. Оболонка послідовно інтерпретує і виконує команди, задані в сценарії. Ці ж команди можуть бути виконані простим послідовним викликом в командному рядку. Для файлів сценаріїв оболонки bash прийнято встановлювати розширення .sh. Є інший варіант запуску скрипту - вказати його ім'я в командній оболонці. Для цього треба в параметрах доступу визначити файл як виконуваний, і в перших рядках цього файлу явно вказати оболонку, для якої призначений цей скрипт, в такий спосіб: #!оболонка. Будь-який сценарій для Bash починається з вказівки в першому рядку комбінації: #!/bin/bash. У загальному випадку «#» означає коментар, що означає ігнорування рядка. Але якщо він є першим символом файлу і за ним слідує символ «!» та шлях до файлу, командна оболонка запускає відповідний файл і передає йому його ім'я в якості аргументу.

**Перетворення сценарію у виконуваний файл.** Файл сценарію можна виконати двома способами. Перший – запуститиоболонку з ім'ям файлу сценарію як параметром: $ /bin/sh first. Інший спосіб - запустити сценарій, ввівши його ім'я і тимсамим присвоївши йому статус інших команд Linux задопомогою команди chmod, змінивши file mode і зробивши йоговиконуваним для всіх користувачів:

$ chmod u+x <ім’я сценарія>

$ chmod u+x first

Аналогічну функціональність реалізує наступна команда:

chmod 744 <ім’я\_файла\_сценарія>. Після цього можна виконувати файл за допомогою команди $ first. Запуск сценарію на виконання з поточного каталогу проводиться за допомогою: ./ <ім’я\_файла\_сценарія>.

**Синтаксис команд для створення сценарію.** Всі змінні у мові – текстові. Їх імена мають починатися з літери і складатися з латинських букв, цифр і знаку підкреслення. Щоб скористатися значенням змінної, треба перед нею поставити символ $. Розрізняють два класи змінних: позиційні та з іменем. Позиційні змінні - це аргументи командних файлів, їх іменами слугують цифри:

**$0** - ім'я виконуваної команди (для скрипту - це шлях, вказаний при його виклику, для функції - ім'я оболонки).

**$1** - перший аргумент, другий - $2 і т.д. - змінні, які відповідають аргументам, заданим при виклику сценарію.

При виконанні команд використовується підстановка змінних. В команду «підставляється» будь що. Для підстановки використовується або символ «$», або вираз, розміщений у зворотні апострофи. Якщо в тексті команди зустрічається символ «$», то наступний за ним текст до пробілу або кінця команди інтерпретується як ім'я змінної, значення якої підставляється в текст команди. Командна оболонка дозволяє виконувати арифметичні операції. Для цього вираз, який необхідно інтерпретувати як арифметичний, потрібно записати у подвійні круглі дужки, і перед ними ставиться знак долара. Для отримання значення змінної використовуються наступний синтаксис: ALFA=$BETA+$GAMMA Користувач може надати значення змінній через командну оболонку за допомогою команди read, якої в якості аргументу передається ім'я необхідної змінної.

**Умовний оператор if | then та else.** При написанні скриптів часто виникає є необхідність у перевірці будь-яких процесів. Оператори if | then перевіряють код завершення переліку команд на «успішне завершення». Якщо це так, то виконується одна або більше команд, які записані після оператора then. Якщо перевірка повертає «Не успішне завершення» - виконується оператор else «інакше» як того вимагає умова. На завершення умови обов'язково закриваємо його «fi»!

**Використання дужок.** Квадратні дужки «[» є спеціальною вбудованою командою test , яка сприймаєсвої аргументи як вираз порівняння або файлову перевірку.Подвійні дужки «[[» є розширеним варіантом від «[», є зарезервованим словом,а не командою. bash виконує його як один елемент з кодом повернення. Всередині«[[....]]» дозволяється виконання операторів &&, ||, які призводять до помилки взвичайних дужках «[....]».

Круглі дужки «((»- є арифметичними виразами, яке так само повертають код. Тим самим такі вирази можна використовувати в операціях порівняння. Список логічних операторів, які використовуються для if | then | else:

«-z» - рядок порожній, «-n» - рядок не порожній,

«=, (==)» рядки рівні, «!=» - рядки нерівні,

«-eq» - дорівнює, «-ne» - не дорівнює, «-lt, (<)» - менше,

«-le, (<=)» - менше або дорівнює, «-gt, (>)» - більше,

«-ge, (> =)» - більше або дорівнює, «!» - заперечення логічного виразу.

**Цикл ПОКИ**. Якщо потрібно повторити виконання послідовності команд, але заздалегідь не відомо, скільки разів слід їх виконати, застосовується цикл while:

while <умова> do

<оператори>

done

До тих пір, поки код завершення останньої команди <списка1> є 0, виконуються команди <списка2>. При заміні службового слова while на until умова виходу з циклу змінюється на протилежне.

**Цикл until.** Цикл виконується, поки умова не стане істинною (true):

until <умова> do

<оператори>

done

**Оператор циклу for.** Оператор циклу for призначений для обробки в циклі низки значень, які можуть являти собою будь-яку множина рядків. Рядки можуть бути перераховані в програмі або являти собою результат виконаної командною оболонкою підстановки імен файлів. Синтаксис оператора циклу:

for <змінна> in <значення>

do

<оператори>

done

**Функції.** Синтаксис оголошення функції:

function <ім'я> ()

{

<список>;

}

**Відладка сценаріїв.** При виявленні помилки при виконанні сценарію командна оболонка виводитьна екран номер рядка, що містить помилку. Якщо помилку відразу не видно,потрібно додати кілька додаткових команд echo для виведення значень змінних,протестувати фрагменти програмного коду, вводячи їх в командній оболонці вінтерактивному режимі. Основний спосіб відстеження помилок, які найбільшскладно виявляються - використання опцій відладки командної оболонки.

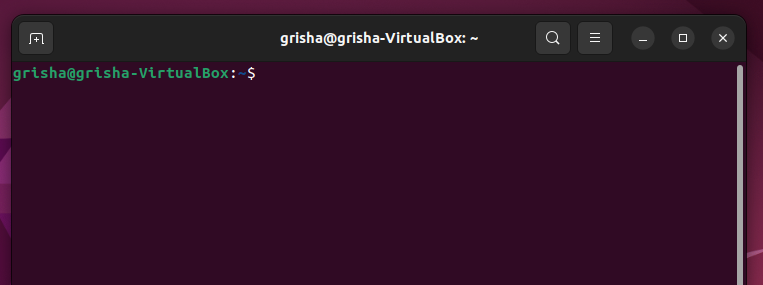
**Поставлене завдання:**

Розробити скрипт, який виконує зазначені дії згідно варіанту по списку журналу групи. Створити файл з назвою власного прізвища, записати до нього ПІБ студента, групу навчання, за бажанням додаткову інформацію щодо ваших уподобань. Створити скрипт, який виводить ПІБ студента, групу навчання, коментарі щодо майбутніх дій, необхідно робити затримку після видачі результатів на екран. Скрипт повинен містити функції, які виконують зазначені дії, а також додатково записати дію відповідно до варіанту.

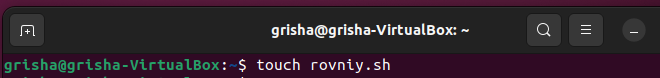
Варіант 19: Вивести на екран дату створення файлу (власного/будь-якого).

**Результат виконання роботи**

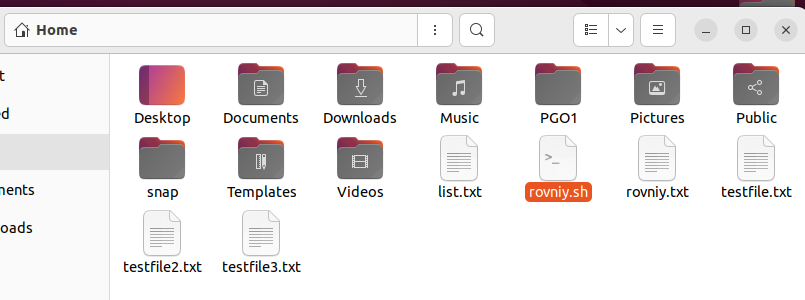
1. Для початку відкриваємо термінал за допомогою комбінації клавіш CTRL+ALT+T.



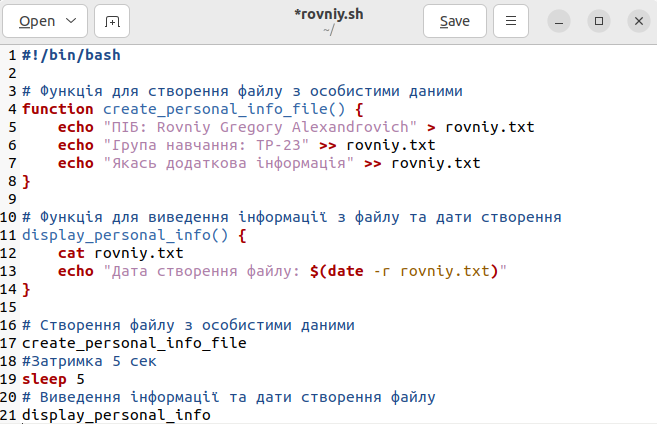
1. У терміналі створюємо файл з розширенням .sh, оскільки для файлів сценаріїв оболонки bash необхідно встановлювати розширення .sh. Назвемо його rovniy.sh, скориставшись командою *touch rovniy.sh.*



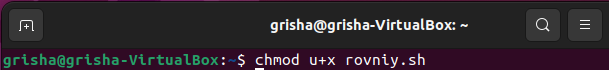
1. У навігаційному меню переходимо до файлового меджера. Відкиваємо наш новостворений файл за допомогою текстового редактора, в якому будемо писати наш скрипт (сценарій).



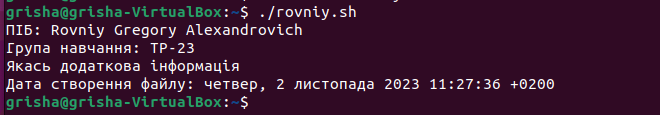
1. В першу чергу на початку файлу прописуємо #!/bin/bash, оскільки саме ці рядки визначають що у файлі міститься сценарій і саме в цьому рядку вказаний шлях до інтерпретатора bash. Наступним кроком створюємо функцію, яка буде створювати файл rovniy.txt та записувати у файл необхідні дані. Для створення функції потрібно просто написати (function) її ім’я, додати круглі дужки «()» і в фігурних дужках «{ }» описувати скрипт функції. Для одночасного запису та створення файлу використовуємо команду *echo «Your Text» > rovniy.txt.* Далі створимо функцію, яка буде виводити інформацію з файлу та, згідно мого варіанту, виводити дату створення файлу. Для цього скористаємось командою *cat rovniy.txt,* яка безпосередньо виведе інформацію з файлу. Далі пишемо *echo* *“Дата створення файлу: $(date –r rovniy.txt)”*, де echo – команда виведення тексту на екран, *$()* конструкція, яка дозоляє скористатися командою, яка знаходиться всередині дужок, date з ключом *–r* дозволяє вивести дату створення/останньої зміни мого файлу *rovniy.sh.* На завершення прописуємо *create\_personal\_info\_file* та *display\_personal\_info* для того щоб їх викликати. Окрім того використовуємо sleep <к-сть секунд> для того щоб була затримка між викликом функцій. Після всіх маніпуляції зберігаємо файл натиснувши на Save.



1. Перетворюємо сценарій за у виконуваний файл. Для цього вводимо у термінал команду *chmod u+x rovniy.sh.* Дана команда надає права на виконання власнику файлу.



1. Перевіримо роботу нашого сценарію. Для цього знову відкриємо термінал та введемо команду *./rovniy.sh,* де ‘./’ вказує на поточну директорію. Зазначення шляху, який починається з «./», дає перевагу в тому, щоб випадково не виконати іншу команду з тим же ім'ям, що і у файлу сценарію. В результаті чого скрипт успішно виводить всю інформацію, при чому також спрацьовує п’ятисекундна затримка для виведення.



**Висновок:**

У результаті виконання лабораторної роботи набуто практичні навички з використання мови програмування bash для написання скриптів для виконання завдання.