“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №7**

з дисципліни: «Операційні системи»

**Тема: «** **Створення скриптових сценаріїв та визначення апаратної конфігурації системи»**

Виконала студентка

групи РПЗ-13б

Дімітрова С.П.

Перевірив викладач

Сушанова В.С.

Київ 2024

**Мета роботи:**

1. Отримання практичних навиків роботи з командною оболонкою Bash.
2. Знайомство знайомство з базовими діями при роботі зі скриптовими сценаріями.

**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

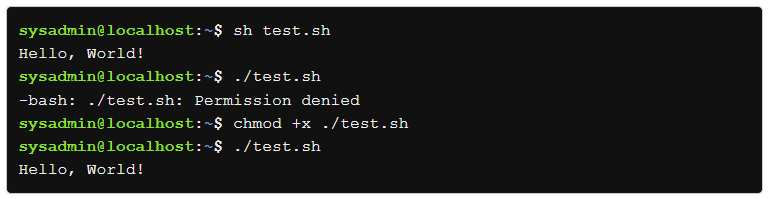
**Короткі теоретичні відомості:**

A script can be as simple as one command:

*echo "Hello, World!"*

The script, *test.sh*, consists of just one line that prints the string Hello, World! to the console.

Running a script can be done either by passing it as an argument to your shell or by running it directly:



In the example above, first, the script is run as an argument to the shell. Next, the script is run directly from the shell. It is rare to have the current directory in the binary search path *$PATH* so the name is prefixed with *./* to indicate that it should be run out of the current directory.

The error Permission denied means that the script has not been marked as executable. A quick *chmod* later and the script works. *chmod* is used to change the permissions of a file, which will be explained in detail in a later chapter.

There are various shells with their own language syntax. Therefore, more complicated scripts will indicate a particular shell by specifying the absolute path to the interpreter as the first line, prefixed by *#!* as shown:

*#!/bin/sh*

*echo "Hello, World!"*

or

*#!/bin/bash*

*echo "Hello, World!"*

The two characters *#!* are traditionally called the hash and the bang respectively, which leads to the shortened form of “shebang” when they’re used at the beginning of a script.

Incidentally, the shebang (or crunchbang) is used for traditional shell scripts and other text-based languages like Perl, Ruby, and Python. Any text file marked as executable will be run under the interpreter specified in the first line as long as the script is run directly. If the script is invoked directly as an argument to an interpreter, such as sh script or bash script, the given shell will be used no matter what’s in the shebang line.

**Editing Shell Scripts**

UNIX has many text editors. The merits of one over the other are often hotly debated. Two are specifically mentioned in the LPI Essentials syllabus: The GNU nano editor is a very simple editor well suited to editing small text files. The Visual Editor, vi, or its newer version, VI improved (*vim*), is a remarkably powerful editor but has a steep learning curve. We’ll focus on nano.

Type *nano* test.sh and you’ll see a screen similar to this:

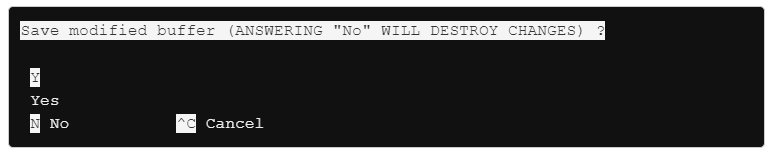


The *nano* editor has few features to get you on your way. You simply type with your keyboard, using the arrow keys to move around and the delete/backspace button to delete text. Along the bottom of the screen you can see some commands available to you, which are context-sensitive and change depending on what you’re doing. If you’re directly on the Linux machine itself, as opposed to connecting over the network, you can also use the mouse to move the cursor and highlight text.

To get familiar with the editor, start typing out a simple shell script while inside *nano*:



Note that the bottom-left option is *^X Exit* which means “press control and X to exit”. Press Ctrl and X together and the bottom will change:

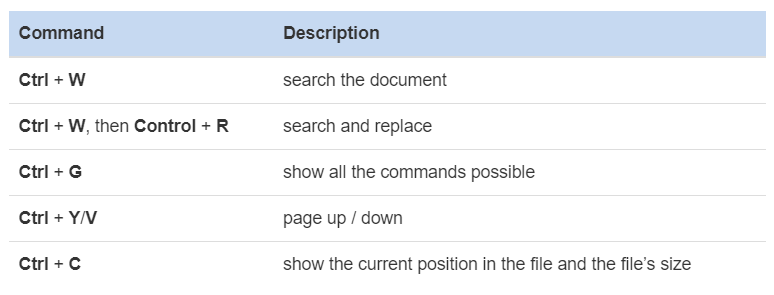


At this point, you can exit the program without saving by pressing the ***N*** key, or save first by pressing ***Y*** to save. The default is to save the file with the current file name. You can press the ***Enter*** key to save and exit.

You will be back at the shell prompt after saving. Return to the editor. This time press ***Ctrl*** and ***O*** together to save your work without exiting the editor. The prompts are largely the same, except that you’re back in the editor.

This time use the arrow keys to move your cursor to the line that has "The time is”. Press ***Ctrl***  and ***K*** twice to cut the last two lines to the copy buffer. Move your cursor to the remaining line and press ***Ctrl***  and ***U*** once to paste the copy buffer to the current position. This makes the script echo the current time before greeting you and saved you needing to re-type the lines.

Other helpful commands you might need are:



**Scripting Basics**

You got your first taste of scripting earlier in this chapter where we introduced a very basic script that ran a single command. The script started with the shebang (or hashbang) line, telling Linux that */bin/bash* (which is Bash) is to be used to execute the script.

Other than running commands, there are 3 topics you must become familiar with:

* Variables, which hold temporary information in the script
* Conditionals, which let you do different things based on tests you write
* Loops, which let you do the same thing over and over.

**Завдання для попередньої підготовки:**

1. \*Прочитайте короткі теоретичні відомості до лабораторної роботи та зробіть невеликий словник базових англійських термінів з питань призначення команд та їх параметрів.

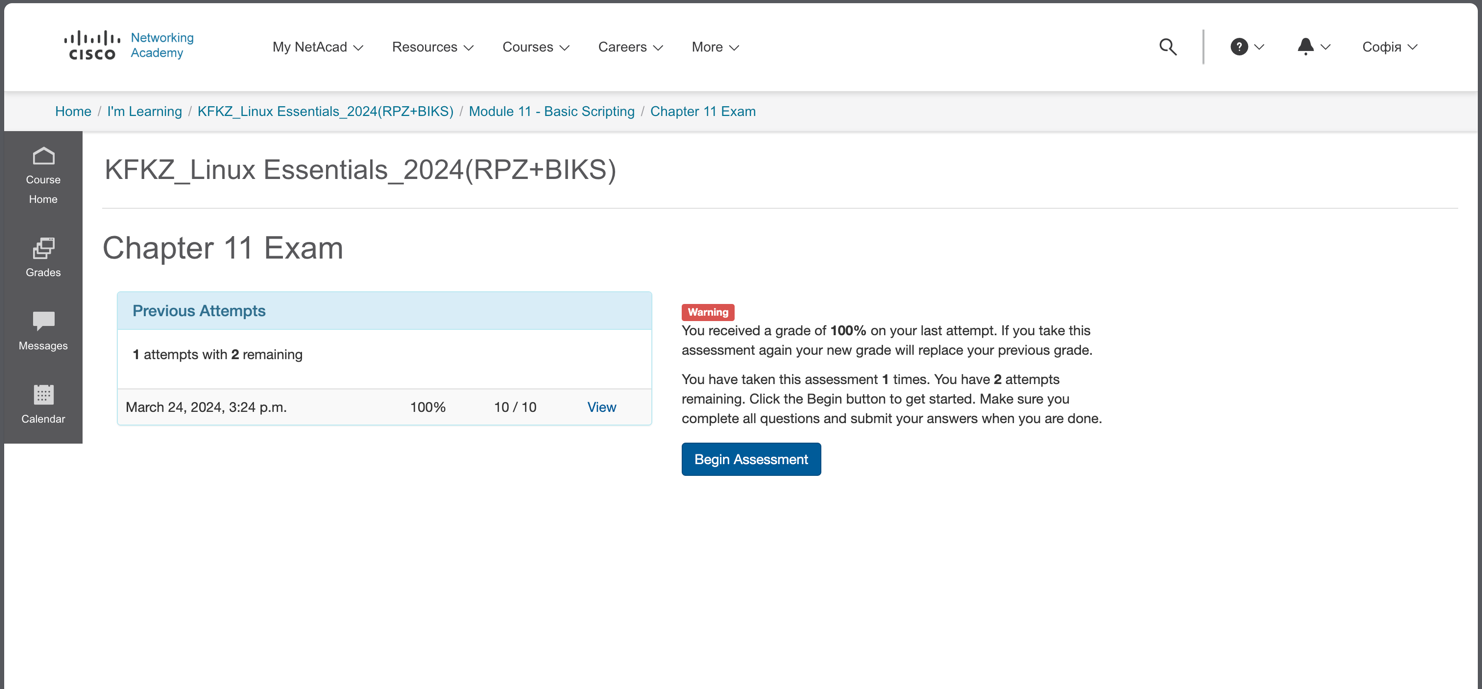
|  |  |
| --- | --- |
| Термін англійською | Термін українською |
| Shell script | Скрипт оболонки |
| Script editor | Редактор сценаріїв |
| Repetitive | Повторюваний |
| Consistency | Послідовність |
| Execute | Виконати |
| Loop | Цикл |
| Shebang | Шебанг (#!) |

1. Вивчіть матеріали онлайн-курсу академії Cisco “NDG Linux Essentials”:

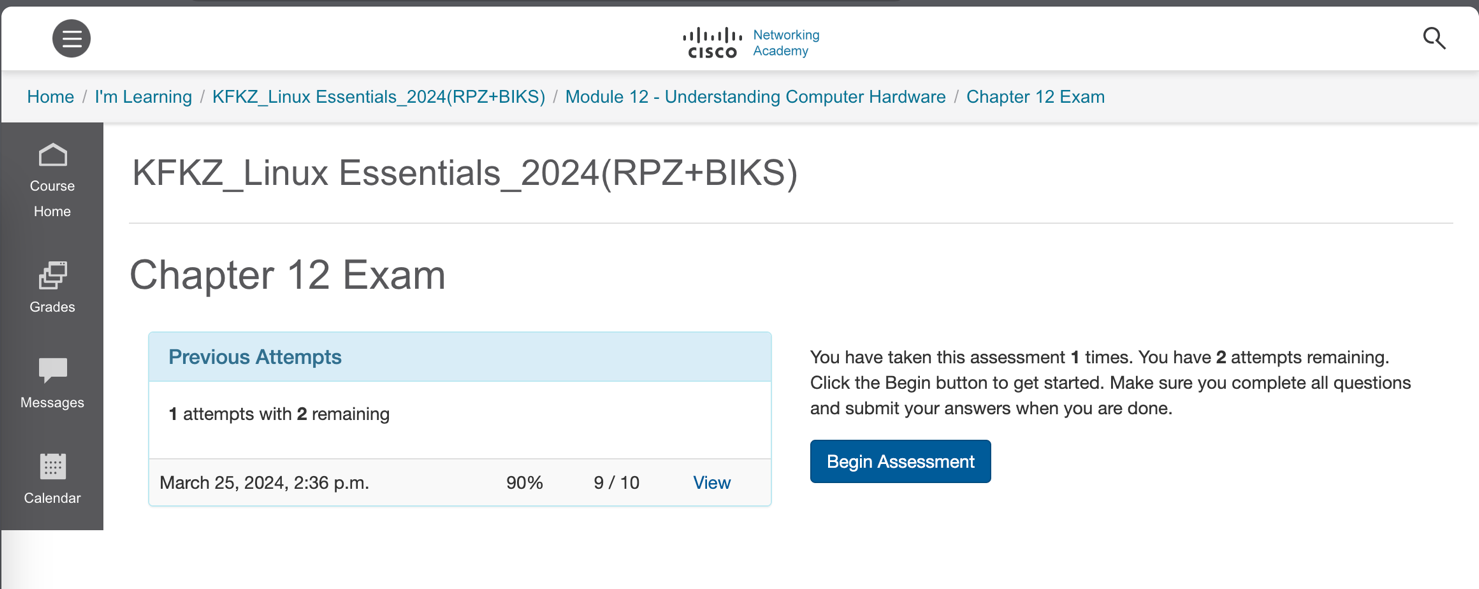
* Chapter 11 - Basic Scripting
* Chapter 12 - Understanding Computer Hardware

1. Пройдіть тестування у курсі NDG Linux Essentials за такими темами:

* Chapter 11 Exam



* Chapter 12 Exam



1. На базі розглянутого матеріалу дайте відповіді на наступні питання:
   1. \*Охарактеризуйте поняття скриптового сценарію у командній оболонці.

A shell script is a file that contains a set of commands that are executed one after the other when the script is run. Shell scripts have access to all shell commands, including logic. Thus, a script can check for a file or look for a certain output and change its behaviour accordingly. Scripts automate repetitive tasks, improve consistency, and can perform complex operations.

* 1. \*Яким чином створюються та редагуються скрипти, що треба зробити щоб запустити скрипт?
  2. \*\*Які основні компоненти материнської плати ви знаєте?
  3. \*\*Коротко охарактеризуйте для яких пристроїв оперують поняттями MBR та GPT?
  4. \*\*В чому суть операції монтування, для чого вона потрібна?

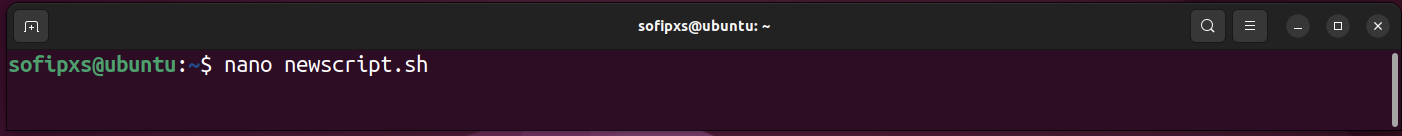
**Хід роботи:**

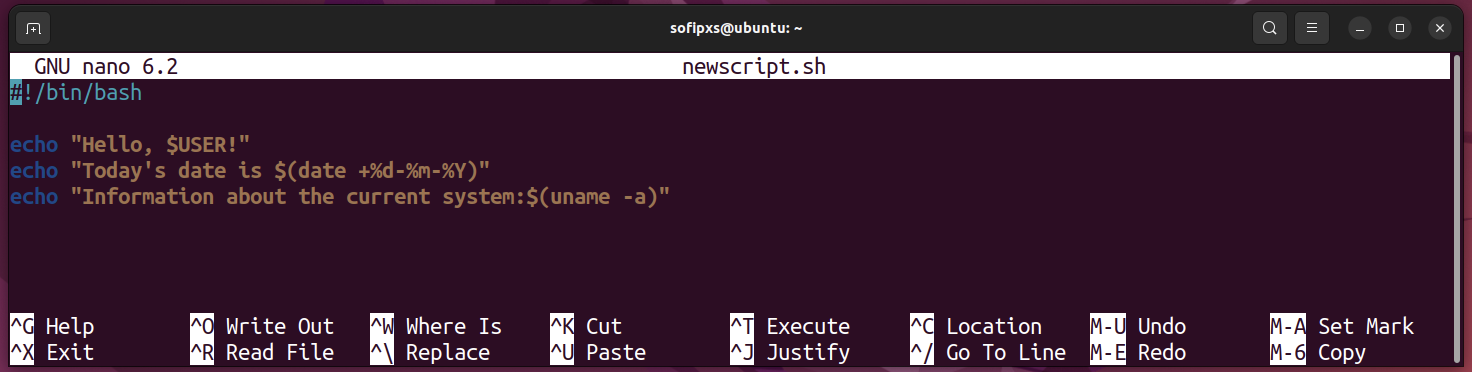
1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:
   1. Запустіть віртуальну машину VirtualBox, оберіть CentOS та запустіть її. Виконайте вхід в систему під користувачем: CentOS, пароль для входу: reverse ***(якщо виконуєте ЛР у 401 ауд.)*** та запустіть термінал.
   2. Запустіть віртуальну машину Ubuntu\_PC ***(якщо виконуєте завдання ЛР через академію netacad)***
   3. Запустіть свою операційну систему сімейства Linux ***(якщо працюєте на власному ПК та її встановили)*** та запустіть термінал.
2. Опрацюйте всі приклади команд, що представлені у лабораторних роботах курсу ***NDG Linux Essentials - Lab 11: Basic Scripting*** та ***Lab 12: Understanding Computer Hardware.*** Створіть таблицю для опису цих команд.

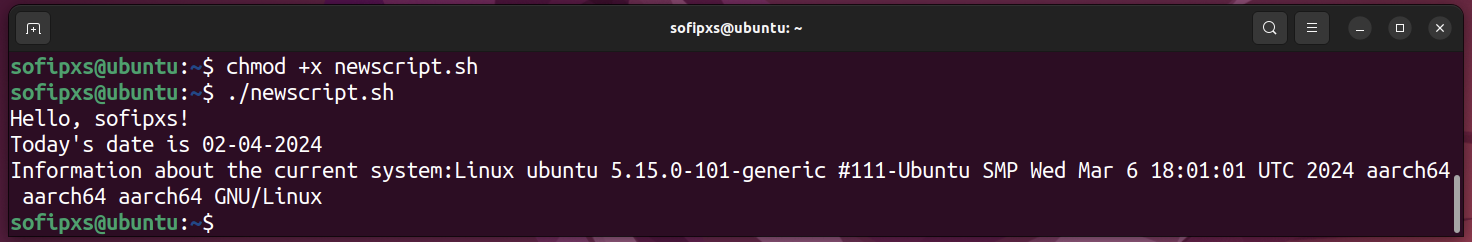
|  |  |
| --- | --- |
| Назва команди | Її призначення та функціональність |
| **Lab 11  Basic Scripting** | |
| vi myfile | Сreate a new file. |
| wq | Write the file to disk and quit. |
| j | Moves cursor down one line (same as down arrow) |
| k | Moves cursor up line (same as up arrow) |
| l | Moves cursor to the right one character (same as right arrow) |
| h | Moves cursor to the left one character (same as left arrow). |
| w | Moves cursor to beginning of next word. |
| e | Moves cursor to end of word. |
| b | Moves cursor to beginning of previous word. |
| $ | Moves cursor to end of current line (same as End key). |
| 0 (zero) | Moves cursor beginning of current line (same as Home key). |
| 3G | Jumps to third line (nG jumps to the nth line). |
| 1G | Jumps to first line. |
| Shift + G | Jumps to the last line. |
| dw | Delete the word |
| u | Undo the last operation. |
| 2dw | Delete two words. |
| xxxx | Delete four characters, one at a time. |
| 4u | Undo the last 4 operations and recover the deleted characters. |
| 14x | Delete 14 characters. |
| 5x | Delete the five characters to the left of the cursor (type 5 then **Shift+x**). |
| dd | Delete the current line. |
| p | Whatever was lasted deleted or yanked can be “pasted”. Paste the deleted lines below the current line. |
| 2u | Undo the last two operations |
| 2dd | Delete two lines, the current and the next. |
| 4w  D | Move to the fourth word then delete from the current position to the end of the line Shift+D. |
| J | Join two lines, the current and the next by typing a capital J (Shift+J). |
| yw | Copy (or “yank”) the current word. |
| P | Paste (or “put”) the copied word before the current cursor by typing Shift+p. |
| 1G  3J | Move to the first line, then join three lines. |
| :%s/text //g | Search for and delete the word text (add a space after the word text). |
| i | Enter insert mode |
| ~ | Shift+` changes letter to lower case |
| :w | Save the file. |
| 10l | 10 followed by the lowercase ‘L’ |
| a | Enter insert mode. |
| o | Open a blank line below the current line. |
| O | Open a blank line above the current line by pressing uppercase O. |
| :x | Will save and close the file. |
| :wq | Will write to file and quit. |
| :wq! | Will write to a read-only file, if possible, and quit. |
| ZZ | Will save and close. Notice that no colon : is used in this case. |
| :q! | Exit without saving changes. |
| :e! | Discard changes and reload file. |
| :w! | Write to read-only, if possible. |
| /line | Search forward for the word line. The cursor moves to the beginning of the first instance of the word line. |
| n | Search for the next instance of the word line. |
| ?line | Search backward for the word line. Cursor moves to the beginning of the previous instance of the word line. |
| cw  entry | You will replace the word line with the word entry. When you press cw you will be in insert mode and you will be able to type over the word line. |
| bash sample.sh | Run bash script. |
| chmod a+x | Give some new rules to script. |
| **Lab12 Understanding Computer Hardware** | |
| lscpu | In order to determine the type of CPU execute the lscpu command. |
| head -n 20 /proc/cpuinfo | Use the head command with the -n option to list the first 20 lines of the cpuinfo file. |
| head -n 20 /proc/cpuinfo | Use the head command with the -n option to list the first 20 lines of the cpuinfo file. |
| free -m  free -g | To discover how much RAM and swap space is being used, use the free command. The output shows the amount of memory in megabytes when the -m option is used and in gigabytes when the -g option is used: |
| lspci | To see what devices are connected to the PCI bus, use the lspci command. |
| lspci -k | lspci command with the -k option shows devices along with the kernel driver and modules used |
| lsusb | Attempt to list the USB connected devices. |
| lsmod | For hardware to function, the Linux kernel usually loads a driver or module. Use the lsmod command to view the currently loaded modules. |
| fdisk | The fdisk command is useful for identifying and manipulating disk storage resources on a system. Without the -l option, the fdisk command enters an interactive mode that is typically used to modify partitions on a disk device. |
| fdisk -l | The -l option lists the partition tables for the specified devices and then exits. |

1. Створіть скриптові сценарії з виводом текстових повідомлень для користувача (продемонструйте скріншоти):

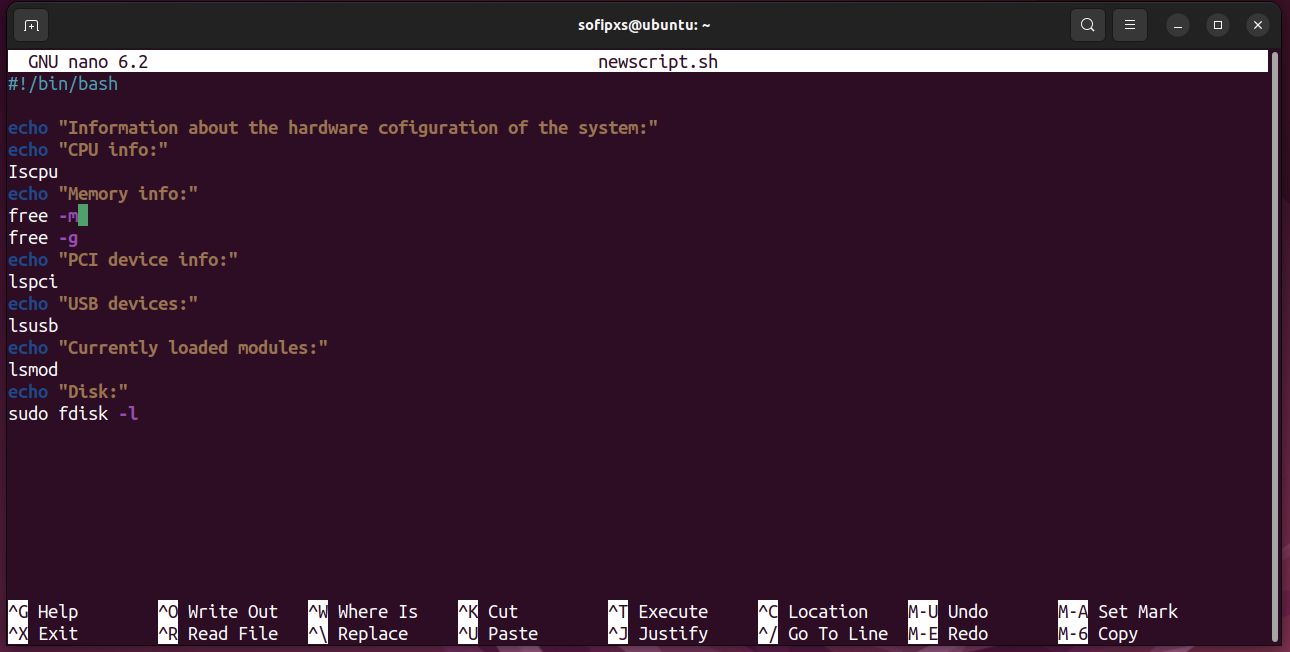
* сценарій має виводити привітання до поточного користувача вказуючи поточну дату та інформацію про поточну систему;

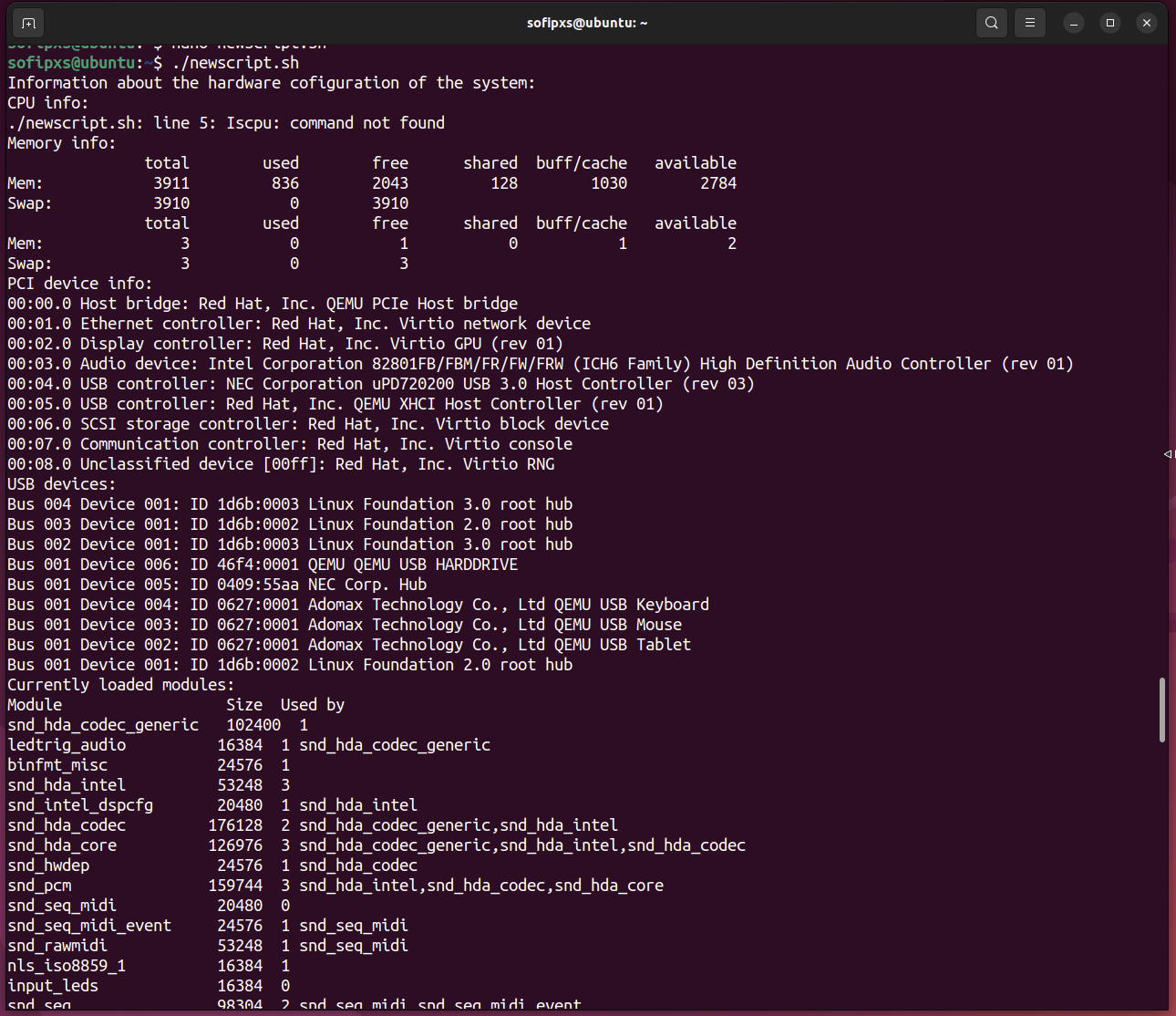




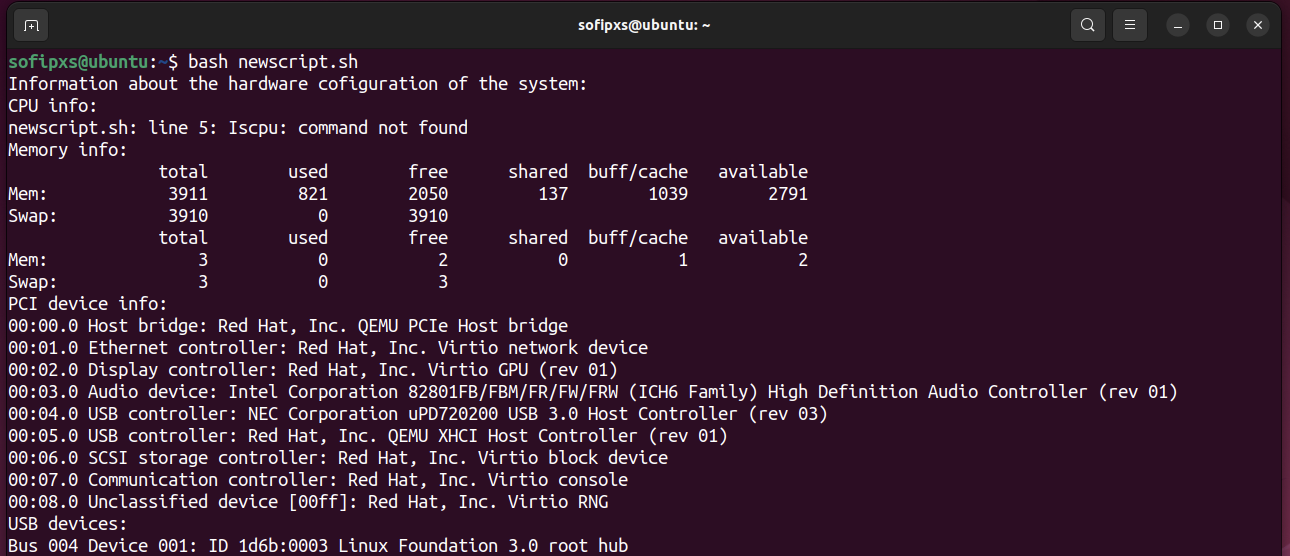


* \*сценарій має виводити інформацію про апаратну конфігурацію поточної системи (використовуйте команди розглянуті в Lab 12: Understanding Computer Hardware);

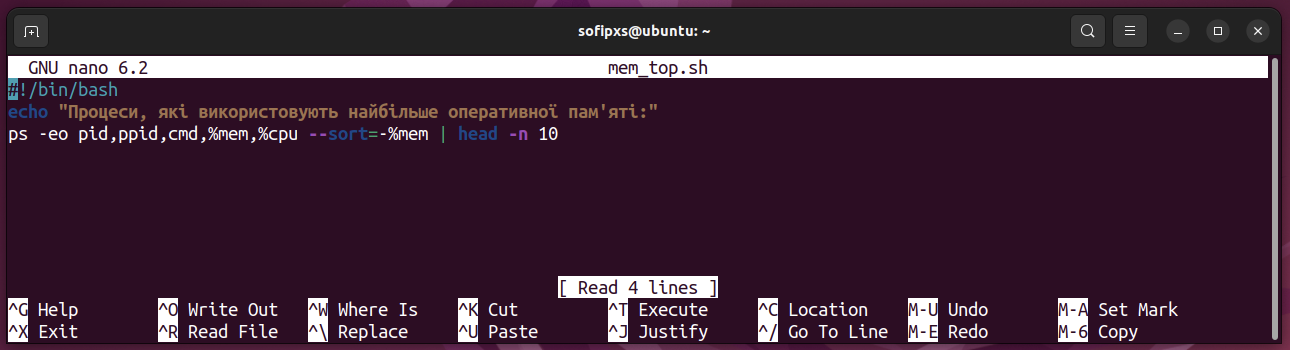


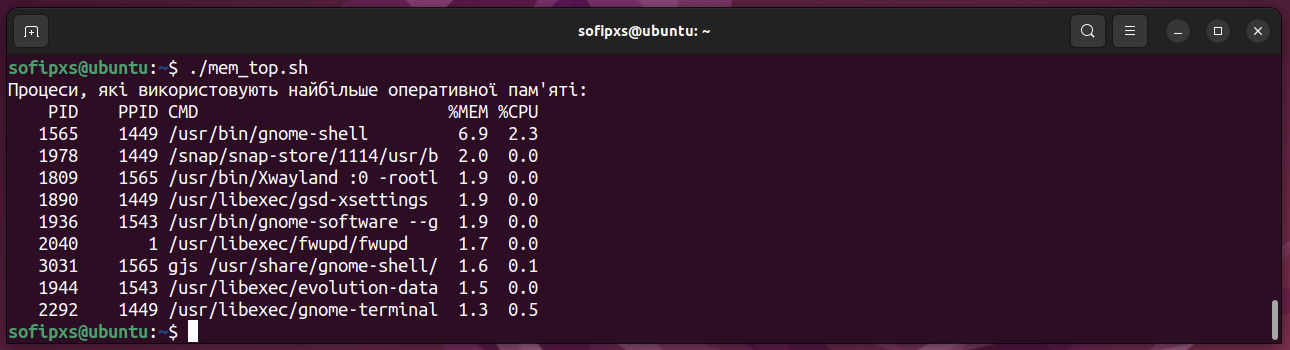


Another way to run the program is to type bash before the filename:



* \*\*наведіть свій приклад скриптового сценарію.





**Відповіді на контрольні запитання:**

1. В чому відмінність між командами arch та lscpu?
2. Якою командою можна отримати інформацію про стан використання RAM поточною системою?
3. \*Яким чином у скриптах можна опрацьовувати змінні та створювати розгалужені та циклічні сценарії?
4. \*Які команди для перегляду стану підключення периферійних пристроїв можна використати в терміналі?
5. \*\*Які можливості застунку gparted?

**Висновки:**

В ході виконання лабораторної роботи мною було досліджено … , більш детально теоретично досліджено питання …. Отримано практичні навики роботи з командами …, налаштування … ***(Якщо виникли труднощі, то їх описати)***