**“Київський фаховий коледж зв’язку”**

**Циклова комісія комп’ютерної та програмної інженерії**

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №7**

**з дисципліни: «Операційні системи»**

**Тема:** **“Створення скриптових сценаріїв та визначення апаратної конфігурації системи”**

**Виконавли студенти**

**групи КСМ-23Б**

**VelikieBomji: Бережний Т.І,**

**Лобода Д.Р. Чухманенко М.В.**

**Перевірила викладач**

**Сушанова В.С.**

**Тема: “Створення скриптових сценаріїв та визначення апаратної конфігурації системи”**

**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:**

**Shell Scripts in a Nutshell**

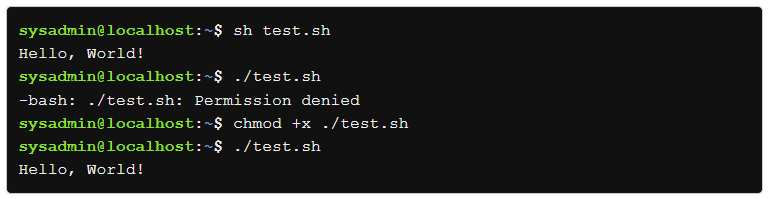
A shell script is a file of executable commands that has been stored in a text file. When the file is run, each command is executed. Shell scripts have access to all the commands of the shell, including logic. A script can therefore test for the presence of a file or look for particular output and change its behavior accordingly. You can build scripts to automate repetitive parts of your work, which frees your time and ensures consistency each time you use the script. For instance, if you run the same five commands every day, you can turn them into a shell script that reduces your work to one command.

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:

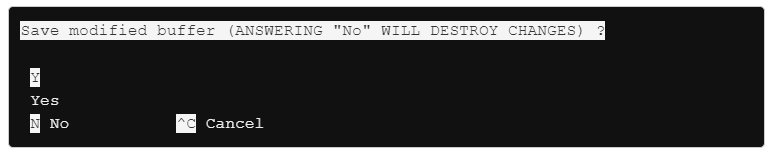


The *nano* editor has few features to get you on your way. You simply type with your keyboard, using the arrow keys to move around and the delete/backspace button to delete text. Along the bottom of the screen you can see some commands available to you, which are context-sensitive and change depending 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:

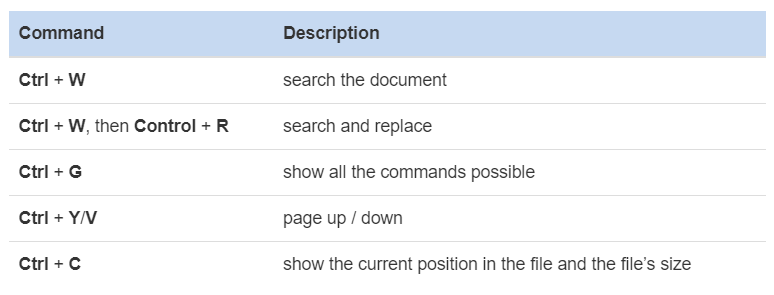


At this point, you can exit the program without saving by pressing the ***N*** key, or save first by pressing ***Y*** to save. The default is to save the file with the current file name. You can press the ***Enter*** key to save and exit.

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

* Титульний аркуш, тема та мета роботи
* Словник термінів
* Відповіді на п.4.1 та п.4.5 з завдань для попередньої підготовки

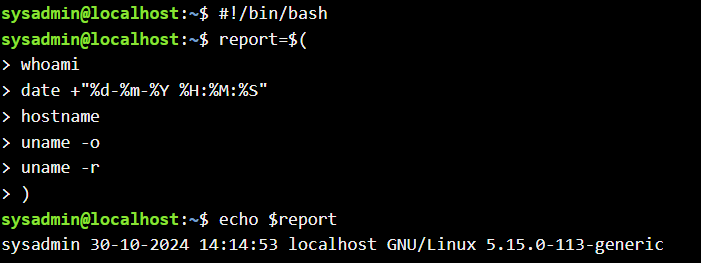
**Хід роботи:**

* 1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:
  2. Запустіть операційну систему Linux Ubuntu. Виконайте вхід в систему та запустіть термінал ***(якщо виконуєте ЛР у 401 ауд.)***..
  3. Запустіть віртуальну машину Ubuntu\_PC ***(якщо виконуєте завдання ЛР через академію netacad)***
  4. Запустіть свою операційну систему сімейства Linux ***(якщо працюєте на власному ПК та її встановили)*** та запустіть термінал.
  5. 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. ***(Чухманенко Максим)***

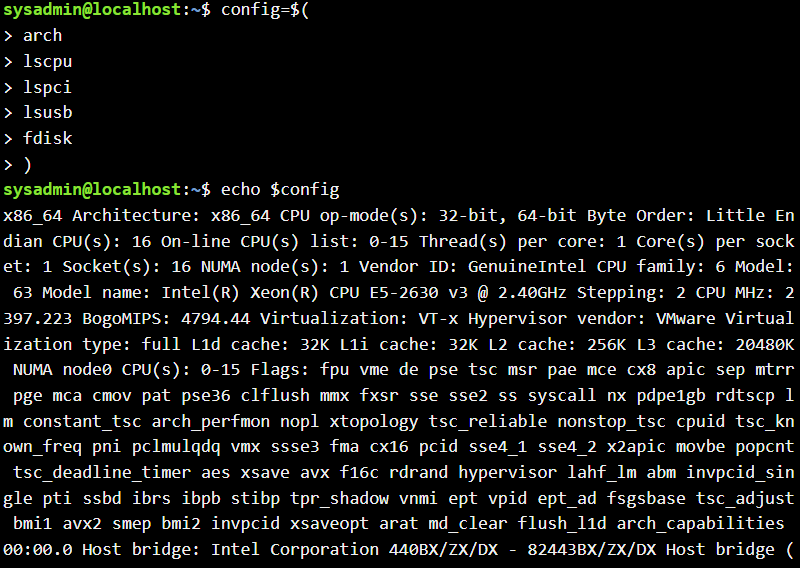
|  |  |
| --- | --- |
| Назва команди | Її призначення та функціональність |
| nano file | простий редактор тексту який легко освоїти |
| vi | складний але потужний редактор тексту |
| #!/bin/bash | вказує який інтерпретатор має бути використаний для виконання цього скрипту. |
| dw | видалити слово |
| u | відмінити останню операцію |
| grep | Команда grep повертає 0, якщо рядок знайдено, і 1 в іншому випадку. |
| x | видалити символ |
| dd | видалити поточну строку |
| p | вставити видалені строки під поточноюю строкою |
| 4w  D | Перейдіть до четвертого слова, а потім видаліть його з поточної позиції до кінця рядка Shift D |
| J | поєднати поточну та наступну строку |
| yw | копіювати поточне слово |
| P | вставити |
| :w | зберегти файл |
| if, elif, else | оператори умови |
| for | Цикли for використовуються, коли у вас є обмежений асортимент, наприклад список файлів, які ви хочете повторити (повторити та виконати код). |
| while | Цикл while працює зі списком невідомого розміру. |
| arch | виводить сімейство процесору |
| lscpu | розширена інформація про процесор |
| free | Щоб переглянути обсяг оперативної пам’яті у вашій системі, включаючи простір підкачки, виконайте команду free. Команда free має параметр -m, щоб примусово округлити вихідні дані до найближчого мегабайта (МБ), і параметр -g, щоб примусово округлити вихідні дані до найближчого гігабайта (ГБ): |
| lspci | пристрої підключені до шини pci |
| lsusb | пристрої підключені по usb |
| fdisk | характеристики диску |
| umount | відключення диску |
| head -n 20 /proc/cpuinfo | перелічити перші 20 рядків файлу cpuinfo |
| lsmod | загружені модулі |

**Примітка:** **Скріншоти** виконання команд в терміналі можна **не представляти**, достатньо **коротко описати команди в таблиці**.

* 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);



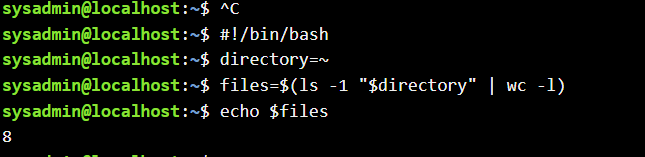
* \*\*наведіть свій приклад скриптового сценарію.

Сценарій підраховує кількість файлів в каталозі.

ls -1q directory: ls-виводить список файлів каталогу, -1-виводить кожен файл на новому рядку.

|- перенаправляє вхід команди

ws -l підраховує кількість рядків яке буде дорівнювати кількості файлів у каталозі



**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?

**Оформлення звіту:**

1. Титульний аркуш
2. Тема та мета роботи
3. Завдання попередньої підготовки
4. Основні позиції ходу роботи
5. Відповіді на контрольні запитання
6. Висновки за результатами роботи **(обов’язково!!!)**

**Система оцінювання лабораторної роботи:**

Виконано завдання базового рівня складності - **3 бали**

Виконано завдання базового та середнього рівня складності - **4 бали**

Виконано завдання всіх рівнів складності (в тому числі й підвищеного) - **5 балів**

Завдання середнього рівня складності позначені в завданнях (\*)

Завдання підвищеного рівня складності позначені в завданнях (\*\*)

**Примітка**: за виконання робіт в командах та оформлення звітів з використанням системи контролю версій (git) та англійської мови може бути нараховано **додатковий 1 бал**.