“Київський фаховий коледж зв’язку”

Циклова комісія комп’ютерної та програмної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни: «Операційні системи»

**Тема: “Команди Linux для управління процесами”**

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

*О.Михайленко*

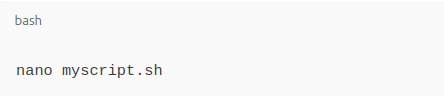
*1.*A **script** in a command shell is a text file that contains a sequence of commands executed by the shell in an operating system, typically Unix-like or Linux. Such scripts allow automating various tasks, making it easier to perform routine or complex operations. Scripts can be used to manage system processes, administer servers, configure environments, process files, and perform other operations.

*2.*

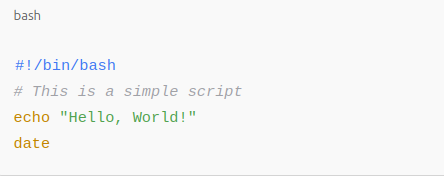
1. **Creating a Script**

Scripts are just plain text files containing shell commands. To create a script:

1. **Choose a text editor**: You can use any text editor to create a script. This can be simple editors like **nano**, **vi,** **vim**, or graphical editors if you're working in a graphical interface.



2. **Add script content**: Once in the editor, you can write the shell commands for your script. Here is an example of a simple script:

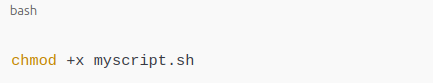


3. **Save and exit**: After writing your script, save the file and exit the editor.

* In nano, press Ctrl + O to save the file, then Ctrl + X to exit.
* In vim, press Esc, then type :wq to save and quit.

2. **Giving Execute Permissions**

Before running the script, you need to give it execute permissions. You can do this using the chmod command.

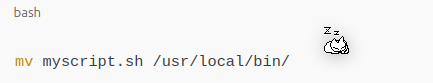


### 3. **Running the Script**

Once the script is created and has execute permissions, you can run it.

1. **Running from the current directory**: To run the script if it's in the current directory, use ./ to indicate the shell should look for the script here:
2. 

2. **Running the script from anywhere**: To be able to run the script from any directory without specifying the path, you can move it to a directory included in the PATH variable, or modify the PATH itself. For example:



Now, you can run the script by simply typing its name:



### 4. **Editing the Script**

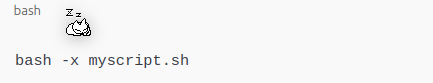
You can edit a script using the same text editor you used to create it. For example:



### 5. **Testing and Debugging**

Once the script is written, it's important to test it:

* **Console output**: Use echo statements in the script to print information to the console during execution to track results.
* **Debugging**: You can run the script with the -x option to see each command as it is executed:



*3.*The **motherboard** is the central component of a computer that houses all the key elements necessary for its operation. Here are the main components of a motherboard:



### 1. **Central Processing Unit (CPU)**

* **Description**: This is the main component responsible for performing calculations and managing all processes in the computer. The processor is installed in a **socket** on the motherboard.
* **Location**: Typically located in the center of the board.

### 2. **CPU Socket**

* **Description**: This is the connector where the central processor is installed. There are several types of sockets, depending on the processor manufacturer (Intel, AMD).

### 3. **RAM Slots**

* **Description**: These are slots for installing memory modules (RAM). The more slots, the more memory you can install.
* **Memory Types**: Typically uses DDR4 or DDR5 memory, depending on the platform.

### 4. **Chipset**

* **Description**: The chipset is a set of microchips that manages the communication between the CPU, memory, peripheral devices, and other motherboard components. It determines the functions and features supported by the system.
* **Main Components of the Chipset**:
  + **Southbridge**: Manages peripheral devices, storage, and low-speed devices.
  + **Northbridge**: Connects the CPU, memory, and graphics card.

### 5. **Expansion Slots**

* **Description**: These slots are for installing expansion cards such as graphics cards, sound cards, network cards, video capture cards, etc.
* **Types of Slots**:
  + **PCI Express (PCIe)**: Modern slots for graphics cards, network cards, and other high-speed devices.
  + **PCI**: Older standard for expansion cards.
  + **AGP**: An outdated standard used for graphics cards, now no longer used.

### 6. **Storage Slots (SATA / M.2 Slots)**

* **Description**: These slots are used to connect storage devices (hard drives, SSDs).
  + **SATA**: For connecting traditional hard drives and SSDs.
  + **M.2**: A modern interface for high-speed SSDs.

### 7. **I/O Ports**

* **Description**: Various ports for connecting external devices.
  + **USB**: For connecting devices like a mouse, keyboard, flash drives, and other peripherals.
  + **Audio Jacks**: For connecting headphones, microphones, and speakers.
  + **Ethernet (LAN)**: For network connectivity.
  + **HDMI/DisplayPort**: For connecting a monitor (if the motherboard has integrated graphics).

### 8. **Power Connector**

* **Description**: This is the connector that connects the power supply to the motherboard.
  + **24-pin ATX Connector**: The primary power connector for the motherboard.
  + **8-pin (4+4) CPU Power Connector**: An additional power connector for the CPU.

### 9. **CMOS Battery**

* **Description**: A small battery on the motherboard that powers the CMOS circuit, which stores the BIOS/UEFI settings such as date and time when the computer is powered off.

### 10. **BIOS/UEFI Chip**

* **Description**: This chip stores the basic input/output system (BIOS) or the Unified Extensible Firmware Interface (UEFI). It initializes hardware components and boots the operating system when the computer starts.

### 11. **Integrated Graphics**

* **Description**: Some motherboards come with integrated graphics chips, which provide video output without the need for a separate graphics card.

### 12. **Fan Headers**

* **Description**: These are connectors for attaching cooling fans (both for the CPU and the computer case).

### 13. **Front Panel Connectors**

* **Description**: These connectors allow the attachment of front panel components, such as power buttons, LEDs, USB ports, and audio connectors on the computer case.

### 14. **Thermal Sensors**

* **Description**: Temperature sensors that monitor the temperature of critical components, such as the CPU, chipset, and the case.

### 15. **Wi-Fi and Bluetooth Connectors**

* **Description**: Some motherboards feature built-in wireless modules for Wi-Fi and Bluetooth connectivity or support additional expansion cards for these functions.

4.

The terms **MBR (Master Boot Record)** and **GPT (GUID Partition Table)** refer to disk partitioning schemes used for organizing data on storage devices. These concepts are primarily associated with **hard drives (HDDs)** and **solid-state drives (SSDs)**.

### 1. **MBR (Master Boot Record)**

* **Used for**: Older storage devices (HDDs and SSDs).
* **Characteristics**:
  + Supports partitions up to 2 TB in size.
  + Can only have up to 4 primary partitions (or 3 primary partitions and 1 extended partition).
  + Used with BIOS-based systems.
* **Limitations**: MBR is limited by its structure and lacks the flexibility of GPT, especially for larger disks and modern systems.

### 2. **GPT (GUID Partition Table)**

* **Used for**: Modern storage devices (HDDs, SSDs) in newer systems.
* **Characteristics**:
  + Supports disks larger than 2 TB.
  + Allows for virtually unlimited partitions (typically up to 128).
  + Used with UEFI-based systems.
* **Advantages**: More robust, supports larger disks, and includes redundancy (a backup of the partition table) for better data integrity.

5.

**Mounting** is the process of making a storage device (such as a hard drive, SSD, or USB drive) accessible to the operating system by attaching it to a specific directory in the file system hierarchy. The term "mount" refers to linking the storage device's file system to the operating system so that files can be read from and written to it.

### Purpose of Mounting:

* **Accessing Data**: The main purpose of mounting is to allow the operating system and users to access files on a storage device. Without mounting, the system would not be able to recognize or interact with the device's contents.
* **File System Integration**: When a device is mounted, it is integrated into the system's file structure, allowing files and directories on the device to be accessed just like those on the system's primary drive.
* **Supporting Multiple Devices**: Mounting allows multiple devices (e.g., external drives, network shares) to be used simultaneously, each with its own file system, without interfering with each other.

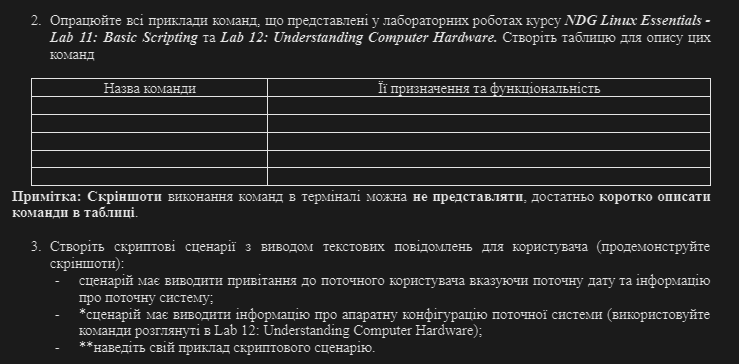
### Process:

1. **Mounting a Device**: When a device is mounted, the operating system assigns a directory (mount point) to the device's file system, making it accessible to users and applications.
   * For example, a USB drive might be mounted to /media/usb on Linux or D:\ on Windows.
2. **Unmounting**: When you are done using the device, it should be **unmounted** to ensure that all data is written and the device can be safely removed.

### Why It's Needed:

* **Access to External Devices**: Mounting is necessary for accessing external or secondary devices (such as USB drives, external HDDs, or network drives) that are not directly integrated into the system's main file system.
* **Seamless File Management**: It allows the operating system to treat different storage devices as part of the unified file system, making file management easier for users.

*Я.Трощинський*



2. Work through all of the sample commands presented in the NDG Linux Essentials labs - Lab 11: Basic Scripting and Lab 12: Understanding Computer Hardware. Create a table to describe these commands

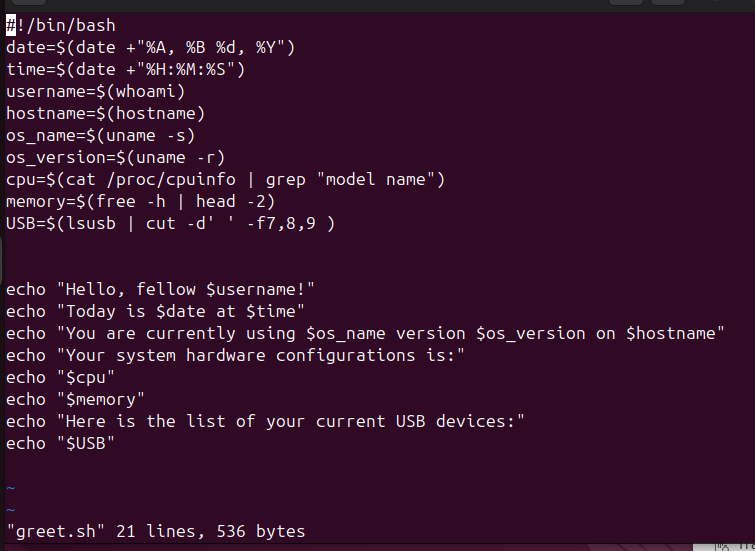
|  |  |
| --- | --- |
| **Command** | **Its function** |
| **vi file** | **Creates a file, and if file is already created opens a text editor** |
| **vi file.sh** | **Runs a script written in the file** |
| **bash file.sh** | **Same as vi file.sh** |
| **chmod file** | **Change permission to the file** |
| **read var** | **Read user input and put it in the $var** |
| **Lscpu** | **Determines the type of CPU** |
| **free -m** | **Show how much RAM is being used in mb** |
| **Lspci** | **Shows what devices are connected** |
| **Lsusb** | **Shows list of USB connected devices** |
| **Lsmod** | **Used to view the currently loaded modules** |
| **fdisk -l** | **Lists the disk devices** |

3. Create scripts that display text messages for the user (show screenshots):

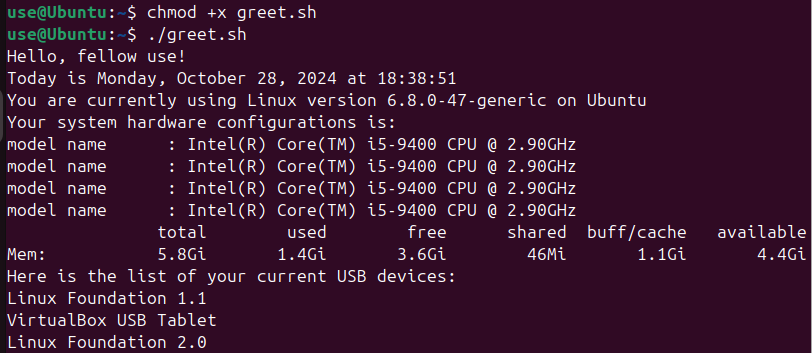
* the script should display a greeting to the current user, indicating the current date and information about the current system;
* the script should display information about the hardware configuration of the current system (use the commands discussed in Lab 12: Understanding Computer Hardware);

Because of the similarity between the end of first and the start of second I thought to combine them into one script.

What is inside of file:



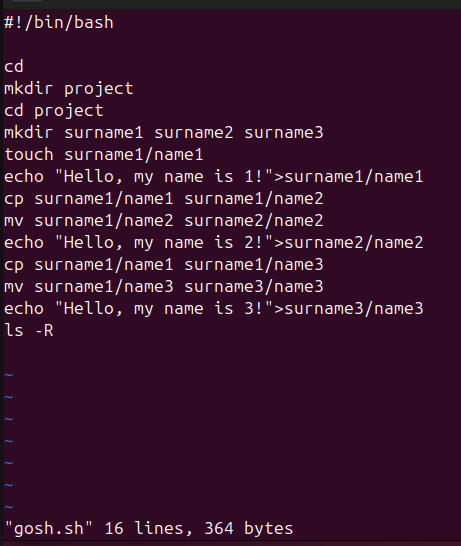
What is it doing:



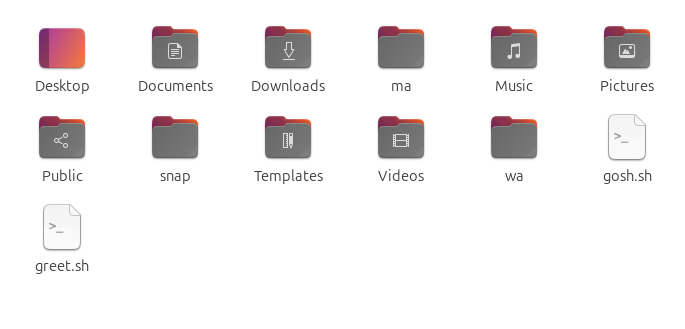
* Provide an example of your script.

For practice with vi text redactor I reopened lab 5 and did the same thing from there but using one file-function

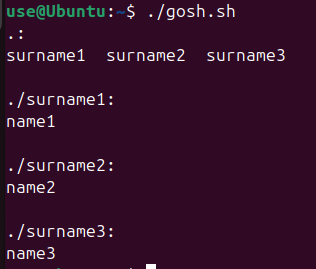
Inside of file:

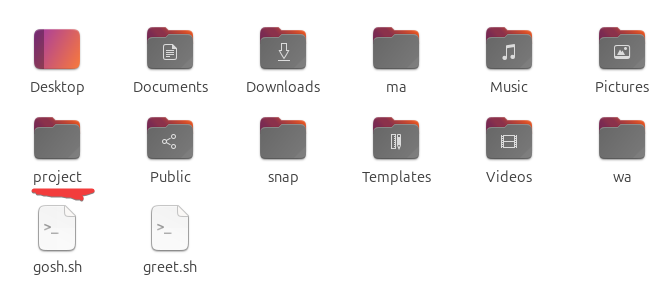


What is it doing:

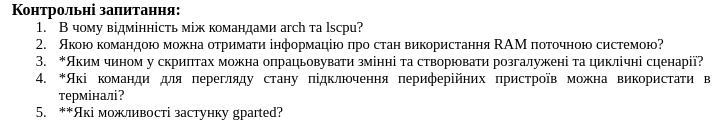
Before:

After:





*Б.Когут*

**

1. What is the difference between the arch and lscpu commands?

2. Which command can be used to obtain information about the current system’s RAM usage?

3. How can variables be processed in scripts and branching or looping scenarios created?

4. Which terminal commands can be used to view the connection status of peripheral devices?

5. What features does the gparted application offer?

1. Command arch shows the processor architecture,

Command lscpu show more detailed information about the processor architecture

2. Can use command free -h to look your information about RAM.

3. In Bash, variables can be defined by assigning value to them, for example:

variable\_name=value. To use the value of a variable, you need to write $variable\_name.

Conditional statements are created using if, else, and for loops – for, while.

4. lsusb – to display a list of USB devices;

lspci – to display a list of PCI devices;

dmesg – to view system messages that may contain information about connected devices;

cat /proc/partitions – to view partitions and storage devices.

5. gparted (Gnome Partition Editor) is a graphical tool for disk partition management. Its features include:

* Creating, deleting, resizing, moving, formatting, and checking partitions.
* Supporting various file systems: ext4, NTFS, FAT32, etc.
* Visualizing partitions and their usage on the disk.
* Applying partition changes with the option to undo operations before completing.

**Conclusion**

With this lab work, we gained practical skills in working with the Bash command shell and became familiar with the basic operations involved in working with script scenarios.