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

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

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №1**

**з дисципліни: «Операційні системи»**

**Тема: « Знайомство з робочим середовищем віртуальних машин та особливостями операційної системи Linux »**

**Виконав(ли) студент(и)**

**групи РПЗ-23б**

**Команда Booblick Team: Мірошніченко А.**

**Михальов В.**

**Перевірила викладач**

**Сушанова В.С.**

**Київ 2025**

**Тема: “Знайомство з робочим середовищем віртуальних машин та особливостями операційної системи Linux”**

**Мета роботи:**

1. Знайомство з гіпервізорами різного типу, віртуалізацією при роботі з операційними системами.

2. Знайомство з основними видами сучасних ОС, короткий огляд їх можливостей.

**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:**

**VIRTUAL MACHINES REDISCOVERED**

While IBM has had a virtual-machine product available for four decades, and a few other companies, including Oracle and Hewlett-Packard, have recently added virtual-machine support to their high-end enterprise servers, the idea of virtualization has largely been ignored in the PC world until recently. But in the past few years, a combination of new needs, new software, and new technologies have combined to make it a hot topic.

First the needs. Many companies have traditionally run their mail servers, Web servers, FTP servers, and other servers on separate computers, sometimes with different operating systems. They see virtualization as a way to run them all on the same machine without having a crash of one server bring down the rest.

Virtualization is also popular in the Web hosting world. Without virtualization, Web hosting customers are forced to choose between **shared hosting** (which just gives them a login account on a Web server, but no control over the server software) and dedicated hosting (which gives them their own machine, which is very flexible but not cost effective for small to medium Websites). When a Web hosting company offers virtual machines for rent, a single physical machine can run many virtual machines, each of which appears to be a complete machine. Customers who rent a virtual machine can run whatever operating system and software they want to, but at a fraction of the cost of a dedicated server (because the same physical machine supports many virtual machines at the same time).

Another use of virtualization is for end users who want to be able to run two or more operating systems at the same time, say Windows and Linux, because some of their favorite application packages run on one and some run on the other. This situation is illustrated in Fig. 1-29(a), where the term "virtual machine monitor'' has been renamed **type 1 hypervisor,** which is commonly used nowadays because "virtual machine monitor'' requires more keystrokes than people are prepared to put up with now. Note that many authors use the terms interchangeably though.

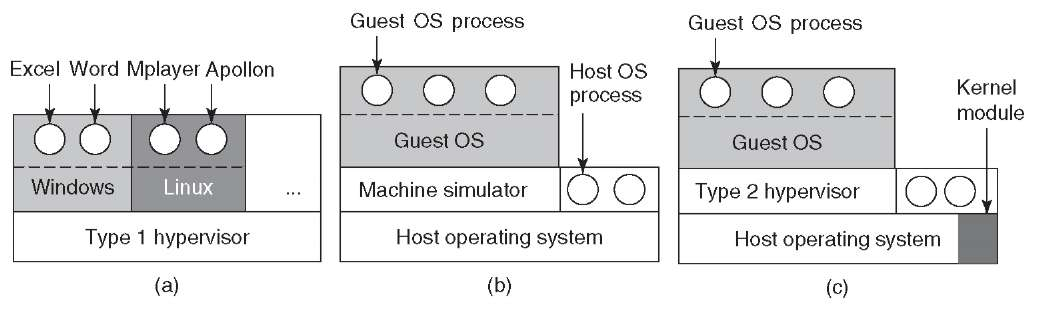


Figure 1-29. (a) A type 1 hypervisor. (b) A pure type 2 hypervisor. (c) A practical type 2 hypervisor

While no one disputes the attractiveness of virtual machines today, the problem then was implementation. In order to run virtual machine software on a computer, its CPU must be virtualizable (Popek and Goldberg, 1974). In a nutshell, here is the problem. When an operating system running on a virtual machine (in user mode) executes a privileged instruction, such as modifying the PSW or doing I/O, it is essential that the hardware trap to the virtual-machine monitor so the instruction can be emulated in software. On some CPUs—notably the Pentium, its predecessors, and its clones—attempts to execute privileged instructions in user mode are just ignored. This property made it impossible to have virtual machines on this hardware, which explains the lack of interest in the x86 world. Of course, there were interpreters for the Pentium, such as *Bochs,* that ran on the Pentium, but with a performance loss of one to two orders of magnitude, they were not useful for serious work.

This situation changed as a result of several academic research projects in the 1990s and early years of this millennium, notably Disco at Stanford (Bugnion et al., 1997) and Xen at Cambridge University (Barham et al., 2003). These research papers led to several commercial products (e.g., VMware Workstation and Xen) and a revival of interest in virtual machines. Besides VMware and Xen, popular hypervisors today include KVM (for the Linux kernel), VirtualBox (by Oracle), and Hyper-V (by Microsoft).

Some of these early research projects improved the performance over interpreters like *Bochs* by translating blocks of code on the fly, storing them in an internal cache, and then reusing them if they were executed again. This improved the performance considerably, and led to what we will call **machine simulators,** as shown in Fig. 1-29(b). However, although this technique, known as **binary translation,** helped improve matters, the resulting systems, while good enough to publish papers about in academic conferences, were still not fast enough to use in commercial environments where performance matters a lot.

The next step in improving performance was to add a kernel module to do some of the heavy lifting, as shown in Fig. 1-29(c). In practice now, all commercially available hypervisors, such as VMware Workstation, use this hybrid strategy (and have many other improvements as well). They are called **type 2 hypervisors** by everyone, so we will (somewhat grudgingly) go along and use this name in the rest of this book, even though we would prefer to called them type 1.7 hypervisors to reflect the fact that they are not entirely user-mode programs.

In practice, the real distinction between a type 1 hypervisor and a type 2 hyper- visor is that a type 2 makes uses of a **host operating system** and its file system to create processes, store files, and so on. A type 1 hypervisor has no underlying support and must perform all these functions itself.

After a type 2 hypervisor is started, it reads the installation CD-ROM (or CD- ROM image file) for the chosen **guest operating system** and installs the guest OS on a virtual disk, which is just a big file in the host operating system's file system. Type 1 hypervisors cannot do this because there is no host operating system to store files on. They must manage their own storage on a raw disk partition.

When the guest operating system is booted, it does the same thing it does on the actual hardware, typically starting up some background processes and then a GUI. To the user, the guest operating system behaves the same way it does when running on the bare metal even though that is not the case here.

**The Java Virtual Machine**

Another area where virtual machines are used, but in a somewhat different way, is for running Java programs. When Sun Microsystems invented the Java programming language, it also invented a virtual machine (i.e., a computer architecture) called the **JVM (Java Virtual Machine).** The Java compiler produces code for JVM, which then typically is executed by a software JVM interpreter. The advantage of this approach is that the JVM code can be shipped over the Internet to any computer that has a JVM interpreter and run there. If the compiler had produced SPARC or x86 binary programs, for example, they could not have been shipped and run anywhere as easily. (Of course, Sun could have produced a compiler that produced SPARC binaries and then distributed a SPARC interpreter, but JVM is a much simpler architecture to interpret.) Another advantage of using JVM is that if the interpreter is implemented properly, which is not completely trivial, incoming JVM programs can be checked for safety and then executed in a protected environment so they cannot steal data or do any damage.

**BASIC INFORMATION ABOUT LINUX**

**Linux is a kernel**

The definition of the word Linux depends on the context in which it is used. Linux means the kernel of the system, which is the central controller of everything that happens on the computer.

When most people refer to Linux, they are really referring to a combination of software called GNU/Linux, which defines the operating system. GNU is the free software that provides open source equivalents of many common UNIX commands. The Linux part of this combination is the Linux kernel, which is the core of the operating system. The kernel is loaded at boot time and stays running to manage every aspect of the functioning system.

The story of Linux begins with UNIX, an operating system developed at AT&T Bell Labs in the 1970s. UNIX is written in the C language making it uniquely portable amongst competing operating systems, which were typically closely tied to the hardware for which they were written. It quickly gained popularity in research and academic settings, as well as amongst programmers who were attracted to its modularity. Over time it was modified and forked (that is, people modified it, and those modifications served as the basis for other systems) such that at present there are many different variants of UNIX. However, UNIX is now both a trademark and a specification, owned by an industry consortium called the Open Group. Only software that has been certified by the Open Group may call itself UNIX.

Linux started in 1991 as a hobby project of Linus Torvalds, a Finnish-born computer scientist studying at the University of Helsinki. Frustrated by the licensing of MINIX, a UNIX-like operating system designed for educational use, and its creator’s desire not to make it a full operating system, Linus decided to create his own OS kernel.

From this humble beginning, Linux has grown to be the dominant operating system on the Internet, and arguably the most important computer program of any kind. Despite adopting all the requirements of the UNIX specification, Linux has not been certified, so Linux really isn’t UNIX! It’s just… UNIX-like. Prior to and alongside this development was the GNU Project, created by Richard Stallman in 1983. While GNU initially focused on building their own operating system, they ultimately were far more effective at building tools that go along with a UNIX-like operating system, such as the editors, compilers and user interfaces that make a kernel usable. Since the source was all freely available, Linux programmers were able to incorporate the GNU tools to provide a complete operating system. As such, many of the tools and utilities that are part of the Linux system evolved from these early GNU tools.

**Linux is Open Source**

Historically, most software has been issued under a closed-source license, meaning that you get the right to use the machine code, but cannot see the source code. Often the license explicitly says that you may not attempt to reverse engineer the machine code back to source code to figure out what it does! The development of Linux closely parallels the rise of open source software. Open source takes a source-centric view of software. The open source philosophy is that you have a right to obtain the software source code and to modify it for your own use. Linux adopted this philosophy to great success. Linus made the source programming code (the instructions a computer uses to operate) freely available, allowing others to join in and shape this fledgling operating system. It was not the first system to be developed by a volunteer group, but since it was built from scratch, early adopters could influence the project’s direction. People took the source, made changes, and shared them back with the rest of the group, greatly accelerating the pace of development, and ensuring mistakes from other operating systems were not repeated.

**Linux Has Distributions**

People that say their computer runs Linux usually refer to the kernel, tools, and suite of applications that come bundled together in what is referred to as a distribution.

Take Linux and the GNU tools, add some user-facing applications like a web browser and an email client, and you have a full Linux system. Individuals and even companies started bundling all this software into distributions almost as soon as Linux became usable. The distribution includes tools that take care of setting up the storage, installing the kernel, and installing the rest of the software. The full-featured distributions also include tools to manage the system and a package manager to help you add and remove software after the installation is complete.

Like UNIX, there are distributions suited to every imaginable purpose. There are distributions that focus on running servers, desktops, or even industry-specific tools such as electronics design or statistical computing. The major players in the market can be traced back to either Red Hat, Debian or Slackware. The most visible difference between Red Hat and Debian derivatives is the package manager though there are other differences in everything from file locations to political philosophies.

**Linux Embraces the CLI**

There are two basic types of interfaces available that allow you to interact with the operating system. The typical computer user today is most familiar with a *graphical user interface (GUI)*. In a GUI, applications present themselves in windows that can be resized and moved around. There are menus and tools to help users navigate. Graphical applications include web browsers, graphics editing tools and email, to name a few.

The second type of interface is the *command line interface (CLI)*, a text-based interface to the computer. The CLI relies primarily on keyboard input. Everything the user wants the computer to do is relayed by typing commands rather than clicking on icons. It can be said that when a user clicks on an icon, the computer is telling the user what to do, but, when the user types a command, they are telling the computer what to do.

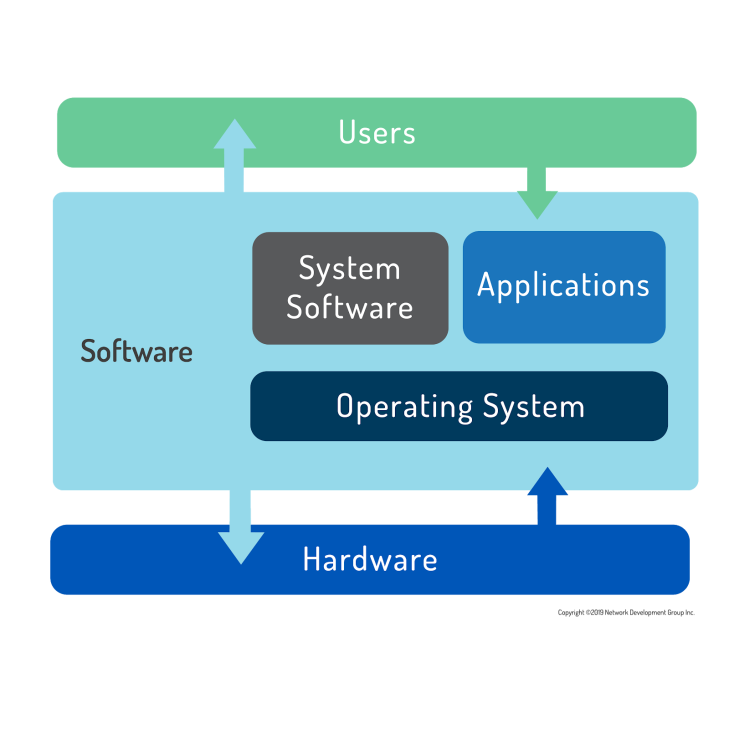
Typically operating systems offer both GUI and CLI interfaces. However, most consumer operating systems (Windows, macOS) are designed to shield the user from the complexity of the CLI. The Linux community is different in that it positively celebrates the CLI for its power, speed and ability to accomplish a vast array of tasks with a single command line instruction. The virtual machines used for the chapters and labs in this course provide a CLI for you to practice on without fear of damaging anything. When a user first encounters the CLI, they can find it challenging because it requires memorizing a dizzying amount of commands and their options. However, once a user has learned the structure of how commands are used, where the necessary files and directories are located and how to navigate the hierarchy of a filesystem, they can be immensely productive. This capability provides more precise control, greater speed and the ability to easily automate tasks through scripting.

Furthermore, by learning the CLI, a user can easily be productive almost instantly on ANY distribution of Linux, reducing the amount of time needed to familiarize themselves with a system because of variations in a GUI.

**OPERATING SYSTEMS**

An *operating system* is software that runs on a computing device and manages the hardware and software components that make up a functional computing system.

Modern operating systems don’t just manage hardware and software resources, they schedule programs to run in a multi-tasking manner (sharing the processor so that multiple tasks can occur apparently simultaneously), provide standard services that allow users and programs to request something happen (for example a print job) from the operating system, and provided it’s properly requested, the operating system will accept the request and perform the function needed. Desktop and server operating systems are by nature more complex than an operating system that runs on a single-purpose device such as a firewall, or a mobile phone. From a simple set-top box that provides a menu interface for a cable provider, to supercomputers and massive, parallel computing clusters, the generic term operating system is used to describe whatever software is booted and run on that device.



Computer users today have a choice mainly between three major operating systems: Microsoft Windows, Apple macOS, and Linux. Of the three major operating systems listed only Microsoft Windows is unique in its underlying code. Apple’s macOS is a fully-qualified UNIX distribution based on BSD Unix (an operating system distributed until 1995), complemented by a large amount of proprietary code. It runs on hardware specifically optimized to work with Apple software. Linux can be any one of hundreds of distribution packages designed or optimized for whatever task is required. Only Microsoft Windows is based on a proprietary code base that isn’t either UNIX- or Linux-based. A user can easily interact with any of these systems by pointing and clicking their way through everyday productivity tasks that all behave similarly regardless of the underlying operating system. Except for Windows, which is mostly administered via the GUI, most system administration tasks are performed using typed commands in a terminal. An administrator that is familiar with UNIX can typically perform tasks on a Linux system and vice versa. Many Linux command line functions also have Microsoft equivalents that administrators use to do their work efficiently.

**Завдання для попередньої підготовки.**

1. \*Прочитайте короткі теоретичні відомості до лабораторної роботи та зробіть невеликий словник базових англійських термінів з питань класифікації віртуальних середовищ.

|  |  |
| --- | --- |
| Термін англійською | Термін українською |
| Operating System | Операційна система |
| Kernel | Ядро |
| Multitasking | Багатозадачність |
| Multi-user | Багатокористувацька |
| Real-time Operating System (RTOS) | Операційна система реального часу |
| Distributed Operating System | Розподілена операційна система |
| Embedded Operating System | Вбудована операційна система |
| File System | Файлова система |
| GUI (Graphical User Interface) | Графічний інтерфейс користувача |
| CLI (Command-Line Interface) | Командний рядок |
| Virtual Memory | Віртуальна пам'ять |
| Process | Процес |
| Thread | Потік |
| Booting | Завантаження |
| Device Driver | Драйвер пристрою |

1. \*\*Прочитавши матеріал з коротких теоретичних відомостей дайте відповіді на наступні питання:
   1. Охарактеризуйте поняття «гіпервізор». Які бувають їх типи?

Гіпервізором називається програмне забезпечення, яке дозволяє відокремити фізичне обладнання від програм, запущених в операційній системі комп'ютера.

Гіпервізор 1-го типу - це шар програмного забезпечення, який ми встановлюємо безпосередньо поверх фізичного сервера та його базового обладнання

Гіпервізор 2-го типу – це така технологія встановлюється «другим шаром» поверх встановленої операційної системи.

* 1. Перерахуйте основні компоненти та можливості гіпервізорів відповідно до свого варіанту (порядковий номер по журналу), табл.1.

В-12

**Основні компоненти VMware ESXi та екосистеми VMware**

1. **ESXi Hypervisor** – основа всієї платформи віртуалізації.
2. **vCenter Server** – централізоване управління кластерами ESXi, моніторинг, балансування навантаження.
3. **vSphere Client (HTML5/Web Client)** – веб-інтерфейс для адміністрування.
4. **vMotion** – міграція віртуальних машин між серверами без простою.
5. **vSAN (Virtual SAN)** – програмно-реалізоване сховище для віртуалізації.
6. **HA (High Availability)** – автоматичний перезапуск ВМ у разі збою апаратного забезпечення.
7. **DRS (Distributed Resource Scheduler)** – балансування навантаження між хостами.
8. **NSX** – програмно-налаштована мережа для безпеки та автоматизації.
9. **VMware Horizon** – віртуалізація робочих столів (VDI).
10. **vCloud Suite** – комплексне рішення для хмарної інфраструктури.

**Основні можливості гіпервізорів VMware**

* Підтримка великої кількості ОС (Windows, Linux, macOS, BSD тощо).
* Висока продуктивність та ефективне використання ресурсів.
* Гнучке керування мережею, зокрема VLAN, SDN через NSX.
* Інтеграція з хмарними платформами (AWS, Azure, Google Cloud).
* Автоматизація та оркестрація через **vRealize Automation**.
* Захист даних (шифрування, бекапи, вбудовані засоби безпеки).
* Масштабованість та підтримка великих кластерів.

В-11

**Основні компоненти VirtualBox**

1. **VirtualBox Manager** – графічний інтерфейс для створення та управління віртуальними машинами (ВМ).
2. **VBoxManage** – консольний інструмент для автоматизації та управління ВМ через командний рядок.
3. **Guest Additions** – додаткові драйвери та утиліти для покращеної інтеграції гостьової ОС із хост-системою.
4. **Extension Pack** – розширює можливості VirtualBox (наприклад, підтримка USB 2.0/3.0, шифрування дисків, PXE-завантаження).
5. **Snapshot** – функція створення знімків стану ВМ для швидкого відновлення.
6. **Shared Folders** – спільні папки між хостовою та гостьовою ОС.
7. **Bridged Networking** – режим мережевого підключення, який дозволяє ВМ отримувати IP-адресу в локальній мережі.

**Основні можливості VirtualBox**

* Підтримка різних операційних систем (Windows, Linux, macOS, BSD тощо).
* Підтримка апаратної віртуалізації (Intel VT-x, AMD-V).
* Створення та клонування віртуальних машин.
* Підтримка декількох типів віртуальних дисків (VDI, VMDK, VHD).
* Віддалене керування через VirtualBox Remote Desktop Protocol (VRDP).
* Гнучке налаштування ресурсів ВМ (ЦП, ОЗП, відеопам’ять, мережа).
* Сумісність із образами VMware, Hyper-V та інших платформ.
* Підтримка експорту ВМ у формат OVA/OVF для перенесення в інші середовища.
* Безкоштовне використання з відкритим вихідним кодом.

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

* Chapter 1 - Introduction to Linux
* Chapter 2 - Operating Systems

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

* Chapter 02 Exam

1. Підготувати в електронному вигляді початковий варіант звіту:

* Титульний аркуш, тема та мета роботи
* Словник термінів
* Відповіді на п.2.1 та п.2.2 з завдань для попередньої підготовки

**Хід роботи.**

* 1. Подивіться ознайомчі відео та демонстраційні матеріали з наступних напрямків:
  2. Linux - Кращі дистрибутиви 2023 <https://youtu.be/PahmJBU9HKA?si=maxRf0nZlqs2hFGU>
  3. ТОП 5 ПРИЧИН ЧОМУ АЙТІШНИКУ ВАРТО ПЕРЕЙТИ НА ЛІНУКС

Доступ: <https://youtu.be/bP3_mZKezvM?si=sM3Mpc9JQ_0bY9Yd>

* 1. Як встановити Linux разом з Windows спосіб #1 Microsoft Store

Доступ: <https://youtu.be/eEdGl6HvSdM?si=WDbwa71i034D2rQj>

* 1. Як встановити Linux разом з Windows спосіб #2 Dual Boot

Доступ: <https://youtu.be/Hfky8TEyXss?si=ilduY167LS-vKl9y>

* 1. Як встановлювати програми на Linux. Linux українською #1

Доступ: <https://youtu.be/M8XHJME6cxI?si=L0Koom59jTRnPXnU>

* 1. Як зробити панель завдань Linux як у Windows. Linux українською #2

Доступ: <https://youtu.be/9szAz-A4gaM?si=LxaVueluI3tKRb1r>

* 1. Як встановити Ubuntu на VirtualBox <https://youtu.be/ADOaHm1VZII?si=hG5kDRsajFn7se8d>
  2. The Shell (Linux) <https://drive.google.com/open?id=0B0PV0_SM0LoDSVNPWUVRdUxaN2s>
  3. Linux Desktop Environments: XFCE vs GNOME vs KDE

Доступ: <https://youtu.be/2JBGQfPR5xQ?si=euswD7IHrODd-6JH>

1. Дайте відповіді на наступні питання.
   1. Перерахуйте етапи для розгортання операційної системи на базі віртуальної машини VirtualBox.
   2. Чи є якісь апаратні обмеження при встановленні 32- та 64-бітних ОС?
   3. Які основні етапи при встановленні OS Linux в текстовому режимі?
   4. \*Яким чином можна до установити графічні оболонки Gnome та KDE в Linux, якщо вона вже встановлена в текстовому режимі (вкажіть необхідні команди та пакети)?
   5. \*\*Дайте коротку характеристику графічних інтерфейсів, що використовуються в різних дистрибутивах Linux відповідно до свого варіанту (порядковий номер по журналу), табл.2.

Таблиця 2

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант | 1, 4, 7, 10, 13, 16, 19, 22, 25 | 2, 5, 8, 11, 14, 17, 20, 23 | 3, 6, 9, 12, 15, 18, 21, 24 |
| Графічна оболонка | KDE та Fluxbox | Gnome та JWM | Xfce та Fvwm |

**Контрольні запитання**

1. Порівняйте гіпервізори типу 1 та типу 2, яка між ними відмінність та сфера їх застосування?
2. Розкрийте поняття «GNU GPL», яка його основна концепція?
3. В чому суть програмного забезпечення з відкритим кодом?
4. \*Що таке дистрибутив?
5. \*Які задачі системного адміністрування можна реалізувати на базі ОС Linux?
6. \*Як пов'язані між собою ОС Android та Linux?
7. \*\*Основні можливості та сфера використання Embedded Linux?
8. \*\*Яким чином можна змінити типу завантаження Linux: в текстовому режимі (3 рівень) або графічному (рівень 5)? Чим відрізняються режими CLI та GUI?

**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!!!)**