Педяш В.В., Йона Л.Г., Мазур Г.Д.

ОПЕРАЦІЙНІ СИСТЕМИ

Методичні рекомендації для практичних та лабораторних занять

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Міжнародний гуманітарний університет
Кафедра комп’ютерної інженерії та інноваційних технологій

Педяш В.В., Йона Л.Г., Мазур Г.Д.

ОПЕРАЦІЙНІ СИСТЕМИ

Методичні рекомендації для практичних та лабораторних занять здобувачів вищої освіти зі спеціальностей:

A4.09 Середня освіта (Інформатика),
F2 Інженерія програмного забезпечення,
F3 Комп'ютерні науки,
F7 Комп'ютерна інженерія,
F5 Кібербезпека та захист інформації,
G5 Електроніка, електронні комунікації, приладобудування та радіотехніка

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету (протокол № 12 від 23 червня 2025 року)

Педяш В.В., Йона Л.Г., Мазур Г.Д.

Операційні системи: Методичні рекомендації для практичних та лабораторних занять [Електронне видання]. / Педяш В.В., Йона Л.Г., Мазур Г.Д. Кафедра комп’ютерної інженерії та інноваційних технологій Міжнародного гуманітарного університету. Одеса, 2025. – 121 с.

Методичні рекомендації для практичних та лабораторних занять з дисципліни "Операційні системи" розроблено відповідно до навчального плану підготовки здобувачів вищої освіти. Посібник містить тематичні практичні завдання, рекомендації щодо їх виконання, а також лабораторні завдання, спрямовані на поглиблення знань і формування практичних навичок роботи з операційними системами, зокрема у середовищі Linux.

Матеріали призначені для здобувачів першого (бакалавського) рівня вищої освіти спеціальностей: A4 Середня освіта (Інформатика), F2 Інженерія програмного забезпечення, F3 Комп'ютерні науки, F7 Комп'ютерна інженерія, F5 Кібербезпека та захист інформації, G5 Електроніка, електронні комунікації, приладобудування та радіотехніка.

Виконання практичних та індивідуальних завдань проводиться в безпечному віртуальному середовищі (Oracle VirtualBox), що дозволяє відпрацювати навички роботи з командним рядком, управлінням файловою системою, процесами, мережевими підключеннями та іншими компонентами операційної системи без ризику для основної системи. Це сприяє розвитку системного мислення, поглибленню розуміння архітектури ОС, а також підготовці до виконання індивідуальних та кваліфікаційних робіт у галузі інформаційних технологій.

ЗМІСТ

ВСТУП

Сучасна інформаційна інфраструктура будь-якої організації – від невеликого підприємства до глобальної корпорації ґрунтується на ефективному використанні операційних систем, які забезпечують узгоджену роботу всіх програмних і апаратних компонентів. Саме операційна система виконує роль базового програмного середовища, що керує ресурсами, забезпечує безпеку, взаємодію з мережевими сервісами та стабільність функціонування обчислювальних систем. Розуміння принципів її роботи, механізмів управління ресурсами, організації доступу та взаємодії з користувачами є фундаментальною компетенцією фахівця в галузі інформаційних технологій. Дисципліна «Операційні системи» спрямована на формування базових практичних навичок роботи з сучасними Unix-подібними операційними системами, зокрема Linux, через освоєння командного рядка, управління файловою системою, процесами, користувачами, мережевими налаштуваннями та автоматизацію типових адміністративних завдань.

Дані методичні вказівки призначені для використання під час проведення практичних і лабораторних занять з дисципліни «Операційні системи» та містять матеріали для проведення десяти навчальних робіт. Навчальний матеріал структуровано таким чином, щоб забезпечити послідовний і логічний перехід від опрацювання теоретичних аспектів до їх безпосереднього застосування під час виконання практичних і лабораторних завдань, що сприяє глибшому та цілісному засвоєнню тем курсу.

Практичні заняття проводяться на основі розділу «Ключові положення», у якому викладено необхідні теоретичні відомості та пояснення принципів функціонування основних підсистем операційної системи, зокрема файлової системи, механізмів управління процесами, засобів мережевої взаємодії та забезпечення безпеки. У цьому ж розділі наведено опис основних інструментів, команд і механізмів, що використовуються в середовищі Linux. Опрацювання ключових положень дозволяє студентам сформувати системне уявлення про відповідну тему, підготуватися до виконання практичних дій і усвідомлено підійти до розв’язання завдань лабораторної роботи.

Після засвоєння матеріалу практичного заняття навчальний процес логічно продовжується лабораторним заняттям з тієї ж тематики, яке виконується відповідно до розділу «Лабораторне завдання». У цьому розділі подано структуровану покрокову інструкцію з виконання конкретних завдань, що передбачають налаштування системних компонентів, перевірку працездатності служб, аналіз поточного стану системи та фіксацію отриманих результатів. Лабораторні роботи виконуються у безпечному віртуальному середовищі Oracle VirtualBox, що дозволяє відпрацьовувати навички роботи з командним рядком, управління обліковими записами та правами доступу, мережевими підключеннями й іншими компонентами операційної системи без ризику для основної робочої системи.

Запропонована організація навчального матеріалу забезпечує логічну послідовність навчання, за якої теоретичні знання, отримані під час практичних занять, закріплюються та поглиблюються в процесі виконання лабораторних робіт. Використання віртуального середовища дозволяє моделювати реальні сценарії системного адміністрування, експериментувати з конфігураціями та поступово набувати практичного досвіду роботи з операційними системами в умовах, наближених до реальних.

Сформовані в межах цього курсу компетенції створюють міцний фундамент для подальшого вивчення суміжних дисциплін, зокрема комп'ютерних мереж, кібербезпеки, системного програмування та DevOps-практик, а також для підготовки до міжнародних сертифікацій у сфері системного адміністрування й управління IT-інфраструктурою.

Лабораторна робота № 1

Тема: Інсталяція та налаштування віртуального середовища Oracle VirtualBox

Мета роботи: Ознайомитися з технологією віртуалізації, навчитися створювати віртуальну машину, встановлювати Linux, налаштовувати параметри ВМ та встановлювати гостьові додатки для покращення інтеграції.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

У сучасному світі інформаційних технологій віртуалізація стала фундаментальною концепцією, що дозволяє максимально ефективно використовувати апаратні ресурси комп'ютерних систем. Актуальність засобів віртуалізації зумовлена необхідністю оптимізації витрат на IT-інфраструктуру, підвищення гнучкості та масштабованості обчислювальних ресурсів, а також забезпечення ізоляції різних програмних середовищ. Віртуалізація дозволяє запускати кілька операційних систем одночасно на одному фізичному комп'ютері, створюючи незалежні віртуальні машини з власними ресурсами пам'яті, процесора та дискового простору. Це особливо важливо для навчальних закладів, де студенти можуть безпечно експериментувати з різними операційними системами без ризику пошкодження основного робочого середовища. Крім того, віртуалізація є основою сучасних хмарних технологій та контейнеризації, що робить її вивчення критично важливим для майбутніх фахівців з інформаційних технологій.

Гіпервізор є ключовим компонентом будь-якої системи віртуалізації, представляючи собою програмне або програмно-апаратне забезпечення, яке створює та управляє віртуальними машинами. Існують два основні типи гіпервізорів: Type 1 (bare-metal) та Type 2 (hosted). Гіпервізори першого типу встановлюються безпосередньо на апаратне забезпечення та мають прямий доступ до фізичних ресурсів, що забезпечує високу продуктивність та стабільність. Гіпервізори другого типу функціонують як додатки всередині операційної системи хоста, що робить їх менш ефективними, але більш зручними для розробки та тестування.

У таблиці 1.1 представлено порівняння основних сучасних гіпервізорів з їх ключовими характеристиками та областями застосування.

VMware ESXi є одним з найпопулярніших enterprise-рішень для віртуалізації, що забезпечує високу продуктивність та надійність у корпоративному середовищі. Цей гіпервізор першого типу встановлюється безпосередньо на сервери та підтримує широкий спектр апаратного забезпечення, забезпечуючи ефективне управління ресурсами та високу доступність віртуальних машин. ESXi інтегрується з екосистемою VMware vSphere, що дозволяє створювати масштабовані віртуальні інфраструктури з функціями живої міграції, балансування навантаження та автоматичного відновлення після збоїв.

Microsoft Hyper-V представляє собою гіпервізор, глибоко інтегрований з операційними системами Windows Server та Windows 10/11. Архітектура Hyper-V базується на мікроядерному підході, де батьківська операційна система управляє гіпервізором та надає послуги для дочірніх віртуальних машин. Особливістю Hyper-V є його здатність ефективно працювати з Windows-навантаженнями завдяки оптимізації на рівні ядра системи та підтримці спеціальних драйверів Integration Services.

Таблиця 1.1 - Порівняння сучасних гіпервізорів

Гіпервізор	**Тип**	**Розробник**	**Ліцензія**	**Область застосування**
VMware ESXi	Type 1	VMware	Комерційна	Корпоративні дата-центри
Hyper-V	Type 1	Microsoft	Комерційна/Безкоштовна	Windows-інфраструктура
Xen	Type 1	Xen Project	Open Source	Хмарні платформи
VirtualBox	Type 2	Oracle	Open Source/Комерційна	Розробка та тестування
VMware Workstation	Type 2	VMware	Комерційна	Професійна розробка
QEMU/KVM	Type 1	Red Hat/Community	Open Source	Linux-інфраструктура

Xen гіпервізор відіграє важливу роль у сфері хмарних обчислень, будучи основою для багатьох великих хмарних платформ, включаючи Amazon Web Services. Унікальною особливістю Xen є підтримка як повної віртуалізації, так і паравіртуалізації, що дозволяє досягти високої продуктивності навіть на старіших процесорах без апаратної підтримки віртуалізації. Архітектура Xen передбачає наявність привілейованої доменної машини Dom0, яка управляє іншими віртуальними машинами DomU через спеціальний інтерфейс гіпервізора.

Oracle VirtualBox займає особливе місце серед гіпервізорів завдяки своїй універсальності та доступності для широкого кола користувачів. Створений компанією Innotek у 2007 році, VirtualBox пройшов шлях від невеликого стартап-продукту до одного з найпопулярніших засобів віртуалізації після придбання Oracle Corporation у 2010 році. Розвиток VirtualBox характеризується постійним розширенням функціональності та підтримкою нових операційних систем, при цьому продукт залишається безкоштовним для особистого та навчального використання. Архітектура VirtualBox побудована за принципом модульності, де основний движок віртуалізації доповнюється спеціалізованими компонентами для різних операційних систем хостів.

Архітектурно VirtualBox складається з кількох ключових компонентів, що забезпечують його кросплатформну сумісність та функціональність. Ядром системи є VirtualBox VMM (Virtual Machine Monitor), який відповідає за емуляцію апаратного забезпечення та управління ресурсами віртуальних машин. GUI-інтерфейс VirtualBox Manager надає зручні засоби для створення,

налаштування та управління віртуальними машинами через графічний інтерфейс. Додатково система включає VBoxSVC - фонову службу, яка забезпечує комунікацію між різними компонентами та процесами VirtualBox. Гостьові доповнення (Guest Additions) представляють собою набір драйверів та утиліт, які встановлюються всередині віртуальної машини для покращення інтеграції з хостовою системою та підвищення продуктивності.

Рисунок 1.1 демонструє архітектуру файлової системи VirtualBox з організацією віртуальних машин та їх компонентів. На верхньому рівні розташовується VirtualBox Manager, який керує всією інфраструктурою віртуалізації. Під ним знаходяться каталоги окремих віртуальних машин (VM1, VM2, VM3), кожна з яких містить власний набір конфігураційних та даних файлів. У каталозі кожної віртуальної машини розташовані різноманітні типи файлів, кожен з яких виконує специфічні функції в роботі віртуальної інфраструктури. Структура файлів віртуальної машини є логічно організованою та включає як основні компоненти для функціонування системи, так і допоміжні елементи для управління та моніторингу.

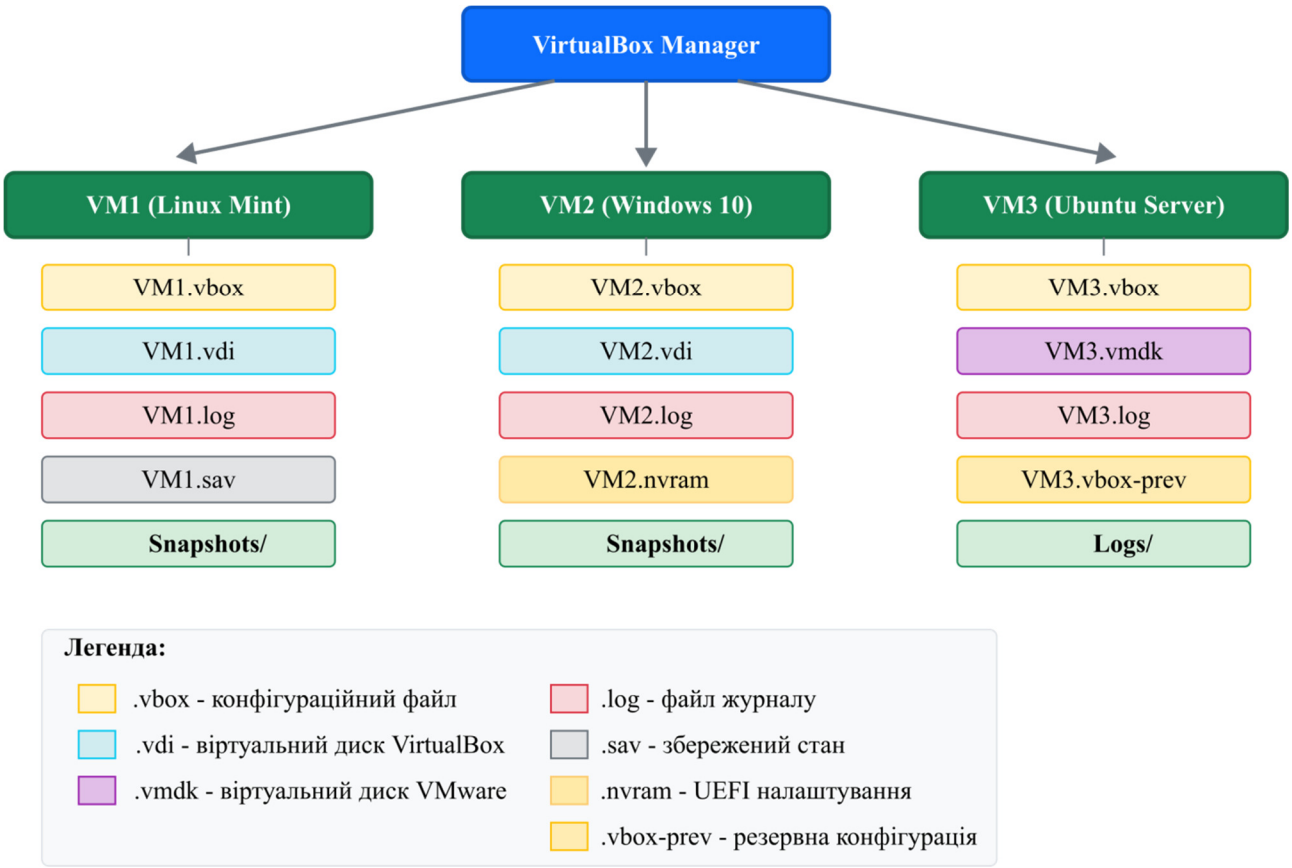

Рисунок 1.1 – Архітектура файлової системи VirtualBox

Таблиця 1.2 наведена нижче систематизує основні типи файлів віртуальних машин VirtualBox та їх функціональне призначення.

Детальний аналіз призначення типових файлів віртуальної машини демонструє складність внутрішньої архітектури VirtualBox. Конфігураційний файл .vbox є текстовим документом у форматі XML, який містить повний опис віртуального апаратного забезпечення машини, включаючи параметри

процесора, обсяг пам'яті, налаштування мережевих адаптерів, підключені носії та інші характеристики. Віртуальні диски .vdi представляють собою контейнери, які зберігають дані файлової системи гостьової ОС у спеціальному форматі, що дозволяє динамічно розширювати або стискати їх розмір залежно від реального використання дискового простору. Файли журналів .log автоматично створюються VirtualBox для кожного сеансу роботи віртуальної машини і містять детальну інформацію про процеси завантаження, помилки системи та взаємодію з віртуальним апаратним забезпеченням. Каталог Snapshots зберігає файли знімків стану віртуальної машини, які включають як конфігураційні дані, так і повну копію оперативної пам'яті на момент створення снепшота, що дозволяє миттєво відновити роботу системи до попереднього стану.

Таблиця 1.2 - Основні типи файлів віртуальних машин VirtualBox

Розширення файлу	**Назва типу**	**Призначення**
.vbox	Конфігураційний файл	Зберігає налаштування ВМ у XML-форматі
.vdi	Віртуальний диск VirtualBox	Містить дані файлової системи гостьової ОС
.vmdk	Віртуальний диск VMware	Диск у форматі VMware для сумісності
.vhd	Віртуальний диск Microsoft	Диск у форматі Hyper-V для сумісності
.log	Журнал роботи	Записи про функціонування ВМ
.sav	Збережений стан	Повний знімок стану ВМ у пам'яті
.vbox-prev	Резервна конфігурація	Попередня версія налаштувань ВМ
.nvram	Енергонезалежна пам'ять	UEFI/BIOS налаштування ВМ

У контексті роботи з VirtualBox важливо розуміти базову термінологію віртуалізації. Хостова система (Host OS) - це операційна система, яка безпосередньо встановлена на фізичному комп'ютері та на якій запускається VirtualBox. Гостьова система (Guest OS) являє собою операційну систему, що функціонує всередині віртуальної машини та ізольована від хостової системи. Віртуальна машина (Virtual Machine, VM) - це програмний контейнер, який емулює повноцінний комп'ютер з власними віртуальними компонентами апаратного забезпечення. Снепшот (Snapshot) дозволяє зберегти поточний стан віртуальної машини для можливості швидкого відновлення у майбутньому.

Суть практичного завдання полягає у засвоєнні студентами основних навичок роботи з системою віртуалізації через практичне створення та налаштування віртуальної машини. Завдання охоплює повний цикл роботи з VirtualBox - від інсталяції програмного забезпечення до запуску повноцінної гостьової операційної системи Linux Mint. Цей процес дозволяє студентам на практиці зрозуміти принципи віртуалізації, особливості розподілу ресурсів між хостовою та гостьовою системами, а також специфіку налаштування віртуального апаратного забезпечення.

Виконання лабораторної роботи передбачає не лише технічну реалізацію віртуальної машини, але й розуміння архітектурних особливостей взаємодії

різних компонентів віртуальної інфраструктури. Студенти отримають практичний досвід роботи з ISO-образами операційних систем, налаштування віртуальних носіїв інформації, управління ресурсами віртуальної машини та інсталяції гостьових доповнень для оптимізації продуктивності системи.

2 КЛЮЧОВІ ПИТАННЯ

1. Які переваги надають засоби віртуалізації в сучасному IT-середовищі та чому їх використання є актуальним для навчального процесу?
2. У чому полягають принципові відмінності між гіпервізорами першого типу (Type 1) та другого типу (Type 2), та які приклади кожного з них ви можете навести?
3. Які основні компоненти архітектури Oracle VirtualBox та яке призначення кожного з них у забезпеченні функціонування віртуальних машин?
4. Яке призначення мають файли з розширеннями .vbox, .vdi, .log та .sav у структурі віртуальної машини VirtualBox?
5. Які додаткові компоненти можна встановити разом з Oracle VirtualBox під час інсталяції та навіщо вони потрібні?
6. У чому різниця між хостовою та гостьовою операційними системами у контексті віртуалізації?
7. Які рекомендовані параметри апаратних ресурсів (оперативна пам'ять, процесор, дисковий простір) для створення віртуальної машини з Linux Mint?
8. Що таке гостьові доповнення (Guest Additions) і які функції вони забезпечують для покращення роботи віртуальної машини?
9. Яким чином здійснюється підключення ISO-образу операційної системи до віртуальної машини для інсталяції?
10. Які типи віртуальних жорстких дисків підтримує VirtualBox та в чому переваги використання динамічно розширюваних дисків?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Виконання інсталяції Oracle VirtualBox

3.1.1 Завантажити виконуваний файл на сайті VirtualBox (https://www.virtualbox.org).

3.1.2 Інсталяцію Oracle VM VirtualBox можна розпочати двічі клацнувши на виконуваному файлі. Після запуску інсталятор відобразить діалогове вікно привітання і дозволить вам вибрати, куди встановити Oracle VM VirtualBox і які компоненти інсталювати. На додаток до програми Oracle VM VirtualBox доступні наступні компоненти:

– Підтримка USB. Цей пакет містить спеціальні драйвери для вашого хоста Windows, які потрібні Oracle VM VirtualBox для повної підтримки USB-пристроїв у ваших віртуальних машинах;

– Мережева робота. Цей пакунок містить додаткові мережеві драйвери для вашого хоста Windows, які потрібні Oracle VM VirtualBox для підтримки з'єднаних мереж (Bridged Networking). Це дозволяє отримати доступ до віртуальних

мережевих карт вашої віртуальної машини з інших комп'ютерів у фізичній мережі;

– Підтримка Python. Цей пакунок містить підтримку сценаріїв Python для API Oracle VM VirtualBox.

Рисунок 1.2 – Сторінка завантаження Oracle VM VirtualBox

Для роботи пакунка потрібно, щоб у системі вже було інстальовано Windows Python. Залежно від конфігурації Windows, ви можете побачити попередження про непідписані драйвери або подібні. Натисніть кнопку Продовжити для цих попереджень, оскільки інакше Oracle VM VirtualBox може працювати некоректно після інсталяції.

Інсталятор створить групу Oracle VM VirtualBox у меню "Пуск" Windows, за допомогою якої ви зможете запустити програму і отримати доступ до її документації. Зі стандартними налаштуваннями Oracle VM VirtualBox буде встановлено для всіх користувачів локальної системи.

Після встановлення ви можете запустити Oracle VM VirtualBox наступним чином: - Хости Windows. У меню "Програми" клацніть на пункті в групі VirtualBox. На деяких платформах Windows ви також можете ввести VirtualBox в пошуковому рядку меню "Пуск".

3.2 Створення нової віртуальної машини.

3.2.1 Перейти до вікна Менеджера VM VirtualBox та натиснути кнопку "Створити".

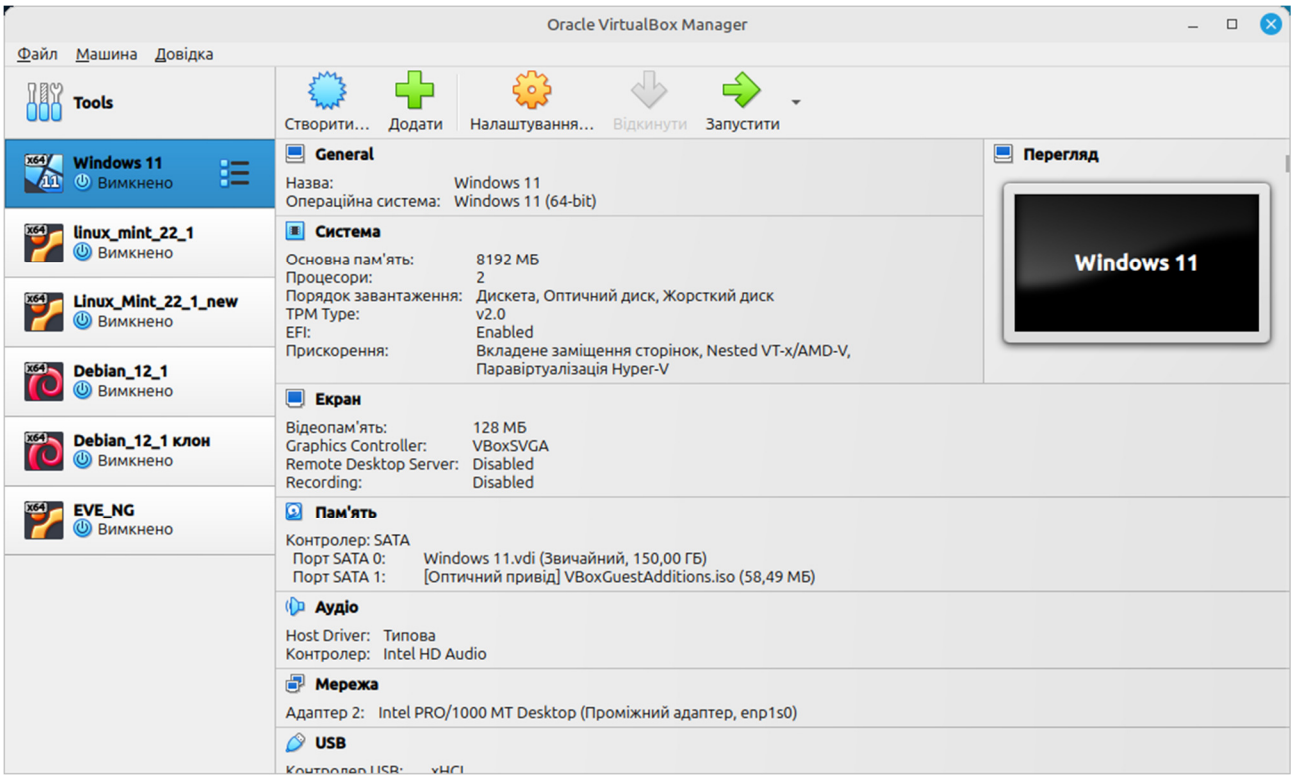

Рисунок 1.3 – Вікно менеджера VM VirtualBox

3.2.2 У вкладці "Name and Operation system" ввести назву (довільними, але краще латинськими символами), тип та версію операційної системи згідно рисунку. Далі відкрити вкладку "Hardware".

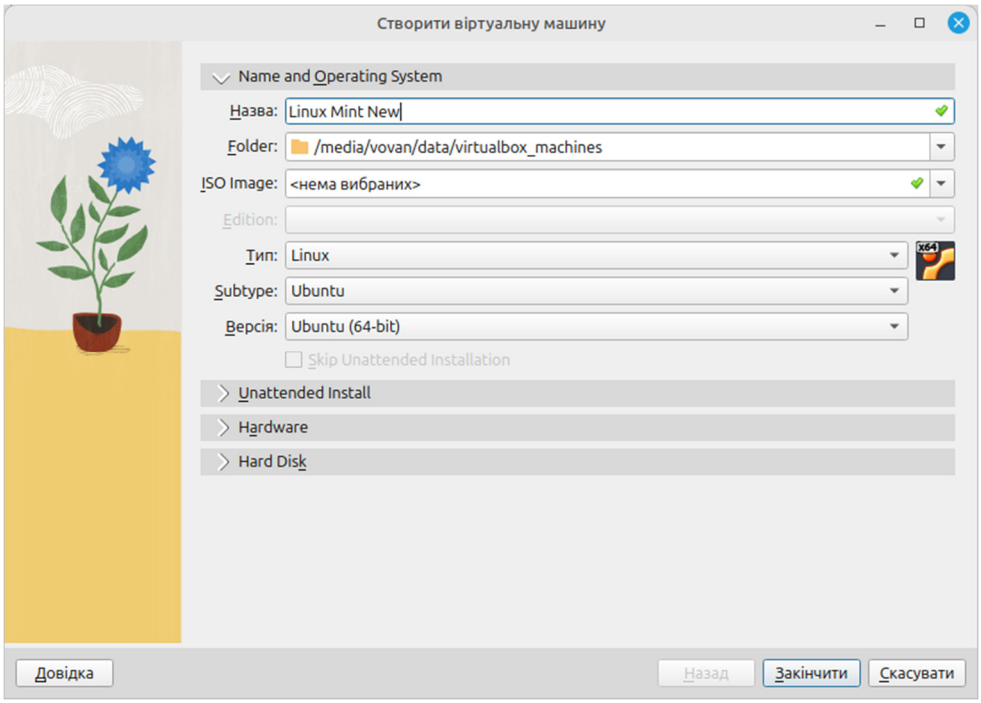

Рисунок 1.4 – Вікно вводу назви та типу ОС

3.2.3 Обрати об’єм оперативної пам'яті (2-4 Гб буде достатньо) та кількість ядер процесора (1), що буде відведена під роботу ВМ.

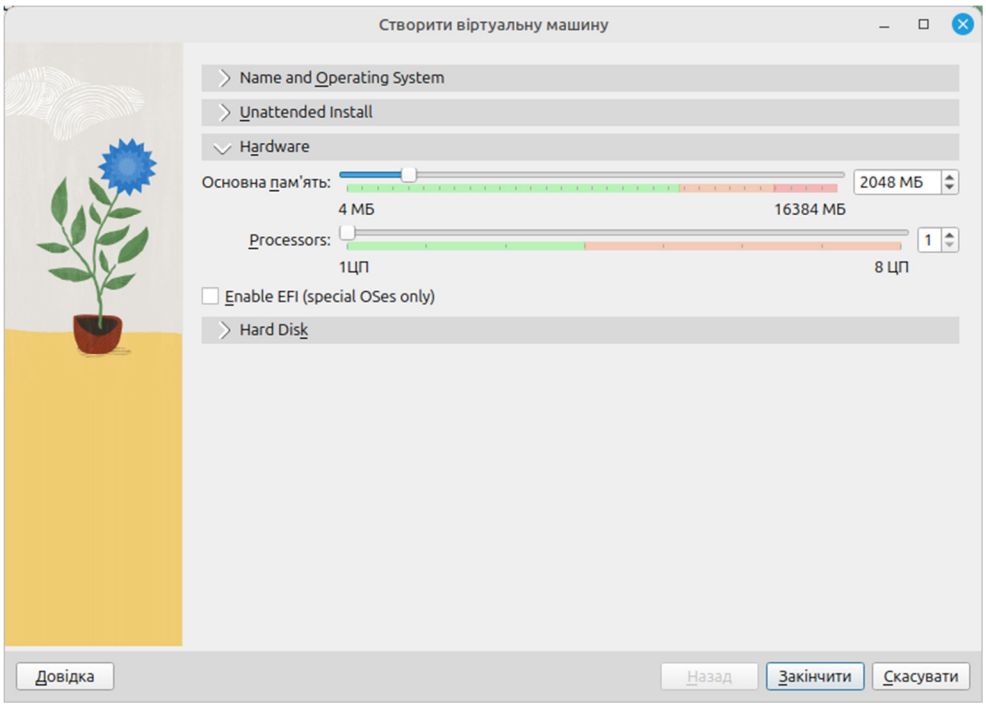

Рисунок 1.5 – Налаштування обсягу пам'яті та конфігурації ЦП

3.2.4 У вкладці "Hard Disk" обрати опцію створення нового віртуального жорсткого диска та вказати його об’єм (20 Гб).

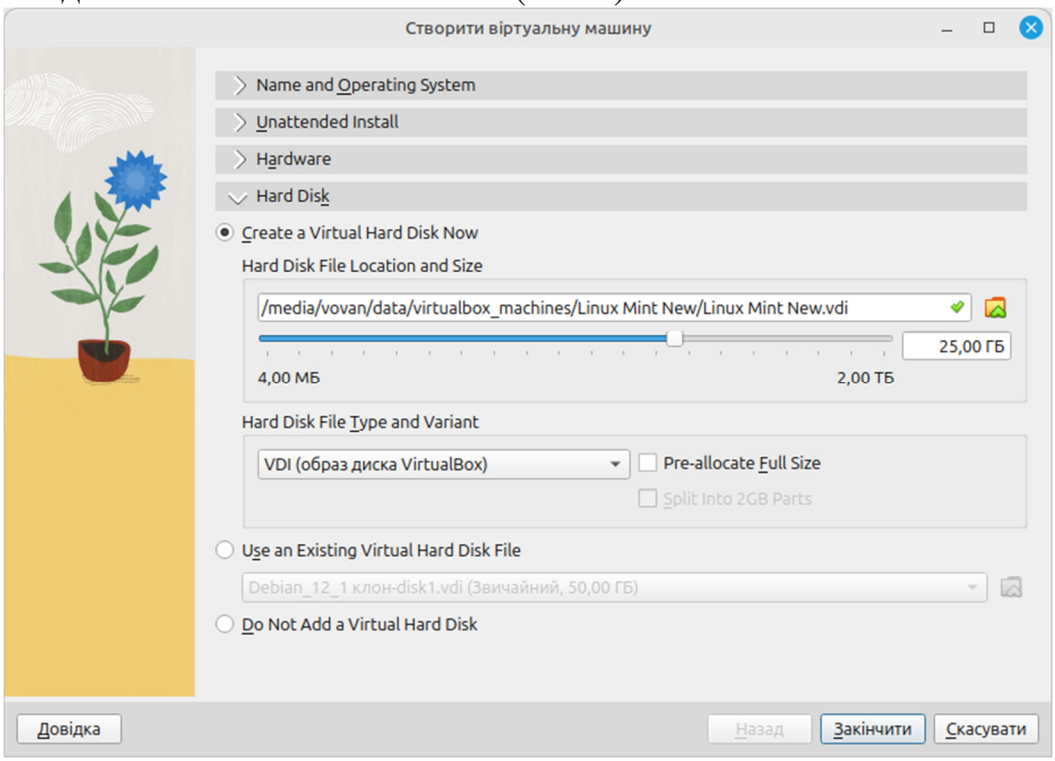

Рисунок 1.6 – Налаштування віртуального накопичувача

3.2.5 Натиснути "Закінчити" для завершення процесу створення ВМ.

3.3 Інсталяція ОС Linux Mint 21.2 "Victoria"

3.3.1 Перейти на сторінку https://linuxmint.com/download.php, натиснути Download в розділі MATE Edition.

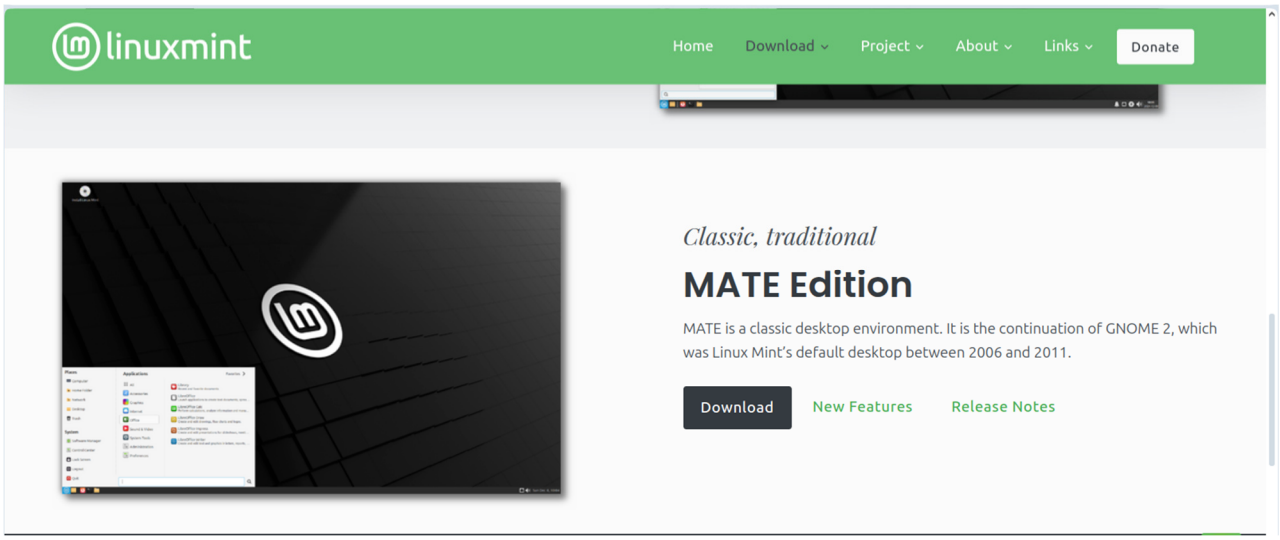

Рисунок 1.7 – Сторінка завантаження образу гостьової ОС

3.3.2 Завантажити ISO файл з ОС можна через мережу Bittorrent або з географічно розподіленої файлової мережі (розділ Download mirrors). Зберегти файл на локальному диску.

3.3.3 В менеджері Oracle VM VirtualBox виділити створену ВМ і натиснути кнопку "Налаштувати".

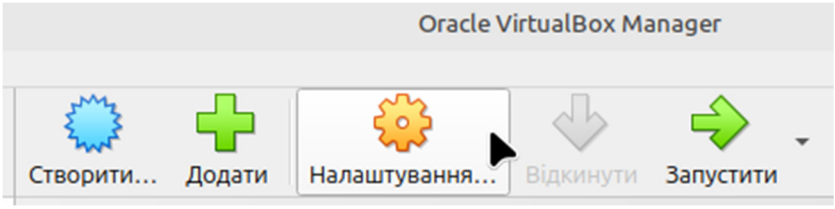

Рисунок 1.8 – Вибір налаштування гостьової ОС

3.3.4 В розділі "Пам'ять" обрати віртуальний контролер оптичного приводу, натиснути на виділену на рисунку піктограму, вибрати пункт "Вибрати файл диску" та обрати файл завантаженого раніше ISO образу диска.

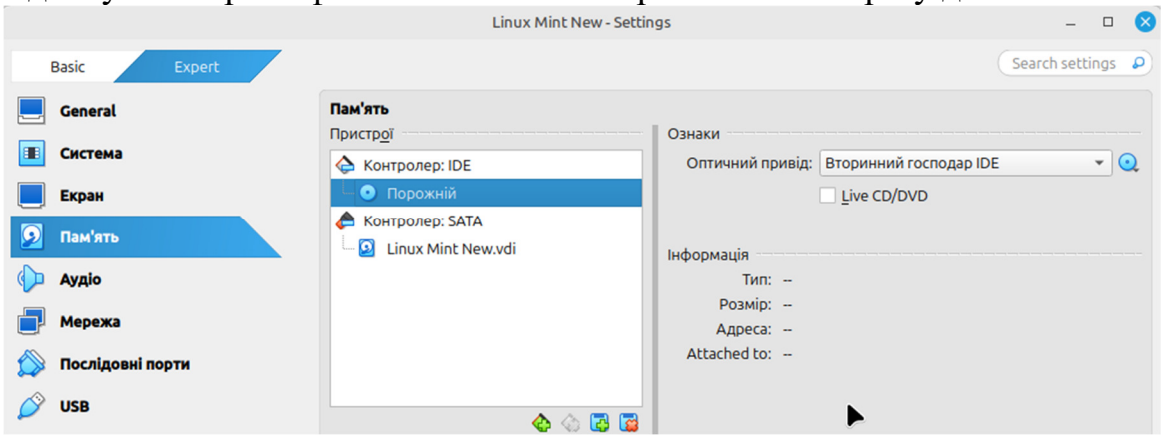

Рисунок 1.9 – Підключення ISO образу диску з гостьовою ОС

3.3.5 Натиснути "ОК".

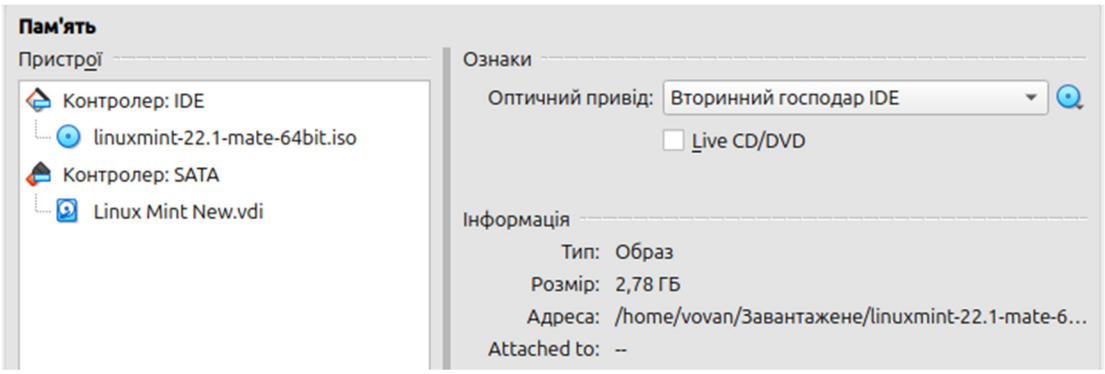

Рисунок 1.10 – Вигляд вікна з підключеним ISO образом ОС

3.3.6 У вкладці "Система - процесор" увімкнути підтримку апаратної віртуалізації VT-x/AMD-V.

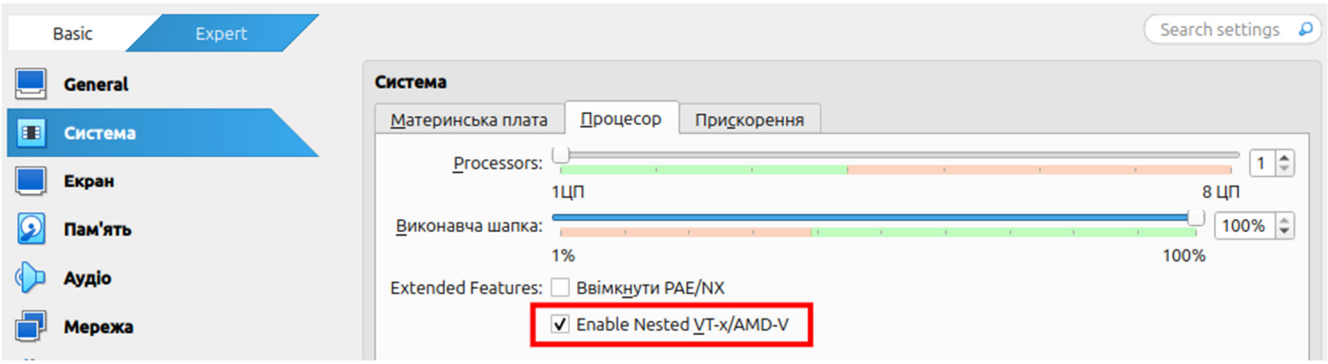

Рисунок 1.11 – Увімкнення віртуалізації

3.3.7 Перейти в менеджер ВМ і натиснути кнопку "Запустити" на панелі інструментів.

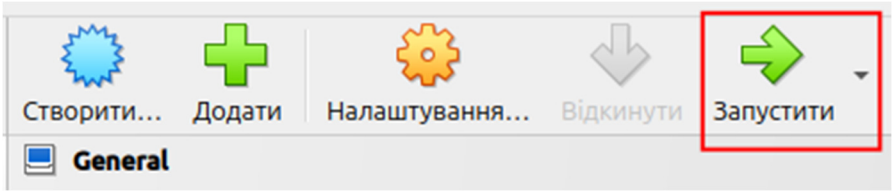

Рисунок 1.12 – Запуск гостьової ОС

3.3.8 Після завантаження ОС Linux з віртуального оптичного приводу, клікнути на іконці "Install Linux Mint".

3.3.9 Обрати бажану мову інтерфейсу для встановлення ОС, розкладку клавіатури (можна залишити за замовчуванням). Додаткові мультимедіа кодеки можна не встановлювати для виконання подальших практичних завдань в рамках циклу даної дисципліни.

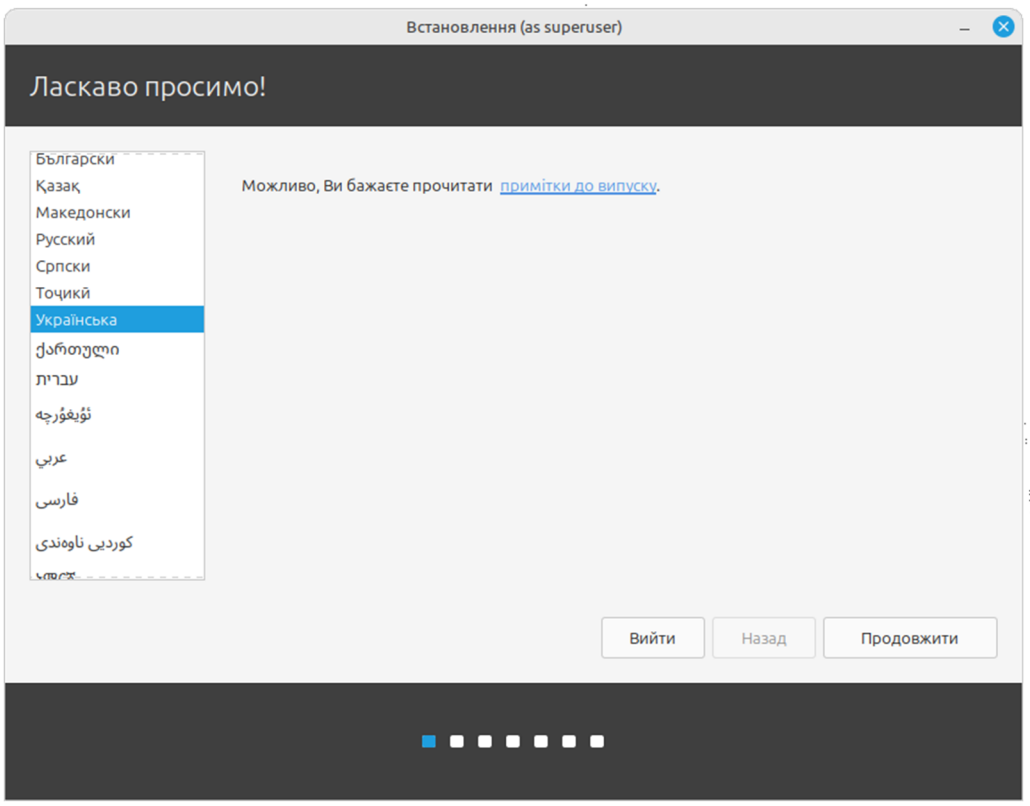

Рисунок 1.13 – Вибір мови

3.3.10 Обрати пункт меню "Стерти диск та встановити зараз" та натиснути кнопку "Встановити зараз".

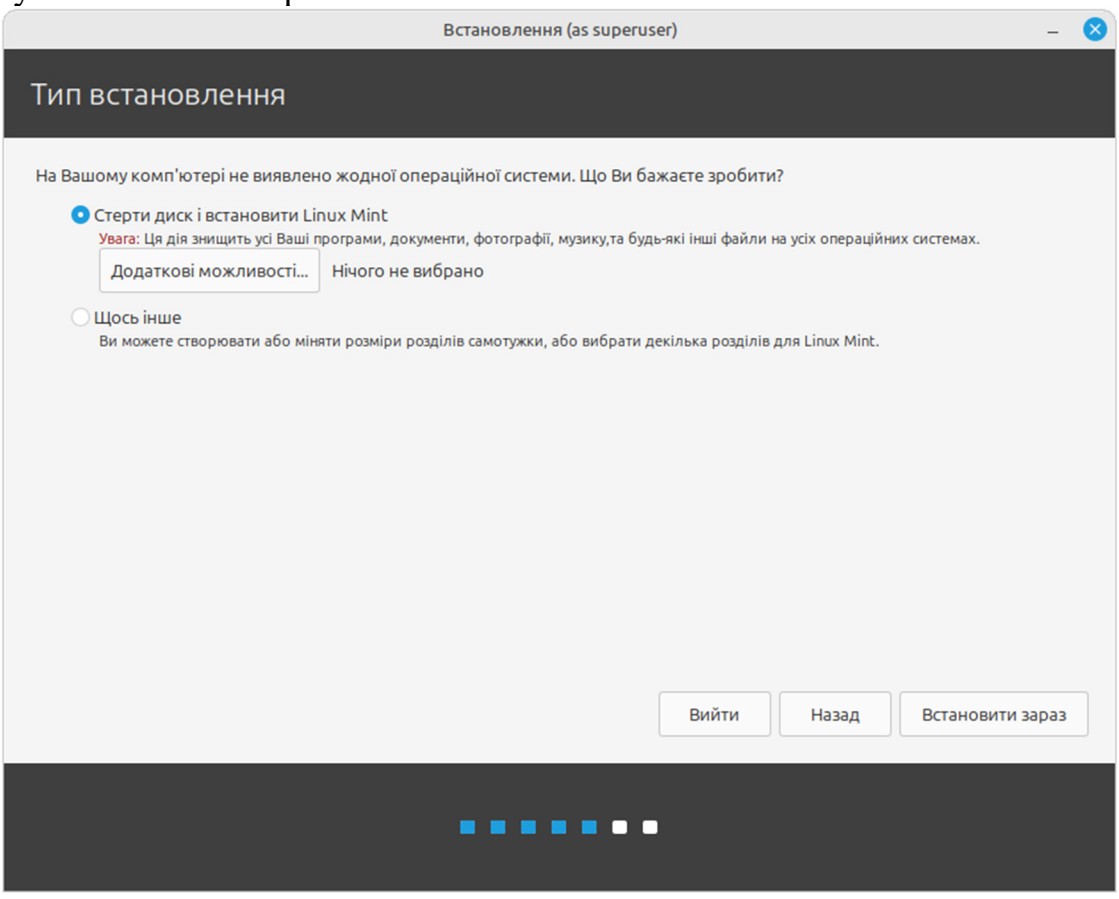

Рисунок 1.14 – Опції вибору типу встановлення ОС

3.3.11 Натиснути кнопку "Продовжити" для запису змін на віртуальний диск.

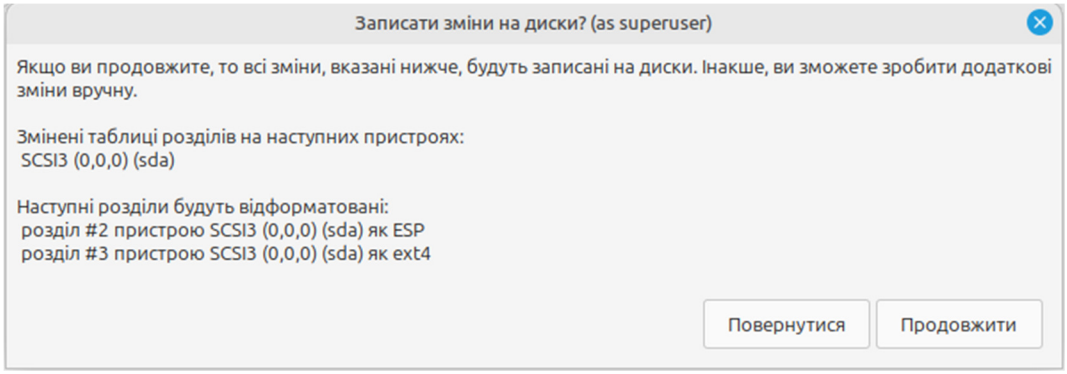

Рисунок 1.15 – Конфігурація розділів на віртуальному накопичувачі

3.3.12 Обрати часовий пояс (Київ).

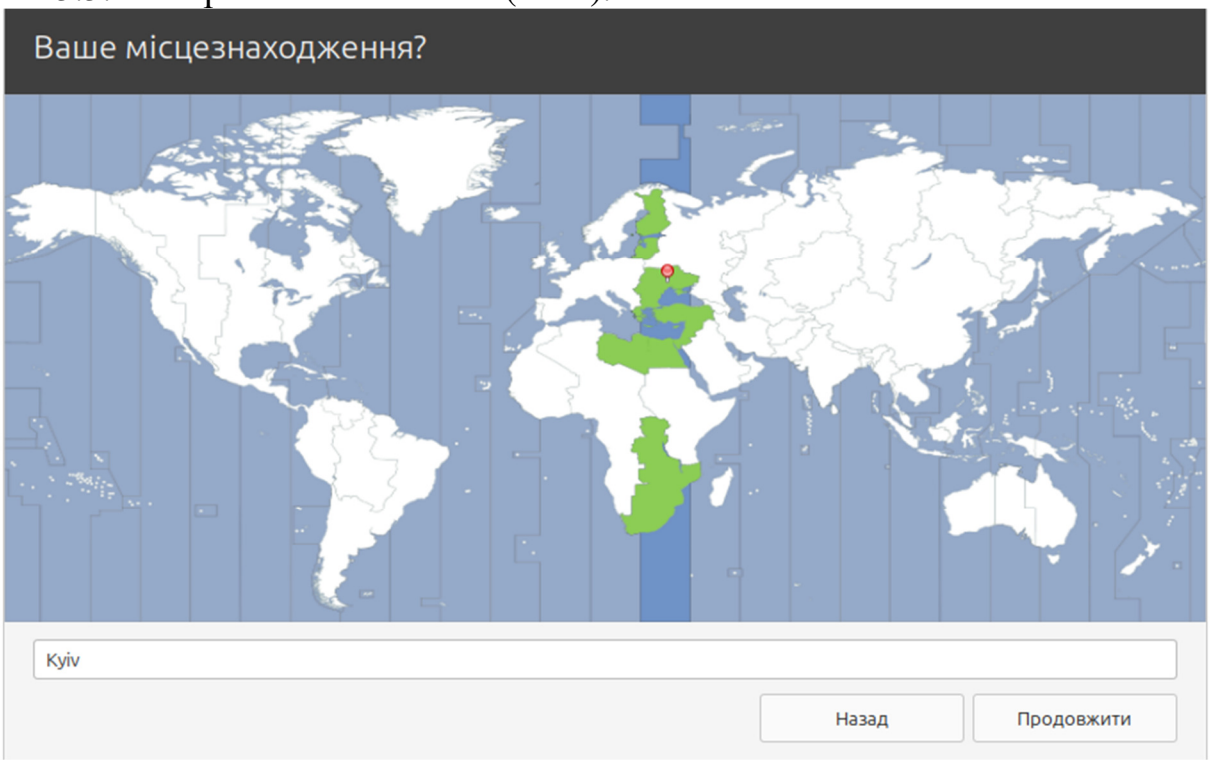

Рисунок 1.16 – Вибір часового поясу ОС

3.3.13 Ввести логін та пароль користувача. Вибрати пункт для автоматичного входу в систему.

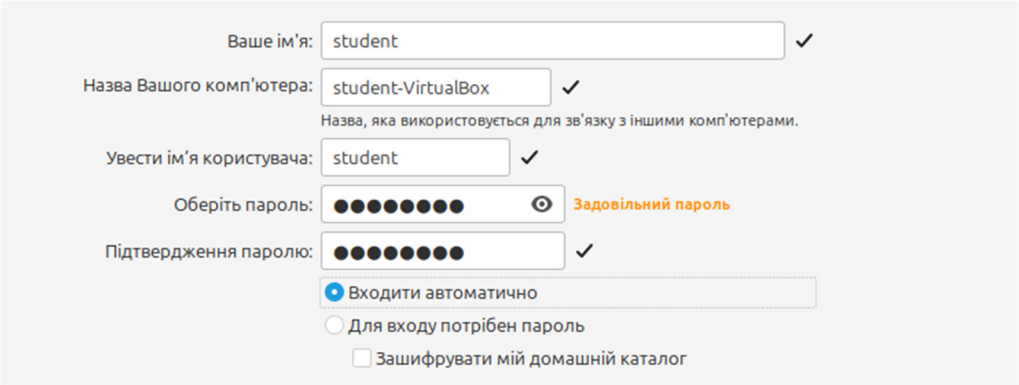

Рисунок 1.17 – Логін та пароль користувача

3.3.14 Дочекатися копіювання файлів. Натиснути "Перезавантажити" для запуску системи з жорсткого диску.

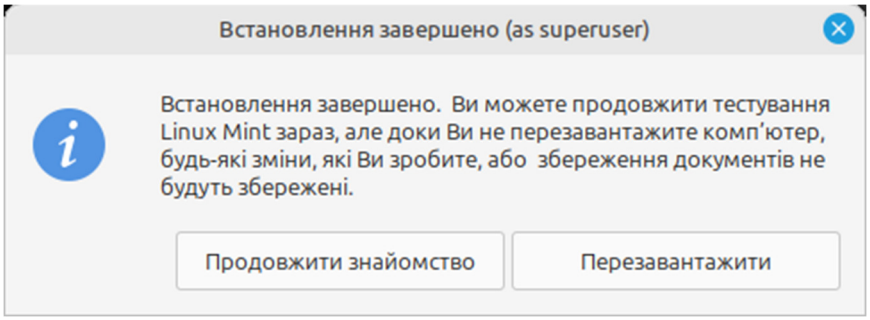

Рисунок 1.18 – Вікно завершення встановлення ОС

3.4 Встановлення доповнень гостьової системи

3.4.1 В розділі "Пристрої" головного меню поточної віртуальної машини обрати пункт "Встановити гостьові доповнення".

3.4.2 Після автоматичного монтування образу віртуального оптичного диску, виділити файл "autorun.sh" і за допомогою контекстного меню запустити його від імені адміністратора. Дочекатися встановлення пакету програм і перезавантажити ВМ через меню "Пуск".

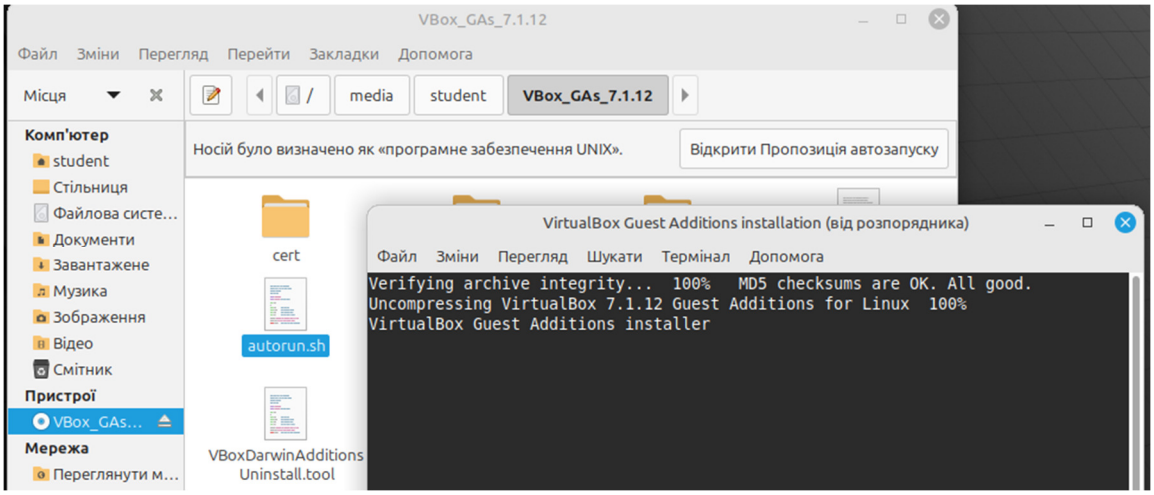

Рисунок 1.19 – Вікно встановлення доповнень гостьової ОС

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам'яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов'язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються

способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 2

Тема: Основи роботи з командним рядком Linux

Мета роботи: Ознайомитися з базовими командами терміналу Linux, навчитися виконувати навігацію по файловій системі, керувати файлами та каталогами, використовувати довідку та історію команд.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

Термінал у операційній системі Linux є потужним інструментом командного рядка, який дозволяє користувачу взаємодіяти з системою через текстові команди. На відміну від графічного інтерфейсу, термінал надає прямий доступ до функціональності операційної системи без використання миші та графічних елементів. Використання командного рядка є фундаментальною навичкою для системних адміністраторів, розробників та досвідчених користувачів Linux, оскільки багато системних операцій можуть бути виконані швидше та ефективніше саме через термінал. Крім того, деякі функції операційної системи доступні виключно через командний рядок, що робить знання терміналу необхідною умовою для повноцінної роботи з Linux-системами.

Файлова система Linux організована у вигляді ієрархічної структури, що починається з кореневого каталогу "/". На відміну від Windows, в Linux всі диски, розділи та пристрої інтегруються в єдину файлову систему через процес монтування. Кореневий каталог містить стандартні системні каталоги, кожен з яких має своє специфічне призначення та функції в системі. Домашній каталог користувача позначається символом "~" і є місцем, де зберігаються особисті файли та налаштування кожного користувача. Поточний робочий каталог можна визначити за допомогою команди pwd (print working directory), яка відображає повний шлях до каталогу, в якому користувач знаходиться в даний момент.

Структура стандартних каталогів в Linux регламентується стандартом Filesystem Hierarchy Standard (FHS) та представлена в таблиці 2.1. Розуміння призначення цих каталогів є критично важливим для ефективної роботи з системою та розуміння логіки організації файлів.

Домашні каталоги користувачів розташовані в /home і мають структуру /home/username, де username - це ім'я користувача в системі. Кожен користувач має повні права на свій домашній каталог і може створювати, модифікувати та видаляти файли в ньому без обмежень. Домашній каталог містить особисті файли користувача, конфігураційні файли програм (зазвичай приховані файли, що починаються з крапки), та підкаталоги для організації особистої інформації. Винятком є користувач root, домашній каталог якого розташований в /root, а не в /home/root, що підкреслює його особливий статус системного адміністратора.

Таблиця 2.1 - Стандартні каталоги файлової системи Linux

Каталог	Призначення	Приклади вмісту
/	Кореневий каталог, основа всієї файлової системи	Містить всі інші каталоги
/bin	Основні виконувані файли системи	ls, cp, mv, cat, bash
/sbin	Системні виконувані файли для адміністратора	mount, fsck, iptables
/usr	Програми та файли користувачів	/usr/bin, /usr/local, /usr/share
/var	Змінні дані системи	логи, черги печатки, кеші
/etc	Файли конфігурації системи	passwd, fstab, hosts
/home	Домашні каталоги користувачів	/home/username
/root	Домашній каталог адміністратора root	налаштування root
/tmp	Тимчасові файли	тимчасові дані програм
/opt	Додаткове програмне забезпечення	сторонні програми
/dev	Файли пристроїв	/dev/sda, /dev/tty
/proc	Віртуальна файлова система процесів	інформація про процеси
/sys	Інформація про апаратне забезпечення	системна інформація
/lib	Системні бібліотеки	.so файли, модулі ядра
/boot	Файли завантаження системи	ядро Linux, завантажувач
/media	Точки монтування змінних носіїв	USB, CD/DVD
/mnt	Тимчасові точки монтування	ручне монтування пристроїв

Навігація по файловій системі здійснюється за допомогою команди cd (change directory), яка дозволяє переміщуватися між каталогами. Команда cd без параметрів повертає користувача до домашнього каталогу, cd ~ має той самий ефект, а cd .. переміщує на один рівень вище в ієрархії каталогів. Для переходу до кореневого каталогу використовується cd /, а cd - повертає до попереднього робочого каталогу.

Перегляд вмісту каталогів здійснюється командою ls (list), яка має численні опції для різних режимів відображення. Команда ls -l виводить детальну інформацію про файли, включаючи дозволи, власника, розмір та дату модифікації, тоді як ls -a показує всі файли, включаючи приховані (які починаються з крапки). Комбінування опцій можливе за допомогою ls -la або ls -al для одночасного відображення детальної інформації та прихованих файлів. Команда ls -R виконує рекурсивний перегляд, показуючи вміст усіх підкаталогів.

Основні команди для роботи з файлами та каталогами в Linux наведені в таблиці 2.2. Ці команди забезпечують повний спектр операцій для управління файловою системою через термінал.

Створення файлів та каталогів є основними операціями в роботі з файловою системою. Команда mkdir створює новий каталог, а touch створює порожній файл або оновлює час модифікації існуючого файлу. Для запису тексту в файл використовується команда echo в поєднанні з операторами перенаправлення ">" (перезапис) та ">>" (додавання до кінця файлу). Наприклад, echo "Hello World" > file.txt створить файл з текстом, а echo "New line" >> file.txt додасть новий рядок до існуючого файлу.

Таблиця 2.2 - Основні команди Linux для роботи з файловою системою

Категорія	Команда	Опис	Приклад використання
Навігація	pwd	Показує поточний каталог	pwd
	cd	Зміна каталогу	cd /home/user
	cd ~	Перехід до домашнього каталогу	cd ~
	cd ..	Перехід на рівень вище	cd ..
	cd -	Перехід до попереднього каталогу	cd -
Перегляд	ls	Список файлів та каталогів	ls
	ls -l	Детальний список	ls -l
	ls -a	Показати приховані файли	ls -a
	ls -R	Рекурсивний перегляд	ls -R
	cat	Вивід вмісту файлу	cat file.txt
Створення	mkdir	Створення каталогу	mkdir newdir
	touch	Створення порожнього файлу	touch newfile.txt
	echo	Вивід тексту	echo "text" > file.txt
Копіювання	cp	Копіювання файлів	cp file1.txt file2.txt
	cp -r	Рекурсивне копіювання каталогів	cp -r dir1 dir2
Переміщення	mv	Переміщення/перейменування	mv oldname newname
Видалення	rm	Видалення файлів	rm file.txt
	rm -r	Рекурсивне видалення каталогів	rm -r directory
	rmdir	Видалення порожнього каталогу	rmdir emptydir
Допоміжні	man	Довідка по командах	man ls
	history	Історія команд	history
	clear	Очищення екрану	clear

Копіювання файлів здійснюється командою cp, яка приймає два аргументи: джерело та призначення. Для копіювання каталогів необхідно використовувати опцію -r (рекурсивно), яка забезпечує копіювання всіх файлів та підкаталогів. Команда mv виконує функції як переміщення, так і перейменування файлів та каталогів, залежно від того, чи змінюється шлях до файлу.

Видалення файлів виконується командою rm, яка безповоротно видаляє зазначені файли. Для видалення каталогів разом з їх вмістом використовується rm -r, що виконує рекурсивне видалення. Команда rmdir призначена виключно для видалення порожніх каталогів та буде видавати помилку, якщо каталог містить файли чи підкаталоги. При роботі з командою rm слід бути особливо обережним, оскільки видалені файли не потрапляють до кошика і не можуть бути легко відновлені.

Перенаправлення потоків введення-виведення є важливою концепцією в Linux-терміналі, що дозволяє керувати тим, куди направляється вивід команд та звідки команди отримують вхідні дані. Оператор ">" перенаправляє стандартний вивід команди в файл, повністю перезаписуючи його вміст, тоді як ">>" додає вивід до кінця існуючого файлу. Команда cat може використовуватися не лише для перегляду файлів, а й для об'єднання декількох файлів за допомогою конструкції cat file1 file2 > combined_file.

Система довідки в Linux надає вичерпну інформацію про команди та їх параметри через команду man (manual). Введення man followed by назвою

команди відкриває детальну документацію з описом функціональності, доступних опцій та прикладів використання. Альтернативним способом отримання короткої довідки є використання опції --help з більшістю команд, наприклад ls --help. Ця інформація є критично важливою для освоєння нових команд та розуміння всіх можливостей уже знайомих інструментів.

Історія команд автоматично зберігається системою і може бути переглянута за допомогою команди history, що виводить пронумерований список раніше виконаних команд. Навігація по історії здійснюється за допомогою клавіш зі стрілками: стрілка вгору повертає попередні команди, а стрілка вниз - наступні. Це значно прискорює роботу, дозволяючи швидко повторювати або модифікувати раніше виконані команди без необхідності їх повторного введення.

Автозавершення імен файлів та команд за допомогою клавіші Tab є потужним інструментом для підвищення продуктивності роботи в терміналі. Натискання Tab після введення частини назви файлу, каталогу або команди автоматично доповнює назву, якщо існує єдиний варіант, або показує список можливих варіантів при подвійному натисканні Tab. Це не лише прискорює введення команд, але й значно зменшує кількість помилок через друкарські помилки в назвах файлів.

Управління відображенням терміналу включає команду clear, яка очищає екран терміналу, роблячи його більш зручним для читання. Хоча ця команда не є обов'язковою для виконання системних завдань, вона значно покращує читабельність та організованість роботи в терміналі. Для завершення роботи з терміналом можна використовувати команду exit або комбінацію клавіш Ctrl+D, що закриває поточний сеанс терміналу.

Робота з прихованими файлами має свої особливості в Linux-системах. Файли та каталоги, назви яких починаються з крапки (наприклад, .config, .bashrc), вважаються прихованими і не відображаються при звичайному виконанні команди ls. Для їх перегляду необхідно використовувати опцію -a, що показує всі файли, включаючи приховані. Ці файли зазвичай містять конфігураційні дані програм та системні налаштування користувача.

Ефективна робота в терміналі вимагає розуміння принципів організації файлової системи Linux та систематичної практики використання основних команд. Комбінування різних команд та опцій дозволяє виконувати складні операції з файлами та каталогами, що є основою для подальшого вивчення більш просунутих аспектів адміністрування Linux-систем. Важливо пам'ятати, що операції видалення в терміналі є незворотними, тому слід завжди перевіряти правильність команд перед їх виконанням.

2 КЛЮЧОВІ ПИТАННЯ

1. Що таке термінал і чому він важливий у Linux?
2. Які основні команди використовуються для навігації по файловій системі?
3. Як відрізняється команда ls від ls -a або ls -l?

4. Що робить команда cd ~ або cd ..?
5. Як створити, скопіювати, перемістити або видалити файл через командний рядок?
6. Як використовувати man, щоб дізнатися про можливості команди?
7. Що таке потоки вводу-виводу у терміналі?
8. Як переглянути історію виконаних команд?
9. Що робить команда clear і чи є вона обов'язковою?
10. Як вийти з терміналу або завершити сеанс?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Відкрийте термінал у середовищі Linux (наприклад, у віртуальній машині з Ubuntu або іншої дистрибутиві). Усі наступні дії виконуються виключно в домашньому каталозі користувача (~). Не виходьте за межі цього каталогу під час виконання завдання.

3.2 За допомогою команд pwd і ls переконайтеся, що ви знаходитеся в домашньому каталозі, і перегляньте його вміст. Використовуйте команду ls -a, щоб побачити приховані файли, і ls -l — для детального виведення.

3.3 Виконайте послідовність команд, яка відповідає вашому варіанту з наведеної нижче таблиці. Кожен варіант передбачає створення певної структури каталогів і файлів, роботу з вмістом файлів (використовуючи echo, cat, >, >>), операції копіювання (cp), переміщення (mv) та видалення (rm), а також навігацію (cd, ..).

3.4 Під час виконання завдання використовуйте автозавершення імен за допомогою клавіші Tab, щоб уникнути помилок і прискорити роботу. Для повторного виклику попередніх команд використовуйте стрілки вгору/вниз або команду history.

3.5 Для отримання довідки про команди використовуйте man <команда> або <команда> --help. Наприклад, man cp або ls --help. Ознайомтеся з основними опціями команд, які використовуєте.

3.6 Після виконання усіх дій вашої варіанти, виконайте команду ls -R, щоб вивести повну структуру створених файлів і каталогів, і зробіть скріншот цього виводу для включення до звіту.

3.7 У звіті (протоколі) вкажіть тему, мету роботи, поясніть кожен крок, наведіть виконані команди та їх результати, додайте скріншоти. У висновках порівняйте роботу через командний рядок із графічним інтерфейсом, відмітьте, які команди виявилися найбільш зручними, і чому знання терміналу важливе для роботи з Linux.

Таблиця 2.3 – Варіанти індивідуальних завдань

№	Завдання
1	Створіть каталог lab2. У цьому каталозі створіть два підкаталоги: docs та temp. У каталозі docs створіть файл file1.txt, записавши до нього текст "Hello, Linux!" за допомогою команди echo. Скопіюйте цей файл до каталогу temp, назвавши копію backup.txt. Перегляньте вміст обох файлів за допомогою команди cat.

2	Створіть каталог project. У ньому створіть два підкаталоги: src та bin. У каталозі src створіть файл main.c і запишіть до нього текст #include <stdio.h> за допомогою echo. Перемістіть файл main.c до каталогу bin і перейменуйте його на main.o. Після цього видаліть порожній каталог src.
3	Створіть каталог practice. У ньому створіть два підкаталоги: dir1 та dir2. У каталозі dir1 створіть файл test.txt і запишіть до нього текст “Test data”. Скопіюйте весь каталог dir1 (включаючи вміст) до dir2, щоб у dir2 з’явився підкаталог dir1. Перевірте структуру за допомогою ls -R.
4	Створіть каталог temp. У ньому створіть файл data.txt і запишіть до нього текст “Linux OS”. Додайте до цього файлу новий рядок: “Terminal is powerful”, використовуючи перенаправлення >>. Перевірте фінальний вміст файлу командою cat.
5	Створіть каталог myfiles. У ньому створіть два порожніх файлів: fileA та fileB за допомогою команди touch. Об’єднайте їх вміст (хоча воно порожнє) у новий файл combined.txt, використовуючи cat fileA fileB > combined.txt. Після цього видаліть файли fileA та fileB.
6	Створіть каталог work. У ньому створіть файл notes.txt. За допомогою команди echo і перенаправлення >> додайте до файлу три різні рядки тексту. Перегляньте вміст файлу. Потім видаліть увесь каталог work разом із усім вмістом за допомогою rm -r.
7	Створіть каталог archive. У ньому створіть файл old.txt і запишіть до нього текст “Backup content”. Скопіюйте цей файл до домашнього каталогу, назвавши копію current.txt. Після цього видаліть файл old.txt. Перейменуйте каталог archive на backup.
8	Створіть каталог hidden. У ньому створіть прихований файл .config (з крапкою на початку) і запишіть до нього текст “Hidden settings”. Перевірте, чи відображається файл, виконавши ls -a у каталозі hidden. Потім видаліть увесь каталог hidden.
9	Створіть каталог data. У ньому створіть файл input.txt і запишіть до нього текст “Test”. За допомогою команди cat і перенаправлення > скопіюйте вміст input.txt до нового файлу output.txt. Перевірте, що обидва файли існують і містять однаковий текст. Видаліть обидва файли.
10	Створіть каталог scripts. У ньому створіть файл hello.txt. За допомогою man echo дізнайтеся, як вивести текст у файл. Використовуючи echo і >, запишіть до hello.txt рядок “Script started”. Перегляньте вміст файлу.
11	Створіть два каталоги: dir1 та dir2. У каталозі dir1 створіть файл log.txt і запишіть до нього текст “Error log”. Перемістіть цей файл до каталогу dir2. Потім перейменуйте каталог dir2 на logs.
12	Створіть каталог files. У ньому створіть два файли: a.txt та b.txt. За допомогою echo запишіть у кожен унікальний текст. Об’єднайте їх вміст у новий файл all.txt (використовуючи cat a.txt b.txt > all.txt). Після цього видаліть файли a.txt та b.txt.
13	Створіть каталог test_dir. У ньому створіть три порожніх файлів: f1, f2, f3 (за допомогою touch). Виведіть список файлів з деталізацією за допомогою ls -l. Потім видаліть файл f2.
14	Створіть каталог backup. У ньому створіть файл config.txt і запишіть до нього текст “Default settings”. Скопіюйте цей файл до домашнього каталогу, назвавши його settings.txt.
15	Створіть каталог project. У ньому створіть файл readme.txt. Виконайте команду cd .., щоб перейти на один рівень вище (у домашній каталог). Потім знову увійдіть у project. Перевірте поточний шлях за допомогою pwd.

16	Створіть каталог docs. У ньому створіть файл info.txt. Скопіюйте увесь каталог docs до нового каталогу docs_backup. Потім у оригінальному каталозі docs створіть новий файл note.txt. Порівняйте вміст обох каталогів.
17	Створіть каталог temp_files. У ньому створіть два файли: temp1.txt та temp2.txt. Створіть новий каталог archive. Перемістіть обидва файли з temp_files до archive. Після цього видаліть порожній каталог temp_files.
18	Створіть каталог long_name_test. У ньому створіть файл з довгим ім'ям: very_long_filename_for_testing.txt. Спробуйте використати клавішу Tab, щоб автоматично доповнити ім'я файлу під час виконання команд cat або rm.
19	Створіть каталог data. У ньому створіть файл log.txt. За допомогою echo і >> додайте до файлу два рядки тексту. Перегляньте фінальний вміст файлу за допомогою cat.
20	Створіть каталог copy_test. У ньому створіть файл source.txt і запишіть до нього будь-який текст. Скопіюйте цей файл у той самий каталог під назвою dest.txt. Змініть текст у dest.txt, переконавшись, що source.txt залишився без змін.
21	Створіть каталог move_test. У ньому створіть файл file.txt. Перемістіть цей файл на один рівень вище (у домашній каталог) і перейменуйте його на moved_file.txt.
22	Створіть каталог dirA. У ньому створіть файл data.log і запишіть до нього будь-який текст. Створіть новий каталог dirB. Перемістіть файл data.log до dirB. Потім видаліть порожній каталог dirA.
23	Створіть каталог practice, у ньому — level1, а в level1 — level2. Перейдіть у каталог level2. Виконайте команду cd ../.., щоб повернутися у домашній каталог. Перевірте, що ви дійсно у домашньому каталозі, за допомогою pwd.
24	Створіть каталог temp. У ньому створіть файл data.txt і запишіть до нього початковий текст. За допомогою cat data.txt > copy.txt створіть копію. Потім додайте новий рядок до data.txt і використайте cat data.txt >> copy.txt, щоб додати його вміст у кінець copy.txt. Перевірте результат.
25	Створіть каталог my_project. У ньому створіть файл readme.txt. За допомогою man touch дізнайтеся, як встановити час модифікації файлу. Виконайте touch -d "1 day ago" readme.txt, щоб змінити дату. Перевірте результат за допомогою ls -l.
26	Створіть каталог test. У ньому створіть два файли: file1 та file2. Перейменуйте file1 на document.txt. Потім скопіюйте вміст document.txt у файл file2 (використовуючи cat document.txt > file2). Перевірте вміст file2.
27	Створіть каталог archive. У ньому створіть два файли: backup1.txt та backup2.txt. Видаліть кожен файл окремо командою rm. Потім створіть ще два файли (temp1.txt, temp2.txt) і видаліть їх разом за допомогою rm *.
28	Створіть каталог logs. У ньому створіть файл app.log і запишіть до нього рядок "Start". Додайте ще кілька рядків через echo і >>. Перегляньте фінальний вміст файлу.
29	Створіть каталог config. У ньому створіть прихований файл .env (з крапкою) і запишіть до нього текст "DEBUG=true". Виконайте ls -a, щоб переконатися, що файл відображається. Потім скопіюйте .env до домашнього каталогу.
30	Створіть каталог final. У ньому створіть файл output.txt. Запишіть у нього перший рядок: echo "Line 1" > output.txt. Потім додайте другий рядок: echo "Line 2" >> output.txt. Перевірте, що файл містить обидва рядки.

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам'яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов'язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються

способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 3

Тема: Управління обліковими записами користувачів в Linux

Мета роботи: Навчитися створювати, редагувати та видаляти облікові записи користувачів і груп, розуміти структуру системних файлів, що відповідають за автентифікацію.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

Операційні системи сімейства Linux побудовані на принципах багатокористувацького доступу, де кожен користувач має власний простір для роботи та чітко визначені права доступу до системних ресурсів. Ця архітектура забезпечує безпеку та ізоляцію між різними користувачами, дозволяючи одночасну роботу декількох осіб на одній системі без взаємного втручання. Система розрізнення користувачів базується на унікальних ідентифікаторах та групах, що дозволяє гнучко налаштовувати права доступу до файлів, каталогів та системних ресурсів.

Сучасне системне адміністрування Linux вимагає від фахівців глибокого розуміння механізмів управління користувачами та групами. Ці знання є фундаментальними для забезпечення безпеки системи, організації командної роботи та підтримки належного розподілу ресурсів. Правильне управління обліковими записами дозволяє створювати ефективні робочі середовища, де кожен користувач має доступ лише до тих ресурсів, які необхідні для виконання його завдань.

Управління обліковими записами користувачів є однією з найважливіших задач системного адміністрування в операційних системах сімейства Linux. Кожен користувач системи має унікальний ідентифікатор (UID - User ID), який використовується ядром для ідентифікації та контролю доступу до ресурсів. Системні користувачі зазвичай мають UID менше 1000, тоді як звичайні користувачі отримують UID починаючи з 1000. Аналогічно, кожна група в системі має свій унікальний ідентифікатор групи (GID - Group ID), який дозволяє організувати колективний доступ до файлів та каталогів. Первинна група користувача визначається при створенні облікового запису, але користувач може бути додатково включений до декількох вторинних груп для розширення своїх прав доступу.

Інформація про користувачів зберігається в декількох ключових системних файлах, структура яких визначає функціональність системи автентифікації. Файл /etc/passwd містить основні дані про користувачів: ім'я користувача, UID, GID первинної групи, домашній каталог, оболонку за замовчуванням та додаткову інформацію в полі коментарів. Кожен рядок файлу представляє окремого користувача, а поля розділені двокрапками. З міркувань безпеки, хеші паролів зберігаються в окремому файлі /etc/shadow, доступ до якого має лише суперкористувач. Цей файл також містить інформацію про терміни дії паролів, дати останньої зміни та інші параметри безпеки облікового запису.

Для ефективного управління користувачами та групами в Linux використовується широкий спектр команд, які дозволяють виконувати всі необхідні операції від створення до видалення облікових записів. Як показано в таблиці 3.1, ці команди можна розділити на кілька категорій за функціональним призначенням. Кожна команда має свої специфічні опції та параметри, які дозволяють точно налаштовувати параметри користувачів відповідно до потреб конкретного середовища. Розуміння синтаксису та можливостей цих команд є ключовим для успішного виконання завдань системного адміністрування.

Таблиця 3.1 - Основні команди для управління користувачами та групами в Linux

Категорія	Команда	Опції	Призначення
Створення користувачів	`adduser`	`[ім'я]`	Інтерактивне створення користувача з налаштуваннями за замовчуванням. Приклад: `sudo adduser kovalenko` - створює користувача kovalenko з автоматичним створенням домашнього каталогу /home/kovalenko
	`useradd`	`-m -d [каталог] -s [оболонка] [ім'я]`	Створення користувача з вказівкою параметрів. Приклад: `sudo useradd -m -d /home/temp_user -s /bin/bash temp_user` - створює користувача з кастомним домашнім каталогом
Модифікація користувачів	`usermod`	`-d [новий_каталог] -m [ім'я]`	Зміна домашнього каталогу з переміщенням файлів. Приклад: `sudo usermod -d /home/temp_kovalenko -m kovalenko` - змінює домашній каталог та переносить всі файли
	`usermod`	`-s [оболонка] [ім'я]`	Зміна оболонки користувача. Приклад: `sudo usermod -s /bin/sh kovalenko` - змінює оболонку на sh замість bash
	`usermod`	`-c "[коментар]" [ім'я]`	Додавання коментаря (ПІБ). Приклад: `sudo usermod -c "Ivan Kovalenko" kovalenko` - додає повне ім'я користувача
	`usermod`	`-e [YYYY-MM-DD] [ім'я]`	Встановлення терміну дії облікового запису. Приклад: `sudo usermod -e 2025-04-10 kovalenko` - обліковий запис стане недоступним після зазначеної дати
	`usermod`	`-g [група] [ім'я]`	Зміна первинної групи. Приклад: `sudo usermod -g interns kovalenko` - робить групу interns первинною для користувача
	`usermod`	`-aG [група1,група2] [ім'я]`	Додавання до додаткових груп. Приклад: `sudo usermod -aG developers,testers kovalenko` -

			додає користувача до груп developers та testers
	usermod	-l [нове_ім'я] [старе_ім'я]	Перейменування користувача. Приклад: sudo usermod -l user_temp kovalenko - змінює ім'я користувача на user_temp
	usermod	-L [ім'я]	Блокування облікового запису. Приклад: sudo usermod -L kovalenko - блокує можливість входу в систему
	usermod	-U [ім'я]	Розблокування облікового запису. Приклад: sudo usermod -U kovalenko - відновлює можливість входу
Видалення користувачів	userdel	[ім'я]	Видалення користувача без домашнього каталогу. Приклад: sudo userdel kovalenko - видаляє користувача, але залишає /home/kovalenko
	userdel	-r [ім'я]	Видалення користувача разом із домашнім каталогом. Приклад: sudo userdel -r kovalenko - повністю видаляє користувача та всі його файли
Управління групами	groupadd	[назва_групи]	Створення нової групи. Приклад: sudo groupadd developers - створює групу developers з автоматично призначеним GID
	groupmod	-n [нова_назва] [стара_назва]	Перейменування групи. Приклад: sudo groupmod -n dev_team developers - змінює назву групи developers на dev_team
	groupdel	[назва_групи]	Видалення групи. Приклад: sudo groupdel developers - видаляє групу, якщо вона не є первинною для жодного користувача
Перевірка та перегляд	id	[ім'я]	Відображення UID, GID та членства в групах. Приклад: id kovalenko - показує uid=1001(kovalenko) gid=1001(kovalenko) groups=1001(kovalenko),1002(developers)
	getent passwd	[ім'я]	Перегляд запису користувача в /etc/passwd. Приклад: getent passwd kovalenko - виводить kovalenko:x:1001:1001:Ivan Kovalenko:/home/kovalenko:/bin/bash
	getent group	[назва_групи]	Перегляд складу групи. Приклад: getent group developers - показує developers:x:1002:kovalenko,petrov
	getent shadow	[ім'я]	Перегляд інформації з /etc/shadow. Приклад: sudo getent shadow kovalenko - показує хеш пароля та параметри безпеки

Управління паролями	`passwd`	`[ім'я]`	Зміна пароля користувача. Приклад: `sudo passwd kovalenko` - запитує новий пароль та зберігає його хеш у /etc/shadow

Створення нових користувачів може здійснюватися двома основними способами за допомогою команд adduser та useradd, як детально описано в таблиці 3.1. Команда adduser є більш зручною для інтерактивного використання, оскільки автоматично створює домашній каталог, копіює файли за замовчуванням та запитує пароль користувача. Команда useradd є більш низькорівневою та зазвичай використовується в скриптах, вимагаючи явного вказування всіх необхідних параметрів. При створенні користувача система автоматично призначає наступний доступний UID та створює первинну групу з таким же іменем, якщо не вказано інше.

Модифікація параметрів існуючих користувачів здійснюється за допомогою команди usermod, яка дозволяє змінювати практично всі атрибути облікового запису. За допомогою цієї команди можна змінити домашній каталог користувача (опція -d), оболонку за замовчуванням (опція -s), додати коментар або ПІБ (опція -c), встановити термін дії облікового запису (опція -e) та змінити первинну групу (опція -g). Для додавання користувача до додаткових груп використовується опція -aG, яка зберігає існуюче членство в групах та додає нові. Важливо пам'ятати, що при зміні домашнього каталогу потрібно також використовувати опцію -m для переміщення файлів із старого каталогу.

Групи в Linux-системах створюються та управляються за допомогою команд groupadd, groupmod та groupdel, які представлені в таблиці 3.1. Ці команди дозволяють створювати нові групи, змінювати їх параметри та видаляти непотрібні групи. Кожен користувач обов'язково належить до однієї первинної групи, яка використовується за замовчуванням при створенні файлів. Додатково користувач може бути членом декількох вторинних груп, що розширює його права доступу до ресурсів системи. Інформація про групи зберігається у файлі /etc/group, де кожен рядок містить назву групи, GID та список користувачів-членів групи.

Перевірка результатів виконаних операцій є критично важливим етапом адміністрування користувачів, який дозволяє переконатися в коректності внесених змін. Команда id показує поточний UID користувача, GID його первинної групи та список всіх груп, до яких він належить. Команди getent passwd та getent group дозволяють переглядати записи в системних файлах без безпосереднього доступу до них, що є більш безпечним підходом. Для перевірки налаштувань безпеки та термінів дії паролів використовується команда getent shadow, яка відображає інформацію з файлу /etc/shadow. Ці команди особливо корисні в середовищах з централізованою автентифікацією, де дані про користувачів можуть зберігатися в LDAP або інших службах каталогів.

Видалення користувачів та груп потребує особливої уваги до деталей для забезпечення повного очищення системи від непотрібних записів. Команда userdel за замовчуванням видаляє лише запис користувача з системних файлів, залишаючи домашній каталог та файли недоторканими. Використання опції -r

забезпечує повне видалення користувача разом із домашнім каталогом та поштовою скринькою. При видаленні важливо пам'ятати про файли, що належать користувачеві в інших частинах файлової системи, які можуть залишитися без власника. Групи видаляються командою groupdel, але система не дозволить видалити групу, якщо вона є первинною для якого-небудь користувача.

Безпека управління обліковими записами вимагає використання привілейованих прав для виконання адміністративних операцій. Всі команди управління користувачами та групами повинні виконуватися з правами суперкористувача, що досягається за допомогою команди sudo. Це забезпечує належний контроль доступу та ведення журналу виконаних операцій. При роботі з терміном дії облікових записів слід враховувати, що заблоковані або прострочені облікові записи можуть створювати проблеми безпеки, оскільки файли та процеси користувача можуть залишатися активними. Регулярний аудит облікових записів та очищення непотрібних користувачів є важливою частиною підтримки безпеки системи.

Практичне застосування команд управління користувачами вимагає розуміння взаємозв'язків між різними компонентами системи автентифікації. При створенні користувача система автоматично створює домашній каталог з відповідними правами доступу та копіює в нього шаблони конфігураційних файлів із каталогу /etc/skel. Зміна параметрів користувача може вимагати додаткових дій, таких як переміщення файлів при зміні домашнього каталогу або оновлення прав доступу при зміні груп. Важливо також враховувати, що деякі операції, такі як перейменування користувача, можуть вплинути на працюючі процеси та збережені дані, тому їх слід виконувати з особливою обережністю в робочому середовищі.

2 КЛЮЧОВІ ПИТАННЯ

1. Що таке UID (User ID) і GID (Group ID), і яку роль вони відіграють у системі безпеки Linux?
2. Які основні поля містить файл /etc/passwd, і яке призначення кожного з них?
3. Чому паролі користувачів зберігаються у файлі /etc/shadow, а не у /etc/passwd? Які переваги це дає з точки зору безпеки?
4. У чому різниця між командами adduser та useradd? Яка з них є більш зручною для інтерактивного використання і чому?
5. Як створити нового користувача, вказавши для нього кастомний домашній каталог та оболонку за замовчуванням?
6. Як додати існуючого користувача до додаткової групи? Яка опція команди usermod забезпечує збереження попереднього членства в групах?
7. Що таке первинна (основна) група користувача, і чим вона відрізняється від вторинних (додаткових) груп?
8. Як змінити домашній каталог користувача та перенести всі його файли до нового місця? Які опції команди usermod для цього використовуються?

9. Як перевірити, до яких груп належить користувач, і яку інформацію виводить команда id?

10. Як повністю видалити користувача разом із його домашнім каталогом та поштовою скринькою? Чим відрізняється виклик userdel з опцією -r від виклику без неї?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Лабораторна робота виконується в терміналі операційної системи Linux (наприклад, у віртуальній машині з дистрибутивом Ubuntu або іншим). Усі операції, пов’язані з управлінням користувачами та групами, вимагають прав суперкористувача. Для їх виконання обов’язково використовуйте команду sudo.

3.2 Кожен студент виконує одне індивідуальне завдання, відповідно до свого номера у списку групи (від 1 до 30). Усі дії виконуються згідно з описом у таблиці варіантів, наведеної нижче. У завданнях використовується ім’я користувача, утворене шляхом транслітерації прізвища студента латиницею (наприклад, *Коваленко* → kovalenko, *Бондаренко* → bondarenko).

3.3 У процесі виконання завдання необхідно виконати послідовність команд, що передбачає: створення облікового запису користувача за допомогою adduser; створення однієї або кількох груп за допомогою groupadd; додавання користувача до груп за допомогою usermod -aG; зміну параметрів користувача (наприклад, домашній каталог, строк дії, коментар, оболонка); перевірку результатів за допомогою команд id, getent passwd, getent group; видалення користувача та/або груп згідно з вимогами варіанту.

3.4 Після виконання кожної операції необхідно перевіряти її результат. Наприклад, після створення користувача — переконатися, що запис з’явився в /etc/passwd і створено домашній каталог; після додавання до групи — перевірити склад групи через getent group; після видалення — переконатися, що каталоги та записи в системних файлах видалено.

3.5 Усі виконані дії мають бути зафіксовані в протоколі лабораторної роботи. Обов’язково додайте скріншоти ключових етапів: вивід команди id, вміст файлів /etc/passwd та /etc/group, структура домашнього каталогу, результат видалення користувача. Кожен скріншот супроводжується коротким поясненням.

Таблиця 3.2 – Варіанти індивідуальних завдань

№	ЗАВДАННЯ
1	Створіть користувача за своїм прізвищем (наприклад, ivanov) за допомогою adduser. Перевірте запис у /etc/passwd через getent passwd. Додайте користувача до групи developers. Перевірте через id.
2	Створіть користувача за своїм прізвищем. Створіть групу testers за допомогою groupadd. Додайте користувача до цієї групи. Перевірте склад групи через getent group testers.
3	Створіть користувача за своїм прізвищем. Змініть його домашній каталог на /home/temp_user за допомогою usermod -d. Переконайтеся, що каталог створено, і права встановлено.

4	Створіть користувача за своїм прізвищем. Встановіть для нього строк дії облікового запису на наступний тиждень (наприклад, usermod -e 2025-04-10 [ім’я]). Перевірте через getent shadow.
5	Створіть користувача за своїм прізвищем. Створіть групу admins. Додайте користувача до групи admins як додаткової. Перевірте через id [ім’я].
6	Створіть користувача за своїм прізвищем. Перейменуйте обліковий запис на user_temp за допомогою usermod -l. Перевірте, чи зберігся домашній каталог.
7	Створіть користувача за своїм прізвищем. Змініть оболонку на /bin/sh за допомогою usermod -s. Перевірте через getent passwd.
8	Створіть користувача за своїм прізвищем. Заблокуйте обліковий запис командою usermod -L. Спробуйте увійти. Розблокуйте через usermod -U.
9	Створіть користувача за своїм прізвищем. Створіть групу support. Додайте користувача до груп support і developers. Перевірте список груп через id.
10	Створіть користувача за своїм прізвищем. Видаліть користувача за допомогою userdel без опції -r. Перевірте, чи залишився домашній каталог. Потім видаліть його вручну.
11	Створіть користувача за своїм прізвищем. Видаліть користувача разом із домашнім каталогом за допомогою userdel -r. Перевірте, що каталог видалено.
12	Створіть користувача за своїм прізвищем. Створіть групу interns. Зробіть цю групу первинною для користувача через usermod -g. Перевірте через id.
13	Створіть користувача за своїм прізвищем. Додайте йому коментар (ПІБ) через usermod -c “Petro Ivanov”. Перевірте через getent passwd.
14	Створіть користувача за своїм прізвищем. Створіть групу devops. Додайте користувача до групи devops. Перевірте, чи він є у списку getent group devops.
15	Створіть користувача за своїм прізвищем. Змініть пароль за допомогою passwd [ім’я]. Перевірте, чи змінився хеш у /etc/shadow.
16	Створіть користувача за своїм прізвищем. Створіть групу qa. Додайте користувача до групи qa. Перевірте, чи він є у списку getent group qa.
17	Створіть користувача за своїм прізвищем. Створіть групу trainees. Зробіть її первинною та додайте користувача до групи support як додаткової. Перевірте через id.
18	Створіть користувача за своїм прізвищем. Створіть групу staff. Додайте користувача до груп staff та developers. Перевірте права доступу до файлу, створеного від імені користувача.
19	Створіть користувача за своїм прізвищем. Створіть групу consultants. Додайте користувача до групи. Перевірте, чи він може читати файл із правами g+r.
20	Створіть користувача за своїм прізвищем. Створіть групу freelancers. Додайте користувача до групи. Перевірте, чи він з’явився в /etc/group.
21	Створіть користувача за своїм прізвищем. Змініть ім’я користувача на temp_user та домашній каталог на /home/temp_user. Перевірте через getent passwd.
22	Створіть користувача за своїм прізвищем. Встановіть для облікового запису автоматичне вимикання через 7 днів (usermod -e). Перевірте через getent shadow.
23	Створіть користувача за своїм прізвищем. Створіть групу analysts. Додайте користувача до групи. Перевірте, чи може він створити файл у каталозі з правами g+w.
24	Створіть користувача за своїм прізвищем. Створіть групу junior. Додайте користувача до групи junior та developers. Видаліть користувача разом із домашнім каталогом. Переконайтеся, що всі сліди видалено.
25	Створіть користувача за своїм прізвищем. Створіть групу support_team. Додайте користувача до цієї групи. Перевірте склад групи через getent group support_team.

26	Створіть користувача за своїм прізвищем. Змініть його домашній каталог на /home/temp_[ім'я]. Переконайтеся, що новий каталог створено, а старий видалено.
27	Створіть користувача за своїм прізвищем. Створіть групу test_group. Додайте користувача до групи. Перевірте через id.
28	Створіть користувача за своїм прізвищем. Створіть групу project_team. Додайте користувача до групи project_team та developers. Перевірте список груп.
29	Створіть користувача за своїм прізвищем. Змініть оболонку на /bin/bash (якщо вона була змінена раніше). Перевірте через getent passwd.
30	Створіть користувача за своїм прізвищем. Видаліть користувача разом із домашнім каталогом. Потім видаліть усі групи, які були створені для цього користувача. Перевірте, що записів у /etc/group немає.

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам'яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та

виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов’язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 4

Тема: Права доступу та безпека файлів у Linux

Мета роботи: Опанувати команди для встановлення прав доступу, зміни власників і груп у файловій системі Linux, а також оцінити безпеку встановлених налаштувань.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

Файлова система Linux забезпечує багаторівневу модель безпеки, що базується на системі прав доступу, яка регулює взаємодію користувачів із файлами та каталогами. Ця модель ґрунтується на принципі розмежування доступу між трьома категоріями користувачів: власником файлу (owner), групою користувачів (group) та всіма іншими користувачами системи (others). Кожна з цих категорій може мати різний рівень доступу до конкретного файлу або каталогу, що дозволяє гнучко налаштовувати безпеку та забезпечувати принцип найменших привілеїв. Система прав доступу є основоположним механізмом забезпечення конфіденційності, цілісності та доступності даних у багатокористувацьких середовищах Unix-подібних операційних систем.

Для кожної категорії користувачів визначаються три типи прав доступу, які позначаються символами r (read), w (write) та x (execute). Право читання дозволяє переглядати вміст файлу або отримувати список файлів у каталозі. Право запису надає можливість модифікувати вміст файлу або створювати, видаляти та перейменовувати файли в каталозі. Право виконання для файлів дозволяє запускати їх як програми або скрипти, а для каталогів — здійснювати перехід до них та доступ до вкладених елементів.

Права доступу можуть бути представлені у двох форматах: символьному та числовому (восьмеричному). У символьному форматі права записуються у вигляді дев'ятисимвольного рядка, де перші три символи відповідають правам власника, наступні три — правам групи, останні три — правам інших користувачів. Кожна тріада може містити символи r, w, x або дефіс (-), який означає відсутність відповідного права. Числовий формат представляє права у вигляді трицифрового числа, де кожна цифра є сумою значень прав для відповідної категорії: r=4, w=2, x=1. Наприклад, права rwxr-xr-- відповідають числовому представленню 754.

Таблиця 4.1 містить основні команди для адміністрування прав доступу та власності файлів у Linux, згруповані за функціональним призначенням.

Команда ls -l є основним інструментом для перегляду прав доступу та демонструє інформацію у форматі, де перший символ вказує на тип об'єкта (- для файлу, d для каталогу), а наступні дев'ять символів показують права доступу для власника, групи та інших користувачів. Додатково команда відображає кількість посилань, ім'я власника, назву групи, розмір файлу, дату останньої модифікації та ім'я файлу. Ця інформація є критично важливою для аналізу безпеки файлової системи та виявлення потенційних проблем з правами доступу.

Таблиця 4.1 - Основні команди для роботи з правами доступу в Linux

Група команд	Команда	Опис	Приклад використання
Перегляд прав та власності	`ls -l`	Детальний список файлів з правами	`ls -l file.txt`
	`stat`	Повна інформація про файл	`stat file.txt`
	`id`	Інформація про поточного користувача	`id username`
Зміна прав доступу	`chmod`	Зміна прав доступу (числовий формат)	`chmod 755 file.txt`
	`chmod`	Зміна прав доступу (символьний формат)	`chmod u+x,g-w file.txt`
	`chmod`	Зміна прав для всіх категорій	`chmod a+r file.txt`
Зміна власності	`chown`	Зміна власника файлу	`chown user file.txt`
	`chown`	Зміна власника та групи	`chown user:group file.txt`
	`chown`	Зміна тільки групи	`chown :group file.txt`
	`chgrp`	Зміна групи файлу	`chgrp group file.txt`
Управління групами	`groupadd`	Створення нової групи	`sudo groupadd team`
	`usermod`	Додавання користувача до групи	`usermod -aG group user`
	`groups`	Перегляд груп користувача	`groups username`
Налаштування umask	`umask`	Встановлення маски прав	`umask 027`
	`umask`	Перегляд поточної маски	`umask`
Пошук файлів	`find`	Пошук за правами доступу	`find ~ -perm 777`
	`find`	Пошук виконуваних файлів	`find ~ -perm -u+x`
Створення файлів та каталогів	`touch`	Створення файлу	`touch file.txt`
	`mkdir`	Створення каталогу	`mkdir directory`

Команда chmod підтримує два основні режими роботи: встановлення абсолютних прав за допомогою числового формату та відносну зміну прав за допомогою символьного формату. При використанні символьного формату можна вказувати категорії користувачів (u - власник, g - група, o - інші, a - всі), операцію (+, -, =) та права (r, w, x). Наприклад, команда chmod go-r file забирає право читання у групи та інших користувачів, а команда chmod u=rw,g=r,o= встановлює точні права для кожної категорії.

Механізм umask (user mask) визначає права доступу за замовчуванням для новостворених файлів та каталогів шляхом вказівки прав, які повинні бути виключені з повного набору прав. Для файлів максимальні права становлять 666 (rw-rw-rw-), а для каталогів — 777 (rwxrwxrwx). Значення umask віднімається від цих максимальних прав для отримання фактичних прав нових об'єктів.

Наприклад, при umask 027 новий файл матиме права 640 (666-027=640), а новий каталог — права 750 (777-027=750). Це забезпечує автоматичне застосування політики безпеки при створенні файлів.

Рисунок 4.1 демонструє структуру прав доступу в Linux та взаємозв'язок між символьним та числовим представленням.

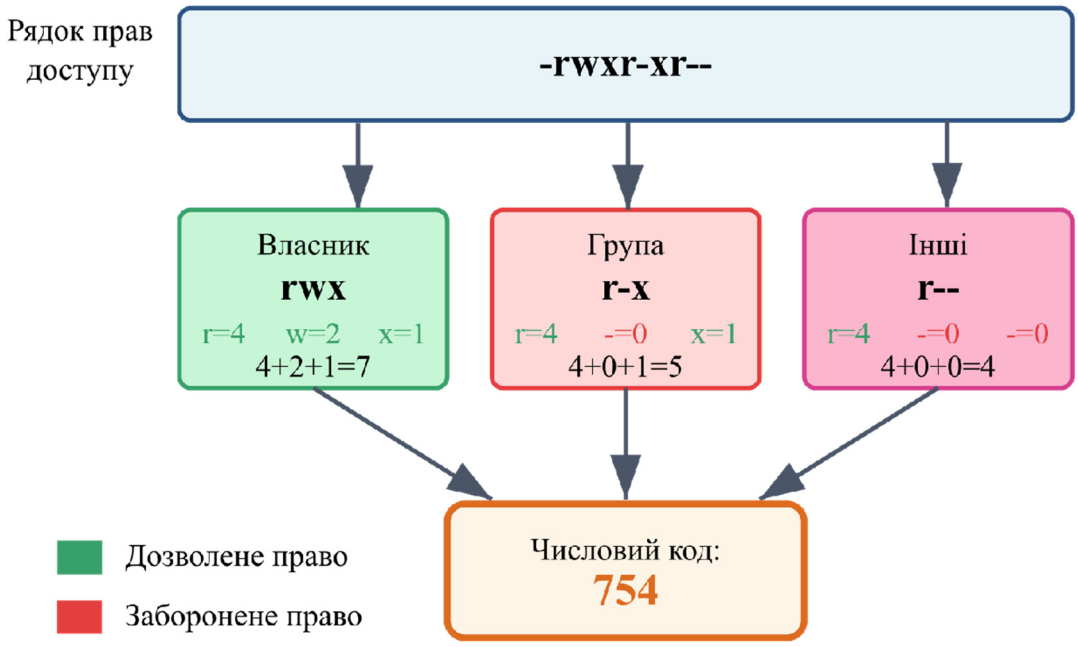

Рисунок 4.1 – Структура прав доступу в Linux

На рисунку зображена схема, що показує декомпозицію рядка прав доступу "-rwxr-xr--" на окремі компоненти. У верхній частині схеми розташований повний рядок прав, під яким знаходяться три блоки, що відповідають правам власника (rwx), групи (r-x) та інших користувачів (r--). Кожен блок містить три позиції для прав читання, запису та виконання. Під кожним блоком вказані відповідні числові значення: 4+2+1=7 для власника, 4+0+1=5 для групи, 4+0+0=4 для інших користувачів. У нижній частині схеми показано результуючий числовий код 754. Додатково на рисунку присутні кольорові індикатори: зелений для дозволених прав, червоний для заборонених, що наочно демонструє різницю між наявними та відсутніми правами доступу.

Безпека файлової системи значною мірою залежить від правильного налаштування прав доступу та дотримання принципу найменших привілеїв. Надмірні права доступу, особливо право виконання для файлів даних або повні права доступу (777) для всіх користувачів, створюють серйозні ризики безпеки. Файли з правами 777 дозволяють будь-якому користувачу системи читати, модифікувати та виконувати їх, що може призвести до несанкціонованого доступу до даних, модифікації критичних файлів або запуску шкідливого коду. Домашні каталоги користувачів повинні мати права 700 або 750, що забезпечує приватність особистих даних та запобігає несанкціонованому доступу інших користувачів.

Команда find є потужним інструментом для аудиту безпеки файлової системи, що дозволяє знаходити файли з певними правами доступу. Використання опції -perm дозволяє шукати файли з точними правами (наприклад, -perm 777) або файли, які мають принаймні вказані права (наприклад, -perm -u+x для пошуку всіх файлів, які власник може виконувати). Регулярний аудит файлової системи за допомогою таких команд допомагає виявляти потенційні вразливості безпеки та забезпечувати відповідність політикам безпеки організації. Особливо важливо контролювати наявність файлів з надмірними правами доступу в системних каталогах та домашніх директоріях користувачів.

2 КЛЮЧОВІ ПИТАННЯ

1. Які три категорії прав доступу в Linux?
2. Що означають символи r, w, x для файлів і каталогів?
3. Як переглянути права файлу за допомогою ls -l?
4. Як змінити власника файлу за допомогою chown?
5. Як встановити права 755 на файл?
6. Що робить команда chmod go-r file?
7. Як umask впливає на права нових файлів?
8. Чому важливо не давати зайвих прав, особливо на виконання?
9. Які права повинні бути у домашнього каталогу користувача?
10. Як виявити файли з небезпечними правами (наприклад, 777)?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Створіть у своєму домашньому каталозі окремий робочий каталог для виконання лабораторної роботи. Усі подальші дії виконуються виключно в межах цього каталогу, щоб забезпечити безпечне та контрольоване середовище для роботи з правами доступу.

3.2 Згідно з вашим індивідуальним варіантом із таблиці варіантів виконайте послідовність дій, яка передбачає створення файлів і каталогів, налаштування для них прав доступу, зміну власників та груп, а також аналіз безпечності встановлених параметрів.

3.3 Перед тим як змінювати права або призначати нового власника, обов’язково створіть потрібні файли або каталоги. Використовуйте відповідні команди для встановлення прав у числовому та символьному форматах, дотримуючись вказівок вашого варіанта.

3.4 У разі необхідності створіть додаткову групу та додайте до неї потрібних користувачів, щоб перевірити механізм спільного доступу. Виконайте зміну групи або власника для вказаних об’єктів, щоб проілюструвати роботу моделі власник–група–інші.

Таблиця 4.2 – Варіанти індивідуальних завдань

№	Завдання
1	Створіть файл file1.txt. Надайте права: власник — rwx, група — r-x, інші — — (chmod 750). Змініть власника на свого користувача, групу — на свою основну групу.
2	Створіть каталог dir1. Встановіть права 750. Змініть власника на свого користувача. Перевірте, чи може інший користувач (за умовою) увійти в каталог.
3	Встановіть umask 027. Створіть файл test.txt і каталог test_dir. Перевірте їхні права за допомогою ls -l. Поясніть, чому вони такі.
4	Створіть файл script.sh. Надайте власнику права rwx, групі — r-x, іншим — відсутні (chmod 750). Зробіть файл виконуваним.
5	Створіть файл data.log. Встановіть права 644, використовуючи символьний формат: chmod go-wx data.log. Перевірте результат.
6	Створіть файл shared.txt. Надайте права 640. Поясніть, хто може читати/змінювати файл. Переконайтеся, що “інші” не мають доступу.
7	Створіть групу team (sudo groupadd team). Додайте свого користувача до групи (usermod -aG team $USER). Створіть файл note.txt із правами 660. Перевірте, чи група має доступ.
8	Створіть файл danger.txt. Встановіть права 777. Поясніть, чому це небезпечно. Знайдіть його командою find ~ -type f -name “danger.txt” -perm 777.
9	Створіть файл secret.conf. Встановіть права 600. Переконайтеся, що інші користувачі не можуть його читати (за умовою).
10	Створіть файл config.ini. Використайте chmod o-r config.ini, щоб забрати право читання від “інших”. Перевірте зміни через ls -l.
11	Створіть каталог shared. Надайте права 775. Встановіть власника на свого користувача, групу — на team (або іншу існуючу). Перевірте доступ для групи.
12	Встановіть umask 002. Створіть файл group_file.txt. Які права він матиме? Поясніть, чому (очікується 664).
13	Створіть файл report.pdf. Встановіть права 640 (власник: rw-, група: r–, інші: —) за допомогою числового формату. Перевірте.
14	Створіть файл notes.txt. Змініть його групу на team за допомогою chown :team notes.txt. Переконайтеся, що група змінилася.
15	Створіть файл launch.sh. Надайте права 744. Перевірте, чи можуть інші користувачі його виконати (теоретично).
16	Виконайте команду find ~ -type f -perm 777 2>/dev/null. Знайдіть файли з відкритими правами. Поясніть, чому їх наявність небезпечна.
17	Створіть каталог private. Встановіть права 700. Поясніть, чому такі права важливі для особистих даних.
18	Створіть файл app.bin. Надайте групі право виконання: chmod g+x app.bin. Перевірте, чи з’явилося право x у групи.
19	Створіть файл public.txt. Надайте права 644. Переконайтеся, що всі користувачі можуть його читати.
20	Встановіть umask 077. Створіть файл private.txt. Які права він матиме? Чому (очікується 600)?
21	Створіть файл backup.tar.gz. Надайте права 600. Якщо дозволено, спробуйте змінити власника на root (sudo chown root:root backup.tar.gz).
22	Створіть файл file.conf. Встановіть точні права: chmod u=rw,g=r,o= file.conf. Перевірте результат — має бути 640.

23	Створіть каталог projects з правами 755. Додайте у нього файл readme.txt. Перевірте, чи можуть інші користувачі переглядати вміст каталогу.
24	Створіть файл install.sh. Зробіть його виконуваним для всіх: chmod a+x install.sh. Поясніть, які ризики це створює.
25	Створіть файл log.txt. Встановіть права 666. Поясніть, чому це небажано. Виправте командою chmod 640 log.txt.
26	Створіть каталог mydir. Змініть його власника та групу на свого користувача: sudo chown $USER$:USER mydir (або без sudo, якщо власник — ви). Перевірте.
27	Створіть файл temp.data. Встановіть права 620. Поясніть, хто крім власника може читати файл (відповідь: жоден, бо права групи — w–, без читання).
28	Створіть каталог locked. Встановіть права 711. Перевірте, чи можуть інші користувачі переглядати його вміст (відповідь: ні, бо немає права r).
29	Створіть файл readme.md. Встановіть права 744. Перевірте, чи власник може читати, записувати, виконувати, а інші — тільки читати.
30	Виконайте команду find ~ -type f -perm -u+x 2>/dev/null. Знайдіть усі файли, які власник може виконувати. Поясніть, навіщо такий пошук корисний для аудиту безпеки.

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам'яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються

виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов'язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 5

Тема: Управління пристроями зберігання даних в ОС Linux

Мета роботи: Навчитися створювати, форматувати та монтувати розділи на віртуальних дисках, розуміти процес ручного та автоматичного монтування.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

Сучасні інформаційні системи неможливо уявити без надійних та ефективних механізмів зберігання даних, які в операційних системах сімейства Linux реалізуються через складну ієрархію взаємопов'язаних компонентів від фізичних накопичувачів до логічних файлових систем. Розуміння принципів роботи з пристроями зберігання даних є критично важливим для системних адміністраторів, розробників та всіх фахівців, які працюють з серверними системами, оскільки від правильної організації дискового простору залежить не лише продуктивність системи, але й безпека та цілісність збережених даних. Операційна система Linux пропонує потужний набір інструментів для роботи з різноманітними типами накопичувачів, включаючи традиційні жорсткі диски, твердотільні накопичувачі, USB-пристрої та мережеві сховища, що дозволяє гнучко налаштовувати архітектуру зберігання відповідно до специфічних потреб конкретного застосування. Процес управління пристроями зберігання включає в себе множину взаємопов'язаних операцій: від низькорівневого розбиття дисків на розділи та форматування у відповідні файлові системи до високорівневого налаштування автоматичного монтування, управління правами доступу та інтеграції з системами резервного копіювання. Особливу увагу слід приділити розумінню того, що помилки у конфігурації дискових підсистем можуть призвести до втрати даних або неможливості завантаження операційної системи, тому всі операції з пристроями зберігання повинні виконуватися з дотриманням строгих процедур безпеки та обов'язковим створенням резервних копій критично важливої інформації.

Управління пристроями зберігання даних в операційній системі Linux є фундаментальною навичкою системного адміністратора, що охоплює широкий спектр операцій від створення розділів до автоматизації процесів монтування. Для розуміння принципів організації зберігання даних у Linux необхідно розглянути ієрархічну структуру пристроїв, що представлена на рисунку 1.1.

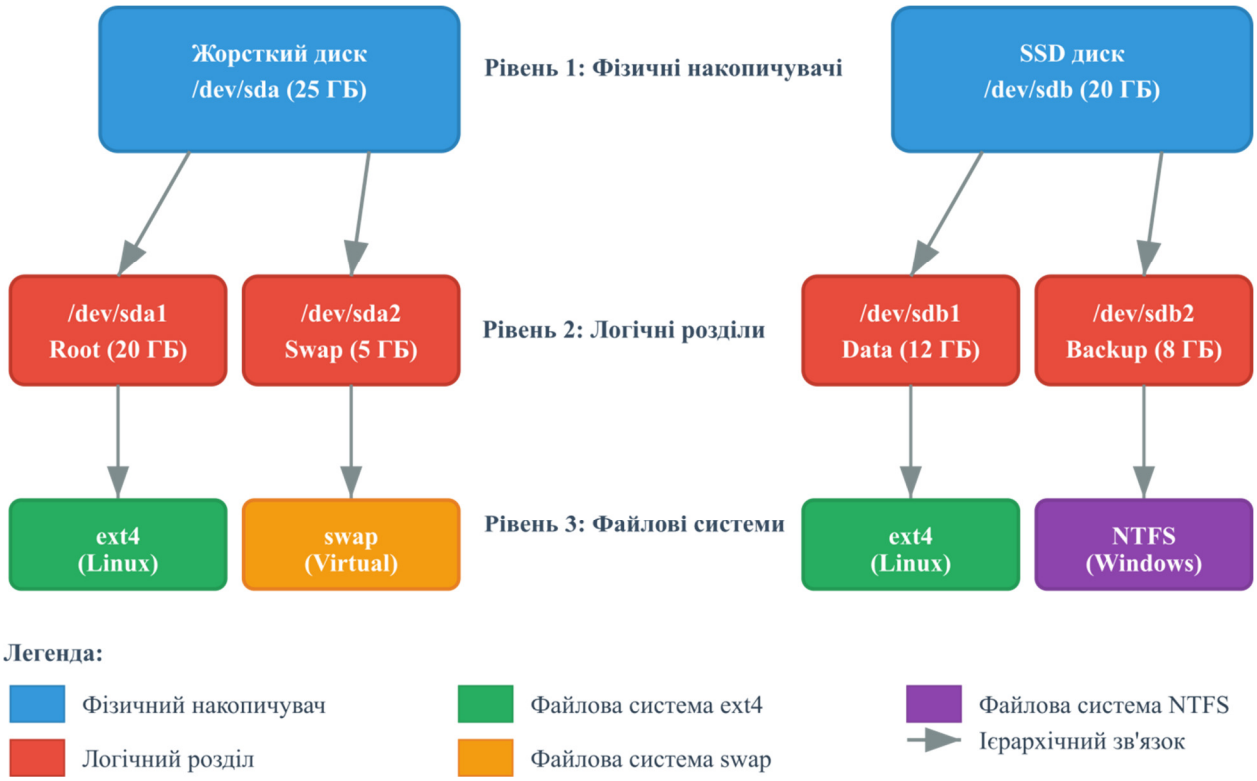

Рисунок 5.1 – Ієрархічна структура пристроїв зберігання даних у Linux

Рисунок демонструє трирівневу структуру організації даних на накопичувачах: на верхньому рівні знаходиться фізичний накопичувач (наприклад, /dev/sda, /dev/sdb), який представляє собою цілісний блочний пристрій як жорсткий диск чи SSD. На середньому рівні розташовуються логічні розділи (наприклад, /dev/sda1, /dev/sdb1, /dev/sdb2), які створюються шляхом поділу фізичного накопичувача на окремі сегменти за допомогою таблиці розділів GPT або MBR. На нижньому рівні в кожному розділі розміщується конкретна файлова система (ext4, NTFS, FAT32), яка визначає спосіб організації файлів і директорій, методи індексації та алгоритми доступу до даних. Стрілки на схемі показують, що один накопичувач може містити декілька розділів, але кожен розділ може мати лише одну файлову систему, що забезпечує чітку ієрархію та логічну організацію дискового простору.

Розділ диска представляє собою логічно виділену частину фізичного накопичувача, яка може містити окрему файлову систему та використовуватися для зберігання різних типів даних. Створення розділів дозволяє ефективно організувати дисковий простір, відокремити системні файли від користувацьких даних, підвищити безпеку та надійність зберігання інформації. Розбиття диска на декілька розділів також сприяє кращому управлінню ресурсами та полегшує процес резервного копіювання.

Сучасні операційні системи Linux підтримують два основні типи таблиць розділів: застарілу MBR (Master Boot Record) та більш сучасну GPT (GUID Partition Table). GPT є рекомендованим стандартом для нових систем, оскільки підтримує диски розміром понад 2 ТБ, дозволяє створювати до 128 розділів на

одному диску та забезпечує вищу надійність завдяки резервуванню критичних структур даних.

Операційна система Linux надає системним адміністраторам широкий набір утиліт командного рядка для виконання всіх необхідних операцій з пристроями зберігання даних, від базового перегляду інформації про диски до складних операцій з управління правами доступу. Ці команди можна логічно згрупувати за функціональним призначенням, що полегшує їх вивчення та практичне застосування. Знання основного набору команд є обов'язковою умовою для ефективної роботи з дисковими підсистемами та забезпечення надійного функціонування серверних систем. Таблиця 1.1 містить систематизований перелік найважливіших команд для адміністрування пристроїв зберігання.

Таблиця 5.1 - Основні команди для управління пристроями зберігання в Linux

Група команд	**Команда**	**Опис**	**Приклад використання**
Перегляд дисків та розділів	`fdisk -l`	Відображення списку всіх блочних пристроїв	`sudo fdisk -l`
	`lsblk`	Показ дерева блочних пристроїв	`lsblk -f`
	`df -h`	Відображення інформації про змонтовані файлові системи	`df -h`
	`du -sh`	Показ розміру директорії	`du -sh /home`
Створення та управління розділами	`fdisk <device>`	Інтерактивне управління розділами	`sudo fdisk /dev/sdb`
	`parted`	Альтернативна утиліта для роботи з розділами	`sudo parted /dev/sdb`
	`gdisk`	Управління GPT розділами	`sudo gdisk /dev/sdb`
Форматування файлових систем	`mkfs.ext4`	Створення файлової системи ext4	`sudo mkfs.ext4 /dev/sdb1`
	`mkfs.ntfs`	Створення файлової системи NTFS	`sudo mkfs.ntfs /dev/sdb1`
	`mkfs.vfat`	Створення файлової системи FAT32	`sudo mkfs.vfat /dev/sdb1`
Монтування та демонтування	`mount`	Монтування файлової системи	`sudo mount /dev/sdb1 /mnt/data`
	`umount`	Демонтування файлової системи	`sudo umount /mnt/data`
	`mountpoint`	Перевірка чи є директорія точкою монтування	`mountpoint /mnt/data`
Управління правами доступу	`chown`	Зміна власника файлів та директорій	`sudo chown user:group /mnt/data`
	`chmod`	Зміна прав доступу	`chmod 755 /mnt/data`

	`chgrp`	Зміна групи файлів	`chgrp users /mnt/data`

Основним інструментом для роботи з розділами дисків у Linux є утиліта fdisk, яка дозволяє переглядати існуючі розділи, створювати нові, змінювати їх розміри та типи. Команда fdisk -l відображає список всіх доступних блочних пристроїв у системі з детальною інформацією про їх розділи, розміри та файлові системи. Інтерактивний режим fdisk активується вказанням конкретного пристрою як параметра команди, наприклад fdisk /dev/sdb, після чого користувач може виконувати різноманітні операції через систему команд-літер.

Таблиця 5.2 - Внутрішні команди утиліти fdisk

Команда	**Призначення**	**Детальний опис**
p	Відображення таблиці розділів	Показує поточну структуру всіх розділів з інформацією про розміри, типи та межі
l	Список типів розділів	Виводить перелік всіх підтримуваних типів розділів з їх кодами
n	Створення нового розділу	Інтерактивне створення первинного або розширеного розділу
d	Видалення розділу	Видаляє існуючий розділ за його номером
t	Зміна типу розділу	Дозволяє змінити тип існуючого розділу (Linux, swap, NTFS тощо)
a	Встановлення/зняття прапора завантаження	Позначає розділ як завантажувальний або знімає цю позначку
w	Запис змін та вихід	Зберігає всі зміни на диск та завершує роботу програми
q	Вихід без збереження	Завершує роботу без збереження внесених змін
m	Відображення довідки	Показує список всіх доступних команд з коротким описом
g	Створення таблиці розділів GPT	Створює нову порожню таблицю розділів GPT
o	Створення таблиці розділів DOS	Створює нову порожню таблицю розділів MBR
s	Зміна порядку розділів	Змінює порядок розділів у таблиці
v	Перевірка таблиці розділів	Виконує перевірку цілісності та коректності структури розділів
i	Інформація про розділ	Відображає детальну інформацію про конкретний розділ
F	Відображення вільного простору	Показує незайнятий простір на диску

Утиліта fdisk працює в інтерактивному режимі та використовує систему односимволних команд для виконання різноманітних операцій над розділами диска. При запуску fdisk з конкретним пристроєм користувач потрапляє в командний режим, де може переглядати поточну структуру розділів, створювати нові розділи, змінювати типи існуючих розділів та виконувати інші адміністративні операції. Програма підтримує як застарілий стандарт MBR (Master Boot Record), так і сучасний GPT (GUID Partition Table), автоматично

визначаючи тип таблиці розділів або дозволяючи користувачу створити новий. Особливістю fdisk є те, що всі операції виконуються в пам'яті і не впливають на фізичний диск до моменту явного збереження змін командою w, що забезпечує додатковий рівень безпеки при роботі з критично важливими даними. Перед збереженням користувач може переглянути всі запропоновані зміни командою p і при необхідності скасувати всі операції командою q без збереження. Важливо пам'ятати, що всі зміни в fdisk зберігаються лише після виконання команди w, що дозволяє безпечно експериментувати з налаштуваннями без ризику пошкодження даних.

Після створення розділу необхідно відформатувати його у відповідну файлову систему, яка визначає спосіб організації та зберігання файлів на диску. В Linux найпоширенішою файловою системою є ext4, яка забезпечує високу продуктивність, надійність та підтримку великих файлів і розділів. Форматування розділу здійснюється за допомогою утиліт родини mkfs, де mkfs.ext4 спеціально призначена для створення файлової системи ext4. Параметр -F при форматуванні дозволяє примусово створити файлову систему навіть якщо пристрій вже містить дані або іншу файлову систему.

Процес монтування є ключовою операцією, що робить файлову систему доступною для користувачів операційної системи. Монтування встановлює зв'язок між блочним пристроєм та певною директорією у файловій системі, яка називається точкою монтування. Після успішного монтування всі файли та директорії з підключеного пристрою стають доступними через вказану точку монтування, інтегруючись у загальну структуру файлової системи Linux.

Команда mount без параметрів відображає список всіх поточно змонтованих файлових систем з детальною інформацією про типи файлових систем та параметри монтування. Для ручного монтування пристрою використовується синтаксис mount <пристрій> <точка_монтування>, де точка монтування повинна бути існуючою директорією з відповідними правами доступу. Важливо попередньо створити точку монтування за допомогою команди mkdir, якщо вона ще не існує. Параметри монтування можуть включати різноманітні опції, такі як режим доступу (тільки для читання або для читання-запису), параметри безпеки, налаштування кешування та інші специфічні для файлової системи опції.

Демонтування файлової системи здійснюється командою umount, яка приймає як параметр точку монтування або пристрій, який необхідно відключити. Перед демонтуванням система перевіряє, чи не використовуються файли з відповідної файлової системи будь-якими процесами, і в разі виявлення активних посилань операція буде заблокована з відповідним повідомленням про помилку.

Автоматичне монтування файлових систем при завантаженні операційної системи налаштовується через конфігураційний файл /etc/fstab, який містить інформацію про всі файлові системи, що повинні монтуватися автоматично. Кожен рядок файлу описує окрему файлову систему та включає шість полів: пристрій або мітку, точку монтування, тип файлової системи, параметри монтування, параметр резервного копіювання та порядок перевірки при

завантаженні. Помилки в конфігурації /etc/fstab можуть призвести до неможливості завантаження системи, тому будь-які зміни у цьому файлі слід виконувати з особливою обережністю та обов'язково створювати резервні копії.

Управління правами доступу до змонтованих файлових систем є критично важливим аспектом безпеки системи, оскільки за замовчуванням новостворені файлові системи належать користувачу root. Команди chown та chmod дозволяють змінювати власника та права доступу до файлів і директорій відповідно, забезпечуючи належний рівень безпеки та функціональності для різних користувачів системи. Рекурсивний параметр -R у команді chown застосовує зміни до всіх файлів та піддиректорій у вказаній директорії.

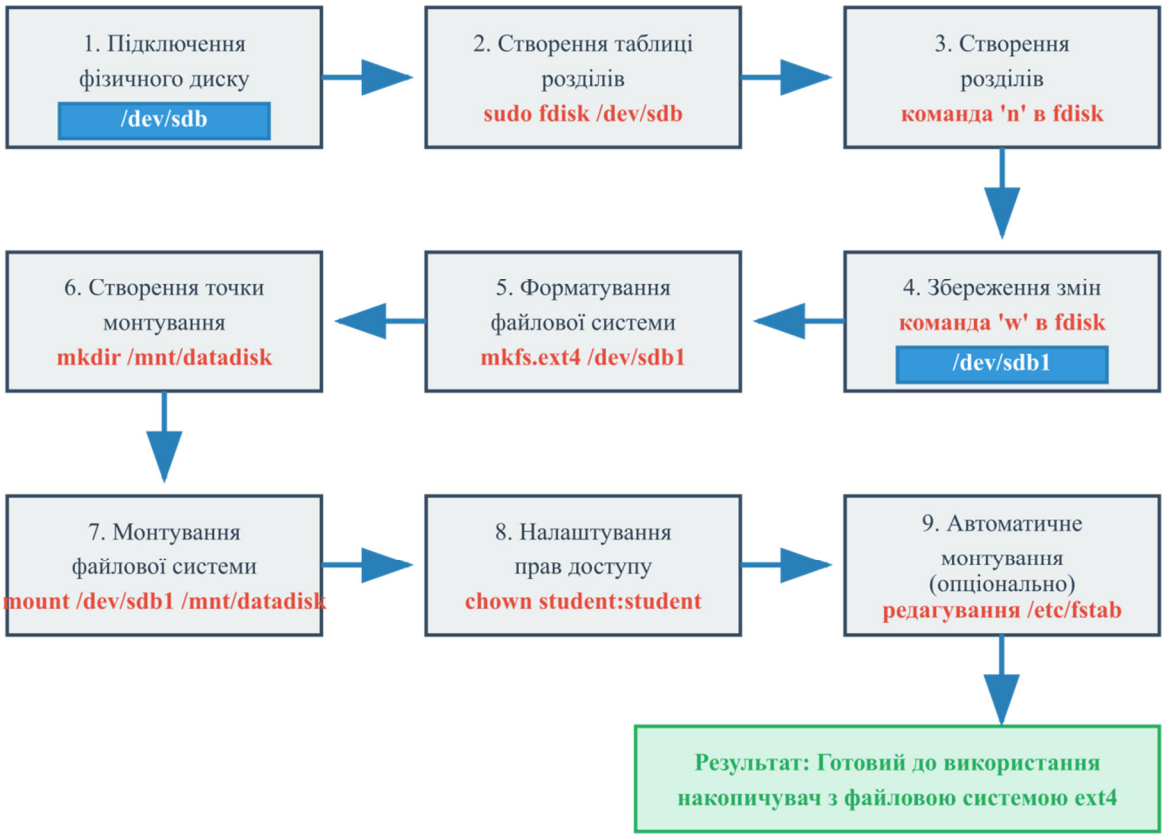

Рисунок 5.2 – Структурна схема процесу управління пристроями зберігання в Linux

Рисунок показує послідовність основних етапів роботи з пристроями зберігання: спочатку фізичний диск підключається до системи та розпізнається ядром як блочний пристрій з відповідним іменем у директорії /dev. Далі за допомогою утиліти fdisk створюється таблиця розділів та один або кілька логічних розділів. Кожен розділ форматується у певну файлову систему за допомогою відповідної утиліти mkfs. Після цього створюється точка монтування у файловій системі, і розділ монтується до неї командою mount. Нарешті, налаштовуються права доступу та, за необхідності, додається запис до файлу /etc/fstab для автоматичного монтування при завантаженні системи.

Практичне застосування отриманих знань дозволяє ефективно розширювати дисковий простір серверів, організовувати окремі розділи для

різних типів даних, налаштовувати системи резервного копіювання та забезпечувати надійне зберігання критично важливої інформації. Розуміння принципів роботи з файловими системами Linux є основою для подальшого вивчення більш складних тем, таких як логічні томи, RAID-масиви та мережеві файлові системи.

2 КЛЮЧОВІ ПИТАННЯ

1. Що таке розділ на диску і навіщо він потрібен?
2. Як визначити, які диски підключені до системи?
3. Як створити новий розділ за допомогою fdisk?
4. Як відформатувати розділ у файлову систему ext4?
5. Що робить команда mount і які параметри вона приймає?
6. Як змонтувати диск автоматично при завантаженні?
7. Як перевірити, чи диск вже змонтований?
8. Що таке точка монтування?
9. Як відмонтувати диск за допомогою umount?
10. Які наслідки помилки в /etc/fstab?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Створити новий віртуальний жорсткий диск в наявній ВМ в VirtualBox.

3.1.1 Запустити Менеджер VirtualBox, вибрати попередньо створену ВМ з встановленою ОС Linux та натиснути кнопку «Налаштувати» на панелі інструментів.

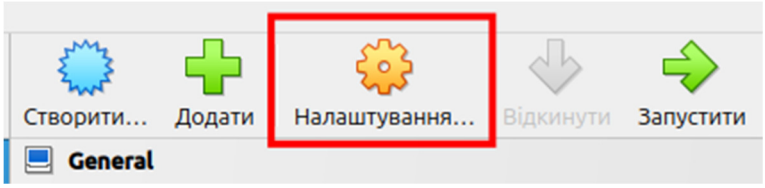

Рисунок 5.3 – Налаштування гостьової ОС

3.1.2 Перейти в розділ «Пам'ять», вибрати контролер SATA і натиснути кнопку для створення нового віртуального жорсткого диска.

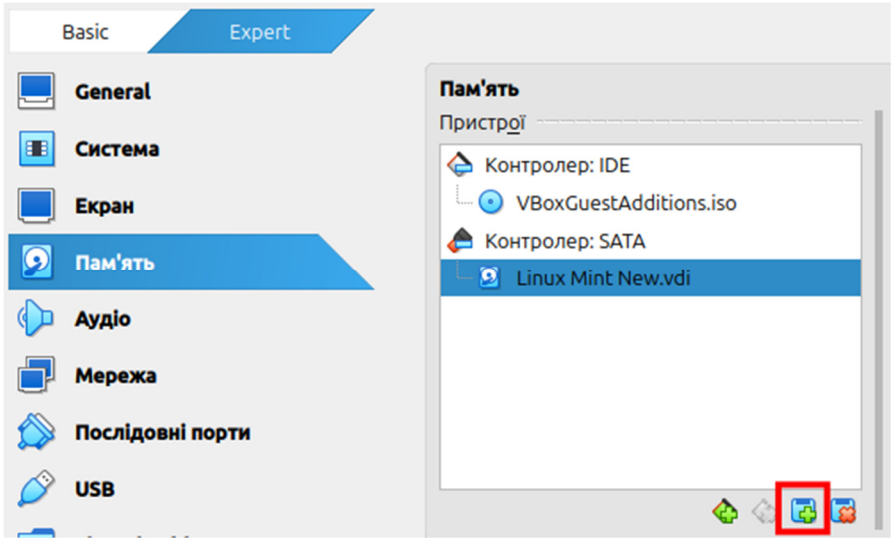

Рисунок 5.4 – Вікно вибору накопичувачів

3.1.3 Натиснути кнопку «Create (Створити)».

3.1.4 Вибрати тип «VDI» диска та обсяг диску, що створюється. З метою зменшення дискового простору, що займає гостьова ОС, обрати *динамічний тип* нового жорсткого диску.

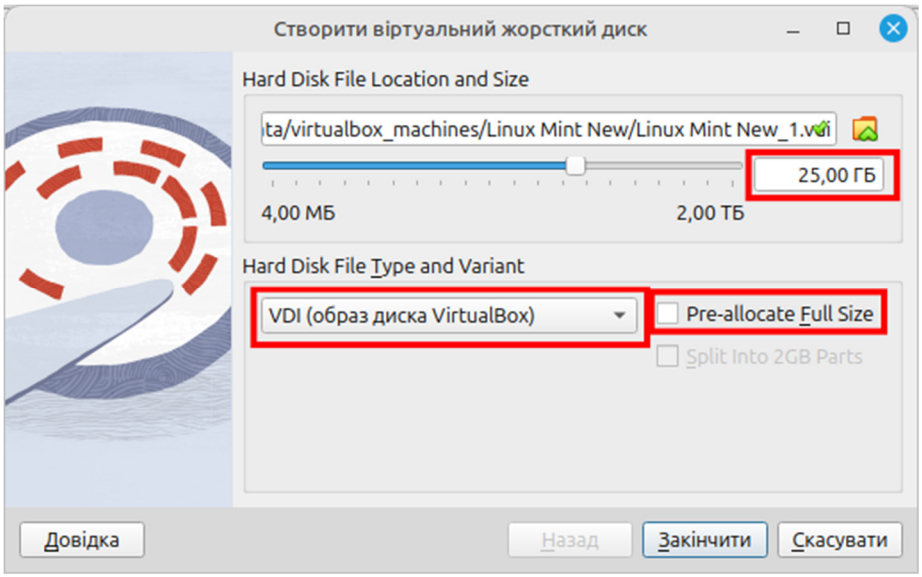

Рисунок 5.5 – Налаштування параметрів нового накопичувача

Зважаючи на динамічний тип створеного диску, можна залишити рекомендований розмір носія і натиснути кнопку «Закінчити». Віртуальний жорсткий диск (файл з розширенням vdi) буде створено, але він ще не підєднаний до віртуальної машини.

3.1.5 Натиснути кнопку "Choose (Обрати)" для підключення образу диску (файлу vdi) до віртуальної машини.

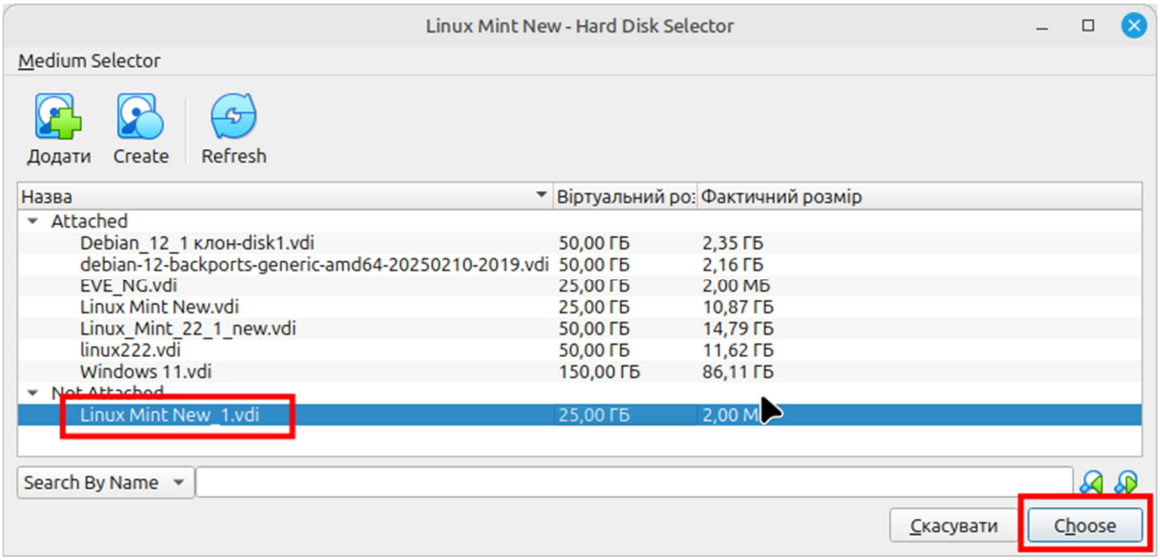

Рисунок 5.6 – Список накопичувачів в програмі Oracle VirtualBox

3.1.6 Перевірити наявність диску в списку підєднаних до контролера SATA.

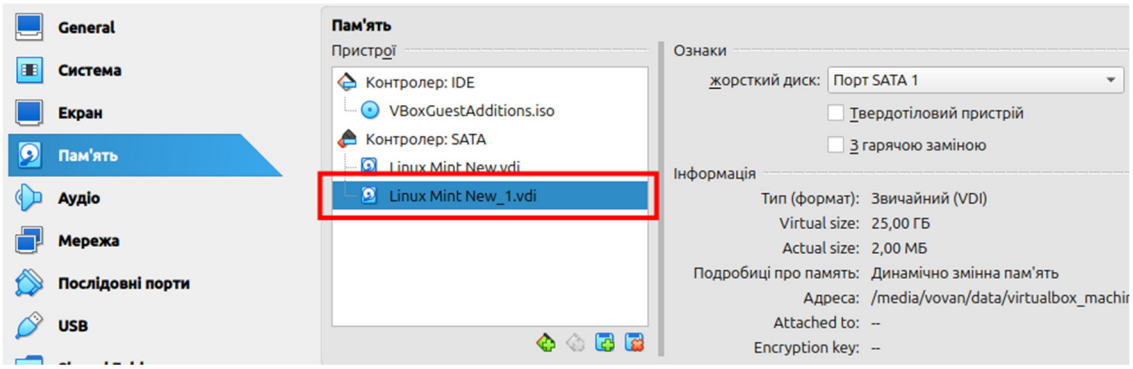

Рисунок 5.7 – Список накопичувачів в гостьовій ОС

3.2 Конфігурування розділу диску

3.2.1 Запустити віртуальну машину та дочекатись завантаження ОС. Запустити термінал і переглянути список доступних накопичувачів за допомогою утиліти fdisk

```
student@student-virtualbox:~$ sudo fdisk –l
```

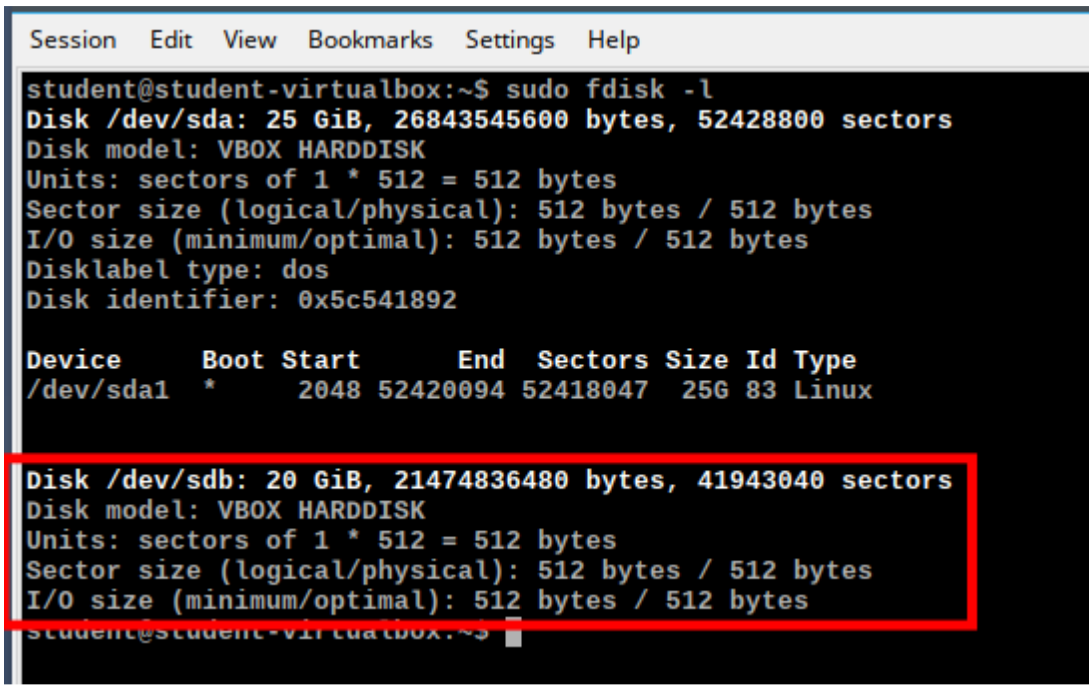

Рисунок 5.8 – Список накопичувачів в програмі fdisk

На даний момент в системі присутні два накопичувачі розміром в 25 та 20 Гб. Перший (/dev/sda) з них містить файлову систему ext4 та файли ОС Linux. Другий (/dev/sdb) поки що не містить таблиці розділів та файлової системи.

3.2.2 Щоб розпочати розбиття диска, запустіть fdisk з ім'ям пристрою.

```
student@student-virtualbox:~$ sudo fdisk /dev/sdb
```

3.2.3 Якщо ви розбиваєте новий диск на розділи, перш ніж починати створення розділів, вам необхідно створити таблицю розділів. Введіть g, щоб створити нову порожню таблицю розділів GPT.

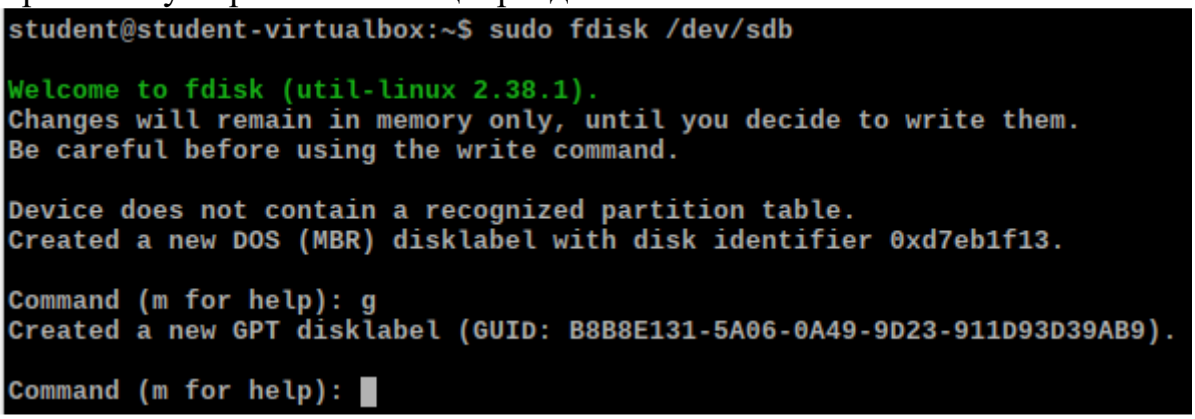

Рисунок 5.9 – Створення таблиці розділів

3.2.4 Виконайте команду n щоб створити новий розділ. Далі розмічаємо щойно створений розділ. Якщо розділ буде на весь диск, то залишаємо все за замовчуванням, нічого не вводячи а просто натискаючи Enter, або вводимо те, як вам потрібно розмітити диск.

```
Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-41943006, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-41943006, default 41940991):

Created a new partition 1 of type 'Linux filesystem' and of size 20 GiB.

Command (m for help):
```

Рисунок 5.10 – Створення нового розділу

3.2.5 Після всіх виконаних дій вводимо команду «w» що означає зберегти налаштування та вийти.

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

student@student-virtualbox:~$
```

Рисунок 5.11 – Збереження змін в налаштуваннях нового розділу

3.2.6 Знову переглянути список доступних накопичувачів та впевнитись в створенні розділу /dev/sdb1 на пристрої /dev/sdb.

```
student@student-virtualbox:~$ sudo fdisk –l
```

```
student@student-virtualbox:~$ sudo fdisk -l
Disk /dev/sda: 25 GiB, 26843545600 bytes, 52428800 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x5c541892

Device     Boot Start      End  Sectors Size Id Type
/dev/sda1  *     2048 52420094 52418047  25G 83 Linux


Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: B8B8E131-5A06-0A49-9D23-911D93D39AB9

Device     Start      End  Sectors Size Type
/dev/sdb1   2048 41940991 41938944  20G Linux filesystem
student@student-virtualbox:~$
```

Рисунок 5.12 – Перевірка характеристик накопичувачів в ОС

3.2.7 Відформатуємо розділ в файлову систему ext4.

```
student@student-virtualbox:~$ sudo mkfs.ext4 -F /dev/sdb1
```

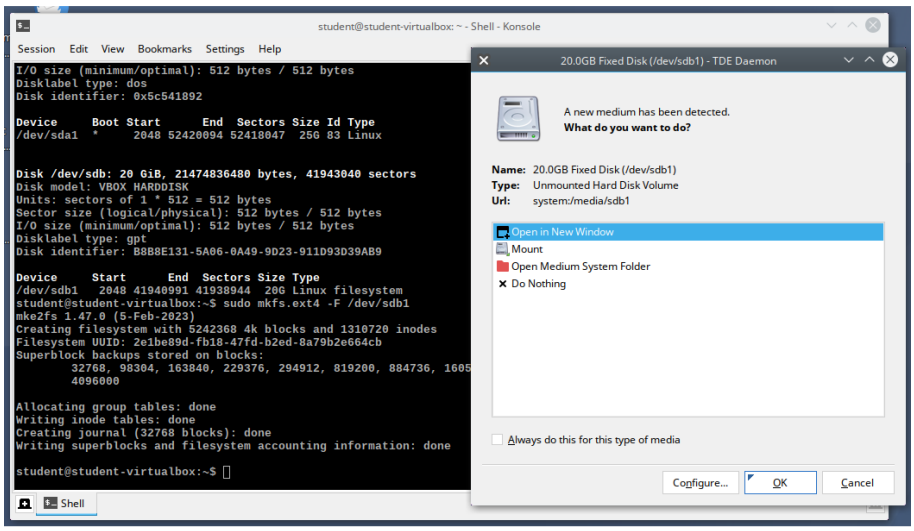

Рисунок 5.13 – Форматування накопичувача

Не погоджуйтесь на автоматичне монтування накопичувача зі створеною файловою системою.

3.3 Ручне монтування накопичувача.

3.3.1 Створіть точки монтування за допомогою команди mkdir.

```
mkdir /mnt/datadisk
```

3.3.2 Примонтуйте диск до створеної точки.

```
sudo mount /dev/sdb1 /mnt/datadisk
```

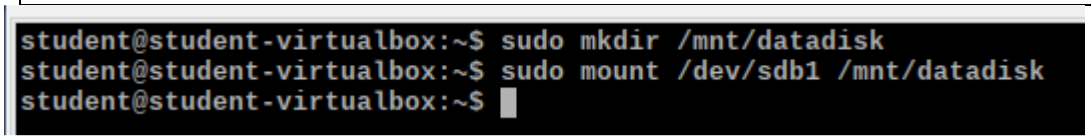

Рисунок 5.14 – Створення каталогу для монтування накопичувача в ОС хоста

3.3.3 Перейти в каталог /mnt та переглянути права доступу до диска

```
student@student-virtualbox:/mnt$ cd /mnt
student@student-virtualbox:/mnt$ ls -l
```

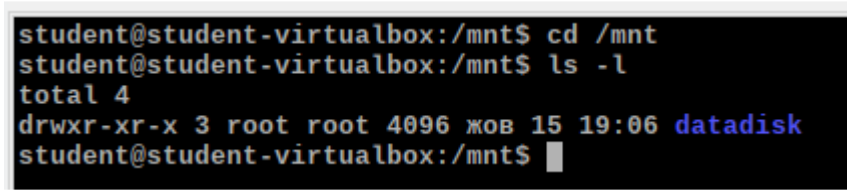

Рисунок 5.15 – Перегляд прав доступу до диску

Власником диску є адміністратор (root), тому звичайний користувач не зможе створювати файли та каталоги.

3.3.4 Ввести команду для зміни групи та власника каталогу

```
student@student-virtualbox:/mnt$ sudo chown -R student:student /mnt/datadisk
```

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам'яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов'язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються

способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 6

Тема: Побудова масштабованої системи зберігання даних в Linux за допомогою LVM

Мета роботи: Ознайомитися з технологією LVM та набути практичних навичок побудови масштабованої системи зберігання даних у Linux

1 КЛЮЧОВІ ПОЛОЖЕННЯ

У сучасних операційних системах ефективне управління дисковим простором є однією з ключових задач адміністрування. Традиційна схема розбиття фізичних дисків на розділи має серйозні обмеження: розмір розділу фіксується на етапі створення, його важко змінити без втрати даних, а ресурси одного диска не можуть бути легко об'єднані з іншими для створення єдиного логічного простору. Ці обмеження особливо гостро відчуваються в умовах постійного зростання обсягів даних і потреби у гнучкій інфраструктурі зберігання. Саме тому в Linux було розроблено технологію Logical Volume Manager (LVM), яка вводить додатковий рівень абстракції між фізичними носіями та файловими системами, дозволяючи адміністраторам працювати з дисковим простором набагато гнучкіше та масштабованіше. LVM стає необхідним інструментом у середовищах, де важливі динамічне розширення сховищ, створення знімків, висока доступність і ефективне використання ресурсів. Ця технологія інтегрована безпосередньо в ядро Linux і працює на основі device mapper, забезпечуючи надійну та продуктивну роботу з логічними томами.

Особливістю LVM є побудова багаторівневої архітектури, що включає фізичні томи, групи томів і логічні томи, що дозволяє відокремити фізичне обладнання від логічної організації даних. Такий підхід дає змогу об'єднувати кілька фізичних дисків — навіть різного розміру та швидкодії — в єдиний пул сховища, з якого потім можна виділяти логічні томи потрібного розміру. Це значно спрощує планування дискового простору, оскільки немає потреби заздалегідь точно визначати, скільки місця знадобиться для кожної файлової системи. При необхідності томи можна збільшувати або зменшувати, переміщати дані між дисками, створювати знімки для резервного копіювання — все це без зупинки системи. Така гнучкість робить LVM незамінним інструментом як у серверних, так і у віртуальних середовищах, де вимоги до сховищ постійно змінюються. Впровадження LVM дозволяє не лише оптимізувати використання дискового простору, але й підвищити загальну надійність та керованість системи.

Менеджер логічних томів (англ. Logical Volume Manager, ***LVM****)* - підсистема операційних систем Linux, що дає змогу використовувати різні області фізичного жорсткого диска або різних жорстких дисків як один логічний том. LVM вбудована в ядро Linux і реалізується на базі device mapper.

Головні переваги LVM - високий рівень абстракції від фізичних дисків, гнучкість і масштабованість. Адміністратор системи може на льоту змінювати

розмір логічного тому, додавати (і видаляти) нові диски. Для LVM томів підтримується зекалювання, снапшоти (persistent snapshot) і striping (розшарування даних між кількома дисками з метою збільшення продуктивності).

Насамперед потрібно розібратися з рівнями дискових абстракцій LVM. Розглянемо приклад диска зі стандартними дисковими абстракціями в ОС Linux без LVM (рис. 6.1).

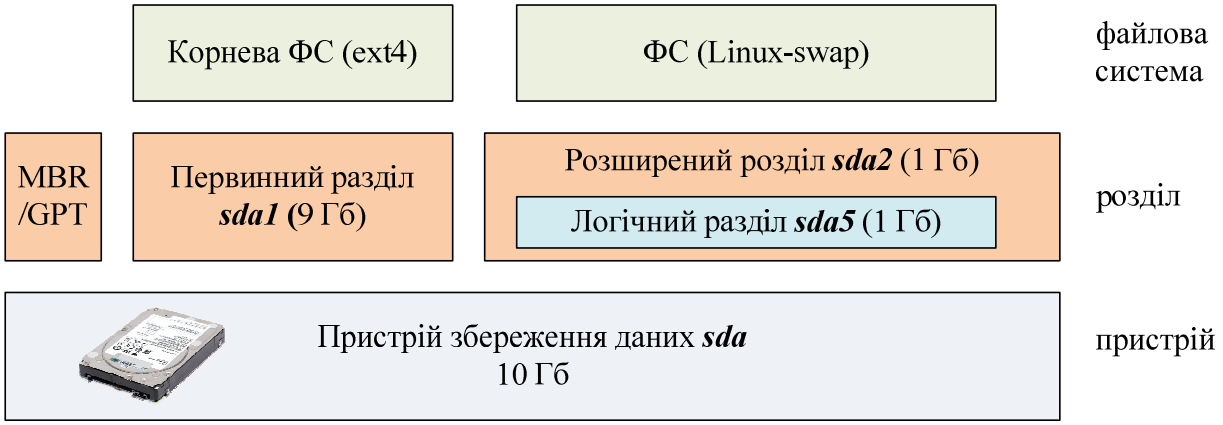

Рисунок 6.1 – Дискові абстракції стандартної системи ОС Linux без LVM

У наведеному прикладі на одному фізичному пристрої *sda* об'ємом 10 Гб створено два розділи: первинний sda1 (9 Гб) і розширений розділ sda2 (1 Гб). Первинний розділ зазвичай використовується для зберігання файлів ОС і даних користувача, а на розширеному в цьому випадку створено логічний розділ sda5. На первинному розділі створено файлову систему типу ext4, а логічний використовується для зберігання файлу підкачки (Linux swap).

Для гнучкої інтеграції декількох накопичувачів в ОС доцільно використовувати менеджер LVM і відповідні дискові абстракції (пристрій LVM):

1) *Фізичний том (англ. physical volume)* - розділ на жорсткому диску або жорсткий диск, на якому створюються логічні томи;

2) *Група томів (англ. volume group):* Набір фізичних томів в один об'єкт. Містять логічні томи. Групу томів можна уявити як жорсткі диски;

3) *Логічний том (англ. logical volume):* аналогічний розділу (hda1, sda1) на не-LVM системах. Так само, як і на них, представляється як блоковий пристрій і може нести файлову систему;

4) *Фізичний діапазон (англ. physical extent)* : Невелика частина диска (зазвичай має розмір 4MB), яка може бути додана до логічного тому. Може бути доданий до будь-якого розділу;

5) *Логічні діапазони (англ. logical extent):* діапазони, на які розбивається логічний том. Їхній обсяг однаковий по всій групі томів.

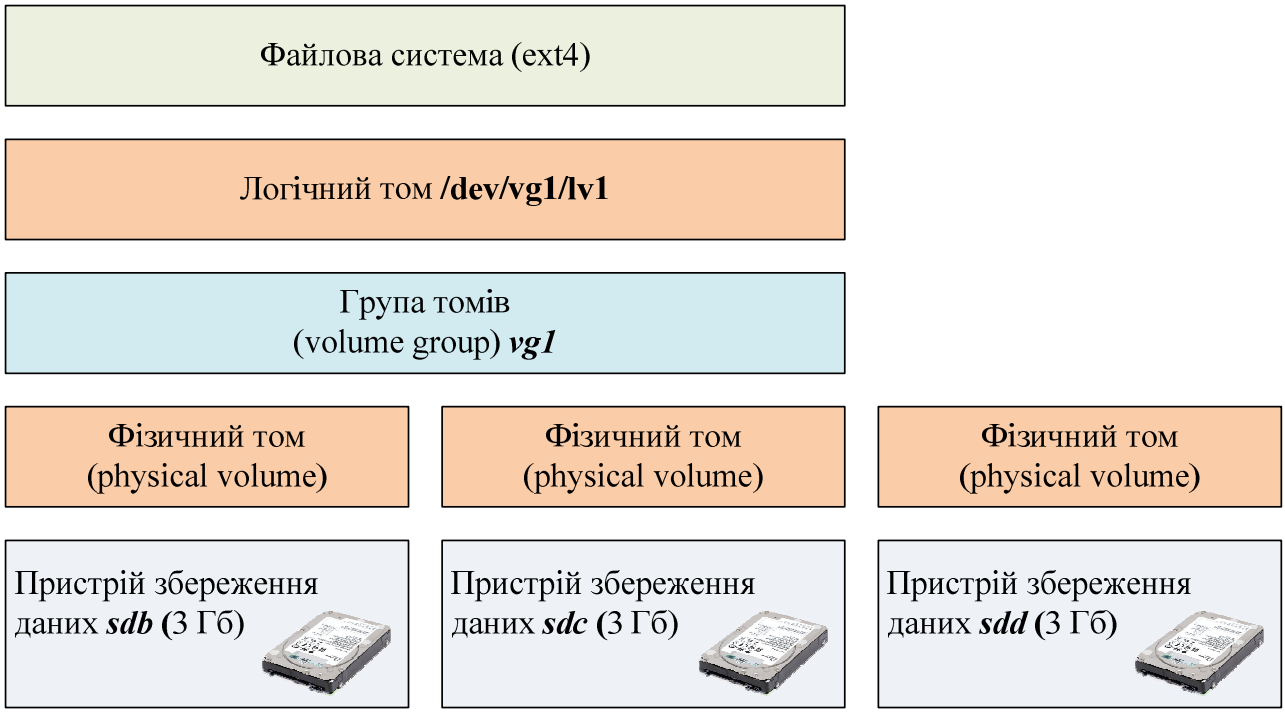

Рисунок 6.2 – Дискові абстракції стандартної системи ОС Linux з LVM

Таблиця 6.1 - Особливості реалізації LVM

Переваги LVM	Недоліки
1) використання будь-якої кількості жорстких дисків або їхніх розділів як один великий розділ. 2) логічні томи можуть бути "розмазані" на кілька жорстких дисків. Максимальний розмір дорівнює обсягу всіх жорстких дисків. 3) зміна/створення/видалення логічних томів у будь-якому вигляді. Тобто не залежить від фізичного розташування тома. 4) зміна/створення/видалення логічних томів і груп томів у режимі "online"(Увага: не всі файлові системи підтримують зміну розміру в режимі "online") 5) Снапшоти дають змогу створювати резервну копію замороженої файлової системи практично на льоту. 6) Підтримка пристроїв для різних цілей, включно з прозорим шифруванням файлової системи та кешуванням часто використовуваних даних. 7) динамічне збільшення розміру логічних томів у міру їх наповнення. Збільшення розміру здійснюється користувачем, деякі файлові системи підтримують зміну розміру в режимі "online".	1) Тільки Linux-сумісний. Відсутність офіційної підтримки в більшості інших ОС (FreeBSD, Windows...). 2) Додаткові кроки в складнішому налаштуванні системи

Створення та видалення

Як уже зазначалося, LVM будується на основі розділів жорсткого диска та/або цілих жорстких дисків. На кожному з дисків/розділів має бути створено фізичний том (physical volume). Наприклад, ми використовуємо для LVM диск sda і розділ sdb2:

```
pvcreate /dev/sda
pvcreate /dev/sdb2
```

На цих фізичних томах створюємо групу томів, яка називатиметься, скажімо, vg1:

```
vgcreate -s 32M vg1 /dev/sda /dev/sdb2
```

Подивимося інформацію про нашу групу томів:

```
vgdisplay vg1
```

Груп можна створити кілька, кожна зі своїм набором томів. Але зазвичай це не потрібно.

Тепер у групі томів можна створити логічні томи lv1 і lv2 розміром 20 Гбайт і 30 Гбайт відповідно:

```
lvcreate -n lv1 -L 20G vg1
lvcreate -n lv2 -L 30G vg1
```

Тепер у нас є блокові пристрої /dev/vg1/lv1 і /dev/vg1/lv2. Залишилося створити на них файлову систему. Тут відмінностей зі звичайними розділами немає:

```
mkfs.ext4 /dev/vg1/lv1
mkfs.reiserfs /dev/vg1/lv2
```

Видалення LVM (або окремих його частин, наприклад, логічних томів або груп томів) відбувається у зворотному порядку - спершу потрібно відмонтувати розділи, потім видалити логічні томи (lvremove), після цього можна видалити групи томів (vgremove) і непотрібні фізичні томи (pvremove).

Додавання фізичних томів

Щоб додати новий диск sdc до групи томів, створимо фізичний том:

```
pvcreate /dev/sdc
```

І додамо його в нашу групу:

```
vgextend vg1 /dev/sdc
```

Тепер можна створити ще один логічний диск (lvcreate) або збільшити розмір наявного (lvresize).

Видалення фізичних томів

Щоб прибрати з працюючої групи томів диск sda спочатку перенесемо всі дані з нього на інші диски:

```
pvmove /dev/sda
```

Потім видалимо його з групи томів:

```
vgreduce vg1 /dev/sda
```

І, нарешті, видалимо фізичний том:

```
pvremove /dev/sda
```

Остання команда просто прибирає позначку про те, що диск є членом lvm. Після видалення з LVM для подальшого використання диск доведеться перерозбивати/переформатувати.

Зміна розмірів

LVM дає змогу легко змінювати розмір логічних томів. Для цього потрібно спочатку змінити сам логічний том:

```
lvresize -L 40G vg1/lv2
```

а потім файлову систему на ньому:

```
resize2fs /dev/vg1/lv2
resize_reiserfs /dev/vg1/lv2
```

2 КЛЮЧОВІ ПИТАННЯ

1. Що таке LVM і які основні рівні абстракції він використовує для управління дисковим простором у Linux?
2. Яка різниця між фізичним томом, групою томів і логічним томом у системі LVM?
3. Навіщо потрібно створювати фізичні томи перед додаванням дисків до групи томів?
4. Як команда vgcreate впливає на організацію сховища і що означає параметр -s 32M?
5. Чому для створення логічного тому використовується синтаксис 100%FREE і як це забезпечує гнучкість системи?
6. Як відрізняється процес форматування логічного тому від форматування звичайного розділу диска?
7. У чому полягає перевага підключення LVM-тому до директорії /media порівняно з іншими точками монтування?
8. Які кроки потрібно виконати, щоб додати новий фізичний диск до існуючої групи томів і розширити логічний том?
9. Навіщо потрібна команда resize2fs після виконання lvresize і чи можна її пропустити?
10. Які переваги і недоліки має використання LVM порівняно з традиційним розділенням диска на статичні розділи?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Створення однієї групи томів у LVM

3.1.1. Створити окрему копію віртуальної машини (ВМ) для виконання лабораторної роботи.

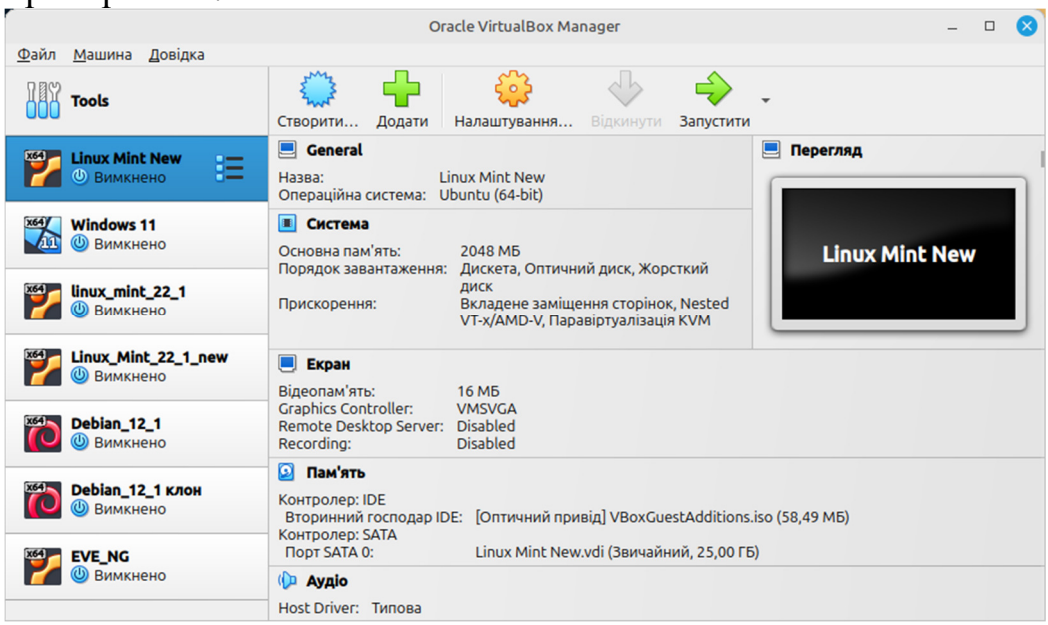

Рисунок 6.3 – Вікно менеджера ВМ VirtualBox

Запустити VirtualBox, навести курсор на зазначену нижче ВМ, натиснути праву кнопку миші і вибрати пункт "*Клонувати*".

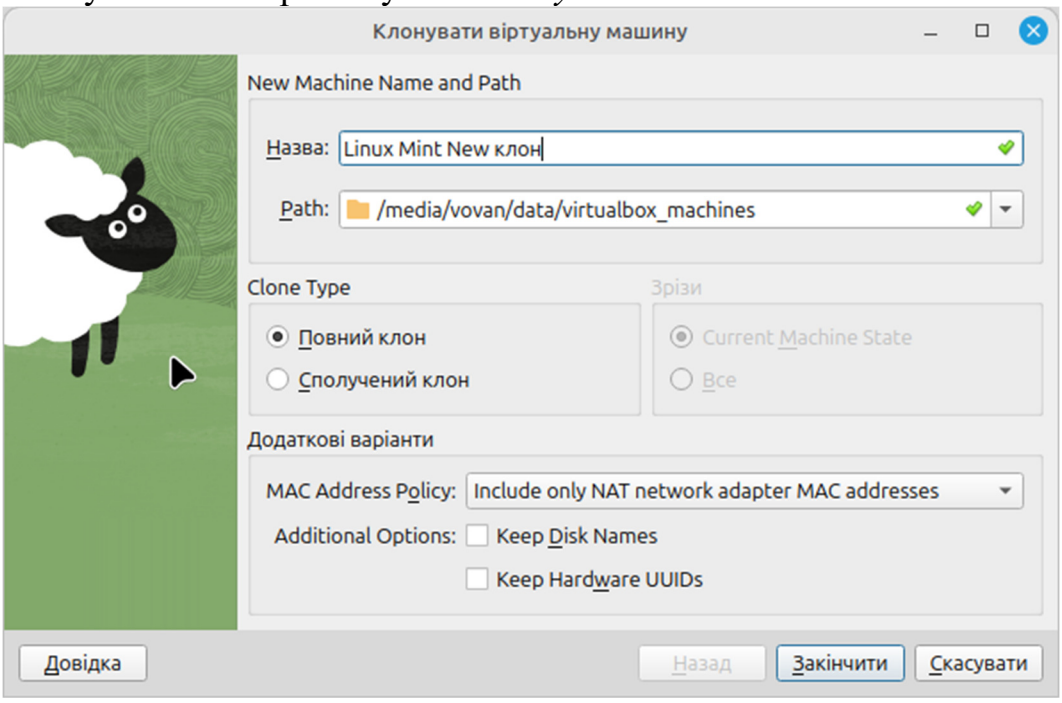

Рисунок 6.4 – Клонування ВМ в VirtualBox

Створити та додати до клонованої ВМ три накопичувачі (жорстких диски) з ємністю (об’ємом) згідно варіанту (таблиця 6.2).

Таблиця 6.2 – Варіанти індивідуальних завдань

№ варіанту	Ємність накопичувача			№ варіанту	Ємність накопичувача		
	1	2	3		1	2	3
1	5	5	5	16	5	20	5
2	5	5	10	17	5	20	10
3	5	5	15	18	5	20	15
4	5	5	20	19	5	20	20
5	5	5	25	20	5	20	25
6	5	10	5	21	5	25	5
7	5	10	10	22	5	25	10
8	5	10	15	23	5	25	15
9	5	10	20	24	5	25	20
10	5	10	25	25	5	25	25
11	5	15	5	26	10	5	5
12	5	15	10	27	10	5	10
13	5	15	15	28	10	5	15
14	5	15	20	29	10	5	20
15	5	15	25	30	10	5	25

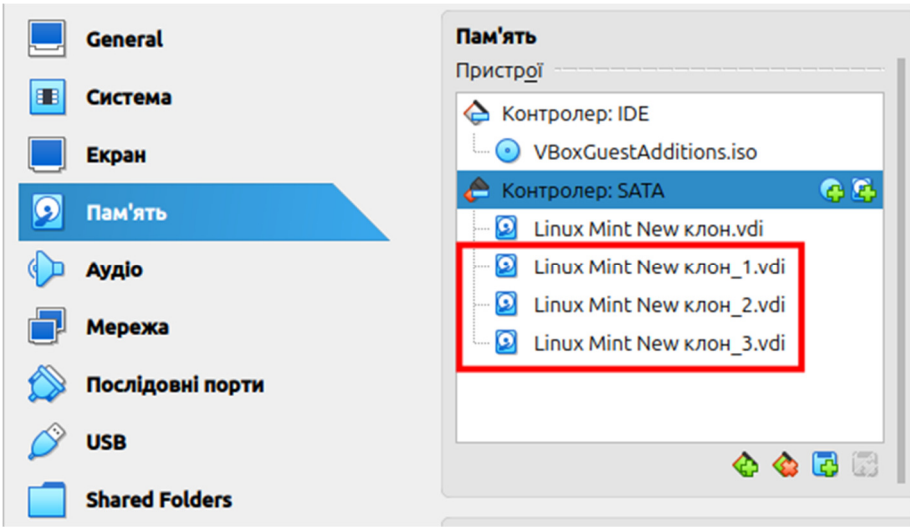

Рисунок 6.5 – Список накопичувачів в гостьовій ОС

3.1.2. Запустити створену в п. 1 копію ВМ. Перейти на робочий стіл ВМ, натиснути кнопку "Menu" і запустити термінал.

3.1.3. Вивести список накопичувачів і розділів на них.

```
student@student-VB:~$ sudo fdisk -l
[sudo] пароль до student:
Диск /dev/sda: 25 GiB, 26843545600 байтів, 52428800 секторів
Модель диска: VBOX HARDDISK
Одиниці: секторів з 1 * 512 = 512 байтів
Розмір сектора (логічного/фізичного): 512 байтів / 512 байтів
Розмір введення-виведення (мінімальний/оптимальний): 512 байтів / 512 байтів
Тип мітки диска: gpt
Ідентифікатор диска: C616C6CD-0BCA-435A-93DD-1541D1C8E5BA
```

```
Пристрій   Початок   Кінець  Сектори Розмір Тип
/dev/sda1     2048     4095     2048     1M BIOS, завантажувальний
/dev/sda2     4096  1054719  1050624   513M Система EFI
/dev/sda3  1054720 52426751 51372032  24,5G Файлова система Linux

Диск /dev/sdb: 25 GiB, 26843545600 байтів, 52428800 секторів
Модель диска: VBOX HARDDISK
Одиниці: секторів з 1 * 512 = 512 байтів
Розмір сектора (логічного/фізичного): 512 байтів / 512 байтів
Розмір введення-виведення (мінімальний/оптимальний): 512 байтів / 512 байтів

Диск /dev/sdc: 25 GiB, 26843545600 байтів, 52428800 секторів
Модель диска: VBOX HARDDISK
Одиниці: секторів з 1 * 512 = 512 байтів
Розмір сектора (логічного/фізичного): 512 байтів / 512 байтів
Розмір введення-виведення (мінімальний/оптимальний): 512 байтів / 512 байтів

Диск /dev/sdd: 25 GiB, 26843545600 байтів, 52428800 секторів
Модель диска: VBOX HARDDISK
Одиниці: секторів з 1 * 512 = 512 байтів
Розмір сектора (логічного/фізичного): 512 байтів / 512 байтів
Розмір введення-виведення (мінімальний/оптимальний): 512 байтів / 512 байтів
```

3.1.4. Створити *фізичні томи (physical volume)* на розділах *sdb* і *sdc*.

```
student@student-VB:~$ sudo pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created.
student@student-VB:~$ sudo pvcreate /dev/sdc
  Physical volume "/dev/sdc" successfully created.
student@student-VB:~$
```

3.1.5. На цих фізичних томах створюємо **групу томів під назвою** *vg1*.

```
student@student-VB:~$ sudo vgcreate -s 32M vg1 /dev/sdb /dev/sdc
  Volume group "vg1" successfully created
student@student-VB:~$
```

3.1.6. Вивести інформацію про наявні групи томів.

```
student@student-VB:~$ sudo vgdisplay
  --- Volume group ---
  VG Name               vg1
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
```

```
  VG Size               <49,94 GiB
  PE Size               32,00 MiB
  Total PE              1598
  Alloc PE / Size       0 / 0
  Free  PE / Size       1598 / <49,94 GiB
  VG UUID               prKWlJ-La5B-IgGj-0laq-ZaLO-KG0j-A1CwGT
```

3.1.7. Створити один логічний том (логічний пристрій */dev/vg1/lv1*), використовуючи весь простір групи *vg1.*

```
student@student-VB:~$ sudo lvcreate -n lvl -l 100%FREE vg1
  Logical volume "lvl" created.
```

3.1.8. Переглянути інформацію про логічні томи LVM.

```
student@student-VB:~$ sudo lvs
  LV   VG  Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv1  vg1 -wi-a----- <49,94g
```

3.1.9. Створити файлову систему на новому логічному пристрої */dev/vg1/lv1*.

```
student@student-VB:~$ sudo mkfs.ext4 /dev/vg1/lv1
mke2fs 1.47.0 (5-Feb-2023)
Створюємо файлову систему з 13090816 4K блоками та 3276800 inode
UUID файлової системи: 6ebe8c23-caa4-4c9d-8f88-7af430516acc
Резервні копії суперблоку зберігаються у таких блоках:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424

Розміщуємо таблиці груп: виконано
Записуємо таблиці inode: виконано
Створюємо журнал (65536 блоків): виконано
Записуємо дані щодо обліку суперблоків та файлової системи: виконано
```

3.1.10. Підмонтувати пристрій до файлової системи ОС.

```
student@student-VB:/media/student$ sudo mkdir /media/student/lvm-data
student@student-VB:/media/student$ sudo mount /dev/vg1/lv1 /media/student/lvm-data
student@student-VB:/media/student$ df -h | grep lvm-data
/dev/mapper/vg1-lv1   49G   24K   47G   1% /media/student/lvm- data
```

Змінити права доступу до пристрою.

```
student@student-VB:/media/student$ ls -l
drwxr-xr-x 3 student student 4096 сер  8 10:59 lvm-data
student@student-VB:/media/student$
```

Отримуємо пристрій зберігання даних зі звичайними правами доступу до файлів і каталогів.

3.2. Додавання фізичного тому до групи томів

3.2.1. Створити *фізичні томи (physical volume)* на розділі *sdd*.

```
student@student-VB:/media/student$ sudo pvcreate /dev/sdd
  Physical volume "/dev/sdd" successfully created.
```

3.2.2. Додати *фізичний том sdd* у групу *vg1*.

```
student@student-VB:/media/student$ sudo vgextend vg1 /dev/sdd
  Volume group "vg1" successfully extended
```

3.2.3. Збільшити розмір наявного логічного диска (логічного пристрою */dev/vg1/lv1*).

```
student@student-VB:/media/student$ sudo lvresize -l +100%FREE /dev/vg1/lv1
  Size of logical volume vg1/lv1 changed from <49,94 GiB (1598 extents) to <74,91 GiB (2397 extents).
  Logical volume vg1/lv1 successfully resize
```

3.2.4. Розширити наявну файлову систему на весь об'єм логічного пристрою */dev/vg1/lv1*.

```
student@student-VB:/media/student$ sudo resize2fs /dev/vg1/lv1
resize2fs 1.47.0 (5-Feb-2023)
Файлову систему у /dev/vg1/lv1 змонтовано до /media/student/lvm-data; надіслано запит щодо інтерактивної зміни розмірів
old_desc_blocks = 7, new_desc_blocks = 10
У файловій системі /dev/vg1/lv1 тепер 19636224 (4K) блоків.
```

Отримуємо *один накопичувач* із загальною ємністю, що відповідає сумарній ємності накопичувачів у групі томів vg1. Для виводу інформації про накопичувачі в ОС можна використати команди *df –hT* та *lsblk –f*.

```
student@student-VB:/media/student$ df -hT
Ф. система          Тип      Розм   Вик  Дост Вик% змонтований на
tmpfs               tmpfs    197M  1,3M  196M   1% /run
/dev/sda3           ext4      24G   11G   13G  46% /
tmpfs               tmpfs    985M     0  985M   0% /dev/shm
tmpfs               tmpfs    5,0M  8,0K  5,0M   1% /run/lock
/dev/sda2           vfat     512M  6,2M  506M   2% /boot/efi
tmpfs               tmpfs    197M  140K  197M   1% /run/user/1000
/dev/sr0            iso9660   59M   59M     0 100% /media/student/VBox_GAs_7.1.12
/dev/mapper/vg1-lv1 ext4      74G   24K   70G   1% /media/student/lvm-data
```

```
student@student-VB:/media/student$ lsblk -f
NAME  FSTYPE FSVER  LABEL          UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda
├─sda1
│
├─sda2
│    vfat   FAT32                 2EBC-118B                             505,8M     1% /boot/efi
└─sda3
     ext4   1.0                   0dea0e01-9211-42c7-9857-5b7af984af89    12,4G    43% /
sdb   LVM2_m LVM2 0                CEqh07-y9P0-Z7h1-VsfO-pkL5-wtde-403Hny
└─vg1-lv1
     ext4   1.0                   6ebe8c23-caa4-4c9d-8f88-7af430516acc    69,9G     0% /media/student/lvm-data
sdc   LVM2_m LVM2 0                Ar0l1k-HGc0-ytDh-t4UQ-vWar-7LaR-fzAtrN
└─vg1-lv1
     ext4   1.0                   6ebe8c23-caa4-4c9d-8f88-7af430516acc    69,9G     0% /media/student/lvm-data
sdd   LVM2_m LVM2 0                fMifqf-0DQV-SMub-QFM7-wuvo-RCe4-En9uf9
└─vg1-lv1
     ext4   1.0                   6ebe8c23-caa4-4c9d-8f88-7af430516acc    69,9G     0% /media/student/lvm-data
sr0   iso966 Joliet VBox_GAs_7.1.12 2025-07-14-13-06-31-55                     0   100% /media/student/VBox_GAs_7.1.12
```

Рисунок 6.6 – Список файлових систем та організація груп накопичувачів

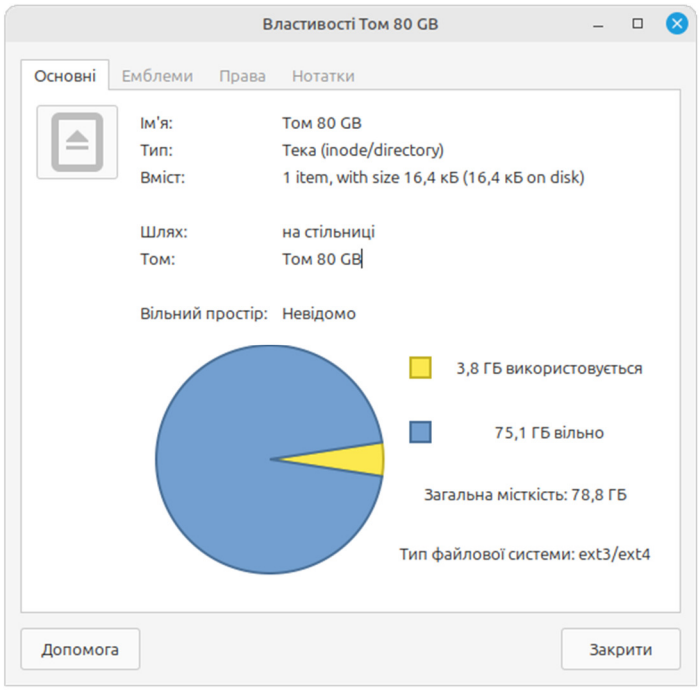

Рисунок 6.7 – Перегляд властивостей накопичувача в ОС

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і

демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам’яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов’язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 7

Тема: Вивчення архіваторів в операційній системі Linux

Мета роботи: Вивчити принципи роботи з поширеними архіваторами в операційній системі Linux.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

Стиснення даних залишається актуальною та важливою операцією в сучасних операційних системах, особливо в середовищах, де обмежені ресурси зберігання або необхідно ефективно передавати великі обсяги інформації. У контексті мережних операційних систем ця технологія дозволяє значно зменшити час передачі файлів між вузлами, оптимізувати використання дискового простору на серверах і забезпечити швидке резервне копіювання даних. Особливо цінним стиснення стає при роботі з лог-файлами, віртуальними машинами та великими масивами даних, які постійно використовуються в корпоративних мережах. В операційній системі Linux архівація тісно інтегрована з іншими системними процесами, зокрема з управлінням пакетами та автоматизацією резервного копіювання. Це робить інструменти стиснення невід'ємною частиною адміністрування серверів і забезпечення їхньої стабільної роботи. Використання різних алгоритмів дозволяє гнучко підбирати оптимальне співвідношення між швидкістю обробки та ступенем стиснення залежно від поточних потреб.

Стиснення даних залишається актуальною та важливою операцією в сучасних операційних системах, особливо в середовищах, де обмежені ресурси зберігання або необхідно ефективно передавати великі обсяги інформації. У контексті мережних операційних систем ця технологія дозволяє значно зменшити час передачі файлів між вузлами, оптимізувати використання дискового простору на серверах і забезпечити швидке резервне копіювання даних. Особливо цінним стиснення стає при роботі з лог-файлами, віртуальними машинами та великими масивами даних, які постійно використовуються в корпоративних мережах. В операційній системі Linux архівація тісно інтегрована з іншими системними процесами, зокрема з управлінням пакетами та автоматизацією резервного копіювання. Це робить інструменти стиснення невід'ємною частиною адміністрування серверів і забезпечення їхньої стабільної роботи. Використання різних алгоритмів дозволяє гнучко підбирати оптимальне співвідношення між швидкістю обробки та ступенем стиснення залежно від поточних потреб.Архівація файлів - це процес об'єднання декількох файлів в один архівний файл, часто з використанням стиснення для зменшення розміру. В операційній системі Linux доступна велика кількість інструментів для архівації, кожен з яких має свої особливості та переваги.

1 Архіватор TAR (Tape Archive)

TAR (Tape ARchive) був розроблений в 1979 році для операційної системи Unix як інструмент для створення резервних копій на магнітні стрічки. Спочатку він був частиною Сьомої редакції Unix (Unix Version 7), випущеної AT&T Bell Laboratories. Автором оригінальної реалізації TAR був Джон Гілмор, який пізніше став одним із засновників Фонду вільного програмного забезпечення (FSF).

У своєму першому втіленні tar був простим інструментом для зберігання файлів та їх метаданих (власника, дозволів, часових міток) у єдиному файлі для зручності перенесення та архівування. Важливо зазначити, що початково tar не виконував функцію стиснення – він лише об'єднував файли без зміни їх розміру. Ця особливість архітектури збереглася до сьогодні.

З розвитком проекту GNU в 1980-х роках, була створена власна версія tar (GNU tar або gtar), яка додала багато нових функцій до оригінального інструменту. Версія GNU tar стала стандартною в більшості дистрибутивів Linux, включно з Debian. Під керівництвом проекту GNU функціональність tar була значно розширена, включаючи підтримку довгих імен файлів, символічних посилань, спеціальних файлів пристроїв та інших специфічних для Unix особливостей файлової системи.

Ключовим етапом у розвитку tar стала інтеграція з програмами стиснення. З появою алгоритму стиснення gzip на початку 1990-х років (автор Жан-лу Гайлі) архіви tar часто почали стискати за допомогою цієї утиліти, що призвело до поширення файлів з розширенням .tar.gz або .tgz. Пізніше з'явилися ще ефективніші методи стиснення: bzip2 (розроблений Джуліаном Сьюардом у 1996 році, дає краще стиснення ніж gzip за рахунок більшого використання CPU) та xz (створений на основі алгоритму LZMA Ігорем Павловим, що забезпечує найвищий ступінь стиснення серед трьох згаданих методів).

У сучасних системах Linux, зокрема в Debian, tar є фундаментальною утилітою, яка використовується не лише для ручного архівування, але й у багатьох автоматизованих процесах, таких як встановлення пакетів (.deb файли є по суті архівами ar, що містять tar-архіви), оновлення системи та резервне копіювання. Сьогодні GNU tar залишається активно підтримуваним проектом і продовжує розвиватися, додаючи нові функції при збереженні зворотної сумісності.

Таблиця 7.1 – Основні ключі архіватора tar

Ключ	Опис
-c	Створення нового архіву
-x	Розпакування архіву
-v	Відображення процесу архівації/розпакування (verbose)
-f	Вказати ім'я архіву
-z	Використовувати gzip для стиснення (створює файли .tar.gz або .tgz)
-j	Використовувати bzip2 для стиснення (створює файли .tar.bz2)
-J	Використовувати xz для стиснення (створює файли .tar.xz)
-t	Перегляд вмісту архіву
-r	Додати файли до існуючого архіву

-u	Оновити файли в архіві
-k	Не перезаписувати існуючі файли при розпакуванні
-C	Змінити каталог перед виконанням операції

Приклади використання:

 # Створення архіву без стиснення
tar -cvf archive.tar /path/to/directory

 # Створення архіву з використанням gzip
tar -czvf archive.tar.gz /path/to/directory

 # Створення архіву з використанням bzip2 (краще стиснення, але повільніше)
tar -cjvf archive.tar.bz2 /path/to/directory

 # Створення архіву з використанням xz (найкраще стиснення, але найповільніше)
tar -cJvf archive.tar.xz /path/to/directory

 # Розпакування архіву
tar -xvf archive.tar

 # Розпакування стисненого архіву (тип стиснення визначається автоматично)
tar -xvf archive.tar.gz

 # Перегляд вмісту архіву
tar -tvf archive.tar.gz

 # Розпакування окремого файлу з архіву
tar -xvf archive.tar.gz path/to/specific/file

 # Розпакування в конкретний каталог
tar -xvf archive.tar.gz -C /destination/directory

2 Архіватор ZIP

Формат ZIP був створений Філом Кацем у 1989 році для його програми PKZIP для MS-DOS. Історія ZIP тісно пов'язана з розвитком технологій стиснення даних у 1980-х роках та конкуренцією між різними форматами архівів.

Передісторія формату ZIP почалася з програми ARC, створеної компанією System Enhancement Associates (SEA) у 1985 році. ARC був стандартом архівації на ранніх BBS (Bulletin Board Systems), але мав певні обмеження. У 1986 році Філ Кац розробив програму PKARC, яка була сумісна з форматом ARC, але працювала швидше. Це призвело до судового позову від SEA до Каца за порушення авторських прав.

Після цієї юридичної суперечки, у 1989 році Кац створив новий формат архівації – ZIP, а також програму PKZIP, яка реалізувала цей формат. Назва "ZIP" натякала на швидкість програми (англійське слово "zip" означає швидкий рух). Формат ZIP був спроектований з урахуванням недоліків попередніх форматів і включав передові на той час алгоритми стиснення, зокрема дефляцію (deflate), яка комбінувала алгоритми LZ77 та кодування Хаффмана.

Значним кроком у розвитку формату ZIP стало створення проекту Info-ZIP у 1989 році. Це був відкритий проект, метою якого була розробка вільних реалізацій програм для роботи з ZIP-архівами для різних операційних систем. Результатом стали програми zip і unzip, які донині широко використовуються в Unix-подібних операційних системах, включаючи Linux.

У 1990-ті роки формат ZIP отримав широке поширення, зокрема завдяки вбудованій підтримці в Windows 95 та більш пізніх версіях Windows (через функцію "стиснуті папки"). Це зробило ZIP фактичним стандартом для архівації файлів у персональних комп'ютерах.

З часом формат ZIP еволюціонував. У 2001 році з'явилася специфікація ZIP 6.3.2, яка додала підтримку шифрування AES (Advanced Encryption Standard) для більш безпечного захисту даних, а також підтримку архівів більше 4 ГБ (64-бітні розширення). У 2006 році Microsoft додала формат Office Open XML (використовується в Microsoft Office 2007 і новіших версіях), який базується на ZIP.

У світі Linux та особливо Debian, утиліти zip і unzip часто використовуються для обміну файлами з користувачами Windows або для роботи з архівами, які мають широку сумісність з різними системами. Незважаючи на появу новіших форматів з кращим стисненням, ZIP залишається популярним через його універсальність та підтримку практично на всіх платформах.

Таблиця 7.2 – Основні ключі архіватора zip/unzip

Утиліта	Ключ	Опис
zip	-r	Рекурсивно архівувати каталоги
zip	-u	Оновити файли в архіві
zip	-d	Видалити файли з архіву
zip	-e	Зашифрувати архів паролем
zip	-9	Максимальний рівень стиснення
zip	-1	Швидке стиснення (мінімальний рівень)
zip	-v	Детальний вивід
unzip	-l	Перегляд вмісту архіву
unzip	-t	Тестування цілісності архіву
unzip	-d	Розпакувати в указаний каталог
unzip	-o	Перезаписувати файли без запиту
unzip	-n	Не перезаписувати існуючі файли
unzip	-P	Вказати пароль

Приклади використання

 # Створення ZIP-архіву

```
zip archive.zip file1 file2 file3

# Створення ZIP-архіву з цілого каталогу
zip -r archive.zip /path/to/directory

# Створення зашифрованого ZIP-архіву
zip -e -r secure_archive.zip /path/to/directory

# Оновлення файлів в архіві
zip -u archive.zip new_file.txt

# Розпакування архіву
unzip archive.zip

# Перегляд вмісту архіву
unzip -l archive.zip

# Розпакування архіву в конкретний каталог
unzip archive.zip -d /destination/directory

# Розпакування зашифрованого архіву
unzip -P password secure_archive.zip
```

3 Архіватор RAR/UNRAR

RAR (Roshal ARchive) був розроблений російським програмістом Євгеном Рошалем у 1993 році. Історія формату RAR є прикладом того, як одна людина може створити продукт, який набуває світової популярності завдяки інноваційним технічним рішенням.

Євген Рошаль почав розробку формату RAR на початку 1990-х років як відповідь на обмеження існуючих архіваторів, таких як ZIP та ARJ. Перша версія програми RAR для DOS була випущена в 1993 році. Назва "RAR" є абревіатурою від "Roshal ARchive" (чи "Roshal ARchiver"), на честь свого творця.

Формат RAR з самого початку був спроектований з урахуванням недоліків попередніх форматів. Серед ключових інновацій були: покращений алгоритм стиснення, який забезпечував кращий коефіцієнт стиснення порівняно з ZIP; підтримка багатотомних архівів, що дозволяло розділяти великі архіви на менші частини (особливо важливо в епоху дискет); розширена підтримка відновлення даних у разі пошкодження архіву; вбудоване шифрування для захисту вмісту.

У 1995 році була випущена версія програми WinRAR для Windows, яка суттєво розширила базу користувачів RAR. Інтуїтивно зрозумілий графічний інтерфейс зробив програму популярною серед звичайних користувачів, а висока ефективність стиснення привернула увагу професіоналів.

З часом формат RAR продовжував розвиватися. У 2002 році з'явилася версія RAR 3.0, яка представила новий формат архівів RAR3, з покращеним

стисненням та додатковими функціями. У 2013 році з випуском WinRAR 5.0 був представлений формат RAR5, який включав новий алгоритм стиснення та підтримку шифрування SHA-256 для підвищення безпеки.

Незважаючи на популярність формату, програми для створення RAR-архівів завжди залишалися власницьким програмним забезпеченням. Євген Рошаль ніколи не відкривав повну специфікацію формату чи вихідний код програми RAR. Однак, він дозволив створити програму unRAR — утиліту для розпакування RAR-архівів, код якої був частково відкритий під спеціальною ліцензією. Ця ліцензія дозволяє вільно використовувати та поширювати unRAR, але забороняє використовувати код для створення програми, яка могла б створювати RAR-архіви.

У світі Linux, особливо в дистрибутивах, орієнтованих на вільне програмне забезпечення, таких як Debian, ця ситуація створила певні труднощі. Оскільки ні програма RAR, ні навіть unRAR не відповідають критеріям вільного програмного забезпечення Debian (DFSG), вони не включені до офіційного репозиторію main, а знаходяться в розділі non-free. Для користувачів Debian це означає, що для роботи з RAR-архівами потрібно спеціально додати репозиторій non-free і встановити пакети звідти.

Незважаючи на ці обмеження, RAR залишається популярним форматом архівації завдяки своїм технічним перевагам, зокрема високому ступеню стиснення та можливості відновлення пошкоджених архівів.

Таблиця 7.3 – Основні ключі архіватора rar/unrar

Утиліта	Ключ	Опис
rar	a	Додати файли до архіву
rar	e	Розпакувати файли з архіву
rar	x	Розпакувати файли з шляхами
rar	t	Перевірити архів
rar	l	Переглянути вміст архіву
rar	-m5	Максимальний рівень стиснення
rar	-m0	Стиснення відсутнє (зберігання)
rar	-p	Установити пароль
rar	-v	Створити багатотомний архів
unrar	x	Розпакувати з шляхами
unrar	e	Розпакувати без шляхів
unrar	l	Перегляд вмісту архіву
unrar	t	Перевірити архів
unrar	-p	Вказати пароль

Приклади використання

 # Створення RAR-архіву
rar a archive.rar file1 file2 file3

 # Створення RAR-архіву з цілого каталогу
rar a -r archive.rar /path/to/directory

 # Створення зашифрованого RAR-архіву
rar a -p archive.rar /path/to/directory

 # Створення багатотомного архіву розміром 10 МБ
rar a -v10m archive.rar /path/to/directory

 # Розпакування архіву зі збереженням шляхів
unrar x archive.rar

 # Розпакування архіву без збереження шляхів
unrar e archive.rar

 # Перегляд вмісту архіву
unrar l archive.rar

 # Перевірка цілісності архіву
unrar t archive.rar

 # Розпакування захищеного паролем архіву
unrar x -p password archive.rar

4 Архіватор 7-Zip (7z)

7-Zip — це програма для архівації файлів, створена російським програмістом Ігорем Павловим у 1999 році. Історія 7-Zip тісно пов'язана з розробкою високоефективних алгоритмів стиснення та розвитком руху за вільне програмне забезпечення.

Перша публічна версія 7-Zip була випущена в 1999 році як проект з відкритим вихідним кодом під ліцензією GNU LGPL. Ігор Павлов прагнув створити архіватор, який би перевершував існуючі рішення за ступенем стиснення, зберігаючи при цьому прийнятну швидкість роботи. Основною інновацією 7-Zip став власний формат архівів 7z, який використовував новий на той час алгоритм стиснення LZMA (Lempel-Ziv-Markov chain Algorithm).

Алгоритм LZMA, розроблений Павловим, забезпечував значно краще стиснення порівняно з існуючими методами, такими як Deflate (використовується в ZIP) або LZSS (використовується в RAR). Це зробило 7-Zip особливо привабливим для архівування великих обсягів даних. LZMA використовує складну схему словникового кодування з адаптивним контекстним моделюванням, що дозволяє досягти високих ступенів стиснення за рахунок більшого використання оперативної пам'яті та обчислювальних ресурсів.

У 2001 році 7-Zip отримав нагороду SourceForge.net Community Choice Awards у категорії "Найкраща технічна розробка", що підтвердило його зростаючу популярність серед користувачів відкритого програмного забезпечення. З часом 7-Zip став одним з найпопулярніших архіваторів для

Windows, завдяки не лише високій ефективності стиснення, але й відкритості коду та безкоштовності.

Важливим моментом в історії 7-Zip стала поява LZMA SDK (Software Development Kit) у 2004 році. Це дозволило іншим розробникам використовувати алгоритм LZMA у своїх проектах. LZMA SDK був випущений під дуже ліберальною ліцензією (подібною до публічного домену для більшості коду), що сприяло швидкому поширенню технології. Сьогодні LZMA використовується в багатьох проектах, включаючи програми стиснення XZ Utils, які широко застосовуються в Linux-системах.

Для користувачів Linux була створена адаптація 7-Zip під назвою p7zip. Розробка p7zip почалася у 2004 році, і програма швидко була включена до репозиторіїв багатьох дистрибутивів Linux, включаючи Debian. В дистрибутиві Debian p7zip доступний у вигляді пакетів p7zip (базова версія) та p7zip-full (повна версія з підтримкою всіх форматів).

З розвитком проекту, 7-Zip отримав підтримку багатьох інших форматів архівів, включаючи ZIP, RAR, TAR, GZIP, BZIP2 та інші, що зробило його універсальним інструментом для роботи з архівами. У 2009 році була представлена версія LZMA2, яка покращила паралельне стиснення та додала інші оптимізації. Ця технологія була реалізована як у 7-Zip, так і в XZ Utils.

Сьогодні 7-Zip залишається активно підтримуваним проектом з відкритим вихідним кодом і продовжує вдосконалюватися. Версія для Linux (p7zip) є важливою частиною екосистеми архіваторів у Debian та інших дистрибутивах, пропонуючи користувачам високоефективне рішення для архівації даних під ліцензією вільного програмного забезпечення.

Таблиця 7.4 – Основні ключі архіватора 7z

Ключ	Опис
a	Додати файли до архіву
x	Розпакувати архів зі збереженням шляхів
e	Розпакувати архів без збереження шляхів
l	Переглянути вміст архіву
t	Перевірити цілісність архіву
-mx=0-9	Рівень стиснення (0 - без стиснення, 9 - максимальний)
-p	Установити пароль
-o	Вказати вихідний каталог
-r	Рекурсивний пошук
-v	Створити багатотомний архів
-mhe=on	Зашифрувати заголовки архіву

Приклади використання

 # Створення архіву 7z
7z a archive.7z file1 file2 file3

 # Створення архіву 7z з цілого каталогу
7z a -r archive.7z /path/to/directory

```
# Створення зашифрованого архіву 7z
7z a -p -mhe=on secure_archive.7z /path/to/directory

# Створення архіву з максимальним стисненням
7z a -mx=9 archive.7z /path/to/directory

# Розпакування архіву зі збереженням шляхів
7z x archive.7z

# Розпакування архіву в конкретний каталог
7z x archive.7z -o/destination/directory

# Перегляд вмісту архіву
7z l archive.7z

# Перевірка цілісності архіву
7z t archive.7z

# Розпакування захищеного паролем архіву
7z x -p password secure_archive.7z
```

2 КЛЮЧОВІ ПИТАННЯ

1. Яка основна відмінність між архіватором tar і іншими архіваторами щодо стиснення?
2. Які алгоритми стиснення найчастіше використовуються з tar і чим вони відрізняються за ефективністю та швидкістю?
3. Чому формат ZIP вважається універсальним для обміну файлами між різними операційними системами?
4. Які переваги надає формат RAR у порівнянні з іншими архіваторами, і чому він вважається власницьким?
5. Який архіватор забезпечує найвищий ступінь стиснення і які ресурси він вимагає?
6. Чому утиліти RAR і unrar розташовані в розділі non-free у Debian?
7. Які формати архівів підтримує 7-Zip, і чим відрізняється його алгоритм LZMA?
8. Для чого використовуються багатотомні архіви, і які архіватори їх підтримують?
9. Які методи шифрування реалізовані в ZIP, RAR і 7z, і який з них найбезпечніший?
10. Як за допомогою команди time виміряти продуктивність процесу архівації та розархівації?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Встановіть необхідні архіватори в системі Debian:

```
sudo apt update
sudo apt install tar zip p7zip-full
```

3.2 Створіть тестовий каталог з різними типами файлів (текстові, зображення, документи).

```
mkdir ~/test-archive
touch ~/test-archive/textfile{1..5}.txt
```

3.3 Виконайте архівацію тестового каталогу, використовуючи кожен з вивчених архіваторів:

- tar (з різними методами стиснення: gzip, bzip2, xz);
- zip;
- 7z.

Для визначення часу виконання команди в Linux можна скористатися утилітою time, яка показує реальний час (real), час процесора в користувацькому режимі (user) та час у режимі ядра (sys). Наприклад, щоб виміряти, скільки часу займає перелік файлів у поточній директорії, виконайте:

time ls -l

Результат виведеться у форматі:

real 0m0.005s
user 0m0.002s
sys 0m0.003s

Це допомагає аналізувати продуктивність скриптів та оптимізувати їхню роботу. Занесіть отримані результати у табл. 7.5.

Таблиця 7.5 – Порівняльна характеристика архіваторів

Архіватор	Розмір архіву	Час архівації	Час розархівації
tar			
tar.gz			
tar.bz2			
tar.xz			
zip			
7z			

3.4 Порівняйте розміри отриманих архівів та швидкість архівації/розархівації.

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов’язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім’я та по батькові студента, назва академічної групи, а також прізвище, ім’я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам’яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов’язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 8

Тема: Налаштування та аналіз мережевих підключень у Linux

Мета роботи: Навчитися налаштовувати мережеві інтерфейси, використовувати мережеві команди для діагностики, підключення та завантаження даних.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

1.1 Роль комп'ютерних мереж у сучасних інформаційних системах

Комп'ютерні мережі є основою сучасної інформаційної інфраструктури, забезпечуючи інтеграцію розподілених обчислювальних ресурсів та створюючи єдине середовище для спільної роботи користувачів і систем. Мережеві технології дозволяють організувати ефективну взаємодію між комп'ютерами, серверами, мобільними пристроями та спеціалізованим обладнанням незалежно від їхнього географічного розташування. Інтеграція систем через мережеві з'єднання забезпечує централізоване управління ресурсами, розподілену обробку даних та можливість створення масштабованих додатків. Сучасні мережі підтримують широкий спектр сервісів: від базового обміну файлами та електронної пошти до складних хмарних обчислювань, систем віртуалізації та інтернету речей. Операційна система Linux відіграє ключову роль у цій екосистемі, надаючи надійну та гнучку платформу для розгортання мережевих служб та управління мережевою інфраструктурою. Розуміння принципів роботи мережевих технологій у Linux є критично важливим для сучасних IT-фахівців, оскільки більшість серверних систем та мережевого обладнання базується на Linux-платформах.

Основні команди для мережевого адміністрування Linux, необхідні для ефективного управління мережевими ресурсами наведено в таблиці 8.1. Ці команди згруповані за функціональними категоріями та представляють собою основний інструментарій сучасного мережевого адміністратора.

Таблиця 8.1 - Основні команди для мережевого адміністрування Linux

Категорія	Команда	Опис	Приклад використання
Управління інтерфейсами	`ip addr show`	Відображення інформації про мережеві інтерфейси	`ip addr show dev eth0`
	`ip addr add`	Призначення IP-адреси інтерфейсу	`ip addr add 192.168.1.10/24 dev eth0`
	`ip addr del`	Видалення IP-адреси з інтерфейсу	`ip addr del 192.168.1.10/24 dev eth0`
	`ip link set up/down`	Активація/деактивація інтерфейсу	`ip link set eth0 up`

	nmcli	Управління NetworkManager	nmcli connection show
Діагностика зв'язності	ping	Перевірка доступності хоста	ping -c 4 google.com
	traceroute	Відстеження маршруту до хоста	traceroute google.com
	tracepath	Альтернатива traceroute	tracepath google.com
	nslookup	Запити до DNS	nslookup google.com
	dig	Детальні DNS-запити	dig google.com A
Аналіз з'єднань	ss -tuln	Відображення прослуховуваних портів	ss -tuln
	ss -tulnp	Показати порти з процесами	ss -tulnp
	ss -s	Статистика мережевих з'єднань	ss -s
	netstat -tuln	Застаріла альтернатива ss	netstat -tuln
Маршрутизація	ip route show	Відображення таблиці маршрутизації	ip route show
	ip route add	Додавання маршруту	ip route add 192.168.2.0/24 via 192.168.1.1
	ip route del	Видалення маршруту	ip route del 192.168.2.0/24
	route -n	Застаріла команда перегляду маршрутів	route -n
Віддалений доступ	ssh	Безпечне підключення до віддаленої системи	ssh user@192.168.1.100
	scp	Копіювання файлів через SSH	scp file.txt user@host:/path/
	sftp	Безпечний FTP через SSH	sftp user@host
Завантаження файлів	wget	Завантаження файлів з веб-ресурсів	wget -O file.txt https://example.com/file
	curl	Універсальний клієнт для передачі даних	curl -o file.txt https://example.com/file
	curl -I	Отримання лише заголовків HTTP	curl -I https://example.com

1.2 Основні принципи мережевого адміністрування в Linux

Операційна система Linux надає потужний набір інструментів для управління мережевими з'єднаннями через командний рядок, що забезпечує максимальну гнучкість та контроль над мережевою конфігурацією. Сучасні дистрибутиви Linux використовують нове покоління утиліт, серед яких команда

ip повністю замінила застарілу ifconfig, а утиліта ss стала альтернативою традиційній netstat. Ці інструменти пропонують більш детальну інформацію, швидше виконання операцій та кращу інтеграцію з ядром системи. Адміністратори повинні володіти навичками роботи з цими сучасними утилітами для ефективного управління мережевими ресурсами. Команда ip addr дозволяє переглядати та модифікувати конфігурацію мережевих інтерфейсів, відображаючи детальну інформацію про IP-адреси, маски підмережі, MAC-адреси та стан інтерфейсів.

Мережева діагностика є критично важливою складовою адміністрування систем Linux. Утиліта ping залишається основним інструментом для перевірки доступності віддалених хостів та вимірювання затримок у мережі. Команди traceroute та tracepath дозволяють відстежувати маршрут проходження пакетів через мережеву інфраструктуру, виявляючи проблемні ділянки та аналізуючи топологію мережі. Для діагностики системи доменних імен використовуються утиліти nslookup та dig, які надають детальну інформацію про DNS-записи та процес розв'язання імен. Ці інструменти дозволяють швидко ідентифікувати та локалізувати мережеві проблеми на різних рівнях протокольного стеку.

1.3 Віддалений доступ та передача файлів

Протокол SSH (Secure Shell) є стандартом для безпечного віддаленого доступу до Linux-систем. Базова команда ssh user@hostname встановлює зашифроване з'єднання з віддаленим сервером, забезпечуючи конфіденційність та цілісність передачі даних. SSH підтримує автентифікацію за допомогою паролів або криптографічних ключів, причому другий метод вважається більш безпечним для автоматизованих процесів. Утиліта scp дозволяє безпечно копіювати файли між локальною та віддаленою системами, використовуючи той же протокол шифрування. Для більш складних операцій передачі файлів доступний sftp, який надає інтерактивний інтерфейс, подібний до традиційного FTP, але з повним шифруванням трафіку.

Завантаження файлів з мережевих ресурсів здійснюється за допомогою спеціалізованих утиліт wget та curl, кожна з яких має свої особливості та переваги. Утиліта wget оптимізована саме для завантаження файлів та підтримує рекурсивне завантаження, продовження перерваних завантажень та роботу в фоновому режимі. Команда curl є більш універсальною та підходить для взаємодії з різними протоколами та веб-сервісами, включаючи REST API. За замовчуванням wget зберігає файли на диск, тоді як curl виводить вміст у стандартний вивід, що робить його зручнішим для обробки даних у скриптах. Обидві утиліти підтримують автентифікацію, роботу з cookie та налаштування заголовків HTTP.

1.4 Аналіз мережевих з'єднань та портів

Моніторинг активних мережевих з'єднань є ключовою функцією забезпечення безпеки та продуктивності системи. Сучасна утиліта ss надає

детальну інформацію про всі активні сокети в системі, включаючи TCP та UDP з'єднання, Unix-сокети та RAW-сокети. Команда ss -tuln відображає всі прослуховувані порти у числовому форматі, що дозволяє швидко ідентифікувати запущені служби. Додавання параметра -p показує ідентифікатори процесів та назви програм, які використовують конкретні порти. Ця інформація критично важлива для виявлення несанкціонованих служб або аналізу використання мережевих ресурсів конкретними додатками.

Таблиця маршрутизації визначає правила направлення мережевого трафіку в системі Linux. Команда ip route show відображає поточну конфігурацію маршрутизації, включаючи маршрути за замовчуванням, статичні маршрути та маршрути до локальних мереж. Розуміння структури таблиці маршрутизації дозволяє адміністраторам оптимізувати мережевий трафік, налаштовувати складні мережеві топології та вирішувати проблеми з доступністю віддалених ресурсів. Модифікація маршрутів здійснюється за допомогою команди ip route add/del, що дозволяє динамічно адаптувати мережеву конфігурацію відповідно до поточних потреб.

1.5 Сучасні підходи до конфігурації мережі в Linux

У сучасних дистрибутивах Linux, включаючи Linux Mint та Ubuntu, традиційний файл /etc/network/interfaces поступово втрачає актуальність і замінюється більш сучасними рішеннями для управління мережевою конфігурацією. NetworkManager стає стандартним інструментом для управління мережевими з'єднаннями, а файл /etc/network/interfaces вважається застарілим. NetworkManager надає єдиний інтерфейс для управління всіма типами мережевих підключень, включаючи дротові, бездротові, VPN та мобільні з'єднання. Основною перевагою NetworkManager є автоматичне виявлення та конфігурація мережевих інтерфейсів, динамічне керування з'єднаннями та інтеграція з графічним інтерфейсом користувача. Утиліта nmcli є командно-рядковим інтерфейсом для NetworkManager, що дозволяє виконувати всі функції управління мережею з терміналу та використовувати NetworkManager у скриптах.

Конфігурація мережевих з'єднань через nmcli включає створення профілів підключень, які зберігають всі необхідні параметри для конкретного типу з'єднання. Команда nmcli connection show відображає список всіх доступних профілів з'єднань та їх поточний стан. Для створення нового профілю використовується синтаксис nmcli connection add con-name [назва] ifname [інтерфейс] type [тип], що дозволяє детально налаштувати всі параметри з'єднання. Модифікація існуючих профілів здійснюється командою nmcli connection modify, яка підтримує зміну IP-адрес, DNS-серверів, маршрутів та інших мережевих параметрів. Активація та деактивація з'єднань виконується за допомогою nmcli connection up/down, що забезпечує миттєве застосування змін без необхідності перезавантаження мережевих служб. Команди nmcli для управління мережевими з'єднаннями, що демонструють основні можливості

сучасного інструменту конфігурації мережі в Linux-системах, наведено в табл. 8.2.

Таблиця 8.2 - Команди NetworkManager CLI (nmcli) для управління мережевими з'єднаннями

Категорія	Команда	Опис	Приклад використання
Загальна інформація	`nmcli general status`	Загальний стан NetworkManager	`nmcli general status`
	`nmcli general hostname`	Відображення/зміна імені хоста	`nmcli general hostname new-name`
	`nmcli radio all`	Управління радіо-інтерфейсами	`nmcli radio wifi off`
Управління пристроями	`nmcli device show`	Інформація про мережеві пристрої	`nmcli device show eth0`
	`nmcli device status`	Статус всіх мережевих пристроїв	`nmcli device status`
	`nmcli device connect`	Автоматичне підключення пристрою	`nmcli device connect eth0`
	`nmcli device disconnect`	Відключення пристрою	`nmcli device disconnect eth0`
Профілі з'єднань	`nmcli connection show`	Список всіх профілів з'єднань	`nmcli connection show --active`
	`nmcli connection add`	Створення нового профілю	`nmcli con add con-name "Static" type ethernet ifname eth0`
	`nmcli connection modify`	Модифікація профілю	`nmcli con mod "Static" ipv4.addresses 192.168.1.10/24`
	`nmcli connection delete`	Видалення профілю	`nmcli connection delete "Profile-name"`
Активація з'єднань	`nmcli connection up`	Активація профілю	`nmcli connection up "Static"`
	`nmcli connection down`	Деактивація профілю	`nmcli connection down "Static"`
	`nmcli connection reload`	Перезавантаження конфігурації	`nmcli connection reload`
Конфігурація IP	Статична IP-адреса	Налаштування статичної IP	`nmcli con mod "Static" ipv4.method manual`
	DNS-сервери	Налаштування DNS	`nmcli con mod "Static" ipv4.dns "8.8.8.8 8.8.4.4"`

	Шлюз за замовчуванням	Налаштування gateway	`nmcli con mod "Static" ipv4.gateway 192.168.1.1`
	DHCP конфігурація	Увімкнення DHCP	`nmcli con mod "Profile" ipv4.method auto`
Wi-Fi управління	`nmcli device wifi list`	Список доступних Wi-Fi мереж	`nmcli device wifi list`
	`nmcli device wifi connect`	Підключення до Wi-Fi	`nmcli device wifi connect "SSID" password "password"`
	`nmcli device wifi hotspot`	Створення точки доступу	`nmcli device wifi hotspot ssid MyHotspot password mypass`

1.6 Управління мережевими інтерфейсами

Управління мережевими інтерфейсами в Linux здійснюється за допомогою команд сімейства ip, які надають повний контроль над конфігурацією мережевого обладнання. Активація та деактивація інтерфейсів виконується командами ip link set [interface] up/down, що дозволяє швидко змінювати стан мережевих адаптерів без перезавантаження системи. Призначення IP-адрес здійснюється за допомогою ip addr add/del, що підтримує як статичну конфігурацію, так і тимчасові зміни для тестування. Для автоматичного отримання мережевих параметрів може використовуватися протокол DHCP, але в сучасних системах рекомендується застосовувати NetworkManager з командою nmcli device connect [interface] для автоматичної конфігурації.

Робота з множинними мережевими інтерфейсами вимагає розуміння принципів їх взаємодії та пріоритетів маршрутизації. Кожен інтерфейс може мати власну IP-адресу, маску підмережі та специфічні маршрути, що дозволяє створювати складні мережеві конфігурації. Моніторинг стану інтерфейсів здійснюється за допомогою ip addr show, яка відображає детальну інформацію про всі налаштування, включаючи MTU, статистику передачі даних та прапорці стану. Правильне налаштування множинних інтерфейсів критично важливе для забезпечення відмовостійкості мережевих з'єднань та оптимального розподілу навантаження.

1.7 Віртуальні мережеві середовища

VirtualBox надає декілька режимів мережевої конфігурації для віртуальних машин, кожен з яких має специфічні характеристики та області застосування. Режим NAT (Network Address Translation) дозволяє гостьовій операційній системі отримувати доступ до зовнішньої мережі через хост-систему, але робить віртуальну машину недоступною для прямих з'єднань ззовні. Цей режим ідеально підходить для базових задач тестування та розробки, коли потрібен лише вихідний доступ до інтернету. Мостовий режим (Bridged) інтегрує

віртуальну машину безпосередньо в фізичну мережу, надаючи їй власну IP-адресу в тій самій підмережі, що й хост-система. Внутрішня мережа створює ізольоване мережеве середовище для взаємодії між кількома віртуальними машинами без доступу до зовнішніх ресурсів.

Конфігурація віртуальних мережевих адаптерів у VirtualBox вимагає розуміння особливостей кожного режиму та їх впливу на функціональність гостьової системи. Host-only режим створює приватну мережу між хост-системою та віртуальними машинами, забезпечуючи безпечне середовище для тестування мережевих додатків без ризику для зовнішньої мережі. Налаштування множинних адаптерів дозволяє створювати складні мережеві сценарії, коли віртуальна машина одночасно має доступ до різних мережевих сегментів. Правильний вибір мережевого режиму критично важливий для досягнення поставлених цілей лабораторної роботи та забезпечення необхідного рівня ізоляції або інтеграції з мережевою інфраструктурою.

Структура мережевої конфігурації Linux, що відображає ієрархічну організацію мережевих компонентів операційної системи показана на рис. 8.1. У верхній частині схеми розташовується рівень додатків, який включає мережеві служби та програми користувача. Середній рівень представлений протокольним стеком TCP/IP з основними протоколами транспортного та мережевого рівнів. Нижня частина схеми демонструє фізичний рівень з мережевими інтерфейсами та драйверами обладнання. Стрілки на схемі показують напрямки передачі даних між рівнями, а також взаємодію з зовнішніми мережевими ресурсами через різні типи з'єднань.

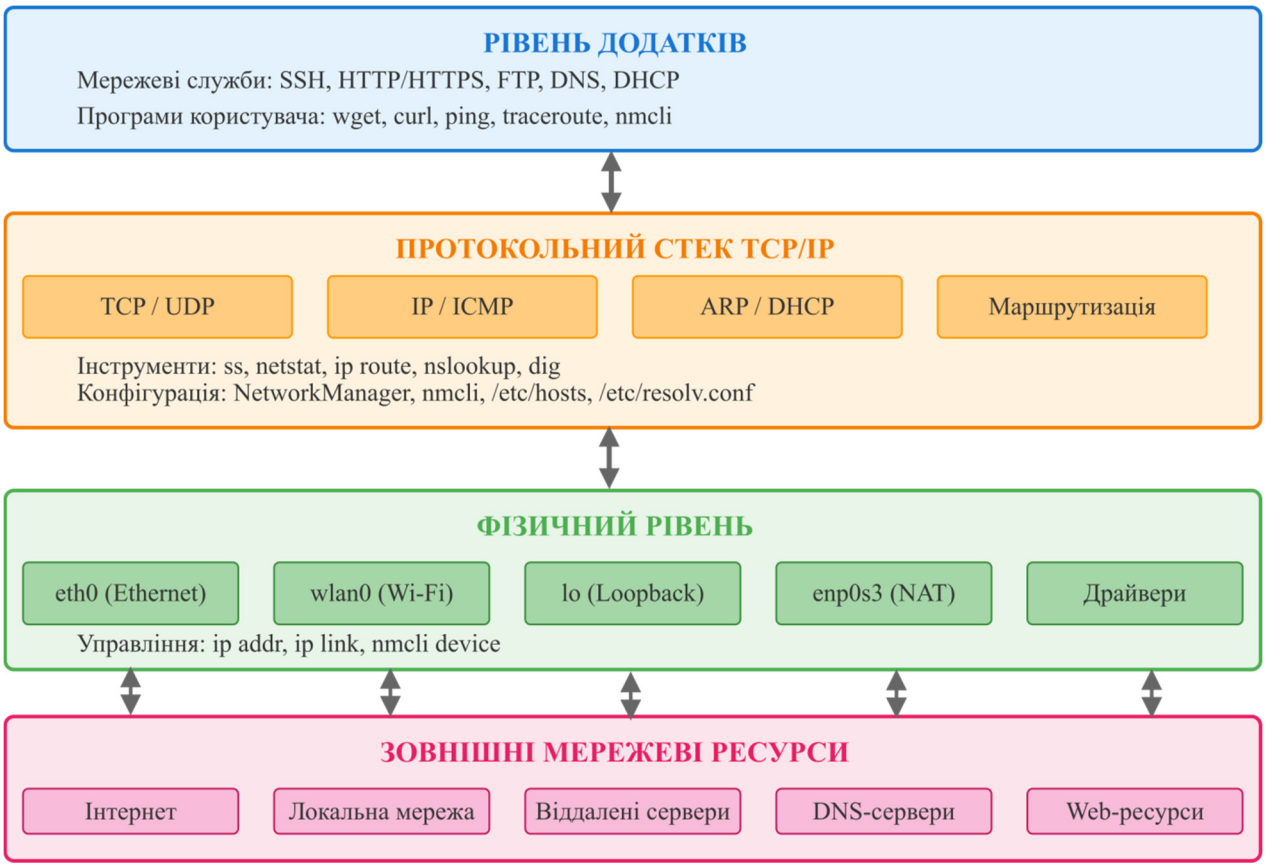

Рисунок 8.1 - Структура мережевої конфігурації Linux

2 КЛЮЧОВІ ПИТАННЯ

1. Які основні переваги використання `NetworkManager` порівняно з традиційним методом налаштування мережі через файл `/etc/network/interfaces` у сучасних дистрибутивах Linux Mint?
2. Як за допомогою команди `nmcli` створити новий профіль підключення зі статичною IP-адресою, шлюзом та DNS-серверами?
3. У чому полягає різниця між командами `ip addr` та `ifconfig`, і чому перша вважається сучасною заміною другої?
4. Які параметри команди `ss` використовуються для відображення всіх активних прослуховуваних TCP та UDP портів у числовому форматі, і як додати інформацію про процеси, що їх використовують?
5. Як перевірити зв'язність з віддаленим хостом (наприклад, 8.8.8.8) та відстежити маршрут, яким проходять пакети до цього хоста?
6. У чому полягає основна відмінність у філософії роботи між утилітами `wget` та `curl`, особливо щодо їхньої поведінки за замовчуванням при завантаженні файлів?
7. Які кроки необхідно виконати для ручного налаштування другого мережевого інтерфейсу (наприклад, `enp0s8`) зі статичною IP-адресою за допомогою команд `ip addr` та `ip link`?
8. Що таке мостовий (Bridged) режим у VirtualBox, і в чому його основна відмінність від режиму NAT?
9. Як за допомогою утиліт `nslookup` або `dig` виконати запит до DNS для визначення IP-адреси доменного імені (наприклад, `google.com`)?
10. Як переглянути поточну таблицю маршрутизації у системі Linux, і що означає маршрут за замовчуванням (default route)?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1 Аналіз мережевих інтерфейсів

Розпочинаємо лабораторну роботу з комплексного аналізу мережевої конфігурації системи. Цей етап є фундаментальним для розуміння поточного стану мережевих підключень та їх параметрів.

Використайте команду ip addr для отримання повної інформації про всі наявні мережеві інтерфейси в системі. Ця сучасна утиліта замінила застарілу команду ifconfig та надає більш детальну і структуровану інформацію про мережеву конфігурацію.

Команда відобразить список усіх інтерфейсів разом із їхніми поточними статусами (активний/неактивний) та призначеними IP-адресами. Особливу увагу приділіть ідентифікації основного активного інтерфейсу, який зазвичай має назву enp0s3, eth0 або подібну.

Для поглибленого дослідження конкретного інтерфейсу застосуйте команду ip addr show dev [назва_інтерфейсу], наприклад ip addr show dev enp0s3.

Це дозволить отримати детальну інформацію про конфігурацію саме цього інтерфейсу, включаючи MAC-адресу, MTU та інші технічні параметри.

3.2 Тестування мережевої зв'язності

Після аналізу інтерфейсів переходимо до перевірки фактичної працездатності мережевого з'єднання. Цей етап допоможе виявити можливі проблеми на різних рівнях мережевого стеку.

Розпочніть з базової перевірки з'єднання на рівні IP-протоколу. Виконайте команду ping -c 4 8.8.8.8 для тестування доступності публічного DNS-сервера Google. Використання параметра -c 4 обмежує кількість відправлених пакетів до чотирьох, що забезпечує швидке отримання результату без непотрібного навантаження на мережу.

Успішне виконання цієї команди свідчить про наявність фізичного підключення та правильну конфігурацію базових мережевих параметрів. Звертайте увагу на час відгуку (RTT) та відсоток втрачених пакетів.

Наступним кроком перевірте функціонування системи доменних імен. Виконайте команду ping -c 4 google.com для тестування можливості розв'язання доменних імен у IP-адреси. Це критично важливо для нормальної роботи більшості мережевих додатків.

Якщо перша команда працює, а друга не виконується, це вказує на проблеми з конфігурацією DNS-серверів. У разі успішного виконання обох команд можна стверджувати про повну функціональність мережевого підключення.

3.3 Аналіз мережевих з'єднань та портів

Третім етапом проводимо детальне дослідження активних мережевих з'єднань та служб, що працюють на системі. Цей аналіз допоможе зрозуміти, які програми та служби використовують мережеві ресурси.

Для початкового огляду відкритих портів використовуйте сучасну утиліту ss -tuln. Ця команда є рекомендованою заміною застарілої netstat та надає більш швидку і точну інформацію про стан мережевих з'єднань.

Параметри команди мають конкретне призначення: -t відображає TCP-з'єднання, -u показує UDP-сокети, -l фільтрує лише прослуховувані (listening) порти, а -n представляє всі адреси та порти у числовому форматі замість символічних імен.

Для отримання розширеної інформації про процеси, що використовують мережеві ресурси, застосуйте команду ss -tulnp. Додатковий параметр -p відображає ідентифікатори процесів (PID) та назви відповідних програм, що дозволяє точно визначити, яка служба відповідає за конкретний порт.

Під час аналізу результатів зверніть увагу на стандартні системні служби (SSH на порту 22, HTTP на 80, HTTPS на 443) та будь-які незнайомі або підозрілі з'єднання. Це важливо з точки зору безпеки системи.

3.4 Завантаження файлів з мережевих ресурсів

Четвертий етап присвячений практичному освоєнню інструментів для роботи з віддаленими файлами та веб-ресурсами. Ці навички є критично важливими для адміністрування систем та автоматизації процесів.

Розпочніть експерименти з утилітою wget, яка спеціально призначена для завантаження файлів. Команда wget https://example.com/testfile.txt завантажить вказаний файл у поточну робочу директорію, зберігаючи оригінальне ім'я файлу.

Для більшого контролю над процесом завантаження використайте команду wget -O newname.txt https://example.com/testfile.txt. Параметр -O дозволяє вказати бажане ім'я для збереженого файлу, що корисно при автоматизації процесів або організації файлової структури.

Далі протестуйте альтернативний інструмент - утиліту curl, яка має дещо іншу філософію роботи. За замовчуванням curl виводить вміст файлу безпосередньо у термінал, але може також зберігати файли на диск.

Команда curl -O https://example.com/testfile.txt завантажить файл, зберігаючи його під оригінальним ім'ям, подібно до wget. Для збереження під іншим ім'ям використайте curl -o customname.txt https://example.com/testfile.txt.

Порівняйте поведінку обох утиліт: wget за замовчуванням зберігає файли, а curl виводить їх вміст. Це робить curl більш універсальним для роботи з API та веб-сервісами, тоді як wget оптимізований саме для завантаження файлів.

3.5 Налаштування додаткового мережевого адаптера

П'ятий етап роботи присвячений практичному налаштуванню множинних мережевих інтерфейсів. Ця навичка є важливою для створення складних мережевих конфігурацій та розуміння принципів роботи віртуального мережевого обладнання.

Перш за все, зупиніть віртуальну машину та увімкніть другий мережевий адаптер через інтерфейс VirtualBox. У налаштуваннях машини перейдіть до розділу "Мережа" та активуйте "Адаптер 2". Оберіть відповідний тип підключення залежно від ваших цілей експерименту.

Після перезавантаження системи новий інтерфейс з'явиться у списку доступних, але може бути неактивним. Перевірте його наявність командою ip addr - зазвичай він матиме назву enp0s8 або подібну.

Спочатку спробуйте автоматичне налаштування через протокол DHCP. Виконайте команду sudo dhclient -v enp0s8, замінивши назву інтерфейсу на актуальну для вашої системи. Параметр -v увімкне детальний вивід процесу отримання конфігурації від DHCP-сервера.

Якщо DHCP недоступний або ви хочете налаштувати статичну конфігурацію, використайте команду sudo ip addr add 192.168.56.10/24 dev enp0s8. Ця команда призначить статичну IP-адресу з відповідною маскою підмережі.

Не забудьте активувати інтерфейс командою sudo ip link set enp0s8 up, якщо він залишається неактивним. Перевірте результат налаштування за

допомогою ip addr show та переконайтеся, що новий інтерфейс з'явився у списку з коректними параметрами.

3.6 Комплексна мережева діагностика

Завершальний етап лабораторної роботи передбачає проведення всебічної діагностики мережевого середовища. Ці інструменти є незамінними для розуміння топології мережі та виявлення потенційних проблем.

Розпочніть з трасування маршруту пакетів до віддаленого хоста. Використайте команду traceroute google.com або її сучасну альтернативу tracepath google.com. Ці утиліти послідовно відображають усі проміжні мережеві пристрої (маршрутизатори, шлюзи), через які проходять пакети на шляху до цілі.

Результат трасування допоможе зрозуміти структуру мережі, виявити вузькі місця або точки відмови. Зверніть увагу на час відгуку кожного проміжного вузла та можливі втрати пакетів на певних ділянках маршруту.

Наступним кроком проведіть детальний аналіз роботи системи доменних імен. Команда nslookup google.com надасть базову інформацію про DNS-записи, включаючи IP-адреси та сервери імен. Для більш детального аналізу використайте сучасну утиліту dig google.com, яка надає розширену інформацію про DNS-запити та відповіді.

Завершіть діагностику вивченням локальної таблиці маршрутизації за допомогою команди ip route show. Ця таблиця містить правила, що визначають, куди система направляє трафік для різних мережевих призначень.

Проаналізуйте отримані дані комплексно: як працює розв'язання імен, який шлях проходять пакети, як налаштована локальна маршрутизація. Це дасть повне розуміння мережевої конфігурації та її особливостей.

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов’язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім’я та по батькові студента, назва академічної групи, а також прізвище, ім’я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам’яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов’язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації ІТ-процесів.

Лабораторна робота № 9

Тема: Управління процесами та службами в Linux

Мета роботи: Навчитися моніторити процеси, керувати службами через systemd, аналізувати навантаження системи та налаштовувати автозавантаження сервісів.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

Ефективне адміністрування Linux-систем неможливе без глибокого розуміння механізмів управління процесами та системними службами. Сучасні багатозадачні операційні системи одночасно виконують десятки або сотні різних програм, кожна з яких потребує системних ресурсів та координації з іншими компонентами системи. Правильне управління цими компонентами забезпечує стабільну роботу системи, оптимальне використання ресурсів та надійність критично важливих сервісів. Адміністратор повинен володіти навичками моніторингу стану системи, діагностики проблем та оперативного втручання у випадку збоїв чи неочікуваної поведінки.

Лабораторне завдання виконується в операційній системі Linux Mint, яка базується на дистрибутивах Ubuntu та Debian і успадковує їхні основні характеристики управління системою. Linux Mint використовує systemd як основний ініціалізатор системи та менеджер служб, що є стандартом для більшості сучасних дистрибутивів Linux, включаючи Ubuntu (починаючи з версії 15.04), Debian (з версії 8), CentOS, Red Hat Enterprise Linux та Fedora. Однак важливо розуміти, що в різних дистрибутивах Linux можуть використовуватися альтернативні системи ініціалізації: деякі дистрибутиви все ще підтримують SysV init, інші використовують OpenRC (Alpine Linux) або runit (Void Linux). Особливості Linux Mint полягають у тому, що система повністю сумісна з пакетами Ubuntu та використовує APT як менеджер пакетів, systemd для управління службами та стандартні утиліти GNU/Linux для роботи з процесами.

Розуміння відмінностей між процесами та службами є ключовим аспектом системного адміністрування, оскільки ці поняття часто плутають, хоча вони мають різне призначення та методи управління. Процес у Linux - це екземпляр виконуваної програми, який існує в оперативній пам'яті та має власний простір адресації, набір ресурсів та унікальний ідентифікатор PID. Процеси можуть бути короткочасними (наприклад, команди ls або cp) або довготривалими (як текстові редактори або веб-браузери), і вони безпосередньо створюються користувачем або іншими програмами. Служба (сервіс) - це спеціальний тип програми, призначений для тривалої роботи в фоновому режимі та надання певних функцій іншим програмам або користувачам системи. Служби зазвичай запускаються автоматично при завантаженні системи, працюють без взаємодії з користувачем та забезпечують критично важливі функції, такі як веб-сервери, бази даних або мережні сервіси.

Операційна система Linux базується на концепції процесів, де кожна запущена програма або системна задача представлена окремим процесом з унікальним ідентифікатором PID (Process ID). Процеси в Linux мають ієрархічну структуру, де кожен процес може мати батьківський процес та дочірні процеси, що дозволяє системі ефективно керувати ресурсами та забезпечувати стабільність роботи. Кожен процес характеризується набором атрибутів: ідентифікатором користувача, який його запустив, пріоритетом виконання, обсягом використовуваної оперативної пам'яті та процесорного часу. Система автоматично призначає PID кожному новому процесу, починаючи з 1 для ініціалізатора системи, і ці ідентифікатори використовуються для всіх операцій управління процесами. Розуміння процесної моделі є фундаментальним для ефективного адміністрування Linux-систем.

Моніторинг процесів здійснюється за допомогою спеціалізованих утиліт командного рядка, основні з яких представлені в таблиці 9.1.

Таблиця 9.1 - Основні команди для адміністрування процесів та служб Linux

Група команд	Команда	Опис
Моніторинг процесів	ps aux	Відображення всіх запущених процесів
	ps -ef	Альтернативний формат відображення процесів
	top	Інтерактивний моніторинг процесів у реальному часі
	htop	Розширений інтерактивний моніторинг процесів
	pgrep назва	Пошук PID процесу за назвою
	pidof назва	Отримання PID активного процесу
Управління процесами	kill PID	Завершення процесу за ідентифікатором
	kill -9 PID	Примусове завершення процесу
	killall назва	Завершення всіх процесів з певною назвою
	pkill назва	Завершення процесів за шаблоном назви
	nohup команда &	Запуск процесу незалежно від терміналу
Управління службами	systemctl start служба	Запуск системної служби
	systemctl stop служба	Зупинка системної служби
	systemctl restart служба	Перезапуск системної служби
	systemctl reload служба	Перезавантаження конфігурації служби
	systemctl status служба	Перевірка стану служби
Налаштування автозавантаження	systemctl enable служба	Увімкнення автозапуску служби

	systemctl disable служба	Вимкнення автозапуску служби
	systemctl is-enabled служба	Перевірка стану автозапуску
	systemctl is-active служба	Перевірка активності служби
Моніторинг системи	journalctl -u служба	Перегляд журналів служби
	journalctl -f	Моніторинг журналів у реальному часі
	systemctl list-units	Список всіх активних юнітів
	systemctl list-unit-files	Список всіх доступних юнітів

Команда ps надає статичний знімок поточного стану процесів у системі, причому опція aux відображає детальну інформацію про всі процеси всіх користувачів, включаючи використання CPU та пам'яті. Для динамічного спостереження за процесами використовується команда top, яка оновлює інформацію в реальному часі та дозволяє інтерактивно сортувати процеси за різними критеріями. Більш функціональною альтернативою є htop, що забезпечує кольорове відображення інформації, більш зручний інтерфейс та розширені можливості фільтрації процесів.

Завершення процесів у Linux може відбуватися кількома способами залежно від конкретної ситуації та вимог адміністратора. Команда kill використовується для надсилання сигналів конкретному процесу за його PID, причому за замовчуванням надсилається сигнал TERM, який дозволяє процесу коректно завершити роботу. У випадках, коли процес не реагує на стандартний сигнал завершення, можна використовувати сигнал KILL за допомогою команди kill -9 PID для примусового завершення. Команда killall надає альтернативний підхід, дозволяючи завершити всі процеси з певним ім'ям програми одночасно, що особливо корисно при роботі з багатопроцесними додатками. Важливо розуміти відмінності між цими підходами для безпечного та ефективного управління системними ресурсами.

Сучасні дистрибутиви Linux використовують systemd як основний ініціалізатор системи та менеджер сервісів, який замінив традиційні System V init скрипти. Systemd забезпечує централізоване управління системними службами, підтримує паралельне завантаження сервісів для прискорення процесу ініціалізації системи та надає розширені можливості моніторингу стану служб. Архітектура systemd базується на концепції юнітів (units), де кожен сервіс описується окремим файлом конфігурації з розширенням .service. Цей підхід забезпечує стандартизований інтерфейс для управління різноманітними системними компонентами та спрощує процедури налаштування й підтримки серверних додатків.

Основним інструментом для взаємодії з systemd є команда systemctl, яка надає повний контроль над життєвим циклом системних служб. Команди start, stop та restart забезпечують базове управління життєвим циклом служб, тоді як команда reload дозволяє перезавантажити конфігурацію служби без її повної зупинки. Статус служби можна перевірити командою status, яка показує

поточний стан, основні параметри роботи та останні записи з журналу подій. Важливо розрізняти поняття активності служби (чи працює вона зараз) та її налаштування автозавантаження (чи буде вона запускатися при старті системи).

Автозавантаження служб налаштовується через команди enable та disable, які не впливають на поточний стан служби, а лише визначають її поведінку при наступному завантаженні системи. Команда enable створює символічні посилання в відповідних директоріях systemd, що дозволяє службі автоматично запускатися на певному рівні виконання системи. Навпаки, disable видаляє ці посилання, запобігаючи автоматичному запуску служби. Для перевірки налаштувань використовуються команди is-enabled та is-active, які повертають поточний стан автозавантаження та активності служби відповідно.

Журналювання подій у systemd здійснюється через службу journald, яка централізовано збирає та зберігає лог-файли всіх системних компонентів. Команда journalctl надає потужні можливості для перегляду та фільтрації журналів за різними критеріями, включаючи конкретні служби, часові інтервали та рівні важливості повідомлень. Опція -u дозволяє переглядати журнали конкретної служби, тоді як -f забезпечує моніторинг нових записів у реальному часі. Аналіз журналів є критично важливим для діагностики проблем, моніторингу безпеки системи та оптимізації продуктивності серверних додатків.

Управління програмним забезпеченням в Linux Mint здійснюється через систему пакетів APT (Advanced Package Tool), успадковану від Debian та Ubuntu. Встановлення нового програмного забезпечення відбувається через команду apt install, яка автоматично вирішує залежності та завантажує необхідні пакети з офіційних репозиторіїв. Перед встановленням рекомендується оновити локальну базу даних пакетів командою apt update, щоб отримати актуальну інформацію про доступні версії програм. Видалення програм здійснюється командою apt remove для простого видалення або apt purge для повного видалення разом з конфігураційними файлами. Система APT забезпечує безпечне та надійне управління програмним забезпеченням, автоматично перевіряючи цифрові підписи пакетів та відстежуючи їх залежності. Таблиця 1.2 демонструє основні команди для управління пакетами в Linux Mint.

Встановлення веб-сервера apache2, який використовується в лабораторному завданні, демонструє типовий процес інсталяції системної служби через APT. Команда sudo apt install apache2 не лише завантажує та встановлює основні файли веб-сервера, але й автоматично налаштовує його як системну службу, створює необхідні конфігураційні файли та призначає відповідні права доступу. Після встановлення служба apache2 стає доступною для управління через systemctl, що демонструє тісну інтеграцію між системою управління пакетами та менеджером служб systemd.

Таблиця 9.2 - Команди управління пакетами в Linux Mint

Операція	Команда	Опис
Оновлення системи	sudo apt update	Оновлення локальної бази даних пакетів
	sudo apt upgrade	Оновлення встановлених пакетів
	sudo apt full-upgrade	Повне оновлення з вирішенням конфліктів
Встановлення пакетів	sudo apt install пакет	Встановлення програми
	sudo apt install пакет1 пакет2	Встановлення кількох пакетів
	sudo apt install --reinstall пакет	Перевстановлення пакету
Видалення пакетів	sudo apt remove пакет	Видалення програми
	sudo apt purge пакет	Повне видалення з конфігураціями
	sudo apt autoremove	Видалення непотрібних залежностей
Пошук та інформація	apt search ключове_слово	Пошук пакетів за назвою
	apt show пакет	Детальна інформація про пакет
	apt list --installed	Список встановлених пакетів
	apt list --upgradable	Список пакетів для оновлення

Ефективний моніторинг системних ресурсів допомагає виявляти вузькі місця продуктивності та попереджати потенційні проблеми з стабільністю системи. Утиліти top та htop відображають інформацію про використання процесора, оперативної пам'яті, процеси з найвищим навантаженням та загальну статистику системи. Ці інструменти дозволяють адміністратору в реальному часі спостерігати за впливом різних служб на продуктивність системи та приймати обґрунтовані рішення щодо оптимізації конфігурації. Регулярний моніторинг ресурсів особливо важливий для серверних систем, де стабільність роботи критично впливає на доступність сервісів для користувачів.

2 КЛЮЧОВІ ПИТАННЯ

1. Що таке процес у Linux і чим він відрізняється від служби за основними характеристиками та призначенням?
2. У чому полягають відмінності між командами ps aux, top та htop для моніторингу процесів?
3. Коли і як використовувати команди kill, kill -9, killall та pkill для завершення процесів?
4. Що таке systemd і які переваги він має порівняно з традиційними System V init скриптами?
5. Які команди systemctl використовуються для управління життєвим циклом системних служб?
6. У чому різниця між поточною активністю служби та налаштуванням її автозавантаження?
7. Як здійснюється встановлення, оновлення та видалення програмного забезпечення через APT в Linux Mint?

8. Як використовувати journalctl для перегляду та аналізу журналів роботи системних служб?

9. Навіщо потрібно оновлювати базу даних пакетів командою apt update перед встановленням нового ПЗ?

10. Як визначити вплив запущених служб на використання CPU та оперативної пам'яті системи?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1. Моніторинг і керування процесами

3.1.1. Увімкніть віртуальну машину з Linux Mint та відкрийте термінал.

3.1.2. За допомогою команд ps aux, top та htop виведіть список усіх запущених процесів. Проаналізуйте отримані дані, звертаючи увагу на PID, користувача, використання CPU та пам’яті.

3.1.3. Запустіть тестову програму (наприклад, gedit) у фоновому режимі:

```
gedit &
```

3.1.4. Визначте PID запущеного процесу за допомогою команд:

```
ps aux | grep gedit
або
pgrep gedit
```

3.1.5. Примусово завершіть процес за його PID:

```
kill <PID>
```

3.1.6. Альтернативно, завершіть процес за ім’ям:

```
killall gedit
```

3.1.7. Поясніть у протоколі різницю між командами kill і killall.

3.2. Керування системними службами через systemd

3.2.1. Встановіть веб-сервер apache2:

```
sudo apt update && sudo apt install apache2 -y
```

3.2.2. Запустіть службу apache2:

```
sudo systemctl start apache2
```

3.2.3. Перевірте, чи працює служба:

```
sudo systemctl status apache2
```

3.2.4. Зупиніть службу:

```
sudo systemctl stop apache2
```

3.2.5. Перезапустіть службу:

```
sudo systemctl restart apache2
```

3.2.6. Переконайтеся, що служба працює, ще раз перевіривши її статус.

3.3. Налаштування автозавантаження та аналіз стану служб

3.3.1. Налаштуйте автоматичне запускання служби apache2 при завантаженні системи:

```
sudo systemctl enable apache2
```

3.3.2. Перевірте, чи ввімкнено автозавантаження:

```
sudo systemctl is-enabled apache2
```

3.3.3. Перевірте, чи активна служба в даний момент:

```
sudo systemctl is-active apache2
```

3.3.4. Якщо потрібно, вимкніть автозавантаження для тестування:

```
sudo systemctl disable apache2
```

3.4. Аналіз журналів і навантаження системи

3.4.1. Перегляньте журнали роботи служби apache2 за допомогою:

```
sudo journalctl -u apache2
```

3.4.2. Спостерігайте за журналами в реальному часі (за бажанням):

```
sudo journalctl -u apache2 -f
```

3.4.3. Використовуючи top або htop, проаналізуйте навантаження на процесор і використання оперативної пам’яті до та після запуску служби apache2.

3.4.4. Зробіть висновки щодо впливу служби на ресурси системи.

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов’язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім’я та по батькові студента, назва академічної групи, а також прізвище, ім’я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам’яті, дисковий простір), а також стану

системи на момент початку виконання завдання. Далі послідовно описується виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов’язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

Лабораторна робота № 10

Тема: Розробка bash-скриптів для автоматизації типових завдань в ОС Linux

Мета роботи: Набути практичних навичок створення та виконання bash-скриптів для автоматизації типових завдань в ОС Linux.

1 КЛЮЧОВІ ПОЛОЖЕННЯ

1.1 Вступ до автоматизації в Linux

Автоматизація процесів є фундаментальною складовою ефективного адміністрування систем на базі Linux. Операційні системи сімейства Unix, до яких належить Linux, з самого початку проектувалися з урахуванням принципів автоматизації та скриптування. Bash (Bourne Again Shell) виступає основним інтерпретатором команд у більшості дистрибутивів Linux і надає потужні можливості для створення скриптів автоматизації. Написання bash-скриптів дозволяє системним адміністраторам та користувачам автоматизувати рутинні завдання, зменшувати кількість помилок, пов'язаних з людським фактором, та підвищувати продуктивність роботи. Скрипти можуть виконувати широкий спектр завдань: від простого копіювання файлів до складного моніторингу системи та обробки великих обсягів даних.

Bash-скрипти представляють собою текстові файли, що містять послідовність команд, які виконуються автоматично без втручання користувача. Вони дозволяють об'єднувати функціональність різних утиліт командного рядка, створюючи комплексні рішення для специфічних завдань. Можливість параметризації скриптів через змінні та аргументи командного рядка робить їх гнучкими та придатними для багаторазового використання в різних контекстах.

1.2 Основні елементи bash-скриптів

Для розуміння структури та можливостей bash-скриптів розглянемо основні команди та конструкції, представлені в таблиці 10.1.

Таблиця 10.1 - Основні команди та директиви bash-скриптів

Категорія	Команда/ Конструкція	Призначення	Синтаксис	Приклад
Структура скрипта	`#!/bin/bash`	Шебанг для визначення інтерпретатора	`#!/bin/bash`	`#!/bin/bash`
	`# коментар`	Коментування коду	`# текст коментаря`	`# Початок обробки файлів`
Змінні	Оголошення змінної	Присвоєння значення	`name=value`	`count=10`
	Використання змінної	Отримання значення	`$name` або `${name}`	`echo $count`

	Аргументи скрипта	Параметри командного рядка	`$1, $2, $@, $#`	`filename=$1`
	Підстановка команд	Виконання команди у змінній	`var=$(command)`	`date_now=$(date)`
Введення/виведення	`echo`	Виведення тексту	`echo "text"`	`echo "Привіт $name"`
	`printf`	Форматоване виведення	`printf "format" args`	`printf "%s\n" "$text"`
	`read`	Читання введення користувача	`read variable`	`read username`
	`read -p`	Читання з підказкою	`read -p "prompt" var`	`read -p "Ім'я: " name`
Перенаправлення	`>`	Запис у файл (перезапис)	`command > file`	`echo "text" > file.txt`
	`>>`	Додавання до файлу	`command >> file`	`date >> log.txt`
	`2>/dev/null`	Приховування помилок	`command 2>/dev/null`	`ls *.txt 2>/dev/null`
	`|`	Конвеєр (pipe)	`cmd1 | cmd2`	`cat file.txt | grep "word"`
Умовні оператори	`if-then-fi`	Умовна конструкція	`if [ test ]; then cmd; fi`	`if [ -f file ]; then echo "OK"; fi`
	`if-then-else-fi`	Умова з альтернативою	`if [ test ]; then cmd1; else cmd2; fi`	`if [ -s file ]; then cat file; else echo "Empty"; fi`
	Тест файлу	Перевірка існування	`[ -f file ]`	`if [ -f data.txt ]`
	Тест каталогу	Перевірка каталогу	`[ -d directory ]`	`if [ -d backup ]`
	Тест порожності	Перевірка непорожності	`[ -s file ]`	`if [ -s log.txt ]`
Цикли	`for-do-done`	Цикл по списку	`for var in list; do cmd; done`	`for file in *.txt; do echo $file; done`
	`while-do-done`	Цикл за умовою	`while [ test ]; do cmd; done`	`while read line; do echo $line; done`
	`break`	Вихід з циклу	`break`	`if [ "$line" = "exit" ]; then break; fi`
	`continue`	Перехід до наступної ітерації	`continue`	`if [ -z "$line" ]; then continue; fi`

1.3 Структура та ініціалізація скриптів

Кожен bash-скрипт повинен починатися з рядка шебангу #!/bin/bash, який інформує операційну систему про те, який інтерпретатор використовувати для виконання файлу. Цей рядок має розташовуватися на самому початку файлу без пробілів перед символом решітки. Шебанг є критично важливим елементом, оскільки без нього система може спробувати виконати скрипт за допомогою неправильного інтерпретатора або взагалі відмовитися його запускати. Після шебангу зазвичай розміщуються коментарі з описом призначення скрипта, автора та дати створення. Коментарі починаються з символу # і ігноруються інтерпретатором, але є цінними для документування коду та полегшення його розуміння іншими розробниками. Приклад правильної структури початку скрипта:

```
#!/bin/bash
# Скрипт для підрахунку файлів
# Автор: Студент групи IT-21
# Дата: 2025-08-09

echo "Початок роботи скрипта"
```

Після створення файлу скрипта необхідно надати йому права на виконання за допомогою команди chmod +x назва_скрипта.sh. Запуск скрипта здійснюється вказанням його шляху: ./script.sh для запуску з поточного каталогу або повним шляхом для запуску з будь-якого місця файлової системи. Типовий цикл створення та запуску виглядає наступним чином:

```
# Створення скрипта
nano my_script.sh
# Надання прав
chmod +x my_script.sh
# Запуск
./my_script.sh
```

1.4 Робота зі змінними та параметрами

Змінні в bash є основним механізмом зберігання та маніпулювання даними під час виконання скрипта. Оголошення змінної відбувається простим присвоєнням у форматі name=value без пробілів навколо знаку рівності. Використання змінної здійснюється через префікс долара $name або у фігурних дужках ${name} для явного визначення меж імені змінної. Фігурні дужки особливо корисні при конкатенації змінних з текстом або при використанні параметрів підстановки. Змінні можуть містити рядки, числа або результати виконання команд через конструкцію підстановки команд $(command). Наприклад, current_time=$(date '+%H:%M') збереже поточний час у форматі година:хвилина. Типові операції зі змінними включають:

```
# Оголошення різних типів змінних
username="student"
count=0
```

```
filename="data.txt"
current_date=$(date '+%Y-%m-%d')

# Використання змінних
echo "Користувач: $username"
echo "Файл: ${filename}.backup"
new_count=$((count + 1))
```

Скрипти можуть отримувати параметри з командного рядка через позиційні змінні $1, $2, $3 тощо, де число відповідає порядковому номеру аргументу. Змінна $@ містить усі передані аргументи як окремі елементи, $* об'єднує їх в один рядок, а $# повертає кількість переданих аргументів. Це дозволяє створювати універсальні скрипти, поведінка яких змінюється залежно від параметрів запуску. Приклад роботи з аргументами:

```
#!/bin/bash
# Обробка аргументів командного рядка
source_file=$1
destination=$2

echo "Кількість аргументів: $#"
echo "Джерело: $source_file"
echo "Призначення: $destination"
echo "Всі аргументи: $@"
```

1.5 Введення та виведення даних

Система введення-виведення в bash-скриптах базується на стандартних потоках Unix: stdin (введення), stdout (виведення) та stderr (помилки). Команда echo є найпростішим способом виведення тексту на екран та може працювати як з літеральними рядками, так і зі змінними. Команда printf надає більш гнучкі можливості форматування, подібно до функції printf у мові C, дозволяючи контролювати ширину полів, вирівнювання та спеціальні символи. Для зчитування даних від користувача використовується команда read, яка може працювати з підказками через параметр -p та зберігати введення у вказаній змінній. Основні прийоми роботи з введенням-виведенням:

```
# Виведення тексту та змінних
echo "Простий текст"
echo "Значення змінної: $count"
printf "Форматований вивід: %s має %d файлів\n" "$user"
"$file_count"

# Читання введення користувача
read username
read -p "Введіть назву файлу: " filename
echo "Ви ввели: $filename"
```

Перенаправлення виводу реалізується операторами > для перезапису файлу та >> для додавання до існуючого файлу. Оператор pipe | створює конвеєр, передаючи вивід однієї команди як ввід для наступної, що дозволяє будувати

складні ланцюжки обробки даних. Приховування повідомлень про помилки здійснюється перенаправленням stderr до /dev/null за допомогою конструкції 2>/dev/null. Практичні приклади перенаправлення:

```
# Запис у файл та додавання
echo "Новий запис" > log.txt
echo "Додатковий запис" >> log.txt

# Конвеєр команд
cat names.txt | grep "John" | wc -l

# Приховування помилок
ls *.backup 2>/dev/null || echo "Файлів backup не знайдено"
```

1.6 Умовні конструкції та логічні перевірки

Умовні оператори в bash дозволяють створювати логіку прийняття рішень у скриптах на основі різноманітних умов. Базовий синтаксис if [умова]; then команди; fi забезпечує виконання блоку команд лише при виконанні заданої умови. Квадратні дужки навколо умови є обов'язковими, при цьому необхідно дотримуватися пробілів між дужками та умовою. Конструкція може бути розширена за допомогою else для альтернативної гілки виконання та elif для додаткових перевірок. Тести для файлів включають -f для перевірки існування звичайного файлу, -d для каталогу, -s для непорожнього файлу, -r, -w, -x для перевірки прав читання, запису та виконання відповідно. Приклади типових перевірок:

```
# Перевірка існування файлу
if [ -f "$filename" ]; then
    echo "Файл $filename існує"
    cat "$filename"
else
    echo "Файл не знайдено, створюю новий"
    touch "$filename"
fi

# Перевірка порожності файлу
if [ -s "data.txt" ]; then
    echo "Файл містить дані"
    head -1 data.txt
else
    echo "Файл порожній або не існує"
fi

# Множинні умови
if [ -d "backup" ] && [ -f "source.txt" ]; then
    cp source.txt backup/
    echo "Файл скопійовано у backup"
fi
```

Порівняння рядків здійснюється операторами = та !=, а числові порівняння використовують -eq, -ne, -lt, -le, -gt, -ge. Логічні оператори && (І) та || (АБО) дозволяють комбінувати кілька умов в одній перевірці. Конструкція elif дозволяє створювати складні розгалуження логіки:

```
# Числові порівняння
if [ $count -eq 0 ]; then
    echo "Лічильник дорівнює нулю"
elif [ $count -lt 10 ]; then
    echo "Лічильник менше 10"
else
    echo "Лічильник більше або дорівнює 10"
fi
```

1.7 Цикли та ітеративна обробка

Цикли забезпечують можливість багаторазового виконання блоків команд, що є основою автоматизації повторюваних завдань. Цикл for використовується для ітерації по заданому списку значень, файлів або результатів команд. Синтаксис for змінна in список; do команди; done дозволяє обробляти кожен елемент списку окремо. Найчастіше цикл for застосовується для обробки файлів за шаблоном, наприклад for file in *.txt, або для ітерації по послідовності чисел. Цикл while виконується доки заданна умова залишаєтьсяістинною, що робить його придатним для читання файлів рядок за рядком або для реалізації інтерактивних меню. Конструкція while read line; do обробка_рядка; done < файл є стандартним способом послідовної обробки вмісту файлу. Практичні приклади використання циклів:

```
# Цикл for для обробки файлів
for file in *.txt; do
    if [ -f "$file" ]; then
        echo "Обробляю файл: $file"
        wc -l "$file"
    fi
done

# Цикл for з числовою послідовністю
for i in {1..5}; do
    echo "Створюю файл doc$i.txt"
    touch "doc$i.txt"
done

# Цикл while для читання файлу
while read line; do
    if [[ "$line" == *"error"* ]]; then
        echo "Знайдено помилку: $line"
    fi
done < log.txt

# Цикл while з лічильником
counter=1
while [ $counter -le 3 ]; do
```

```
    echo "Ітерація $counter"
    counter=$((counter + 1))
done
```

Для контролю виконання циклів використовуються команди break для негайного виходу з циклу та continue для переходу до наступної ітерації, пропускаючи решту команд у поточній ітерації. Ці команди особливо корисні при обробці даних з умовними пропусками або при необхідності дострокового завершення циклу:

```
# Використання break та continue
for file in *; do
    if [ "$file" = "stop.txt" ]; then
        echo "Знайдено файл зупинки"
        break
    fi
    if [ ! -f "$file" ]; then
        continue  # Пропустити каталоги
    fi
    echo "Обробляю: $file"
done
```

1.8 Перенаправлення потоків та конвеєри

Система перенаправлення потоків у bash дозволяє гнучко керувати виводом команд та вводом даних. Оператор > перезаписує цільовий файл повністю, тоді як >> додає дані в кінець існуючого файлу, зберігаючи попередній вміст. Це дозволяє вести журнали подій, накопичувати результати обробки або створювати звіти. Перенаправлення stderr за допомогою 2> або 2>> дозволяє окремо обробляти повідомлення про помилки, а конструкція 2>/dev/null повністю їх приховує. Оператор pipe | створює конвеєр обробки даних, де вивід однієї команди стає вводом для наступної, дозволяючи будувати складні ланцюжки фільтрації та трансформації інформації. Приклади роботи з перенаправленням:

```
# Базове перенаправлення
echo "Початок роботи" > status.log
date >> status.log
echo "Завершення роботи" >> status.log

# Робота з помилками
ls /nonexistent 2> errors.log
ls *.backup 2>/dev/null | wc -l

# Конвеєри обробки
cat data.txt | grep "important" | sort | uniq > filtered.txt
ps aux | grep "bash" | awk '{print $2}' > bash_pids.txt
```

Комбінування перенаправлення з конвеєрами дозволяє створювати потужні інструменти обробки даних. Наприклад, grep "error" logfile.txt | wc -l > error_count.txt знайде всі рядки з помилками у файлі журналу, підрахує їх

кількість та збереже результат у окремий файл. Можливість об'єднання stdout та stderr за допомогою &> або розділення їх на різні файли створює гнучкі механізми логування та діагностики.

Розглянемо декілька практичних прикладів bash-скриптів, які демонструють застосування вивчених концепцій.

Приклад 1. Підрахунок файлів певного типу

Завдання: Створити скрипт, який підраховує кількість .txt файлів у домашньому каталозі і виводить результат на екран та у файл file_count.txt.

```
#!/bin/bash

# Підрахунок txt-файлів у домашньому каталозі
cd ~
count=$(ls *.txt 2>/dev/null | wc -l)

echo "Знайдено $count файлів з розширенням .txt"
echo "Кількість .txt файлів: $count" > file_count.txt

if [ $count -eq 0 ]; then
    echo "У домашньому каталозі немає .txt файлів"
else
    echo "Список файлів:"
    ls *.txt 2>/dev/null
fi
```

Цей скрипт демонструє використання підстановки команд для збереження результату підрахунку у змінній, перенаправлення помилок для уникнення повідомлень про відсутність файлів, умовну перевірку кількості знайдених файлів та запис результату у файл. Конструкція 2>/dev/null забезпечує коректну роботу скрипта навіть при відсутності txt-файлів у каталозі.

Приклад 2. Фільтрація та обробка текстових даних

Завдання: Створити файл names.txt зі списком імен. Скрипт має знайти всі імена, що містять літеру "о" або "О", зберегти їх у filtered_names.txt та вивести статистику.

```
#!/bin/bash

# Перевірка існування вхідного файлу
if [ ! -f "names.txt" ]; then
    echo "Файл names.txt не знайдено. Створюю приклад..."
    echo  -e  "Олексій\nМарія\nВолодимир\nАнна\nПетро\nОксана"  >
names.txt
fi

# Фільтрація імен, що містять "о" або "О"
grep -i "о" names.txt > filtered_names.txt
found_count=$(grep -c -i "о" names.txt)
total_count=$(wc -l < names.txt)
```

```
echo "Загальна кількість імен: $total_count"
echo "Знайдено імен з літерою 'o': $found_count"
echo "Результат збережено у файл filtered_names.txt"

if [ $found_count -gt 0 ]; then
    echo "Знайдені імена:"
    cat filtered_names.txt
fi
```

Скрипт ілюструє перевірку існування файлу з автоматичним створенням тестових даних, використання grep для фільтрації за регулярним виразом, підрахунок збігів та загальної кількості рядків, а також умовне виведення результатів залежно від кількості знайдених збігів.

1.9 Команди операційної системи Linux

Bash-скрипти активно використовують стандартні команди операційної системи Linux для виконання різноманітних операцій з файлами, каталогами та системою. Ці команди, представлені в таблиці 10.2, складають основу функціональності скриптів автоматизації.

Таблиця 10.2 - Основні команди Linux для скриптування

Категорія	Команда	Призначення	Основні параметри	Приклад
Перегляд файлів	`cat`	Виведення вмісту файлу	`-n` (нумерація рядків)	`cat -n data.txt`
	`head`	Початок файлу	`-n` (кількість рядків)	`head -5 log.txt`
	`tail`	Кінець файлу	`-n` (кількість рядків)	`tail -10 error.log`
	`less/more`	Постраничний перегляд		`less longfile.txt`
Аналіз тексту	`wc`	Підрахунок рядків/слів/символів	`-l, -w, -c`	`wc -l names.txt`
	`grep`	Пошук за шаблоном	`-c` (підрахунок), `-i` (ігнорування регістру), `-v` (інверсія)	`grep -c "error" log.txt`
	`sort`	Сортування рядків	`-n` (числове), `-r` (зворотне)	`sort -n numbers.txt`
	`uniq`	Видалення дублікатів	`-c` (підрахунок)	`sort data.txt | uniq`
	`diff`	Порівняння файлів	`-u` (unified формат)	`diff file1.txt file2.txt`
	`tac`	Виведення у зворотному порядку		`tac input.txt`

Операції з файлами	`touch`	Створення файлу/оновлення часу		`touch newfile.txt`
	`cp`	Копіювання	`-r` (рекурсивно)	`cp source.txt backup/`
	`mv`	Переміщення/перейменування		`mv old.txt new.txt`
	`rm`	Видалення	`-f` (примусово), `-r` (рекурсивно)	`rm temp*.txt`
	`ln`	Створення посилань	`-s` (символічне)	`ln -s target.txt link.txt`
Операції з каталогами	`ls`	Список файлів	`-l` (детально), `-a` (приховані), `-h` (розміри)	`ls -la *.txt`
	`mkdir`	Створення каталогу	`-p` (батьківські каталоги)	`mkdir -p backup/logs`
	`rmdir`	Видалення порожнього каталогу		`rmdir empty_dir`
	`pwd`	Поточний каталог		`current_dir=$(pwd)`
	`cd`	Зміна каталогу		`cd /home/user`
Системна інформація	`date`	Дата та час	Формати: `%Y-%m-%d`, `%H:%M:%S`	`date '+%Y-%m-%d %H:%M:%S'`
	`whoami`	Поточний користувач		`user=$(whoami)`
	`hostname`	Ім'я комп'ютера		`host=$(hostname)`
	`ps`	Список процесів	`-aux` (детальна інформація)	`ps aux | grep bash`
Права доступу	`chmod`	Зміна прав доступу	`+x` (додати виконання), `755`, `644`	`chmod +x script.sh`
	`chown`	Зміна власника		`chown user:group file.txt`

1.10 Обробка файлів та текстових даних

Робота з файлами складає основу більшості завдань автоматизації в Linux. Команди cat, head та tail дозволяють переглядати вміст файлів повністю або частково, що корисно для аналізу логів, конфігураційних файлів та даних. Команда wc (word count) підраховує кількість рядків, слів та символів у файлах, надаючи статистичну інформацію про обсяг даних. Параметри -l, -w, -c дозволяють отримувати специфічні метрики, а комбінування з іншими командами через pipe створює потужні інструменти аналізу. Команда grep

реалізує пошук за регулярними виразами, дозволяючи знаходити рядки, що відповідають певним критеріям, підраховувати їх кількість або виконувати інверсний пошук. Параметр -c підраховує збіги без виведення самих рядків, -i ігнорує регістр символів, а -v виводить рядки, що не містять шаблон. Приклади роботи з текстовими файлами:

```
# Аналіз статистики файлу
echo "Статистика файлу $filename:"
echo "Рядків: $(wc -l < $filename)"
echo "Слів: $(wc -w < $filename)"
echo "Символів: $(wc -c < $filename)"

# Пошук та фільтрація
grep -i "linux" text.txt > linux_lines.txt
error_count=$(grep -c "error" log.txt)
echo "Знайдено $error_count помилок"

# Перегляд початку та кінця файлу
echo "Перші 3 рядки:"
head -3 data.txt
echo "Останні 3 рядки:"
tail -3 data.txt
```

Сортування та обробка даних здійснюються командами sort та uniq, які часто використовуються разом для видалення дублікатів зі списків. Команда diff порівнює файли та виводить відмінності, що корисно для аналізу змін у конфігураціях або даних. Команда tac виводить рядки файлу у зворотному порядку, що може бути корисно для аналізу журналів або інвертування послідовностей. Типові операції сортування та унікалізації:

```
# Сортування та видалення дублікатів
sort names.txt | uniq > unique_names.txt
sort -n numbers.txt | tail -1 > max_number.txt

# Порівняння файлів
if ! diff -q file1.txt file2.txt >/dev/null; then
    echo "Файли відрізняються"
    diff file1.txt file2.txt > differences.txt
fi

# Інвертування порядку рядків
tac input.txt > reversed.txt
```

1.11 Робота з файловою системою

Управління файлами та каталогами в bash-скриптах здійснюється стандартними командами Unix. Команда touch створює порожні файли або оновлює час модифікації існуючих, що корисно для ініціалізації робочих файлів або маркування часу останніх змін. Копіювання файлів командою cp може здійснюватися як для окремих файлів, так і для цілих каталогів з параметром -r.

Переміщення та перейменування файлів виконується командою mv, яка може працювати як всередині одного каталогу (перейменування), так і між різними каталогами (переміщення). Видалення файлів командою rm вимагає обережності, особливо з параметром -f, який примусово видаляє файли без підтвердження. Основні операції з файловою системою:

```
# Створення файлів та каталогів
touch data.txt config.txt
mkdir -p backup/logs/$(date '+%Y-%m')

# Копіювання з перевіркою
if [ -f "source.txt" ]; then
    cp source.txt backup/source_$(date '+%Y%m%d').txt
    echo "Файл скопійовано з часовою міткою"
fi

# Переміщення та перейменування
mv old_name.txt new_name.txt
mv *.log logs/

# Безпечне видалення з підтвердженням
read -p "Видалити всі temp_* файли? (y/n): " confirm
if [ "$confirm" = "y" ]; then
    rm temp_*
    echo "Тимчасові файли видалено"
fi
```

Робота з каталогами включає створення за допомогою mkdir, видалення порожніх каталогів командою rmdir та навігацію за допомогою cd і pwd. Параметр -p для mkdir дозволяє створювати вкладені структури каталогів одним викликом. Команда ls з різними параметрами надає інформацію про файли та каталоги, включаючи детальні атрибути, приховані файли та розміри у зручному для читання форматі. Приклади роботи з каталогами:

```
# Створення структури каталогів
mkdir -p project/{src,docs,tests}
cd project/src

# Аналіз вмісту каталогу
file_count=$(ls -1 *.txt 2>/dev/null | wc -l)
echo "У каталозі $file_count txt-файлів"

# Навігація та збереження шляхів
original_dir=$(pwd)
cd /tmp
echo "Тимчасово у $(pwd)"
cd "$original_dir"
```

1.12 Робота з датою, часом та системною інформацією

Команда date є універсальним інструментом для роботи з часовими мітками в скриптах автоматизації. Вона підтримує широкий набір форматів

виведення через специфікатори, починаючи з символу %. Найпоширеніші формати включають %Y для року, %m для місяця, %d для дня, %H для години, %M для хвилини та %S для секунди. Комбінування цих специфікаторів дозволяє створювати часові мітки для логування, іменування файлів резервних копій або планування завдань. Команда може також використовуватися для арифметичних операцій з датами, що корисно для розрахунку періодів або термінів виконання завдань. Практичні приклади роботи з датою та часом:

```
# Різні формати дати
current_date=$(date '+%Y-%m-%d')
timestamp=$(date '+%Y-%m-%d %H:%M:%S')
log_time=$(date '+[%H:%M:%S]')

echo "Сьогодні: $current_date"
echo "$log_time Скрипт запущено" >> script.log

# Створення файлів з часовими мітками
backup_name="backup_$(date '+%Y%m%d_%H%M%S').tar.gz"
touch "report_$(date '+%Y-%m-%d').txt"

# Логування подій
echo "$(date): Початок обробки файлів" >> process.log
```

Отримання системної інформації через команди whoami, hostname та ps дозволяє скриптам адаптуватися до поточного контексту виконання та включати метадані про систему у свої результати. Ці команди особливо корисні для створення звітів, логування дій користувачів або налаштування скриптів під конкретне середовище:

```
# Збір системної інформації
current_user=$(whoami)
machine_name=$(hostname)
script_pid=$

echo "Користувач: $current_user"
echo "Комп'ютер: $machine_name"
echo "PID скрипта: $script_pid"

# Включення метаданих у результати
echo "Звіт створено $current_user на $machine_name $(date)" >
report_header.txt
```

1.13 Управління правами доступу

Права доступу до файлів та каталогів у Linux контролюються командами chmod та chown. Команда chmod змінює права читання, запису та виконання для власника, групи та інших користувачів. Для скриптів критично важливим є право на виконання, яке надається параметром +x. Команда може працювати як з символьною нотацією (chmod u+x script.sh), так і з числовою (chmod 755 script.sh). Команда chown змінює власника та групу файлу, що може бути

необхідно для забезпечення правильного доступу до результатів роботи скрипта. Розуміння системи прав доступу є критично важливим для створення безпечних та функціональних скриптів автоматизації.

```
# Надання прав на виконання
chmod +x my_script.sh

# Встановлення специфічних прав
chmod 755 script.sh  # rwxr-xr-x
chmod 644 data.txt   # rw-r--r--

# Перевірка прав доступу
if [ -x "script.sh" ]; then
    echo "Скрипт може бути виконаний"
    ./script.sh
else
    echo "Скрипт не має прав на виконання"
    chmod +x script.sh
fi
```

1.14 Рекомендації щодо створення bash-скриптів

При створенні bash-скриптів важливо дотримуватися певних принципів та рекомендацій для забезпечення їх надійності, читабельності та безпеки. Завжди починайте скрипт з шебангу #!/bin/bash та додавайте описові коментарі про призначення скрипта, автора та дату створення. Використовуйте змістовні назви змінних та функцій, які відображають їх призначення. Перевіряйте існування файлів та каталогів перед їх використанням, щоб уникнути помилок виконання.

Обробляйте помилки належним чином, використовуючи умовні конструкції та перенаправлення stderr. Уникайте жорстко закодованих шляхів та значень, натомість використовуйте змінні та параметри командного рядка для забезпечення гнучкості скрипта. Тестуйте скрипти в безпечному середовищі перед використанням у продакшні. Документуйте складні частини коду коментарями для полегшення подальшого супроводу та розуміння іншими розробниками.

Дотримання цих принципів забезпечить створення якісних, надійних та зручних для використання bash-скриптів, які ефективно автоматизуватимуть поставлені завдання та сприятимуть підвищенню продуктивності роботи з операційною системою Linux.

2 КЛЮЧОВІ ПИТАННЯ

1. Що таке шебанг і навіщо він потрібен?
2. Як зробити скрипт виконуваним?
3. Як оголосити та використати змінну в bash?
4. Як отримати введення від користувача?
5. Як перевірити наявність файлу в скрипті?
6. Як використовувати if для умовного виконання?

7. Як створити цикл for по файлах у каталозі?
8. Як передати аргументи в скрипт?
9. Як виміряти час виконання скрипту?
10. Як записати вивід команди у файл?

3 ЛАБОРАТОРНЕ ЗАВДАННЯ

3.1. Підготовка віртуального середовища

Перш ніж розпочати виконання лабораторної роботи, запустіть віртуальну машину з операційною системою Linux Mint у середовищі VirtualBox. Після завантаження системи увійдіть до неї під своїм обліковим записом. Це забезпечить доступ до вашого домашнього каталогу та дозволить безпечно виконувати всі необхідні операції. Далі відкрийте термінал — основний інструмент для взаємодії з системою, у якому ви будете створювати файли, писати скрипти, запускати команди та аналізувати результати.

3.2. Створення вхідних файлів

Створіть у домашньому каталозі (~) один або кілька текстових файлів (наприклад, data.txt, names.txt, log.txt тощо), вміст яких відповідатиме умовам вашого індивідуального завдання (див. таблицю варіантів). Наповніть їх прикладними даними — числами, словами, рядками, логами — залежно від поставленого завдання. Ці файли стануть вхідними даними для вашого bash-скрипта.

3.3 Написання bash-скрипта

Напишіть bash-скрипт, який автоматизує виконання вашого індивідуального завдання. Скрипт має починатися з шебангу #!/bin/bash, містити використання змінних, умовних операторів (if), циклів (for, while) за необхідності, а також команди для читання, обробки та запису даних. Виконання скрипта має включати як виведення результату на екран, так і його збереження у вказаний файл.

3.4 Надання прав на виконання

Після створення скрипта надайте йому права на виконання за допомогою команди chmod +x ім’я_вашого_скрипта.sh. Це необхідно для того, щоб система могла виконати файл як програму, а не лише відкрити його для перегляду.

3.5 Запуск скрипта

Виконайте створений скрипт командою ./ім’я_скрипта.sh. Спостерігайте за виводом у терміналі: переконайтеся, що скрипт працює коректно, обробляє дані

згідно з умовою, виводить результат на екран і правильно зберігає його у вихідний файл.

3.6 Перевірка результатів

Проаналізуйте результат виконання скрипта. Переконайтеся, що вивід відповідає очікуваному, дані у файлі записані правильно, а скрипт адекватно реагує на відсутність файлів, порожній вміст або інші можливі помилки. За потреби виправте логічні або синтаксичні помилки та повторіть запуск.

Таблиця 10.3 - Варіанти індивідуальних завдань

№	Завдання
1	Створіть файл numbers.txt з кількома числами по одному на рядок. Скрипт має підрахувати кількість чисел і вивести результат на екран та у файл result.txt.
2	Створіть файл names.txt" зі списком імен. Скрипт має вивести всі імена, що містять літеру ""a"" або ""A"", і зберегти їх у "filtered_names.txt.
3	Скрипт створює три порожні файли: doc1.txt", "doc2.txt", "doc3.txt. Потім об’єднує їх у один файл all_docs.txt і виводить повідомлення про успішне об’єднання.
4	Створіть файл text.txt. Скрипт має використовувати wc" для підрахунку слів, рядків і символів, і зберегти результат у "stats.txt.
5	Скрипт запитує назву файлу (read). Якщо файл існує — виводить його вміст (cat"), якщо ні — створює його з текстом ""Файл створено автоматично"". Результат виводиться на екран."
6	Створіть файл data.txt" з кількома рядками. Скрипт має підрахувати, скільки рядків містять слово ""Linux"" ("grep -c"), і вивести результат на екран та у "found.txt.
7	Скрипт підраховує кількість .txt файлів у домашньому каталозі (ls *.txt 2>/dev/null | wc -l) і виводить список. Результат зберігається у file_list.txt.
8	Створіть два файли: file1.txt і file2.txt. Скрипт об’єднує їх у merged.txt", додаючи між ними роздільник "=== Роздільник ===.
9	Скрипт створює файл timestamp.txt із поточною датою і часом (date '+%Y-%m-%d %H:%M:%S') і виводить його вміст на екран.
10	Створіть файл log.txt. Скрипт додає до нього новий запис: "echo ""Подія: скрипт запущено o $(date)"" >> log.txt".
11	Скрипт запитує від користувача три рядки (read) і записує кожен у user_input.txt. Виводить кількість збережених рядків.
12	Створіть файл words.txt" зі списком слів. Скрипт підраховує, скільки слів мають довжину більше 6 символів (через "wc -c" у циклі), і виводить результат."
13	"Скрипт перевіряє, чи порожній файл "data.txt ([-s file]). Якщо порожній — виводить попередження, якщо ні — виводить перший рядок (head -1).
14	Створіть файл lines.txt. Скрипт пронумеровує рядки (використовуючи cat -n або nl) і зберігає результат у numbered.txt.
15	Скрипт шукає всі файли з розширенням .log у домашньому каталозі. Копіює їх у каталог logs_backup" (створює, якщо немає). Виводить кількість скопійованих файлів."
16	Створіть файл emails.txt з рядками. Скрипт знаходить усі рядки з @ (grep '@'"), зберігає їх у "valid_emails.txt і виводить кількість.
17	Скрипт створює каталог my_files", а потім у ньому 5 файлів: "file1.txt", ..., "file5.txt", кожен із текстом ""Це файл №X""."
18	Створіть файл urls.txt". Скрипт виводить усі рядки, що містять "http://", і зберігає їх у "insecure_urls.txt.

19	Скрипт копіює вміст input.txt у backup_input.txt", додаючи до кожного рядка префікс "[копія]. Якщо input.txt немає — створює його.
20	"Скрипт видаляє всі файли, що починаються з "temp_ у домашньому каталозі. Перед видаленням питає підтвердження ("read -p ""Видалити? (y/n): """).
21	Створіть файл text.txt". Скрипт підраховує кількість рядків, що починаються з ""A"" або ""a"" ("grep -c '^[AaAa]'"), і виводить результат."
22	Скрипт об'єднує вміст усіх .txt файлів у домашньому каталозі у all_content.txt і виводить кількість об'єднаних файлів.
23	Створіть файл data.txt з числами. Скрипт знаходить найбільше число (через цикл for із sort -n | tail -1"), виводить його на екран та у "max.txt.
24	Скрипт створює файл reverse.txt", у який записує вміст "source.txt у зворотному порядку рядків (tac source.txt > reverse.txt). Якщо source.txt немає — створює його.
25	Створіть файл duplicate.txt з повторюваними рядками. Скрипт видаляє дублікати (sort duplicate.txt | uniq) і зберігає унікальні рядки у unique.txt.
26	Скрипт запитує префікс (наприклад, "doc") і створює 3 файли: "${prefix}1.txt", "${prefix}2.txt", "${prefix}3.txt", кожен із текстом "Файл створено".
27	Створіть файл sentences.txt", де кожне речення закінчується крапкою. Скрипт підраховує кількість таких рядків ("grep -c '\.$') і виводить результат.
28	"Скрипт аналізує розширення файлів у домашньому каталозі, підраховує кількість ".txt", ".log", ".sh файлів і зберігає статистику у file_types.txt.
29	Створіть два файли: old.txt і new.txt. Скрипт порівнює їх за допомогою diff і виводить різницю на екран. Якщо файлів немає — створює прості приклади.
30	Скрипт створює файл summary.txt", у який записує: кількість ".txt" файлів, загальну кількість рядків у них ("wc -l *.txt | tail -1"), і поточну дату."

4 ЗМІСТ ПРОТОКОЛУ

Протокол лабораторної роботи оформлюється відповідно до загальноприйнятих вимог до навчальної документації з використанням шрифту Times New Roman розміром 12 pt та міжрядкового інтервалу 1,5. Протокол має чітку структуру та складається з трьох обов'язкових частин, які подаються у визначеній послідовності.

Вступна частина містить основні відомості про виконану лабораторну роботу. У ній зазначаються тема та мета роботи, прізвище, ім'я та по батькові студента, назва академічної групи, а також прізвище, ім'я та по батькові викладача. Мета лабораторної роботи формулюється відповідно до навчальних цілей дисципліни та вимог методичних рекомендацій. У цій же частині наводяться відповіді на ключові питання, передбачені завданням до роботи. Відповіді мають бути чіткими, лаконічними та логічно обґрунтованими, базуватися на теоретичному матеріалі розділу «Ключові положення» і демонструвати розуміння студентом основних понять, механізмів та принципів функціонування операційної системи Linux.

Основна частина протоколу присвячена безпосередньому виконанню лабораторного завдання. Вона починається з короткого опису робочого середовища, у якому виконувалася робота, із зазначенням версії операційної системи, типу та конфігурації віртуальної машини (кількість виділених процесорних ядер, обсяг оперативної пам'яті, дисковий простір), а також стану системи на момент початку виконання завдання. Далі послідовно описується

виконання кожного етапу лабораторної роботи відповідно до індивідуального варіанта студента. Для кожного кроку подається короткий опис виконуваних дій, наводяться використані команди командного рядка або фрагменти скриптів із поясненням їх призначення, синтаксису та умов застосування, а також фіксуються отримані результати. Результати виконання підтверджуються виводом командного рядка, змінами у файловій системі, станом процесів або служб і супроводжуються відповідними скріншотами.

Під час оформлення основної частини особлива увага приділяється коректному використанню довідкової системи операційної системи (man, параметри --help), механізмів перенаправлення стандартних потоків введення та виведення, конвеєрів команд, а також засобів автозавершення командного рядка. Наприкінці розділу обов'язково наводяться результати перевірки коректності виконання лабораторного завдання, які підтверджують працездатність налаштувань, правильність роботи створених скриптів або функціонування відповідних системних механізмів, таких як права доступу, монтування файлових систем, запуск і робота служб.

У висновковій частині студент формулює узагальнені висновки щодо отриманих теоретичних знань, практичних умінь і професійних компетенцій, набутих у процесі виконання лабораторної роботи. Аналізуються типові помилки або труднощі, що виникали під час виконання завдання, зокрема помилки в синтаксисі команд або скриптів, неправильне налаштування прав доступу чи відсутність необхідних програмних компонентів, а також описуються способи їх усунення. Окремо акцентується увага на практичному значенні отриманих навичок для майбутньої професійної діяльності студента, зокрема у сфері системного адміністрування, DevOps-практик, кібербезпеки та автоматизації IT-процесів.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Матвєєва Н.О. Основи роботи та програмування в операційній системі Linux: навчальний посібник /Н.О. Матвєєва, В.С. Хандецький, О.В. Спірінцева. – Д.: ЛІРА, 2018, –156 с.

2. Операційні системи: [Електронний ресурс]: навч. посіб. для студ. спеціальності 123 «Комп'ютерна інженерія» / В. Г. Зайцев, І. П. Дробязко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2019. – 240 с.

3. Авраменко В. С., Авраменко А. С. Основи операційних систем. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2018. – 524 с.

4. Погребняк Б. І. Операційні системи : навч. посібник / Б. І. Погребняк, М. В. Булаєнко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 104 с.

5. Харченко В. П., Знаковська Є. А., Бородін В. А. Операційні системи та системи програмування: навч. посіб /В. П. Харченко, Є. А. Знаковська, В. А. Бородін – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2012.– 360с.

6. Tanenbaum, Andrew S., Bos, Herbert. Modern Operating Systems (4th Edition) / Andrew S. Tanenbaum, Herbert Bos. - ISBN 978-013-359162-0, Published by Pearson, 2014. – 1136 p.

7. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Електронні текстові дані (1 файл: 2,02 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192с.

8. Голубничий Д.Ю. Операційні системи [Електронний ресурс]/ Д.Ю. Голубничий, А.В. Холодкова. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 317 с. Режим доступу: http://repository.hneu.edu.ua/handle/12 3456789/23844.

9. Операційна система Linux Mint [Електронний ресурс]. – Режим доступу: https://linuxmint.com

10. VirtualBox [Електронний ресурс]. – Режим доступу: https://www.virtualbox.org/

11. Osamu Aoki. Debian Reference. URL: https://www.debian.org/doc/manuals/debian-reference/debian-reference.en.pdf