МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

Методичні вказівки
до виконання лабораторних занять
з дисципліни «Безпека та захист операційних систем. Ч.І.»
для студентів всіх форм навчання
спеціальностей 122 Комп’ютерні науки та 125 Кібербезпека

Луцьк -2023

УДК 004.451.056(072)

М 54

Методичні вказівки до виконання лабораторних занять. з дисципліни «Безпека та захист операційних систем. Ч.І.» для студентів всіх форм навчання спеціальностей 122 Комп'ютерні науки та 125 Кібербезпека [Електронне видання] / уклад. Я.Ю. Дорогий. – Луцьк : ДВНЗ «ДонНТУ», 2023. – 187 с.

Методичні вказівки призначені для студентів спеціальностей 122 Комп'ютерні науки та 125 – Кібербезпека кафедри прикладної математики та інформатики всіх форм навчання. В методичних вказівках наведена тематика перших 5-и лабораторних робіт, теоретичні відомості, список літератури.

Укладач: Я.Ю. Дорогий, д.т.н., доц. кафедри ПМІ

Рецензент: В.В. Цуркан, к.т.н., доцент спеціальної кафедри №5 ІСЗЗІ КПІ ім. Ігоря Сікорського

Відповідальний за випуск: Маслова Н. О., к. т. н, проф., зав. Кафедри прикладної математики та інформатики

Затверджено навчально-методичним відділом ДонНТУ,
протокол № 2 від 24.10.2023 р.

Розглянуто на засіданні кафедри прикладної математики та інформатики
протокол № 10 від 18.10.2023 р.

ЗМІСТ

ВСТУП

Розподіл годин на дисципліну (із загальних 180 годин на дисципліну, 32 виділяється на лабораторні роботи та 100 — на самостійну роботу студентів) визначає основне навантаження із засвоєння цієї дисципліни, а саме на практичну роботу студентів.

Практична суть освіти полягає в тому, що завдання викладача, є не стільки передача певних знань студенту, скільки вміння навчити студента, маючи деякий мінімальний обсяг інформації, методам збільшення цього обсягу як на практичних заняттях під керівництвом викладача, так і вдома, в бібліотеці тощо.

Лише постійне самостійне навчання дає можливість наблизитись до глибин знань певної галузі, оволодіти достатньою сумою знань і вмінь, які б дали змогу почувати себе професіоналом.

ЛАБОРАТОРНА РОБОТА №1. ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ VIRTUALBOX

Мета – навчитися підключатися до віддаленого сервера за протоколом SSH. Навчитися працювати з базовими командами та скриптами. Вивчити основні інструменти роботи з файлової системою. Навчитися використовувати базові інструменти обробки тексту в Linux.

1.1 Теоретичні відомості

1.1.1 Встановлення VirtualBox

Пароль адміністратора на всіх заздалегідь створених віртуальних машинах password.

VirtualBox – це спеціальний засіб для віртуалізації, що дозволяє запускати операційну систему усередині іншої. Воно поставляється у двох версіях – з відкритим та закритим вихідним кодом. За допомогою VirtualBox ми можемо не лише запускати ОС, а й налаштовувати мережу, обмінюватися файлами та робити багато іншого. VirtualBox поставляється у безкоштовному доступі для Linux, Solaris, macOS та Microsoft Windows. Завантажити її можна з офіційного сайту: https://www.virtualbox.org/wiki/Downloads (див. рис. 1.1).

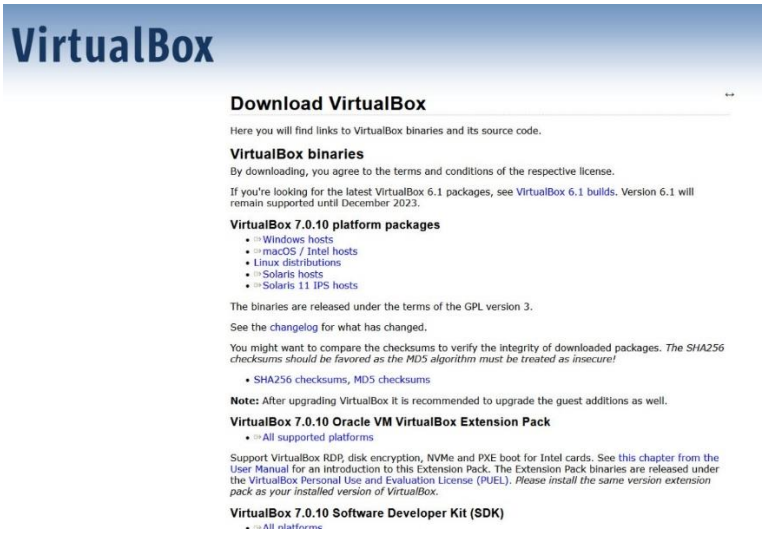


Рисунок 1.1 – Сторінка завантаження VirtualBox

Як тільки інсталяцію буде завершено, перед нами з'явиться головний екран програми (див. рис. 1.2).

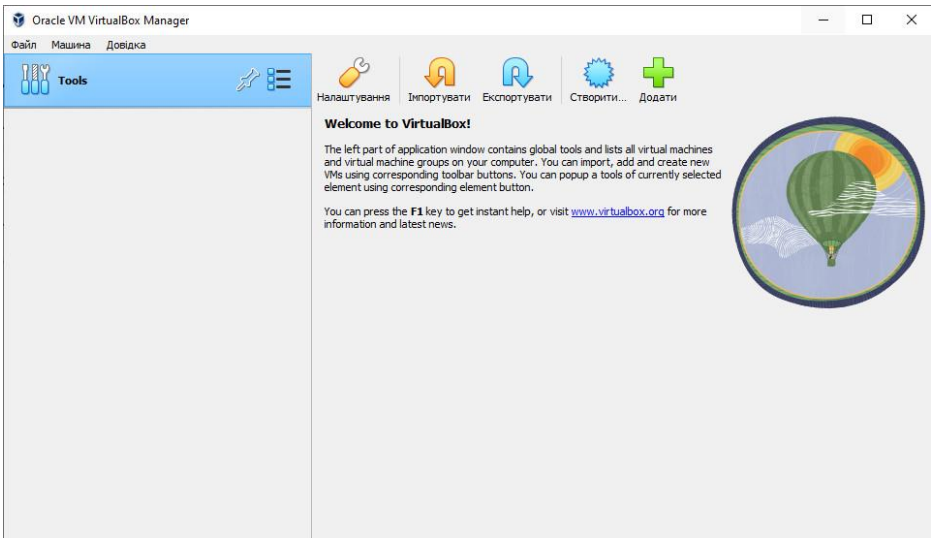


Рисунок 1.2 – Головна форма VirtualBox

Розглянемо, як створити віртуальну машину та провести додаткові налаштування.

1.1.2 Створення віртуальної машини

Основна функція VirtualBox – віртуалізація. Щоб запустити нову операційну систему, потрібно створити для неї віртуальну машину. Для

цього необхідно виконати наступне: запускаємо VirtualBox і в правій частині вибираємо "Створити" (див. рис. 1.3).

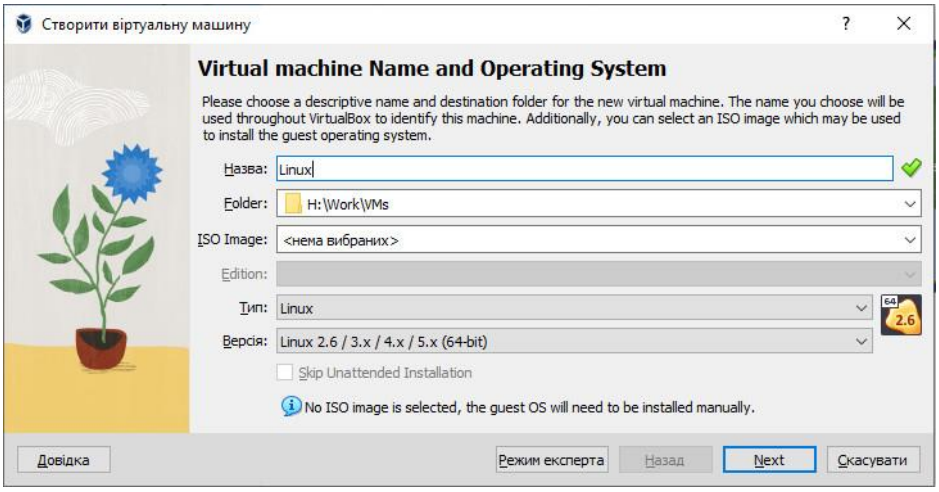


Рисунок 1.3 – Форма створення віртуальної машини

У вікні прописуємо ім'я операційної системи і вказуємо шлях до машини. Зверніть увагу, що тип ОС вибирається автоматично залежно від введеної назви. Також тут можна одразу вказати шлях до образу настановного диска ОС, тоді наступним пунктом будуть параметри автоматичної установки (див. рис. 1.4).

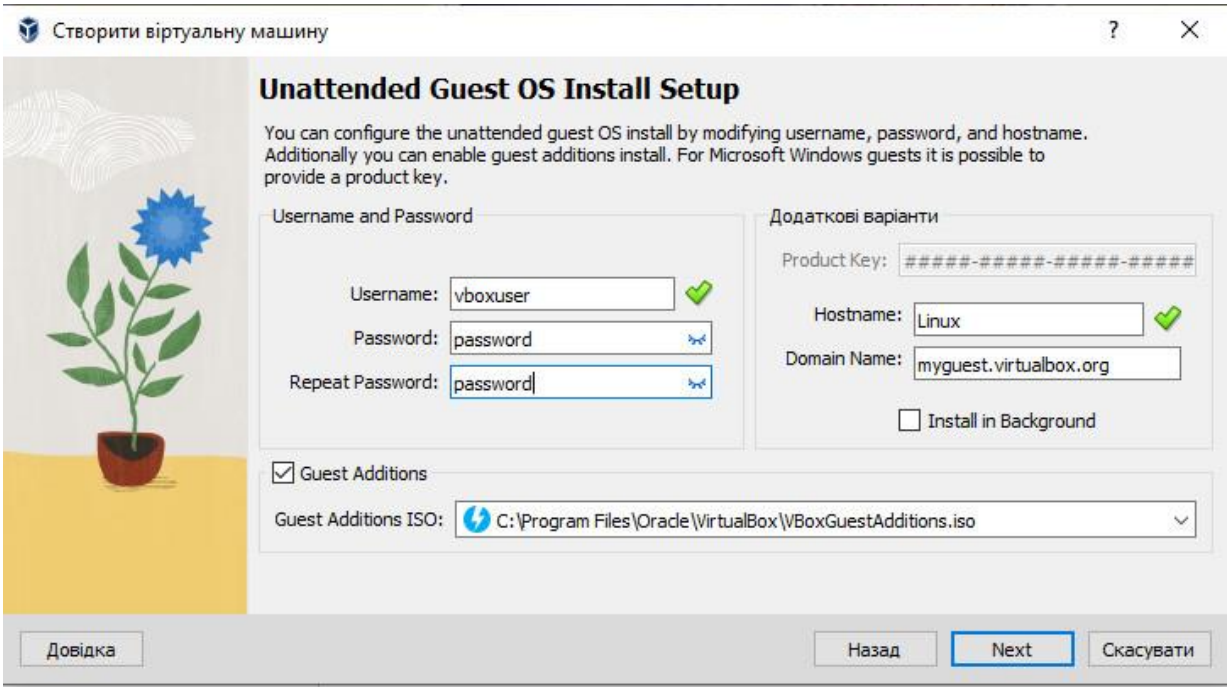


Рисунок 1.4 – Форма конфігурації користувача

Вибираємо, скільки оперативної пам'яті та ядер процесора буде відведено під майбутню ОС (див. рис. 1.5).

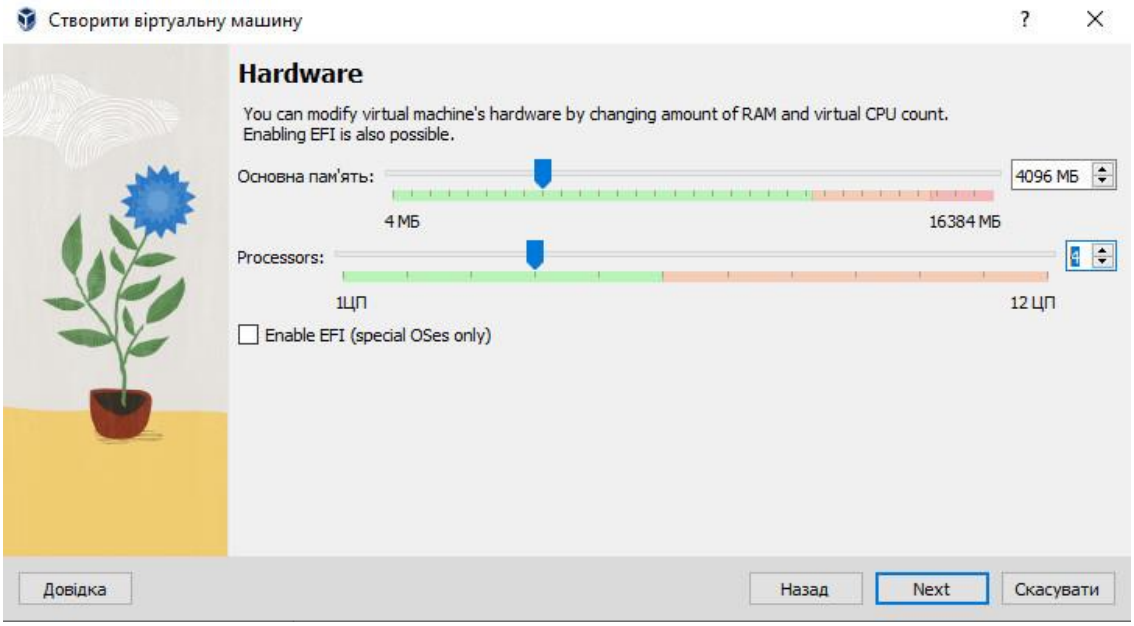


Рисунок 1.5 – Форма конфігурації віртуальної машини

Далі вибираємо об'єм та тип віртуального жорсткого диска: динамічний чи фіксований (виділити місце у повному обсязі). При динамічному віртуальному жорсткому диску розмір файлу диска збільшуватиметься залежно від його заповнення у віртуальній машині (див. рис. 1.6).

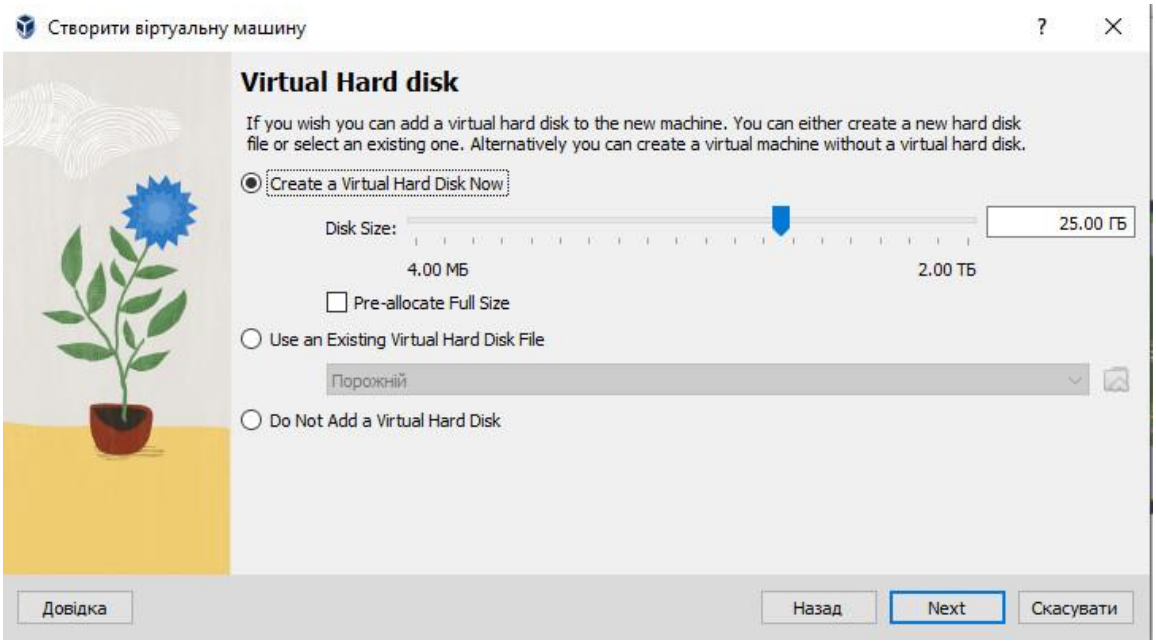


Рисунок 1.6 – Форма конфігурації жорсткого диску

В результаті буде створено нову віртуальну машину. Якщо ми ще не встановлювали операційну систему, то тепер ми можемо запустити віртуальну машину та поставити на неї потрібну ОС, але перед цим пройдемося за деякими параметрами. Клацаємо правою кнопкою миші по

віртуальній машині та вибираємо «Налаштування…» – «Система» (див. рис. 1.7).

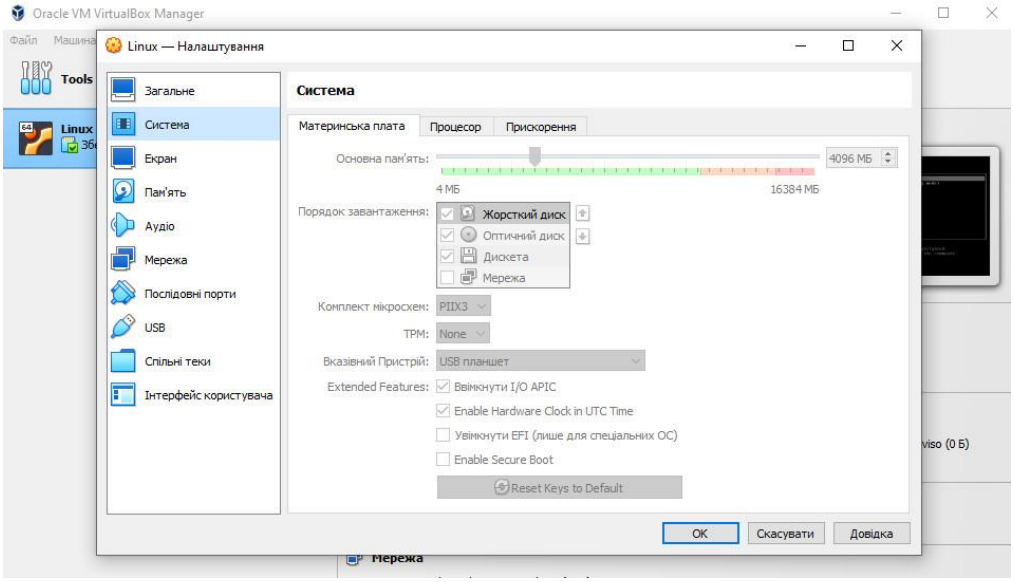


Рисунок 1.7 – Форма налаштування системи

У вікні в розділі «Процесор» можна змінити кількість процесорів і межу завантаження ЦПУ (див. рис. 1.8).

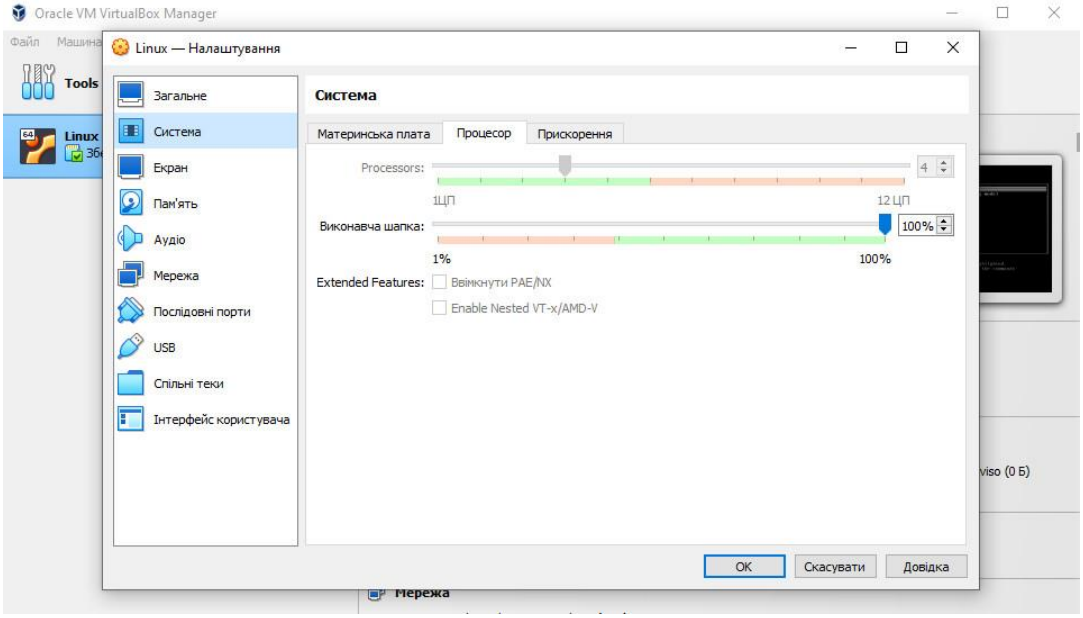


Рисунок 1.8 – Форма налаштування процесора

Функція "Ввімкнути PAE/NX" призначена для підтримки 4 і більше Гб ОЗУ в 32-бітових системах. У вкладці «Прискорення» ми можемо вибрати режим віртуалізації та налаштувати додаткові параметри для збільшення швидкості роботи (див. рис. 1.9).

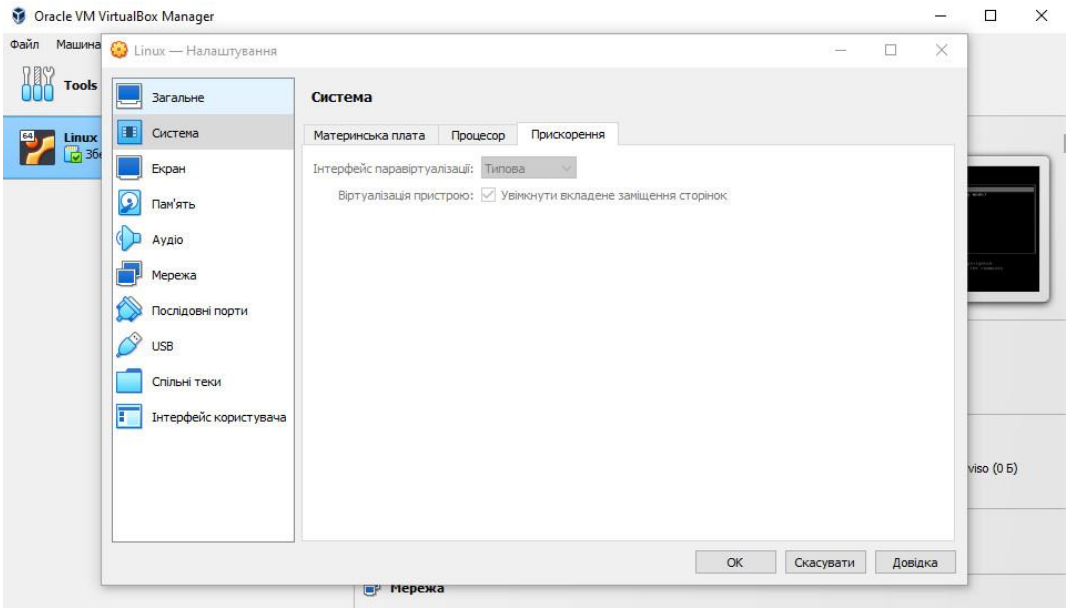


Рисунок 1.9 – Форма налаштування прискорення

Іноді при встановленні нової віртуальної машини значення відеопам'яті за замовчуванням становить 16 Мб, тоді як рекомендується виділяти щонайменше 128 Мб. Змінити це можна в налаштуваннях розділу "Екран" (див. рис. 1.10).

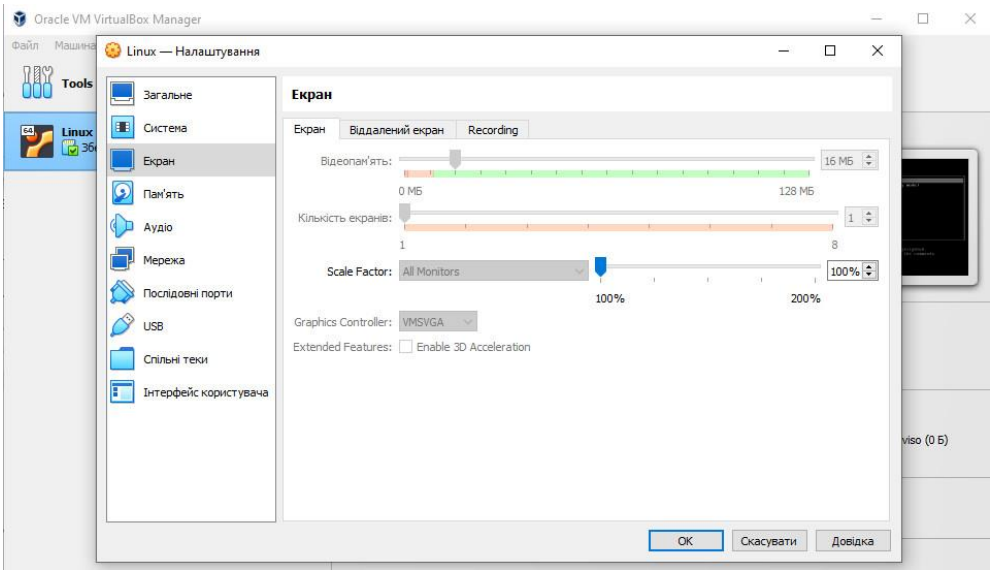


Рисунок 1.10 – Форма налаштування екрану

У цьому розділі можна встановити кількість моніторів, змінити коефіцієнт масштабування та багато іншого. Спочатку віртуальна машина використовує мережу NAT, що цілком зручно, якщо потрібно отримати доступ до інтернету. Якщо вам потрібно налаштувати взаємозв'язок між

різними ВМ, то потрібно виконати додаткові налаштування. У налаштуваннях переходимо до розділу «Мережа» та заходимо до підрозділу «Адаптер 2». Там активуємо пункт «Ввімкнути мережевий адаптер» та вказуємо тип підключення «Лише головний адаптер» (див. рис. 1.11).

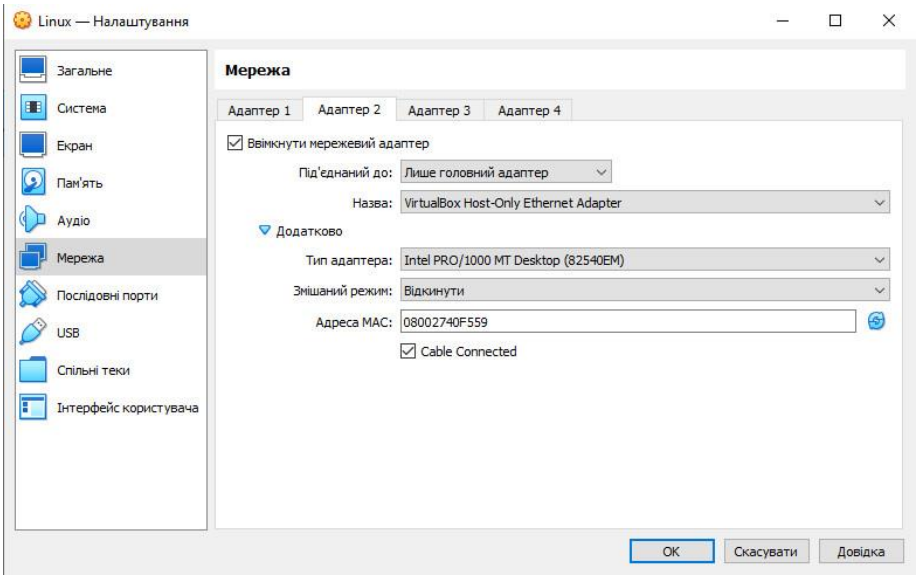


Рисунок 1.11 – Форма налаштування другого мережевого адаптеру

Зверніть увагу на ім'я – тепер усі, хто його використовуватиме, автоматично підключаться до єдиної віртуальної мережі. Ще одна корисна функція - "Клонування". З її допомогою ми можемо зробити резервну копію віртуальної машини, щоб у подальшому звернутися до неї у разі виникнення різноманітних проблем. Для цього клацаємо правою кнопкою миші по віртуальній машині та вибираємо «Клонувати...». Для запуску створеної віртуальної машини у VirtualBox знадобиться завантажувальний диск необхідної операційної системи. Це звичайний образ, який використовується для встановлення ОС на ПК. Встановити його у VirtualBox ми можемо наступним чином:

1. Завантажити ISO-образ дистрибутива системи.

2. Вибираємо створену раніше віртуальну машину та у правій частині натискаємо на кнопку «Запустити».

3. Після запуску віртуальної машини в меню вибрати "Пристрої" - "Вибрати файл диска".

4. Встановити ОС, дотримуючись інструкцій інсталятора.

Як варіант системи для виконання лабораторних вибираємо Linux Mint, дистрибутив якого доступний на офіційному сайті за адресою https://www.linuxmint.com/ (див. рис. 1.12).

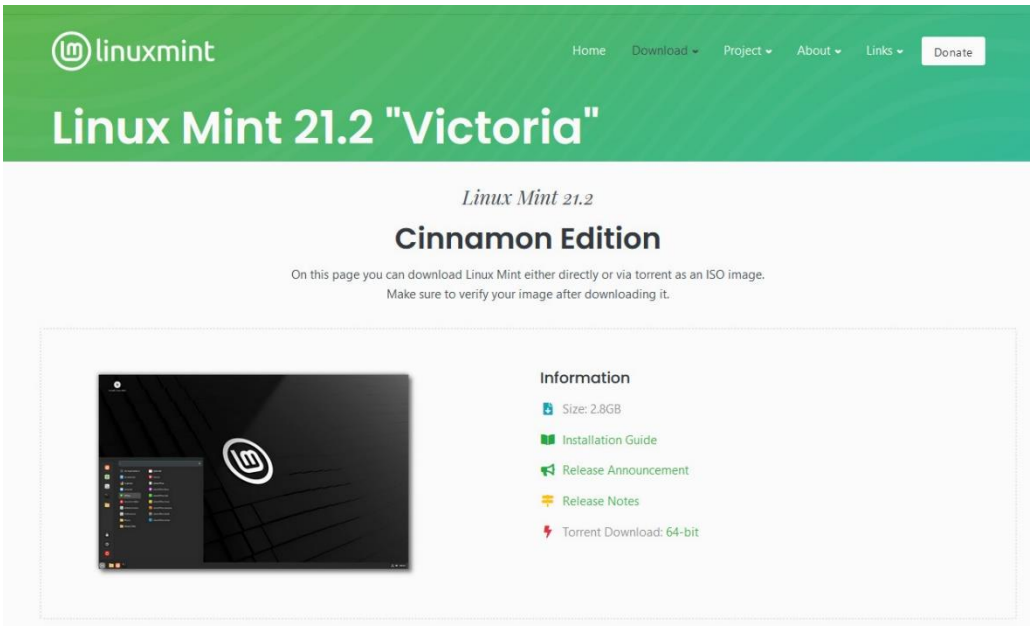


Рисунок 1.12 – Форма завантаження дистрибутиву Linux Mint

Коли операційна система буде встановлена, ви отримаєте доступ до неї через вікно VirtualBox. Наприклад, ось так виглядатиме Linux Mint (див. рис. 1.13):

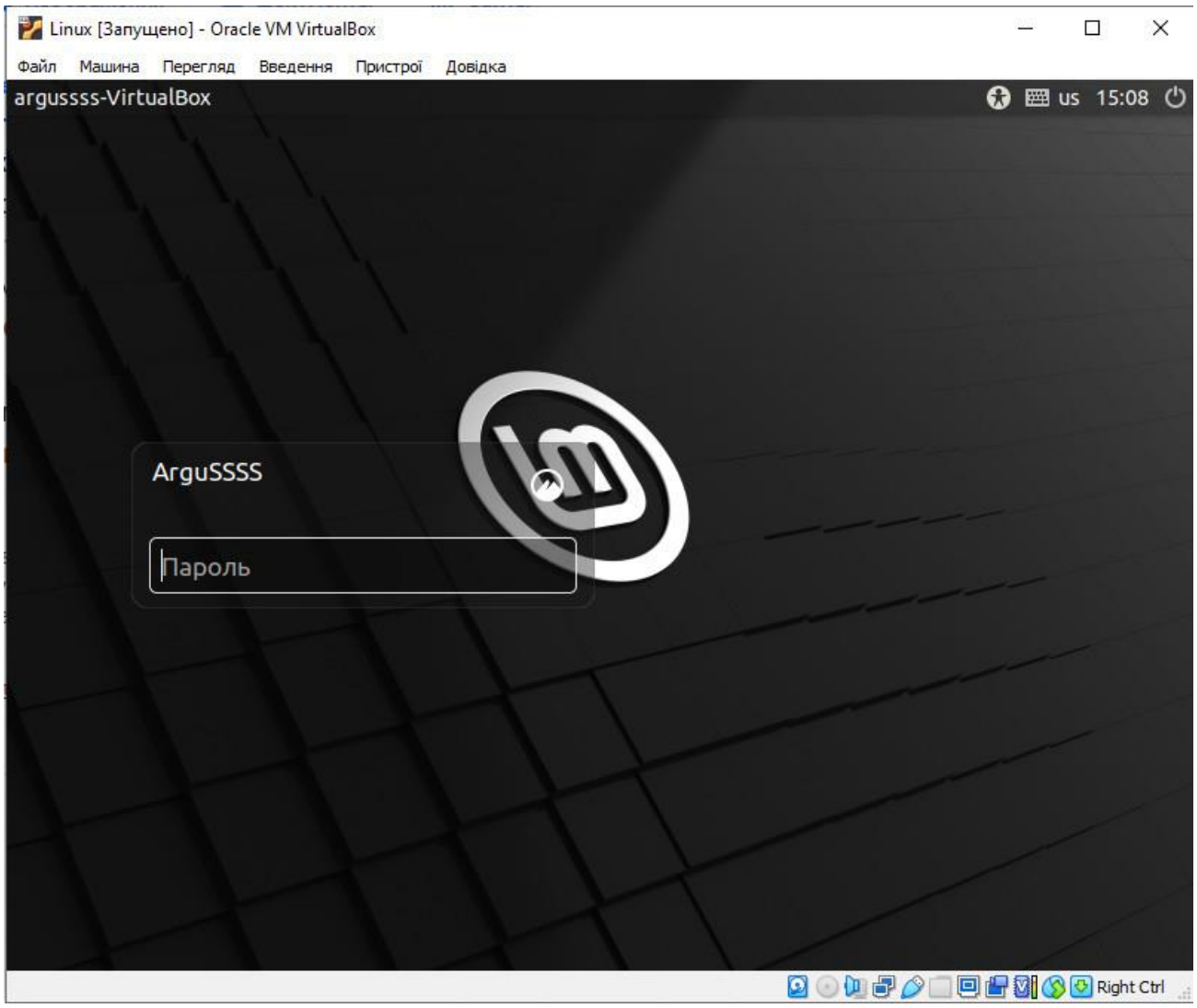


Рисунок 1.13 – Форма входу в Linux Mint

Раніше ми вже створили копію віртуальної машини, якою можна скористатися у разі непередбачених проблем. Але це не єдиний спосіб створення резервної копії – ми можемо також використовувати спеціальну функцію «Зріз стану». Вона дозволяє повертати систему до попереднього стану. Створити зріз можна так: Запускаємо віртуальну машину і у верхній частині вибираємо "Машина" -> "Зробити зріз ..." (див. рис. 1.14).

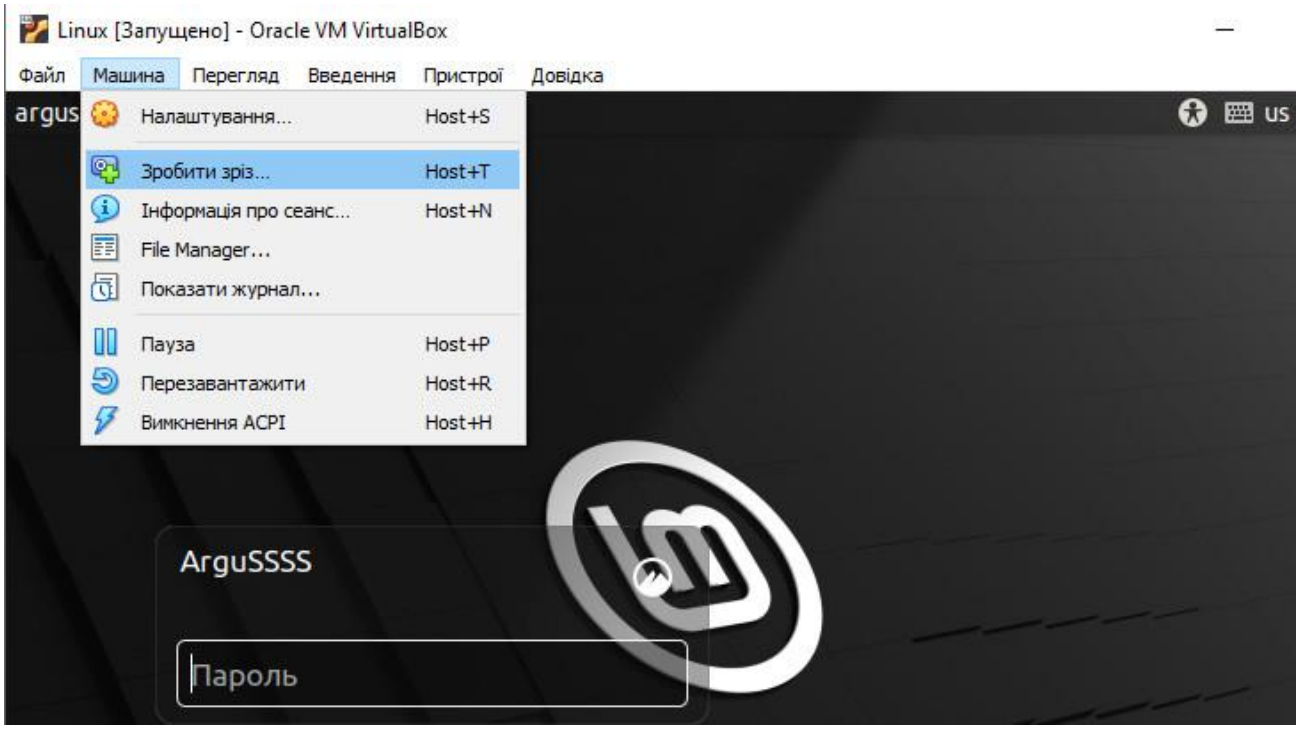


Рисунок 1.14 – Зріз

Задаємо йому ім'я та за бажанням прописуємо опис (див. рис. 1.15).

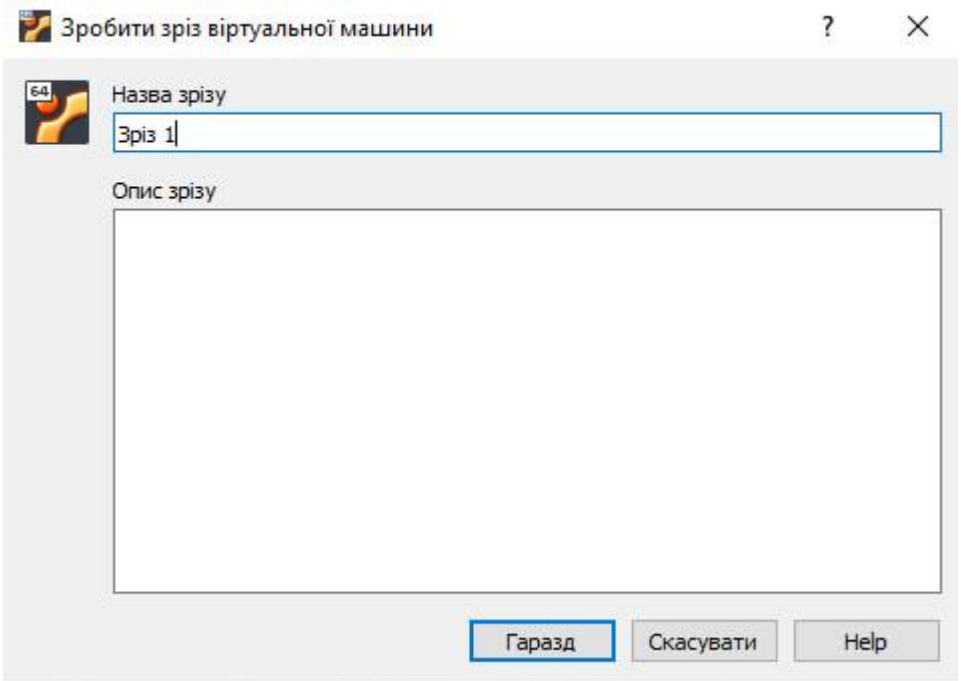


Рисунок 1.15 – Форма створення зрізу віртуальної машини

Повернутися до створеного зрізу ми можемо через меню "Машина" -> "Tools" -> "Зрізи" (див. рис. 1.16).

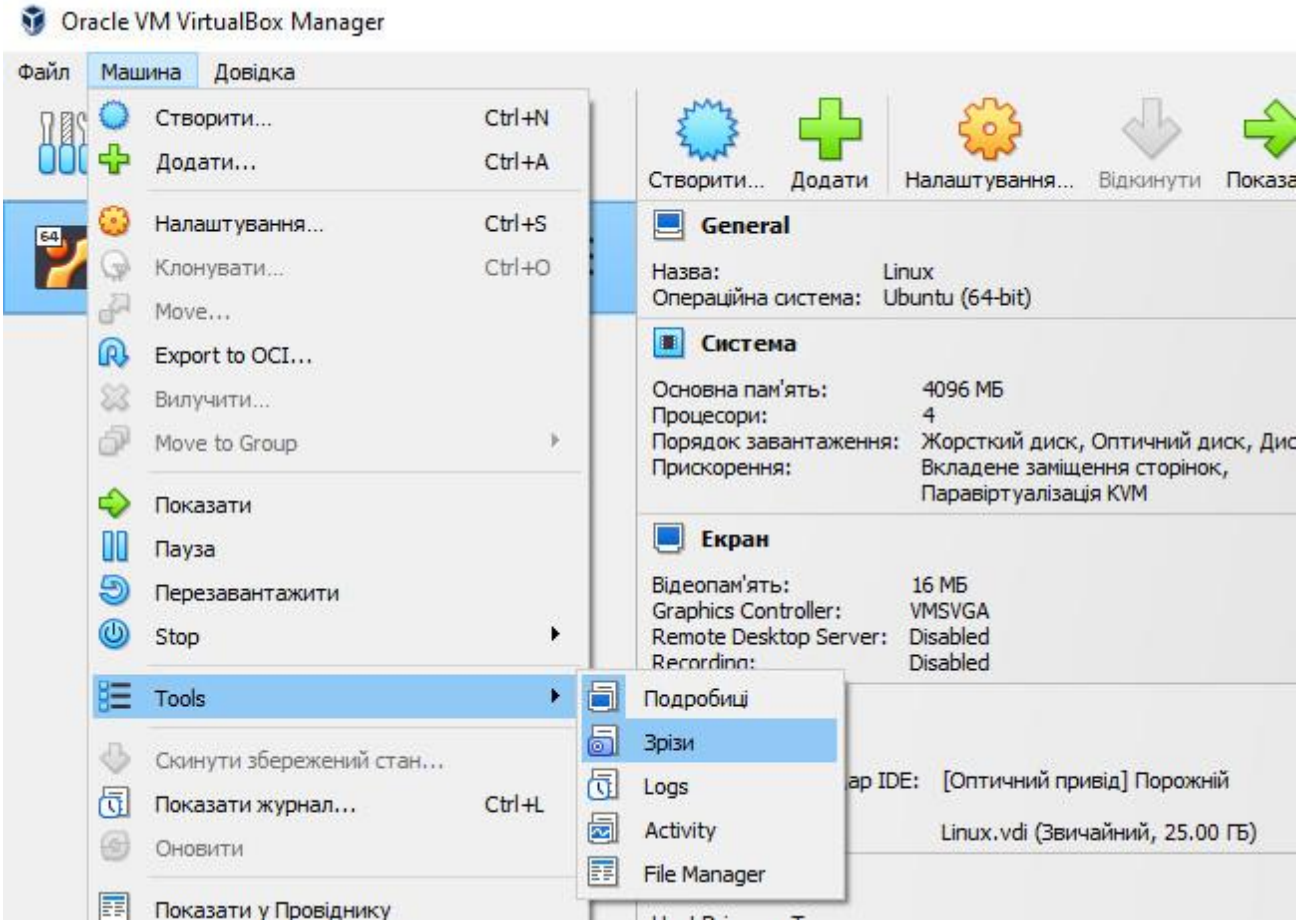


Рисунок 1.16 – Форма вибору зрізу віртуальної машини

1.1.3Підключення нових дисків до віртуальної машини

Виберіть віртуальну машину для якої ви хочете додати ще один віртуальний диск і натисніть кнопку «Налаштування» або клавіші Ctrl+s. Перейдіть на вкладку Носії. Клацніть на «Контролер: SATA», а потім на іконку «Додає жорсткий диск». Якщо ви перенесли віртуальний диск з іншого комп'ютера і він відсутній у цьому списку, але вже існує, то натисніть кнопку "Додати" і вкажіть шлях до нього у файловій системі. Якщо ви хочете створити новий диск, натисніть кнопку «Create». Відкриється знайомий із встановлення віртуальних машин майстер створення віртуальних дисків.

(!) Для виконання завдання з LVM потрібно додати як мінімум 1 новий нерозмічений віртуальний жорсткий диск до віртуальної машини.

1.1.4Налаштування портів віртуальної машини для SSH-підключення

Розглянемо можливість програми VirtualBox прокидати порти в гостьову операційну систему, щоб мати доступом до неї з хоста, тобто з реальної машини, наприклад, у тих випадках, коли підключення (мережа) до гостьової ОС у VirtualBox працює в режимі «NAT».

Ця можливість буде корисна, наприклад, коли Вам потрібен одночасно і доступ до Інтернету в гостьовій операційній системі, і доступ до яких-небудь сервісів у гостьовій ОС з хоста.

Доступ до Інтернету в гостьовій ОС можна отримати за допомогою вибору типу підключення NAT, але як Ви, напевно, знаєте, при цьому втрачається доступ до сервісів гостьової ОС з реального Вашого комп'ютера. Цю проблему якраз і вирішує прокидання портів.

Наприклад, Ви не можете підключитися до віртуальної машини по SSH зі свого комп'ютера, або звернутися до Web сервера, або навіть просто скопіювати команду і вставити в консоль. Тому прокинемо порт у гостьову ОС, наприклад, для того щоб підключитися до неї по SSH всім відомою програмою PuTTY. За допомогою неї ми зможемо без проблем керувати сервером і в разі потреби копіювати команди до неї.

PuTTY – це безкоштовна клієнтська програма для віддаленого підключення, наприклад, до серверів за протоколами SSH, Telnet та іншими. Вона дозволяє керувати віддаленим комп'ютером.

PuTTY – це клієнтська частина, серверна має бути реалізована на віддаленій стороні, має бути встановлений SSH сервер. Завантажити PuTTY можна з офіційного сайту - www.putty.org.

1.1.4.1 Налаштування прокидання портів у VirtualBox для SSH

SSH – сервер, що зазвичай прослуховує 22 порт, тому нам необхідно прокинути підключення саме на 22 порт. Для того, щоб прокинути порт, запускаємо VirtualBox і заходимо в налаштування нашої віртуальної машини. Потім переходимо до розділу налаштувань «Мережа», і відкриваємо вкладку

з увімкненим адаптером. Тип підключення – «NAT» – для доступу в Інтернет. Далі клацаємо на пункт "Додатково", щоб відобразити додаткові налаштування даного адаптера, і після цього натискаємо кнопку "Переадресування порту" (див. рис. 1.17).

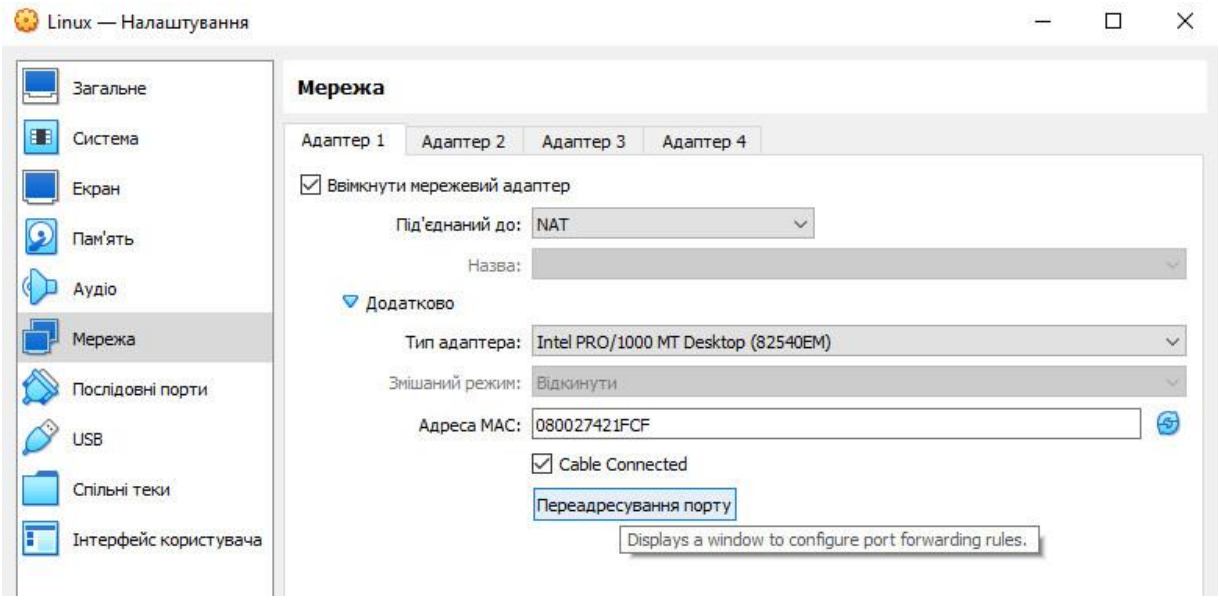


Рисунок 1.17 – Переадресування порту

У результаті відкриється вікно «Правила переадресування порту». Для додавання нового правила натискаємо на іконку з плюсиком (див. рис. 1.18).

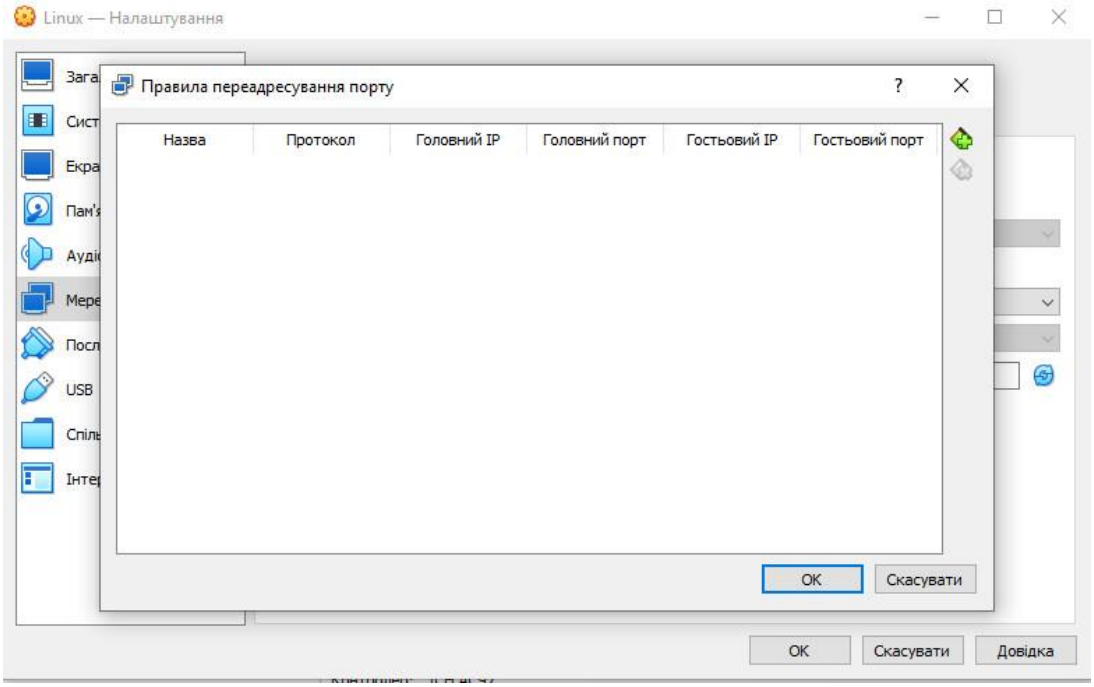


Рисунок 1.18 – Правила переадресування порту

На даному етапі Ви вже повинні знати IP-адресу гостьової операційної системи та порт, який прослуховує SSH сервер (за замовчуванням це 22

порт). IP-адресу можна дізнатися за допомогою команди ifconfig (за замовчуванням це 10.0.2.15).

Опис колонок таблиці з правилами:

Назва – назва правила, у разі розумно назвати SSH;

Протокол – протокол, за яким відбуватиметься взаємодія. У звичайних випадках це TCP;

Головний IP – IP адреса Вашого реального комп'ютера, можна вказати 127.0.0.1 або залишити це поле порожнім;

Головний порт – будь-який вільний порт комп'ютера, який буде використовуватися для перенаправлення на потрібний порт у гостьовій ОС, наприклад 2222;

Гостьовий IP – тут вказуємо IP адресу гостьової операційної системи, на яку відбуватиметься перенаправлення, за замовчуванням це 10.0.2.15;

Гостьовий порт – порт гостьової ОС, на який нам необхідно прокидати наші запити. У нашому випадку це 22 порт, який прослуховує сервер SSH. Після заповнення таблиці з правилом натискаємо "ОК". На цьому налаштування перенаправлення портів закінчено, тепер ми можемо перевірити роботу цього правила (перенаправлення) (див. рис. 1.19).

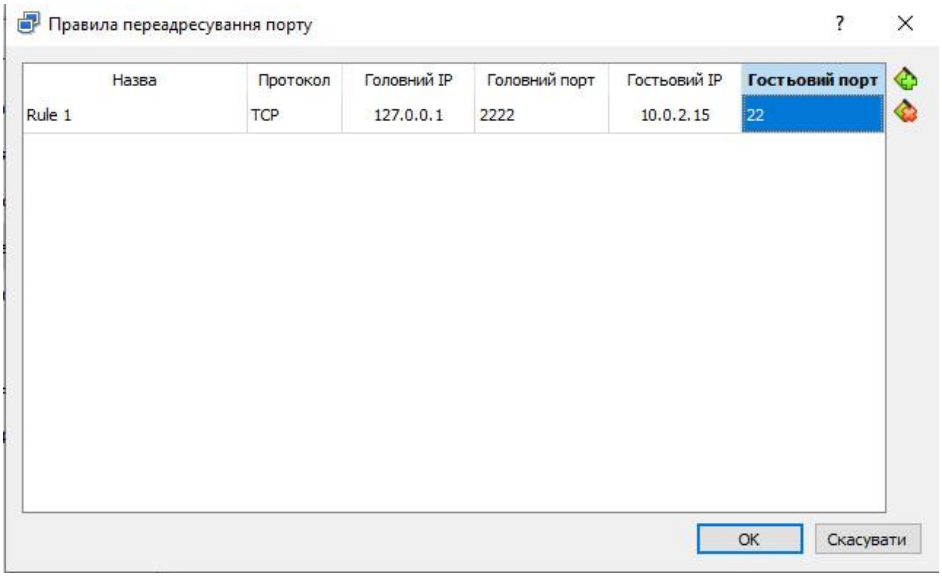


Рисунок 1.19 – Налаштування правила переадресування порту

Для цього спочатку запускаємо віртуальну машину, перевіряємо наявність встановленого пакету openssh-server (команда в терміналі apt install

openssh-server), а потім запускаємо програму PuTTY. На вкладці Session вибираємо тип з'єднання SSH і в полі Host Name пишемо локальну адресу комп'ютера (хоста), якою ми вказували в колонці «Головний IP» в правилах перенаправлення портів. Якщо нічого не вказували, то пишемо 127.0.0.1. У полі порт вказуємо номер порту, який вказували в колонці «Головний порт», тобто. 2222 (див. рис. 1.20).

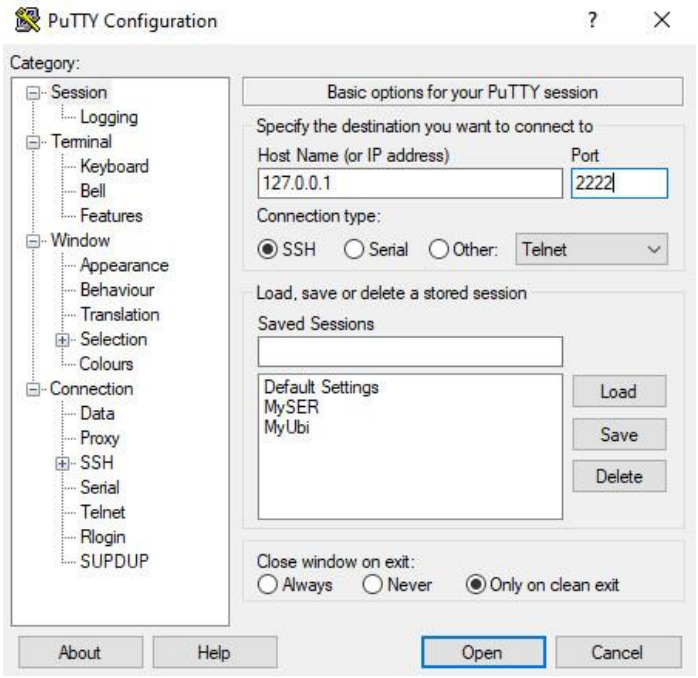


Рисунок 1.20 – Налаштування Putty

Натискаємо кнопку Open. Якщо все добре, то відображається наступне вікно (див. рис. 1.21).

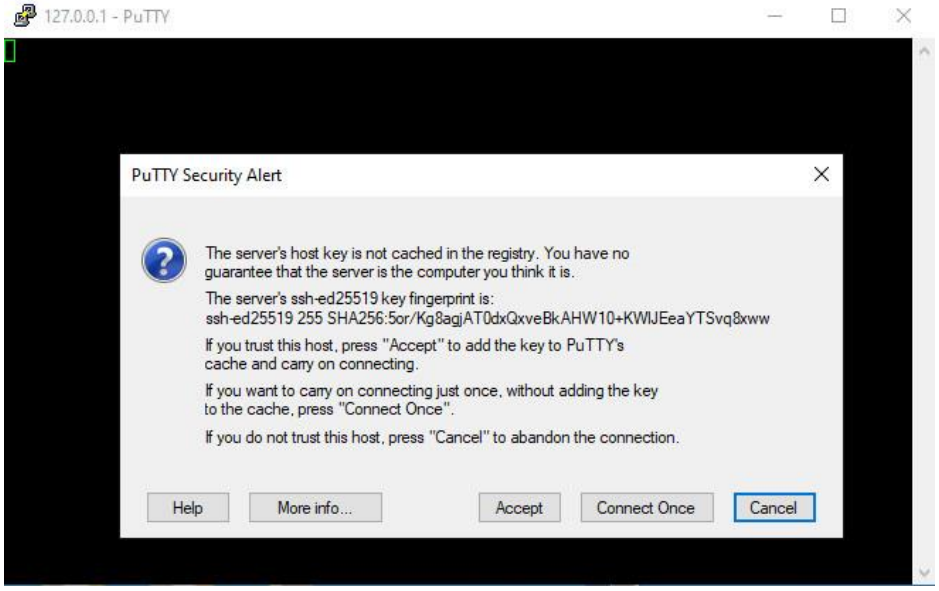


Рисунок 1.21 – Вікно Putty

Натискаємо кнопку Accept і переходимо до консолі введення, в якій пропонується ввести логін та пароль для доступу до віртуальної машини (див. рис. 1.22).

Рисунок 1.22 – Консоль введення

1.1.4.2 Альтернативний варіант підключення: OpenSSH

У Windows 10 і Windows Server 2019 з'явився вбудований SSH клієнт, який ви можете використовувати для підключення до *Nix серверів, ESXi хостів та інших пристроїв із захищеним протоколом, замість Putty, MTPuTTY або інших сторонніх SSH клієнтів. Вбудований SSH клієнт Windows заснований на порту OpenSSH і встановлено в ОС, починаючи з Windows 10 1809.

Встановлення клієнта OpenSSH у Windows 10. Клієнт OpenSSH входить до складу Features on Demand Windows 10 (як і RSAT). Клієнт SSH встановлено за замовчуванням у Windows Server 2019 та Windows 10 1809 та новіших білдах. Перевірити, що SSH клієнт встановлений можна наступною командою в консолі PowerShell (див. рис. 1.23):

```
Get-WindowsCapability -Online | ? Name -like 'OpenSSH.Client*'
```

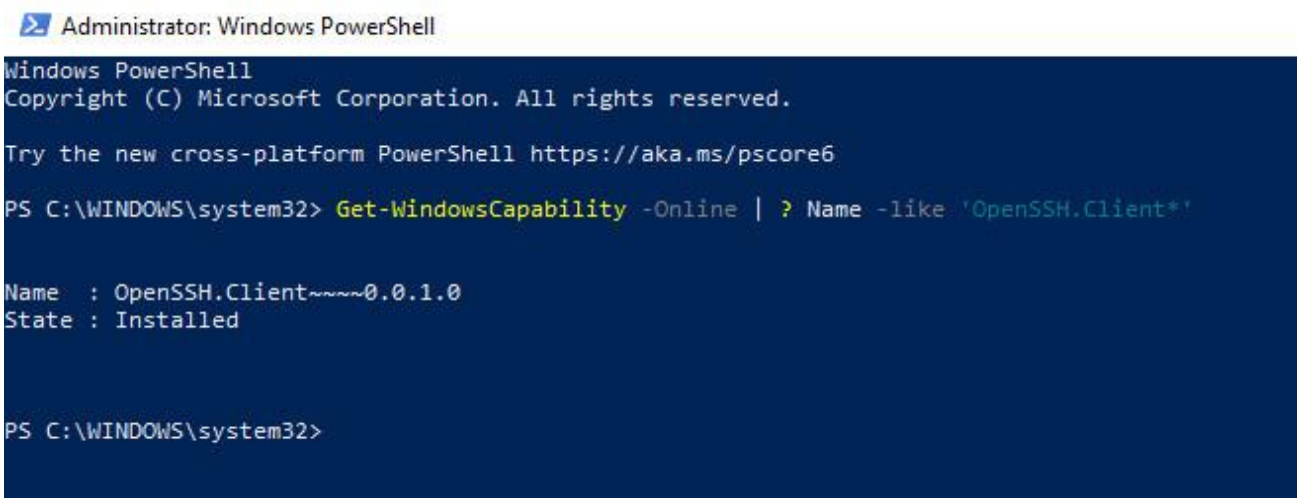


Рисунок 1.23 – Консоль PowerShell

У нашому випадку клієнт OpenSSH встановлений (статус: State: Installed). Якщо SSH клієнт відсутній (State: Not Present), його можна встановити:

✓ за допомогою команди PowerShell (див. рис. 1.24):

```
Add-WindowsCapability -Online -Name OpenSSH.Client*
```

✓ за допомогою DISM (див. рис. 1.25):

```
dism /Online /Add-Capability /CapabilityName:OpenSSH.Client
~~~~0.0.1.0
```

✓ через "Параметри" - "Додатки" - "Додаткові можливості" - "Додати компонент". Знайдіть у списку Клієнт OpenSSH та натисніть кнопку «Встановити».

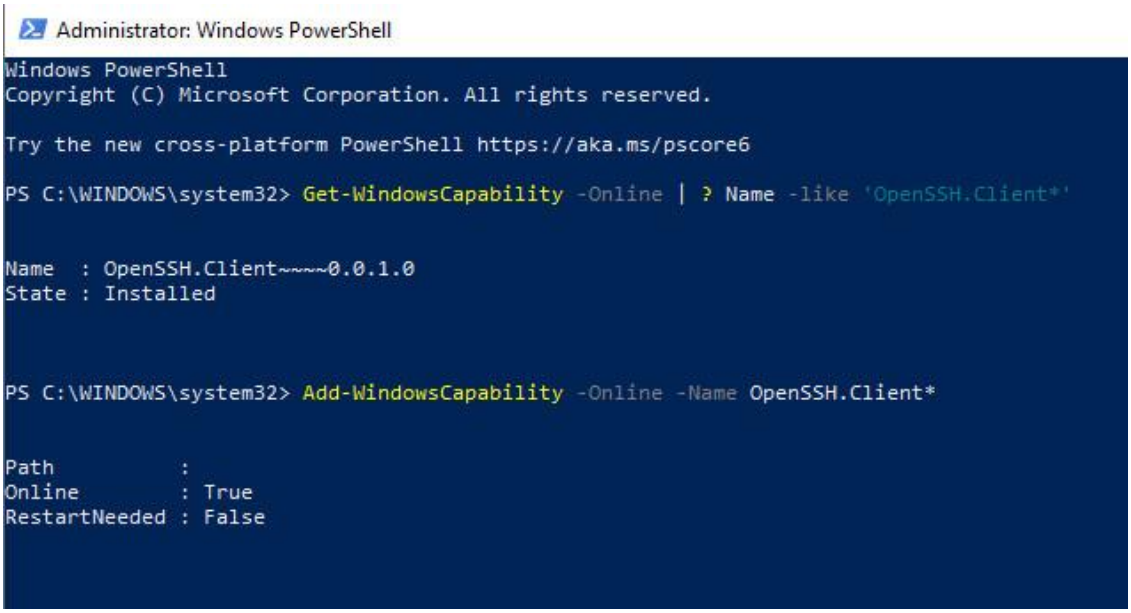


Рисунок 1.24 – Консоль PowerShell

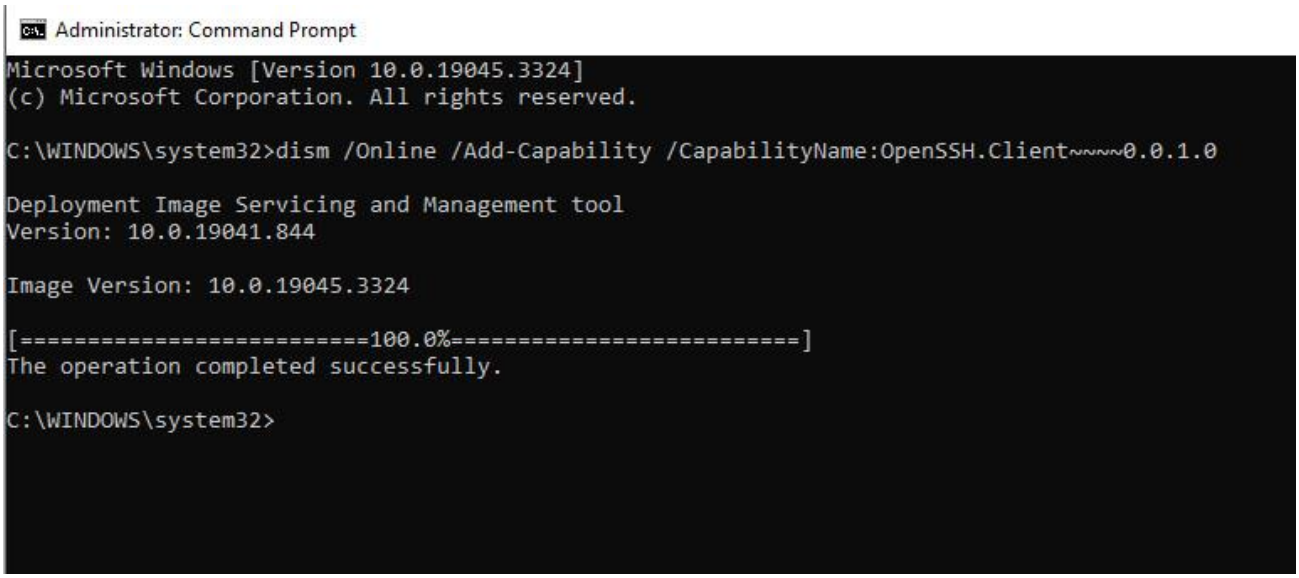


Рисунок 1.25 – Консоль CMD

1.1.4.3 Використання SSH клієнту у Windows 10

Щоб запустити клієнт SSH, запустіть командний рядок `PowerShell` або `cmd.exe`. Виведіть доступні параметри та синтаксис утиліти `ssh.exe`, набравши команду (див. рис. 1.26): `ssh`.

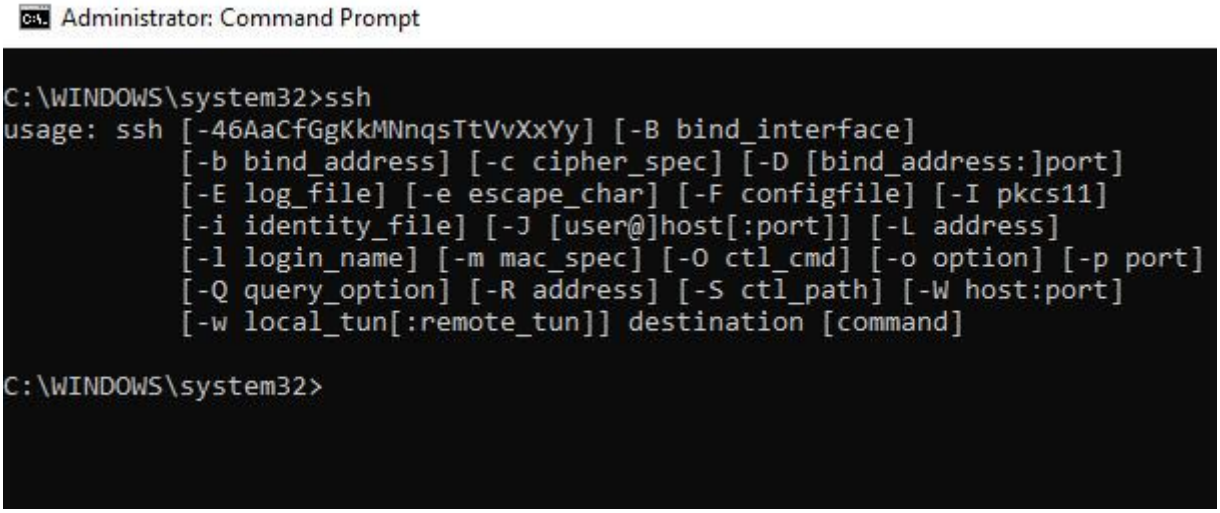


Рисунок 1.26 – Консоль CMD

Для підключення до віддаленого сервера SSH використовується команда:

```
ssh username@ip
```

Якщо сервер SSH запущено на нестандартному порту, відмінному від TCP/22, можна вказати номер порту:

```
ssh username@ip -p port
```

Наприклад, щоб підключитися до нашого Linux хосту під root, виконайте:

```
ssh root@127.0.0.1 -p 2222
```

При першому підключенні з'явиться запит на додавання ключа хоста в довірені, наберіть `yes -> Enter` (при цьому відбиток ключа хоста додається до файлу `C:\Users\username\.ssh\known_hosts`) (див. рис. 1.27).

Рисунок 1.27 – Використання SSH клієнту

1.1.4.4 Створення SSH-ключа

SSH-ключі використовуються для авторизації на сервері SSH без використання пароля з метою безпеки та зручності. Хорошим підходом є повна заборона авторизації на SSH-сервері за паролем та використання виключно SSH-ключів, створених для індивідуальних користувачів. У ключа є відкрита (зберігається на сервері SSH) та закрита (зберігається на комп'ютері користувача) частини. Ці частини називаються парою ключів.

1. Згенерувати ключ. Виконати команду `"ssh-keygen -t rsa"`. При виникненні повідомлення `"Enter passphrase"` допустимо не задавати жодну ключову фразу.

2. Вказати місце створення ключа. При виникненні повідомлення `"Enter file in which to save the key"` допустимо або вказати кращий файл, у який згенерується ключ, або залишити за замовчуванням - `"/home/username/.ssh/id_rsa"`.

3. Ключ буде згенеровано. Закрита частина буде розташована за замовчуванням у файлі `id_rsa` у зазначеній вище директорії, а відкрита - у файлі `id_rsa.pub`.

4. Скопіювати пару в потрібні місця. Відкрита частина копіюється у файл `/home/username/.ssh/authorized_keys`, закрита - на свій локальний комп'ютер (для копіювання файлу на локальний комп'ютер можна використовувати SCP або PSCP, які розглянуті далі).

При використанні openssh команда виглядає так:

```
ssh –i /home/username/.ssh/id_rsa username@ip –p port,
```

де `username` – ім'я користувача на сервері, `ip` – адреса підключення (127.0.0.1 у разі віртуальної машини, `port` – номер порту для підключення).

Або, як приклад для шляху в Windows:

```
ssh username@ip -i "C:\Users\username\.ssh\id_rsa"
```

1.1.4.5 Передача файлів між локальним комп'ютером та віртуальною машиною

SCP: копіювання файлів з/в Windows через SSH. За допомогою утиліти scp.exe, яка входить до складу пакета клієнта SSH, ви можете скопіювати файл із вашого комп'ютера на SSH сервер:

```
scp.exe "E:\docs\report.txt" username@ip:/home
```

Можна рекурсивно скопіювати весь вміст каталогу:

```
scp.exe -r E:\docs\username@ip:/home
```

І навпаки, ви можете скопіювати файл з віддаленого сервера на ваш комп'ютер:

```
scp.exe username@ip:/home/report e:\tmp
```

Якщо ви налаштуєте автентифікацію за ключами RSA, то при копіюванні файлів не з'являтиметься запит на введення пароля для

підключення до SSH сервера. Це зручно, коли вам потрібно настроїти автоматичне копіювання файлів за розкладом.

(!) У утиліті scp.exe номер порту для підключення, на відміну від команди ssh, вказується через опцію `-P`*, а чи* `не -p`.

Альтернативний спосіб передачі файлів: команда `pscp`. `pscp` – це реалізація протоколу SCP, в якій ми можемо безпечно передавати та копіювати файли та папки по мережі за допомогою з'єднання SSH.

Утиліту pscp можна завантажити за наступним посиланням: https://putty.org.ru/download.html.

`pscp` можна встановити в автономному режимі або з пакетом установки putty.

Використання pscp.exe.

1. Завантажити утиліту pscp.exe в довільну директорію на комп'ютері адміністратора, наприклад, `C:\Putty`.

2. На комп'ютері адміністратора запустити консоль і перейти до директорії установки Putty командою: `cd C:\Putty`.

3. Запустити утиліту pscp.exe, наприклад, з наступними параметрами:

`pscp E:\test.cer username@ip:/home`,

де `E:\test.cer` – шлях до файлу на комп'ютері адміністратора, який потрібно доставити на віддалений сервер;

`ip` – адреса інтерфейсу віддаленого сервера;

`/home` – директорія на сервері, в яку буде скопійовано файл.

4. З'явиться запит на введення пароля користувача `root`. Ввести пароль та натиснути `Enter`.

`username@ip's password`:

Завантаження файлу з віддаленого сервера та додаткові параметри команди `PSCP` виконуються аналогічно `SCP`.

1.1.5 Робота з файловою системою Linux

1.1.5.1 Деякі корисні команди

BASH має вбудовані команди, а також здатний запускати зовнішні програми. Деякі з корисних команд наведено в таблиці 1.1.

Таблиця 1.1 – Корисні команди Linux

Тип	Назва команди	Опис
Операції з файлами	`touch, cp, mv, rm`	Створення, копіювання, переміщення, видалення
Операції з директоріями	`mkdir, rmdir`	Створення, видалення директорії
	`cd, pwd, ls`	Зміна директорії, виведення поточного шляху, лістинг директорії
Виведення файлів	`cat, tail, less`	Виведення, виведення останніх рядків, виведення перших рядків
Операції з процесами	`ps, kill`	Виведення списку процесів, завершення процесу
Операції з правами	`chown, chmod`	Зміна власника, зміна прав доступу до файлу/директорії
Пошук	`grep`	Пошук рядка у файлах
Довідка	`man`	Виведення довідкової сторінки про конкретну програму
Виведення	`Echo`	Виведення текстового рядка

Конвеєр – механізм, що дозволяє перенаправити виведення однієї команди на вхід іншої та складати ланцюжки виконання команд.

Приклад:

```
cat./file | grep test
```

Цикли – послідовне виконання команд із заданою кількістю кроків.

Приклад:

```
for i in {1..10}; do
echo $i
done
```

Змінні оточення – задані змінні, до яких мають доступ програми, що запускаються.

Приклад:

```
echo $PATH
```

Пошук з історії команд BASH можна здійснювати не тільки пошуком файлу `~/.bash_history`, але й за допомогою контекстного пошуку, натиснувши на "`ctrl+r`" і почавши вводити початок шуканого рядка.

Цикли можна використовувати не тільки у скриптах, а й безпосередньо в командному рядку: `for i in {1..10}; do echo $i; done`

У цьому випадку функцію перенесення рядка виконує знак "`;`".

Рядок з командою в командному рядку можна закоментувати так само, як і у файлі скрипта, поставивши "`#`". Далі до неї можна повернутись пошуком.

Щоб команда, що виконується, не збереглася в `~/.bash_history` достатньо прямо перед нею поставити один знак пробілу.

Для продуктивної роботи з виведенням на екран одночасно кількох bash сесій прийнято використовувати термінальні мультиплексори: `tmux`, `terminator` тощо.

Щоб тривала команда не перервалася при відключенні користувача від сесії bash використовують програму "`screen`".

Виконувану команду можна відправити у фоновий режим натисканням «`ctrl+z`», а повернути – набравши `fg` (переміщення фонового процесу в активний).

(!) "Все є файл" - концепція побудови ієрархії ФС та роботи в UNIX-подібних ОС, згідно з якою робота з системою зводиться до роботи з файлами.

1.1.5.2 Ієрархія файлової системи

FHS – Filesystem Hierarchy Standard, стандарт ієрархії файлової системи, що уніфікує розташування та призначення файлів та структури директорій у Linux:

`/boot` - файли, необхідні для завантаження ОС: ядро ОС, конфігураційні файли та модулі;

`/dev` – файли всіх пристроїв у ОС;

`/etc` – конфігураційні файли програм та системних модулів;

`/home` – домашні директорії користувачів (крім суперкористувача);

`/opt` - директорія для інших програм. Наприклад для тих, що не поставляються у складі дистрибутива;

`/root` – домашня директорія суперкористувача (root);

`/mnt` – директорія для монтування інших ФС;

`/proc, /sys` – "віртуальні" ФС для зберігання змінних системи/ядра ОС;

`/usr/bin` – двійкові файли застосунків;

`/usr/lib` – бібліотеки застосунків;

`/var/log` – логи застосунків та системи;

Основні конфігураційні та інформаційні файли Linux:

`/etc/fstab` – список монтованих ФС та опції монтування;

`/ets/resolv.conf` – список використовуваних DNS-серверів;

`/etc/hosts` – список відповідностей DNS-імен та IP-адрес;

`/var/log/messages, /var/log/syslog` – основні системні журнали (`syslog`), розташування відрізняється у різних дистрибутивах;

/etc/crontab, /etc/cron.d/ – файл та директорія із завданнями планувальника cron;

/etc/passwd, /etc/groups – список користувачів та груп у Linux;

/etc/sudoers, /etc/sudoers.d/ – список правил підвищення повноважень користувачами та групами;

/etc/default/grub – настроювальний файл для завантажувача GRUB;

/home/username/.bashrc – конфігураційний файл BASH для конкретного користувача;

/home/username/.bash_history – список останніх команд, виконаних інтерпретатором BASH від конкретного користувача;

1.1.5.3 Приклади файлових систем Linux

Приклади файлових систем, що використовуються в Linux наведено в таблиці 1.2.а

Таблиця 1.2 – Приклади файлових систем Linux

Назва ФС	**Де використовується**	**Особливості**
ext2	Накопичувачі з обмеженими циклами запису	Немає журналу операцій
ext3, ext4	Системні розділи загального призначення	Використовуються за замовчуванням у безлічі дистрибутивів
xfs	Файлові сховища	Ефективна робота з великими файлами, не зменшується
fat	Завантажувальні розділи, флеш накопичувачі	Підтримується величезною кількістю пристроїв
reiserfs	Системні розділи	Ефективна робота з безліччю дрібних файлів, слабо розвивається

/boot – містить дані, необхідні для завантаження ОС;

`swap` – містить розділ підкачування та використовується при вичерпанні оперативної пам'яті;

`/` - містить всю ієрархію ФС;

`/home` – містить домашні каталоги користувачів;

`/mnt/storage` – містить «зовнішню» ФС, що підключається.

1.1.5.4 Основні команди для роботи з ФС

Основні команди для роботи з ФМ наведено в таблиці 1.3.

Таблиця 1.3 – Основні команди для роботи з ФС

Назва команди	**Опис**	**Приклад**
`df`	Виведення інформації про утилізацію підключених ФС	`df -h`
`du`	Розрахунок розміру файлів та директорій	`du -ha /var/log/`
`resize2fs`	Зміна розміру ФС ext2/ext3/ext4	`resize2fs /dev/sda`
`fdisk`	Перегляд інформації про розділи дисків	`fdisk /dev/sda`
`fsck`	Запуск перевірки ФС на цілісність	`fsck /dev/sda`
`mkfs`	Створення ФС на диску	`mkfs.ext4 /dev/sda`
`mount, umount`	Монтування, відмонтування ФС	`mount /dev/sda /mnt/storage`

1.1.5.5 Основні команди для роботи з LVM

Менеджер логічних томів (англ. logical volume manager) – підсистема операційних систем Linux і OS/2, що дозволяє використовувати різні області одного жорсткого диска та/або області з різних жорстких дисків як один логічний том. Реалізовано за допомогою підсистеми device mapper.

LVM додає рівень абстракції між фізичними/логічними дисками (звичними розділами, з якими працює fdisk та аналогічні програми) та файловою системою. Це досягається шляхом розбивки початкових розділів

на блоки або використання окремих розділів або блокових пристроїв (physical volume (pv)) та об'єднання їх у єдиний віртуальний том, точніше групу томів (volume group (vg)), яка далі розбивається на логічні томи (logical volume (lv)). Для файлової системи логічний том представлений як звичайний блоковий пристрій, хоча окремі pv-томи можуть бути на різних фізичних пристроях (і навіть сам pv може бути розподілений подібно до RAID).

Терміни:

Фізичний том (physical volume, pv) – пристрій, що представляється системі як один диск (жорсткий диск або його розділ, RAID-масив).

Група томів (англ. volume group, vg) – кілька фізичних томів pv (група, набір).

Логічний том (англ. logical volume, lv) – логічний розділ; аналог розділів hda1, sdb3 та ін; віртуальний блоковий пристрій.

Фізичний діапазон (англ. physical extent, pe) – область на фізичному томі pv розміром у кілька мегабайт. pv розбивається на ділянці pe рівного розміру.

Логічний діапазон (англ. logical extent, le) – область на логічному томі lv. lv розбивається на ділянці le рівного розміру.

Основні команди для роботи з LVM наведено в таблиці 1.4.

Таблиця 1.4 – Основні команди для роботи з LVM

Назва команди	**Опис**	**Приклад**
`pvs, vgs, lvs`	Виведення інформації про атрибути фізичних, груп томів, логічних томів	`pvs`
`pvcreate, pvremove`	Створення, видалення фізичних томів	`pvcreate /dev/sda`
`vgcreate, vgremove`	Створення, видалення груп томів	`vgcreate vg01 /dev/sda`
`lvcreate`	Створення логічних томів	`lvcreate -n lv01 -l 100%VG vg01`
`lvextend`	Зміна розміру логічних	`lvextend`

	томів	/dev/vg01/lv01 -L+1000M
pvmove	Переміщення фізичних томів	pvmove /dev/sda /dev/sdb

За допомогою pvs, lvs, vgs можна створювати звіти про стан об'єктів LVM. Кожен рядок у звіті містить інформацію про один об'єкт. Висновок можна відфільтрувати за фізичними томами, групою томів, логічними томами, сегментами фізичних або логічних томів.

Незалежно від того, чи ви використовуєте pvs, lvs або vgs, вибрана команда за замовчуванням відображає стандартний набір полів, що можна перевизначити за допомогою різних опцій.

Аргумент -o дозволяє вибрати поля для виведення. Наприклад, стандартне виведення команди pvs виглядає так:

```
pvs
PV VG Fmt Attr PSize PFree
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G
```

Наступна команда покаже лише ім'я та розмір фізичних томів.

```
pvs -o pv_name,pv_size
PV PSize
/dev/sdb1 17.14G
/dev/sdc1 17.14G
/dev/sdd1 17.14G
```

Додаткове поле можна додати за допомогою "+" у комбінації з "-o".

Приклад додавання ідентифікатора UUID до стандартного звіту:

```
pvs -o +pv_uuid
PV VG Fmt Attr PSize PFree PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-01S9-X08M-mcpsVe
```

```
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-UqkCS
```

`--noheadings` сховає рядок заголовків, що використовується під час створення сценаріїв.

Наступний приклад використовує аргумент `--noheadings` у комбінації з `pv_name` для відображення списку всіх фізичних томів.

```
pvs --noheadings -o pv_name
/dev/sdb1
/dev/sdc1
/dev/sdd1
```

`--separator` – роздільник дозволяє відокремити поля один від одного.

У прикладі поля виведення команди `pvs` розділені знаком рівності.

```
pvs --separator =
PV=VG=Fmt=Attr=PSize=PFree
/dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
/dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
/dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

Для вирівнювання полів під час використання роздільника можна додатково вказати `--aligned`.

```
pvs --separator = --aligned
PV = VG = Fmt = Attr = PSize = PFree
/dev/sdb1 =new_vg=lvm2=a- =17.14G=17.14G
/dev/sdc1 =new_vg=lvm2=a- =17.14G=17.09G
/dev/sdd1 =new_vg=lvm2=a- =17.14G=17.14G
```

Для додавання до звіту списку несправних томів використовується параметр `-P` команд `lvs` та `vgs`.

Довідкові сторінки `pvs`, `vgs` та `lvs` містять повний перелік параметрів.

Поля групи томів можуть бути змішані з полями фізичного або логічного тому (або їх сегментів), але поля фізичного тому не можуть бути змішані з полями логічного тома. Наприклад, наступна команда покаже по одному фізичному тому у рядку:

```
vgs -o +pv_name
VG #PV #LV #SN Attr VSize VFree PV
new_vg 3 1 0 wz--n- 51.42G 51.37G /dev/sdc1
new_vg 3 1 0 wz--n- 51.42G 51.37G /dev/sdd1
new_vg 3 1 0 wz--n- 51.42G 51.37G /dev/sdb1
```

1.2 Завдання на лабораторну роботу

1. Створити віртуальну машину.

2. Запустити SSH-клієнт. У разі використання графічного клієнта запустити програму. У разі використання термінальних клієнтів – відкрити термінал.

3. Підключитися SSH до своєї віртуальної машини. У графічному клієнті здійснити необхідні налаштування. У термінальному – виконати команду підключення. Наприклад, для openssh: `ssh username@ip_address -p port_number`.

4. Створити пару SSH-ключів із налаштуваннями за замовчуванням. Користуючись розібраним у методичних вказівках алгоритмом створити пару ключів, не змінюючи значення за промовчанням.

5. Розмістити пару ключів у потрібних місцях. Користуючись розібраним у методичних вказівках алгоритмом розмістити пару ключів у місцях використання у авторизації.

6. Вимкнутись від ОС сервера та спробувати авторизуватися за ключом.

7. Створити файл `script.sh` та зробити його виконуваним. За допомогою команд «`touch`» та «`chmod +x`» створити у своїй домашній директорії файл скрипту та надати йому права на виконання.

8. Відкрити файл скрипта будь-яким редактором. Вибір редактора індивідуальний і залежить від цілей і особистих переваг.

9. Придумати та написати алгоритм створення дерева каталогів. За допомогою відомих логічних конструкцій чи команд написати у скрипті алгоритм створення дерева каталогів виду `/home/username/1/2/3`.

10. Запустити скрипт. Виконати свій скрипт та отримати потрібний результат.

11. Переглянути список змонтованих ФС. Подивитися вміст файлу "`/etc/fstab`", зробити висновок про те, які ФС монтуються при завантаженні і який тип вони мають, для чого використовуються.

12. Переглянути утилізацію змонтованих ФС. За допомогою команди «`df`» переглянути загальний та вільний обсяг ФС, змонтованих на даний момент до ОС.

13. Порахувати обсяг файлів у домашній директорії. За допомогою команди «`du`» порахувати обсяг всіх файлів, які розташовані у своїй домашній директорії.

14. Створити LVM із новим диском, підключеним до ВМ. Створити структуру як мінімум з одним Physical Volume, Volume Group та Logical Volume. Створити на ньому ФС `ext4`, яка монтуватиметься в директорію `/mnt/storage` під час завантаження ВМ.

15. Оформити звіт з лабораторної роботи згідно стандарту кафедри ПМІ.

ЛАБОРАТОРНА РОБОТА №2. БАЗОВІ ІНСТРУМЕНТИ. КОНСОЛЬ

2.1 Ознайомлення з базовими інструментальними засобами лабораторного практикуму

2.1.1 Теоретичні відомості

Лабораторний практикум виконується в віртуальному середовищі Red Hat Linux/Debian/Mint (див. Лабораторну роботу №1), доступ до якої здійснюється з робочого місця, яке функціонує в середовищі ОС Windows через протокол SSH.

При виконанні першої частини лабораторного практикуму Ви стаєте клієнтом сервера Linux і використовуєте вікно програми Putty як термінал сервера.

У ході виконання другої лабораторної роботи Вам належить оволодіти деякими інструментальними засобами, які будуть використовуватися Вами в усіх наступних роботах.

2.1.2 Порядок виконання першої частини лабораторної роботи

1. Вхід в систему.

У середовищі Windows запустіть програму Putty та зайдіть до віртуальної машини.

У сеансі роботи з Linux поточним (домашнім) каталогом є каталог: `/home/ім'я`, де `ім'я` – Ваше мережеве ім'я. До цього каталогу Ви маєте права читання, запису, виконання. Ви не маєте права запису до каталогів, які не є підкаталогами вашого домашнього каталогу.

2. Виконання команд.

Робота ведеться в режимі командного рядка. Стандартним запрошенням у системах Linux і Linux є символ '$'. Зазвичай команда має вигляд:

```
імя_команди [параметри] ... [Параметри] ...
```

Опції команд є флаговими параметрами. У Linux, як правило, прапори мають дві форми - коротку і довгу. Коротка форма передує символом - і кодується однією літерою. Довга форма передує двома символами - і кодується цілим словом або навіть фразою.

Не забувайте, що командна мова Linux чутлива до регістру. Для перших експериментів з командами використовуйте команди `ls`, `cd` і `pwd`. Команда `'ls -la'` виведе інформацію про вміст поточного каталогу. Команда `'cd ..'` переведе в батьківський каталог. Команда `'cd імя_подкаталога'` переведе у вказаний підкаталог поточного каталогу. Команда 'pwd' покаже, який каталог є поточним.

Якщо Ви "заблукаєте", подорожуючи по каталогах, команда 'cd' (без параметрів) поверне Вас у Ваш домашній каталог. Не забувайте, що в Linux символ "слеш" - роздільник імен каталогів нахилений праворуч: `'/'`!

3. Отримання підказки.

Стандартним засобом отримання підказки у Linux є команда `man`. Параметром команди `man` є `ім'я команди`, для якої подрібно отримати підказку. При введенні команди `man` на екран виводиться текст – опис заданої команди. Можна переміщатися по цьому опису вгору-вниз, використовуючи клавіші управління курсором і клавіші PageUp та PageDown. Для виходу з режиму команди `man` введіть символ `'!'` (Знак оклику).

У використовуваній нами версії Linux деякі розділи `man` переведені на російську мову. Зверніть увагу на те, що в більшості описів опції команд даються у версії POSIX і у версії GNU. POSIX є стандартом для ОС Linux, але оскільки ми користуємося ОС Linux, ми повинні вибирати версію GNU.

Альтернативним засобом отримання підказки в Linux є команда `info`. Параметром команди `info` також є `ім'я команди`, що нас цікавить. При введенні команди `info` без параметрів виводиться список розділів, які можна переглянути за допомогою команди `info`. Перегляд інформації в `info` виконується точно так само, як у `man`, крім того, `info` забезпечує елементи гіпертекстового режиму. Засвойте роботу з підказками – вони будуть потрібні ще неодноразово.

4. Збереження результатів.

Для тих робіт, в ході яких потрібно розробити і виконати команди та/або скрипти та продемонструвати їх виконання, використовуйте команду `script`, яка дозволяє створити протокол роботи користувача на терміналі. Рекомендується вводити команду `script` перед виконанням остаточної (звітної) версії створеної команди/скрипта.

Для того, щоб результати роботи накопичувалися у файлі протоколу, використовуйте команду `script` з опцією `-a`.

5. Зв'язок з колегами.

Визначте, хто працює в системі (за допомогою команди `who`). Обміняйтеся повідомленнями (за допомогою команди `mail`) з одним або кількома своїми товаришами. У наступній роботі буде потрібна кооперація з одним зі своїх колег. Виберіть собі партнера і поштою домовтеся з ним про подальшу співпрацю.

Іншим способом обміну повідомленнями є команда `write`. Правда, її можна використовувати лише в тому випадку, коли і відправник, і адресат працюють в системі. Спробуйте команду `write`, можливо, вона сподобається Вам більше.

6. Створення та редагування текстових файлів.

Редактор `VI`

В процесі роботи в системі вам необхідно буде створювати і редагувати текстові файли.

Всі ці дії можна виконати за допомогою екранного текстового редактора vi.

Для початку спробуємо створити новий файл, наприклад 'testvi':

```
$ Vi testvi
```

З'явиться порожній екран з курсором в першому рядку. Решта рядка (також порожні) будуть починатися з символу '~' (тильда). В останньому рядку буде повідомлення приблизно такого змісту:

```
testvi: new file: line 1.
```

Надалі цей рядок буде також використовуватися і для введення команд. Трохи перепочинемо від нашого файлу і розглянемо систему команд vi. Більшість команд – це поодинокі клавіші або комбінації клавіш, які виконують прості функції редагування. vi працює в двох основних режимах – в режимі "введення тексту" і в режимі "команд".

Після запуску vi опиняється в режимі "команд". Для переходу в режим "введення тексту" необхідно натиснути на клавішу 'a' або 'i' (звертайте увагу на регістр клавіш!). Після цього можна набирати текст. Кожну введену рядок слід, як це прийнято, завершувати натисканням клавіші [Enter].

Виконайте наступні дії. Натисніть клавішу 'a', перейдіть в режим "введення тексту" і наберіть 3 рядки:

```
Line 1
Line 2
Line 3
```

Для переходу в режим "команд" натисніть на клавішу Esc. Ця ж клавіша використовується для скасування не докінця набраної команди. Якщо команда введена неправильно, редактор повідомить про це одиночним звуковим сигналом.

Припустимо, необхідно вставити в початок другого рядка ще одне слово. Для цього в режимі "команд" перейдіть курсором на потрібний рядок, встановіть курсор в першу позицію і натисніть клавішу 'i'. Після цього вставте потрібний текст, наприклад, слово 'Insert':

```
Line 1
```

```
Insert Line 2
Line 3
```

Для переходу в режим "команд" знову натисніть клавішу 'Esc', призначення якого вже зрозуміле. Якщо ж з якихось причин заплуталися в якому режимі знаходитесь, натисніть два рази поспіль Esc. Редактор видасть звуковий сигнал, повідомляючи таким чином, що ви перебуваєте в режимі "команд".

Щоб зробити редактор трохи «балакучішим», в режимі "команд" введіть наступну команду в нижньому рядку:

```
: Set verbose showmmode [Enter]
```

У даному прикладі символ ' : ' означає ознаку введення команди.

Після виконання цієї команди редактор буде повідомляти про досягнення кінця рядка та файлу, перехід в режим "команд" і т.д.

Для переміщення курсору по тексту в режимі "команд" можна використовувати клавіші управління курсором, більшість з них наведена у таблиці 2.1.

Таблиця 2.1 – Команди управління курсором

Команда	Призначення
h або ПРОБІЛ	Переміщення позиції курсору на один символ вліво
l або BACKSPASE	Переміщення позиції курсору на один символ вправо
j	Переміщення курсор вниз на один рядок.
k	Переміщення курсор вгору на один рядок.
^	Перемістити курсор на початок першого слова поточного рядка.
$	Перемістити курсор у кінець поточного рядка.

Продовження таблиці 2.1

|	Перемістити курсор в першу позицію поточного рядка.
w	Перемістити курсор вперед на початок наступного слово.
b	Перемістити курсор назад на початок поточного слова.
e	Перемістити курсор у кінець цього слова.
числоПРОБІЛ	Перемістити курсор на вказане число позицій вправо
числоBACKSPASE	Перемістити курсор на вказане число позицій вліво.
числоG	Перемістити курсор у вказаний рядок.
число |	Перемістити курсор у вказану позицію поточного рядка.

Для переходу в режим "введення тексту" можна використовувати наступні команди з таблиці 2.2.

Таблиця 2.2 – Команди для переходу в режим "введення тексту"

Команда	Призначення
a	Вставити текст після курсору.
i	Вставити текст перед курсором.
o	Вставити новий рядок після поточного.
O	Вставити новий рядок перед поточним.

Поточним будемо називати рядок у якому розташований курсор, а поточним символом – символ в якому знаходиться курсор.

У режимі команд можна виконувати редагування набраного тексту за допомогою наступних команд в таблиці 2.3.

Таблиця 2.3 – Команди для редагування набраного тексту

Команда	Призначення
x	Видалити поточний символ.
dd	Видалити поточний рядок.
dw	Видалити поточне слово.
r	Замінити поточний символ на символ, набраний слідом за 'r'.

Якщо необхідно поміняти місцями рядки 3 і 2 нашого файлу – для цього перейдіть в режим "команд" (Esc), встановіть курсор в другий рядок, введіть комбінацію 'dd`. Рядок 'Line 2' буде вилучено (поміщено в буфер), і весь текст зміститься вгору. У поточному рядку з'явиться рядок 'Line 3':

```
Line 1
Line 3
```

Далі натисніть клавішу 'p'. Після поточного рядка ('Line 3') з буфера буде відновлений і стане поточним рядок 'Line 2':

```
Line 1
Line 3
Line 2
```

Відповідно для вставки віддаленого рядка перед поточним можна використовувати команду 'P'. Якщо вам необхідно зберегти в буфері рядок без його видалення, використовуйте команду 'yy'. Надалі цей рядок можна скопіювати в інше місце файлу. Для роботи з буфером і для переміщення (копіювання) рядків можна використовувати наступні команди з таблиці 2.4.

Таблиця 2.4 – Команди для работи з буфером і для переміщення (копіювання) рядків

Команда	Призначення
yw	Зберегти слово в буфері.
число yw	Зберегти вказане число слів у

	буфері.

Продовження таблиці 2.4

`yy`	Зберегти поточний рядок в буфері.
`число yy`	Зберегти вказане число рядків у буфері.
`p`	Копіювати і помістити інформацію з буфера після поточного рядка.
`P`	Копіювати і помістити інформацію з буфера перед поточної рядком.

Існує ще одна цікава команда - `'.'` (Крапка). Вона виконує останню введену команду. Наприклад, якщо за допомогою команди `'dd'` був видалений рядок, то натискання на клавішу `'.'` призведе до видалення наступного рядка. Якщо рядок був поміщений у вікно редагування з буфера по команді `'p'`, то натискання на `'.'` призведе до переміщення у вікно редагування ще однієї копії рядка.

При запуску `vi` можна вказати параметри, наведені в таблиці 2.5.

Таблиця 2.5 – Команди при запуску та виході з редактора `vi`

Команда	Призначення
`vi`	Редагує тимчасовий файл, якому при збереженні тексту необхідно дати ім'я.
`vi +45 file`	Переходить на рядок з номером 45.
`vi + / word file`	Шукає перше входження слова `'word'`
`$ vi-c cmd`	Виконує команду `cmd` негайно після початку сеансу редагування.

Продовження таблиці 2.5

`$ vi-r`	Відновлює вказані файли, якщо відбувся аварійний вихід з редактора або раптове завершення системи. Якщо файли не визначено, вивести список файлів, які можуть бути відновлені.
`: W`	Зберегти текст без виходу з редактора
`: W ім'я_файлу`	Зберегти текст у вказаному файлі.

`: Wq`
`або`
`: X`
	Зберегти текст і вийти з редактора.

`: Q`	Вийти з редактора. Якщо файл був модифікований, вам буде запропоновано для виходу без збереження використати команду: q!
`: Q!`	Вийти з редактора без збереження тексту.

Для більшості версій `vi` у тексті, яким потрібно замінити вихідним, за останнім знаком повинен стояти $. І коли досягається цей останній знак, режим заміни переходить у режим вставки, а частина вихідного тексту, що залишилася перед курсором, починає просто зрушуватися вправо. Це дає прекрасну можливість швидко виправити слово або рядок, а потім спокійно продовжувати введення. В таблиці 2.6 наведені параметри `vi`.

Таблиця 2.6 – Параметри `vi`

Параметр	Опис
`ai`	Автоматичний відступ: знову створювані рядки вирівнюються за відступом попереднього
`noai`	Автоматичний відступ відключений (за замовчуванням)
`aw`	Автоматичний запис: поточний вміст буфера записується при виводі з редактора або при переході до наступного файлу за командою :n
`noaw`	Автоматичний запис відключений (за замовчуванням)
`flash`	При помилці замість звукового сигналу спалахує екран (за замовчуванням)
`noflash`	Звуковий сигнал при помилці
`ic`	Не враховувати регістр при пошуку й порівнянні із символами шаблону
`noic`	Враховувати регістр (за замовчуванням)
`lisp`	Встановити режим автоматичної вставки відступів, прийнятих при редагуванні Lisp-Програм
`nolisp`	Встановити режим вирівнювання початку рядка за початком попереднього (за замовчуванням)
`magic`	Використовувати повний набір символів шаблону при порівнянні зі зразком (за замовчуванням)
`nomagic`	Обмежує набір символів шаблону символами і $
`mesg`	Дозволяє повідомленням, відправленим вам іншими користувачами, з'являтися на екрані вашого термінала
`nomesg`	Скасовує дозвіл появи тексту на вашому терміналі
`nu`	Виводить номери рядків уздовж лівого краю екрана
`nonu`	Номера рядків не виводяться (за замовчуванням)

Додаткові відомості наведені в таблиці 2.7.

Таблиця 2.7 – Додаткові відомості по командам

Назва	Визначення	Синтаксис	Опис	Опції
`ls`	Виведення вмісту каталогу	`ls [опції] [файл...]`	Команда `ls` спочатку виводить список усіх файлів (не каталогів), перерахованих у командному рядку, а потім виводить список усіх файлів, що перебувають у каталогах, перерахованих у командному рядку. Якщо не зазначено жодного файлу, то за замовчуванням аргументом призначається `'.'` (поточний каталог). Опція `-d` змушує `ls` не вважати аргументи-каталоги каталогами. Будуть відображатися тільки файли, які не починаються з `'.'` або всі файли, якщо задана опція `-a`. Результати друкуються в стандартний пристрій виведення, по одному файлу на рядок, якщо за допомогою опції `-C` не заданий багатостовбцеве виведення. Кожний список файлів (для файлів, які не є каталогами й для кожного каталогу, що містить список файлів) сортується окремо в алфавітній послідовності.	`-l` На додаток до імені кожного файлу, виводяться тип файлу, права доступу до файлу, кількість посилань на файл, ім'я власника, ім'я групи, розмір файлу в байтах і часовий штамп (час останньої модифікації файлу, якщо не задане інше). `-a` Видавати всі файли в каталогах, включаючи всі файли й підкаталоги, імена яких починаються з `'.'`. `-d` Видавати імена каталогів, начебто вони звичайні файли, замість того, щоб показувати їхній вміст. `-L` Видавати інформацію про файли, на які вказують символічні посилання, замість інформації про самі

				символічні посилання. `-R` Рекурсивно видавати список вмісту всіх каталогів. `-h` Додавати до кожного розміру файлу букву розміру, наприклад, M (мегабайт). `-X` Робити сортування за абеткою по розширеннях файлів (символи після останньої `'.'`); файли без розширень будуть показані першими. `-S` Робити сортування по розміру файлу, замість сортування за алфавітом. Таким чином, найбільші файли будуть показані спочатку. `-c` Сортувати вміст каталогу відповідно до часу зміни стану файлу. Якщо за допомогою опції `-l` заданий довгий формат, то видавати час зміни стану файлу замість часу його модифікації. `-t` Сортувати

				за часом останньої модифікації замість того, щоб робити сортування за алфавітом. Самі свіжі файли будуть відображатися першими. `-u` Сортувати за часом останнього доступу до файлу, замість часу останньої модифікації.
`cd`	Зміна поточного каталогу	`cd [каталог]`	`cd` змінює поточний каталог на `каталог`. Ім'я `каталог` може задаватися абсолютним (від кореневого каталогу) - у цьому випадку воно починається із символу `'/'` - або відносним (від поточного каталогу) - у цьому випадку воно починається із символів `'./'` або `'../'`. Якщо каталог не зазначений, то поточним стає "домашній" каталог користувача, обумовлений значенням змінної оточення $HOME.	
`pwd`	Виведення імені поточного каталогу	`pwd`	Команда `pwd` видає ім'я поточного каталогу.	
`mkdir`	Створення каталогу	`mkdir [опції] каталог`	Команда `mkdir` створює каталоги із заданими іменами. За замовчуванням	`-m права` Встановлює права доступу до

			права доступу до каталогів встановлюються в 0777 (`a+rwx').	створюваних каталогів. Ці права можуть бути задані або в символьному виді, або у вигляді восьмеричного числа
rmdir	Видалення порожніх каталогів	rmdir [опції] каталог	Команда rmdir видаляє порожні каталоги. Якщо який-небудь із аргументів каталог не вказує на існуючий порожній каталог, то буде видане повідомлення про помилку.	-p Якщо каталог включає більш, ніж один компонент шляху, то видаляється каталог, потім убирається останній компонент шляху й видаляється каталог, що вийшов, і т.д. доти, поки всі компоненти не будуть вилучені. Таким чином, команда rmdir -p a/b/c еквівалентна rmdir a/b/c; rmdir a/b; rmdir a.
man	Форматування й відображення онлайнових довідкових сторінок	man [розділ] ім'я...	Команда man виконує форматування й відображення онлайнових довідкових сторінок Linux. Якщо заданий розділ, то man шукає тільки в заданому розділі керівництва. ім'я - це звичайне ім'я сторінки керівництва, яке, як правило, є іменем	

			команди, функції або файлу. Наявні довідкові сторінки розбиті на кілька розділів. Найважливішими є розділи: 1 - команди Linux; 2, 3 - системні виклики Linux; 4, 5 - формати файлів Linux. Коли команда відображає сторінку підказки, у нижньому рядку екрана виводиться запрошення `man` - символ `':'`. Після запрошення можна вводити внутрішні команди `man`. У короткому керівництві слід згадати тільки дві внутрішні команди `man`: `h` одержання докладної інформації про внутрішні команди `man`; `q` вихід з `man` або перехід до наступної сторінки, якщо команда `man` була введена із вказівкою декількох імен команд. Рухатися по відображуваній сторінці можна за допомогою клавіш керування курсором.	
`info`	Відображення онлайнових довідкових сторінок	`info ім'я...`	Команда `info` виконує форматування й відображення онлайнових довідкових сторінок Linux. У текстах, відображуваних командою, можуть бути набори рядків,	

			озаглавлені "* Menu", кожний рядок такого набору починається із символу "*". Вибравши курсором пункт меню й нажавши клавішу Enter, можна одержати сторінку підказки по цьому пункту. Незалежно від положення курсору, після запрошення можна вводити внутрішні команди info. У короткому керівництві слід згадати тільки дві внутрішні команди info: h одержання докладної інформації про внутрішні команди info; q вихід з info або перехід до наступної сторінки, якщо команда info була введена із вказівкою декількох імен команд. Рухатися по відображуваній сторінці можна за допомогою клавіш керування курсором.	
script	Протоколю-вання сеансу	script [-a] файл	Команда script починає "вкладений" сеанс і протоколює все термінальне введення й виведення у заданому файлі. Завершення вкладеного сеансу й виконання команди script відбувається по натисканню комбінації клавіш Ctrl+D.	-a додавання протоколу нового сеансу до вмісту файлу, якщо ця опція не задана, то файл створюється заново.

who	Хто в системі	who [опції]	Команда who повідомляє ім'я користувача, ім'я термінальної лінії, астрономічний час початку сеансу, тривалість бездіяльності термінальної лінії з моменту останнього обміну, ідентифікатор процесу для кожного з користувачів, що працюють у системі. Повідомлення команди who мають наступний формат: NAME STATE LINE TIME IDLE PID COMMENT де NAME - вхідне ім'я користувача; LINE - ім'я термінальної лінії, під яким вона фігурує в каталозі /dev; TIME - час початку сеансу; IDLE - час (години та хвилини), що протік з останнього моменту активізації даної лінії. Крапка (.) свідчить про те, що це діючий термінал. PID - ідентифікатор процесу інтерпретатора shell, що обслуговує даного користувача; COMMENT - коментар, що характеризує дану лінію.	-H відображення заголовків стовпців у виведеній інформації -i відображається поле IDLE -q відображення тільки імен і кількості користувачів, що працюють у системі в цей момент; усі інші опції при цьому ігноруються -T аналогічно -s, але при цьому відображається також поле STATE, як: + термінал, на який можна передавати повідомлення - термінал, на який не можна передавати повідомлення ? термінал несправний -s виводяться тільки поля NAME, LINE і TIME; це опція за замовчуванням.
write	Передача повідомлення іншому користувачеві.	write *адресат*	*Адресат* задається як мережне ім'я користувача. Після запуску команда write встановлює зв'язок з адресатом і переходить у режим	

			очікування введення. У момент установки зв'язки на термінал адресата виводиться повідомлення: `Message from` *`відправник`* `...` Відправник вводить будь-який текст, який відображається на терміналі одержувача. Відправник закінчує повідомлення натисканням комбінації клавіш Ctrl+D на початку рядка. В адресата закінчення повідомлення ідентифікується рядком: `EOF` Одержувач може заблокувати/розблокувати виведення повідомлень на свій екран за допомогою команди `mesg`. При спробі передати повідомлення на заблокований термінал відправник одержує діагностику: `write:` *`адресат`* `has messages disabled`	
`tee`	Відгалуження каналу	`tee [опції]... [файл]...`	Команда `tee` переписує стандартне введення на стандартне виведення і робить копії у файлах. Ознакою закінчення введення є комбінація клавіш Ctrl+D.	`-a` додавати виведену інформацію у файли, а не переписувати їх з початку.
`cat`	Злиття й виведення файлів	`cat [-опції] файл ...`	Команда `cat` по черзі читає зазначені файли й видає їхній вміст на стандартне виведення. Так, наприклад, `cat f` роздруковує вміст файлу `f, a cat f1 f2 > f3` зливає	`-b` Нумеруються непусті рядки файлу. `-s` Нумеруються всі рядки файлу. (Поле

			перші два файли й поміщає результат у третій. Щоб додати файл `f1` до файлу `f2`, треба виконати команду `cat f1 >> f2`. Якщо не зазначений жоден файл або серед аргументів зустрівся -, команда `cat` читає дані зі стандартного введення.	номера виділяється від тексту символом табуляції). `-v` Візуалізація недрукованих символів. Керуючі символи зображуються у вигляді `X` (CTRL+X); символ `DEL` (восьмеричне 0177) - у вигляді ?. Символи, що не входять у набір ASCII (тобто з восьмим бітом, установленим в 1) видаються у вигляді `M-x`, де `x` - обумовлений молодшими 7-ми бітами символ. З опцією `-v` можна використовувати наступні опції: `-t` Візуалізація символів табуляції у вигляді `I`. `-e` Візуалізація символів переводу рядка у вигляді $ (рядок при цьому все-таки переводиться). Якщо опція `-v` не зазначена, то опції `-t` і `-e` ігноруються.
`vi`	Текстовий	`vi ім'я_файлу`	Командний - у цьому	

	редактор		режимі можна переміщатися по файлу й виконувати команди над текстом, що редагують.
			Команди викликаються ЗВИЧАЙНИМИ ЛАТИНСЬКИМИ БУКВАМИ.
			Уведення тексту - у цьому режимі звичайні латинські букви будуть вставлятися в текст.
			Режим рядкового редактора `vi` використовується для керування файлами (типу зберегти файл, зачитати файл і т.д.)
			VI у КОМАНДНОМУ РЕЖИМІ.
			ЩОБ ВИЙТИ З ФАЙЛУ БЕЗ ЗБЕРЕЖЕННЯ, натисніть:
			`ESC : q ! Enter`
			щоб вийти з файлу, зберігши зміни, натисніть:
			`ESC : w ! Enter`
			`ESC : q Enter`
			вийти з файлу зі збереженням, однією командою:
			`ESC : wq Enter`
			для переходу В РЕЖИМ УВЕДЕННЯ потрібно нажати команди типу:
			`i` вставляти тут
			`A` вставляти з кінця рядка
			`Cw` заміняти поточне слово
			`ESC` для ПОВЕРНЕННЯ В КОМАНДНИЙ РЕЖИМ
			`CTRL-[` для повернення в	

			командний режим. Для переходу В РЕЖИМ КЕРУВАННЯ ФАЙЛАМИ потрібно нажати : Рухатися по файлу можна командами: `h,j,k,l` вліво, вниз, нагору, вправо Ctrl-F на сторінку вниз Ctrl-B на сторінку нагору Підведіть курсор до потрібного місця й натисніть: `i` перехід у режим введення та вводите необхідний текст `ESC` припинити введення, перейти в командний режим. Підведіть курсор до непотрібного місця й натисніть `x` вилучити символ `dd` вилучити рядок. Ще парочка корисних команд: `o` вставляти з нового рядка (під поточним рядком) `a` у режим введення ЗА курсором `5yy` запам'ятати 5 рядків Підведіть курсор до потрібного місця: `p` вставити запам`ятовані рядка під курсором `P` вставити запам`ятований рядок НАД курсором `J` склеїти два рядки /Шаблон пошуку Enter пошук: `n` повторити пошук.	
`cp`	Копіювання	`cp [опції] файл`	Команда `cp` копіює файли або каталоги.	`-d` Копіює символьні

	файлів і каталогів	шлях cp [опції] файл каталог	Якщо останній аргумент є існуючим каталогом, то cp копіює кожний вихідний файл у цей каталог (зберігаючи імена). А якщо ні, то, якщо задане тільки два файли, то cp копіює перший файл у другий. (Так, cp -R /a /b буде копіювати /a в/b/a і /a/x в/b/a/x у випадку, якщо /b уже існує, але ця ж команда буде копіювати /a в /b і /a/x в /b/x, якщо /b не існує). За замовчуванням cp не копіює каталоги (див. опцію -R).	посилання як символьні посилання, а не файли, на які вони вказують, і зберігає тверді посилання між вихідними файлами в копіях. -f Видаляє існуючі файли, у які відбувається копіювання, і не задає питань перед тим, як це зробити. -i Запитує, чи потрібно перезаписувати існуючі звичайні файли. -l Робить тверді посилання замість копіювання звичайних файлів (не каталогів). -R Копіювати каталоги рекурсивно, зберігаючи спеціальні файли (див. -r вище). -v Виводити ім'я кожного файлу перед його копіюванням.
unlink	Викликає системну функцію	unlink file	Викликає системну функцію unlink для видалення зазначеного файлу file.	

	unlink для видалення зазначеного файлу.			
rm	Видалення файлів або каталогів	rm [опції] файл...	Команда rm видаляє кожний заданий файл. За замовчуванням каталоги не видаляються, але якщо задана опція -r, то буде видалятися все дерево каталогів нижче заданого каталогу, включаючи й заданий каталог (без обмеження на глибину дерева).	-f Ігнорувати неіснуючі файли й ніколи не запитувати підтвердження на видалення. -i Видавати запит на видалення кожного файлу. (Прийнята за замовчуванням). -r Рекурсивно видаляти вміст каталогів. -v Видавати ім'я кожного файлу перед його видаленням.
ln	Створення посилання на файл.	ln [-f] файл1 [файл2 ...] цільовий_файл	Команда ln робить цільовий_файл посиланням на файл1. Файл1 не повинен збігатися із цільовим_файлом. Якщо цільовий_файл є каталогом, то в ньому створюються посилання на файл1, файл2,... з тими ж іменами. Тільки в цьому випадку можна вказувати кілька вихідних файлів. Якщо цільовий_файл існує й не є каталогом, його старий вміст губиться.	-f видалення існуючого цільового файлу -s створення символічного посилання (за замовчуванням створюється тверде посилання)

chmod	Зміна режиму доступу до файлів.	chmod режим файл...	Команда chmod змінює права доступу до зазначених файлів (серед яких можуть бути каталоги) відповідно до зазначеного режиму. Режим може бути заданий в абсолютному або символьному виді. *Абсолютний вид* - восьмеричне число, що є поразрядным АБО наступних режимів (названі не всі режими): 00400 Доступний для читання власником. 00200 Доступний для запису власником. 00100 Доступний для виконання (у випадку каталогу - для перегляду) власником. 00040 Доступний для читання членами групи. 00020 Доступний для запису членами групи. 00010 Доступний для виконання (перегляду) членами групи. 00004 Доступний для читання іншими користувачами. 00002 Доступний для запису іншими користувачами. 00001 Доступний для виконання (перегляду) іншими користувачами. *Символьний вид* заснований на однобуквених позначеннях, які визначають клас доступу й права доступу для членів даного класу. Права	

			доступу до файлу залежать від ідентифікатора користувача й ідентифікатора групи, у яку він входить. Режим у цілому описується в термінах трьох послідовностей, по три букви в кожній: Власник (u) Група (g) rwx rwx Для завдання режиму доступу в символьному виді використовується синтаксис: `[кому] операція права` Частина `кому` є комбінація букв `u`, `g` і `o` (власник, члени групи та інші користувачі відповідно). Якщо частина кому опущена або зазначено `a`, то це еквівалентно `ugo`. Операція може бути: + (додати право), - (позбавити права), = (у межах даного класу привласнити права абсолютно, тобто додати зазначені права й відняти незазначені). Права - будь-яка осмислена комбінація наступних букв (не все): `r` Право на читання. `w` Право на запис. `x` Право на виконання (пошук у каталозі). Вилучити частина права можна тільки якщо операція є = (для позбавлення всіх прав). Якщо треба зробити більш одного вказівки	

			про зміну прав, то при використанні символьного виду в правах не повинно бути пробілів, а вказівки повинні розділятися комами. Наприклад, команда `chmod u+w,go+x f1` додасть для власника право писати у файл `f1`, а для членів групи й інших користувачів - право виконувати файл. Права встановлюються в зазначеному порядку. Змінити режим доступу до файлу може тільки його власник або суперкористувач. Для перегляду прав доступу й контролю при їхній зміні використовується команда `ls` із прапором `-l`.	
`chown`	Зміна власника й групи файлів.	`chown [опції] користувач[:група] файл...`	Команда `chown` змінює власника й/або групу для кожного заданого файлу. У якості імені власника/групи береться перший аргумент, що не є опцією. Якщо задане тільки ім'я користувача (або числовий ідентифікатор користувача), то даний користувач стає власником кожного із зазначених файлів, а група цих файлів не змінюється. Якщо за іменем користувача через двокрапку слідує ім'я групи (або	`-R` Рекурсивна зміна власника для каталогів і їх вмісту.

			числовий ідентифікатор групи), без пробілів між ними, то змінюється також і група файлу.	
`ps`	Висновок інформації про стан процесів	`ps [опції]`	Команда `ps` виводить у стандартне виведення інформацію про поточний стан процесів.	`-a` усі процеси, крім лідерів груп і процесів, не асоційованих з терміналом. `-d` усі процеси, крім лідерів груп. `-e` усі процеси. `-g`*список* вибирати процеси за *списком* лідерів груп. `-p`*список* вибирати процеси за *списком* ідентифікаторів процесів. `-t`*список* вибирати процеси за *списком* терміналів. `-u`*список* вибирати процеси за *списком* ідентифікаторів користувачів. `-f` генерувати повний лістинг. `-l` генерувати лістинг у довгому форматі.

2.1.3Завдання до першої частини лабораторної роботи

1. Створіть у Вашому домашньому каталозі два текстових файлу, вміст яких визначається самостійно. Імена файлів можуть бути довільними. Ці файли будуть використовуватися в наступних роботах. У Linux є багатий набір засобів введення-редагування текстів.

а) Створіть файл, що містить Текст1 індивідуального завдання, за допомогою повноекранного текстового редактора vi.

б) Створіть файл, що містить Текст2 індивідуального завдання, за допомогою команди tee. Потім переглянути вміст файлу (за допомогою, наприклад, команди cat) і виправити в ньому помилки за допомогою текстового редактора ed.

2 Виконати команди `help, ls, cd, pwd, mkdir, rmdir`.

3 Вивчити описи цих команд за допомогою інструкцій `man` і `info`.

4 Вивчити опису команди `script`, запротоколювати з її допомогою виконання пункту 3.

5 Встановити зв'язок із сусідами, використовуючи команди `who`, `write`, запротоколювати переписку.

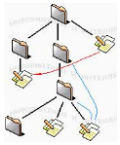
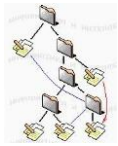
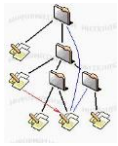
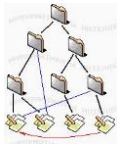
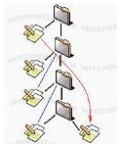
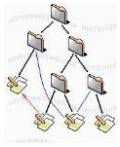
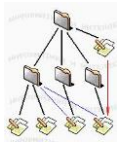
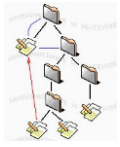
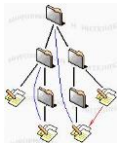
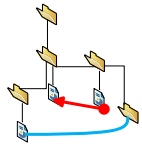
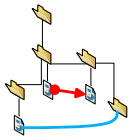
6 Створити текстовий файл із довільним змістом за допомогою команди `tee`. Переглянути вміст файлу за допомогою команди `cat` і виправити в ньому помилки за допомогою текстового редактора `vi`.

7 Вивчити довідку на команди `cp`, `unlink`.

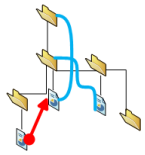
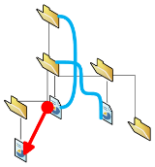
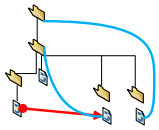
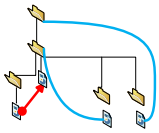
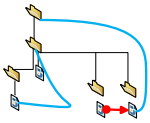
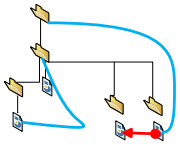
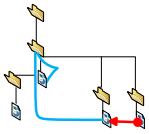
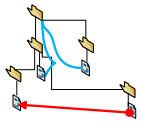
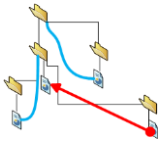
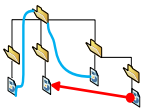
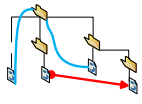
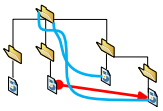
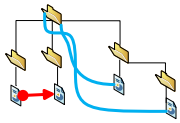
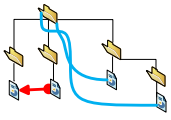
8 Зробити копію файлу командою `cp`, вилучити її командою `unlink`, запротоколювати ці дії.

9 Створіть структуру каталогів відповідно до варіанта в таблиці 2.8. Чорними лініями представлена вкладеність файлів/підкаталогів у каталоги. Синіми лініями представлені посилання. Червоними лініями - символічні посилання. Стрілка на червоній лінії вказує на цільовий файл посилання. Файли створюються копіюванням раніше створеного файлу командою `cp` із внесенням у копії деяких змін. Посилання створюються командою `ln`, символічні посилання – цією ж командою, але із ключем `-s`:

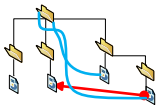
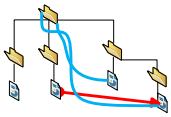
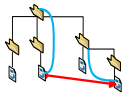
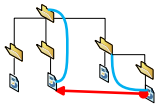
Таблиця 2.8 –Варіанти завдань

Варіант 1. 	Варіант 16. 
Варіант 2. 	Варіант 17. 
Варіант 3. 	Варіант 18. 
Варіант 4. 	Варіант 19. 
Варіант 5. 	Варіант 20. 
Варіант 6 	Варіант 21 

Продовження таблиці 2.8

Варіант 7 	Варіант 22 
Варіант 8 	Варіант 23 
Варіант 9 	Варіант 24 
Варіант 10 	Варіант 25 
Варіант 11 	Варіант 26 
Варіант 12 	Варіант 27 
Варіант 13 	Варіант 28 

Продовження таблиці 2.8

Варіант 14	Варіант 29
	
Варіант 15	Варіант 30
	

10 Для всіх варіантів виконати наступні дії:

10.1. Створити посилання (сині лінії).

10.2. Створити символічні посилання (червоні лінії).

10.3. Зберегти протокол виконаних дій.

10.4. Провести ряд експериментів, що ілюструють доступ до файлів по основним іменам, по посиланнях і по символічних посиланнях. Для доступу використовувати команду `cat` або редактор `vi`.

10.5. Провести ряд експериментів, що ілюструють реакцію системи на видалення файлу, на який є посилання, і файлу, на який є символічні посилання. Перевіряти результати командою `ls -la`.

10.6. Знищити створені підкаталоги й файли в них, використовуючи команди `rmdir` і `unlink`, зберігши, однак, файл, створений у пункті 10.3 і одну його робочу копію в домашньому каталозі.

10.7. Вивчити довідку до команд `chmod` і `chown`.

10.8. Відкрити для своєї групи доступ до свого домашнього каталогу - для пошуку в каталозі й до робочої копії файлу в домашньому каталозі - для читання й запису.

10.9. Послати своєму партнерові повідомлення про відкриття доступу, указавши в ньому ім'я свого каталогу й файлу в ньому.

10.10. Одержавши від свого партнера аналогічне повідомлення, виконати спробу читання файлу в каталозі партнера, а потім - внесення змін у цей файл.

10.11. Послати своєму партнерові повідомлення про те, що в його файл внесені зміни.

10.12. Одержавши від партнера аналогічне повідомлення, прочитати свій файл і знайти в ньому зміни, зроблені партнером.

10.13. Закрити доступ до свого домашнього каталогу.

10.14. Зберегти протокол виконання пунктів 10.8 - 10.13.

10.15. Вивчити довідку до команди `ps`, виконати її із ключами `-a`, `-e`, `a`, `x`, `ax`, записати результати у файл, наприклад: `ps -e > ps.log`.

10.16. Вивчити довідку до файлового менеджера `Midnight Commander`, запустити його, вивчити перелік доступних команд, комбінації клавіш для виконання часто застосовуваних команд, особливості вбудованого текстового редактора.

2.1.4 Контрольні запитання до першої частини лабораторної роботи

1. Як отримати доступ до терміналу Linux?
2. Як вводити команди Linux?
3. Як отримати підказку по командах Linux?
4. Як зберегти результати роботи?
5. Як обмінюватися повідомленнями зі своїми товаришами, які працюють в системі?
6. Як створювати і редагувати текстові файли в середовищі Linux?
7. Як отримати доступ до ваших файлів, створеним в Linux, із середовища Windows?
8. Які ключі команди `ls` Ви знаєте? Що вони дають?
9. Чим відрізняються `man` і `info`? Як з ними працювати?
10. Команда `script` - призначення й застосування.
11. Як обмінюватися повідомленнями з користувачами сервера Linux? Як блокувати приймання повідомлень?
12. Команди `tee` і `cat`. Призначення й застосування. Чим `cat` відрізняється від `more` і `less`?

13. Основні команди редактора `vi`.

14. Посилання й символічне посилання. Створення й застосування.

15. Створення й копіювання файлів і папок в Linux.

16. Переміщення й видалення файлів і папок в Linux.

17. Команди `chmod` і `chown`. Призначення й застосування.

18. Які права доступу Ви маєте до свого домашнього каталогу, каталогів `/home` і `/` ?

19. Команда `ps`. Призначення й застосування. Ключі команди.

20. Особливості виклику команд `Midnight Commander` через комбінації клавіш.

21. Особливості вбудованого текстового редактора файлового менеджера `Midnight Commander`.

22. Які комбінації клавіш `Midnight Commander` можна було застосувати для виконання цієї лабораторної роботи?

2.2 ОС Linux. Текстовий режим функціонування

2.2.1 Теоретичні відомості

Продовжимо роботу в консольному режимі. Багато користувачів, а тим паче спеціалістів, надають йому перевагу. Є низка корисних програм та команд, які можуть функціонувати або лише у текстовому режимі , або як у текстовому, так і у графічному режимах. Це пов'язано з тим, що із самого початку Linux розроблялась як система, що успадковувала головні риси ОС Linux, яка працює здебільшого у текстовому режимі.

Під текстовим режимом розуміють введення команд користувачем лише з командного рядка, для чого їх потрібно спочатку набрати на клавіатурі. Прикладом консольної команди для запуску програми менеджера файлів MC є `mc`.

Розглянемо три варіанти запуску сеансу роботи ОС Linux у текстовому режимі:

• реєстрація у текстовому режимі (Session — Failsave);

• використання спеціальних програм-емуляторів терміналу;

• переведення на іншу віртуальну консоль.

Щоб запустити текстовий режим з графічної оболонки, потрібно за допомогою пункту меню `Виконати` ввести назву відповідної програми, наприклад, `xterm`, `konsole` тощо. Програма `konsole` є стандартною для KDE.

В ОС Linux реалізована можливість одночасної роботи на одному комп'ютері декількох користувачів — це багатокористувацька система. Завдяки концепції віртуальних консолей користувачам не потрібно кожного разу перереєстровуватись. Для переходу на консоль з номером n слід натиснути `Ctrl + Alt + Fn`. Під час переходу на іншу консоль користувач автоматично потрапляє у текстовий режим, де необхідно увести свій логін (`localhost login`) та пароль (`Password`) . Після закінчення сеансу роботи потрібно виконати логоф, увівши команду `logout`. Для повернення у графічну оболонку треба натиснути `Alt + F8`. У текстовому режимі можна виконувати практично усі дії, як і у графічному. Однак для цього потрібно знати відповідні команди. Деякі команди були розглянуті в попередній частині роботи. Згадаємо їх ще раз і продовжимо знайомитись з іншими.

Оскільки команд є багато і їх усіх запам'ятати складно, варто користуватися командою `man <назва команди>`, щоб [мати довідку про призначення, синтаксис та дію тієї чи іншої команди. Щоб запустити набрану на клавіатурі команду необхідно натиснути на клавішу Enter. Команда негайно буде виконана. Якщо в тексті команди допущено помилку, система виведе повідомлення: команда не знайдена (`command not found`). Зауважимо, що команди набирають у командному рядку, що містить запрошення системи у вигляді символу „$". Загальний вигляд команди такий:

```
<назва команди> <список опцій> < список параметрів>
```

Назва команди складається з декількох малих латинських літер. Назва опції — це одна літера, перед якою є символ мінус. Опцій у списку може бути 0, 1 або декілька. Якщо опцій декілька, їх можна записувати підряд з одним символом мінус на початку. Команди можуть мати багато опцій. Повний список опцій для певної команди можна знайти у довідниках або в `help`-описах команди. Одні й ті ж опції в різних командах мають різне призначення. Параметри, якщо вони є, наприклад, назви файлів, на які поширюється дія команди, записують через пропуск.

Розглянемо основні команди ОС Linux.

Сервісні команди та програми

Сервісні команди та програми призначені для налаштування екрану та параметрів системи. Розглянемо деякі у таблиці 2.9.

Таблиця 2.9 – Сервісні команди та програми

Команда	Дія команди
`clear`	Очистити екран
`who`	Відобразити імена користувачів, які працюють у мережі у цей час
`write`	Надіслати повідомлення іншим користувачам
`date`	Вивести на екран або змінити значення системної
`df`	Вивести дані про розподіл дискового простору
`free`	Вивести повідомлення про розподіл пам'яті
`be`	Здійснити перехід у режим калькулятора
`cal`	Відобразити календар дат
`me`	Запустити менеджер файлів
`passwd`	Змінити пароль користувача
`reboot`	Перезавантажити систему

Деякі команди мають параметри. Наприклад, команда `cal` виведе на екран календар за поточний місяць, а `cal 2015` — за 2015 рік. Тут 2015 є параметром команди `cal`.

Деякі команди для роботи з файлами. Поняття про жорсткі та символічні посилання.

Розглянемо у таблиці 2.10 команди роботи з файлами.

Таблиця 2.10 – Команди для роботи з файлами

Назва команди	Дія команди
less або more	Переглянути файл
cat	Об'єднати декілька файлів в один
ep	Копіювати файли
mv	Перейменувати файл
rm	Вилучити файли
vi	Викликати текстовий редактор vi
ln	Утворити посилання
ls	Вивести детальну інформацію про файли та каталоги
zip, unzip	Архівувати (розархівувати) файли
file	Визначити тип файлу
lp	Вивести вміст файлу на принтер

Розглянемо дію команд:

cp file1 file2 — для файлу file1 робить копію з назвою file2;

cp .<шлях призначення> — копіює усі файли із поточного залогу за шляхом призначення;

cp .<повний шлях звідки/*.*> — копіює усі файли із зазначеного каталогу в поточний.

Для копіювання варто використовувати маску файлів. Наприклад , щоб скопіювати усі файли поточного каталогу, назва яких починається з літери „a", у підкаталог Stud, треба застосувати команду cp a*.* Stud.

Кожний файл в ОС Linux має свій ідентифікаційний номер, наприклад, 1230, який називають індексним дескриптором, під яким файл реєструється у

системі. Отримати на екрані індексні дескриптори можна командою `ls -i`. В ОС Linux виділяють два види посилань: жорсткі та символічні. Жорсткі посилання можна створити лише в текстовому режимі за допомогою команди `ln <назва файлу> <назва посилання>`. Вони мають той самий індексний дескриптор, що й файл. Отримати список усіх жорстких посилань можна командою `ln -i`. Система веде облік кількості жорстких посилань на файл і відображає відповідне число в таблиці детальних властивостей файлу (див. далі).

Символічне посилання – це окремий короткий файл, який містить адресу того файлу, на який воно вказує. У текстовому режимі символічне посилання можна створити за допомогою команди `ln -s <назва файлу> <назва посиланнях>`. Відмінність між посиланнями така: якщо файл-оригінал перемістити на інше місце у файловій системі, то його символічні посилання не функціонуватимуть без переналаштування, а жорсткі функціонуватимуть далі. Якщо вилучити файл-ори-гінал із системи, то символічні посилання не функціонуватимуть зовсім, а жорсткі будуть. Лічильник обліку жорстких посилань зменшиться на одиницю. Власна назва файлу трактується як його (перше) жорстке посилання. Отже, файл-оригінал буде вилучено із системи після вилучення його останнього жорсткого посилання.

Архіватори і редактор текстів

В ОС Linux є декілька стандартних архіваторів: `zip` (підтримується також в ОС Dos), `tar`, `cpio` тощо. Тут розглядатимемо архіватор `zip`. Заархівувати групу файлів можна так:

```
zip  <повна назва архівного файлу><файл 1> <файл 2>...<файл n>.
```

Наприклад, заархівувати усі файли з поточного каталогу і розмістити архів з назвою `myarhiv.zip` у підкаталозі `/Stud` можна за допомогою команди:

```
zip /Stud/myarhiv.
```

Тут тип архівного файлу зазначати не обов'язково. Розархівувати файл можна так:

```
unzip <назва архівного файлу>.
```

Щоб розархівувати файл і розташувати результати у заданому каталозі, потрібно виконати таку команду:

```
unzip <назва архівного файлу> -d <назва каталогу>.
```

Наприклад, командою

```
unzip /Stud/myarhiv.zip
```

файл `myarhiv.zip` буде розархівований у поточний каталог, а командою

```
unzip /Stud/myarhiv.zip -d /Stud/Name
```

файли з архіву `myarhiv.zip` будуть розархівовані у підкаталог `Name` каталогу `Stud`.

Для створення та редагування файлів призначені текстові редактори. Одним з найпоширеніших редакторів, як вже знаєте, призначених для роботи у текстовому режимі, є редактор `vi`. Щоб викликати редактор, треба виконати команду **vi** або `vi <назва файлу>`. Якщо такий файл уже існує, то він буде відкритий для редагування, інакше буде створено новий файл. Для переміщення у файлі можна використовувати клавіші із зображенням стрілок, а також клавіші PageUp та PageDown. Щоб перейти з режиму введення інформації у командний режим, потрібно натиснути на клавішу Esc. Щоб зберегти інформацію, треба увести команду `:w` або `:w <назва файлу>`, щоб вийти із редактора — команду `:q`, щоб вийти без збереження змін — `:q!`

Створити файл за допомогою клавіатури можна також командою `cat > <назва файлу>`, наприклад, `cat > text.txt`. Тут символ „>" означає операцію перенаправлення введення з клавіатури у відповідний файл. Закінчують введення даних за допомогою комбінації клавіш Ctrl+D.

Основні команди для роботи з каталогами

У текстовому режимі над каталогами можна виконувати ті ж самі дії, що й у графічному. Для цього призначені команди, наведені у таблиці 2.11.

Таблиця 2.11 – Команди для роботи з каталогами

Назва команди	Дія команди
`Is або dir`	Виводить на екран зміст поточного каталогу
`cd`	Перехід у каталог

mkdir	Створює новий каталог
pwd	Відображає повну назву каталогу

Продовження таблиці 2.11

rrndir	Вилучає порожній каталог
chmod	Змінює права доступу до файлу
chown	Змінює ім'я власника файлу
chsrp	Змінює назву групи-власниці файлу

Зауважимо, що команди chown, chsrp доступні лише користувачеві root. Розглянемо приклади застосування команди cd:

cd < назва каталогу> — відбудеться перехід у каталог із зазначеною назвою; cd .. — повертає користувача у надкаталог;

cd .. /.. — перехід на два рівні (надкаталоги) вгору; cd / — активізує кореневий каталог.

Права доступу до файлів і каталогів та керування ними

Кожний файл чи каталог має власний набір атрибутів щодо прав доступу до нього. Є три основні типи (рівні) Власників, які можуть мати різні права доступу:

1) власник файлу чи каталогу,

2) група, до якої належить власник,

3) усі інші користувачі системи.

Є три головні способи (дії) доступу до файлу та до каталогу: читання (атрибут r), записування (w), виконання (x). Права доступу до файлу чи каталогу треба задавати у зазначеному порядку. Дозвіл на читання файлу означає, що його вміст можна переглядати, а дозвіл на записування — що його вміст можна переглядати та редагувати (змінювати, записувати зміни). Дозвіл на виконання означає, що файл можна запускати на виконання; це стосується програм і сценаріїв. Для каталогу дія читання означає; що його вміст можна переглядати, записування - у ньому можна створювати та

вилучати підкаталоги та файли, виконання — стають доступними усі атрибути прав доступу для підкаталогів чи файлів, які у ньому розміщені.

Щоб з'ясувати усі атрибути файлів та каталогів, треба застосувати команду `ls -l <назва каталогу>`. Нехай у поточному каталозі є підкаталог `Grupa` та два файли `grl.txt` та `gr2.txt`, власником яких є `Petro`, який належить до групи користувачів `Class`. Після виконання команди `ls -l` отримаємо властивості файлу.

Наступні три групи з трьох символів кожна (наприклад, `rw-rw-r--` для файлу `grl.txt`) означають права доступу до файлу, які належать відповідно власникові (`rw`-означає, що файл доступний для читання, записування та недоступний для виконання), групі (`rw-`) та усім іншим користувачам (`r-`). Далі зазначено кількість імен файлу разом з жорсткими посиланнями, імена власника та групи, обсяг, дата і час створення та власна назва файлу.

Визначити чи змінити права доступу для певного файлу чи каталогу можна за допомогою команди `chmod`. Її загальний вигляд такий:

`chmode <рівень>+` або

`-<спосіб доступу> <назва >файлу чи каталогу>`

Можливі такі рівні, які зазначені в таблиці 2.12 і такі способи доступу, які наведені в таблиці 2.13:

Таблиця 2.12 – Рівні доступу

u	власник
g	група
o	інші
a	всі

Таблиця 2.13 – Способи доступу

r	читання
w	записування

x	виконання

Символи "+" чи "-" відповідно вмикають або вимикають спосіб доступу. Наприклад, розглянемо файл `gr2.txt`, описаний вище. Він має такі атрибути прав доступу: `rw-rw-r --.` Змінити (скасувати) право на читання для власника можна командою `chmod u-r gr2.txt`. Отримаємо такий набір прав: `-w-rw-r--.`

Заборонити групі та усім іншим переглядати файл можна командою `chmod g-r,o-r gr2.txt`. Після виконання цієї команди будуть визначені такі права доступу: `-w— w----.`

Дозволити усім користувачам системи читати файл можна командою `chmod a+r gr2.txt`. Після її виконання отримаємо такі права: `rw-rw-r--.`

2.2.2 Порядок виконання другої частини лабораторної роботи

`Хід роботи 1`

1.1. Після завантаження ОС перейдіть у текстовий режим. Увімкніть першу віртуальну консоль `Ctrl+Alt+F1`.

1.2. Зареєструйтесь у системі. Уведіть свій логін та пароль, якщо потрібно.

1.3. Перегляньте, хто на цей момент працює у мережі. Зазначимо, що імена користувачів мережі можна отримати командою `who`. Скільки людей на цей момент зареєстровано у мережі? Запишіть декілька імен у звіт.

1.4. Отримайте інформацію про себе. Уведіть команду `whoami`.

1.5. Перевірте поточну дату. Виконайте команду `date`. Уточніть дату, якщо потрібно.

1.6. Отримайте інформацію про розподіл дискового простору (команда `df`). Скільки розділів має ваш комп'ютер? Занотуйте відповідь у звіті.

1.7. Проаналізуйте розподіл пам'яті на вашому комп'ютері (`free`). Занотуйте у звіт кількість загальної, використаної та вільної пам'яті на вашому комп'ютері.

1.8. Перейдіть у режим калькулятора (`be`).

1.9. Обчисліть значення таких виразів: а) 2956 + 8754; б) 345 * 143; в) 193 : 54. Для виконання обчислень почергово вводьте вирази та натискайте на клавішу Enter. Для обчислення суми використовуйте знак "+", різниці - "-", добутку - "*", ділення -"/". Щоб задати точність обчислень, тобто явно задати, скільки цифр треба відображати після десяткової коми , використовують команду `scale = <кількість цифр>`. Обчисліть значення третього виразу з точністю дві цифри після коми та п'ять цифр після коми. Занотуйте значення виразів у звіт. Щоб вийти з режиму калькулятора, натисніть на Ctrl + D.

1.10. Отримайте довідку про роботу калькулятора. Уведіть команду `man be`.

1.11. Виведіть календар за поточний місяць (`cal`).

1.12. Виведіть календар за поточний рік. Уведіть команду `cal <номер року>`.

1.13. Дізнайтесь, якого дня випало 1 квітня 1950 року. Уведіть `cal 4 1950`.

1.14. Дізнайтесь, якого дня тижня ви народилися. Формат використаної команди запишіть у звіт.

1.15. Очистіть екран (`clear`).

1.16. Запустіть менеджер файлів MC (`mc`) і вийдіть з нього. Для виходу з менеджера натисніть F10, далі виберіть Yes => Enter.

1.17. Визначте шлях до вашого поточного каталогу (`pwd`). Під час реєстрації система автоматично активізує каталог з вашою домівкою. Занотуйте шлях до каталогу домівки у звіт.

1.18. Перегляньте вміст поточного каталогу (`dir`).

1.19. Вилучіть усі файли з поточного каталогу. Виконайте команду `rm`. Що у вас не вийшло і чому?

1.20. Створіть у каталозі домівки каталог Kat1. Виконайте команду mkdir Kat1. Активізуйте (увійдіть у) Kat1. Виконайте команду cd Kat1.

1.21. У Kat1 створіть каталог Kat2.

1.22. У Kat2 створіть текстовий файл text.txt. Активізуйте Kat2 та викличте текстовий редактор командою vi text.txt. Уведіть текст — назву вашого навчального закладу. Для збереження тексту та виходу з текстового редактора почергово натисніть Esc => :w => :q.

1.23. Перегляньте вміст створеного файлу. Виконайте одну з команд less text.txt або more text.txt.

1.24. Вилучіть файл text.txt. Уведіть команду rm text.txt.

1.25. Вилучіть Kat2. Для вилучення каталогу спочатку з нього треба вийти (піднятись на рівень вище). Для цього виконайте команду cd .. Enter. Далі задайте команду rmdir Kat2 і каталог буде вилучено.

1.26. Вилучіть Kat1 з домашнього каталогу.

1.27. Виконайте контрольне завдання.

1.28. Вийдіть із системи (logout).

Хід роботи 2

1.1. Увійдіть в ОС Linux у текстовому режимі.

1.2. Утворіть дерево каталогів, описане в контрольному завданні попередньої практичної роботи. Якщо воно є, то перейдіть до виконання п. 3.

1.3. Отримайте інформацію про повний шлях до каталогу Country. Для цього відкрийте цей каталог командою cd Country та виконайте команду pwd. Занотуйте шлях у звіт.

1.4. Скопіюйте усі файли з каталогу Capital у каталог Country. Увійдіть у каталог Capital та виконайте команду cp *.* <шлях до каталогу Country>. У каталозі Capital перейменуйте файл

Nasel.txt на Nasel2.txt. Поверніться у каталог Country (cd ..) та виконайте команду mv Nasel.txt Nasel2.txt.

1.5. Ознайомтеся із вмістом каталогу Country. Виконайте команду ls. Які файли є у цьому каталозі? Занотуйте їхні назви у звіт.

1.6. Перегляньте вміст файлу Number.txt трьома різними способами, використовуючи: а) команду more; б) команду less; в) команду cat. Відповідні команди занотуйте у звіт.

1.7. Утворіть жорстке посилання на файл adress.txt, який є в підкаталозі MyKat каталогу City. У каталозі City виконайте команду ln adress.txt padress.

1.8. Утворіть тепер символічне посилання на файл adress.txt. Виконайте команду ln -s adress.txt sadress.

1.9. Перегляньте індексні дескриптори файлів та каталогів поточного каталогу City. Виконайте команду ls -i. Переконайтесь, що файл adress.txt та посилання padress мають один і той самий індексний дескриптор.

1.10. Утворіть ще одне жорстке посилання (padress2) для файлу adress.txt, яке розмістіть у каталозі Country.

1.11. Відобразіть детальну інформацію про вміст поточного каталогу City. Виконайте команду ls -l. Скільки жорстких посилань має файл adress.txt?

1.12. Передивіться вміст файлу, використовуючи його символічне посилання. Виконайте команду vi sadress. Допишіть у файл поштовий індекс і телефонний код вашого населеного пункту => Esc => :w=> :q.

1.13. Вилучіть файл adress.txt. Виконайте команду rm adress.txt.

1.14. Ще раз відобразіть детальну інформацію про вміст поточного каталогу City. Скільки тепер жорстких посилань має файл adress.txt?

1.15. Передивіться файл sadress. Яку інформацію ви отримали? Занотуйте її у звіт.

1.16. Вилучіть файл sadress.

1.17. Заархівуйте усі файли з каталогу Capital. Архівний файл arhiv.zip розташуйте у каталозі Country. Для цього увійдіть у каталог Capital та виконайте команду zip <шлях до архівного файлу>/arhiv *.*.

1.18. Розархівуйте файл arhiv.zip. Увійдіть у каталог Country та виконайте команду unzip arhiv.

1.19. Розархівуйте файл arhiv.zip у каталог MainCity. Увійдіть у каталог Country та виконайте команду unzip arhiv -d <шлях до каталогу MainCity>.

1.20. Заархівуйте усі файли з MyKat. Архів (arhiv2.zip) розташуйте у каталозі Capital.

1.21. Розархівуйте arhiv2.zip. Увійдіть у каталог Capital та виконайте команду unzip arhiv2.

1.22. Розархівуйте arhiv2.zip у каталог MainCity.

1.23. Перемістіть arhiv2.zip у каталог MyKat. Спочатку скопіюйте цей файл у каталог MyKat, а потім вилучіть його з каталогу Country.

1.24. Вилучіть усі файли з каталогу Country.

1.25. Виведіть на екран детальну інформацію про вміст каталогу Capital. Зайдіть у цей каталог та уведіть команду ls -l.

1.26. Об'єднайте файли Number.txt та Nasel.txt у файл info.txt. Для цього виконайте команду cat Number.txt Nasel.txt > info.txt.

1.27. Передивіться вміст файлу info.txt.

1.28. Скасуйте право доступу для читання файлу arhiv.zip для інших користувачів. Для цього виконайте команду chmod o-r arhiv.zip.

1.29. Скасуйте право доступу для записування для усіх користувачів.

1.30. Виконайте команду `chmod a-w arhiv.zip`.

1.31. Надайте право доступу для записування для власника та групи.

1.32. Виконайте команду `chmod u+w,g+w arhiv.zip`.

1.33. Скасуйте право доступу для записування для файлу `Number.txt` для групи. Запишіть відповідну команду в звіт.

1.34. Скасуйте право доступу для читання для усіх користувачів. Запишіть відповідну команду для файлу `Number.txt` у звіт.

1.35. Надайте право доступу для читання для власника та групи. Запишіть відповідну команду для файлу `Number.txt` у звіт.

1.36. Закінчіть роботу. Здайте звіт.

2.2.3 Завдання до другої частини лабораторної роботи

На парі відпрацювати лабораторну роботу (хід роботи 1 або хід роботи 2 в залежності від номеру бригади: для парного номеру – 2-ий хід роботи, для непарного – 1-ий хід роботи).

Контрольне завдання аналогічно обираємо за номером бригади.

Контрольне завдання(1)

У домашньому каталозі створіть дерево каталогів відповідно до наведеного нижче завдання. Створіть каталог про свою країну (`Country`), у ньому два підкаталоги про столицю (`Capital`) та своє місто (`City`). У каталозі `Capital` створіть два текстових файли: `Regions.txt` та `Nasel.txt`. У першому файлі вкажіть кількість обласних центрів країни, а в іншому — населення країни. У каталозі `City` створіть ще два підкаталоги: один з вашими особистими даними `MyKat`, інший - з інформацією про ваш обласний центр `MainCity`. У каталозі `MyKat` створіть два файли. У першому (`adress.txt`) запишіть свою адресу, в іншому (`school.txt`) - в якій школі ви навчались. У каталозі `MainCity` створіть файл (`info.txt`) з

довідковою інформацією про ваш обласний центр — назва, рік заснування, кількість населення тощо. Намалюйте дерево створеного вами каталогу у звіті.

Контрольне завдання (2)

За результатами роботи у домашньому каталозі має бути каталог `Country`, у якому є два підкаталоги `Capital` та `City`, зазначені у таблиці 2.14. У каталозі `Capital` має бути чотири файли: `Number.txt`, `Nasel.txt`, `info.txt` та `arhiv.zip`. Файли `Number.txt` і `arhiv.zip` мають мати влаштовані відповідно до ходу роботи права доступу. Каталог `City` повинен містити два підкаталоги: `MyKat` і `MainCity`. У `MyKat` мають бути файли `adress.txt`, `school.txt`, `adress.txt` і `arhiv.zip`, а у каталозі `MainCity` — файли `info.txt`, `adress.txt`, `school.txt`, `adress.txt`, `Number.txt`, `Nasel.txt`.

Таблиця 2.14 – Список країн для бригад:

№ бригади	Країна
1	Велика Британія
2	Італія
3	Франція
4	Німеччина
5	Іспанія
6	Україна
7	Австрія
8	Польща
9	Білорусь
10	Болгарія
11	Греція
12	Данія
13	Естонія

14	Ірландія
15	Ісландія
16	Латвія
17	Литва
18	Ліхтенштейн
19	Люксембург
20	Норвегія
21	Португалія
22	Росія
23	Румунія

Продовження таблиці 2.14

24	Сербія
25	Словаччина
26	Фінляндія
27	Чорногорія
28	Швейцарія
29	Швеція
30	Чехія

2.2.4Контрольні запитання до другої частини лабораторної роботи

1. Що таке текстовий режим роботи в ОС Linux?
2. Як увійти в ОС Linux у текстовому режимі?
3. Як з графічної оболонки перейти на консоль з номером n?
4. Як повернутися у графічну оболонку?
5. Як отримати довідкову інформацію про роботу деякої команди?
6. Наведіть загальний вигляд команди.
7. Які сервісні команди Linux вам відомі?
8. Якою командою можна переглянути вміст файлу?
9. Яка команда призначена для копіювання об'єктів?

10. Яка команда призначена для вилучення файлів?

11. Як викликати текстовий редактор під час роботи у консолі?

12. За допомогою якої команди можна отримати детальну інформацію про файли та каталоги?

13. Які команди використовують для архівування та розархіву-вання даних?

14. Як створити жорстке посилання на файл?

15. Яка відмінність між жорсткими та символічними посиланнями?

16. Як створити символічне посилання на файл?

17. Яким чином можна зархівувати декілька файлів?

18. Як вивести на екран вміст поточного каталогу?

19. Як створити новий каталог?

20. Як вилучити каталог?

21. За допомогою якої команди можна змінити права доступу до файлу чи каталогу?

22. Які способи доступу до файлів чи каталогів ви знаєте?

23. Як визначити право доступу на читання для членів групи?

24. Які права надасть команда `chmod g-r file.txt`?

25. Які спеціальні права доступу вам відомі?

26. Назвіть команди для роботи з файлами?

27. Які ви знаєте сервісні команди та програми?

28. Назвіть три варіанти запуску сеансу роботи ОС Linux у текстовому режимі?

29. Які ви знаєте команди для роботи з каталогами?

30. Що виконує команда `cat`?

ЛАБОРАТОРНА РОБОТА №3. УПРАВЛІННЯ ФАЙЛОВОЮ СИСТЕМОЮ

3.1 Управління файловою системою

3.1.1 Теоретичні відомості

Робота з файлами та пристроями збереження даних в ОС Linux відрізняється від звичайної ОС Windows. Є файли та ієрархічна структура каталогів, але в системі Linux нема букв, що позначають диски. В Linux є тільки одна файлова структура. Вона починається від кореня (/),і всі локальні файлові системи, всі локальні пристрої и всі віддаленні системи представленні в цій структурі підкаталогів.

```
/
|-- bin
|-- boot
|-- dev
|-- etc
|-- mnt
|-- root
|-- sbin
|-- tmp
|-- usr

|   |-- X11R6
|   |   |-- bin
|   |   |-- include
|   |   |-- lib
|   |   |-- man
```

| | `-- share

| |-- bin

Коли Linux завантажується вперше, він будує файлову структуру на основі інформації з `/etc/fstab` файлу. Там, де Windows присвоює розділам жорсткого диска і інших пристроїв зберігання даних літери, Linux визначає їм каталоги в кореневій файловій структурі. Структуру ієрархії можна повністю налаштовувати і міняти на «льоту».

Слово, що означає додавання пристрою до файлової системи - монтування (`mounting`). Linux автоматично монтує кореневу (/) файлову систему. Окремо може бути присутнім файлова система / boot, в якій розташовані завантажувальні файли ядра. Linux також монтує деякі особливі файлові системи. Область «свопінгу» не показується як частина файлової системи, але управляється ядром.

Інші файлові системи, такі як змінний носій або видалені файлові системи необхідно монтувати вручну. Монтуючи файлову систему, ви повинні знати правильний шлях, щоб послатися на нього з Linux і мати порожній каталог, щоб використовувати його як точку монтування (mount point). Для змінних носіїв Linux, як правило, створює точки монтування під час інсталяції. У Red Hat Linux пристрій читання компакт-дисків передбачається монтувати до каталогу `/media/cdrecorder`. Це означає, що коли ви вставляєте CD в пристрій CDROM, ви вводите команду: `mount /dev/cdrom /media/cdrecorder` (якщо не прописаний в `/etc/fstab`).

CD додається у файлову систему і пристрій читання CD блокується так, що випадково воно не відкриється. Для доступу до вмісту компакт-диска просто використовуйте каталог `/media/cdrecorder`. Коли ви закінчите працювати з CD, можна буде видалити його з файлової системи за допомогою команди:

```
umount /media / cdrecorder
```

Каталог буде очищений, а пристрій читання CD розблоковано. Точно також слід чинити і з іншими змінними носіями інформації, такими як дискета: `/media/floppy`.

Якщо виконати `mount` без аргументів, то будуть показані файлові системи, які підмонтовані зараз.

Стандартні імена пристроїв в Linux

Linux не бачить різниці між файлами та пристроями, всі файли пристроїв знаходяться у каталозі `/dev`.

Розглянемо стандартні IDE-диски. Всього в комп'ютері з інтерфейсом IDE може бути встановлено 4 пристрої - Primary Master, Primary Slave, Secondary Master, Secondary Slave. У Linux це відповідає іменам `/dev/hda, /dev/hdb, /dev/hdc, /dev/hdd`.

Розділи на дисках нумеруються: `/dev/hda1` – перший розділ на Primary Master. Всього може бути 4 первинних розділів. Наприклад, у вас є 3 розділи - завантажувальний для Windows, Linux Native, Linux Swap.

У Linux це буде представлятися так:

```
/dev/hda1 - Windows
/dev/hda2 - Linux Native
/dev/hda3 - Linux Swap
```

В Windows - тільки C:

Монтування Windows файлової системи

Коли ви встановлюєте Linux, ви створюєте кореневу файлову систему. Під кореневої файлової системою будемо розуміти розділ, на який ви встановили Linux.

Щоб працювати з інформацією, розташованої на розділах Windows, вам потрібно їх примонтувати, тобто підключити до файлової системи. Точка монтування – це всього лише каталог, через який буде відбуватися звернення до інших файлових систем. Зазвичай інші розділи підключаються до підкаталогу каталогу `/mnt (/media)`. Наприклад, якщо Windows-розділ

примонтувати до підкаталогу win каталогу /mnt, то ви можете продивитись його вміст командою ls /mnt/win.

Усунення неполадок

Якщо в змонтованому Windows-розділі замість імен файлів у кирилиці суцільні «???????» необхідно в опціях монтування вказати використовувану кодування для файлової системи vfat:

- codepage=866,iocharset=koi8-r
- codepage=866,iocharset=cp1251
- iocharset=utf8

```
mount -t vfat iocharset=utf8 /dev/hda1 /mnt/win
```

Для NTFS замість iocharset =% імя_кодування% слід використовувати nls =% імя_кодування%. Файлова система NTFS, яка використовується в Windows NT лінійки (включаючи Windows 2000, Windows XP і більш старші версії), є повністю закритою розробкою. З цієї причини створення NTFS-драйвера пов'язане з цілою низкою проблем. Однак у більшості популярних дистрибутивів є можливість використовувати NTFS-розділи без яких-небудь складнощів. Монтуються вони так само, як і у випадку з FAT32, тільки замість vfat потрібно вказати ntfs.

Для користувачів Red Hat Linux творці цього дистрибутива не включають в його комплект драйвер NTFS, тому буде потрібно самим завантажити його і встановити. Для Debian можна використати пакет ntfs-3g.

Автоматизація процесу монтування

Зв'язок між пристроєм і його точкою монтування налаштовується у файлі /etc/fstab. Його можна редагувати вручну або залишити програмі адміністрування. Приклад /etc/fstab/ забражений на рисунку 3.1.

/dev/hda5	/	ext3	defaults	1 1
/dev/hda2	/boot	ext3	exec,dev,duid,rw	1 2
/dev/hda6	swap	swap	defaults	0 0
/dev/scd0	/mnt/cdrom	auto	ro,noauto,exec	0 0
none	/dev/pts	devpts	id=5,mode=620	0 0
none	/proc	proc	defaults	0 0
none	/dev/shm	tmpfs	defaults	0 0

Рисунок 3.1 – Приклад `/etc/fstab/`

Кожен рядок представляє вмонтовувану файлову систему. Перша колонка визначає пристрій, який повинен бути змонтований. У другій колонці знаходиться точка монтування – місце розташування даного пристрою у файловій системі. Третя колонка визначає тип файлової системи. У четвертій містяться опції, що показують як ця файлова система буде оброблятися. В останній колонці написані прапори пов'язані з файловою системою. Перша цифра, 0 або 1, показує, чи повинна система копіюватися за допомогою команди `dump` (це потрібно для системних резервних копій). Друга цифра може бути 0, 1 або 2, вона показує порядок, у якому файлова система повинна бути перевірена при завантаженні. 0 - не повинна перевірятися зовсім. 1 - повинна перевірятися першою і використовуватися як коренева (/). Для всіх інших систем ставиться 2.

У файлі `fstab`, наведеному вище, коренева файлова система розташована на першому IDE жорсткому диску, у п'ятому розділі, перший логічний диск в розширеному розділі. Файлова система `/boot`, де знаходяться файли запуску ядра, розташована на першому IDE жорсткому диску, у другому первинному розділі. Простір свопінгу розташований на першому IDE жорсткому диску, у шостому розділі, другому логічному диску в розширеному розділі. Інші перераховані файлові системи показують свій пристрій, як "none". Ми висвітлимо це коротко пізніше. А зараз давайте сконцентруємося на фізичних дисках.

Опції в четвертій колонці будуть змінюватися в залежності від типу файлової системи. У наведеному вище прикладі `/` і `/boot` підмонтувати з

опціями "default". Це означає, що вони монтуються автоматично, доступні для читання і запису з асинхронним I/O (введення/виведення). Тільки root може монтувати і відмонтувати пристрої, але користувачі можуть виконувати бінарники. Для /mnt/cdrom, однак, опції інші. Він не монтується автоматично і буде відкритий тільки для читання. Користувачі зможуть виконувати скрипти і програми в цій файловій системі. Нижче зазначена таблиця 3.1 із прапорцями, пов`язаними з файловою системою та їх значенням.

Таблиця 3.1 – Значення прапорів, пов`язаних з файловою системою

Опція	Значення
auto	Файлова система може «монтуватися» командою mount з опцією -a
defaults	Використати набір опцій, що задається умовно: rw, suid, dev, exec, auto, nouser, async
noauto	Файлова система може «монтуватися» тільки явно. Опція -a не приведена для монтування файлової системи.
exec	Дозволяє виконання «двійкових» файлів
remount	Дозволяє «перемонтувати» вже змонтовану файлову систему. Зазвичай використовується для зміни опцій монтування файлової системи, особливо для того, щоб розширити права доступу (замість прав тільки на читання встановить права на читання/ запис)
ro	Монтує файлову систему тільки для читання
rw	Монтує файлову систему для читання/запису
user	Дозволяє непривілейованому користувачу монтувати файлову систему. Для таких користувачів монтування завжди виконується з опціями noexec, nosuid, nodev

Продовження таблиці 2.1

nodev	Файли байт-орієнтованих та блок-орієнтованих пристроїв в файловій системі не інтерпретуються як спеціальні файли.
nouser	Забороняється непривілейованому користувачу монтувати файлову систему.

Розділи в Linux працюють в загальному випадку так само як і в Windows. Консольна команда `fdisk` використовується для створення та управління розділами. При виконанні цієї команди потрібно вказати їй пристрій. Щоб подивитися доступні пристрої, скористайтеся командою `fdisk-l`.

Запустіть `fdisk` з пристроєм і ви отримаєте коротке запрошення. Введення "`m`" дасть вам меню команд. Поточна таблиця розділів відобразиться після натискання "`p`". Ви можете створювати, видаляти і змінювати тип існуючих розділів. "`l`" покаже вам повний список доступних типів розділів. Свої зміни в таблицю розділів вносите за допомогою "`w`", закривайте програму або виходьте без збереження використовуючи "`q`". Деякі зміни вступлять в силу негайно. Деякі можуть зажадати перезавантаження системи.

Використання файлових ресурсів відаленнів хостів

Багато файлових ресурсів підключаються через мережеві системи. Найбільш поширеними є NFS (Network File System - мережева файлова система) і SMB ресурси. Для доступу до цих файлових систем необхідно знати ім'я хоста і ім'я «розшареного» ресурсу. Для визначення того, які файлові системи експортуються віддаленою системою, використовуються наступні команди:

- для NFS:

```
showmount –e ім’я_віддаленої_системи
```

- для SMB:

```
smbclient –L ім’я_віддаленої_системи –N
```

Коли ім'я хоста і ім'я «розшареного» ресурсу відомі, то використовують наступні команди для підключення мережевих файлових систем до дерева локальної файлової системи.

- для NFS:

```
mount ім’я_віддаленого_хоста:/shared/dir /mnt/ім’я_ntf_ресурсу
```

- для SMB:

```
mount //ім’я_віддаленого_хоста/share /mnt/ім’я_samba_ресурсу
```

Типи файлових систем

Linux може обслуговувати будь-який тип файлової системи, про який знає ядро. Неабияка їх кількість компілюється за замовчуванням, а нові можна додавати. Найбільш цікаві з них представлені в таблиці 3.2.

Таблиця 3.2 – Типи файлових систем

ext2	Стандартна файлова система Linux
ext3	Стандартна файлова система Linux з доданим протоколом
vfat	Файлова система Microsoft Fat32
jfs	«Журнальована» файлова система IBM
reiserfs	Ще одна популярна «журнальована» файлова система

«Журнальовані» файлові системи допомагають уберегти дані при непередбачених виключеннях системи. Якщо та вимикається невідмонтованою, то можуть залишитися незакінчені процеси і файли в проміжному стані. При використанні звичайної файлової системи буде потрібна повна перевірка, що може зайняти немало часу для великих обсягів. Журнальована файлова система зберігає коректні записи кожного введення інформації на диск за період часу, наприклад п'ять секунд. Коли та не відмонтована акуратно, файлова система просто відкочується назад до останнього нормального стану, яке вона запам'ятала. І те, що могло б відновлюватися хвилин двадцять тепер піднімається за кілька секунд.

Форматування розділів

Створені розділи форматуються підходящої версією команди `mkfs`. У файлових систем є свої власні версії `mkfs`, наприклад, `mkfs.ext2` або `mkfs.ext3`. Ці допоміжні скрипти дозволять створити файлову систему, просто вказавши на розділ. Ось декілька прикладів:

- `mkfs.ext2 /dev/hda3`
- `mkfs.ext3 /dev/sdb1`

Існують різні просунуті параметри, щоб впливатимуть на те, як буде форматуватися розділ, але в загальному випадку цілком підходять значення за замовчуванням. Відформатований розділ може бути змонтований в корінь (/). Файлову систему слід відмонтувати перед форматуванням.

3.1.2 Порядок виконання першої частини лабораторної роботи

1. Ознайомтеся з вмістом файлу `/etc/fstab`.
2. Змонтуйте різні змінні носії: дискету, CD-ROM, флеш накопичувач.
3. Вивчіть інші програми файлової системи.

Є кілька програмних засобів для перегляду стану дисків і файлових систем. `df` означає "disk free". Ця команда повідомляє, скільки місця на диску використовується і є доступним в підмонтованій файловій системі. Перевірка дискового простору:

df -l

Обмежує список до локальних файлових систем; за замовчуванням показуються ще й віддалені.

`du` означає "disk usage" (використання диска). Команда виводить, скільки дискового простору використовується певними файлами і для кожного підкаталогу (якщо аргумент - каталог).

Перевірка використання диска:

```
du -a
```

Показує лічильники для всіх файлів, не тільки каталогів.

```
du -h
```

При відображенні розміру файлу використовує дружній для людини вивід (в Mb, Kb), а не відображає його в байтах.

3.1.3Завдання до першої частини лабораторної роботи

Зміст завдяня згідно номеру варіанту знаходиться у таблиці 3.3.

Таблиця 3.3 – Завдання до лабораторної роботи

Варіант	Завдання
1	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте дискету. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключіть файлову систему до дерева локальної файлової системи (для NFS)
2	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте CD-ROM. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для SMB)
3	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте флеш накопичувач. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключити файлову систему до дерева локальної файлової системи (для NFS)
4	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключити файлову систему до дерева локальної файлової системи (для SMB)
5	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте дискету. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для

	NFS)
6	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте CD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключити файлову систему до дерева локальної файлової системи (для SMB)

Продовження таблиці 3.3

7	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте флеш накопичувач. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS)
8	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключити файлову систему до дерева локальної файлової системи (для SMB)
9	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте дискету. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS)
10	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте CD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключити файлову систему до дерева локальної файлової системи (для SMB)
11	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте флеш накопичувач. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової

	системи (для NFS)
12	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Підключити файлову систему до дерева локальної файлової системи (для SMB)

Продовження таблиці 3.3

13	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте дискету. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS)
14	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте CD-ROM. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для SMB)
15	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте флеш накопичувач. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS)
16	Ознайомтесь зі змістом файла `/etc/fstab`. Вмонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Вмонтуйте файлову систему тільки для читання. Підключити файлову систему до дерева локальної файлової системи (для SMB)
17	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте дискету. Створіть файлову систему (*) і відформатуйте. Підключити

	файлову систему до дерева локальної файлової системи (для NFS)
18	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте CD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work` (створив файл з назвою - свого призвіща). Підключити файлову систему до дерева локальної файлової системи (для SMB)

Продовження таблиці 3.3

19	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте флеш накопичувач. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS)
20	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Вмонтуйте файлову систему тільки для читання. Підключити файлову систему до дерева локальної файлової системи (для SMB).
21	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте дискету. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS).
22	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте CD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work` (створити файл з назвою - своє призвіще). Підключити файлову систему до дерева локальної файлової системи (для SMB).
23	Ознайомтесь зі змістом файла `/etc/fstab`. Змонтуйте флеш

	накопичувач. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS).
24	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Змонтуйте файлову систему тільки для читання. Підключити файлову систему до дерева локальної файлової системи (для SMB).

Продовження таблиці 3.3

25	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте дискету. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS).
26	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте CD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work` (створити файл з назвою - своє призвіще). Підключити файлову систему до дерева локальної файлової системи (для SMB).
27	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте флеш накопичувач. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS).
28	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте DVD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work`. Змонтуйте файлову систему тільки для читання. Підключити файлову систему до дерева локальної файлової системи (для SMB).
29	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте

	дискету. Створіть файлову систему (*) і відформатуйте. Підключити файлову систему до дерева локальної файлової системи (для NFS).
30	Ознайомтесь зі змістом файла `/etc/fstab`. Демонтуйте CD-ROM. Створіть файлову систему (*) і помістіть її в розділ `work` (створити файл з назвою - своє призвіще). Підключити файлову систему до дерева локальної файлової системи (для SMB).

(*) `ext2` і `vfat` – парної бригади.

`ext3` і `jfs` – непарної бригади.

Примітка: якщо змінні носії інформації не присутні, потрібно спочатку змонтувати, а потім вже демонтувати їх.

3.1.4Контрольні запитання до першої частини лабораторної роботи

1. Чим робота з файлами та пристроями збереження даних в ОС Linux відрізняється від звичайної ОС Windows.?

2. Що відбувається, коли Linux завантажується вперше?

3. Що таке монтування?

4. Які стандартні імена пристроїв в Linux?

5. У якому файлі налаштовується зв'язок між пристроєм і його точкою монтування? За що відповідає кожна з колонок?

3.2 Робота з операційною системою Linux

3.2.1Теоретичні відомості

Розглянемо найбільш вживані команди для роботи з операційною системою.

`top` – команда видачі даних про активність процесів. Програма `top` динамічно видає в режимі реального часу інформації про працюючу систему, тобто про фактичну активність процесів. За замовчуванням вона видає завдання, що найбільш завантажують процесор сервера, і оновлює список

кожні п'ять секунд. При роботі команди `top` можна скористатися наступними корисними гарячими клавішами (таблиця 3.4).

Таблиці 3.4 – Гарячі клавіши для роботи з командою `top`

Гаряча клавіша	Використання
`t`	Включення і виключення видачі на екран сумарних даних.
`m`	Включення і виключення видачі на екран інформації про використання пам'яті.

Продовження таблиці 3.4

`A`	Сортування рядків за максимальним споживанням різних системних ресурсів. Корисна для швидкої ідентифікації завдань, для яких в системі бракує ресурсів.
`f`	Вхід в меню інтерактивної конфігурації даних, що видаються на екран командою `top`. Корисна для налаштування команди `top` для виконання специфічного завдання.
`o`	Дозволяє вам інтерактивно задавати порядок рядків, що видаються командою `top`.
`r`	Зміна пріоритету процесів за допомогою команди `renice`.
`k`	Видалення процесу за допомогою команди `kill`.
`z`	Перемикання між кольоровим/монохромним варіантом видачі зображення.

`vmstat` – активність системи, інформація про систему і апаратні ресурси. Команда `vmstat` видає інформаційний звіт про активність

процесів, пам'яті, свопінгу, поблочного введення/виводу, переривань і процесора.

Приклад виведення даних команди наступний:

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu--
---- r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy
id wa st 0  0      0 2540988 522188 5130400    0    0     2    32    4    2
4  1 96  0  0 1  0      0 2540988 522188 5130400    0    0     0   720 1199
665  1  0 99  0  0 0  0      0 2540956 522188 5130400    0    0     0     0
1151 1569  4  1 95  0  0 0  0      0 2540956 522188 5130500    0    0     0
6 1117  439  1  0 99  0  0 0  0      0 2540940 522188 5130512    0    0     0
536 1189  932  1  0 98  0  0 0  0      0 2538444 522188 5130588    0    0
0     0 1187 1417  4  1 96  0  0 0  0      0 2490060 522188 5130640    0    0
0    18 1253 1123  5  1 94  0  0
```

Для видачі статистики використання пам'яті можна виконати наступну команду:

`vmstat - m`.

Для отримання даних про активність/неактивність сторінок пам'яті –

`vmstat - aw`.

Команда `w` видає інформацію про те, які ористувачі зараз знаходяться в системі і які процеси запущені від їх імені.

```
# w username
# w vivek
```

Приклад виведення даних команди:

```
17:58:47 up 5 days, 20:28,  2 users,  load average: 0.36, 0.26, 0.24
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    10.1.3.145       14:55    5.00s  0.04s  0.02s vim
/etc/resolv.conf
root     pts/1    10.1.3.145       17:43    0.00s  0.03s  0.00s w
```

Команда `uptime` повідомляє, як довго працює система.

Команду `uptime` можна використовувати для того, щоб визначити, як довго працює сервер. Видаються: поточний час, скільки часу працює система, скільки у нинішній момент зареєстровано користувачів і яке середнє навантаження на систему в останні 1, 5 і 15 хвилин.

Приклад виведення команди `uptime`:

```
18:02:41 up 41 days, 23:42,  1 user,  load average: 0.00, 0.00, 0.00
```

Навантаження може мінятися від системи до системи. Для системи з одним процесором прийнятним може вважатися значення від 1 до 3, для мультипроцесорних систем - від 6 до 10.

Команда `ps` видасть короткий список поточних процесів. Для того, щоб вибрати усі процеси, використовуєте параметр `-A` або `-e`:

```
# ps - A
```

Приклад виведення даних командою наступний:

```
 PID TTY          TIME CMD
    1 ?        00:00:02 init
    2 ?        00:00:02 migration/0
    3 ?        00:00:01 ksoftirqd/0
    4 ?        00:00:00 watchdog/0
    5 ?        00:00:00 migration/1
    6 ?        00:00:15 ksoftirqd/1
....
.....
 4881 ?        00:53:28 java
 4885 tty1     00:00:00 mingetty
 4886 tty2     00:00:00 mingetty
 4887 tty3     00:00:00 mingetty
 4888 tty4     00:00:00 mingetty
 4891 tty5     00:00:00 mingetty
 4892 tty6     00:00:00 mingetty
 4893 ttyS1    00:00:00 agetty
12853 ?        00:00:00 cifsoplockd
12854 ?        00:00:00 cifsdnotifyd
14231 ?        00:10:34 lighttpd
14232 ?        00:00:00 php - cgi
54981 pts/0    00:00:00 vim
55465 ?        00:00:00 php - cgi
55546 ?        00:00:00 bind9 - snmp - stat
55704 pts/1    00:00:00 ps
```

Команда `ps` подібна до команди `top`, але видає більше інформації. Розглянемо декілька прикладів використання команди.

1. Показати більше даних:

```
# ps - Al
```

2. Для того, щоб включити режим максимальної видачі даних (будуть показані аргументи командного рядка, передані в процес), використовуємо:

```
# ps - AlF
```

3. Показати потоки (LWP і NLWP):

```
# ps - AlFH
```

4. Показати потоки після процесів:

```
# ps - AlLm
```

5. Видати список усіх процесів на сервері:

```
# ps ax
# ps axu
```

6. Видати дерево процесів:

```
# ps - ejH
# ps axjf
# pstree
```

7. Видати інформацію про параметри безпеки:

```
# ps - eo euser, ruser, suser, fuser, f, comm, label
# ps axZ
# ps - eM
```

8. Показати кожен процес для користувача `argus`:

```
# ps - U argus - u argus u
```

9. Настроїти видачу даних у форматі, визначеному користувачем:

```
# ps - eo pid, tid, class, rtprio, ni, pri, psr, pcpu, stat,wchan :14,
comm
# ps axo stat, euid, ruid, tty, tpgid, sess, pgrp, ppid, pid, pcpu, comm
# ps - eopid, tt, user, fname, tmout, f, wchan
```

10.Показати ID процесів, запущених під Lighttpd:

```
# ps - C lighttpd - o pid=
```

чи

```
# pgrep lighttpd
```

чи

```
# pgrep - u vivek php - cgi
```

11. Показати ім'я для PID 55977:

```
# ps - p 55977 - o comm=
```

12. Видати 10 процесів, споживаючих найбільшу кількість пам'яті:

```
# ps - auxf | sort - nr - k 4 | head - 10
```

13. Видати 10 процесів, споживаючих найбільший ресурс процесора:

```
# ps - auxf | sort - nr - k 3 | head - 10
```

Команда `free` показує загальну кількість вільної і використовуваної системою фізичної пам'яті і пам'яті свопінгу, а також розміри буферів, використовувані ядром.

```
# free
```

Приклад виведення даних командою наступний:

```
             total       used       free     shared    buffers     cached
Mem:      12302896    9739664    2563232          0     523124    5154740
-/+ buffers/cache:    4061800    8241096
Swap:      1052248          0    1052248
```

Команда `iostat` видає статистику використання процесора, а також статистику введення/виведення для пристроїв, розділів і мережевих файлових систем (NFS).

```
# iostat
```

Приклад виведення даних командою наступний:

```
Linux 2.6.18-128.1.14.el5 (www 03.nixcraft.in)  06/26/2009
avg - cpu:  %user   %nice %system %iowait  %steal   %idle
           3.50    0.09    0.51    0.03    0.00   95.86
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda              22.04        31.88       512.03   16193351  260102868
sda1              0.00         0.00         0.00       2166        180
sda2             22.04        31.87       512.03   16189010  260102688
sda3              0.00         0.00         0.00       1615          0
```

Команда `sar` використовується для збору інформації про системну активність і видачі її у вигляді звіту або її збереження. Щоб побачити значення зчитувача мережевої активності, введіть:

```
# sar - n DEV | more
```

Для того, щоб побачити значення лічильників мережевої активності, починаючи з 24-го:

```
# sar - n DEV - f /var/log/sa/sa24 | more
```

За допомогою команди `sar` можна також видавати дані в режимі реального часу:

```
# sar 4 5
```

Приклад виведення даних даною командою наступний:

```
Linux 2.6.18-128.1.14.el5 (www 03.nixcraft.in)   06/26/2009
06:45:12 PM       CPU     %user     %nice   %system   %iowait    %steal
%idle
06:45:16 PM       all      2.00      0.00      0.22      0.00      0.00
97.78
06:45:20 PM       all      2.07      0.00      0.38      0.03      0.00
97.52
06:45:24 PM       all      0.94      0.00      0.28      0.00      0.00
98.78
06:45:28 PM       all      1.56      0.00      0.22      0.00      0.00
98.22
06:45:32 PM       all      3.53      0.00      0.25      0.03      0.00
96.19
Average:          all      2.02      0.00      0.27      0.01      0.00
97.70
```

Команда `mpstat` виводить дані про активність кожного наявного в системі процесора, процесор 0 буде першим. Команда `mpstat - P ALL` виводить дані про середнє використання ресурсів для кожного з процесорів:

```
# mpstat - P ALL
```

Приклад виведення даних командою наступний:

```
Linux 2.6.18-128.1.14.el5 (www 03.nixcraft.in)  06/26/2009
06:48:11 PM  CPU   %user   %nice    %sys %iowait    %irq   %soft  %steal
%idle    intr/s
06:48:11 PM  all    3.50    0.09    0.34    0.03    0.01    0.17    0.00
95.86   1218.04
06:48:11 PM    0    3.44    0.08    0.31    0.02    0.00    0.12    0.00
96.04   1000.31
06:48:11 PM    1    3.10    0.08    0.32    0.09    0.02    0.11    0.00
96.28     34.93
06:48:11 PM    2    4.16    0.11    0.36    0.02    0.00    0.11    0.00
95.25      0.00
06:48:11 PM    3    3.77    0.11    0.38    0.03    0.01    0.24    0.00
95.46     44.80
06:48:11 PM    4    2.96    0.07    0.29    0.04    0.02    0.10    0.00
96.52     25.91
06:48:11 PM    5    3.26    0.08    0.28    0.03    0.01    0.10    0.00
96.23     14.98
06:48:11 PM    6    4.00    0.10    0.34    0.01    0.00    0.13    0.00
95.42      3.75
```

```
06:48:11 PM    7    3.30    0.11    0.39    0.03    0.01    0.46    0.00
95.69     76.89
```

Команда `pmap` видає дані про розподіл пам'яті між процесами. Використання цієї команди дозволить знайти причину вузьких місць, пов'язаних з використанням пам'яті.

```
# pmap - d PID
```

Для того, щоб отримати інформацію про використання пам'яті процесом з `pid # 47394`, введіть:

```
# pmap - d 47394
```

Приклад виведення даних команди наступний:

```
47394:   /usr/bin/php - cgi
Address           Kbytes Mode  Offset           Device    Mapping
0000000000400000    2584 r - x-- 0000000000000000 008:00002 php - cgi
0000000000886000     140 rw--- 0000000000286000 008:00002 php - cgi
00000000008a9000      52 rw--- 00000000008a9000 000:00000   [ anon ]
0000000000aa8000      76 rw--- 00000000002a8000 008:00002 php - cgi
000000000f678000    1980 rw--- 000000000f678000 000:00000   [ anon ]
000000314a600000     112 r - x-- 0000000000000000 008:00002 ld - 2.5.so
000000314a81b000       4 r---- 000000000001b000 008:00002 ld - 2.5.so
000000314a81c000       4 rw--- 000000000001c000 008:00002 ld - 2.5.so
000000314aa00000    1328 r - x-- 0000000000000000 008:00002 libc - 2.5.so
000000314ab4c000    2048 ----- 000000000014c000 008:00002 libc - 2.5.so
.....
......
..
00002af8d48fd000       4 rw--- 0000000000006000 008:00002 xsl.so
00002af8d490c000      40 r - x-- 0000000000000000 008:00002 libnss_files -
2.5.so
00002af8d4916000    2044 ----- 000000000000a000 008:00002 libnss_files -
2.5.so
00002af8d4b15000       4 r---- 0000000000009000 008:00002 libnss_files -
2.5.so
00002af8d4b16000       4 rw--- 000000000000a000 008:00002 libnss_files -
2.5.so
00002af8d4b17000  768000 rw - s - 0000000000000000 000:00009 zero (deleted)
00007fffc95fe000      84 rw--- 00007ffffffea000 000:00000   [ stack ]
ffffffffff600000    8192 ----- 0000000000000000 000:00000   [ anon ]
mapped: 933712K    writeable/private: 4304K    shared: 768000K
```

Останній рядок дуже важливий:

`mapped: 933712K` загальна кількість пам'яті, відведеного під файли

`writeable/private: 4304K` загальна кількість приватного адресного простору

`shared: 768000K` загальна кількість адресного простору, який цей процес використовує спільно іншими процесами.

Утиліта `netstat` виводить інформацію про локальну мережу і засоби TCP/IP. Саме до неї найчастіше звертаються адміністратори, щоб швидко відшукати причину несправності в мережі TCP/IP. Зміст і форма вихідної інформації залежать від операційної системи, але зазвичай виводяться наступні дані: список з'єднань, статистика мережевих інтерфейсів, інформація по буферах даних, зміст таблиці маршрутизації, статистика роботи протоколу. Характер інформації, що виводиться, можна вибирати за допомогою опції командного рядка.

При виведенні параметрів утиліти на екран використовуйте команду `| more` для посторінкового виводу. Ключі команди наведені в таблиці 3.5

Таблиця 3.5 – Ключі та їх функції

Ключ	Функція
`-r route`	Виведення таблиці маршрутизації
`-i interfaces`	Виведення статистики мережевих інтерфейсів
`-s statistics`	Виведення статистики передачі даних (по протоколу SNMP)
`-n numeric`	Імена портів в числовому виді
`-N symbolic`	Імена портів в символічному виді
`-l listening`	Виведення стану портів, очікування, що знаходяться в режимі
`-a all`	Виведення стану усіх портів

Активні з'єднання через порти:	
-st	TCP
-u	UDP
-W	RAW
-X	LINUX

Команда netstat має набір ключів для відображення портів, що знаходяться в активному і пасивному стані. Таким чином, можна отримати список усіх серверних застосувань, працюючих на цьому комп'ютері.

Інформація виводиться стовпцями. У першому з них вказаний протокол, потім розміри черг прийому і передачі для встановленого з'єднання на цій машині (на іншому кінці з'єднання розміри черг можуть бути іншими), локальний і видалений адреси і поточний стан з'єднання.

Приклад виведення команди наступний:

```
st1@pds:~ > netstat - ta
Proto Recv - Q Send - Q Local Address       Foreign Address         State
tcp    0      2  cisco-academy.com.ua: telnet      olymp.cisco-academy.com.ua
:1288 ESTABLISHED
tcp    1      0 cisco-academy.com.ua :4550         cisco-academy.com.ua :3128
CLOSE_WAIT
tcp    1      0 cisco-academy.com.ua :4548         cisco-academy.com.ua :3128
CLOSE_WAIT
tcp       0          0  gw.cisco-academy.com.ua.:  netbios  -  ssn  marya.cisco-
academy.com.ua:1027   ESTABLISHED
tcp       0          0  gw.cisco-academy.com.ua.:  netbios  -  ssn  yanko.cisco-
academy.com.ua :1104   ESTABLISHED
tcp       0          0  gw.cisco-academy.com.ua.:  netbios  -  ssn  mumu.cisco-
academy.com.ua :1065    ESTABLISHED
tcp    0      0 *:6000                  *:*                     LISTEN
tcp    0      0 *:3128                  *:*                     LISTEN
tcp    0      0 *:53333                 *:*                     LISTEN
tcp    0      0 *:389                   *:*                     LISTEN
tcp    0      0 localhost :1032          localhost :1033          ESTABLISHED
tcp    0      0 *: netbios - ssn          *:*                     LISTEN
tcp    0      0 *: smtp                  *:*                     LISTEN
tcp    0      0 *: imap2                 *:*                     LISTEN
tcp    0      0 *: pop3                  *:*                     LISTEN
tcp    0      0 *: login                 *:*                     LISTEN
tcp    0      0 *: shell                 *:*                     LISTEN
tcp    0      0 *:8000                  *:*                     LISTEN
tcp    0      0 *: telnet                *:*                     LISTEN
tcp    0      0 *: ftp                   *:*                     LISTEN
tcp    0      0 *: time                  *:*                     LISTEN
tcp    0      0 *:www                   *:*                     LISTEN
tcp    0      0 *:2049                  *:*                     LISTEN
tcp    0      0 *:832                   *:*                     LISTEN
--More -
```

Як видно з прикладу, більшість серверів знаходяться в режимі очікування запиту на з'єднання (LISTEN). У першому рядку відбито з'єднання (ESTABLISHED) через `telnet` з машиною `olymp.cisco-academy.com.ua`. Стан CLOSE_WAIT означає, що з'єднання розірване, але перемикання в стан LISTEN ще не сталося; TIME_WAIT - що з'єднання чекає розриву. Якщо з'єднання знаходиться в стані SYN_SENT, то це означає наявність процесу, який намагається встановить з'єднання з неіснуючим сервером. Стан з'єднання має значення тільки для протоколу TCP. Протокол UDP факту встановлення з'єднання не перевіряє.

`Зміст таблиці маршрутизації`. Кожне з'єднання машини з мережею називається мережевим інтерфейсом. Машина, що має більше за один інтерфейс, може приймати дані по одному інтерфейсу і передавати їх по іншому, таким чином здійснюючи пересилку даних між мережами. Ця функція називається маршрутизацією, а машина, що виконує її, - шлюзом.

Ці дані маршрутизації зберігаються в одній з таблиць ядра. Для напряму пакету за конкретною адресою ядро підбирає найбільш відповідний маршрут. Якщо такий маршрут відсутній і немає маршруту за замовчуванням, то відправникові повертається повідомлення про помилку.

Команда `netstat -r` дозволяє відображати таблицю маршрутизації, визначення стовпців якої приведено в таблиці 3.6. Пункти призначення і шлюзи можуть показуватися або іменами машин, або їх IP -адресами. Прапори дають оцінку маршруту.

Приклад виконання команди наступний:

```
st1@pds:~ >  netstat - r
Kernel IP routing table
Destination    Gateway                  Genmask         Flags Ifac
pds.sut.ru     *                         255.255.255.255 UH     eth1
195.19.219.120 *                         255.255.255.248 U      eth0
195.19.219.128 *                         255.255.255.192 U      eth1
192.168.1.0    *                         255.255.255.0   U      eth0
195.19.221.0   lgw.cisco-academy.com.ua  255.255.255.0   UG     eth1
193.125.0.0    lgw.cisco-academy.com.ua  255.255.0.0     UG     eth1
loopback       *                         255.0.0.0       U      lo
default        lgw.cisco-academy.com.ua  0.0.0.0         UG     eth1
```

Таблиця 3.6 – Визначення полів таблиці маршрутизації

Назва стовпця	Визначення
Gatway	Імена використовуваних шлюзів
Genmask	Маска, використовувана для відображення загальної частини адреси, що відповідає цьому маршруту
Flags	Прапори, що описують маршрут: G Маршрут використовує шлюз
	U Інтерфейс активний, може використовуватися для передачі даних

Продовження таблиці 3.6

	H Дані можна передавати тільки одному вузлу
	D Запис створений повідомленням протоколу ICMP, що перенаправляє
	M Запис модифікований повідомленням протоколу ICMP, що перенаправляє
Iface	Інтерфейс, використовуваний для передачі пакетів

Статистика мережевих інтерфейсів.

При використанні ключа - і команди netstat на екран будуть виведені статистичні дані усіх використовуваних інтерфейсів. Виходячи з них, можна з'ясувати, чи справне з'єднання з мережею.

Приклад:

```
st1@pds:~ > netstat - i
Kernel Interface table
Iface MTU Met   RX - OK   RX - ERR  RX - DRP RX - OVR   TX - OK  TX - ERR  TX
- DRP TX - OVR Flg
```

```
eth0   1000   0     844904      0      17         0     1454454     5          0     0
BRU
eth0:   1000     0                                -   no   statistics   available   -
BRU
eth1   1500   0     590844     0       7          0     434438     59          0     0
BRU
lo     3924   0     45754      0       0          0     45754       0          0     0
LRU
```

Помилки є наслідком проблем в кабельній системі. У нормально працюючій мережі кількість конфліктів (RX - OVR, TX - OVR) не повинна перевищувати 3% від числа пакетів, а інші помилки не повинні складати більше 0,5% від загального числа пакетів.

Статистика передачі даних. Команда `netstat -s` видає вміст лічильників мережевих програм. У вихідній інформації є розділи, що відносяться до різних протоколів : IP, ICMP, TCP, UDP. З її допомогою можна визначити місце появи помилки в прийнятому пакеті.

Приклад використання команди наступний:

```
st1@pds:~ > netstat - s
Ip:
    179495 total packets received      13 with invalid     headers      8753
forwarded
    0 incoming packets discarded
    168812 incoming packets delivered
    325599 requests sent out
    544 fragments failed
Icmp:     728 ICMP messages received
    3 input ICMP message failed
    ICMP input histogram:
        destination unreachable: 82
        timeout in transit: 55
        source quenchs: 9
        echo requests: 582
    1235 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 646
        time exceeded: 6
        redirect: 1
        echo replies: 582
Tcp:
    2428 active connections openings
    0 passive connection openings
    0 failed connection attempts
    0 connection resets received
    17 connections established
    154840 segments received
    318758 segments send out
    1480 segments retransmited
    99 bad segments received.
    499 resets sent
Udp:
    13397 packets received
    73 packets to unknown port received.
```

```
    12 packet receive errors
    5608 packets sent
TcpExt:
    15 resets received for embryonic SYN_RECV sockets
```

3.2.2 Завдання до другої частини лабораторної роботи

Зміст завдань відповідно до варіанту зазначений в таблиці 3.7.

Таблиця 3.7 – Варіанти завдання

№ бригади	Завдання
1	Виконайте видачу даних про активність процесів з включенням видачі на екран сумарних даних. Визначіть активність системи, інформацію про систему і апаратні ресурси

Продовження таблиці 3.7

2	Виконайте видачу даних про активність процесів з виключенням видачі на екран сумарних даних. Виведіть статистику використання пам'яті.
3	Виконайте видачу даних про активність процесів з сортуванням рядків по максимальному споживанню різних системних ресурсів. Отримайте дані про активність/неактивність сторінок пам'яті.
4	Виконайте видачу даних про активність процесів. Задайте порядок рядків видаваною командою `top`. Визначте зареєстрованих користувачів і що вони роблять.
5	Виконайте видачу даних про активність процесів. Змініть пріоритет процесів. Визначте, як довго працює система. Виведіть статистику використання пам'яті.
6	Виконайте видачу даних про активність процесів. Видаліть процес. Виведіть список процесів. Отримайте дані про

	активність/неактивність сторінок пам'яті.
7	Виконайте видачу даних про активність процесів. Перемикніть на монохромний варіант видачі зображення. Включіть режим максимальної видачі даних.
8	Виконайте видачу даних про активність процесів з включенням видачі на екран сумарних даних. Покажіть потоки після процесів.
9	Виконайте видачу даних про активність процесів. Покажіть потоки (LWP і NLWP). Визначіть активність системи, інформацію про систему і апаратні ресурси.
10	Виконайте видачу даних про активність процесів з сортування рядків по максимальному споживанню різних системних ресурсів. Покажіть кожен процес для будь-якого користувача.

Продовження таблиці 3.7

11	Виконайте видачу даних про активність процесів. Налаштуйте видачу даних у форматі, визначеному користувачем.
12	Виконайте видачу даних про активність процесів. Покажіть ID процеси, запущених під Lighttpd. Визначіть 10 процесів, споживаючих найбільшу кількість пам'яті.
13	Виконайте видачу даних про активність процесів. Видаліть процес. Покажіть ім'я для PID 55977. Виконайте видачу даних про активність процесів з сортування рядків по максимальному споживанню різних системних ресурсів.
14	Виконайте видачу даних про активність процесів Перемикніть на монохромний варіант видачі зображення. Визначіть 10 процесів, споживаючих найбільшу кількість пам'яті. Визначіть активність системи, інформацію про систему і апаратні ресурси.

15	Визначіть 10 процесів, споживаючих найбільший ресурс процесора. Отримайте дані про активність/неактивність сторінок пам'яті.
16	Визначте, як довго працює система. Виконайте видачу даних про активність процесів. Виконайте видачу даних про активність процесів з сортування рядків по максимальному споживанню різних системних ресурсів.
17	Виконайте видачу даних про активність процесів з сортуванням рядків по максимальному споживанню різних системних ресурсів. Визначте зареєстрованих користувачів і що вони роблять.
18	Виведіть статистику передачі даних та зміст таблиці маршрутизації. Визначіть використання процесами оперативної пам'яті. Налаштуйте видачу даних у форматі, визначеному користувачем.

Продовження таблиці 3.7

19	Виведіть статистику мережевих інтерфейсів та значення зчитувача мережевої активності. Визначіть активність системи, інформацію про систему і апаратні ресурси. Покажіть кожен процес для будь-якого користувача.
20	Визначіть використання процесами оперативної пам'яті. Виведіть дані про активність кожного наявного в наявність процесора. Визначіть використання процесами оперативної пам'яті.
21	Виведіть дані про використання мультипроцесора. Визначіть значення зчитувача мережевої активності. Визначіть активність системи, інформацію про систему і апаратні ресурси.

22	Визначіть значення лічильників мережевої активності, починаючи з 24-го. Виведіть дані про використання пам'яті. Покажіть кожен процес для будь-якого користувача.
23	Визначте середнє завантаження процесора, активність дисків. Виведіть дані про використання пам'яті процесом з pid # 47394. Визначте зареєстрованих користувачів і що вони роблять.
24	Визначте список з'єднань. Визначте дані про системну активність. Отримайте дані про активність/неактивність сторінок пам'яті. Визначіть використання процесами оперативної пам'яті.
25	Виконайте видачу даних про активність процесів. Покажіть потоки після процесів. Визначіть 10 процесів, споживаючих найбільшу кількість пам'яті. Визначіть активність системи, інформацію про систему і апаратні ресурси.
26	Змініть пріоритет процесів. Визначте, як довго працює система. Налаштуйте видачу даних у форматі, визначеному користувачем. Отримайте дані про активність/неактивність сторінок пам'яті.

Продовження таблиці 3.7

27	Отримайте дані про активність/неактивність сторінок пам'яті з виключенням видачі на екран сумарних даних. Виведіть статистику використання пам'яті. Визначіть використання процесами оперативної пам'яті.
28	Налаштуйте видачу даних у форматі, визначеному користувачемВизначіть 10 процесів, споживаючих найбільшу кількість пам'яті. Визначте зареєстрованих користувачів і що вони роблять.
29	Виконайте видачу даних про активність процесів з сортуванням

	рядків по максимальному споживанню різних системних ресурсів. Визначіть використання процесами оперативної пам'яті.
30	Визначіть активність системи, інформацію про систему і апаратні ресурси. Визначіть використання процесами оперативної пам'яті. Отримайте дані про активність/неактивність сторінок пам'яті.

3.2.3Контрольні запитання до другої частини лабораторної роботи

1. Для чого потрібна команда `top`?

2. Що робить команда `vmstat`?

3. Як отримати інформацію про те, які користувачі зараз знаходяться в системі і які процеси запущені від їх імені?

4. З якою метою можно використати команду `uptime`?

5. Яка команда видає короткий список поточних процесів?

6. Для чого використовується команда `sar`?

7. Як отримати дані про розподіл пам'яті між процесами?

ЛАБОРАТОРНА РОБОТА №4. УТИЛІТИ LINUX

4.1 Ознайомлення з утилітами системи Linux

4.1.1 Теоретичні відомості

У складі операційної системи Linux є велика кількість системних утиліт, призначених для обробки текстів. Утиліти `cat` і `grep`, з якими Ви вже повинні були познайомитися, відносяться до їх числа. Інші утиліти такого роду: `cmp` - порівняння файлів, `cut` - "вирізування" полів з тексту та `paste` – зчеплення рядків файлів, `head` – роздруківка початку файлу і `tail` – роздруківка останніх рядків файлу, `sort` – сортування, `join` – об'єднання, `sed` – потоковий текстовий редактор і багато інших.

Кожна з таких утиліт виконує в принципі просту обробку текстового файлу, але послідовно застосовуючи одну утиліту за одною, можна скомбінувати їх дії таким чином, що підсумкове перетворення тексту буде досить складним. Обробка тексту за допомогою послідовних викликів системних утиліт називається в Linux "фільтрацією" тексту, а самі утиліти називаються фільтрами. Такі назви походять від метафоричного подання проходження потоку даних через ряд фільтрів, кожний з яких проводить відбір якихось необхідних складових, в результаті чого ми отримуємо ті дані, які нам потрібні - "відфільтровані" дані.

У ланцюжка фільтрації можуть включатися й інші команди операційної системи, наприклад, команди файлової системи. Параметри і результати роботи цих команд представляються у вигляді символьних рядків, тому вони теж можуть бути оброблені текстовими фільтрами.

Як правило більшість команд Linux читає вхідні дані зі стандартного вводу (клавіатура) і направляє вихідні дані в потік стандартного виводу (екран). Як правило, одним із параметрів команди є ім'я (імена) файлу

(файлів), який (які) вона обробляє. Якщо таке ім'я не задано, команда читає вхідні дані з стандартного вводу. Якщо у команді може здаватися декілька файлів, то зазвичай стандартний ввод позначається серед імен файлів символом '-'.

4.1.2 Порядок виконання першої частини лабораторної роботи

Завдання 1

Робота всіх варіантів демонструється на обробці файлу з ім'ям `Hum-Dum.txt`, вихідне вміст якого показано в наступному протоколі:

```
Script started on Thu Sep 5 7:56:10 2002
bash2-2.05 $ cat Hum-Dum
Humpty-Dumpty
Set on the wall.
Humpty-Dumpty
Had a greate fall.
And all the king's horses,
And all the king's man.
Can not Humpty,
Can not Dumpty,
Humpty-Dumpty,
Dumpty-Humpty,
Set on this wall
Again.
bash2-2.05 $
Script done on Thu Sep 5 7:56:21 2002
```

Нижче наводяться деякі загальні міркування щодо вирішення завдань.

Міркування щодо 1-го завдання.

У більшості випадків ми можемо легко сформулювати оператори потокового редактора, необхідні для виконання заданого перетворення, за винятком одного компонента – адреси рядка, до якого це перетворення повинно бути застосоване. У деяких випадках нам також заздалегідь невідомий і текст, який необхідно вставити в цільовий рядок. Тому типова схема розв'язання наступна:

1. Визначаються номери рядків, до яких повинні бути застосовані перетворення, а при необхідності - також і текст, який повинен вставлятися в цільовий файл. Для визначення застосовуються статичні (заздалегідь відомі і закладені в текст команд) команди потокового редактора. Результати цього кроку зберігаються у файлі.

2. Шляхом редагування результатів 1-го кроку динамічно формуються команди `sed` для виконання перетворення, що містять адреси і тексти, визначені на 1-му кроці. Ці команди зберігаються у файлі.

3. Виконується редагування вихідного тексту командами, записаними на 2-му кроці. Можливі два підходи до отримання номерів рядків, що підлягають перетворенню:

* Виконати нумерування всіх рядків вихідного файлу (це можна зробити командою `pr` або командою `cat` з опцією `-n`), а потім з пронумерованої послідовності рядків вибрати рядок з його номером, призначений для обробки; такий прийом застосований у рішеннях для варіантів 1 і 2;

* Вибрати номер рядка, призначеного для обробки, за допомогою команди `sed "=";` такий прийом застосований для варіанту 3.

Оскільки Linux має велике число утиліт для обробки текстів, причому функції багатьох утиліт перекриваються, кожнт із запропонованих завдань може бути вирішент безліччю різних способів.

Приклади розв'язання завдання 1

Завдання 1, варіант 1

Кожне друге слово кожного рядка вивести в окремий наступний рядок. Якщо в рядку тільки одне слово, нічого не робити.

Рішення:

```
pr-n ''-T Hum-Dum.txt |
```

Вивести файл без заголовка і зайвих рядків, але з номерами рядків.

```
sed-n 's / ^ * / / p' |
```

Видалити головні пробіли.

```
cut-f1, 3-d '' |
```

Вирізати номер і 2-е слово.

```
sed-n '/ [^ 0-9] / p'> temp01
```

Видалити рядки, які містять тільки номер, результат зберегти в 1-му тимчасовому файлі.

```
cut-f1-d '' temp01 |
```

Вибрати з 1-го тимчасового файлу номери рядків.

```
sed-n 's / $ / a \ \ / p'> temp02
```

Додати до них `'a \'` і зберегти в 2-му тимчасовому файлі.

```
cut-f2-d '' temp01> temp03
```

Вибрати з 1-го тимчасового файлу слова і зберегти в 3-м тимчасовому файлі.

```
paste-d '\ n' temp02 temp03> temp01
```

Зчепити порядково 2-й і 3-й тимчасові файли, вставивши між зчіплюваними рядками перевід рядка. Результат зберегти в 1-му тимчасовому файлі. Результат - набір команд `sed` на вставку.

```
sed-f temp01 Hum-Dum.txt> result
```

Виконати команди з 1-го тимчасового файлу, зберегти результат.

```
rm-f temp *
```

Видалити тимчасові файли.

Протокол виконання даного завдання наступний:

```
Script started on Thu Sep 5 8:00:29 2002
bash2-2.05 $ pr-n ''-T Hum-Dum.txt |
> Sed-n 's / ^ * / / p' |
> Cut-f1, 3-d '' |
> Sed-n '/ [^ 0-9] / p'> temp01
bash2-2.05 $ cut-f1-d '' temp01 |
> Sed-n 's / $ / a \ \ / p'> temp02
bash2-2.05 $ cut-f2-d '' temp01> temp03
bash2-2.05 $ paste-d '\ n' temp02 temp03> temp01
bash2-2.05 $ sed-f temp01 Hum-Dum.txt> result
bash2-2.05 $ rm-f temp *
bash2-2.05 $ cat result
Humpty-Dumpty
Set on the wall.
on
```

```
Humpty-Dumpty
Had a greate fall.
a
And all the king's horses,
all
And all the king's man.
all
Can not Humpty,
not
Can not Dumpty,
not
Humpty-Dumpty,
Dumpty-Humpty,
Set on this wall
on
Again.
bash2-2.05 $
Script done on Thu Sep 5 8:00:47 2002
```

Зверніть увагу на те, що в наведеному вище протоколі деякі запрошення системи виглядають як "$", а деякі - як ">". Система друкує запрошення ">" (вторинне запрошення), якщо попередня команда закінчується ознакою конвеєра "|" і, отже, обов'язково очікується введення наступної команди. В іншому випадку друкується первинне запрошення - "$". Символи первинного і вторинного запрошення визначаються змінними оточення PS1 і PS2 відповідно.

Міркування щодо завдання 2.

Робота всіх варіантів цього завдання відбувається на наборі файлів: ../metod/query *., query1, query2, query3, query4, query5. Ті, хто не доклав ще достатньо зусиль для того, щоб забути наш курс "Організація баз даних", без праці дізнаються в цих файлах результати виконання запитів до бази даних "Корпорація Кінга". Ці результати являють собою таблиці (реляційні таблиці) і складають як би базу даних, схема якої наведена на рисунку 4.1.

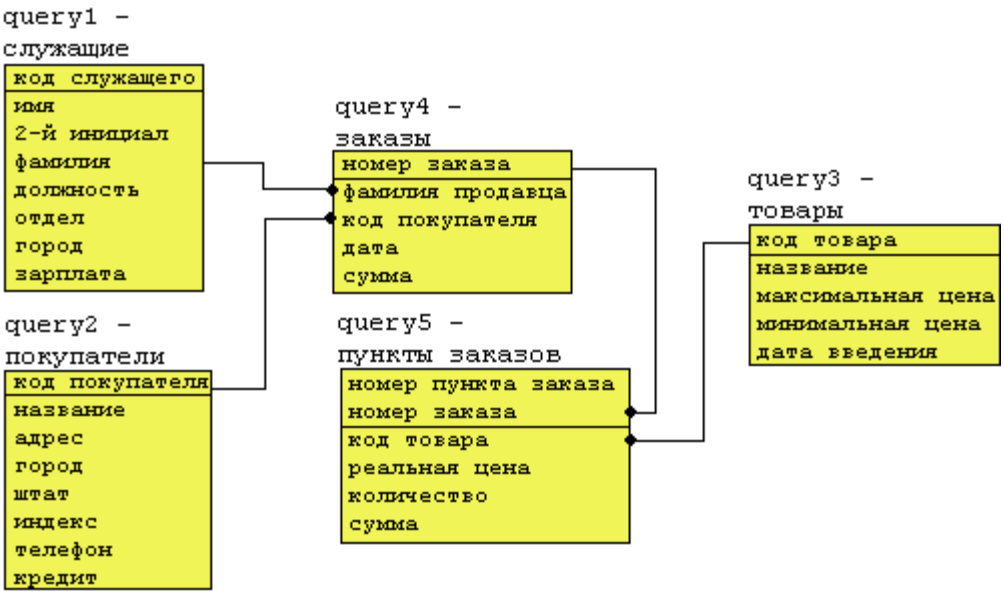


Рисунок 4.1 – Схема бази даних

Варіанти завдання 2 представляють собою завдання на вибірку даних з таблиць "цієї" бази даних ". Утиліти Linux надають у наше розпорядження такі засоби, які в тій чи іншій мірі можуть служити аналогом реляційних операцій:

* Операція проекції може бути здійснена вирізанням певних полів рядка – командою `cut`.

* Операція обмеження може бути здійснена будь-якою з утиліт, що здійснюють пошук рядка по шаблону – `grep` або `sed`.

* Операція природного з'єднання може бути здійснена утилітою `join`. Слід однак пам'ятати, що для застосування утиліти `join` таблиці повинні бути відсортовані за тим стовпчиком, за яким відбувається з'єднання, це можна зробити за допомогою утиліти `sort`.

* Дублікати у файлах-таблицях можуть бути усунені за допомогою утиліти `uniq` або утиліти `sort` з опцією `-u`. Слід мати на увазі, що в першому випадку дублікатами вважаються збіги повних рядків, а в другому - тільки тих полів, за якими виконується сортування.

* Розглянуті нами утиліти не надають тих можливостей, які надають агрегатні функції SQL, проте функції MAX () та MIN () можна

промоделювати, виконавши сортування таблиці і вибравши потім перший (утиліта `head`) або останній (утиліта `tail`) рядок.

У поясненнях до наших прикладів виконання ми часто використовуємо термінологію реляційних операцій.

Більшість утиліт, що працюють з полями форматованого тексту, за замовчуванням припускають символом-роздільником полів символ табуляції. Однак працювати з символом табуляції незручно, оскільки він за замовчуванням явно не відображається. Тому ми рекомендуємо призначати роздільником будь-який відображуваний символ. Зверніть увагу на те, що в наших файлах-таблицях використовуються різні роздільники полів. Для виконання операції з'єднання необхідно встановити загальний розділювач для обох таблиць, що з'єднуються.

Ми завжди виконували проекцію (відбір необхідних стовпців) перш, ніж з'єднання. Можливо виконувати проекцію і після з'єднання. У другому випадку може навіть бути заощаджено кілька команд, але попереджуючий відбір тільки необхідних стовпців зменшує обсяг проміжних результатів, чим суттєво полегшує відладку.

Приклади розв'язання завдання 1

Завдання 2 варіант 1

Визначити прізвища продавців, які виконували замовлення, що складаються з більш ніж 5 пунктів.

Рішення:

```
sed 's / \ (1, \) / \: / g' ../metod/query5 |
```

Встановлення у файлі `query5` роздільника **":"**.

```
cut-f1, 2-d ':' |
```

Проекція за номером пункту і номером замовлення.

```
sort 0 -1-t ':'-n |
```

Числова опція `-n` – сортування за номером пункту.

```
sed '/ ^ [1-4]: / d'> temp01
```

Видалення тих рядків, в яких номер пункту 1-4.

```
sort 1 -2-t ':'> temp01
```

Сортування за номером замовлення. Результат зберігається в `temp01`.

```
sed 's / \ (1, \) / \: / g' ../metod/query4 |
```

Встановлення у файлі `query4` роздільника **":"**.

```
cut-f1, 2-d ':' |
```

Проекція за номером замовлення та прізвищем продавця.

```
sort 0 -1-t ':' |
```

Сортування за номером замовлення.

```
join-j1 1-j2 2-t ':' - temp01 |
```

З'єднання зі збереженим в `temp01`.

```
cut-f2-d ':' |
```

Проекція на прізвищем продавця.

```
sort 0 -1-t ':'-u> result
```

Сортування за прізвищем продавця з усуненням дублікатів.

```
rm-f temp *
```

Видалення тимчасового файлу.

Протокол виконання даного завдання наступний:

```
Script started on Thu Sep 5 8:13:56 2002
bash2-2.05 $ sed 's / \ (1, \) / \: / g' ../metod/query5 |
> Cut-f1, 2-d ':' |
> Sort 0 -1-t ':'-n |
> Sed '/ ^ [1-4]: / d' |
> Sort 1 -2-t ':'> temp01
bash2-2.05 $ sed 's / \ (1, \) / \: / g' ../metod/query4 |
> Cut-f1, 2-d ':' |
> Sort 0 -1-t ':' |
> Join-j1 1-j2 2-t ':' - temp01 |
> Cut-f2-d ':' |
> Sort 0 -1-t ':'-u> result
bash2-2.05 $ rm-f temp *
bash2-2.05 $ cat result
DUNCAN
ROSS
SHAW
TURNER
WARD
bash2-2.05 $
Script done on Thu Sep 5 8:14:14 2002
```

Міркування щодо завдання 3.

Джерелом вхідних даних для всіх варіантів цього завдання може бути команда друку вмісту каталогів ls або команда пошуку файлів find. Вихідний потік першої команди направляється в конвеєр, де він може послідовно оброблятися командами обробки текстів: grep, sed, cut, sort і т.п. Якщо в завданні потрібно підрахувати число елементів, в останньому ланці конвеєра може бути застосована команда wc.

При виконанні завдання слід мати на увазі, що користувацькі групи в системі збігаються з кодами студентських груп (наприклад: "ap109", "ap070b" і т.д.), всі коди починаються з літер "ap"; а імена користувачів формуються як: імя_групиnn, де nn – порядковий номер у групі.

Приклади розв'язання завдання 3

Завдання 3 варіант 1

Визначити загальну кількість студентських груп.

Рішення:

```
ls-ld .. / * |
```

Команди виконуються у домашньому каталозі користувача - /home/ім'я_користувача, а інформацію про створені для груп каталогах можна отримати з каталогу /home/, який може адресуватися з поточного каталогу як: ../. Виводимо інформацію про вміст цього каталогу. Опція -l вказується, щоб отримати повну інформацію, включаючи групу, опція -d запобігає обходу підкаталогів. Друк команди ls перенаправляється в потік.

```
grep "^ d. \ (24 \) ap" |
```

Ознака підкаталогу - буква "-d" у першій позиції видачі команди ls, а імена груп починаються с 25-ї позиції видачі. Команді grep задається шаблон, який визначає ознаку каталогу в 1-й позиції і назву групи, що починається з літер "ap".

```
sed-n 's / [] \ (2, \) / / gp' |
```

Оскільки далі буде потрібно виділяти поля, позбудемося множинних пробілів за допомогою команди `sed`.

```
cut-f4-d '' |
```

Виділяється 4-е поле, що містить назву групи.

```
sort |
```

Результат сортується, це знадобиться для наступної команди. Оскільки зараз в тексті залишився тільки один стовпець, ніяких опцій для сортування ми не вказуємо.

```
uniq |
```

Одна і та ж група повторюється для багатьох каталогів, тому варто позбутися від повторюваних рядків.

```
wc-l
```

Команда `wc` підраховує кількість рядків, що залишилися, результат виводиться на друк.

Протокол виконання завдання наступний:

```
Script started on Thu Sep 5 8:20:56 2002
bash2-2.05 $ ls-lad .. / * | grep "^ d. \ (24 \) ap" | sed-n 's / [] \ (2, \)
/ / gp' | cut-f4-d ' '|
> Sort 0 -1 | uniq | wc-l
      7
bash2-2.05 $
Script done on Thu Sep 5 8:21:03 2002
```

4.1.3Завдання до першої частини лабораторної роботи

Звіт по лабораторній роботі повинен містити тексти трьох послідовностей команд та протокол їх виконання. Для протоколювання роботи використовуйте команду `script`.

Варіант 1

1. В одному з текстових файлів практичної роботи N2 перенести третій від кінця рядок у початок файлу.

2. З інформації, що міститься у файлах `query ...`, визначити відділи (назва та місто), які отримували замовлення на загальну суму більше 1000.

3. Визначити кількість груп користувачів.

Варіант 2

1. В одному з текстових файлів практичної роботи N2 перенести третій від початку рядок в кінець файлу.

2. З інформації, що міститься у файлах `query ...`, визначити міста, в яких розташовані відділи, які виконували замовлення в лютому 1991 р.

3. Визначити кількість підкаталогів в `/home`, до яких немає публічних прав доступу.

Варіант 3

1. В одному з текстових файлів практичної роботи N2 перед кожним рядком вставити поточний час.

2. З інформації, що міститься у файлах `query ...`, визначити прізвища продавців, які виконували замовлення на поставку товару 'SP JUNIOR RACKET'.

3. Визначити кількість підкаталогів в `/home`, до яких є публічні права на пошук і читання в них.

Варіант 4

1. В одному з текстових файлів практичної роботи N2 видалити другий рядок, що починається з букви 'H'.

2. З інформації, що міститься у файлах `query ...`, визначити штат, в якому було зроблено замовлення на найбільшу загальну суму.

3. Визначити кількість користувачів з вашої студентської групи.

Варіант 5

1. В одному з текстових файлів практичної роботи N2 залишити в кожному рядку не більше двох слів. Слова, що виходять за цю межу помістити в окремий файл. У першому файлі-результаті в тих рядках, які містять менше двох слів, в кінець рядка має бути додано символ '='. У другому файлі-результаті порожніх рядків залишатися не повинно, а перед непустими рядками повинні бути вказані їх номери у вихідному файлі, відокремлені від решти тексту одним пропуском.

2. З інформації, що міститься у файлах `query` ..., визначити товар, якого було замовлено найбільшу кількість примірників одночасно.

3. Визначити кількість (не підкаталогів і не посилань) файлів в каталозі `/home`.

Варіант 6

1. В одному з текстових файлів практичної роботи N2 залишити в кожному рядку не більше 2-х слів. Залишок перенести в наступний рядок. Якщо другий рядок в парі виявляється порожнім – надрукувати в ньому символ '='.

2. З інформації, що міститься у файлах `query` ..., визначити 5 покупців, які зробили замовлень на найбільшу загальну суму в 1990 р.

3. Визначити кількість файлів або підкаталогів в кореневому каталозі, до яких всі мають повні права доступу.

Варіант 7

1. В одному з текстових файлів практичної роботи N2 перший символ кожного рядка замінити на перший символ попереднього рядка. Перший рядок залишається без змін.

2. З інформації, що міститься у файлах `query` ..., визначити назви товарів, які продавалися за мінімальною ціною.

3. Визначити кількість файлів в каталозі `/etc`, які є символічними посиланнями.

Варіант 8

1. В одному з текстових файлів практичної роботи N2 поміняти місцями два перших і два останніх символи кожного рядка.

2. З інформації, що міститься у файлах `query` ..., визначити прізвище продавця, який продав товар 'SP JUNIOR RACKET' на максимальну суму в одному замовленні.

3. Визначити кількість файлів в каталозі `/etc`, на які є більше одного жорсткого посилання.

Варіант 9

1. В одному з текстових файлів практичної роботи N2 поміняти місцями перший і останній рядки файлу.

2. З інформації, що міститься у файлах `query` ..., визначити прізвище продавця, який першим продав товар 'SP JUNIOR RACKET' в 1991 р.

3. Визначити кількість файлів в каталозі `/etc`, які створені не в цьому році.

Варіант 10

1. В одному з текстових файлів практичної роботи N2 після рядків, які закінчуються крапкою або комою, вставити пустий рядок.

2. З інформації, що міститься у файлах `query` ..., визначити назви товарів, які першими були виставлені у продаж.

3. Вибрати алфавітний список підкаталогів в `/home`, до яких немає публічних прав.

Варіант 11

1. В одному з текстових файлів практичної роботи N2 перенести останнє слово в кожному рядку в новий рядок. Для рядків, що складаються з одного слова - не робити нічого.

2. З інформації, що міститься у файлах `query ...`, визначити назви товарів, які замовлялися разом з товаром 'SP JUNIOR RACKET'.

3. Вибрати алфавітний список підкаталогів в `/home`, до яких є публічні права на пошук і читання в них.

Варіант 12

1. В одному з текстових файлів практичної роботи N2 перенести перше слово кожного рядка в початок наступного рядка.

2. З інформації, що міститься у файлах `query ...`, визначити назви товарів, які коли-небудь замовляв покупець 'JOCKSPORTS'.

3. Вибрати алфавітний список користувачів з вашої студентської групи.

Варіант 13

1. В одному з текстових файлів практичної роботи N2 у всіх парних рядках перенести перше слово рядка в кінець рядка. Рядки, які містять лише одне слово, не змінюються.

2. З інформації, що міститься у файлах `query ...`, визначити прізвища продавців, які коли-небудь продавали товари за їх максимальною ціною.

3. Вибрати алфавітний список файлів (не підкаталогів, не посилань) в каталозі `/home`.

Варіант 14

1. В одному з текстових файлів практичної роботи N2 у всіх непарних рядках перенести останнє слово рядка на початок рядка. Рядки, які містять лише одне слово, не змінюються.

2. З інформації, що міститься у файлах `query ...`, визначити назви товарів, на які не було замовлень у 1990 р.

3. Вибрати алфавітний список файлів в каталозі `/etc`, на які є більше одного жорсткого посилання.

Варіант 15

1. В одному з текстових файлів практичної роботи N2 поміняти місцями парні рядки з непарними.

2. З інформації, що міститься у файлах `query` ..., визначити назви товарів, які не були в продажу в 1990 р.

3. Вибрати алфавітний список файлів в каталозі `/etc`, які створені не в цьому році.

4.1.4Контрольні запитання до першої частини лабораторної роботи

1Яка утиліта порівнює файли?

2Що виконує утиліта `tail`?

3Як називається в Linux обробка тексту за допомогою послідовних викликів системних утиліт?

4Як перенаправити стандартний поток вводу?

5Як можуть бути усунені дублікати у файлах-таблицях?

4.2 Команди пошуку в Linux

4.2.1Теоретичні відомості

Пакет Findutils містить програми для пошуку файлів, зазначені у таблиці 4.1, у тому числі "на льоту" (шляхом рекурсивного пошуку від директорії і показуючи тільки файли, що задовольняють параметрам пошуку) або пошук через базу даних.

Таблиця 4.1– Короткі описи

Команда	Визначення
`bigram`	Раніше використовувався для створення пошукових баз

	даних.
code	Раніше використовувався для створення пошукових баз даних. Попередник frcode.

Продовження таблиці 4.1

find	Шукає всередині вказаного дерева каталогів файли, відповідні заданим критеріям.
frcode	Викликається updatedb для стискування списку імен файлів. Вона використовує front – стиснення, зменшуючи розмір бази даних в 4-5 разів.
locate	Шукає в базі цих імен файлів і виводить імена, що містять заданий рядок або відповідні шаблону.
updatedb	Оновлює пошукову базу даних. Він сканує усю файлову систему (включаючи інші смонтовані файлові системи, якщо не вказано зворотнє) і заносить усі знайдені імена файлів в базу даних.
xargs	Може використовуватися для виконання команди із списком файлів. Команда find як файловий апогей.

У цій частини роботи мова піде про набір первинних команд, відомий в проекті GNU як findutils. І насамперед - про команду find (як, втім, і про тісно пов'язану з нею команду xargs). Така висока честь випадає їм тому, що за допомогою цих двох команд можна вирішити якщо не усі, то більшість проблем, що виникають при роботі з файлами.

Отже, файловий апогей – команда find. Строго кажучи, всупереч своєму імені, команда ця виконує не пошук файлів як такий, але – рекурсивний обхід дерева каталогів, починаючи із заданого як аргумент, відбирає з них файли у відповідність за деякими критеріями і виконує над

відбракованими файлами деякі дії. Саме цю її особливість підкреслює резюме команди `find`, що отримується (у деяких системах) за допомогою:

```
$ $ whatis find
find(1)                  - walk a file hierarchy
```

що стосовно випадку можна перевести як "прогулянка по файловій системі".

Команда `find` по своєму синтаксису істотно відрізняється від більшості інших команд. У загальненому виді формат її можна представити таким чином:

```
$ $ find аргумент [опція_пошуку] [значення] [опція_дії]
```

Аргумент – це шлях пошуку, тобто каталог, починаючи з якого слід шукати файли, наприклад, кореневий

```
$ $ find / [опція_пошуку] [значення] [опція_дії]
```

чи домашній каталог користувача

```
$ $ find ~/ [опція_пошуку] [значення] [опція_дії]
```

Опція пошуку – критерій, за яким слід шукати файл (файли). Такими можуть виступати ім'я файлу (`-name`), його тип (`-type`), атрибути приналежності, доступу або часу. Ну а опція дії визначає, що ж належить зробити зі знайденим файлом або файлами. А зробити з ними можна немало – починаючи з виводу на екран (`-print`) і закінчуючи передачою результатів у вигляді аргументів будь-якій іншій команді (`-exec`).

Як можна помітити, опція пошуку і опція дії передують знаком дефіса, значення першої відділяється від її імені пробілом. Проте почнемо по порядку. Опції пошуку команди `find` дозволяють виконати вищезазначений пошук за наступними критеріями (символ дефіса перед опціями нижче опущений, але не слід забувати його ставити), вказаних у таблиці 4.2.

Таблиця 4.2 – Критерії пошуку команди `find`

Критерій пошуку	Визначення
`name`	Пошук за іменем файлу або за маскою імені; у останньому випадку метасимволи маски повинні обов'язково екрануватися (наприклад, `-name *.tar.gz`) або братися в лапки (одинарні або подвійні, залежно від правил, прийнятих для цієї командної

	оболонки; цей критерій чутливий до регістра, але близький за сенсом критерій `iname` дозволяє здійснювати пошук за іменем без розрізнення малих і великих літер.

Продовження таблиці 4.2

`type`	Пошук за типом файлу; цей критерій набуває наступних значень – `f` (регулярний файл), `d` (каталог), `s` (символічний зв'язок), `b` (файл блокового пристрою), `c` (файл символьного пристрою).
`user i group`	Пошук за іменем або ідентифікатором власника або групи, що виступає значенням критерію; існує також критерії `nouser` і `nogroup` – вони відшукують файли, що не мають власників (тобто тих, облікові записи для яких відсутні у файлах `/etc/passwd` і `/etc/group`); ці критерії не мають потреби в значеннях.
`size`	Пошук за розміром, що задається у вигляді числа в блоках або в байтах, - у вигляді числа з наступним символом `c`; можливі значення `n` (рівно `n` блоків), `+n` (більші за `n` блоки), `−n` (менші `n` блоків).
`perm`	Пошук файлів за значенням їх атрибутів доступу, що задаються в символьній формі.
`atime, ctime, mtime`	Пошук файлів з вказаними тимчасовими атрибутами; значення тимчасових атрибутів вказуються в добі (точніше, в періодах, кратних 24 годинам); можливі форми значень цих атрибутів: `n` (рівно вказаному значенню n*24 години), `+n` (раніше n*24 годин), `−n` (пізніше n*24 годин).

newer	Пошук файлів, створених після файлу, вказаного як значення критерію (тобто, який має менше значення mtime).

Продовження таблиці 4.2

maxdepth, mindepth	Дозволяють конкретизувати глибину пошуку у вкладених підкаталогах - меншу або рівну чисельному значенню для першого критерію і більшу або рівну - для другого.
depth	Проводить відбір в зворотному порядку, тобто не від каталогу, вказаного як аргумент, а з найглибше вкладених підкаталогів; сенс цієї дії - дістати доступ до файлів в каталозі, для якого користувач не має права читання і виконання.
prune	Дозволяє вказати підкаталоги всередині шляху пошуку, в яких відбір файлів проводити не слід.

Окрім цього, існує ще одна опція пошуку -fstype, що виконує пошук тільки у файловій системі вказаного типу; очевидно, що вона може поєднуватися з будь-якими іншими опціями пошуку. Наприклад, команда

```
$ $ find / -fstype ext3 -name zsh*
```

шукатиме файли, що мають відношення до оболонки Z-Shell, починаючи з кореня, але тільки в межах тих розділів, на яких розміщена файлова система Ext3fs (на тестовій машині, - це саме корінь, за вирахуванням каталогів /usr, /opt, /var, /tmp і, звичайно ж, /home.

Критерії відбору файлів можуть групуватися практично будь-яким чином. Так, у формі

```
$$find ~/ -name *.tar.gz newer filename
```

вона вибере в домашньому каталозі користувача усі компресовані архіви, створені після файлу з ім'ям `filename`. За замовчуванням між критеріями відбору передбачається наявність логічного оператора `"I"`. Тобто шукатимуться файли, що задовольняють і масці імені, і відповідному атрибуту часу. Якщо вимагається використання оператора `"АБО"`, він має бути явно визначений у вигляді додаткової опції `-o` між опціями пошуку. Так, команда:

```
$$find ~/ -mtime -2 -o newer filename
```

покликана відібрати файли, створені менше двох діб тому, або ж – пізніше, ніж файл `filename`.

Особливість GNU-реалізації команди `find` (як, втім, і її тезки з числа BSD-утиліт) – та, що вона за замовчуванням виводить список відібраних відповідно до заданих критеріїв файлів на екран, не вимагаючи додаткових дій. Проте, як то кажуть, в інших Linux-системах (пам'ятається, навіть і в деяких реалізаціях Linux) вказівка якої-небудь з таких опцій – обов'язкова.

Отже розглянемо їх по порядку. Для виведення списку відібраних файлів на екран в загальному випадку призначена опція `-print`. Результат має наступний вигляд:

```
find . -name f* -print
././file1
././file2
././dir1/file3
```

Схожий сенс має і опція `-ls`, проте вона виводить повніші відомості про знайдені файли, аналогічно команді `ls` з опціями `-dgils`:

```
$ $ find / -fstype ext3 -name zsh -ls
88161  511 - rwxr - xr - x   1 root root 519320 Ниючи 23 15:50 /bin/zsh
```

Важливе, як мені здається, зауваження. Якщо команда вказаного виду буде дана від імені звичайного користувача (не root-оператора), окрім приведеного вище рядка виводу, послідують численні повідомлення типу

```
find: /root: Permission denied
```

що вказують на каталоги, закриті для перегляду звичайним користувачем, і що дуже заважають сприйняттю. Щоб подавити їх вивід, слід перенаправити

відображення повідомлення про помилки у файл `/dev/null`, тобто вказати їм "Дорогу нікуди":

```
$ $ find / -fstype ext3 -name zsh -ls 2> /dev/null
```

Опція `-delete` знищить усі файли, відібрані за вказаними критеріями. Так, командою

```
$ $ find ~ -atime +100 -delete
```

будуть автоматично стерті усі файли, до яких не було звернення за останні 100 днів (з мовчазного припущення, що раз до них три місяці не зверталися - значить, вони і взагалі не потрібні). Знищенню піддадуться файли в підкаталогах будь-якого рівня вкладеності – але підкаталоги (якщо, звичайно, останні самі не підпадають під критерії відбору), що не включають їх.

І, нарешті, опція `-exec` – саме нею обумовлена велич утиліти `find`. Як значення її можна вказати будь-яку команду з необхідними опціями - і вона буде виконана над відібраними файлами, які розглядатимуться як її аргументи.

Використовувати опцію `-delete`, як ми це тільки що зробили - не найкраще рішення, бо файли при цьому видаляться без запиту, і можна випадково видалити що-небудь потрібне. І тому досягнемо тієї ж мети таким чином:

```
$ $ find ~/ -atime +100 -exec rm - i {} ;
```

В цьому випадку на видалення кожного відібраного файлу запрошуватиметься підтвердження. Звертаю увагу на послідовність символів {} ; (з пропуском між закриваючою фігурною дужкою і зворотним слешем) у кінці рядка. Пара фігурних дужок {} символізує, що свої аргументи виконувана команда (у прикладі `-rm`) отримує від результатів відбору команди `find`, крапка з комою означає завершення рядка (без неї на натиснення Enter послідувало б вторинне запрошення командного рядка), а зворотний слеш екранує її спеціальне значення.

Окрім опції дії `-exec`, у команди `find` є ще одна, близька за сенсом, опція `-ok`. Вона також викликає деяку довільну команду, якій як аргументи передаються імена файлів, відібрані за критеріями, заданих опцією (опціями)

пошуку. Проте перед виконанням кожної операції над кожним файлом запрошується підтвердження. Наведений приклад, хоча і цілком життєвий, досить елементарний. Розглянемо складніший випадок – збирання в один каталог усіх скриншотів у форматі PNG, розкиданих по дереву домашнього каталогу :

```
$ $ find ~/ -name *.png -exec cp {} imagesdir ;
```

В результаті усі png-файли будуть вишукані і скопійовані (чи переміщені, якщо скористатися командою `mv` замість `cp`) в одне місце.

А тепер – варіант вирішення задачі, яка здавалася спочатку важкою: рекурсивне присвоювання необхідних атрибутів доступу в розгалуженому дереві каталогів – різних для регулярних файлів і каталогів.

Навіщо і чому це треба? Ну, по-перше, всі файли, скопійовані з DVD і FAT-дисків, отримали біти виконання, хоча все це були файли даних. Здавалося б, дрібниця, але що іноді дуже заважає: у деяких системах це не дозволяє, наприклад, проглянути html-файл в `Midnight Commander` простим натисненням Enter. По-друге, для деяких каталогів, навпаки, виконання не було передбачене ні для кого. В-третіх, каталоги (і файли) з DVD часто не мали атрибуту зміни – а вони потрібні були для роботи (в т.ч. і редагування). Звичайно, від усіх цих артефактів можна було б позбавитися, передбачивши належні опції монтування накопичувачів (кожного накопичувача – а їх число, повторюю, вимірювалося вже об'ємом займаного простору), та про це ніхто не подумав – що виросло, то виросло. Отже, ситуація явно вимагала виправлення, проте виконати вручну таку роботу над даними більш ніж в 200 Гбайт бачилось нездійсненним. Та так воно, власне, і було б, якщо б не опція `-exec` утиліти `find`. Ця опція дозволила змінити права доступу необхідним чином.

Отже, спочатку відбираємо всі регулярні файли і знімаємо з них біт виконання для всіх, заразом привласнюючи атрибут зміни для себе:

```
$ $ find ~/dir_data -type f -exec chmod a - x, u+w {} ;
```

Далі – пошук каталогів і зворотна процедура над підсумковою вибіркою:

```
$ $ find ~/dir_data -type d -exec chmod a+xr, u+w {} ;
```

І справа зроблена, всі права доступу стали однаковими (і тими, що нам потрібні).

Так, за допомогою команди `find` легко налагодити періодичну архівацію результатів поточної роботи. Для цього насамперед створюємо командою `tar` повний архів результатів своєї життєдіяльності:

```
tar cvf alldata.tar ~/*
```

А потім в міру своєї зіпсованості (чи, навпаки, акуратності), час від часу запускаємо команду

```
$$find ~/ -newer alldata.tar -exec tar uvf alldata.tar{};
```

Ще один практично корисний варіант використання команди `find` в мирних цілях – періодичне додавання окремо написаних фрагментів до підсумкової праці життя. Втім, щоб зробити це, необхідно спочатку ознайомитися з командами обробки файлів, до яких ми незабаром звернемося.

А доки – про обмеження можливостей такого чудового зчеплення команди `find` з опцією дії `-exec` (що поширюються і на опцію `-ok`). Воно достатнє очевидне: команда, що викликається будь-якій з цих опцій, виконується у рамках самостійного процесу, що на слабких машинах, як то кажуть, призводить до падіння продуктивності (повинен відмітити, що на машинах сучасних помітити цього практично неможливо).

У джерелах зустрічалася вказівка на ще одне обмеження зв'язки `find` з опцією `-exec` - обмеження на довжину командного рядка. Проте на практиці з таким обмеженням нам зіткнутися не доводилося (навіть у штучно створених ситуаціях, типу пошуку за шаблоном `s*` і тому подібними). І в документації ніяких відомостей з цього приводу не знайдено.

Проте, якщо яка-небудь з цих проблем виникне - вона цілком здолана. І зробити це покликана команда `xargs`. Вона визначається як будівник і виконавець командного рядка із стандартного введення. А оскільки на

стандартне введення може бути спрямоване виведення команди find – xargs сприйме результати її роботи як аргументи якої-небудь команди, яку у свою чергу, можна розглядати як аргумент її самої (за замовчуванням такою командою-аргументом є /bin/echo).

Спробуємо використовувати find для того, щоб не просто знайти необхідні файли, але і перемістити їх в правильний каталог:

```
find . -name "*.pdf" -print | grep -v "^\./pdfs/" | xargs -J X mv
```

Для того, щоб переконатися в тому, що це спрацювало, виконаємо первинну команду find:

```
find . -name "*.pdf" -print
././pdfs/50130201a.pdf
././pdfs/50130201b.pdf
././pdfs/50130201c.pdf
././pdfs/IWARLab.pdf
././pdfs/DoS_trends.pdf
././pdfs/Firewall - Guide.pdf
././pdfs/2000_ports.pdf
```

Подивимося, яким чином працює ця конструкція. Тоді як grep закінчує фільтрацію виведення команди find, ми передаємо результати його роботи команді xargs, для завершення операції. Ключ J говорить xargs взяти усі отримані рядки і підставити їх у вказану команду. Наприклад, перед викликом команди find ми не знали, скільки файлів доведеться переміщати – такий файл може бути один, або їх може бути декілька. Незалежно від того, скільки файлів буде знайдено, нам необхідно, щоб усі вони були переміщені в каталог pdfs. Це зробити нам допоможе маленька магія ключа J. Для того, щоб ключ J спрацював правильно, ми визначили символ "X" і помістили його по обидві сторони команди mv.

Пам'ятайте, що імена файлів в системі не обов'язково мають розширення. Тому вам може знадобитися задати складніший зразок для пошуку. Скажімо, потрібно знайти всі файли, що містять у своєму імені підрядок "bsd". Для цього слід скористатися наступною командою:

```
find . -name "*bsd*" -print
./../.kde/share/icons/favicons/www.freebsd.org.png
```

```
./../.kde/share/icons/favicons/www.freebsddiary.org.png
./../.kde/share/wallpapers/bsdbg1280x1024.jpg
././mnwclient-1.11/contrib/freebsd
```

Команди which, apropos, locate, ls

Розглянемо декілька завдань.

Завдання 1: пошук бінарного файлу, початкового коду, і довідкової сторінки команди.

Команда whereis знаходить де розташований початковий код, бінарний файл і довідкова сторінка певних файлів. Ім'я файлу передається без шляху і розширення. Після цього команда whereis намагається виявити шукану програму в списку стандартних місць в Linux. Наприклад, знайдемо, де знаходиться команда ls.

```
$ $ whereis ls
```

Результат:

```
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

Завдання 2: перегляд короткого опису команди.

У кожній довідковій сторінці знаходиться короткий опис відповідної команди. Команда whatis шукає серед назв довідкових сторінок, і виводить описи для усіх імен, що співпали.

```
$ $ whatis ls
```

Результат:

```
ls (1)               - list directory contents
```

Завдання 3: пошук команди за допомогою команди which.

Команда which повертає шлях до файлів, які були б виконані в поточному оточенні, маючи як аргументи команди у форматі POSIX. Вона робить це, використовуючи пошук виконуваних файлів в PATH, співпадаючих з іменами аргументів.

```
$ $ which ls
$ $ which -a date
```

Завдання 4: пошук довідкових сторінок і описів команд, з використанням команди apropos.

У кожній довідковій сторінці знаходиться короткий опис відповідної команди. Команда apropos шукає в описах відповідні ключові слова. Це дуже корисно, якщо потрібно знайти команду, відповідну завданню. Наприклад, знайдемо команду, яка видаляє користувача:

```
$ $ apropos 'delete a user'
```

Результат:

```
userdel (8)          - Delete a user account and related files
```

Інші приклади:

```
$ $ apropos 'delete'
$ $ apropos 'icmp'
```

Завдання 5: виводимо список файлів в базі даних, використовуючи команду locate.

Ця команда використовується, щоб дізнатися, де знаходиться файл. Якщо ви забули місцезнаходження файлу, наприклад, httpd, використовуйте команду locate:

```
$ $ locate httpd.conf
```

Результат:

```
//etc/apache2/httpd.conf
//etc/lighttpd/lighttpd.conf
//etc/lighttpd/lighttpd.conf.BAK
//home/vivek/etc/apache2/httpd.conf
//home/vivek/etc/lighttpd/lighttpd.conf
//home/vivek/etc/lighttpd/lighttpd.conf.BAK
//usr/share/doc/lighttpd/examples/lighttpd.conf.gz
//var/lib/dpkg/info/lighttpd.conffiles
```

Найбільш універсальним засобом отримання практично вичерпної інформації про файли є команда ls. Як ми вже знаємо, загальна форма її запуску:

```
$ $ ls [options] names
```

де аргументом names можуть виступати імена файлів або каталогів в будь-якій кількості. Команда ця має численні опції, основні з яких ми і згадаємо.

Почнемо з того, що команда ls, дана без всяких опцій, за замовчуванням виводить тільки імена файлів, причому опускаючи т.з. dot-файли, імена яких починаються з точки (це деякі аналоги hidden-файлів в MS DOS). Крім того, якщо як аргумент вказано ім'я каталогу (чи аргумент не вказаний взагалі, що має на увазі поточний каталог), із списку імен його файлів не виводяться поточний (.) і батьківський (..) каталог.

Для виведення усіх без виключення імен файлів (у тому числі і прихованих) призначена опція -a. Сенс опції -A близький – вона виводить список імен всіх файлів, за винятком імені поточного (.) і батьківського (..) каталогу. Окрім імені, будь-який файл ідентифікується своїм номером inode. Для його виводу використовується опція -i:

```
$ $ ls -i
12144 content.html       12149 gentoo02.html
```

і так далі. Як і багато інших, команда ls має здатність рекурсивної обробки аргументів, для чого призначена опція -R, що виводить список імен файлів не лише поточного каталогу, але і усіх вкладених підкаталогів:

```
$ $ ls - R
unixforall:
about/  apps/     diffimages/  distro/  signature.html  sys/
anons/  content/  difftext/    gentoo/  statistics/     u4articles/
unixforall/about:
about_lol.html  about_lol.txt  index.html
unixforall/anons:
anons_dc.html
```

У виведенні команди ls за замовчуванням імена файлів різних типів даються абсолютно однаково. Для їх візуальної відмінності використовується опція -F, що завершує імена каталогів символом слеша, здійснимих файлів - символом зірочки, символічних посилань - "собакою"; імена регулярних файлів, що не мають атрибуту виконання, ніякого символу не включають:

```
$ $ ls - F
```

```
dir1/   dir2/   dir3@   file1   file2*  file3@
```

Інший засіб для візуальної відмінності типів файлів – колоризація, для чого застосовується опція `-G`. Кольори шрифту, відтворюючого імена, за замовчуванням – синій для каталогів, ліловий (magenta) для символічних посилань, червоний – виконуваних файлів, і так далі. Для файлів пристроїв, виконуваних файлів з атрибутом "суідности", каталогів, що мають атрибут `sticky`, додатково колоризується і фон, на якому виводиться шрифт, відтворюючий їх імена. Подробиці можна подивитися в секції ENVIRONMENT `man`-сторінки для команди `ls`. Втім, колоризація працює не при всіх налаштуваннях терміналів (і не в усіх командних оболонках).

За замовчуванням команда `ls` виводить список файлів в порядку ASCII-коду першого символу імені. Проте є можливість його сортування в зворотному порядку (`-r`), в порядку часу модифікації (`-t`) або часу доступу (`-tu`). Крім того, опція `-f` відміняє будь-яке сортування списку взагалі.

Інформацію про об'єм файлів можна отримати, використовуючи опцію `-s`, що виводить для імені кожного файлу його розмір в блоках, а також сумарний об'єм усіх виведених файлів:

```
$ $ ls - s ./book
total 822
656 book.html
4 content1.html
86 var_part2.html
24 command.html
38 part2.html
6 command.txt
8 shell_tmp.html
```

Додавання до опції `-s` ще і опції `-k` (тобто `ls -sk`) виведе всю ту ж інформацію в кілобайтах. Очевидно, що якщо розмір блоку файлової системи, як це часом буває, складає 1024 байти, виведення `ls` з цими опціями буде однаковим.

Як можна бачити з усіх наведених вище прикладів, списки файлів по команді `ls` виводяться в багатоколоночному виді (чому відповідає опція `-C`,

проте вказувати її немає необхідності – багатоколоночний вигляд вибраний для короткого формату за замовчуванням). Але можна задати і таке представлення списку за допомогою опції `-1`:

```
$ $ ls -1
dir1
dir2
dir3
file1
file2
file3
```

Досі йшлося про короткий формат команди `ls`. Проте більш інформативним є т.з. довгий її формат, вивід в якому досягається опцією `-l` і автоматично викликає за собою одноколоночне представлення списку:

```
$ $ ls - l
total 8
drwxr - xr - x  2 alv  alv  512  8 травня 18:04 dir1
drwxr - xr - x  3 alv  alv  512  8 травня 17:43 dir2
lrwxr - xr - x  1 alv  alv    4  9 травня 07:59 dir3 -> dir2
-- rw - r--r--  1 alv  alv   14  8 травня 10:39 file1
-- rwxr - xr - x  1 alv  alv   30  9 травня 08:02 file2
lrwxr - xr - x  1 alv  alv    2  8 травня 10:57 file3 -> f1
```

Можна бачити, що за замовчуванням в довгому форматі виводяться: відомості про тип файлу (- регулярний файл, `d` - каталог, `l` – символічне посилання, `c` – файл символьного пристрою, `b` – файл блокового пристрою) і атрибути доступу для різних атрибутів приналежності; кількість жорстких посилань на цей ідентифікатор `inode`; `ім'я користувача` – власника файлу, і групи користувачів, якій файл належить; розмір файлу в блоках; час модифікації файлу з точністю до місяця, дня, години і хвилини (у форматі, прийнятому в цій `locale`); ім'я файлу і (для символічних посилань) ім'я файла-джерела.

Проте це ще не усе. Додавши до команди `ls -l` ще і опцію `-i`, можна додатково отримати ідентифікатор `inode` кожного файлу, опція `-n` замінить ім'я власника і групу на їх чисельні ідентифікатори (UID і GUID, відповідно), а опція `-T` виведе в полі часу модифікації ще роки, і секунди:

```
$ $ ls - linT
total 8
694402 drwxr - xr - x  2 1000  1000  512  8 травня 18:04:56 2002 dir1
694404 drwxr - xr - x  3 1000  1000  512  8 травня 17:43:31 2002 dir2
673058 lrwxr - xr - x  1 1000  1000    4  9 травня 07:59:08 2002 dir3 ->
dir2
673099 - rw - r--r--  1 1000  1000   14  8 травня 10:39:38 2002 file1
673059 - rwxr - xr - x  1 1000  1000   30  9 травня 08:02:23 2002 file2
673057 lrwxr - xr - x  1 1000  1000    2  8 травня 10:57:07 2002 file3 -
> f1
```

Зрозуміло, ніхто не забороняє використовувати в довгому форматі і опції візуалізації (`-F` і `-G`), і опції сортування (`-r`, `t`, `tu`), і будь-які інші, за винятком опції `-C` – вказівка її веде до примусового виведення списку в багатоколоночній формі, що природним чином пригнічує довгий формат представлення. Пошук інформації про `rpm`-пакети з використанням механізму конвеєра і команди `grep`. Дуже часто з'являється необхідність впізнати номер релізу великого числа пакетів в системі Linux. Скажімо, наприклад, Ви не впевнені яку версію GNOME використовується. За допомогою команд `rpm`, `grep`, механізму конвеєра, і пари ключів можна дуже швидко це з'ясувати. Ці дві команди (`rpm` і `grep`) об'єднуються для спільного використання. Цей метод називається конвеєром і позначається (|).

Синтаксис команд виглядає таким чином:

```
rpm -qa | grep PACKAGE_NAME
```

При розгляді складових частин цієї команди, ми бачимо, що `grep` використовує дані отримані при запиті (ключ `q`) до усіх (ключ `a`) входжень в базу `rpm` пакету PACKAGE_NAME. Хоча ця конструкція не відображує усі можливості команди, вона дає початкові поняття про їх використання.

Допустимо, потрібно подивитися, які пакети встановлені для GNOME. Якщо використовувати наступну конструкцію:

```
rpm -qa | grep gnome
```

ймовірно буде отримано щось наступного виду:

```
[jwallen@giles jwallen]$ rpm -qa | grep gnome
sawfish - gnome - 0.28.1-0_helix_2
```

```
gnome - pin - conduits - 1.2.0-0_helix_1
gnome - games - devel - 1.2.0-0_helix_1
gnome - audio - 1.0.0-7
gnome - audio - extra - 1.0.0-7
gnome - linuxconf - 0.23-1
.....
```

[частина тексту пропущена]

Як видно з прикладу, на екран виведені усі пакети, що містять слово `gnome`. Це ілюструє очевидне обмеження – не всі пакети, які потрібні пакету GNOME містять слово `gnome`. Наприклад, GNOME залежить від великого числа бібліотек, таких як `gtk`. Для знаходження всіх встановлених компонентів `gtk` введіть команду:

`rpm - qa| grep gtk`:

```
[jwallen@giles jwallen]$ rpm - qa | grep gtk
pygtk - libglade - 0.6.4-1
rep - gtk - libglade - 0.11-0_helix_2
gtk - 1.0.3-1
gtk+10-1.0.6-6
pygtk - 0.6.3-1
gtk+-1.2.7-1_helix_2
gtk - engines - 0.10-1_helix_2
gtk+-devel - 1.2.7-1_helix_1
```

Очевидно, що запропонована система – це лише невелика допомога для знаходження всіх пакетів, які мають залежності від інших пакетів. Проте, конвеєр `rpm` і `grep` допоможе знайти номери релізів та імена необхідних пакетів.

4.2.2Завдання до другої частини лабораторної роботи

1. Команда find

За допомогою команди `find` і її критеріїв виконати задані дії:

– вибрати в каталозі усі зкомпресовані архіви, створені після файлу з ім'ям `filename` (можна задати будь-який інший файл);

– відібрати файли, створені менше двох діб тому, або ж - пізніше, ніж файл `filename`;

– вивести список відібраних файлів на екран;

– відібрати каталоги, закриті для перегляду звичайним користувачам;

– автоматично стерти всі файли, до яких не було звернень за останні 100 днів (з припущення за замовчуванням, що раз до них три місяці не зверталися – значить, вони і взагалі не потрібні);

– стерти декілька вибраних файлів, щоб видалення кожного відібраного файлу запрошувалося підтвердженням;

– відшукати і скопіювати (чи – перемістити, якщо скористатися командою `mv` замість `cp`) файли заданого типу в одне місце;

– використати `find` для того, щоб не просто знайти необхідні файли, але і перемістити їх в правильний каталог;

– знайти усі файли, що містять у своєму імені заданий підрядок.

2. Команда `which`

– знайти, де знаходиться задана команда (на вибір).

3. Команда `whatis`

– перегляд короткого опису заданої команди.

4. Команда `which`

– пошук заданої команди за допомогою команди `which`.

5. Команда `apropos`

– пошук довідкових сторінок і описів команд, з використанням команди `apropos`.

6. Команда `locate`

– вивести список файлів в базі даних, використовуючи команду `locate`.

7. Команда `ls`

– вивести список імен файлів не лише поточного каталогу, але і всіх вкладених підкаталогів;

– вивести для імені кожного файлу його розмір в блоках, а також сумарний об'єм всіх виведених файлів;

– отримати ідентифікатор файлу;

– замінити ім'я власника файлу і групу на їх чисельні ідентифікатори;

– вивести в поле часу модифікації роки і секунди.

4.2.3Контрольні запитання до другої частини лабораторної роботи

1. Що таке опція пошуку в команді `find` і що може нею виступати?

2. Яка особливість GNU-реализації команди `find`?

3. Для чого ще можна використовувати команду `find` крім знаходження файлів?

4. Що знаходить команда `whereis`?

5. Для чого необхідна команда `whatis`?

6. Яка команда повертає шлях до файлів, які були б виконані в поточному оточенні, маючи як аргументи команди у форматі POSIX?

7. Яку команду потрібно використати, якщо потрібно згадати місцезнаходження файлу?

4.3 Мережеві утиліти Linux

4.3.1Теоретичні відомості

1. Мережа в LINUX

Мережі стали невід'ємною складовою сучасних обчислювальних систем. Технології взаємодії між інформаційними системами розвиваються з 1970-х років, ненабагато випередивши розвиток LINUX. Операційна система LINUX майже з самого народження інтегрувала в себе технології організації локальних мереж, на її основі потім була побудована мережа Internet, що поширилася нині по всьому світу.

2. Введення в мережі

Організація взаємодії між пристроями і програмами в мережі є складним завданням. Мережа об'єднує різне устаткування, різні операційні системи і програми – їх успішна взаємодія була б неможлива без прийняття загальноприйнятих правил, стандартів.

У області комп'ютерних мереж існує безліч міжнародних і промислових стандартів, серед яких слід особливо виділити міжнародний стандарт OSI і набір стандартів IETF (Internet Engineering Task Force).

3. Мережевий інтерфейс в LINUX

Осново. мережевої підсистеми LINUX є мережевий інтерфейс. Мережевий інтерфейс – це абстракція, що зв'язує канальний і мережевий (протокол TCP/IP) рівні мережі в LINUX.

Мережевий інтерфейс створюється у момент завантаження драйвера мережевого пристрою (наприклад, мережевої карти) або створення логічного з'єднання (наприклад, у разі PPP). Кожен мережевий інтерфейс в системі має унікальне ім'я, що складається з типу пристрою і номера (0 або більше для однотипних пристроїв). Під типами пристроїв в різних LINUX-системах може розумітися вид протоколу канального рівня (Ethenet -eth) або назва драйвера пристрою (Realtek -rl).

Інтерфейс має набір параметрів, велика частина яких відноситься до мережевого рівня (IP-адрес, маска мережі і тому подібне). Важливим параметром мережевого інтерфейсу є апаратна адреса (у разі Ethernet апаратна адреса називається MAC-адреса і складається з шести байтів, які прийнято записувати в шістнадцятиричній системі числення і розділяти двокрапками).

Основною утилітою для перегляду і управління параметрами мережевих інтерфейсів в LINUX служить `ifconfig`. Нині у ряді систем разом з нею використовується сучасніша утиліта `ip`, з синтаксисом параметрів командного рядка, що відрізняється. Далі в прикладах, що наводяться, ми використовуватимемо в першу чергу традиційну і на сьогодні поширенішу утиліту `ipconfig`.

Утиліта ifconfig, викликана в командному рядку без параметрів, виводить відомості про усі налагоджені в даний момент мережеві інтерфейси. Щоб проглянути інформацію про конкретний інтерфейс, необхідно вказати його ім'я як параметр ifconfig.

Розглянемо декілька прикладів.

Приклад 1. Перегляд параметрів мережевих інтерфейсів (ifconfig)

```
desktop ~ # ifconfig eth0
eth0      Link encap: Ethernet  HWaddr 00:0 D :60:8 D :42: AA
                    inet addr :192.168.1.5  Bcast :192.168.1.255  Mask
:255.255.255.0
                    UP BROADCAST RUNNING MULTICAST  MTU :1500  Metric :1
                    RX packets :6160 errors :0 dropped :0 overruns :0 frame
:0
                    TX packets :5327 errors :0 dropped :0 overruns :0 carrier
:0
                    collisions :1006 txqueuelen :1000
                    RX bytes :3500059 (3.3 Mb)  TX bytes :2901625 (2.7 Mb)
                    Base address :0 x8000 Memory : c0220000 - c0240000
desktop ~ # ifconfig lo
lo        Link encap: Local Loopback
                    inet addr :127.0.0.1  Mask :255.0.0.0
                    UP LOOPBACK RUNNING  MTU :16436  Metric :1
                    RX packets :188 errors :0 dropped :0 overruns :0 frame :0
                    TX packets :188 errors :0 dropped :0 overruns :0 carrier
:0
                    collisions :0 txqueuelen :0
                    RX bytes :14636 (14.2 Kb)  TX bytes :14636 (14.2 Kb)
```

Звичайне виконання ifconfig вимагає прав суперкористувача або дозволяється користувачам, що входять до спеціальної "адміністративної" групи (наприклад, netadmin). Налаштування мережевих параметрів, пов'язаних з інтерфейсом, виконується за допомогою тієї ж утиліти ifconfig, про що буде сказано далі.

4. Конфігурація IP -сетей

У IP-мережах кожному мережевому інтерфейсу привласнюється деяка єдина на усю глобальну мережу адреса, яка не залежить від середовища передачі даних і завжди має один і той же формат.

Формат адреси залежить від версії протоколу. У найбільш поширеній зараз четвертій версії адреса складається з чотирьох байт, записуваних традиційно в десятковій системі числення і таких, що розділяються точкою. Тоді як в шостій версії протоколу IP адреса складається вже з 16 байт і зазвичай записується в шістнадцятковій системі числення.

IP-адреса мережевого інтерфейсу `eth0` з наведеного вище прикладу - 192.168.1.5. Другий мережевий інтерфейс з прикладу - `lo` - так звана заглушка (loopback), яка використовується для організації мережевих взаємодій комп'ютера з самим собою: будь-який посланий в заглушку пакет негайно обробляється як прийнятий звідти. Заглушці зазвичай призначається адреса 127.0.0.1.

Для того, щоб призначити інтерфейсу IP-адресу, досить виконати `ifconfig`, вказавши після імені інтерфейсу IP-адресу:

```
desktop ~ # ifconfig eth0 192.168.1.1
```

5. Маршрутизація

Як було сказано вище, IP-адреса включає дві частини: адресу підмережі і адресу конкретного вузла у рамках цієї підмережі. Маска підмережі показує, скільки біт в IP-адресі містить адреса підмережі (інші - це адреса вузла). Таким чином, IP-адреса вузла призначення і маска підмережі завжди дозволяє визначити, чи відноситься вузол призначення до тієї ж підмережі. В цьому випадку пакети до них доставлятимуться безпосередньо через канальний рівень.

Складніше питання постає, якщо IP-адреса вузла-адресата не входить в локальну мережу вузла-відправника. Адже і в цьому випадку пакет необхідно відіслати якомусь абонентові локальної мережі, з тим, щоб той перенаправив його далі. Цей абонент, маршрутизатор, підключений до декількох мереж, і йому ставиться в обов'язок пересилати пакети між ними за певними

правилами. У найпростішому випадку таких мереж дві: "внутрішня", до якої підключені комп'ютери, і "зовнішня", сполучаюча маршрутизатор з усією глобальною мережею. Таблицю, що управляє маршрутизацією пакетів, можна проглянути за допомогою утиліти `netstat -r` або `route` (обидві утиліти мають ключ `-n`, що примушує їх використовувати у видачі IP-адреси, а не імена комп'ютерів):

```
desktop ~ # route - n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0
eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0
lo
0.0.0.0         0.0.0.0         192.168.1.1     0.0.0.0         UG    0
0        0 eth0
```

Комп'ютер або апаратний пристрій, що здійснює маршрутизацію між локальною мережею і Internet зазвичай називається шлюзом. Для зміни таблиці маршрутизації використовується та ж утиліта `route`. Утиліта `ip` об'єднує можливості як по налаштуванню мережевих інтерфейсів, так і таблиці маршрутизації.

6. Службовий протокол ICMP

Є такі протоколи рівня IP, дія яких цим рівнем і обмежується. Наприклад, службовий протокол ICMP (Internet Control Message Protocol), призначений для передачі службових повідомлень.

Прикладом застосування ICMP є утиліта `ping`, яка дозволяє перевірити працездатність вузлів в мережі. Інше застосування ICMP - повідомляти відправника, чому його пакет неможливо доставити адресатові, або передавати інформацію про зміну маршруту, про можливість фрагментації і тому подібне.

Ще одне завдання, яке можна вирішити використанням протоколу ICMP, являється визначення маршруту доставки пакету. У LINUX існує ряд

альтернативних команд для визначення приблизного маршруту проходження пакету по мережі: `traceroute`, `tracepath` і т.п.

Приклад виконання команди `traceroute` наступний:

```
desktop ~ # traceroute ya.ru
traceroute to ya.ru (213.180.204.8), 64 hops max, 40 byte packets
  1  195.91.230.65 (195.91.230.65)  0.890 ms  1.907 ms  0.809 ms
  2  cs7206.rinet.ru (195.54.192.28)  0.895 ms  0.769 ms  0.605 ms
  3  ix2 - m9.yandex.net (193.232.244.93)  1.855 ms  1.519 ms  2.95 ms
  4  c3 - vlan4.yandex.net (213.180.210.146)  3.412 ms  2.698 ms  2.654 ms
  5  ya.ru (213.180.204.8)  2.336 ms  2.612 ms  3.482 ms
```

Утиліта `traceroute` показує список абонентів, через яких проходить пакет по дорозі до адресата, і витрачений на це час. Проте список цей приблизний. Строго кажучи, невідомо, яким маршрутом йшла чергова група пакетів, тому що відколи була відправлена попередня група, який-небудь з проміжних маршрутизаторів міг змінити рішення і відправити нові пакети іншим шляхом.

7. Інформація про з'єднання

Транспортних протоколів в TCP/IP два - це TCP (Transmission Control Protocol, протокол управління з'єднанням) і UDP (User Datagram Protocol). UDP забезпечує вищу швидкість обміну даними за рахунок простоти реалізації. Призначені для користувача дані поміщаються в єдиний транспортний пакет-датаграму, в яку вкладаються звичайні для транспортного рівня дані: адреси і порти відправника і одержувача, після чого пакет йде в мережу шукати адресата. Перевіряти, чи був адресат здатний цей пакет прийняти, чи дійшов пакет до нього і чи не зіпсувався по дорозі, надається наступному - прикладному - рівню.

Інша справа – TCP. Цей протокол дуже піклується про те, щоб передавані дані дійшли до адресата у повній цілості. Для цього робляться наступні дії:

- встановлення з`єднання;
- обробка підтвердження коректної доставки;
- відстежування стану абонентів.

Для перегляду всіх існуючих зараз мережевих з'єднань можна скористатися командою `netstat`. Приклад виконання команди `netstat`:

```
desktop ~ # netstat - an
Active Internet connections (servers and established)
Proto Recv - Q Send - Q Local Address          Foreign Address        State
tcp        0      0 0.0.0.0:32769           0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:32770           0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*              LISTEN
tcp        0      0 192.168.11.5:34949      83.149.196.70:5223
ESTABLISHED
tcp        0      0 192.168.11.5:39833      213.248.55.180:5223
ESTABLISHED
tcp        0      0 192.168.11.5:59577      192.168.11.1:22        TIME_WAIT
udp        0      0 0.0.0.0:32768           0.0.0.0:*
udp        0      0 0.0.0.0:32769           0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
```

Згідно з стандартами Internet для більшості протоколів прикладного рівня існують стандартні порти, на яких відповідні застосування повинні приймати з'єднання. Наприклад, веб-сервер, що виконує обробку з'єднань по протоколу HTTP, повинен працювати на порту з номером 80.

У LINUX існує прозорий механізм іменування протоколів прикладного рівня. У файлі `/etc/services` можна побачити список відповідності імен протоколів номерам портів. Цей файл використовується базовою системною бібліотекою мережевої взаємодії, так що в усіх утилітах замість номера порту можна вказувати ім'я відповідного протоколу.

Для того, щоб кожного разу при завантаженні не задавати параметри мережевим інтерфейсам, існує можливість задати налаштування мережі в конфігураційних файлах операційної системи. В цьому випадку при старті системи буде проведена процедура налаштування інтерфейсів, заповнена таблиця маршрутизації і тому подібне. Подібні конфігураційні файли, що зберігають налаштування мережі між сеансами роботи, а також засоби

управління ними не стандартизовані і частенько розрізняються в різних версіях LINUX.

8. Служби Internet. Служба доменних імен

У попередніх прикладах використовувався ключ `-n` багатьох мережевих утиліт, щоб уникнути плутанини між IP-адресами і доменними іменами комп'ютерів. З іншого боку, доменні імена – декілька слів (часто осмислених), - запам'ятовувати набагато зручніше, ніж адреси (чотири числа).

Колись імена усіх комп'ютерів в мережі, що відповідають IP-адресам, зберігалися у файлі `/etc/hosts`. Трудність була не лише в тому, що вміст `hosts` швидко мінявся, але і в тому, що за відповідність імен адресам в різних мережах відповідали різні люди і різні організації. З'явилася необхідність структурувати глобальну мережу не лише топологічно (за допомогою IP і мережевих масок), але і адміністративно, з вказівкою, за які групи адрес хто відповідає.

Найпростіше було структурувати самі імена комп'ютерів. Вся мережа була поділена на домени - зони відповідальності окремих держав ("`us`", "`uk`", "`ru`", "`it`" і тому подібне) або незалежні зони відповідальності ("`com`", "`org`", "`net`", "`edu`" і тому подібне). Для кожного з таких доменів першого рівня має бути присутнім підрозділ, що видає усім абонентам імена, що закінчуються на "`.домен`" Підрозділ зобов'язаний організувати і підтримувати службу, замінюючу файл `hosts`: будь-який охочий має право дізнатися, яка IP-адреса відповідає імені комп'ютера в цьому домені або якому доменному імені відповідає певна IP-адреса. Така служба називається DNS (Domain Name Service, служба доменних імен). Вона має ієрархічну структуру. Якщо за якусь групу абонентів домена відповідають не хазяї домена, а хтось інший, йому виділяється піддомен (чи домен другого рівня), і він сам розпоряджається іменами виду "`ім'я_компьютера.піддомен.домен`". Таким чином, виходить щось на

зразок розподіленої мережевої бази даних, що зберігає короткі записи про відповідність доменних імен IP-адресам.

У найпростішому випадку для того, щоб сказати системі, який сервер доменних імен використовувати, необхідно змінити файл `/etc/resolv.conf`. У складніших системах можна встановити і настроїти власний сервер доменних імен.

Для перевірки роботи системи DNS використовуються утиліти `dig` і `host`.

`Допоміжні утиліти.` В першу чергу допоміжні утиліти спрямовані на використання системними інженерами і просунутими користувачами, але можуть бути корисні і усім іншим користувачам. В таблиці 4.3 утиліти, які в тих або інших реінкарнаціях є стандартними для "великих" операційних систем.

Таблиця 4.3 – Список утиліт

Утиліта	Призначення
`Ifconfig`	Утиліта, що надає інформацію про усі мережеві інтерфейси, присутні в системі.
`NSLookup`	Утиліта для отримання інформації про доменні імена і IP-адреси хостів від служби DNS.
`NTPSync`	Утиліта-синхронізатор локального годинника з сервером мережевого часу.
`Ping`	Утиліта для діагностики зв'язку з видаленими хостами.
`Traceroute`	Утиліта для пошуку і діагностики мережевих маршрутів до видалених

	хостів.
Whois	Утиліта, що запрошує реєстраційну інформацію про домени і персоналії на спеціальних whois-серверах.

Утиліта Ping. Ping будується на основі функції відлуння протоколу ICMP, описаною в документі RFC 792, який рекомендовано застосовувати на всіх хостах. Поза сумнівом, мережеві адміністратори можуть забороняти проходження ping -повідомлень за вимогами мережевої безпеки, що, як правило не є ідеальним рішенням. У будь-якому випадку, Ping надає більше інформації про стан з'єднань, чим багато інших мережевих програм.

Найбільш поширено дві реалізації ping утиліт. Як Ви можете легко здогадатися, це реалізації Linux-подібних систем і Microsoft. Linux-реалізація є повнішою і інформативною, але і складнішою.

Використовуючи Ping, можна легко отримати наступну інформацію:

- переупорядковування, дублювання і зникнення пакетів по шляху доставки;
- порушення цілісності пакетів;
- визначення часу поширення пакетів і зміна цього параметра в часі;
- контроль стійкості з'єднання;
- поява в процесі роботи невідомих пакетів.

За рамками цієї програми залишилися також наступні опції, характерні для Linux-реалізацій: flood ping і preload packets. Ці функції дуже цікаві при випробуваннях мережевих рішень на стійкість, оскільки дозволяють тестувати мережу в умовах граничного навантаження, але для них потрібні швидкі мережеві інтерфейси.

Відсутні і можливості зміни параметра Quality of Service, заборони фрагментації пакетів за шляхом призначення, відправки пакетів з прив'язкою до передвстановленого маршруту і/або в обхід таблиць маршрутизації,

оскільки ці опції застосовуються украй рідко, а остання має сенс тільки при діагностиці помилок маршрутизації і тільки для хостів на відстані одного хопа.

`Connect()` сканер використовує базовий скануючий алгоритм. Функція `connect()` дозволяє хосту з'єднатися з будь-яким портом сервера. Якщо порт, вказаний як параметр функції, прослуховується сервером (тобто порт відкритий для з'єднання), то результатом виконання функції буде встановлення з'єднання з сервером за вказаним портом. Інакше, якщо з'єднання не встановлене, то порт з вказаним номером є закритим.

Великим недоліком цього методу є можливість виявлення і фільтрації такого роду сканування, причому зробити це досить легко. Log-файл сканованого сервера вкаже службам, що відповідають за зовнішні підключення, на наявність численних спроб підключення з однієї адреси і помилок встановлення з'єднання (оскільки хост що досліджує після з'єднання з сервером відразу обриває його), а ті, у свою чергу, негайно заблокують доступ до сервера для хоста з цією адресою.

Формат команди:

```
ping [- t][- a][- n][- l][- f][- i TTL][- v TOS] [-
r][][ім'я машини][[- j списокУзлов]|[- k
списокУзлов]][- w]
```

В таблиці 4.4 зазначені параметри утиліти `ping`.

Таблиця 4.4 – Параметри утиліти `ping`

Ключі	Функції
-t	Відправка пакетів на вказаний вузол до команди переривання
- a	Визначення адрес за іменами вузлів
- n	Число запитів, що відправляються
- l	Розмір буфера відправки

- f	Установка прапора, що забороняє фрагментацію пакету
- i TTL	Завдання часу життю пакету (поле "Time To Live")
- v TOS	Завдання типу служби (поле "Type Of Service")
- r	Запис маршруту для вказаного числа переходів
- s	Штамп часу для вказаного числа переходів
- j список вузлів	Вільний вибір маршруту за списком вузлів
- k список вузлів	Жорсткий вибір маршруту за списком вузлів
- w інтервал	Інтервал очікування кожної відповіді в мілісекундах

На практиці більшість опцій в форматі команди можно пропустити, тоді в командному рядку може бути: ping і'мя вузла.

```
# # ping newslink.org

 Обмін пакетами з 207.227.119.10 по 32 байт
 Відповідь від 207.227.119.10: число байт=32 час=196мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=198мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=193мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=195мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=199мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=196мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=192мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=197мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=197мс TTL=237
 Час очікування запиту витік.
 Відповідь від 207.227.119.10: число байт=32 час=202мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=192мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=191мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=193мс TTL=237
Відповідь від 207.227.119.10: число байт=32 час=200мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=196мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=196мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=199мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=196мс TTL=237
 Відповідь від 207.227.119.10: число байт=32 час=193мс TTL=237

    Статистика Ping для 207.227.119.10: Пакетів: послане = 20, отримане =
19, втрачено = 1 (5% втрат)
    Приблизний час передачі і прийому : найменше = 191 мс, найбільше =  202
мс, середнє =  186 мс
```

Зверніть увагу: максимальне значення TTL за дорівнює 255 вузлам. Отже, щоб визначити кількість вузлів, через які пройшов пакет, потрібно від 255 відняти набуте значення TTL.

`Утиліта traceroute.` Програма `traceroute` дозволяє виявляти послідовність шлюзів, через які проходить IP -пакет на шляху до пункту свого призначення. У цієї команди є дуже багато опцій, більшість з яких застосовуються системними адміністраторами украй рідко.

Формат команди :

```
traceoute ім'я_машини
```

Як завжди, *ім'я_машини* може бути задане в символічній або числовій формі. Вихідна інформація є списком машин, починаючи з першого шлюзу і кінчаючи пунктом призначення. Крім того, показаний повний час проходження кожного шлюзу.

Приклад виконання команди наступний:

```
      st1@pds:~ > traceroute www.newslink.org
      traceroute to www.newslink.org (207.227.119.10), 30 hops max, 40 byte
packets
      18       lgw.ccs.sut.ru #001 ms 1 ms 1 ms
      18       ing - e0.nw.ru #005 ms 2 ms 2 ms
      18       StPetersburg - LE - 4.Relcom.EU.net #004 ms 2 ms 6 ms
      18       cwrussia - relcom.SPB.cwrussia.ru #0029 ms 33 ms 19 ms
      18       bar2 - serial6 - 1-0-0.NewYorknyr.cw.net#00166 ms 168 ms 170
ms
      18       acr2 - loopback.NewYorknyr.cw.net#00166 ms 163 ms 167 ms
      18       p4 - 2.nycmny1 - ba1.bbnplanet.net #00177 ms 175 ms 172 ms
      18       p7 - 0.nycmny1 - br1.bbnplanet.net #00174 ms 176 ms 170 ms
      18       p4 - 0.nycmny1 - br2.bbnplanet.net #00170 ms 175 ms 171 ms
      18       so - 4-0-0.chcgil2 - br1.bbnplanet.net #00184 ms 184 ms 183 ms
      18       p6 - 0.chcgil1 - br1.bbnplanet.net #00181 ms 182 ms 185 ms
      18       p4 - 0.chcgil1 - br2.bbnplanet.net #00181 ms 189 ms 184 ms
      18       p2 - 0.nchicago2 - br1.bbnplanet.net #00184 ms 183 ms 182 ms
      18       p1 - 0.nchicago2 - br2.bbnplanet.net #00187 ms 188 ms 194 ms
      18       p8 - 0-0.nchicago2 - core0.bbnplanet.net #00482 ms 343 ms 331
ms
      18       chi2 - eth.cyberlynk.net #00190 ms 183 ms 193 ms
      18       core0 - fe0.rac.cyberlynk.net #00222 ms 217 ms 239 ms
      18       www.newslink.org #00254 ms 218 ms 229 ms
```

Команда `traceroute` працює шляхом установки поля часу життя (числа переходів) вихідного пакету так, щоб цей час збігав до досягнення пакетом пункту призначення. Коли час життя вийде, поточний шлюз

відправить повідомлення про помилку на машину-джерело. Кожен приріст поля часу життя дозволяє пакету пройти на один шлюз далі.

Команда `traceroute` посилає для кожного значення поля часу життя три пакети. Якщо проміжний шлюз розподіляє трафік за декількома маршрутами, то ці пакети можуть повертатися різними машинами. В цьому випадку на друк виводяться вони усі. Деякі системи не посилають повідомлень про пакети, час життя яких закінчився, а деякі посилають повідомлення, які поступають назад на машину-джерело тільки після того, як закінчився час їх очікування командою traceroute. Ці шлюзи позначаються рядом зірочок. Навіть якщо конкретний шлюз визначити не можна, `traceroute` найчастіше зможе побачити вузли маршруту, що йдуть за ним.

`Утиліта whois`. Сервіс `whois` дозволяє отримати інформацію про доменне ім'я. Для різних доменних зон видається різний набір інформації. У загальному випадку, це відомості про персону або організацію, на яку зареєстрований домен, дата реєстрації домена, дата закінчення терміну реєстрації. У Мережі можна знайти дуже багато сайтів, на яких, ввівши у форму доменне ім'я і натиснувши кнопку Ок, за 5 секунд можна отримати відомості про домен.

Проте, довіряти можна тільки відомостям, отриманим від офіційних координаторів зон, в яких знаходяться доменні імена, що перевіряються.

Опис відповіді `whois` для доменів наступний:

domain - власне ім'я (назва) домена, зазвичай співпадає з тим ім'ям, яке було введено. Назва домена повинна складатися більш ніж з одного символу, починатися і закінчуватися буквою латинського алфавіту або цифрою. Проміжними символами можуть бути букви латинського алфавіту, цифри або дефіс.

type - тип домена, в загальному випадку цей тип CORPORATE. Також є типи для спеціальних доменів: PUBLIC, GEOGRAPHICAL - географічний, GENERIC - загально-обмеженого використання (приміром, com.ua).

descr - текстовий опис домена, вказується при реєстрації. Значення поля можна поміняти у будь-який момент терміну реєстрації. Необов'язкове.

admin - o - унікальний ідентифікатор (nic - handle) будь-якої організації або персони адміністратора домена. Адміністратор домена – та організація або персона, яка має право користування цим доменом, по суті справи власник. Ідентифікатори персони виду NAME - RIPN, ідентифікатори організації виду NAME - ORG - RIPN.

nserver - список DNS -серверів, що підтримують домен (якщо ім'я сервера містить ім'я домена, то вказується також його IP-адрес через пропуск), для успішного делегування домена в зоні .UA потрібна наявність не менше двох DNS-серверів, розташованих в різних мережах класу С.

created - дата створення запису про домен в базі даних регістратора, по суті справи, дата реєстрації домена. Не змінюється при продовженні терміну реєстрації, зміні адміністратора або реєстратора домена.

state - стан домена. Для доменів, зареєстрованих через регістратора, може мати декілька значень:

Delegated till [дата] - дата закінчення терміну реєстрації домена

Not delegated - домен не делегований

Not delegated freeing date [дата] - домен не делегований у зв'язку із закінченням терміну реєстрації; якщо до вказаної дати термін реєстрації домена не буде продовжений, реєстрація домена анулюється**.**

mnt - by - ідентифікатор служби технічної підтримки (служби авторизації), що відповідає за коректність інформації про домен в базі даних, ідентифікатор служби тех. підтримки має вид NAME - MNT - RIPN. Звичайно це ідентифікатор служби підтримки хостинг-провайдера.

source - джерело інформації. Для усіх доменів в зоні .UA - значення TC - RIPN.

Після блоку з інформацією про домен виводиться блок з повнішою інформацією про власника (адміністраторові) домена. Чесно кажучи, ми б

отримали б такий блок інформації, якби ввели у форму для перевірки не ім'я домена, а ідентифікатор (nic - handle) власника домена.

Рядок "Lastupdated on :" містить дату і час останньої зміни в базі даних регістратора, нехай навіть і не пов'язаного з Вашим доменом. При нормальному режимі роботи бази даних відрізняється від поточної не більш, ніж на 10-15 хвилин.

Кожен реєстратор використовує свою базу даних для підтримки ідентифікаторів персон і організацій. Тому в даному випадку Ви не побачите відповідають поля `admin - o`, замість нього введено два поля - `org` і `person`, які використовуються залежно від того, на фізичне або юридичне обличчя зареєстрований домен. Значеннями цього поля відповідно є назва організації або ім'я, прізвище людини, на яку зареєстрований домен, латинськими буквами.

Також замість вказівки закінчення терміну делегування в значенні поля `state`, введено нове поле `paid - till`, яке якраз і містить ці відомості. Саме поле `state` може мати одне з двох значень :

REGISTERED, DELEGATED - домен делегований

REGISTERED, NOT DELEGATED - домен не делегований

В таблиці 4.5 зазначені рідкісні і спеціальні поля в `whois` відповіді.

Таблиця 4.5 – Рідкісні і спеціальні поля в whois відповіді

Назва поля	Визначення
`reg - ch`	У разі, якщо йде процес зміни реєстратора, це поле містить ідентифікатор реєстратора, якому передається домен
`remark`	Довільні текстові коментарі (поле необов'язкове)
`whois - whois`	Сервіс реєстратора (поки не використовується)
`www - URL`	Адрес сайту реєстратора (поки не використовується)

x - freeing	Домен підлягає видаленню з реєстру протягом години

Утиліта Nslookup. Надає відомості, призначені для діагностики інфраструктури DNS. Засіб командного рядка Nslookup доступний, тільки якщо встановлений протокол TCP/IP.

```
nslookup [-підкоманда..] [{шуканий_комп'ютер | -сервер}]
```

--підкоманда..

Задає одну або декілька підкоманд nslookup як параметри командного рядка.

шуканий_комп'ютер

Задає пошук даних для комп'ютера шуканий_комп'ютер з використанням поточного, заданого за замовчуванням сервера імен DNS, якщо ніякого іншого сервера не вказано. Щоб отримати відомості про комп'ютер не з поточного домена DNS, в кінець імені має бути додана точка.

--сервер

Вказує, що цей сервер слід використовувати як сервер імен DNS. Якщо параметр -сервер не вказаний, використовується сервер DNS, заданий за умовчанням.

```
{ help | ?}
```

Виводить короткий опис підкоманд nslookup.

Логіка роботи команди наступна:

- Якщо шуканий_комп'ютер заданий IP -адресом, а запрошується запис ресурсу типу A або PTR, буде повернено ім'я комп'ютера.
- Якщо шуканий_комп'ютер заданий ім'ям без замикаючої точки, ім'я домена DNS, використовуваного за замовчуванням, буде додано до вказаного імені. Поведінка залежить від стану наступних підкоманд команди set : domain, srchlist, defname і search.
- Якщо в командному рядку введений дефіс (-) замість параметра шуканий_комп'ютер, команда nslookup перейде в інтерактивний режим.

Довжина рядка виклику команди не може перевищувати 256 символів.

Команда nslookup може працювати в двох режимах: інтерактивному і звичайному (автономному).

Якщо вимагається вивід тільки невеликої частини інформації, слід використовувати звичайний режим. Як перший параметр слід використовувати `ім'я` або `IP-адрес` комп'ютера, про який потрібно отримати дані. Як другий параметр введіть `ім'я` або `IP-адрес` сервера імен DNS. Якщо другий параметр не заданий, командою `nslookup` використовується сервер імен DNS, встановлений за замовчуванням.

Якщо вимагається отримати більш детальні відомості, слід використовувати інтерактивний режим. Як перший параметр слід ввести знак дефіса (-), як другий параметр - `ім'я` або `IP-адрес` сервера імен DNS. Якщо обидва параметри не задано, командою `nslookup` використовується сервер імен DNS, встановлений за замовчуванням.

Якщо при обробці запиту виникла помилка, командою `nslookup` на екран буде виведено повідомлення. В таблиці 4.6 перераховані можливі повідомлення про помилки.

Таблиця 4.6 – Перелік можливих повідомлень про помилки

Повідомлення про помилку	Опис
`Timed out`	Сервер не відповів на запит протягом певного часу і після певного числа повторних спроб. Є можливість встановити період очікування за допомогою підкоманди `set timeout`. Є можливість встановити число повторних спроб за допомогою підкоманди `set retry`.
`No response from server`	Сервер імен DNS не запущений на сервері

No records	Сервер імен DNS не містить записів про ресурси вказаного типу, хоча ім'я комп'ютера задане вірно. Тип запиту задається командою set querytype.
Nonexistent domain	Заданий комп'ютер або ім'я домена DNS не існує.
Connection refused --или- Network is unreachable	Неможливо підключитися до сервера імен DNS або до сервера служби finger. Ця помилка зазвичай виникає із запитами команд ls і finger.

Продовження таблиці 4.6

Server failure	Сервер імен DNS виявив внутрішню невідповідність у своїй базі даних і не може коректно відповісти на запит.
Refused	Відмовлено в обробці запиту сервером імен DNS.
Format error	Сервер імен DNS виявив помилку у форматі отриманого пакету. Це може свідчити про помилку в команді nslookup.

Наприклад, щоб змінити встановлений за замовчуванням тип запиту про відомості для вузла і встановити початковий час очікування рівним 10 секундам, слід ввести команду:

```
nslookup - querytype=hinfo - timeout=10
```

4.3.2Завдання до третьої частини лабораторної роботи

1. За допомогою утиліти `netstat` отримати список з'єднань, відкритих на сервері `cisco-academy.com.ua`. Прокоментувати з'єднання, що знаходяться в режимі ESTABLISHED (внутрішній або зовнішній інтерфейс, з яким вузлом, за яким протоколом).

2. Отримати таблицю маршрутизації. Вказати, через які інтерфейси з якими мережами відбувається зв'язок, ім'я шлюзу, маски локальних мереж.

3. Отримати статистику мережевих інтерфейсів. Побудувати графіки статистичної інформації для внутрішньої (eth0) і зовнішньої (eth1) локальних мереж, пояснити значення встановлених прапорів. Порівняти кількість помилок з вимогами, що пред'являються до роботи мереж; зробити висновок про роботу локальної мережі.

4. Отримати статистику мережевих інтерфейсів. Проаналізувати роботу кожного з протоколів. Для протоколу ICMP побудувати графіки вхідної і вихідної гістограм.

5. За допомогою команди ping перевірити стан зв'язку з вузлами acts.kpi.ua (каф. АУТС), www.kpi.ua (НТУУ «КРІ»), www.kmda.gov.ua (КМДА), www.gmail.com (GOOGLE) - обов'язково; www.romeguide.it (Італія), www.novol.pl (Польща), www.newslink.org (США) або іншими за бажанням, але не менше 7-и вузлів. Число запитів, що відправляються, рекомендується взяти рівним 20.

6. Результати досліджень представити в таблиці:

Доменне ім'я	IP -адрес	Країна	Число втрачених запитів, %	Середній час проходження запиту, мс	255 - TTL

7.Представити графіки статистичної інформації.

8.Провести трасування вузлів www.google.com, www.romeguide.it, www.novol.pl, www.newslink.org або будь-яких інших за бажанням, але не менше 7-и вузлів. Результати протоколювати у файл st.log.

9.Описати маршрут проходження для двох вибраних вузлів (країна, місто, мережа).

4.3.3Контрольні запитання до третьої частини лабораторної роботи

1. Навіщо потрібна модель OSI, скільки рівнів вона має?
2. Що відбувається на першому рівні?
3. Для чого потрібний канальний рівень?
4. Що забезпечує транспортний рівень?
5. Що являє собою прикладний рівень?
6. Що таке мережевий інтерфейс?
7. Як називається основна утиліта для перегляду і управління параметрами мережевих інтерфейсів в LINUX
8. На що спрямовані допоміжні утиліти?
9. Які дві найбільш поширені реалізації `ping` утиліт?
10. Яку служку `NTPSync` клієнт використовує для своєї роботи?
11. Про що дозволяє отримати інформацію сервіс `whois`?
12. Яка команда надає відомості, призначені для діагностики інфраструктури DNS?

ЛАБОРАТОРНА РОБОТА №5. АРХІВНЕ КОПІЮВАННЯ В LINUX

5.1 Архівне копіювання даних в Linux. Backup-manager

5.1.1 Теоретичні відомості

Системне адміністрування – це все те, що потрібно для підтримки працездатності комп'ютерної системи (наприклад, створення резервних копій деяких файлів, установка й відновлення програм, створення й настроювання облікових записів користувачів, забезпечення роботи мережі т.д.). У даній лабораторній роботі буде розглянута проблема автоматичного резервного копіювання даних в ОС Linux.

Робота в консолі — не змушений захід, викликана відсутністю “нормальних” засобів, і не бравада досвідчених користувачів, а найшвидший і зручний інтерфейс для розв'язку ряду завдань. У чому ж її переваги? Насамперед — в універсальності. Незалежно від того, який дистрибутив ви використовуєте, базові команди будуть ті самі. Не можна забувати й про те, що текстовий режим стійкіше графічного. Згадаєте хоча б знаменитий Bsod (“синій екран смерті”) в Windows. Напис чомусь відображається саме в консолі, а не в красиво промальованому вікні. Оскільки графічний інтерфейс Linux — це по суті звичайна прикладна програма, то її непрацездатність не приводить до загального краху системи. Якщо користувач не боїться текстового режиму, то він швидко внесе необхідні зміни у відповідний конфігураційний файл і заново запустить систему. А якщо ні, то прийде вдатися до повної її переустановки, що значно довше. Нарешті, консольні команди зручні при виконанні деяких рутинних операцій. Адже комп'ютер і придуманий для автоматизації робочого процесу. Зрозуміло, для того щоб робота в консолі була ефективної, користувачеві прийде витратити небагато

часу на вивчення стандартних команд Linux. Але воно компенсується досить швидко.

Перейти в режим командного рядка можна двома способами. Перший — активація текстової консолі. Для цього слід нажати комбінацію клавіш Ctrl+Alt+F[номер консолі]. З'явиться рядок запрошення на реєстрацію в системі, де потрібно послідовно набрати логін і пароль. Як правило, у новій системі за замовчуванням є користувач із логіном root і паролем secure. Його ми й будемо використовувати для роботи в консолі з утилітою Backup-Manager.

Мета нашої роботи – налаштувати `backup-manager` таким чином, щоб задовольнити потреби резервного копіювання даних.

1. Термінологія.

Позначення й поняття викладені у таблиці 4.1:

Таблиця 5.1 – Поняття та їх візначення

Поняття	Визначення
Резервне копіювання	Процес створення копії даних на носії (жорсткому диску, дискеті і т.д.), призначеному для відновлення даних або розташування їх у новому місці у випадку ушкодження або руйнування оригіналу.
Bash	Командний процесор Linux написаний для проекта GNU. Його ім'я це акронім від Bourne-again shell — гра слів (Bourne again / born again) в Bourne shell (sh), який був раніше дуже важливим командним процесором.. Bash це командний процесор за замовчуванням у багатьох Linux системах.

Продовження таблиці 5.1

Архіватор	Програма, що здійснює впакування одного і більше файлів в архів або серію архівів, для зручності переносу або зберігання, а також розпакування архівів. Багато архиваторів використовують стиснення без втрат.
Стиск без втрат	Метод стиснення даних: відео, аудіо, графіки, документів представлених у цифровому виді, при використанні якого закодовані дані можуть бути відновлені з точністю до біта.
Репозито́рій, сховище	Місце, де зберігаються і підтримуються деякі дані. Найчастіше дані в репозиторії зберігаються у вигляді файлів, доступних для подальшого розповсюдження.
Планувальник завдань	Программа або сервіс операційної системи, яка запускає інші програми залежно від різних критеріїв, наприклад: настання певного часу; операційна система переходить у певний стан (бездіяльність, режим сну і т.д.); надійшов адміністративний запит через користувацький інтерфейс або через інструменти віддаленого адміністрування.

Продовження таблиці 5.1

Tar	Найпоширеніший архіватор, використовуваний в Linux-Системах. Сам по собі `tar` не є архіватором у звичному розумінні цього слова, тому що він самостійно не використовує стиснення. У той же час, багато архіваторів (наприклад, Gzip або bzip2) не вміють стискати кілька файлів, а працюють тільки з одним файлом або вхідним потоком. Тому найчастіше ці програми використовуються разом: `tar` створює незжатий архів, у який містяться обрані файли і каталоги, при цьому зберігаючи деякі їхні атрибути (такі як права доступу). Після цього отриманий файл `*.tar` стискується архіватором, наприклад, `gzip`.
Backup-Manager	Це утиліта для роботи в командному рядку, що дозволяє спростити роботу із щоденного створення копій важливих даних. Написана на `bash` і `perl`, ця утиліта дозволяє створювати архіви в багатьох форматах (tar, gzip, bzip2, lzma, dar, zip) і надає такі цікаві можливості як передача створених файлів по мережі або автоматичний запис на CD/DVD... Програма створена для простого використання й досить популярна серед звичайних користувачів і системних адміністраторів. Увесь процес резервного копіювання описаний в одному конфігураційному

	файлі, який настільки простий, що для його настроювання буде потрібно не більш 5 хвилин. Створені архіви зберігаються зазначене число днів. Залежно від зазначеної конфігурації самої утиліти та планувальника завдань можливе регулярне автоматичне створення бекапу. Також утиліта підтримує інкрементальний бекап, який тільки доповнює раніше створений повний, що дозволяє скоротити час резервного копіювання.

2. Будова конфігураційного файлу Backup-Manager `/etc/backup-manager.conf`. Файл конфігурації розділений на секції (наведено основні) в таблиці 5.2.

Таблиця 5.2 – Секції файла конфігурації

Секція	Визначення
`Repository`	Секція в якій зазначається локальний репозиторій (сховище) архівів.
`Archives`	Секція в якій налаштовується термін зберігання бекапів, формат їх назви, методи архівування які будуть задіяні т.п.

Секції налаштування методів зазначені у таблиці 5.3.

Таблиця 5.3 – Секції налаштування методів

Метод	Застосування
`Tarball`	Найпростіший метод, який при кожному запуску робить повну

	копію обраних каталогів.

Закінчення таблиці 5.3

Tarball-incremental	Складніший метод, який дозволяє економити час та дисковий простір, тому що додає в старий архів лише ті дані, які з минулого разу було змінено. Зазвичай використовується для регулярного частого викотистання.

Зверніть увагу, що пропуски в іменах папок, баз, репозиторіїв неприпустимі. Рядки, які закоментовані символом # не впливають на роботу програми і необхідні лише для довідки. Незакоментовані строки для зручності виділено жирним шрифтом.

3. Сам файл конфігурації з коментарями /etc/backup-manager.conf:

```
#################################################################
# Repository - місце зберігання архівів
#################################################################
# Папка, у якій будуть зберігатися архіви
export BM_REPOSITORY_ROOT="/Zia61/arch12s"
# З міркувань безпеки всі створювані архіви
# будуть доступні на читання й запис тільки для зазначених
# нижче користувача й групи. Опція рекомендується в "true"
export BM_REPOSITORY_SECURE="true"
# користувач і група власника архівів
export BM_REPOSITORY_USER="root"
export BM_REPOSITORY_GROUP="root"

#################################################################
# Archives - Режим резервування даних (створення архівів)
#################################################################
# Скільки днів зберігати архіви (час життя)
export BM_ARCHIVE_TTL="14"
```

```
# Заміняти однакові по розміру архіви м'якими посиланнями
export BM_ARCHIVE_PURGEDUPS="true"

# Префікс кожного архіву ( за замовчуванням ім'я хоста $HOSTNAME)
export BM_ARCHIVE_PREFIX="$HOSTNAME"

# Метод(и) створення резервних копій:
# - tarball (повна копія каталогів, стисла архіватором)
# - tarball-incremental (інкрементальна стисла копія каталогів)
# кілька методів пишуться через пропуски
export BM_ARCHIVE_METHOD="tarball-incremental"

#################################################################
# Метод «tarball»
# - повна заархівована копія каталогу
#################################################################
# Формат імені файлів архівів
#       long  : host-full-path-to-folder.tar.gz
#       short : parentfolder.tar.gz
export BM_TARBALL_NAMEFORMAT="long"
# Тип архівів: tar, tar.gz, tar.bz2, zip.
export BM_TARBALL_FILETYPE="tar.gz"
# Заміняти всі м'які посилання файлами (не рекомендується)
export BM_TARBALL_DUMPSYMLINKS="false"
# Директорії для резервування (розділяються пропуском)
export BM_TARBALL_DIRECTORIES="/Zia61/stud12"
# якщо каталогів багато, простіше їх записати в стовпчик от так:
# export BM_TARBALL_DIRECTORIES="/Zia61/stud12 \
# /Zia61/stud13"

# Файли, які НЕ потрібно архівувати (виключення)
export BM_TARBALL_BLACKLIST=""

#################################################################
#  Метод «tarball-incremental»
# - інкрементальна стисла копія каталогу
#################################################################
# Як часто зберігати повну копію?
# Можливі значення: weekly (раз у тиждень), monthly (раз на місяць)
export BM_TARBALLINC_MASTERDATETYPE="weekly"
# Номер дня в BM_TARBALLINC_MASTERDATETYPE (тиждень або місяць),
# коли потрібно зберігати повну копію
export BM_TARBALLINC_MASTERDATEVALUE="6"
# Приклад: зберігати повну копію щоп'ятниці:
# BM_TARBALLINC_MASTERDATETYPE="weekly"
# BM_TARBALLINC_MASTERDATEVALUE="5"
#
# Приклад: зберігати повну копію 1 числа кожного місяця:
# BM_TARBALLINC_MASTERDATETYPE="monthly"
# BM_TARBALLINC_MASTERDATEVALUE="1"

#################################################################
# Установки для досвідчених користувачів
#################################################################

# Записувати вивід в syslog "true"/"false"
export BM_LOGGER="true"

# Запустити цю команду до виконання backup-manager (бэкапа).
```

```
export BM_PRE_BACKUP_COMMAND=""

# Запустити цю команду після виконання backup-manager (бэкапа).
export BM_POST_BACKUP_COMMAND=""
```

4. Приклад налаштування і перевірки Backup-Manager

Для початку запустимо консоль (Ctrl+Alt+F1) і подивимося в якому каталозі перебуваємо:

```
[root@host-7-129 ~]# pwd
/root
```

Створимо тестову папку і файл у ній, для яких ми будемо створювати резервні копії:

`[root@host-7-129 ~]# Mkdir zia61` — створюємо каталог з номером групи

`[root@host-7-129 ~]# Cd zia61` - міняємо робочий каталог на новий

`[root@host-7-129 zia61]# mkdir stud12` - створюємо каталог з номером по списку

`[root@host-7-129 zia61]# cd stud12` - зміна каталогу

`[root@host-7-129 stud12]# vi stud12.txt` - створюємо файл stud12.txt і записуємо текст Hello world закінчивши набір тексту натискаємо ESC w і Enter

```
[root@host-7-129 stud12]# ls –al  - перевіримо чи створився файл
drxwr-xr-x. 2root root  4096 Jan 26 12:25 .
drxwr-xr-x. 2root root  4096 Jan 26 12:25 ..
-rw-------. 1root root 12288 Jan 26 12:25 stud12.txt
```

`[root@host-7-129 stud12]# cd .. –` піднімаємося на 1 рівень вище по дереву папок

`[root@host-7-129 zia61]# mkdir arch12 –` створюємо склад архівів

Текстова папка з файлом та склад архівів готові, тепер відкриємо і налаштуємо файл `etc/backup-manager.conf`:

```
[root@host-7-129 stud12]# cd etc/
```

```
[root@host-7-129 etc]#cp backup-manager.conf backup-manager_old.conf –
```

робимо бекап файлу налаштувань.

[root@host-7-129 etc]# vi backup-manager.conf – налаштовуємо backup-manager за допомогою редактора тексту “Vi” (наведено головні налаштування)

```
# Папка, у якій будуть зберігатися архіви
export BM_REPOSITORY_ROOT="/Zia61/arch12"

# Скільки днів зберігати архіви (час життя)
export BM_ARCHIVE_TTL="14"

# Префікс кожного архіву (ставимо фамілію та номер по списку)
export BM_ARCHIVE_PREFIX="Ivanov12"

# Метод(и) створення резервних копій:
# - tarball (повна копія каталогів, стисла архіватором)
# - tarball-incremental (інкрементальна стисла копія каталогів)
export BM_ARCHIVE_METHOD="tarball-incremental"

#############################################################
# Метод «tarball»
# - повна заархівована копія каталогу
#############################################################
# Формат імені файлів архівів
#       long  : host-full-path-to-folder.tar.gz
#       short : parentfolder.tar.gz
export BM_TARBALL_NAMEFORMAT="short"
# Тип архівів: tar, tar.gz, tar.bz2, zip.
export BM_TARBALL_FILETYPE="tar.gz"
# Директорії для архівування (розділяються пропуском)
export BM_TARBALL_DIRECTORIES="/Zia61/stud12"

#############################################################
#  Метод «tarball-incremental»
# - інкрементальна стисла копія каталогу
#############################################################
# Як часто зберігати повну копію?
# Можливі значення: weekly (раз у тиждень), monthly (раз на місяць)
#export BM_TARBALLINC_MASTERDATETYPE="weekly"
# Номер дня (тиждень або місяць), коли потрібно зберігати повну копію
#export BM_TARBALLINC_MASTERDATEVALUE="5"
# Приклад: зберігати повну копію щоп'ятниці:
```

```
BM_TARBALLINC_MASTERDATETYPE="weekly"
BM_TARBALLINC_MASTERDATEVALUE="5"
#
# Приклад: зберігати повну копію 1 числа кожного місяця:
# BM_TARBALLINC_MASTERDATETYPE="monthly"
# BM_TARBALLINC_MASTERDATEVALUE="1"
```

Натискаєте ESC ZZ Enter для того щоб вийти та зберегти конфігурацію.

Тепер запустимо backup-manager і перевіримо його працездатність:

```
[root@host-7-129 etc]# backup-manager
```

Зайдемо в директорію де будуть зберігатися архіви, щоб перевірити чи зроблено копію.

```
[root@host-7-129 etc]# cd /zia61/arch12
```

Дивимося вміст директорії:

```
[root@host-7-129 arch12]# Ls -al 22.01.2012  15:29    <DIR>          .
22.01.2012  15:29    <DIR>          ..
22.01.2012  15:27               354 Ivanov12arch12.tar.gz
               1 File(s)            354 bytes
2 Dir(s)   5 460 527 230 bytes free
```

5.1.2Завдання до лабораторної роботи

Створити папки «root\Код групи\stud_№ в списку» і «root\Код групи\arch_№ в списку». У папці «root\Код групи\stud_№ в списку» створити текстовий файл з іменем «Прізвище.txt» і заповнити його довільним текстом. Налаштувати та перевірити роботу Backup-manager. Вимоги до налаштування наведені в таблиці 5.4.

Таблиця 5.4 – Вимоги до налаштування Backup-manager

№ за списком	Завдання

1	Час життя архіву – 1 день. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar. Метод «tarball-incremental». Зберігати повну копію щопонеділка.

Продовження таблиці 5.4

2	Час життя архіву – 2 дні. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: tar.gz. Метод «tarball-incremental». Зберігати повну копію щовівторка.
3	Час життя архіву – 3 дні. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar.bz2. Метод «tarball-incremental». Зберігати повну копію щосереди.
4	Час життя архіву – 4 дні. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: zip. Метод «tarball-incremental». Зберігати повну копію щоп’ятниці.
5	Час життя архіву – 5 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar. Метод «tarball-incremental». Зберігати повну копію кожного 5 числа місяця.
6	Час життя архіву – 6 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: tar.gz. Метод «tarball-incremental». Зберігати повну копію кожного 6 числа місяця.
7	Час життя архіву – 7 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar.bz2. Метод «tarball-incremental». Зберігати повну копію кожного 7 числа місяця.
8	Час життя архіву – 8 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: zip. Метод «tarball-incremental». Зберігати повну копію кожного 8 числа

	місяця.
9	Час життя архіву – 9 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar. Метод «tarball-incremental». Зберігати повну копію кожного 9 числа місяця.

Продовження таблиці 5.4

10	Час життя архіву – 10 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: tar.gz. Метод «tarball-incremental». Зберігати повну копію кожного 10 числа місяця.
11	Час життя архіву – 11 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar.bz2. Метод «tarball-incremental». Зберігати повну копію щопонеділка.
12	Час життя архіву – 12 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: zip. Метод «tarball-incremental». Зберігати повну копію щовівторка.
13	Час життя архіву – 13 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar. Метод «tarball-incremental». Зберігати повну копію щосереди.
14	Час життя архіву – 14 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: tar.gz. Метод «tarball-incremental». Зберігати повну копію кожний четвер.
15	Час життя архіву – 15 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar.bz2. Метод «tarball-incremental». Зберігати повну копію щоп’ятниці.

16	Час життя архіву – 16 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: zip. Метод «tarball-incremental». Зберігати повну копію кожного 16 числа місяця.
17	Час життя архіву – 17 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar. Метод «tarball-incremental». Зберігати повну копію кожного 17 числа місяця.

Продовження таблиці 5.4

18	Час життя архіву – 18 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: tar.gz. Метод «tarball-incremental». Зберігати повну копію кожного 18 числа місяця.
19	Час життя архіву – 19 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar.bz2. Метод «tarball-incremental». Зберігати повну копію кожного 19 числа місяця.
20	Час життя архіву – 20 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: zip. Метод «tarball-incremental». Зберігати повну копію кожного 20 числа місяця.
21	Час життя архіву – 21 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar. Метод «tarball-incremental». Зберігати повну копію щопонеділка.
22	Час життя архіву – 22 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: tar.gz.

	Метод «tarball-incremental». Зберігати повну копію щовівторка.
23	Час життя архіву – 23 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Long”. Тип архівів: tar.bz2. Метод «tarball-incremental». Зберігати повну копію щосереди.
24	Час життя архіву – 24 днів. Префікс до ім’я архіву – «Прізвище_№ по списку». Формат імені файлів архівів – “Short”. Тип архівів: zip. Метод «tarball-incremental». Зберігати повну копію кожний четвер.

5.1.3Контрольні запитання до лабораторної роботи

1. Що таке резервне копіювання?
2. Навіщо потрібен архіватор?
3. Що таке `Bash`?
4. Для чого необхідний репозиторій?
5. Що таке `Tar`?
6. Як зробити бекап файлу?
7. Як дізнатися в якому каталозі знаходимося?
8. На які секції розділений файл конфігурації?
9. Як називається метод стиснення даних: відео, аудио, графіки, документів представлених у цифровому виді, при використанні якого закодовані дані можуть бути відновлені з точністю до біта?

6 ЗАГАЛЬНІ РЕКОМЕНДАЦІЇ ТА ВИМОГИ ДО РОБОТИ

6.1 Загальні рекомендації до виконання роботи

Згідно з учбовим планом, студенти денної форми навчання мають 32 годин лабораторних робіт та 100 годин самостійної роботи. Перед виконанням лабораторної роботи студент має детально вивчити матеріал з курсу «Безпека та захист операційних систем», підібрати відповідну навчальну, методичну та спеціальну літературу за рекомендованим списком (а також самостійно підібраною).

6.2 Вимоги до виконання практичних робіт

При виконанні лабораторної роботи необхідно дотримуватись таких вимог:

– розв’язок завдань має супроводжуватись поясненням, при необхідності, формулами тощо.

– лабораторна робота повинна бути охайно оформленою, розбірливо написана без скорочення слів (крім загальновідомих скорочень);

– сторінки роботи повинні бути пронумеровані;

– всі етапи виконання алгоритму роботи/програм/скриптів повинні бути прокоментовані.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Rist O. Getting Started with VirtualBox / Oliver Rist. – Packt Publishing, 2018. - 190 p.
2. Preece R. VirtualBox 6.0: Beginners Guide on how to Install, Configure, and Use VirtualBox / Robert Preece. – Independently published, 2019. – 70 p.
3. Russell D. Oracle VM VirtualBox: Beginner's Guide / Russel Dusty. – Packt Publishing, 2018. – 284 p.
4. Herter N. VirtualBox 6.0 – The Ultimate Beginner's Guide / Nicolas Herter. – Independently published, 2019. – 97 p.
5. Silverman S. Oracle VM VirtualBox: An Ultimate Guide Book on Oracle Virtualization Essentials / Steve Silverman. – Independently published, 2018. – 87 p.
6. Lacroix J. Mastering Ubuntu Server: Master the art of deploying, configuring, managing, and troubleshooting Ubuntu Server 18.04 [Електронний ресурс] / J. Lacroix. - Packt Publishing - ebooks Account; 2nd Revised edition (June 11, 2018). – Режим доступу : http://surl.li/plnvt. - Назва з екрана.
7. Tanenbaum A. Modern Operating Systems, 4th Edition / A. Tanenbaum, H. Bos. – Pearson, 2015. – 1104 p.
8. Nemeth E. UNIX and Linux System Administration Handbook, 5th Edition / Evi Nemeth, Garth Snyder, Trent Hein, Ben Whaley, Dan Mackin. – Addison-Wesley Professional, 2017. – 1232 p.
9. Kerrisk M. The Linux Programming Interface: A Linux and UNIX System Programming Handbook / M. Kerrisk. – No Starch Press, 2010. – 1552 p.
10. Johnson C. Pro Bash Programming, Second Edition: Scripting the GNU/Linux Shell, 2nd Edition / Chris Johnson, Jayant Varma. – Apress, 2015. – 279 p.
11. Зайцев, В. Г. Операційні системи [Електронний ресурс] : навч. посіб для студентів спеціальності 123 «Комп’ютерна інженерія» / В. Г.

Зайцев, І. П. Дробязко ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,22 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 240 с. – Назва з екрана.

12. Погребняк Б.І. Операційні системи: навч. посіб. / 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.

13. Федотова-Півень І.М. Операційні системи: навч. посіб. [] / І.М. Федотова-Півень, І.В. Миронець, О.Б. Півень, та ін. ; за ред. В.М. Рудницького; Черкаський державний технологічний університет. – Харків : ТОВ «ДІСА ПЛЮС», 2019. – 216 с.

14. Микитишин А.Г. Операційні системи: конспект лекцій / уклад. А.Г. Микитишин, І.В. Чихіра. – Тернопіль: ТНТУ імені Івана Пулюя, 2016. – 107 с.

ПРИМІТКИ