

# Основи Linux

## Урок 3

# Порядок денний

- Розуміння процесів: джерело життя системи Linux
- Занурення в типи процесів
- Робота з процесами: основні інструменти та команди
- Інтерпретація кодів завершення команд і сценаріїв
- Знайомство із сигналами переривання
- Вступ до послідовних пристройів і терміналів
- Спілкування з іменованими каналами конвеєра
- Автоматизація завдань за допомогою системного планувальника: Cron
- Слідкуйте за системою: основи моніторингу системи
- Розуміння системних журналів: повідомлення, системний журнал тощо
- Глибше занурення в конкретні журнали: Auth.log, Dpkg.log та інші
- Знайомство з Auditd: моніторинг подій на детальному рівні
- Ефективне керування журналами за допомогою сценаріїв ротації журналів



# Розуміння процесів: джерело життя системи Linux



# Основа системи Linux

Процеси — це запущені екземпляри програми (виконуваний файл) у системі Linux

Вони є джерелом життя системи, оскільки виконують усі важливі завдання, необхідні для підтримки роботи системи

## Важливість процесів у Linux

- Процеси дозволяють системі виконувати кілька завдань одночасно
- Вони дозволяють програмам виконуватися одночасно, покращуючи продуктивність системи
- Процеси допомагають ізолювати програми, запобігаючи їхньому втручанню одна в одну

# Стани та атрибути процесу

## Стани процесу

- Процеси в Linux можуть перебувати в одному з п'яти станів: **запущений, сплячий, зупинений, зомбі та відстежений**
- Стан процесу визначає його поведінку та те, як він взаємодіє з системою

## Атрибути процесу

- Кожен процес у Linux має унікальний ідентифікатор процесу (PID)
- Інші важливі атрибути процесу включають ідентифікатор його батьківського процесу (PPID), використання пам'яті, ЦП використання та пріоритет

# Інструменти моніторингу процесів

Команда "ps" використовується для перегляду процесів, що виконуються в системі Linux. Вона показує статичний знімок процесів на момент виконання команди. За замовчуванням він відображає процеси, якими володіє поточний користувач, у табличному форматі, але його можна використовувати з різними параметрами для відображення більш детальної чи конкретної інформації.

"ps -aux" відображає всі процеси, запущені в системі, разом з їхніми атрибутами.

Команда "top" - це інструмент моніторингу процесів у режимі реального часу, який відображає інформацію про процеси, які зараз виконуються в системі Linux. Він відображає процеси в динамічному інтерактивному форматі, періодично оновлюється, дозволяючи користувачеві контролювати систему в режимі реального часу та сортувати процеси за різними критеріями.

Інші засоби моніторингу процесів включають «htop» і «glances», які надають більш розширені функції для моніторингу процесів у системі Linux.

# Управління процесами в Linux

## Життєвий цикл процесу

- Процес починається, коли користувач або система запускає програму
- Операційна система створює новий процес і виділяє йому системні ресурси
- Потім процес виконує програмний код, виконує свої завдання та чекає на подальші інструкції
- Коли процес завершить виконання своїх завдань, операційна система завершить його роботу

Linux керує процесами, розподіляючи для них системні ресурси. Операційна система надає унікальний ідентифікатор процесу (PID) для кожного процесу, що дозволяє системі ефективно ним керувати.

Процесами керують за допомогою таких команд, як «ps», «kill» і «nice». Команда "kill" використовується для завершення процесу. Команда "nice" використовується для встановлення пріоритету процесу, що дозволяє користувачеві контролювати, скільки процесорного часу виділяється процесу

# Занурення в типи процесів

# Основні та фонові процеси

**Основні процеси** – це процеси, які зараз виконуються в оболонці та потребують введення користувача.

**Фонові процеси** – це процеси, які виконуються незалежно від оболонки й не потребують введення користувача.

Випадки використання:

- Процеси переднього плану: виконання команд, які вимагають введення користувача, наприклад, текстові редактори чи інтерактивні програми.
- Фонові процеси: виконання тривалих завдань, які не потребують введення користувача, наприклад резервне копіювання системи або передача файлів

Інструменти командного рядка: fg, bg, jobs, ctrl-z

# Батьківські та дочірні процеси

**Дочірній процес** — це процес, створений іншим процесом, відомим як батьківський процес. Дочірні процеси успадковують певні характеристики свого батьківського процесу, такі як змінні середовища та дескриптори файлів.

**Батьківські процеси** можуть породжувати кілька дочірніх процесів.

Випадки використання:

- Батьківські процеси: керування кількома екземплярами програми, виконання сценаріїв, які запускають інші програми чи процеси.
- Дочірні процеси: виконання певного завдання або функції, наприклад дочірній процес, створений веб-сервером для обробки вхідних запитів.

Інструменти командного рядка: ps, top, kill

# Демонічні процеси

**Демон-процеси** — це тривалі фонові процеси, які виконують системні завдання або надають послуги.

Вони не мають керуючого терміналу і працюють як користувачі системи.

Випадки використання:

- Надання мережевих послуг, таких як веб-сервери, сервери електронної пошти або сервери DNS.
- Виконання системних завдань, таких як ротація журналів, резервне копіювання або оновлення.

Інструменти командного рядка: `systemctl`, `service`



# Робота з процесами: Основні інструменти та команди



# ps

Призначення: показ інформації про запущені процеси.

Основне використання: ps [параметри]

Опції:

- -e: відобразжати інформацію про всі процеси.
- -f: показати детальну інформацію про процеси.

Поради:

- Використовуйте "ps aux", щоб відобразити повний список процесів із детальною інформацією.
- Використовуйте "ps -C <назва процесу>", щоб відобразити інформацію про певний процес.

# pstree

Призначення: pstree — це утиліта командного рядка, яка використовується для відображення деревоподібної структури процесів у системі.

Основне використання: pstree [ПАРАМЕТИ]

Опції:

- -p: Показує PID процесу.
- -u: Показує імена користувачів.
- -h: висвітлює поточний процес і його предків.
- -s: відображає аргументи командного рядка.

# pgrep

Призначення: pgrep (process grep) — утиліта командного рядка, яка використовується для пошуку процесів на основі їх імені, PID або інших атрибутів. Основне використання: pgrep [ПАРАМЕТРИ] ПАТТЕРН

Опції:

- -a: відображає весь командний рядок для кожного процесу.
- -l: відображає назву процесу та весь командний рядок.
- -u: шукає процеси, що належать певному користувачеві.
- -f: шукає процеси на основі повного командного рядка.

Поради:

- pgrep можна поєднувати з іншими командами, такими як kill, для автоматизації процесу керування процесом.
- За замовчуванням pgrep повертає лише PID відповідних процесів. Щоб відобразити більше інформації про процеси, ви можете використовувати параметри -a або -l.

# top

Призначення: забезпечує моніторинг процесів і ресурсів системи в реальному часі.

Основне використання: top

Опції:

- Натисніть «Shift+P», щоб відсортувати процеси за використанням ЦП.
- Натисніть «Shift+M», щоб відсортувати процеси за використанням пам'яті.

Поради:

- Використовуйте "top -d <seconds>", щоб встановити інтервал оновлення top.
- Використовуйте "top -u <ім'я користувача>", щоб відображати лише процеси певного користувача.

# htop

Призначення: схоже на top, але забезпечує більш зручний інтерфейс.

Основне використання: htop

Опції:

- Натисніть «F6», щоб відсортувати процеси за різними критеріями.
- Натисніть «F9», щоб надіслати сигнал вибраному процесу.

Поради:

- Використовуйте "htop -u <ім'я користувача>", щоб відображати лише процеси певного користувача.
- Використовуйте "htop -p <ідентифікатор процесу>" для моніторингу певного процесу.

# kill

Призначення: надсилає сигнал процесу про його завершення.

Основне використання: kill [параметри] <PID>

Опції:

- -9: негайно завершити процес.

Поради:

- Використовуйте "killall <ім'я процесу>", щоб завершити всі процеси з певним ім'ям

# nice і renice команди

Мета: налаштовувати пріоритет процесу, щоб контролювати використання його ЦП.

Основне використання: nice [параметри] <команда>

Опції:

- -n: встановити рівень пріоритету процесу.

Поради:

- Нижче значення пріоритету означаєвищий пріоритет для процесу.
- Використовуйте "renice <пріоритет> -p <PID>", щоб налаштовувати пріоритет запущеного процесу.

# lsof

Призначення: lsof (спісок відкритих файлів) — утиліта командного рядка, яка використовується для відображення інформації про файли та процеси, які зараз відкриті. Основне використання: lsof [ПАРАМЕТИ]

Опції:

- -р: показує відкриті файли для певного процесу.
- -и: показує відкриті файли для певного користувача.
- -і: показує відкриті файли для певного мережевого сокета.
- -с: показує відкриті файли для певної команди.

Поради:

- Використовуйте lsof, щоб діагностувати проблеми доступу до файлів і побачити, які файли зараз використовуються.
- Вивід lsof може бути досить багатослівним, тому часто корисно передавати його через інші команди, такі як grep або awk, щоб фільтрувати вивід.

# Інтерпретація команд і кодів завершення сценарію

# Розуміння статусів виходу в Linux

У Linux кожна виконана команда повертає статус виходу або код завершення.

Цей статус виходу вказує на успішне або невдале виконання команди.

Розуміння статусів виходу є важливим для усунення несправностей і обробки помилок у Linux.

## Конвенція про статус завершення

- У Linux статуси виходу представлені числовими значеннями.
- Статус виходу 0 вказує на успіх, тоді як будь-яке ненульове значення вказує на невдачу або помилку.
- Різні ненульові значення можуть вказувати на різні типи помилок залежно від команди.

# Розуміння статусів виходу в Linux

## Отримання статусу виходу

- Щоб отримати статус завершення останньої виконаної команди, використовуйте спеціальну змінну оболонки \$?
- Вартість \$? буде статусом завершення останньої виконаної команди.

# Статуси виходу в сценаріях оболонки

У сценаріях оболонки статуси виходу зазвичай використовуються для обробки помилок і керування потоком

Наприклад, ви можете використовувати статус виходу команди, щоб визначити, продовжувати чи виходити зі сценарію.

Ви також можете встановити статус виходу сценарію за допомогою команди "exit", яка може бути корисною для сигналізації про помилки іншим сценаріям або процесам.

# Приклади команд зі статусами виходу

Команда "ls": статус виходу 0 вказує на успіх, відмінний від нуля вказує на помилку (наприклад, дозвіл заборонено або файл не знайдено)

Команда "grep": стан виходу 0 вказує на успіх (відповідність знайдено), відмінне від нуля вказує на помилку (відповідність не знайдено)

Команда "ping": статус виходу 0 вказує на успіх (ціль доступна), не нуль вказує на помилку (ціль недоступна)

# Поради щодо ефективного використання статусів виходу

Завжди перевіряйте статус завершення команд у ваших сценаріях, щоб забезпечити належну обробку помилок і контроль потоку.

Використовуйте описові ненульові значення статусу виходу, щоб полегшити визначення причини помилок.

Комбінуйте команди за допомогою "&&" або "||" виконувати їх умовно на основі статусу виходу попередньої команди.

# Знайомство із сигналами переривання

# Сигнали в Linux

Сигнали – це спосіб для ядра Linux спілкуватися з процесами та сповіщати їх про певні події чи дії. Сигнали необхідні для управління та контролю процесів у Linux.

## Види сигналів

Існує багато різних типів сигналів, але одними з найбільш часто використовуваних сигналів є сигнали переривання, які дозволяють завершити або призупинити процеси.

## Обробка сигналів

- Процеси можуть обробляти сигнали різними способами, залежно від того, як вони запрограмовані.
- Процеси можуть ігнорувати сигнали, обробляти їх певним чином або навіть генерувати власні сигнали.
- Обробники сигналів використовуються для визначення того, як процес має обробляти конкретний сигнал.

# Сигнали переривання

## Сигнали переривання

Сигнали переривання - це сигнали, які надсилаються процесу, щоб перервати або завершити його. Два поширені сигнали переривання: SIGINT (сигнал 2) і SIGTERM (сигнал 15).

SIGINT зазвичай надсилається користувачем, коли він натискає Ctrl+C, тоді як SIGTERM зазвичай надсилається системою для запиту на завершення процесу.

## Надсилання сигналів за допомогою команди kill

Команда kill використовується для надсилання сигналів процесам

- Синтаксис команди kill такий: "kill [сигнал] [ідентифікатор процесу]".
- Наприклад, «kill -SIGINT 1234» надішле сигнал SIGINT процесу з ID 1234.

# Вступ до послідовних пристроїв і терміналів

# Послідовні пристрої та термінали

## Послідовні пристрої

Послідовні пристрої — це апаратні пристрої, які передають дані в послідовному форматі, по одному біту за раз.

Передача даних відбувається через послідовний порт із використанням певного протоколу та швидкості передачі даних.

## Термінали в Linux

Термінал — це інтерфейс, який використовується для взаємодії з комп'ютерною системою.

У Linux термінали можуть бути фізичними або віртуальними.

# Термінали в Linux

## Фізичні термінали

- Фізичний термінал — це пристрій, який забезпечує прямий доступ до комп'ютерної системи, наприклад до монітора та клавіатури.
- Вони зазвичай використовуються в серверних кімнатах або інших місцях, де необхідний прямий доступ

## Віртуальні термінали

- Віртуальний термінал — це програмний термінал, який емулює фізичний термінал.
- До них можна отримати доступ через графічний інтерфейс або за допомогою комбінації клавіш.

## Емулятори терміналу

- Емулятор терміналу — це програма, яка дозволяє користувачам отримувати доступ до віртуального терміналу в графічному інтерфейсі користувача.
- Приклади емуляторів терміналу в Linux включають GNOME Terminal, Konsole і xterm.



# Спілкування з іменованими каналами



# Введення в іменовані канали в Linux

Іменовані канали, також відомі як FIFO, є типом механізму міжпроцесного зв'язку (IPC), який використовується в Linux. Вони забезпечують спосіб для двох або більше процесів спілкуватися один з одним шляхом обміну даними через канал.

На відміну від неіменованих каналів, іменовані канали мають ім'я, пов'язане з ними, і до них можуть отримати доступ декілька процесів одночасно.

## Як працюють іменовані канали

- Іменовані канали подібні до звичайних файлів, але вони не мають вмісту, доки в них не будуть записані дані.
- Коли процес записує дані в іменований канал, вони зберігаються в буфері, поки інший процес не прочитає їх.
- Кілька процесів можуть читувати з одного і того ж іменованого каналу, але дані читаються в порядку FIFO.

## Переваги іменованих каналів

Іменовані канали мають кілька переваг перед іншими механізмами IPC, такими як сокети та черги повідомлень:

- Вони прості у створенні та використанні, не потребують спеціальних дозволів чи привілеїв, і до них може отримати доступ будь-який процес із відповідними дозволами.
- Іменовані канали також дуже ефективні, оскільки не потребують копіювання даних між процесами або простором ядра.

## Створення та використання іменованих каналів

Щоб створити іменований канал, скористайтеся командою `mkfifo`, після якої введіть назву каналу.

- Наприклад: `mkfifo mypipe`

Щоб записати дані в іменований канал, скористайтеся командою `echo`, за якою слідують дані та ім'я каналу

- Наприклад: `echo "Hello, world!" > mypipe`

Щоб прочитати дані з іменованого каналу, скористайтеся командою `cat`, після якої вкажіть назву каналу

- Наприклад: `cat mypipe`

Іменовані канали також можна використовувати в сценаріях оболонки для передачі даних між процесами.

# Автоматизація завдань за допомогою системного планувальника: Cron

# Знайомство з Cron

**Cron** — це планувальник завдань на основі часу в Unix-подібних операційних системах. Це дозволяє користувачам планувати автоматичне виконання завдань через певні проміжки часу.

## Структура завдання Cron

- Завдання cron складається з розкладу та команди.
- Розклад визначає час виконання команди.
- Команда - це завдання, яке потрібно виконати.

## Планування завдань за допомогою Cron

- Cron використовує спеціальний синтаксис для визначення розкладу завдання.
- Синтаксис складається з п'яти полів: хвилина, година, день місяця, місяць і день тижня.
- Кожне поле може містити діапазон значень, список значень або символ підстановки (\*) для відповідності будь-якому значенню

# Знайомство з Cron

## Використання Cron

- Щоб створити нове завдання cron, користувачі можуть редагувати файл crontab за допомогою команди crontab.
- Команда crontab дозволяє користувачам додавати, видаляти та змінювати завдання cron.
- Користувачі також можуть переглядати свої поточні завдання cron за допомогою команди crontab -l.

## Важливість Cron

- Cron — це потужний інструмент для автоматизації системних завдань і підтримки працездатності системи.
- Його можна використовувати для виконання широкого кола завдань, від простого резервного копіювання файлів до складних процедур обслуговування системи.
- Правильно налаштовані завдання cron можуть допомогти запобігти простою системи та забезпечити своєчасне виконання критичних завдань.



# Слідкуйте за системою: Основи моніторингу системи



# Моніторинг системи Linux

Моніторинг системи необхідний для підтримки працездатності системи.

Необхідно регулярно контролювати різні аспекти системи, щоб забезпечити оптимальну продуктивність

## Ключові аспекти для моніторингу:

- Завантаження системи: відстежуйте використання процесора та пам'яті системи.
- Використання пам'яті: відстежуйте обсяг пам'яті, який використовується системою та програмами.
- Використання диска: відстежуйте простір, який використовується файловою системою та пристроями зберігання.
- Активність у мережі: моніторинг мережевих з'єднань і трафіку.

## Навіщо нам моніторинг?

- Виявляйте потенційні проблеми, перш ніж вони стануть проблемами.
- Оптимізація продуктивності системи та використання ресурсів.
- Підвищення доступності та надійності системи.

# Інструменти моніторингу системи Linux

**top**: інструмент командного рядка, який відображає інформацію про систему та використання ресурсів.

- Основне використання: \$ top

**htop**: розширеніша версія top з додатковими функціями та зручним інтерфейсом

- Базове використання: \$ htop

**vmstat**: інструмент командного рядка, який повідомляє статистику віртуальної пам'яті.

- Основне використання: \$ vmstat

**iostat**: інструмент командного рядка, який повідомляє статистику введення/виведення для пристроїв зберігання даних

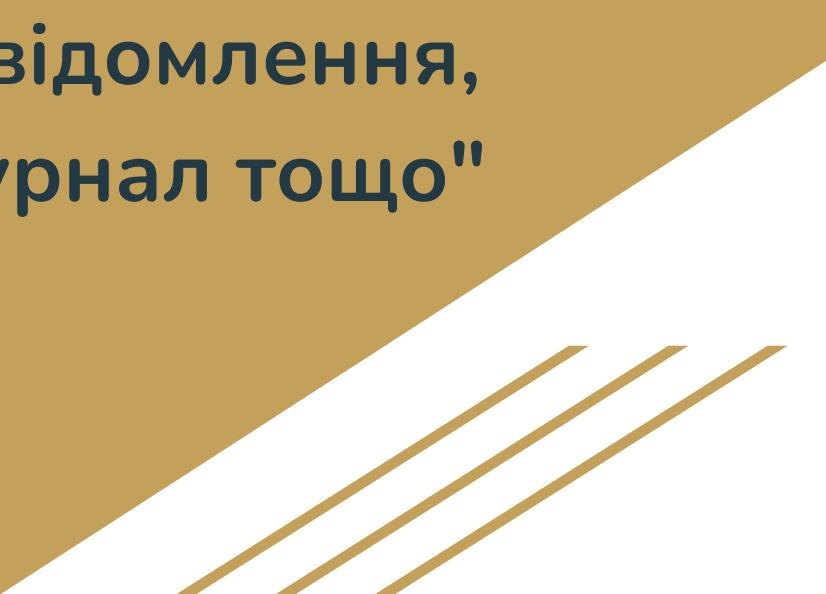
- Основне використання: \$ iostat

**netstat**: інструмент командного рядка, який відображає мережеві підключення та статистику

- Основне використання: \$ netstat



# Розуміння системних журналів: повідомлення, системний журнал тощо"



# Розуміння системних журналів із Syslog у Linux

**Системні журнали** мають вирішальне значення для розуміння поведінки системи Linux і вирішення проблем, які можуть виникнути.

**Syslog** — це стандартний механізм ведення журналів у системах Linux, який відповідає за збір і керування журналами.

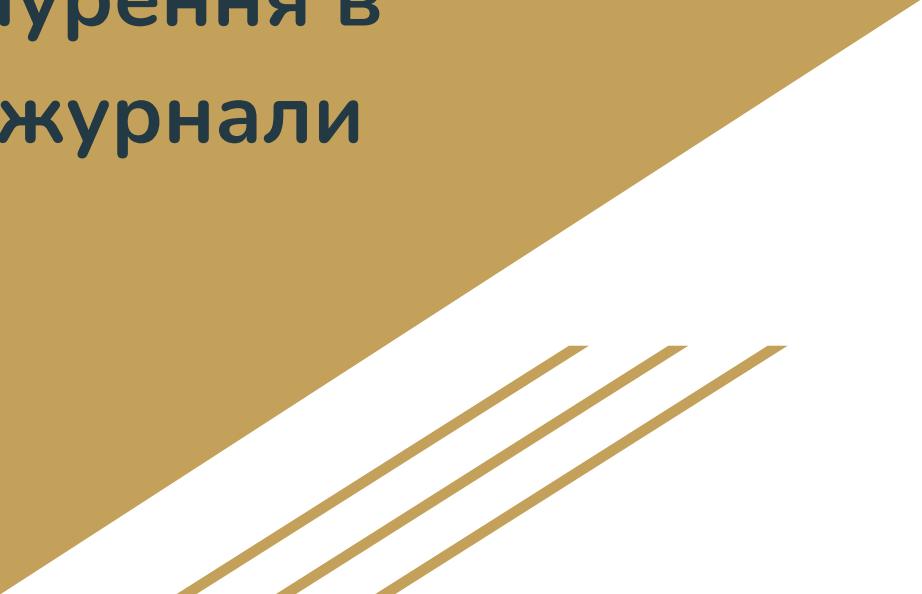
Syslog — це демон, який працює у фоновому режимі та збирає журнали з різних джерел, таких як системні служби та програми.

Журнали зберігаються в різних файлах, розташованих у каталозі `/var/log`

# Перегляд та інтерпретація журналів

Найпоширенішим інструментом для перегляду журналів є інструмент командного рядка «`tail`», який відображає кілька останніх рядків файлу.

Інші корисні інструменти включають «`grep`» для пошуку за певними ключовими словами, «`less`» для навігації великими журнальними файлами та «`journalctl`» для перегляду журналів, зібраних `systemd`.



Глибше занурення в  
конкретні журнали

# Ключові файли журналу

**/var/log/messages:** містить загальносистемні повідомлення, включаючи повідомлення ядра, службові повідомлення та інші системні повідомлення.

**/var/log/syslog:** містить системні повідомлення, створені syslog.

**/var/log/auth.log:** містить журнали, пов'язані з автентифікацією, такі як входи користувачів і події автентифікації системи.

**/var/log/dpkg.log:** містить журнали, пов'язані з керуванням пакунками, наприклад установку та видалення пакунків.

**/var/log/kern.log:** містить журнали ядра, включаючи апаратні помилки та проблеми з драйверами.

**/var/log/boot.log:** містить журнали, створені під час процесу завантаження, включаючи ініціалізацію обладнання та запуск служби.

# auth.log

- Розташований за адресою /var/log/auth.log
- Фіксує події, пов'язані з автентифікацією, наприклад спроби входу, спроби su та використання sudo
- Містить дату, час і ім'я хоста події, а також ім'я користувача та виконану дію
- Корисно для виявлення спроб неавторизованого доступу та розуміння активності користувача в системі

# dpkg.log

- Розташований за адресою `/var/log/dpkg.log`
- Записує події встановлення та видалення пакунків Debian за допомогою інструменту dpkg
- Містить дату, час і назву хоста події, а також назву пакета, версію та виконану дію
- Корисно для відстеження змін пакетів, усунення проблем зі встановленням і розуміння системних оновлень і змін

# syslog

- Розташований у `/var/log/syslog`
- Зберігає широкий спектр системних подій, включаючи повідомлення ядра, повідомлення демона тощо
- Містить детальну інформацію про роботу системи та помилки
- Корисно для налагодження системних проблем, аналізу продуктивності системи та розуміння поведінки системи

# messages

- Розташований у `/var/log/messages`
- Зберігає широкий спектр системних подій, включаючи повідомлення ядра, повідомлення демона тощо
- Містить детальну інформацію про роботу системи та помилки
- Корисно для налагодження системних проблем, аналізу продуктивності системи та розуміння поведінки системи.

# kern.log

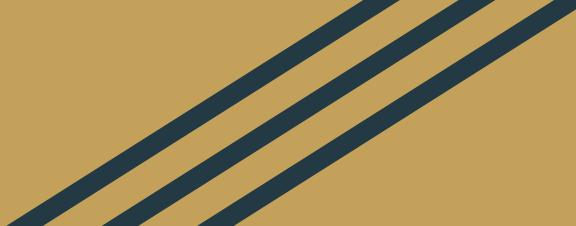
- Розташований за адресою /var/log/kern.log
- Зберігає повідомлення та події, пов'язані з ядром, наприклад апаратні помилки, системні збої та завантаження й вивантаження модулів ядра.
- Містить детальну інформацію про діяльність і помилки на рівні ядра системи.
- Корисно для налагодження системних проблем, аналізу продуктивності системи та розуміння поведінки системи, особливо пов'язаної з апаратним забезпеченням і процесами на рівні ядра.

# boot.log

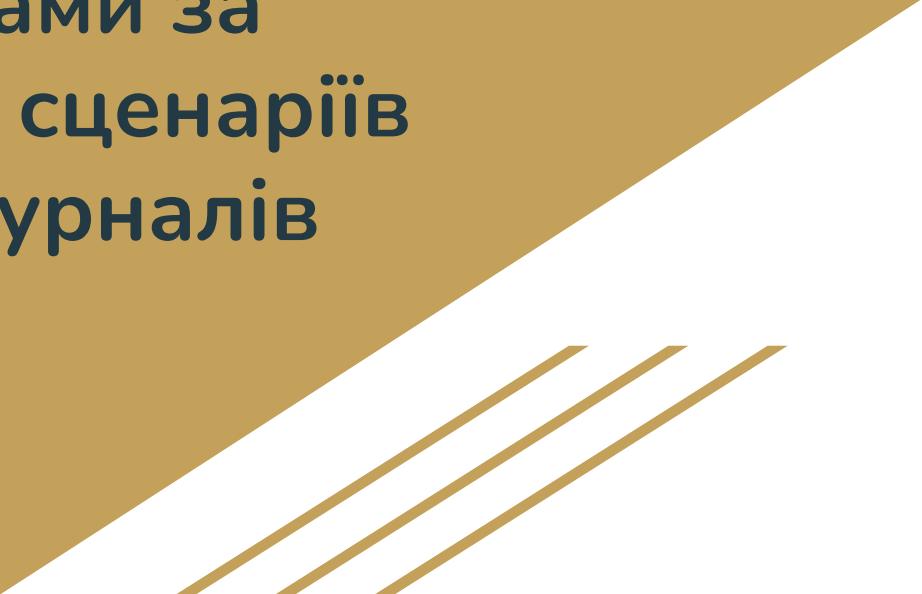
- Розташований за адресою /var/log/boot.log
- Містить інформацію про процес завантаження системи, включаючи повідомлення від завантажувача, ядра та завантажувальних сценаріїв
- Корисно для діагностики проблем завантаження та розуміння послідовності завантаження системи.

# Керування журналами

- Важливо правильно керувати журналами, щоб уникнути заповнення дискового простору та полегшити пошук і аналіз журналів
- Такі інструменти, як logrotate, можуть допомогти керувати журналами, регулярно обертаючи та стискаючи файли журналів
- Керування журналами має вирішальне значення для підтримки працездатності системи, діагностики проблем і підвищення продуктивності системи



# Ефективне керування журналами за допомогою сценаріїв ротації журналів



# Ротація журналів

Ротація журналів — це процес регулярного архівування та видалення файлів журналів для управління дисковим простором і запобігання тому, щоб журнали ставали занадто великими.

У Linux ротацію журналу зазвичай керують за допомогою утиліти logrotate

## Важливість ротації журналів:

- Файли журналу можуть бути дуже великими та займати багато місця на диску.
- Великі файли журналів також можуть ускладнити пошук важливої інформації.
- Регулярна ротація журналів допомагає зберігати файли журналів у прийнятному розмірі та полегшує пошук важливої інформації.
- Ротація журналів також важлива для цілей безпеки та відповідності, оскільки вона гарантує, що журнали не будуть втрачені чи змінені.

# Утиліта Logrotate

- logrotate — це утиліта командного рядка, яка використовується для керування файлами журналів у Linux.
- Це дозволяє вказати, які файли журналів ротувати, як часто їх ротувати та де зберігати заархівовані журнали.
- Logrotate також може стискати заархівовані журнали для економії дискового простору.

# Базова конфігурація та використання Logrotate

- Файл конфігурації logrotate знаходиться за адресою /etc/logrotate.conf.
- Щоб налаштувати ротацію журналу для певного файлу журналу, створіть новий файл конфігурації в каталозі /etc/logrotate.d/.
- У файлі конфігурації вкажіть файл журналу для ротації, частоту його ротації та інші параметри.
- Щоб вручну запустити logrotate, скористайтеся командою "logrotate <config-file>".
- За замовчуванням logrotate запускається щодня через завдання cron.



# Представляємо Auditd: моніторинг подій на детальному рівні



# Введення в Auditd

**Auditd** — це компонент простору користувача системи аудиту Linux, який відстежує та записує діяльність системи.

Він забезпечує детальний рівень контролю над поведінкою системи та відстежує дії користувачів, системні виклики та інші події.

Auditd — важливий компонент для цілей безпеки системи, відповідності та налагодження.

## Роль Auditd

- Auditd надає комплексну систему журналювання, яка відстежує всі дії системи.
- Він відстежує активність користувачів, системні виклики та доступ до файлів, серед іншого.
- Auditd записує дані у двійковий формат файлу, який можна легко шукати та аналізувати.

# Важливість Auditd

- Auditd необхідний для безпеки системи. Він допомагає виявляти загрози безпеці та реагувати на них шляхом визначення спроб несанкціонованого доступу або підозрілих дій.
- Auditd корисний для цілей відповідності, оскільки він може допомогти організаціям виконати нормативні вимоги та переконатися, що користувачі дотримуються політики компанії.
- Auditd є цінним для цілей налагодження, оскільки він надає докладні журнали активності системи, які можуть допомогти виявити системні проблеми та швидко їх вирішити.

## Ресурси для додаткового вивчення:

«Посібник з адміністрування Linux», Еві Немет, Гарт Снайдер, Трент Р. Хайн і Бен Уейлі (в системі LMS)

# Питання та відповіді