Архітектура ОС Linux

Глибоке занурення у внутрішній устрій: **FS, Процеси та Пам'ять**

Файлова система Процеси Пам'ять

Що таке Linux?

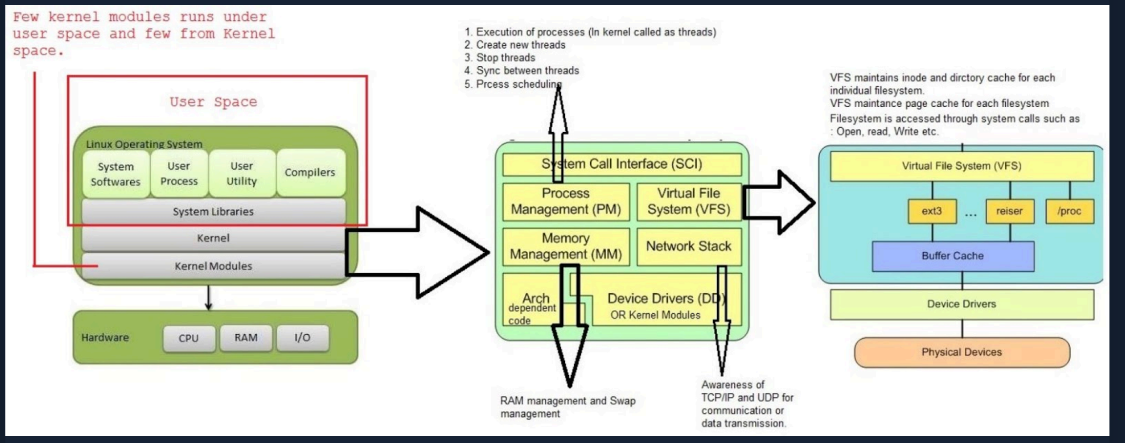
Монолітне ядро

Весь системний функціонал зосереджений в одному ядрі

"Everything is a file"

Уніфікований підхід до всіх ресурсів системи

Перевага: Спрощення взаємодії з пристроями та даними через єдиний інтерфейс



Рівні абстракції файлової системи

VFS — Virtual File System

Прошарок абстракції між ядром та різними файловими системами

Що робить VFS?

- Єдиний інтерфейс для всіх FS
- Підтримка різних типів (ext4, xfs, nfs)
- Прозора взаємодія з ядра

Результат: Програми не потребують знати про конкретний тип FS

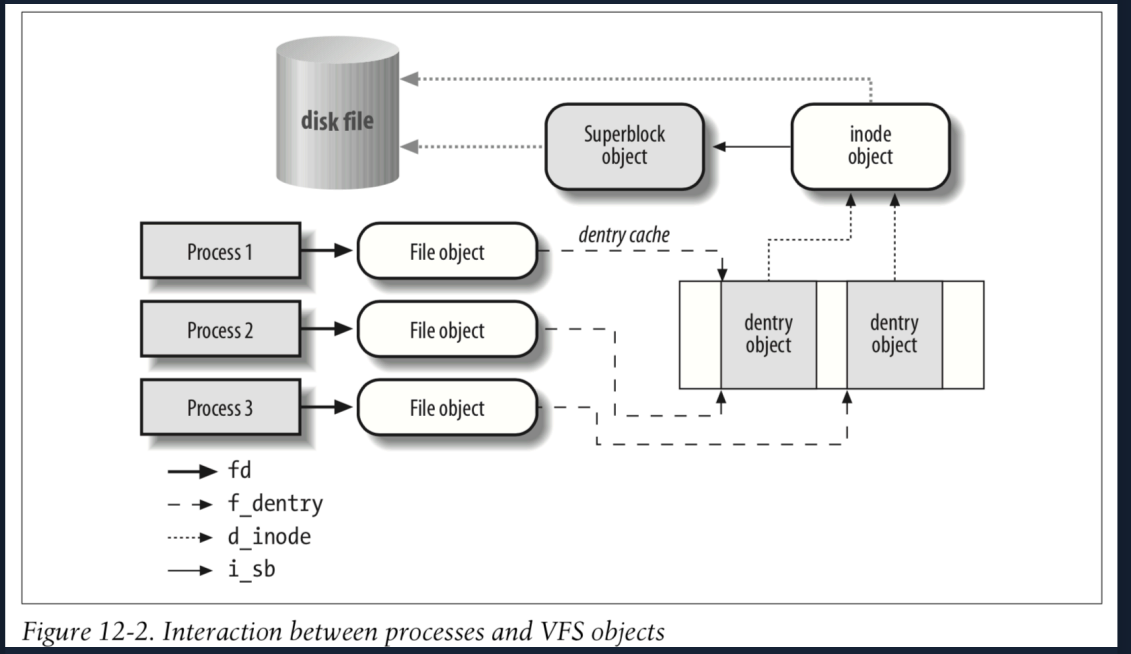


Figure 12-2. Interaction between processes and VFS objects

Що таке Інода (Inode)?

Індексний дескриптор

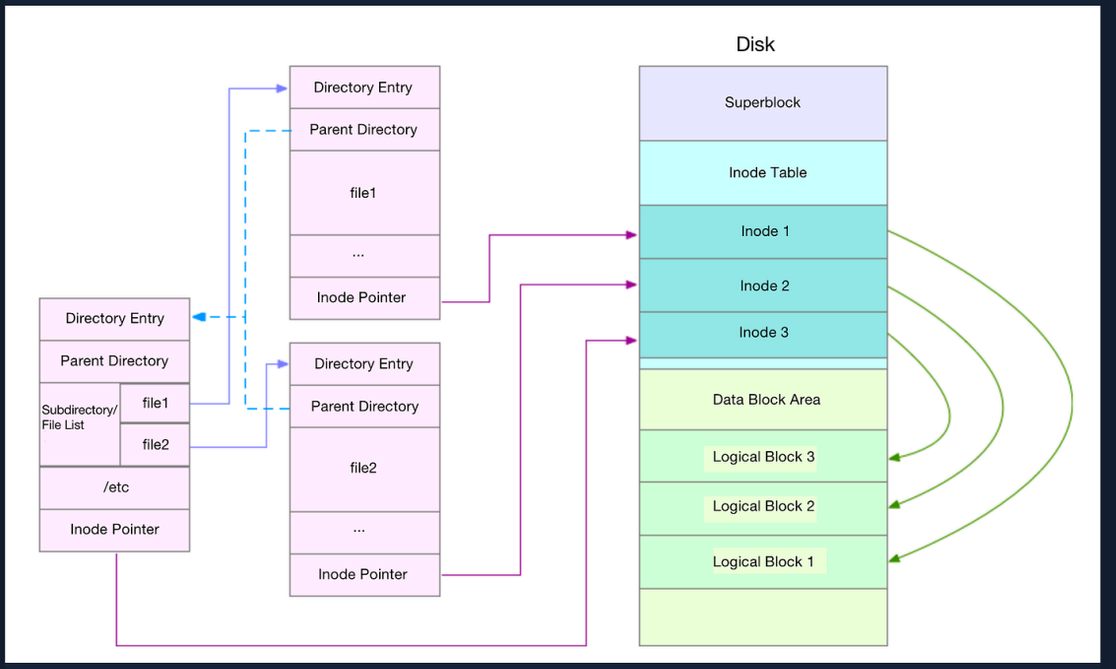
Структура даних, що ідентифікує файл у файловій системі

Що містить Інода?

- Розмір файлу
- Власник (UID)
- Права доступу
- Мітки часу
- Група (GID)
- Посилання на блоки

ВАЖЛИВО!

Інода **НЕ містить** імені файлу. Ім'я зберігається в директорії.



Вміст Іноди

Інода зберігає наступну метадану про файл:

Тип файлу

Звичайний файл, директорія, символьне посилання, пристрій

Права доступу

rwx для власника, групи та інших (mode bits)

UID та GID

Ідентифікатори власника та групи файлу

Посилання на блоки

Прямі та непрямі вказівники на блоки даних на диску

Нагадування: Ім'я файлу НЕ зберігається в іноді

Hard Links (Жорсткі посилання)

Додаткове ім'я

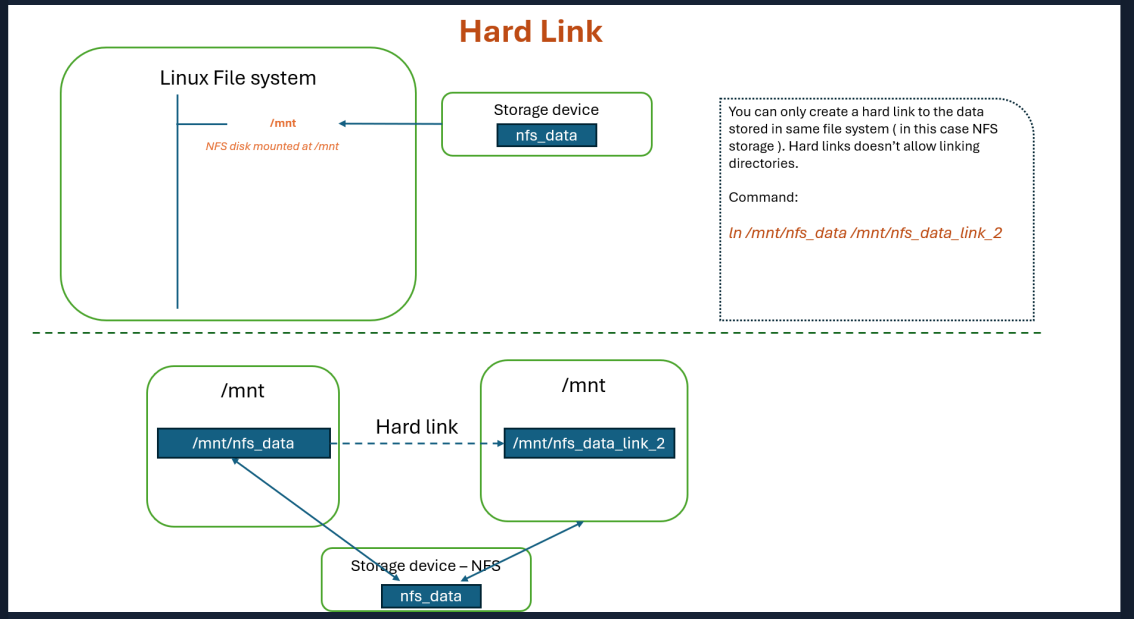
Ще одне ім'я для **тієї ж іноди**

Безпека видалення

Видалення одного імені не видаляє дані, поки існує хоча б одне посилання

Особливості

- Однаковий inode для всіх посилань
- Працюють тільки в межах **одної файлової системи**
- Не можуть посилатися на директорії



Soft Links (Символьні посилання)

Окремий файл

Це **окремий файл** із власним inode

Шлях до іншого файлу

Містить **шлях** до цільового файлу

Як ярлик у Windows

Аналог ярлика Windows — вказівник на інше місце

УВАГА!

Якщо оригінал видалити — посилання стане **«битим»** (broken link)

Різні іноди

- Символьне посилання має **власний inode**
- Цільовий файл має **інший inode**

Гнучкість

- Працюють **між різними файловими системами**
- Можливі посилання на директорії

Порівняння Links

Hard Link

Одна інода

Всі посилання вказують на **один і той самий inode**

Той самий пристрій

Працюють тільки в межах **одної файлової системи**

Захист від видалення

Дані не видаляються, поки існує **хоча б одне посилання**

Soft Link

Різні іноди

Посилання має **власний inode**, відмінний від цільового файлу

Шлях до файлу

Зберігає **шлях** до цільового файлу

Будь-який пристрій

Працюють **між різними файловими системами**

Що таке Mounting (Монтування)?

Приєднання FS

Процес **приєднання файлової системи** до дерева каталогів

Зовнішні пристрої

Жорсткі диски, SSD, USB-накопичувачі, мережеві ресурси

Єдине дерево

Усі пристрої стають частиною **єдиного файлового дерева** Linux

Команда mount

Базовий синтаксис:

```
mount /dev/sdb1 /mnt/data
```

Приклад

- /dev/sda1 → / (root)
- /dev/sdb1 → /home
- /dev/sdc1 → /mnt/backup

Точка монтування (Mount Point)

Директорія

Кожна точка монтування — це **директорія** у файловому дереві

Вхід у FS

Через цю директорію стає **доступним вміст** нової файлової системи

Команда mount

```
mount [options] device directory
```

Приклади точок монтування

- **/** — коренева директорія
- **/home** — домашні каталоги
- **/mnt** — тимчасове монтування
- **/media** — змінні носії

Важливо

Точка монтування повинна бути **існуючою директорією**. При монтуванні вона стає вхідною точкою до файлової системи.

Файл /etc/fstab

Конфігураційний файл

Зберігає **правила монтування** файлових систем

Автоматичне монтування

Виконується **при старті системи** автоматично

Переваги

- Не потрібно вводити команди вручну
- Гарантує монтування при завантаженні
- Узагальнює конфігурацію системи

Структура файлу

Device	Mount Point	FS Type	Options	Dump	Pass

```
/dev/sda1  /  ext4  defaults  0  1
/dev/sdb1  /home  ext4  defaults  0  2
/dev/sdc1  /mnt/data  ext4  defaults  0  0
```

Питання по Файловій системі

VFS

Що таке Virtual File System і як вона працює?

Inodes

Які дані містить інода і чому ім'я файлу не зберігається в ній?

Hard Links

Як працюють жорсткі посилання і чому вони не працюють між різними FS?

Soft Links

Як символьні посилання відрізняються від жорстких?

Mounting

Що таке монтування і як працюють точки монтування?

/etc/fstab

Для чого потрібен файл /etc/fstab і як він налаштовується?

Що таке процес?

Програма у виконанні

Активний екземпляр програми, що **виконується** на CPU

Адресний простір

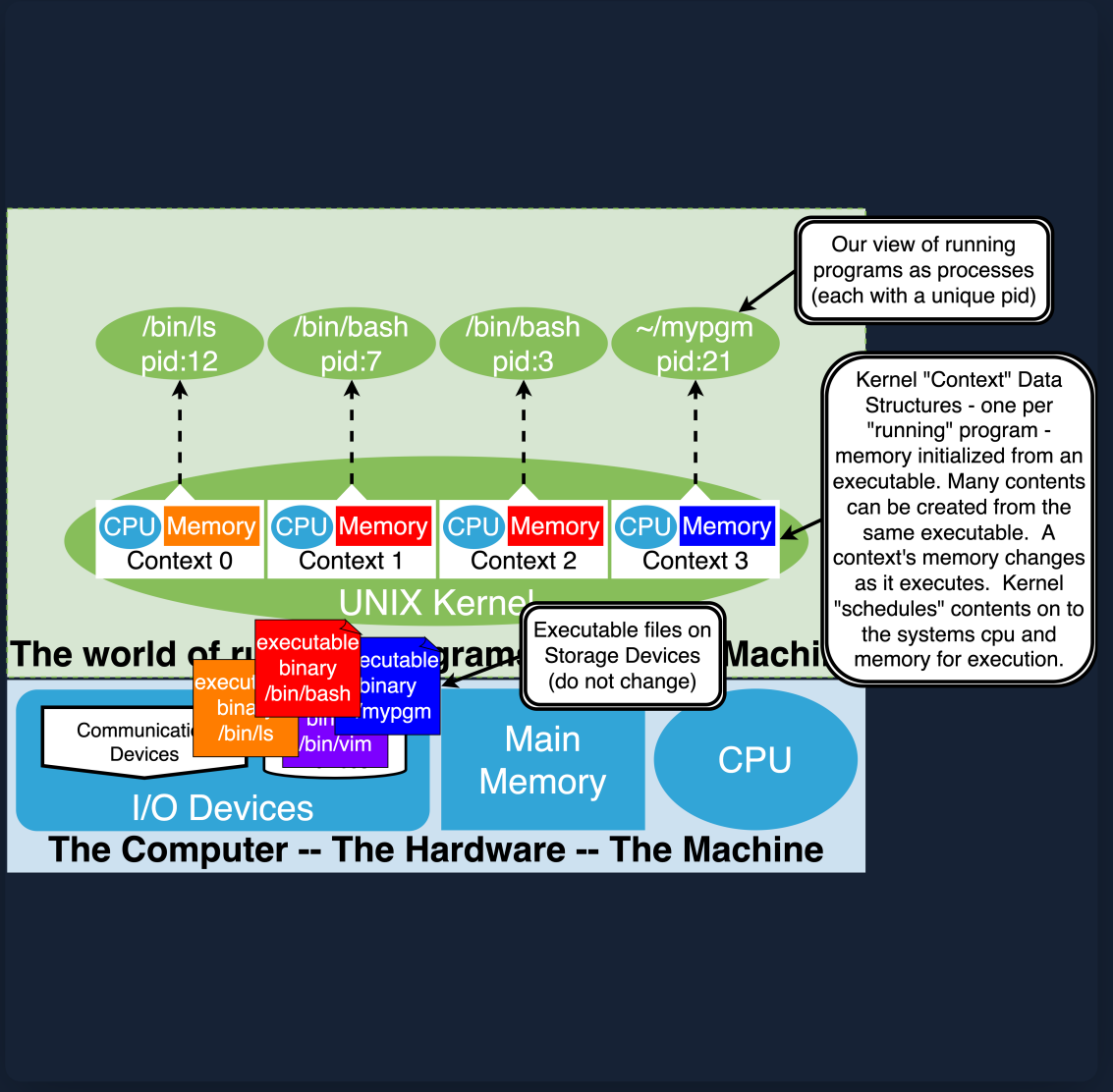
Кожен процес має **власну віртуальну пам'ять**, ізольовану від інших

Унікальний ідентифікатор

PID (Process ID) — унікальний номер для кожного процесу

Системні ресурси

Кожен процес отримує доступ до **CPU, пам'яті, I/O** та інших ресурсів



Життєвий цикл процесу

Створення (Creation)

Процес створюється системним викликом **fork()**, який дублює батьківський процес

Виконання (Execution)

Процес виконує інструкції на CPU або очікує в черзі на виконання

Очікування (Waiting)

Процес чекає на подію, ресурс або завершення I/O операції

Завершення (Termination)

Процес завершує роботу, повертає код виходу та звільняє ресурси

Стани процесів (States)

R — **Running/Runnable**

Виконується або готовий до виконання на CPU

S — **Interruptible Sleep**

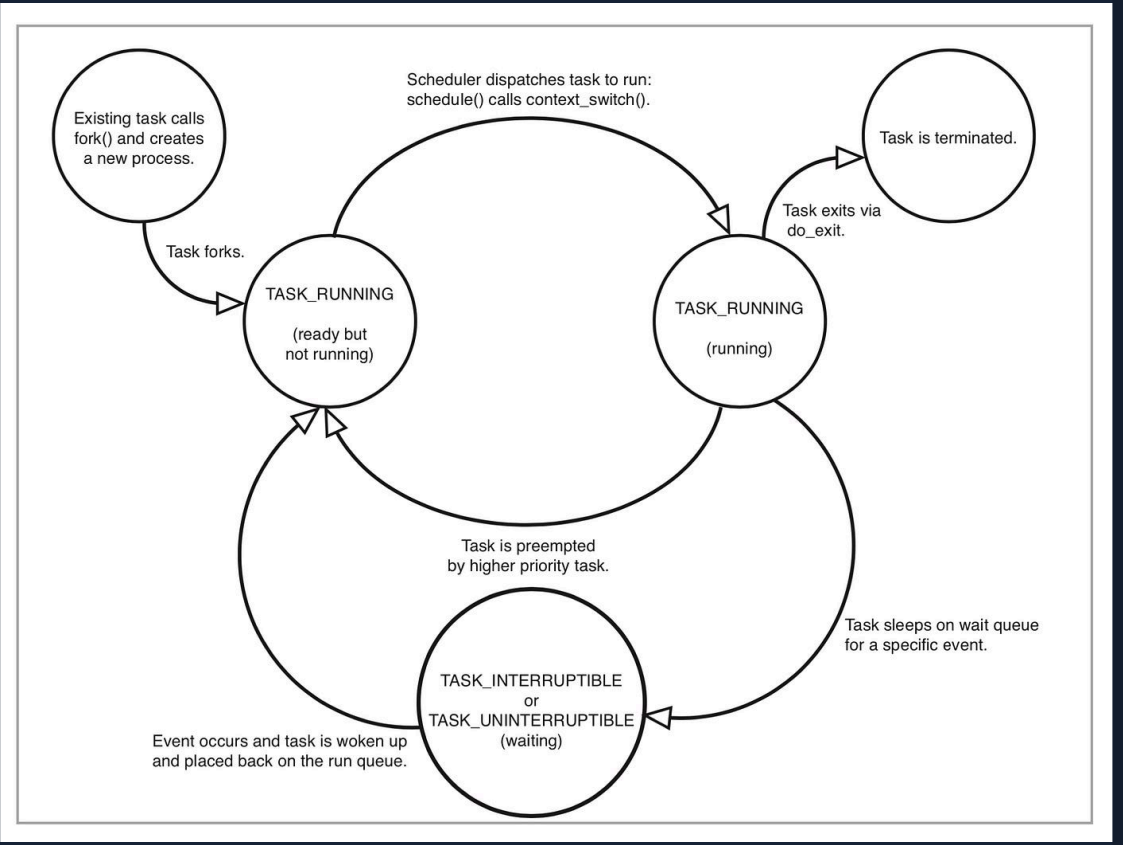
Чекає подію, може бути перерваний сигналом

D — **Uninterruptible Sleep**

Чекає на I/O операцію, не можна перервати

Z — **Zombie**

Завершений, але чекає код виходу від батька



Дочірні процеси (Child Processes)

Батько і дитина

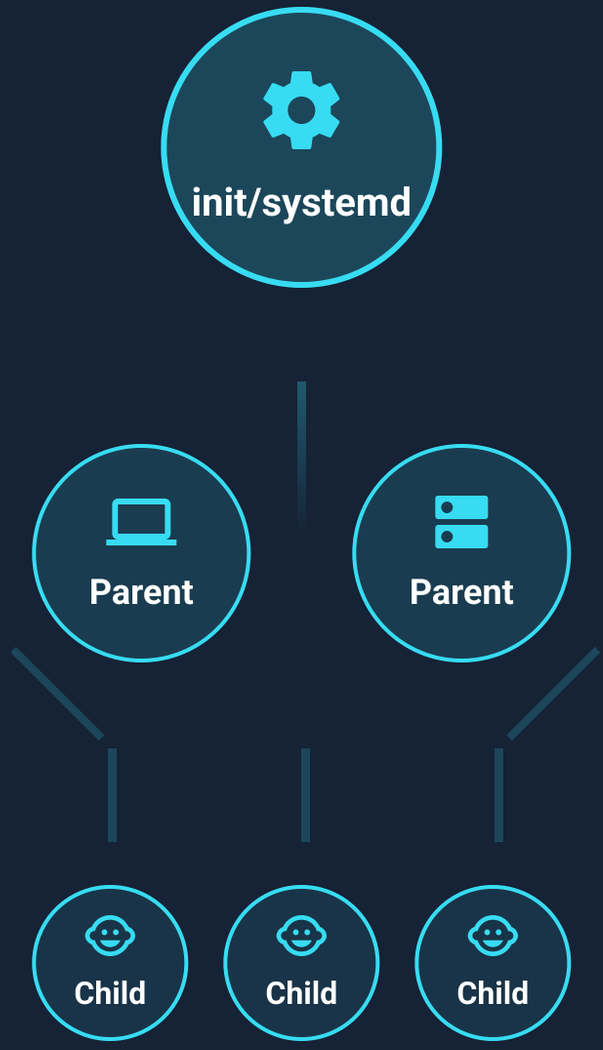
Кожен процес (крім **init/systemd**) має **батьківський процес** (parent process)

Створення

Дочірні процеси створюються через системний виклик **fork()**

Ієрархія

Система процесів утворює **дерево** з init/systemd у корені



Fork та Exec

fork()

Копіює **поточний процес**

- Створює **ідентичний дочірній процес**
- Дочірній процес отримує **новий PID**
- Обидва процеси продовжують виконання з **однакового місця**

exec()

Замінює **код поточного процесу**

- Заміщує **весь адресний простір**
- PID залишається **незмінним**
- Запускає **нову програму** в тому ж процесі

fork() + exec() — стандартна схема для запуску нових програм у Linux

Процеси vs Потоки (Threads)

Процеси

- Має **власну пам'ять**
- Ізольовані один від одного
- Захищені від змін іншими процесами
- Кожен процес має свій PID
- Міжпроцесна комунікація складніша

Потоки (Threads)

- Розділяють **спільну пам'ять**
- Пришвидшує комунікацію
- Прямий доступ до спільних даних
- Потребує **синхронізації**
- Потоки в одному процесі

Процеси забезпечують ізоляцію, **потоки** — ефективну комунікацію

Системні виклики (API)

Інтерфейс з ядром

Системні виклики (syscalls) — це міст між програмою та ядром Linux

Приклади викликів

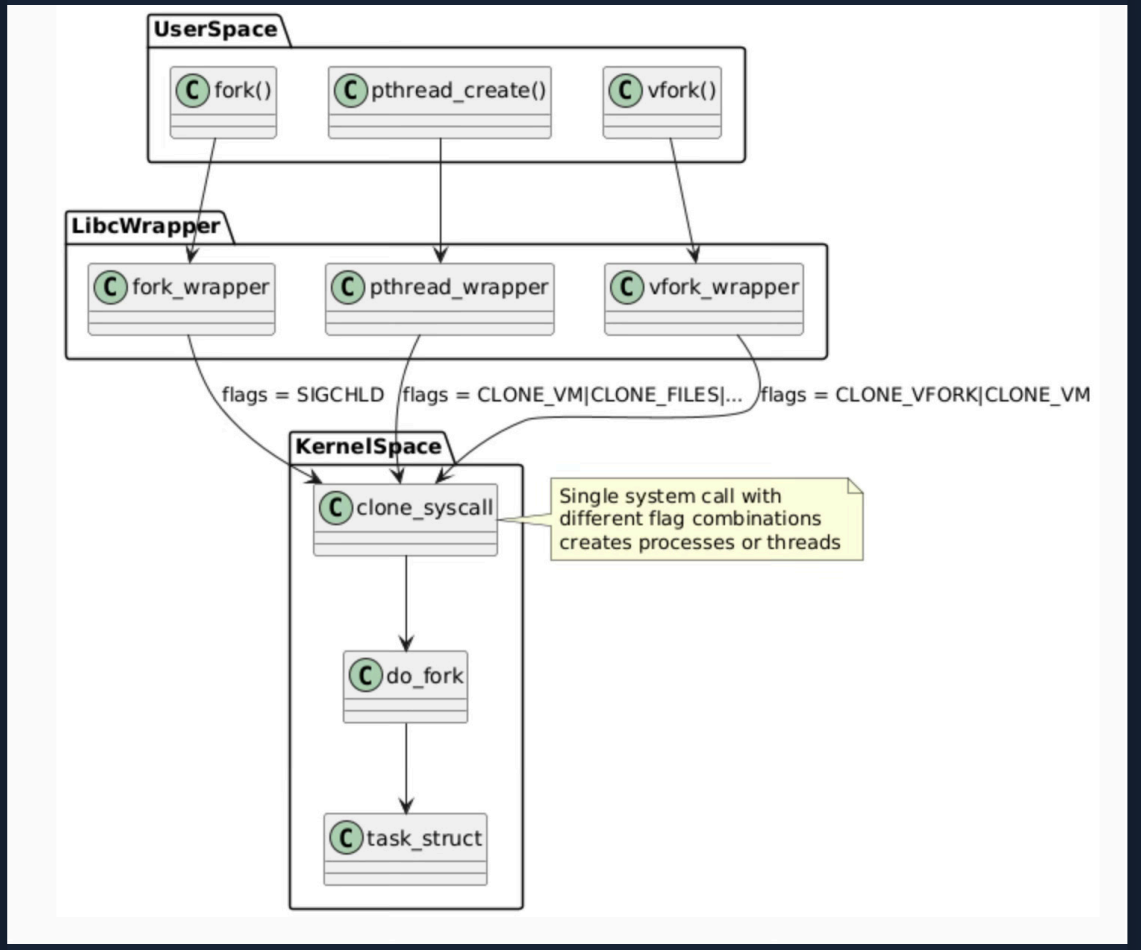
`open` Відкриття файлу

`read` Читання даних

`write` Запис даних

`kill` Завершення процесу

Захищений доступ до апаратних ресурсів



Планувальник процесів (Scheduler)

Роль планувальника

Частина ядра, що **розподіляє CPU** між процесами

Основні функції

- Вибирає **наступний процес** для виконання
- Визначає **часовий квант** виконання
- Балансує навантаження між ядрами

CFS — Completely Fair Scheduler

Принцип роботи CFS

1. **Червоне дерево**
   Червоно-чорне дерево для вибору процесів
2. **Віртуальний час**
   Віртуальний час виконання для справедливості
3. **Динамічний пріоритет**
   Адаптація пріоритетів залежно від поведінки

Швидкість | **Справедливість** | **Гнучкість**

Пріоритети: Nice та Priority

Nice Value Scale

↑ Чим менше число — вищий пріоритет

-20	-15	-10	-5	0	5	10	15	19

Найвищий пріоритет

Найнижчий пріоритет

Високий пріоритет

Nice **-20 ... -1**

Системні процеси, критичні задачі

Нормальний пріоритет

Nice **0**

Звичайні користувацькі програми

Низький пріоритет

Nice **1 ... 19**

Фонові задачі, обробка даних

Моніторинг процесів

top

Реальний час
Динамічне оновлення

Використання ресурсів
CPU, RAM, час виконання

Сортування
За використанням CPU

```
top -d 1
```

htop

Кольоровий інтерфейс
Зручне сприйняття

Інтерактивний
Керування клавішею

Дерево процесів
Відображення ієрархії

```
htop
```

ps aux

Знімок процесів
Статичний список

Повна інформація
PID, PPID, CPU, MEM

Фільтрація
Через grep або sort

```
ps aux | grep nginx
```

Ключові метрики: PID, CPU%, MEM%, TIME, COMMAND

Управління пам'яттю в Linux

Роль ядра

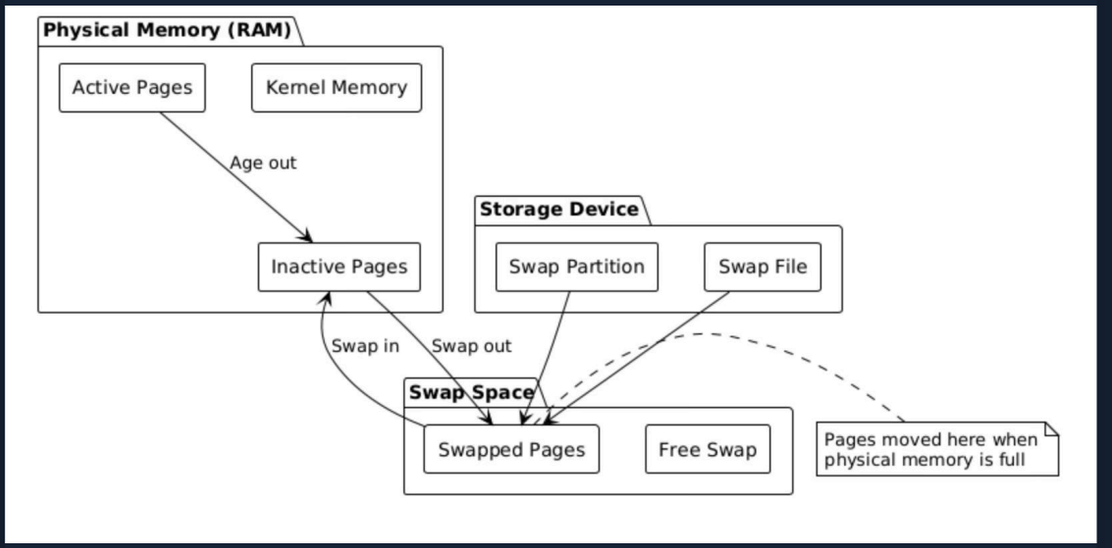
Ядро **керує фізичною RAM** та розподіляє її між процесами

Віртуалізація

Кожен процес отримує **віртуальний адресний простір**, ізольований від інших

Трансляція адрес

MMU транслює віртуальні адреси у **фізичні сторінки** пам'яті



Фізична пам'ять (Physical RAM)

Фізичні модулі

Реальні планки пам'яті встановлені в слоти материнської плати (DDR4, DDR5)

Обмежений ресурс

Має **фіксовану ємність**, що визначається кількістю та об'ємом модулів

Швидкість

Найшвидший тип пам'яті, що доступний для процесів

Характеристики RAM

Тип	**DDR4/DDR5**
Швидкість	**2400-6000 MT/s**
Час доступу	**10-15 нс**
Об'єм на модуль	**4GB - 64GB**

Логічна (Віртуальна) пам'ять

Ілюзія безмежності

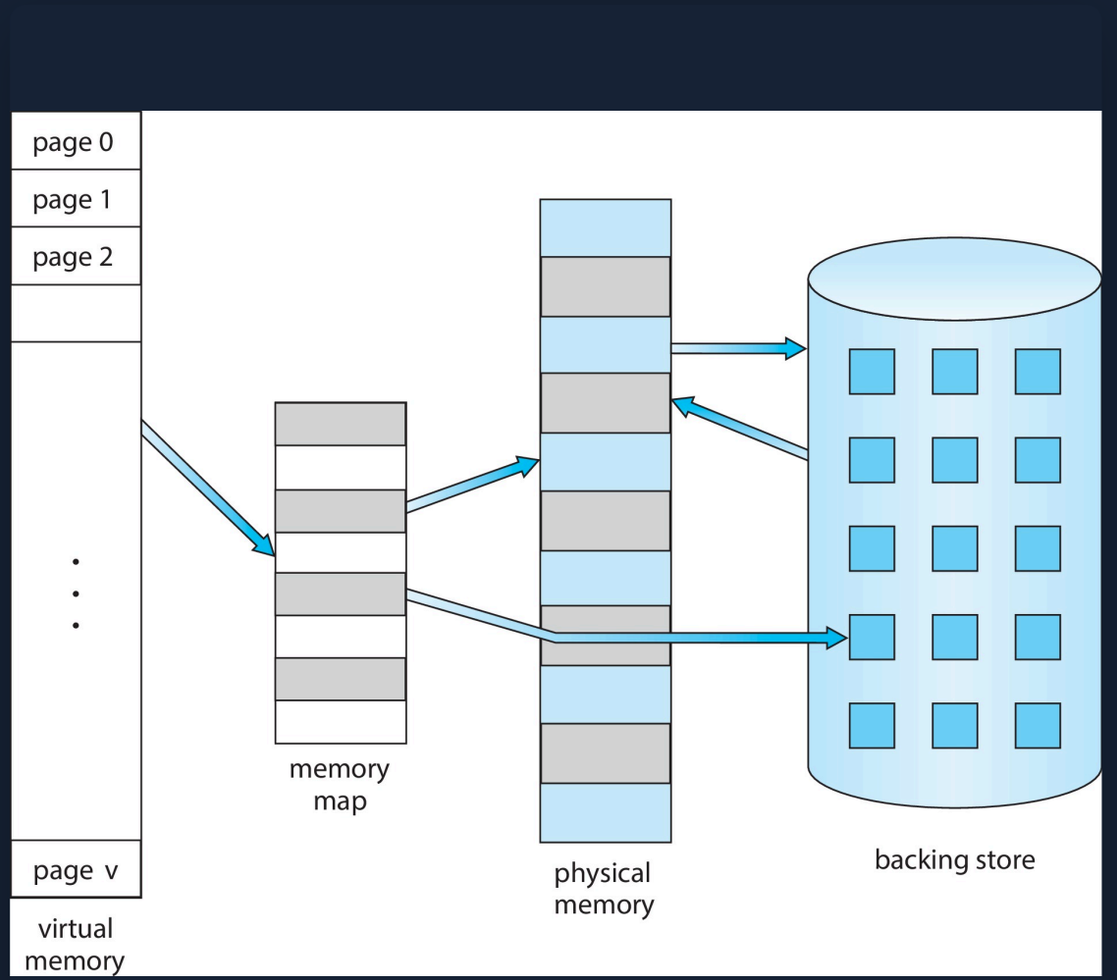
Кожен процес **"думає"**, що має доступ до всього обсягу пам'яті

Ізоляція

Ізолює процеси один від одного, забезпечуючи стабільність системи

Власний простір

Кожен процес має **унікальний віртуальний адресний простір**



Сторінкова організація (Paging)

Сторінки пам'яті

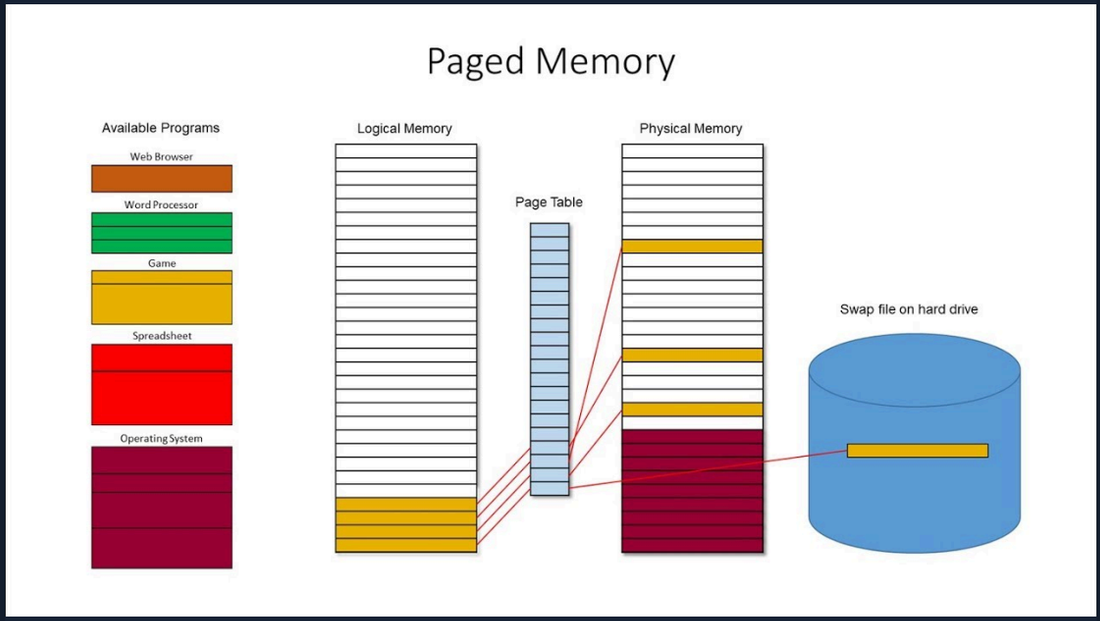
Пам'ять ділиться на **фіксовані блоки — "сторінки"** зазвичай по **4KB**

MMU трансляція

Memory Management Unit транслює віртуальні адреси у фізичні

Таблиці сторінок

Кожен процес має **свою таблицю сторінок** для адресації



Що таке Swap (Підкачка)?

Область на диску

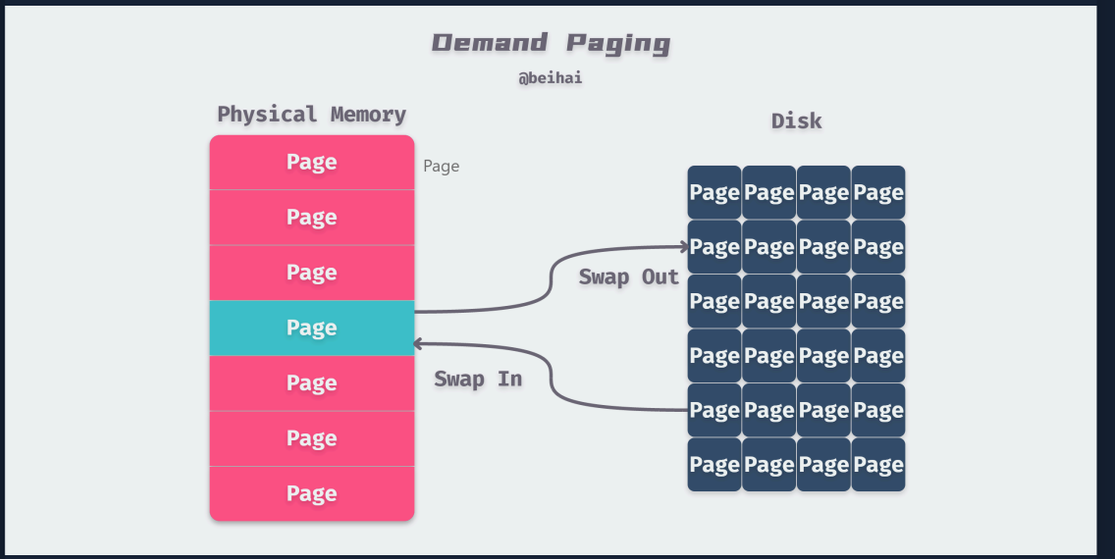
Ділянка на диску, що використовується як розширення RAM

Коли RAM переповнена

Активується, коли **фізична RAM вичерпана** або майже повна

Скидає рідко використовувані

Ядро переміщує **сторінки, що рідко використовуються** в swap



Плюси та мінуси Swap

Плюси

Запобігає падінню

Уникає падіння програм при нестачі RAM

Розширення пам'яті

Додає **віртуальну пам'ять** без фізичних модулів

Гнучкість

Дозволяє запускати **більші програми**, ніж доступна RAM

Мінуси

Повільна швидкість

Диск набагато повільніший за RAM (1000x)

Уповільнення системи

Істотно погіршує продуктивність при активному використанні

Навантаження на диск

Збільшує **I/O навантаження** та зношування диска

OOM Killer (Out of Memory)

Механізм захисту ядра

Автоматичний захист системи від повного виснаження пам'яті

Коли пам'ять закінчилася

Активується, коли **весь RAM та swap** повністю вичерпані

"Вбиває" процес

Припиняє **найменш пріоритетний процес**, щоб врятувати систему

Критерії вибору жертви

1. Оцінка **badness score** для кожного процесу
2. Врахування **використання пам'яті**
3. Вплив **nice значення** та OOMScoreAdjust

Уникнення OOM: Моніторинг пам'яті та коректне налаштування

Підсумок

Файлова система

Файли як **іноди**, VFS, жорсткі та символьні посилання

Процеси

Процеси як **живі сутності**, fork/exec, планувальник

Пам'ять

Магія **віртуальної пам'яті**, сторінки, swap, OOM killer

Питання?