



Операційні системи

Лекція 13

Реалізація файлових систем



План лекції

- FAT
- ufs
- ext2fs
- ext3fs
- /proc
- VFS
- NTFS

Фізична організація FAT

| Завантажувальний сектор (512 Б) | | | |
|---------------------------------|---|-----|---|
| FAT 1 | | | |
| 0 | 1 | 2 | 3 |
| 4 | 5 | ... | |
| (Елементи FAT) | | | |
| FAT 2 (копія) | | | |
| Кореневий каталог (16 кБ) | | | |
| : | | | |
| : | | | |
| Дані | | | |
| 0 | 1 | 2 | 3 |
| 4 | 5 | ... | |

- Логічний розділ, відформатований під файлову систему FAT, містить такі розділи:
 - **Завантажувальний сектор** (*boot sector*) містить програму початкового завантаження ОС
 - **Основна копія FAT**
 - **Резервна копія FAT**
 - **Кореневий каталог** (*root directory*) займає фіксовану ділянку у 32 сектора (16 кБ), що дозволяє зберігати 512 записів про файли і каталоги (кожний запис – 32 Б)
 - **Область даних** призначена для розміщення усіх файлів і усіх каталогів, крім кореневого каталогу



Особливості FAT

- Елемент (індексний покажчик) FAT може мати такі значення:
 - Кластер вільний
 - Кластер використовується (номер наступного кластера)
 - Останній кластер файлу
 - Дефектний кластер
 - Резервний
- Розрядність елементів
 - FAT12 – 12 біт – максимум 4096 кластерів
 - FAT16 – 16 біт – максимум 65536 кластерів
 - Максимальний розмір розділу 4 ГБ (кластер 64 кБ)
 - FAT32 – 32 біт – максимум $2^{32} = 4294967296$ кластерів
- Кожний запис у каталозі – 32 Б
 - Ім'я файлу – 11 Б у форматі 8.3
 - Довге ім'я файлу – до 255 двохбайтних символів Unicode
 - Довге ім'я поміщається порціями по 13 символів у записи, що безпосередньо йдуть за основним записом каталогу (кожний такий запис містить ще 6 Б службової інформації)

Фізична організація ufs (UNIX File System)



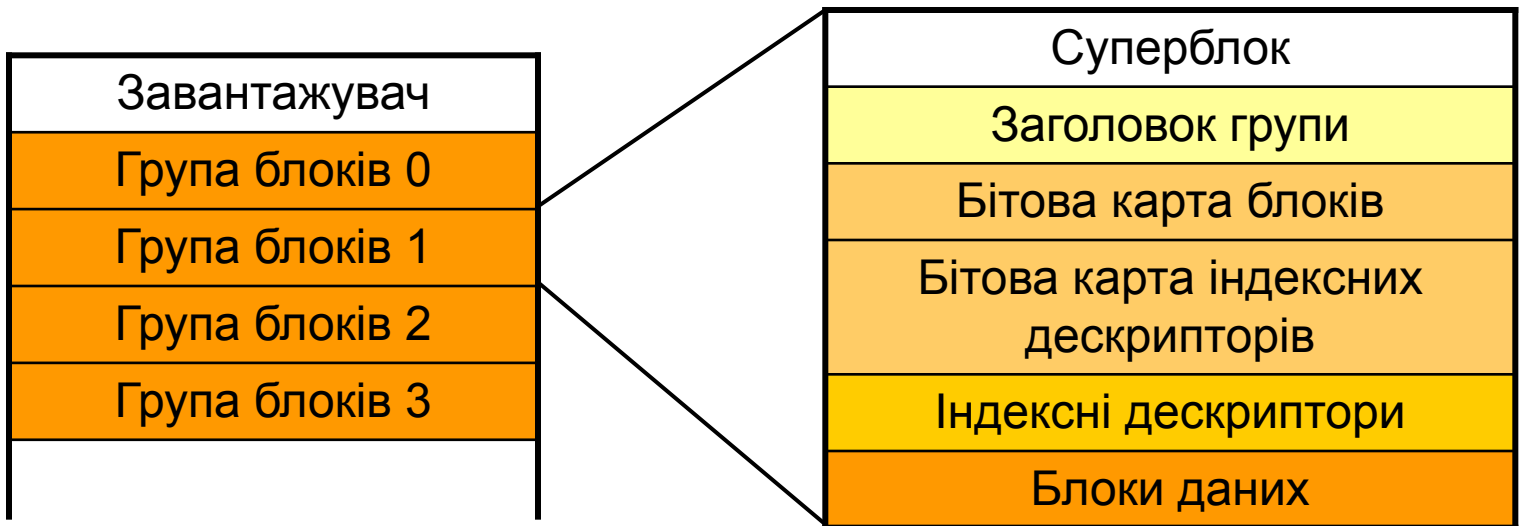
- Розділ містить **завантажувальний блок**, після якого кілька разів повторюється послідовність:
 - **Суперблок** – містить загальну інформацію про файлову систему
 - Розмір файлової системи
 - Розмір області індексних дескрипторів
 - Число індексних дескрипторів
 - Список вільних блоків
 - Список вільних індексних дескрипторів
 - **Блок групи циліндрів** – описує кількість індексних дескрипторів і блоків даних, що розташовані у цій групі циліндрів диска
 - **Область індексних дескрипторів**
 - Дескриптори розташовані за їхніми номерами
 - **Область даних**
 - Містить звичайні файли і каталоги (у тому числі кореневий каталог)
 - Спеціальні файли в області даних не відображені



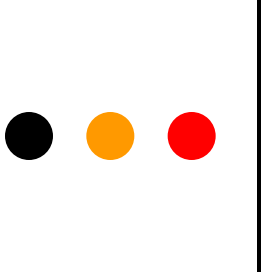
Особливості ufs

- Індексний дескриптор містить
 - Ідентифікатор власника файлу
 - Тип файлу
 - Права доступу до файлу
 - Час останньої модифікації файлу, час останнього звернення до файлу, час останньої модифікації індексного дескриптора
 - Кількість посилань на цей дескриптор (імен файлу)
 - Адресна інформація (була розглянута нами раніше)
 - Розмір файлу в байтах
- Запис про файл у каталозі містить усього два поля: ім'я файлу і номер індексного дескриптора
 - Ім'я файлу в ufs може мати довжину до 255 символів (код ASCII – по одному байту на символ)
- Розмір індексного дескриптора – 64 кБ
- Розмір блоку – 4 або 8 кБ

Фізична організація ext2fs (Linux)

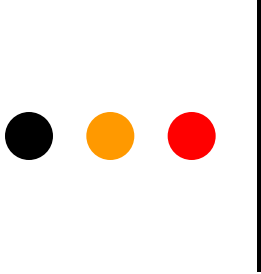


- Група блоків на відміну від групи циліндрів у ufs не прив'язана до геометрії диску
- При створенні файлу намагаються знайти для нього індексний дескриптор у тій групі блоків, де міститься його каталог
- За допомогою бітових карт ведуть облік вільних блоків і індексних дескрипторів
- Розмір бітової карти – 1 кБ, отже група може містити до 8192 блоків
- Розмір блоку – 1 кБ
- Розмір індексного дескриптора – 128 байт



Файлова система ext3fs (Linux)

- Відрізняється від ext2fs наявністю журналу
- Можуть бути задані три режими роботи з журналом
 - Режим журналу (journal)
 - Усі зміни даних зберігаються у журналі
 - Суттєво впливає на продуктивність
 - Упорядкований режим (ordered)
 - Зберігаються тільки зміни в метаданих
 - Блоки даних зберігаються на диску перед метаданими, тому ймовірність їх ушкодження низька
 - Це режим за умовчанням
 - Режим мінімального записування
 - Зберігаються лише зміни в метаданих
- Журнал зберігають у схованому файлі `.journal` у кореневому каталозі



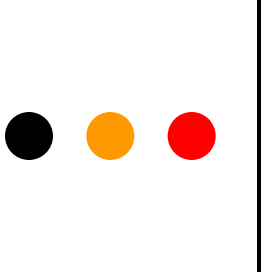
Файлова система /proc (UNIX, Linux)

- Це спеціальна файлова система, яка насправді взагалі не працює з диском
- Вміст кожного файлу і каталогу генерує програмне забезпечення у відповідь на запит файлової операції
- Кожному процесу відповідає каталог файлової системи /proc
 - Наприклад, процесу з PID=25 відповідає каталог /proc/25
- Зчитування файлів цього каталогу надає певну інформацію про процес
 - /proc/<pid>/cmdinfo – вміст командного рядка
 - /proc/<pid>/cpu – відомості про поточне завантаження процесора
 - /proc/<pid>/status – різноманітна статистика
- У сучасних системах доступна також інша інформація
 - /proc/modules – завантажені модулі ядра
 - /proc/mounts – змонтовані файлові системи
 - /proc/devices – зовнішні пристрої
 - /proc/cpuinfo – процесори
 - /proc/meminfo – стан пам'яті
- /proc/sys і /proc/sys/kernel надають доступ до внутрішніх змінних ядра
 - Можна не лише зчитувати значення, а й записувати нові, змінюючи тим самим характеристики системи без перекомпіляції ядра і без перезавантаження системи



Віртуальна файлова система VFS

- Основною метою є забезпечення роботи ОС з максимально широким набором різних файлових систем
 - Дискові файлові системи
 - ext2fs, ext3fs, ReiserFS, ufs, xfs, FAT, NTFS, ISO9660
 - Мережні файлові системи
 - NFS, SMB
 - Спеціальні (віртуальні) файлові системи
 - /proc
- Рівень VFS забезпечує доступ через стандартні файлові системні виклики до будь-якого рівня програмного забезпечення, що реалізує інтерфейс файлової системи
 - Програмні модулі, що реалізують інтерфейс файлової системи, називаються *модулями підтримки файлових систем*



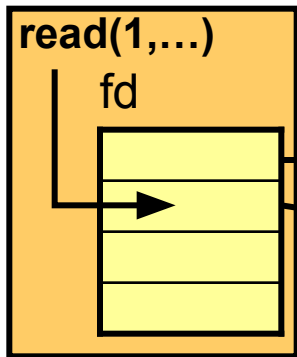
Об'єкти, що підтримує віртуальна файлова система VFS

- ▣ **Об'єкт файлової системи** (*filesystem object* або *superblock object*)
 - Відображає пов'язаний набір файлів, що міститься в ієрархії каталогів
 - Ядро підтримує по одному такому об'єкту для кожної змонтованої файлової системи
- ▣ **Об'єкт індексного дескриптора** (*inode object*)
 - Описує набір атрибутів і методів, за допомогою яких відображують файл на рівні файлової системи
 - Відображає файл як ціле
- ▣ **Об'єкт відкритого файлу** (*file object*)
 - Відображає відкритий файл на рівні процесу
 - Цей об'єкт створюють системним викликом відкриття файлу
 - В об'єкті зберігається режим доступу до файлу і покажчик поточної позиції
- ▣ **Об'єкт елемента каталогу** (*dentry object*) – у Linux
 - Відображає елемент каталогу і його зв'язок з файлом на диску
 - Знаходиться між об'єктами відкритих файлів і об'єктами індексних дескрипторів
 - Об'єкти елемента каталогу, як і об'єкти індексного дескриптора існують незалежно від процесів

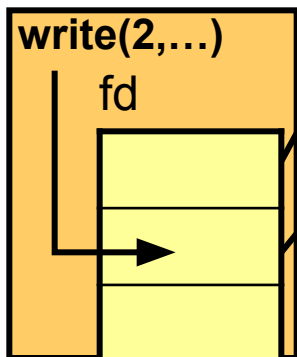
Доступ до файлу з процесу у Linux

Керуючі блоки процесів

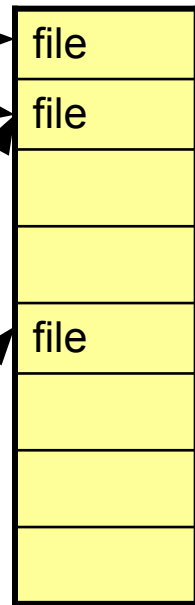
Процес 1



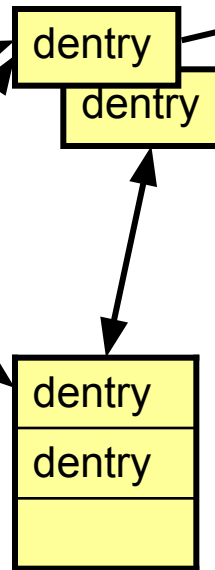
Процес 2



Таблиця відкритих файлів

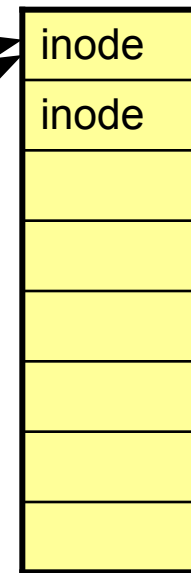


Елементи каталогу



Кеш елементів каталогу

Таблиця індексних дескрипторів



Дискові блоки



Файлова система NTFS

- Була розроблена як основна файлова система для ОС Windows NT на початку 90-х років
- Підтримує великі файли і великі диски (обсягом до 2^{64} байт)
- Стійка до відмов
- Забезпечує високу швидкість операцій
- Має гнучку структуру, що дозволяє додавання нових типів записів і атрибутів файлів
- Підтримує довгі символні імена (до 255 символів Unicode)
- Забезпечує керування доступом до каталогів і файлів



Структура тому NTFS

- Основою структури є *головна таблиця файлів* (Master File Table, MFT)
 - MFT містить щонайменше 1 запис для кожного файлу тому (у тому числі для самої себе), розмір запису однаковий для усього тому і може бути 1, 2 або 4 кБ (типово 2 кБ)
 - Усі файли тому ідентифікуються номером файлу, який визначається його позицією в MFT
- Увесь том є послідовністю кластерів (а не лише область даних)
 - Порядковий номер кластера у томі називають *логічним номером кластера* (Logical Cluster Number, LCN)
 - Номери кластерів зберігаються у 64-розрядних покажчиках
- Базова одиниця розподілу дискового простору в NTFS – це не один кластер, а неперервна область кластерів, яку називають *відрізком*, або *екстентом*
 - Екстент задається логічним номером його першого кластера і кількістю кластерів в екстенті

Структура тому NTFS

| | |
|-------------------------|---------------------------------|
| Завантажувальний сектор | |
| 0 | |
| 1 | |
| 2 | 1-й екстент MFT |
| ... | |
| 15 | |
| | Системний файл 1 |
| | Системний файл 2 |
| | ... |
| | Системний файл n |
| | Копія MFT (перші 3 записи) |
| | Файли |
| | Копія завантажувального сектора |
| | 2-й екстент MFT |
| | Файли |
| | 3-й екстент MFT |
| | |

- Завантажувальний сектор містить
 - Стандартний блок параметрів BIOS
 - Кількість кластерів у томі
 - Початковий логічний номер кластера основної копії MFT і дзеркальної копії MFT
- Копія завантажувального сектора знаходиться у середині тому
- Перший екстент MFT містить 16 стандартних записів, які створюють під час форматування, про системні файли NTFS
- Нульовий запис MFT містить опис самої MFT, зокрема, адреси усіх її екстентів
 - Безпосередньо після форматування MFT має один екстент, за яким записані системні файли
 - В момент створення першого не системного файлу створюють другий екстент MFT



Системні файли NTFS

| | | |
|---------------|-----------|--|
| 0 | \$Mft | Головна таблиця файлів |
| 1 | \$MftMirr | Резервна копія перших 16 записів MFT |
| 2 | \$LogFile | Файл журналу |
| 3 | \$Volume | Ім'я тому, версія NTFS та інша інформація про том |
| 4 | \$AttrDef | Таблиця визначення імен, номерів і описів атрибутів |
| 5 | \$. | Кореневий каталог |
| 6 | \$Bitmap | Бітова карта кластерів тому, що описує вільні й зайняті кластери |
| 7 | \$Boot | Адреса завантажувального сектора розділу |
| 8 | \$BadClus | Список зіпсованих кластерів тому |
| 9 | \$Quota | Квоти дискового простору для кожного користувача |
| 10 | \$Upcase | Таблиця перетворення регістру символів для Unicode |
| 11 – 15 | | Зарезервовані |



Структура файлу NTFS

- Файл складається з набору *атрибутів*
 - Кожний з атрибутів є незалежним *поток* (stream) байтів, який можна створювати, вилучати, зчитувати та записувати
- Деякі атрибути є стандартними для усіх файлів
 - Ім'я (*FileName*)
 - Версія (*Version*)
 - Стандартна інформація про файл, що включає час створення, час відновлення тощо (*Standard Information*)
- Інші атрибути залежать від призначення файлу
 - Каталог має атрибут *Index Root* – структуру даних, що містить список файлів цього каталогу
- Атрибут *Data* містить усі дані файлу
 - В NTFS файл може містити більше одного атрибуту Data, які розрізняють за іменами (дозволена наявність кількох поіменованих потоків даних всередині файлу)
 - За умовчанням файл містить один потік даних, що не має імені
- Атрибути невеликого розміру, які зберігають безпосередньо в запису MFT, називають *резидентними*
 - Атрибут Data може бути резидентним!
- Атрибути великого розміру зберігають на диску окремо, їх називають *нерезидентними*



Розміщення на диску файлів NTFS (1)

- Файли малого розміру (small)
 - Складаються щонайменше з таких атрибутів:
 - Стандартна інформація (SI – standard information)
 - Ім'я файлу (FN – file name)
 - Дані (Data)
 - Дескриптор захисту (SD – security descriptor)
 - Можуть бути повністю розміщені у запису MFT (усі атрибути резидентні)
- Великі файли (large)
 - Атрибут Data є нерезидентним, у його резидентній частині зберігають покажчики на усі екстенти



Розміщення на диску файлів NTFS (2)

- Дуже великі файли (huge)
 - Якщо файл має багато атрибутів або сильно фрагментований, атрибут Data не вміщує усіх покажчиків на екстенти
 - У такому випадку файл займає кілька записів MFT
 - Основний запис називають базовим записом файлу, а решту – записами переповнення
- Надвеликі файли (extremely huge)
 - У базовому записі може не вистачати місця для записів переповнення
 - Атрибут Attribute List можна зробити нерезидентним
 - Можна застосовувати подвійну непряму адресацію (нерезидентний атрибут посилається на інші нерезидентні атрибути)