Підсистеми ядра ОС

Підсистема управління введенням-виведенням (периферійними пристроями)

✔ **Підсистема управління даними (файлові системи)**

Підсистема управління оперативною пам'яттю

Підсистема управління задачами (процесами)

Підсистема забезпечення безпеки

ТЕМА 3:

Управління даними
(файли і файлові системи)

«Операційні системи»

План лекції:

- Основні задачі управління даними
- Файли.
- Файлові системи.
- Файлова система MS DOS
- Файлова система ОС Windows
- Файлова система ОС Unix

Основні задачі управління даними

- **виконання операцій** створення, видалення, перейменування, пошуку файлів, читання і запису даних у файли, а також ряду допоміжних операцій;
- забезпечення **ефективного використання дискового простору** і високої **швидкості доступу** до даних;
- забезпечення **надійності зберігання даних** і їх **відновлення** у випадку збоїв;
- **захист даних** користувача від несанкціонованого доступу;
- **управління одночасним спільним використанням** даних з боку декількох процесів.

Файли

«УПРАВЛІННЯ ДАНИМИ» = УПРАВЛІННЯ **ФАЙЛАМИ**

Файл – набір даних, що зберігається на ПП і доступний за іменем

(конкретне розташування даних на пристрої не цікавить користувача і повністю передовірено операційній системі)

Характеристики файлів, їх розміщення

Атрибути файлів – характеристики файлів

- ✔ Назва файлу.
- ✔ Розширення імені
- ✔ Тип файлу
- ✔ Розмір файлу
- ✔ Часові штампи
- ✔ Номер версії
- ✔ Власник файлу
- ✔ Атрибути захисту
- ✔ Тип доступу
- ✔ Розмір запису
- ✔ Прапори (бітові атрибути)
- ✔ Дані про розміщення файла на диску

Каталоги (папки, директорії) - записи, в яких містяться атрибути кожного файлу

У ранніх ОС – одноріневий каталог, дворівневий каталог

У сучасних ОС - **ієрархічна структура каталогів** (кожен каталог може, крім файлів, містити вкладені підкаталоги, глибина вкладення не обмежується.

Метадані - всі службові дані, що зберігаються у файловій системі і описують атрибути та розміщення файлів, структуру каталогів, загальну структуру дискового тому і т.п.

І завдання ОС - Розміщення файлів

- **Безперервне розміщення** - кожен файл займає безперервну послідовність блоків

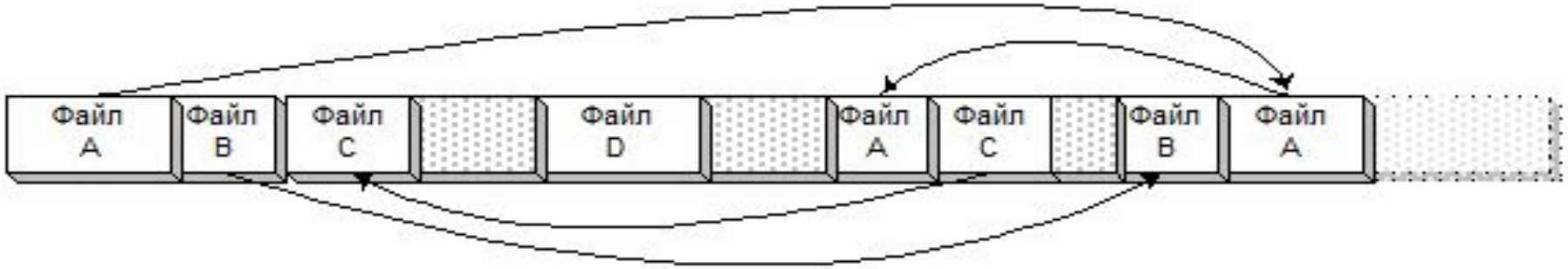
\+

- ✔ Інформація про розміщення файлу дуже проста і займає мало місця: номер початкового блоку файлу і кількість зайнятих блоків
- ✔ Доступ до будь-якої позиції у файлі виконується швидко

\-

- ✔ При створенні файлу потрібно заздалегідь знати його розмір
- ✔ Подальше можливе збільшення файлу утруднено - після кінця файлу може не виявитися достатньо вільного місця
- ✔ ***Фрагментація диска***

- **Сегментоване розміщення** - файли можуть розміщуватися «по шматочках» - один файл може займати декілька несуміжних сегментів різної довжини



\+

- ✔ При створенні файлу йому не виділяється пам’яті (лише при заповнені)
- ✔ При збільшенні розмірів файлу йому виділяються додаткові окремі сегменти

\-

- ✔ Інформація про розміщення файлу набагато складніше і обсяг цієї інформації змінний
- ✔ Уповільнення операцій з файлами (файл розкиданий по диску)
- ✔ Необхідність частої дефрагментації

Результат розміщення файлів:

для файлів – роздробленість

для дисків – фрагментація

ПРОБЛЕМА:

Великий файл →

багато блоків (фізичних) →

великий обсяг інформації про файл

(багато адрес блоків)

ВИХІД:

Кластери (логічний блок) – сукупність фізичних блоків, приймається рівним 2^k секторів.

Кластери?

- ✔ Кожному файлу відводиться ціле число кластерів
- ✔ В інформації про розміщення файлу зберігаються номери кластерів, а не секторів.
- ✔ Чим більший кластер, тим менше кластерів треба для файлу
- ✔ Можна адресувати більший обсяг диску

Чим більший кластер, тим краще?

Кластери?

1. В середньому половина останнього кластера кожного файлу незайнята (внутрішня фрагментацією).
2. Якщо хоча б один із секторів кластеру позначений як дефектний, то і весь кластер вважається дефектним → не може бути використаний → при збільшенні розміру кластера зростає число невикористовуваних секторів диска.

Оптимальний розмір кластера обчислюється або автоматично (при форматуванні диска), або задається вручну.

Способи подання інформації про вільні кластери диску:

1. **Зв'язаний лінійний список**: на початку кожного вільного кластера зберігати номер наступного за списком.

   Недолік: утруднюється пошук вільного безперервного фрагмента потрібного розміру → складніше оптимізувати розміщення файлів.

2. Список **безперервних вільних фрагментів** диска, а не окремих кластерів

   Недолік: працювати з таким списком складніше.

3. Використання **бітової карти** (bitmap) вільних кластерів - масив, що містить по одному біту на кожний кластер:1 – «кластер зайнятий», 0 - «кластер вільний».

І завдання ОС - Поділ файлів між процесами

ПРОБЛЕМА:

Два або більше процесів намагаються одночасно використовувати один файл → порушення цілісності даних

МОЖЛИВИЙ ВИХІД:

1) тільки один процес працює з файлом, виконуючи читання і запис;
2) з файлом працює довільне число процесів, але всі вони виконують тільки читання.

НАСЛІДКИ:

Істотне зниження продуктивності операційної системи

ТИПОВЕ РІШЕННЯ – додаткові параметри:

- ***Режим доступу*** визначає, які операції процес збирається виконувати з файлом:
  - «тільки для читання»,
  - «тільки для запису»,
  - «для читання і запису».

- ***Режим поділу*** визначає, які операції процес готовий дозволити іншим процесам:
  - «заборона запису»,
  - «заборона читання»,
  - «заборона читання і запису»
  - «без заборон».

Правила поведінки процесів і системи при відкритті файлу:

1) режим доступу другого процесу не повинен суперечити режиму поділу, встановленому першим процесом;
2) режим поділу, запитуваний другим процесом, не повинен забороняти той режим доступу, який встановив перший процес.

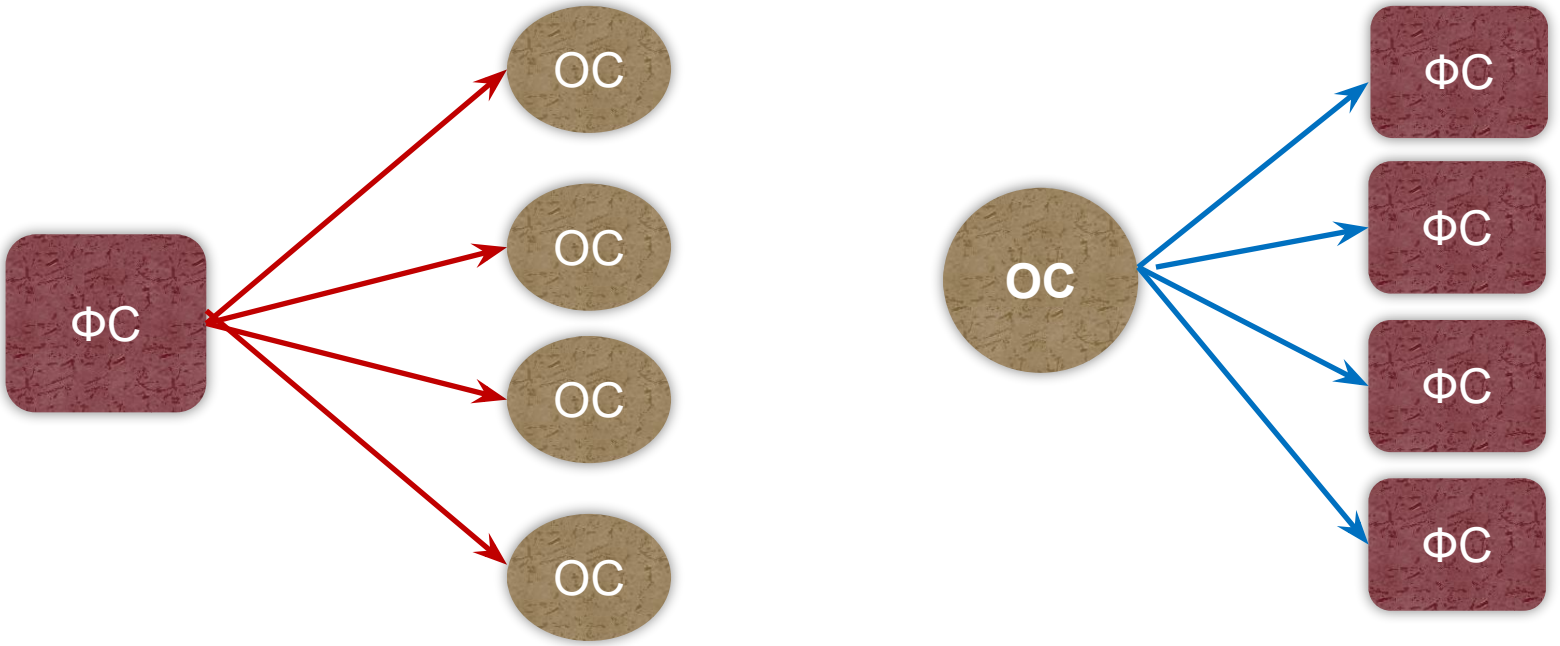
Правила поведінки при роботі з конкретним записом:

- ***ексклюзивне (виключне) блокування*** (на якомога менший інтервал часу) - процес дозволяє собі і читання, і запис, а іншим процесам тимчасово забороняє те й інше.
- ***кооперативне (не виключне) блокування*** – процес забороняє тільки запис всім процесам, у тому числі і собі самому, в той час як читання залишається дозволеним для всіх.

Файлові системи

«УПРАВЛІННЯ ДАНИМИ» = **УПРАВЛІННЯ** ФАЙЛАМИ

Файлова система (ФС) – стандартизована сукупність структур даних, алгоритмів і програм, що забезпечують зберігання файлів і виконання операцій з ними.



Файлова система (ФС) – це частина ОС

ФС призначена

- для забезпечення користувачеві зручного інтерфейсу при роботі з даними на диску,
- для забезпечення спільного використання файлів кількома користувачами і процесами.

ФС включає:

1) сукупність усіх файлів на диску,
2) набори структур даних для організації файлів:
   - каталоги файлів,
   - дескриптори файлів,
   - таблиці розподілу вільного і зайнятого простору на диску,
1) комплекс СПЗ для керування файлами:
   - створення і знищення,
   - читання і запис,
   - іменування,
   - пошук
   - . . .

Загальна модель файлової системи

Визначення за символічним іменем унікального імені файлу

Визначення за унікальним іменем характеристик файлу (переміщаються з диска в ОП)

Перевірка допустимості заданої операції для заданого файлу *(порівнюються повноваження користувача або процесу зі списком дозволених видів доступу до даного файлу*)

Визначення координат логічного запису у файлі

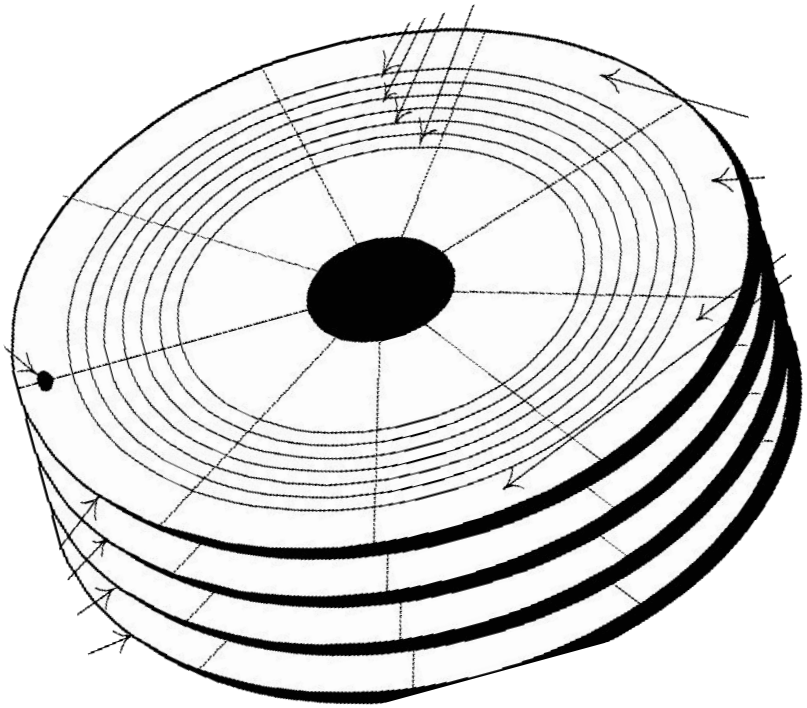
Визначення номера фізичного блоку, що відповідає логічному

Файли – на **диску**

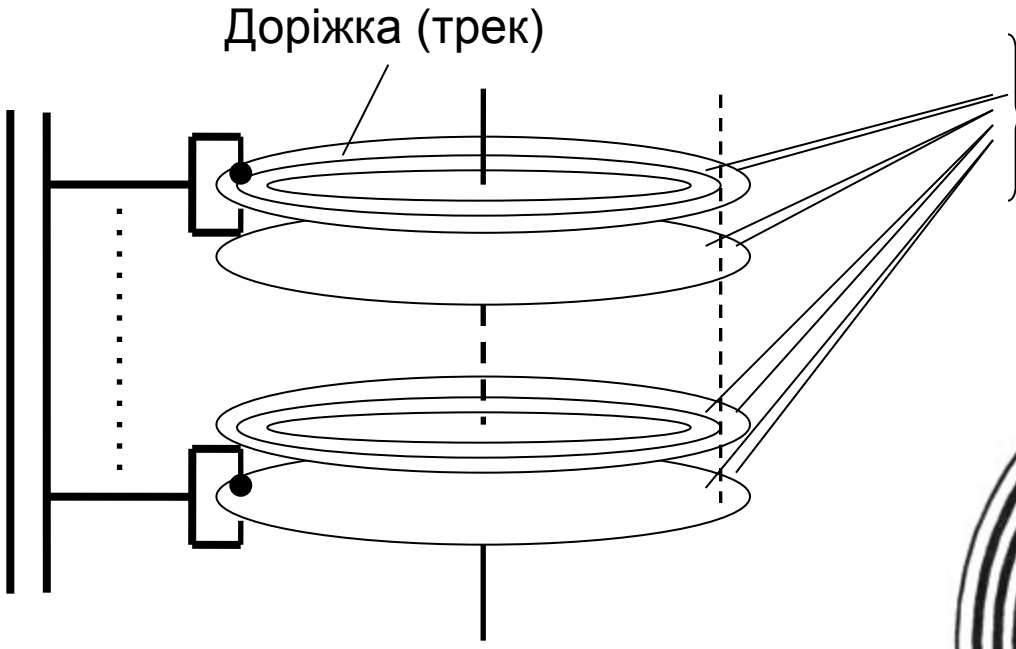
Каталоги – на **диску**

Логічні записи – **на диску**

Фізичні блоки – на **диску**



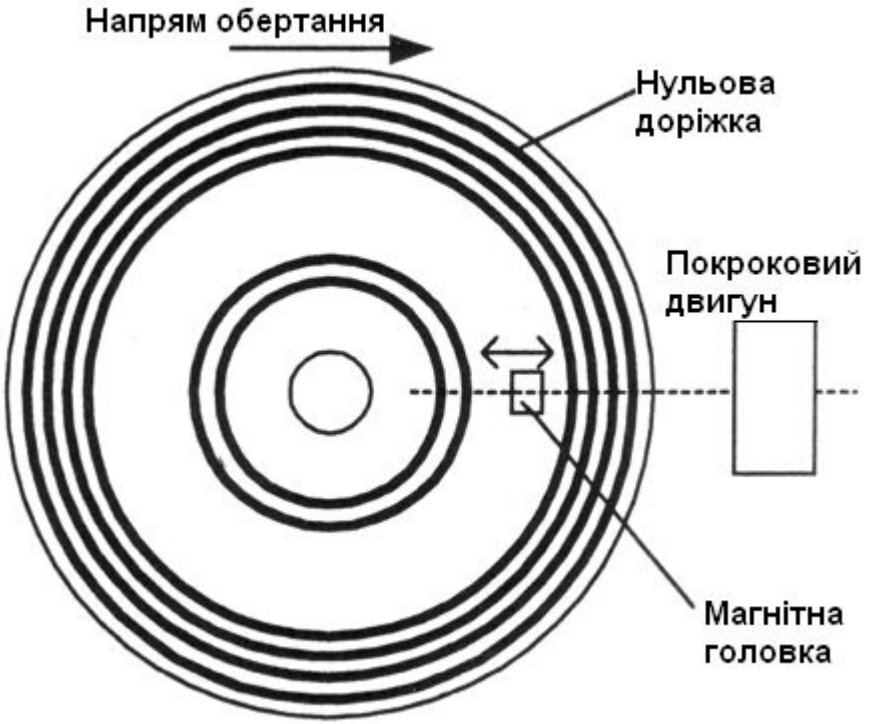
Фізична структура дисків



Доріжки - 0÷T

Циліндри - 0÷C

Головки - 0÷H



Головний завантажувальний запис - MBR

(Master Boot Record)

MBR – основний засіб завантаження з жорсткого диска, що підтримується BIOS.

Розташування – сектор 1, доріжка 0, головка 0

Адреса – 0000h:007Ch

№	*Зсув, байт*	*Розмір, байт*	*Вміст*	*Опис*
1	**0**	**446**	**Програма аналізу PT і завантаження System Boot Strap з активного розділу**	**Non-system bootstrap (NSB)**
2	**1Beh**	**16**	**Partition 1**	**Partition Table (PT)**
	1CEh	**16**	**Partition 2**	
	1DEh	**16**	**Partition 3**	
	1EEh	**16**	**Partition 4**	
3	**1FEh**	**2**	**Сигнатура AA55h**	**Сигнатура MBR.**

Структура елементів таблиці розділів (PT - *Partition Table*)

Зсув, байт	*Розмір, байт*	*Опис*
0	1	Boot indicator: 0 – не активний, 128 (80h) – активний.
1	1	Номер головки, з якої починається перший сектор розділу
2	2	Номер сектора (біти 0÷5) і номер циліндра (біти 6÷15) завантажувального сектора завантажувальника ОС
4	1	File System ID: 00 – невідома ОС (“пустий” розділ); 01 – FAT12; 04 – FAT16 (менше 32 Мбайтів); 05 – extended; 06 – FAT16; 07 – HPFS/NTFS; 0B – Windows 95 FAT32; 0C - Windows 95 FAT32 LBA; 0E - Windows 95 FAT16 LBA; 0F - Windows 95 Extended; 64,65 – Novell NetWare; 82,83,85 – Linux.
5	1	Номер головки для останнього сектора даного розділу
6	2	Номер сектора (біти 0÷5) і номер доріжки (біти 6÷15) останнього сектора даного розділу
8	4	Відносний номер початкового сектора даного розділу. RelSect=(Cyl·Sect·Head)+(Head·Sect)+(Sect-1),
12	4	Розмір розділу (в секторах)

!!! Розділи починаються з парних номерів доріжок.
Перший розділ може починатися із сектора 2 доріжки 0 (головка 0)

System ID	Тип раздела	System ID	Тип раздела
00	Empty («пустой» раздел)	41	PPC PreP Boot
01	FAT12	42	SFS
02	XENIX root	4D	QNX 4.x
03	XENIX usr	4E	QNX 4.x 2nd part
04	FAT16 (<32 Мбайт)	4F	QNX 4.x 3nd part
05	Extended	50	OnTrack DM
06	FAT16	51	OnTrack DM6 Aux
07	HPFS/NTFS	52	CP/M
08	AIX	53	OnTrack DM6
09	AIX bootable	54	OnTrack DM6
0A	OS/2 Boot Manager	55	EZ Drive
0B	Win95 FAT32	56	Golden Bou
0C	Win95 FAT32 LBA	5C	Priam Edisk
0E	Win95 FAT16 LBA	61	Speed Stor
0F	Win95 Extended	64	Novell Netware
10	OPUS	65	Novell Netware
11	Hidden FAT12	75	PC/IX
12	Compaq diagnost	80	Old Minix
14	Hidden FAT16 (<32 Мбайт)	82	Linux swap
16	Hidden FAT16	83	Linux native
17	Hidden HPFS/NTFS	84	OS/2 hidden C:
18	AST Windows swap	85	Linux Extended
1B	Hidden Win95 Fat	86	NTFS volume set
1C	Hidden Win95 Fat	A5	BSD/386
1E	Hidden Win95 Fat	A6	Open BSD
24	NEC DOS	A7	Next Step
3C	Partition Magic	EB	Be OS
40	Venix 80286		

Порядок завантаження операційної системи

1. Процедура початкового завантаження (**Bootstrap Loader**) викликається як переривання INT 19h.
2. Модулі ініціалізації **BIOS** зчитують (з floppy, hdd, CD, …) **MBR** у пам'ять за адресою 0000h:007Ch і передають їй керування (для вінчестерів – NSB).
3. MBR переглядає таблицю розділів **PT** і знаходить активний розділ.
4. MBR зчитує найперший сектор розділу (**BR**) в ОП і передає йому керування.
5. BR за допомогою **System bootstrap** виконує завантаження ОС.
   - завантаження файлів ОС
   - передача керування ОС
6. **ОС** виконує ініціалізацію власних програм і апаратних засобів.

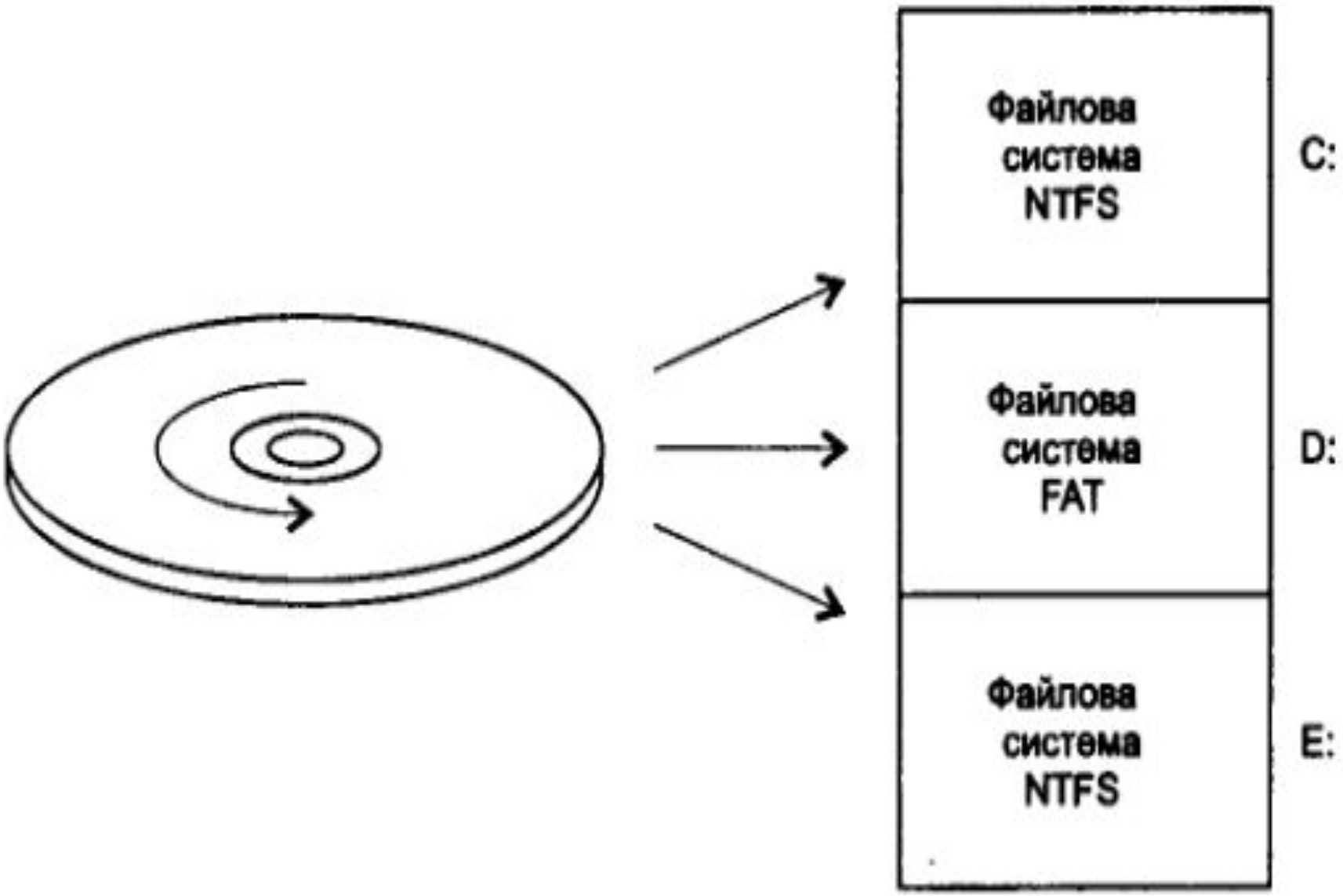
Логічні диски

Причини розбиття диску на логічні диски

- великий обсяг дисків → неможливість використання всього адресного простору;
- при ушкодженні логічного диска пропадає менше інформації;
- реорганізація і вивантаження диска маленького розміру виконується швидше;
- можливість розмістити на одному диску декількох ОС.

Типи логічних розділів:

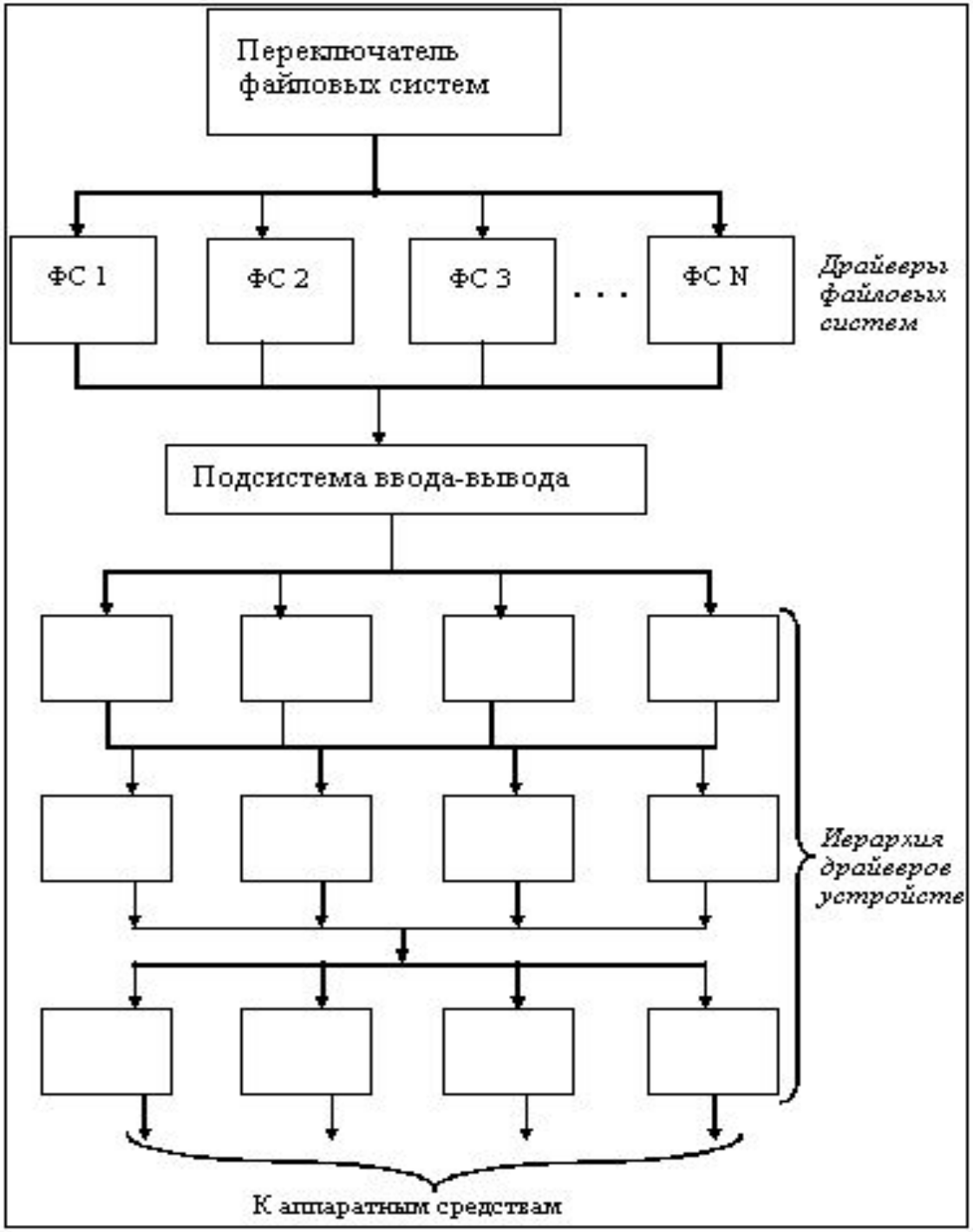
- первинний (*primary*) розділ (>1, <=4);
- розширений (*extended*) розділ (лише 1).


Файлова система NTFS
C:
Файлова система FAT
D:
Файлова система NTFS
E:

Архітектура сучасних файлових систем

Перемикач файлових систем (**IFS** – **I**nstallable **F**ilesystem **M**anager)

Основа роботи ФС – **драйвери** (програми для керування роботою ФС)



Надійність ФС

(руйнування ФС небезпечніше за руйнування комп'ютера)

- своєчасного **дублювання** інформації (backup)
- засоби для підтримки власної **сумісності**

Цілісність ФС

(деякі файлові операції зачіпають відразу декілька об'єктів ФС: копіювання файлу, видалення файлу,…)

- ретельно продуманий **порядок виконання операцій**
- **журналювання**

  Перевірка цілісності ФС

- застосування спеціальних **утиліт** (fsck, chkdsk, scandisk,…)
- евристичні перевірки

Журналювання

- для кожної протокольованої в журналі операції повинна існувати **зворотна операція**
- протоколювати не всі зміни, а лише зміни метаданих
- наявність процедур **відкату**

Управління "поганими" блоками

(блоки диска, для яких обчислена контрольна сума зчитуваних даних не збігається з збереженою контрольної сумою)

- **на рівні апаратури** - зберігати список поганих блоків в контролері диску → механізм обробки запитів до блоків диску працюватиме неефективно
- **на рівні ядра ОС** – ретельно сконструювати файл, що містить дефектні блоки і приховати його від прикладних програм → вони вилучаються зі списку вільних блоків.

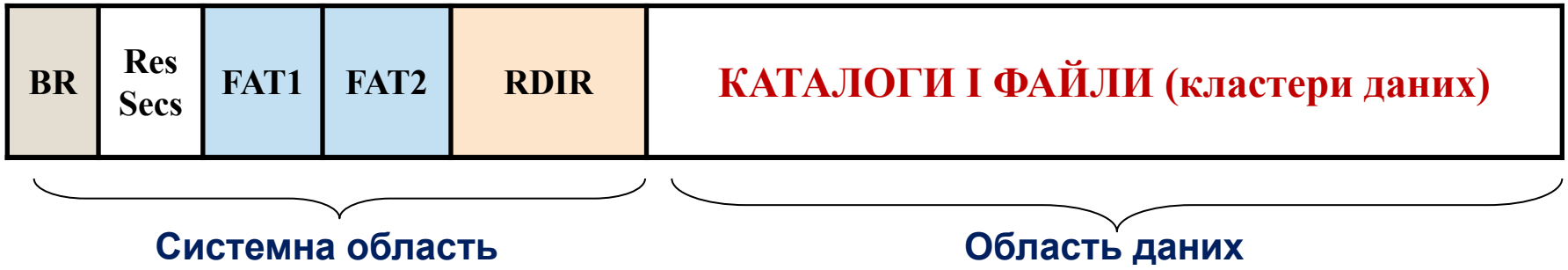
Гарантованих засобів абсолютного збереження інформації в файлах НЕ ІСНУЄ !!!

Файлова система FAT

- **FAT** розроблена для MS-DOS, можливе використання в ОС Windows
- **FAT** - з сегментованим розміщенням файлів
- **FAT** - без багатокористувацького захисту
- структура каталогів **FAT** – деревоподібна

FAT12 → FAT16 → VFAT → FAT32

Структура логічного диска з ФС FAT



BOOT-сектор - містить основні кількісні параметри дискового тому і ФС, а також може містити програму початкового завантаження ОС.

FAT - File Allocation Table - карта (образ) диска, стан кластерів - містить інформацію про розміщення файлів і вільного місця на диску

ROOT - кореневий каталог системи, що містить дані про файли та підкаталоги верхнього рівня, кожен з яких може містити файли і підкаталоги.

Область даних - масив кластерів, що містить всі файли і всі каталоги (крім кореневого).

Кластери - ділянки однакового розміру в області даних (складаються з секторів)

Формат Boot record для FAT32

Зсув	*Розмір*	*Вміст*
0	3	JMP xxxx – перехід на SB (байти 1-2 – команда безумовного переходу, байт 3 – 90h (NOP)
3	8	Системний ідентифікатор (назва фірми-виготовлювача ОС і версія)
11	2	Розмір сектора в байтах
13	1	Кількість секторів у кластері
14	2	Число зарезервованих секторів (як правило, 32)
16	1	Число копій FAT
17	...	.....
21	1	Дескриптор носія
22	2	0000h – для FAT32
24	2	Число секторів на доріжці
26	2	Число робочих поверхонь
28	4	Число прихованих секторів перед завантажувальним сектором
32	4	Число секторів на магнітному лиску
36	1	Число секторів у FAT-таблиці
37	2	Розширені прапорці
39	4	Номер кластера для початкового кластеру кореневого каталогу
43	2	-номер сектора з резервною копією завантажувального сектора
...	...	... ...
62 (3Eh)		Systen Bootstrap
510	2	Сигнатура (1FEh)

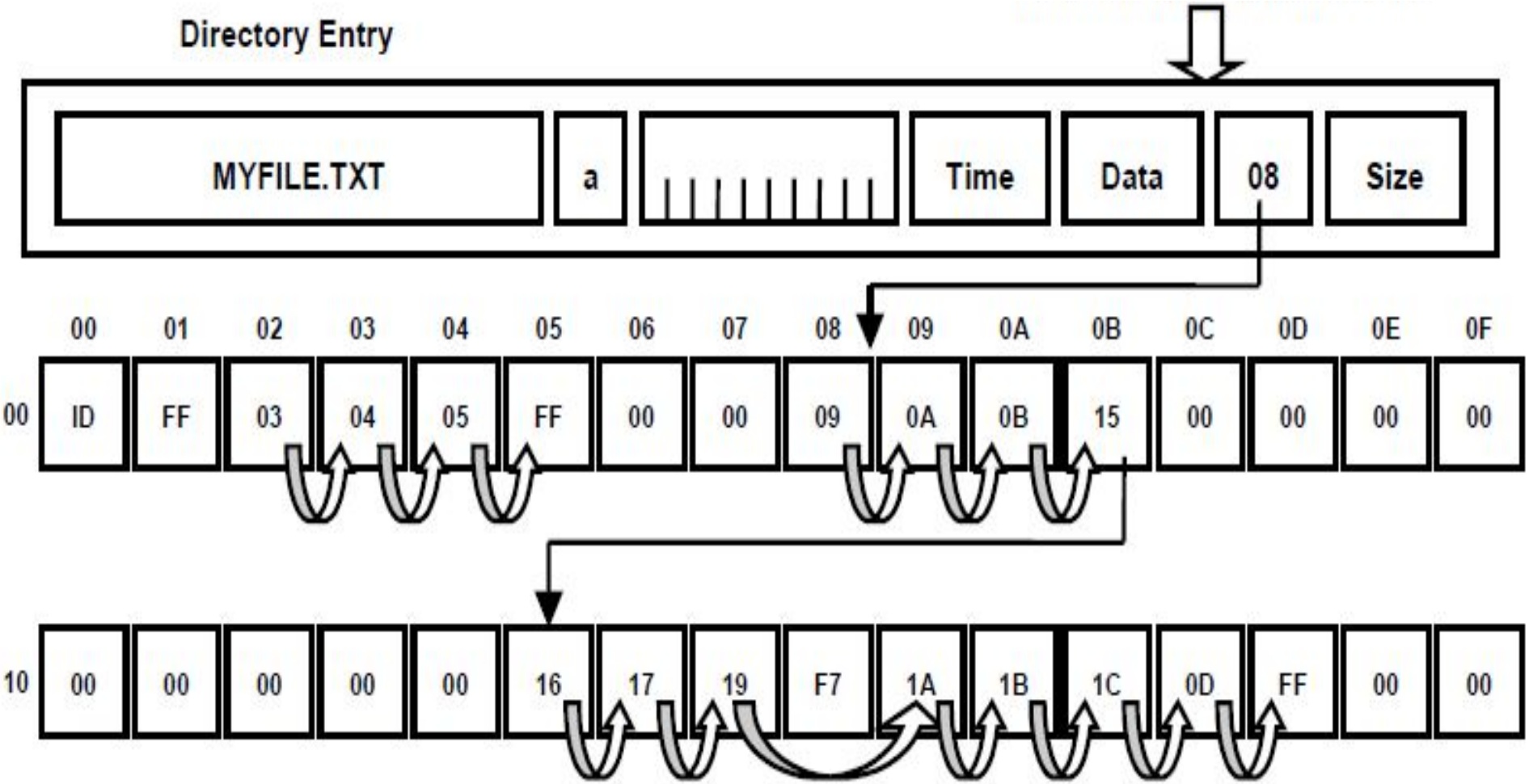
Ідентифікація кластерів у FAT-таблиці

FAT12	*FAT16*	*FAT32*	*Опис*
000h	**0000h**	**00000000h**	**Вільний кластер**
FF0h - FF6h	**FFF0h - FFF6h**	**FFFFFFF0h - FFFFFFF6h**	**Зарезервований кластер**
FF7h	**FFF7h**	**FFFFFFF7h**	**Поганий кластер**
FF8h - FFFh	**FFF8h - FFFFh**	**FFFFFFF8h - FFFFFFFFh**	**Останній кластер у списку**
002h - FEFh	**0002h – FFEFh**	**00000002h – FFFFFFEFh**	**Номер наступного кластера в списку**

Формати FAT-таблиць

12-бітовий формат: 2^{12} = 4096 кластерів ≈ 2Мб
16-бітовий формат: 2^{16} = 65536 кластерів ≈ 32 Мбайти.
32-розрядний формат: 2^{28} кластерів

Ланцюжок кластерів, розподілених файлу



Кореневий каталог *(RDIR – Root Directory)*

RDIR – сукупність дескрипторів

Дескриптори (32-байти) – містять інформацію про файли і каталоги.

Для коротких імен у FAT32

Для довгих імен

Байт атрибутів файлу

Біт	*Опис*
0	тільки для зчитування (не можна писати і не можна стирати)
1	Прихований файл
2	Системний файл
3	Мітка диска.
4	Файл, що є підкаталогом даного каталогу
5	Прапорець архівації. Якщо 1, то файл не був вивантажений утилітою архівації
6-7	Зарезервовані

Приклади

00h = 00000000 – звичайні файли

07h = 00000111 – тільки читати, приховані, системні файли (io.sys, msdos.sys) ;

08h = 00001000 – мітка тому.

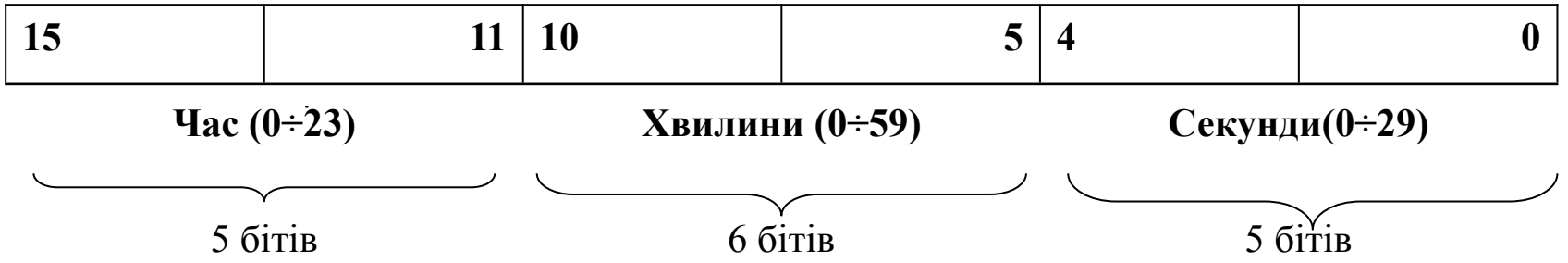
10h = 00010000 – каталог ;

20h = 00100000 - Звичайний файл, до якого не були застосовані програми backup.exe чи xcopy.exe, архівний файл.

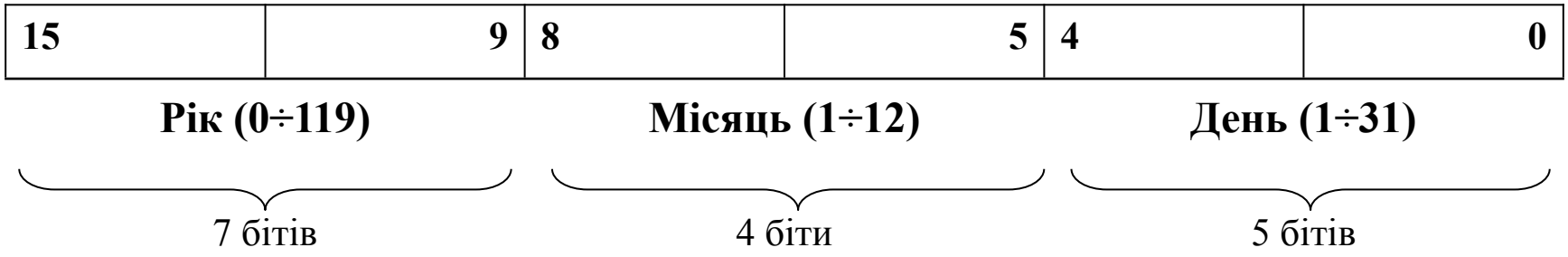
При видаленні файлу перший байт його імені заміняється на байт E5h ("x"). Усі кластери, розподілені файлу, відзначаються у FAT як вільні (00h).

Час і дата створення або модифікації файлу

Формат поля часу:



Формат поля дати:



Робота з файлами в MS-DOS

Системні функції:

- створювати файл
- видаляти
- змінювати атрибути файлу
- перейменовувати
- створювати і видаляти каталоги
- відкривати файл
- отримувати доступ до файлу
- закривати файл
- …

За допомогою команди програмного переривання **Int 21h**
Конкретна функція – значення регістру **AH**

Організація доступу до даних в MS DOS

2 способи доступу до даних:

1. За допомогою **блоку управління файлом** (FCB) – не використовується, залишений для сумісності з MS DOS 1.0
2. Використання **хендлів**.

Хендл - деяке число, яке система повертає програмі користувача при вдалому виконанні операції відкриття або створення файлу (значення його не грає ролі для програми)

Хендл – покажчик на файл.

Стандартні хендли: 0 – стандартне введення, 1 – стандартне виведення, 2 – стандартне виведення повідомлень про помилки, 3 – стандартний пристрій послідовного введення-виведення, 4 – стандартний принтер.

*** ***«handle».*** *рос.: дескриптор, посилання, логічний номер, ключ, маніпулятор, описувач, індекс, рукоятка …*

Структура даних у памяті

Для доступу до відкритих файлів ФС використовує таблиці двох типів:

1) ***Таблиця SFT*** (System File Table) – містить записи про всі файли, в даний момент відкриті програмами користувача і самої ОС
2) ***Таблиці JFT*** (Job File Table) – створюються для кожної програми, що запускається → одночасно може існувати декілька JTF-таблиць.

Таблиця SFT (System File Table):

- зберігається в системній пам'яті
- число записів SFTвизначається параметром FILES в CONFIG.SYS (<255)
- якщо файл був відкритий декілька разів, то для нього буде декілька записів

Записи SFT містять:

✔ копію каталожної інформації про файл;

✔ адресу каталожного запису (сектор та номер запису в секторі);

✔ поточне положення вказівника читання/запису;

✔ номер останнього записаного або прочитаного кластера файлу;

✔ адресу в пам'яті програми, що відкрила файл;

✔ режим доступу, заданий при відкритті;

✔ значення лічильника посилань на цей запис з усіх таблиць JFT(коли =0, запис SFT стає вільним, оскільки файл закритий)

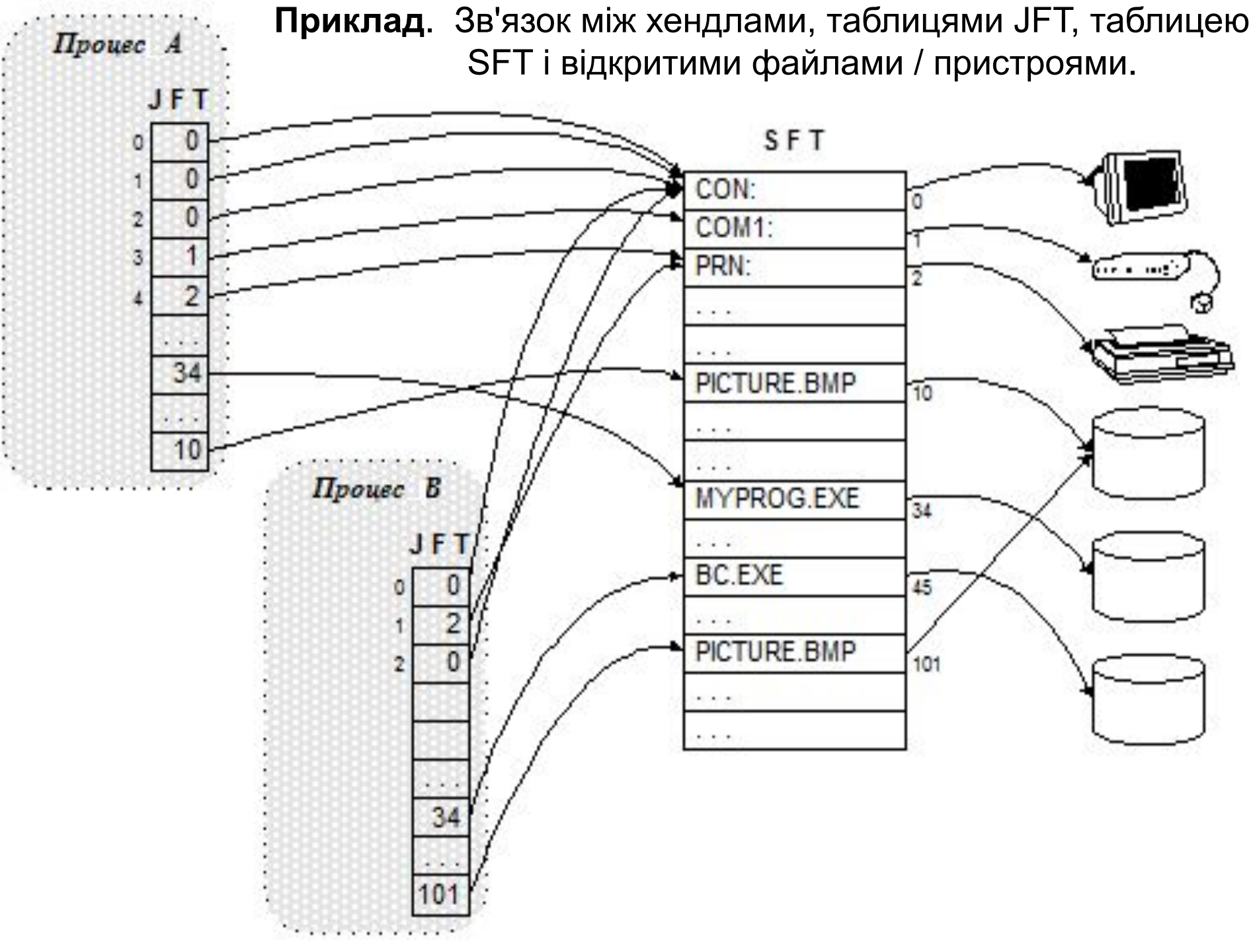
Таблиця JFT (Job File Table):

- складається з однобайтових записів,
- значення кожного запису – індекс (номер запису) в таблиці SFT
- невживані записи містять FF_{16},
- розмір таблиці – до 255 записів (за замовчуванням -20 байт)

* Хендл - індекс запису в таблиці JFT даної програми

** Використання JFT дає можливість відокремити логічне поняття стандартного пристрою

Приклад. Зв'язок між хендлами, таблицями JFT, таблицею SFT і відкритими файлами / пристроями.



Файлова система NTFS

Версії NTFS

Версія	ОС
1.2	Windows NT
3.0	Windows 2000
3.1	Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows 7, Windows Server 2008, Windows Server 2008 R2.

Підтримка ОС

MS-DOS

Драйвер NTFSDOS Марка Руссиновича (Mark Russinovich) — підтримка читання, в версії Professional – читання і запису на NTFS-розділи

Windows

Драйвер NTFS for Windows 98 Марка Руссиновича - підтримує читання розділів NTFS)

Драйвер Paragon Software Group NTFS for Windows 98 – підтримує читання і запис

Linux

Проект Linux-NTFS. Включає модуль ядра та набір утиліт для різних операцій з NTFS (перевірка цілістності, відновлення видалених файлів, зміна размеру і ін.). Включається в ядро Linux з версії 2.2.

Проект NTFS-3G – наслідок попереднього. Більш повна підтримка запису на NTFS-розділи, високий рівень надійності і швидкості. Активно развивається.

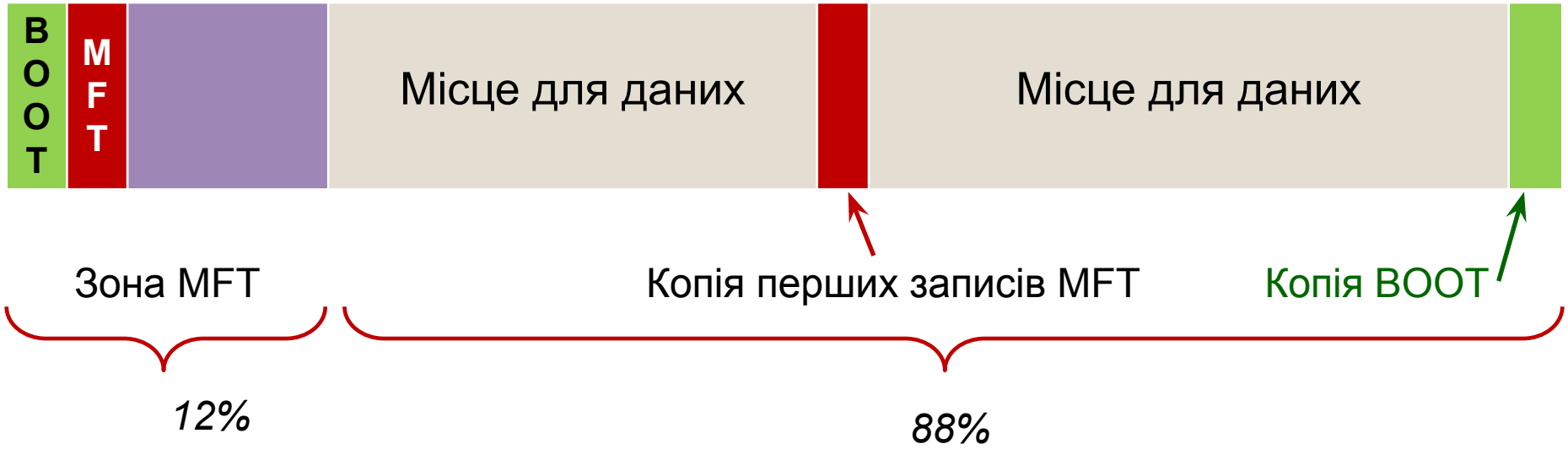
Проект Captive NTFS – створювався як «обгортка» для використання оригинального драйвера Windows NT в ОС Linux. Працює вкрай повільно, має деякі технічні обмеження. Далі не развивається.

Драйвер Paragon NTFS for Linux. Підтримує читання і запис, має ряд інших утиліт.для операцій з NTFS-розділами. Безкоштовний для домашніх користувачів.

Нові можливості NTFS

- Розвинені засоби **захисту даних**
  - захист від несанкціонованого доступу до даних
  - розмежування прав доступу для різних користувачів і груп
- Підтримка **великих файлів** і великих дисків (до 2^{64} бйт)
- Низький рівень фрагментації
- Швидкий **пошук** файлів у великих каталогах, на великих дисках.
- Забезпечення **цілісності**
- Можливість зберігання файлів в **зашифрованому** вигляді.
- Можливість **ущільнення** даних на рівні окремих файлів
- Механізм точок повторного аналізу (reparse points
- Можливість **протоколювання** всіх змін у ФС
- **Розширюваність** системи.

Структура тому NTFS

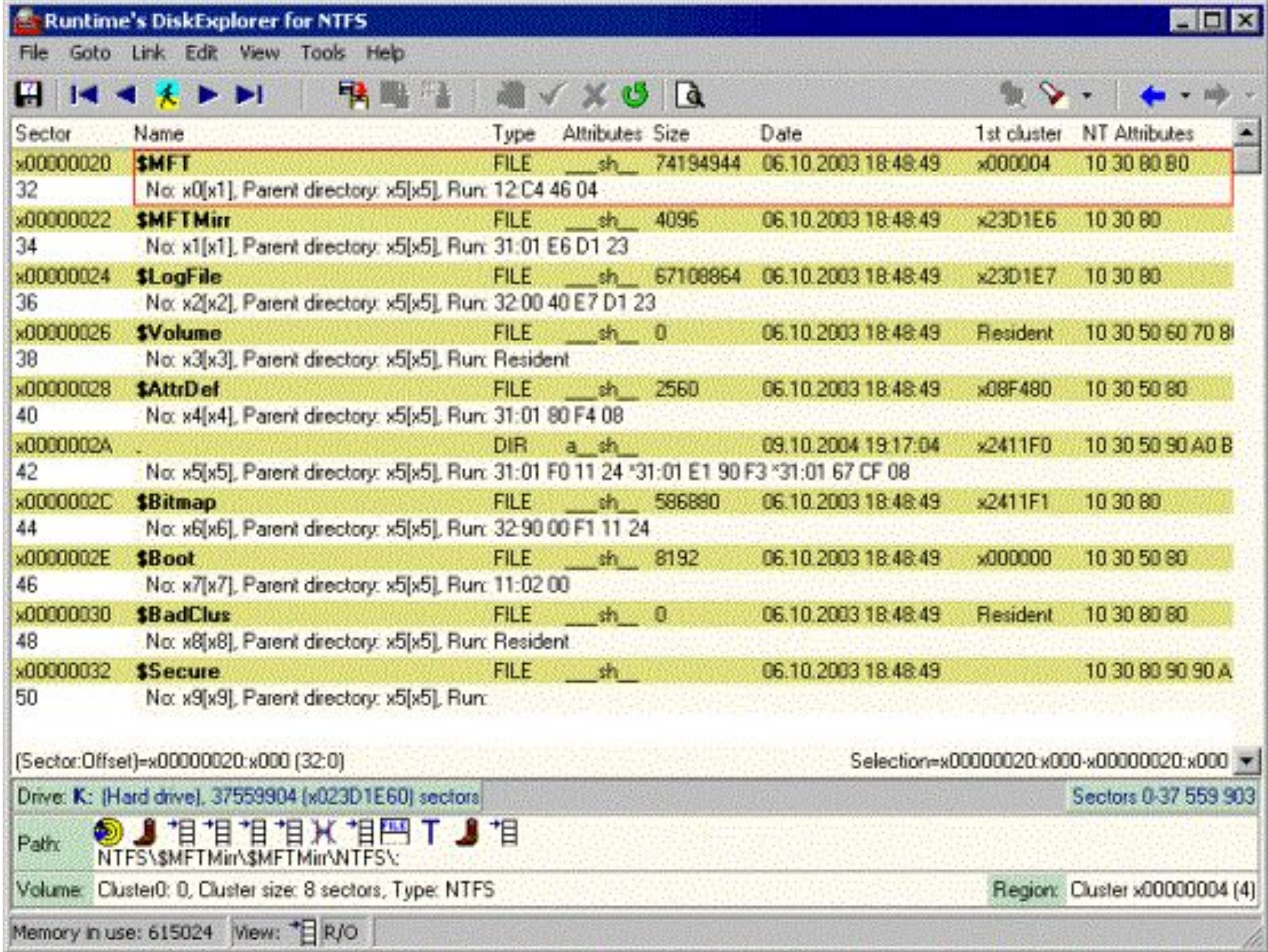


* *На диску немає нічого, крім файлів.*
У файлі немає нічого, крім атрибутів.

** *Весь розділ NTFS – або файл, або його частина.*

*** *Кожен елемент являє собою файл - навіть службова інформація*

0	$MFT	Містить повний список файлів тому NTFS
1	$MFTmirr	Копія перших 3-х записів MFT, яка зберігається як файл десь у середині диска
2	$LogFile	Журнал протоколювання транзакцій
3	$Volume	Ім'я тому, серійний номер, дата форматування і т.п.
4	$AttrDef	Файл з перерахуванням всіх атрибутів, використовуваних для опису файлів на даному томі.
5	$.	Кореневий каталог тому
6	$Bitmap	Бітова карта зайнятості кластерів томи
7	$Boot	BOOT-сектор. Він є першим сектором тому, але теж вважається файлом
8	$BadClus	Таблиця (файл) поганих кластерів
9	$Quota	Квоти вмкористовуваного простору у для кожного користувача
10	$Upcase	Файл, що задає пари великих / малих літер для всіх мов, підтримуваних Windows
11-15		Зарезервовано для майбутнього використання


Runtime's DiskExplorer for NTFS
File Goto Link Edit View Tools Help
Sector Name Type Attributes Size Date 1st cluster NT Attributes
x00000020 $MFT FILE ___sh__ 74194944 06.10.2003 18:48:49 x000004 10 30 80 80
32 No: x0[x1], Parent directory: x5[x5], Run: 12 C4 46 04
x00000022 $MFTMirr FILE ___sh__ 4096 06.10.2003 18:48:49 x23D1E6 10 30 80
34 No: x1[x1], Parent directory: x5[x5], Run: 31:01 E6 D1 23
x00000024 $LogFile FILE ___sh__ 67108864 06.10.2003 18:48:49 x23D1E7 10 30 80
36 No: x2[x2], Parent directory: x5[x5], Run: 32:00 40 E7 D1 23
x00000026 $Volume FILE ___sh__ 0 06.10.2003 18:48:49 Resident 10 30 50 60 70 8
38 No: x3[x3], Parent directory: x5[x5], Run: Resident
x00000028 $AttrDef FILE ___sh__ 2560 06.10.2003 18:48:49 x08F480 10 30 50 80
40 No: x4[x4], Parent directory: x5[x5], Run: 31:01 80 F4 08
x0000002A . DIR a__sh__ 09.10.2004 19:17:04 x2411F0 10 30 50 90 A0 B
42 No: x5[x5], Parent directory: x5[x5], Run: 31:01 F0 11 24 *31:01 E1 90 F3 *31:01 67 CF 08
x0000002C $Bitmap FILE ___sh__ 586880 06.10.2003 18:48:49 x2411F1 10 30 80
44 No: x6[x6], Parent directory: x5[x5], Run: 32:90 00 F1 11 24
x0000002E $Boot FILE ___sh__ 8192 06.10.2003 18:48:49 x000000 10 30 50 80
46 No: x7[x7], Parent directory: x5[x5], Run: 11:02 00
x00000030 $BadClus FILE ___sh__ 0 06.10.2003 18:48:49 Resident 10 30 80 80
48 No: x8[x8], Parent directory: x5[x5], Run: Resident
x00000032 $Secure FILE ___sh__ 06.10.2003 18:48:49 10 30 80 90 90 A
50 No: x9[x9], Parent directory: x5[x5], Run:
(Sector:Offset)=x00000020:x000 (32:0)
Selection=x00000020:x000-x00000020:x000
Drive: K: (Hard drive), 37559904 (x023D1E60) sectors
Sectors 0-37 559 903
Path:
NTFS\$MFTMir\$MFTMir\NTFS\:
Volume: Cluster0: 0, Cluster size: 8 sectors, Type: NTFS
Region: Cluster x00000004 (4)
Memory in use: 615024
View:
R/O

Самостійно!

DiskExplorer от Runtime Software – програма для визначення структури диска:

DiskExplorer for FAT

DiskExplorer for NTFS

MFT і його структура

BOOT-сектор (файл) – містить інформацію про те, де починається MFT

MFT – Master File Table (файл) - головна таблиця файлів – централізований каталог всіх інших файлів диска і себе самого.

- MFT поділений на записи фіксованого розміру (1 Кбайт).
- 1-16 файли – службові – метафайли – мають фіксоване положення
- самий перший метафайл - сам MFT
- друга копія перших 4-х записів – зберігається посередині диска

MFT - массив записів типу ***FILE Record*** (в UNIX - **inodes)**

Файли і потоки

Поняття файлу в NTFS включає:

- ❑ обов'язковий елемент - запис в MFT : вся інформація про файл (ім'я файлу, розмір, положення на диску окремих фрагментів, і т.д.)
- ❑ опціональний елемент - потоки даних файлу.

Наслідки такої організації:

✔Файл не має даних → не витрачається вільне місце самого диска.

✔Файл малого розміру - дані файлу зберігаються прямо в MFT → не мають свого "фізичного" втілення в основній файловій області.

✔ Файли в NTFS – **сукупність потоків** (streams) : потік атрибутів безпеки, потік атрибутів розташування, …, потік безпосередньо даних файлу.

Ім’я файлу - будь-які символи, представлені в Unicode у 16-бітному уявленні, яке дає 65535 різних символів.

Максимальна довжина імені файлу - 255 символів.

Файлові записи (FILE Record)

```
FILE Record
   Header       ; заголовок
   Attribute 1    ; атрибут 1
   Attribute 2    ; атрибут 2
   ...            ; ...
   Attribute N    ; атрибут N
End Marker (FFFFFFFFh)    ; маркер кінця
```

- ✔ кількість і довжина атрибутів різні для різних файлів;
- ✔ розмір структури **FILE Record** строго фіксований і зазвичай = 1 Кбайт (це значеннння зберігається в $boot-файлі);
- ✔ перший байт файлового запису співпадає з початком сектору;
- ✔ якщо реальна довжина атрибутів меньша за размери файлового запису, її хвіст не використовується, якщо більша – створюється додатковий файловий запис (***extra FILE Record***), що має посилання на попередній запис.

Атрибути файлу NTFS

<table>
<tr><th colspan="4">Заголовок атрибута</th><th>Значення атрибута</th></tr>
<tr><td>Тип</td><td>[Ім’я]</td><td>Довжина</td><td>Дані про розмі-щення</td><td></td></tr>
</table>

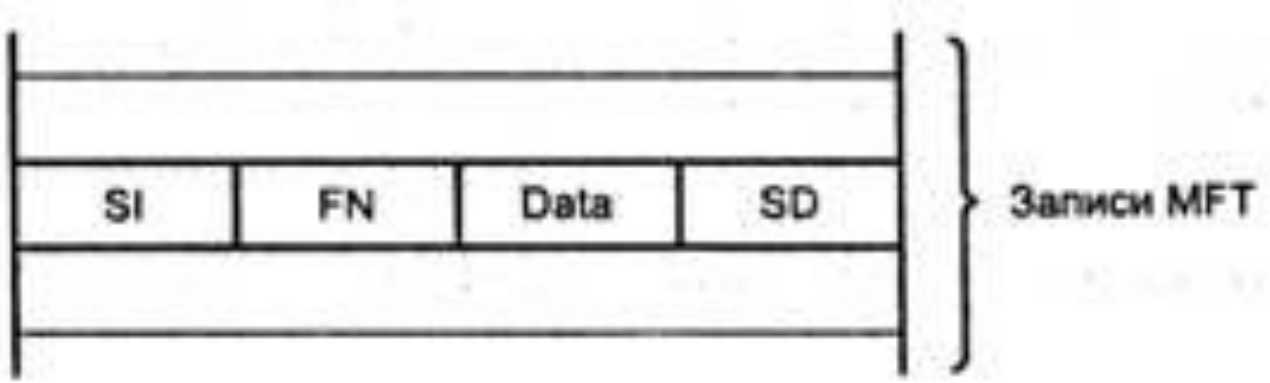
Зберігається в MFT

Зберігається в MFT (***резидентний***)

або в кластерах області даних (***нерезидентний***)

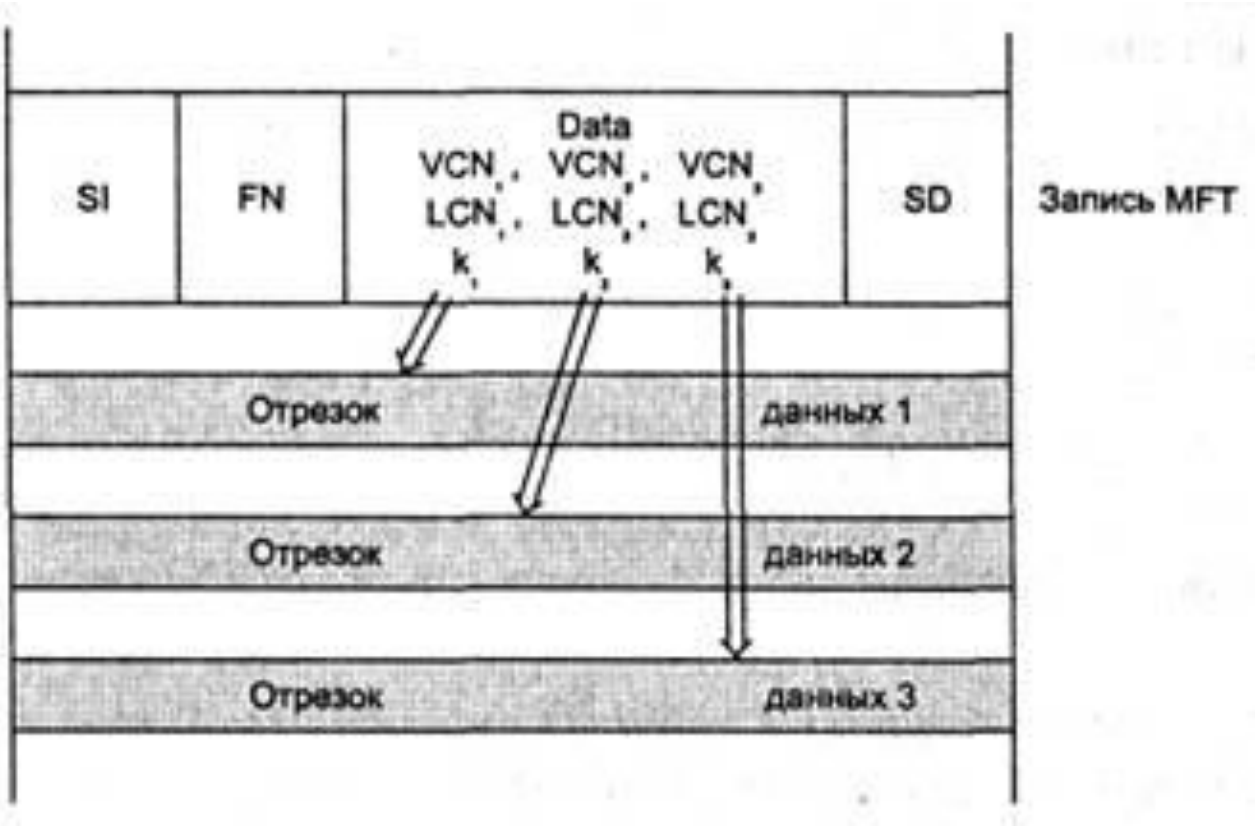
Невеликі файли (small)

- ✔ Можуть розташовуватись усередині одного запису MFT
- ✔ Мають атрибути:
  - стандартна інформація (SI - standard information);
  - ім'я файлу (FN - file name);
  - дані (Data);
  - дескриптор безпеки (SD - security descriptor).



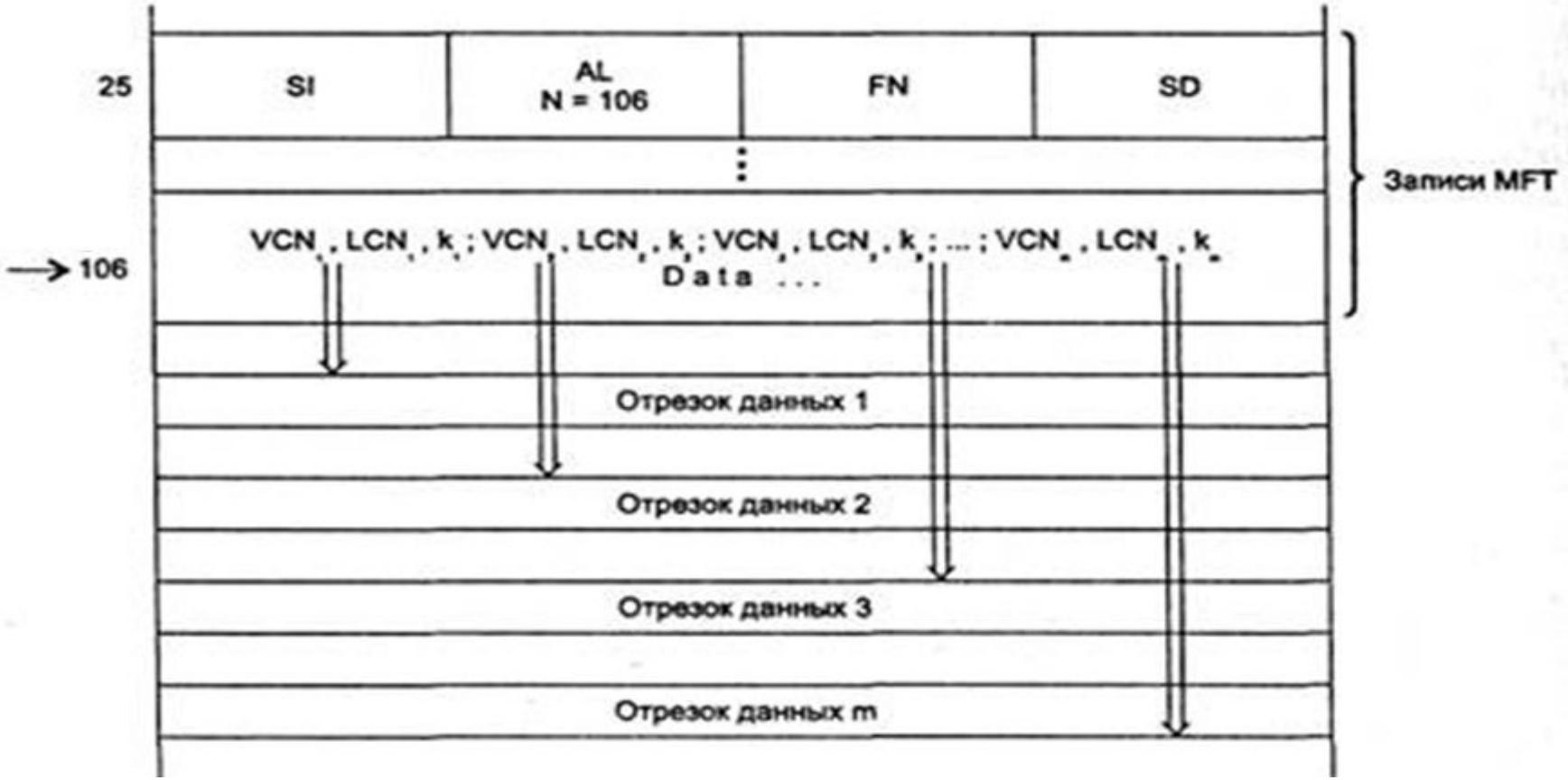
Великі файли (large)

- ✔ Атрибуту Data містить ознаку того, що цей атрибут є нерезидентним (знаходиться у відрізках поза таблиці MFT)
- ✔ Атрибут Data містить адресну інформацію (LCN, VCN, k) кожного відрізку даних



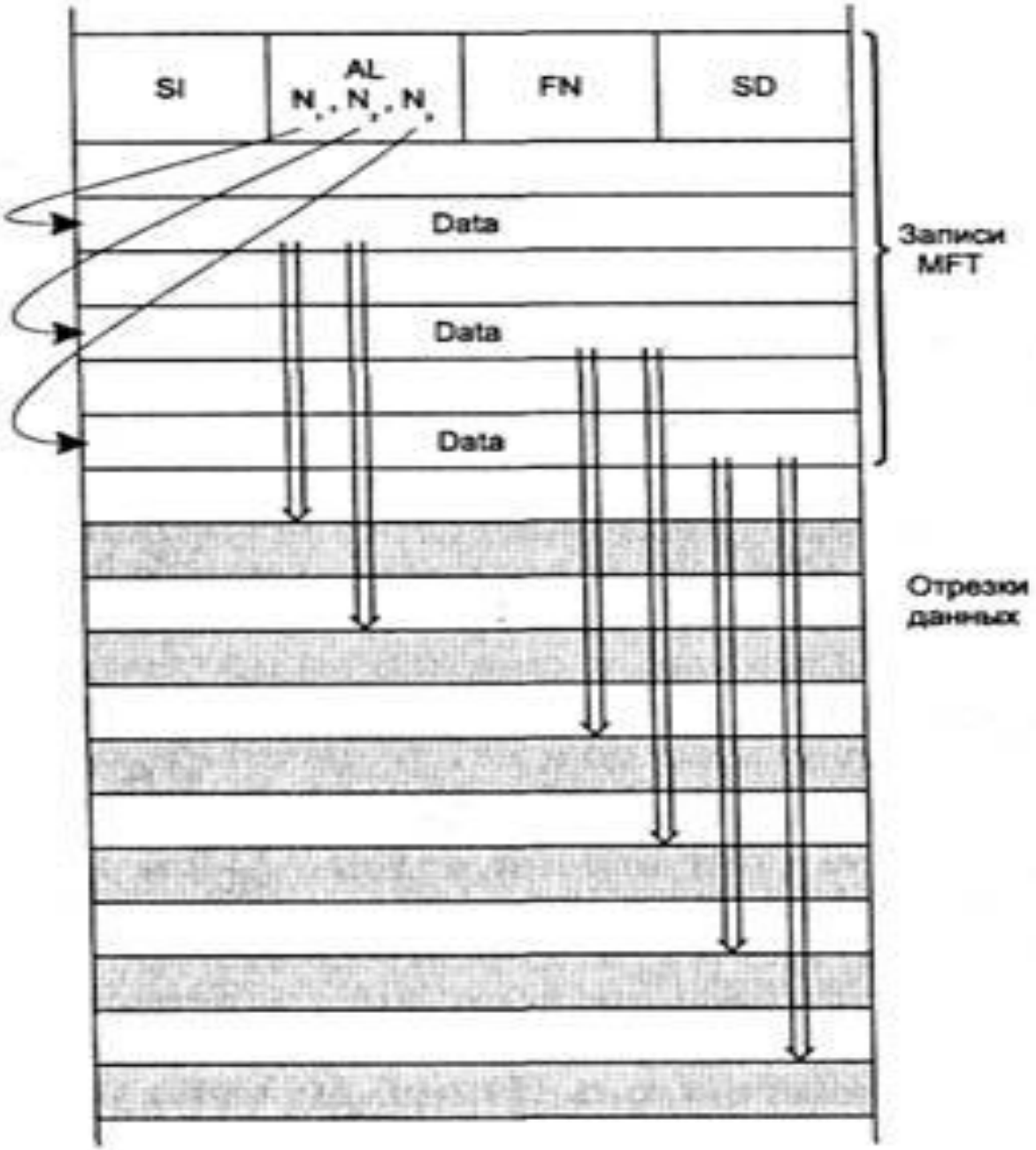
Дуже великі файли (huge)

- ✔ Атрибут даних, що зберігає адреси нерезидентних відрізків даних, не поміщається в одному записі – поміщається в інший запис MFT, а посилання на нього - в атрибуті Attribute List.
- ✔ Сам атрибут даних містить адреси нерезидентних відрізків даних



Надвеликі файли (extremely huge)

- ✔ Attribute List містить покажчики на декілька атрибутів, розташованих в додаткових записах MFT



Каталоги в NTFS

- Каталог на NTFS - специфічний файл, який зберігає посилання на інші файли та каталоги, створюючи ієрархічну будову даних на диску.

- Файл каталогу поділений на блоки, кожен з яких містить:
  - ім'я файлу,
  - базові атрибути,
  - посилання на елемент MFT (там повна інформація про елемент каталогу)

- Внутрішня структура каталогу – бінарне дерево.

Журналювання

- NTFS - відмовостійка система, яка цілком може привести себе в коректний стан при практично будь-яких реальних збоях.
- NTFS не має проміжних (помилкових чи некоректних) станів - квант зміни даних не може бути поділений на до і після збою: він або завершений, або скасований.

Транзакція - дія, що здійснюється цілком і коректно або не здійснюється взагалі.

Сутність механізму журналювання:

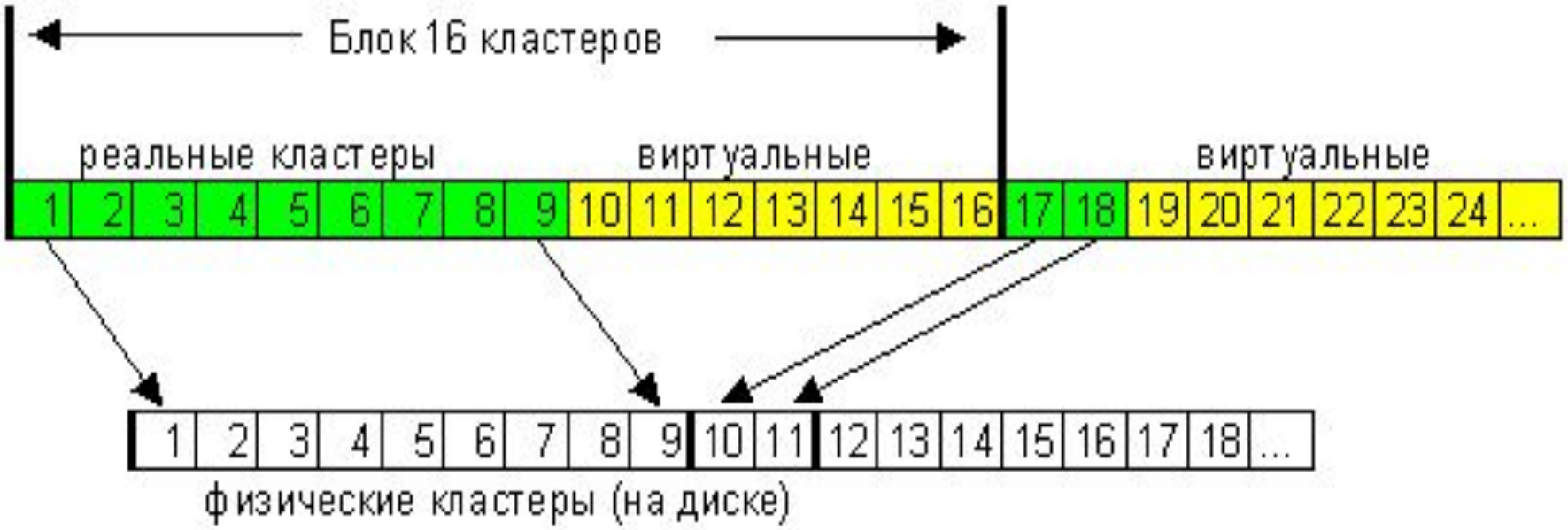
- ✔ запис інформації про всі транзакції і їх результати;
- ✔ після збоїв – повернення до стану, в якому завершення транзакцій успішне.

!!! Журналювання – не панацея, а лише засіб **істотно скоротити число помилок і збоїв системи.**

!!! NTFS гарантує **коректність файлової системи**, а не наших данных.

Ущільнення

- Атрибут – ущільнено/не ущільнено
- NTFS має вбудовану підтримку ущільнення дисків (здійснюється блоками по 16 кластерів)
- З’являються "віртуальні кластери" → фрагментація файлів



Hlinks

- Hard Link - це коли один і той самий файл має два імені (декілька покажчиків файла-каталога або різних каталогів вказують на один MFT-запис) .
- Файл фізично стирається лише тоді, коли буде видалено його останнє ім'я.

Symbolic Links (працює тільки в NT5 - Windows 2000)

- Мета: спрощення системи каталогів
- Сутність: якщо не подобається каталог з довгим іменем, можна прилінкувати його в інший каталог (для нас – короткий шлях, для системи – реальний, довгий)
- Для створення Symbolic Links програма ***junction*** (junction.zip (15 Kb), 36 кб) від Mark Russinovich (http://www.sysinternals.com/).
- Для видалення зв'язку - стандартна команда ***rd***.

Шифрування (NT5)

Файлова система в ОС UNIX

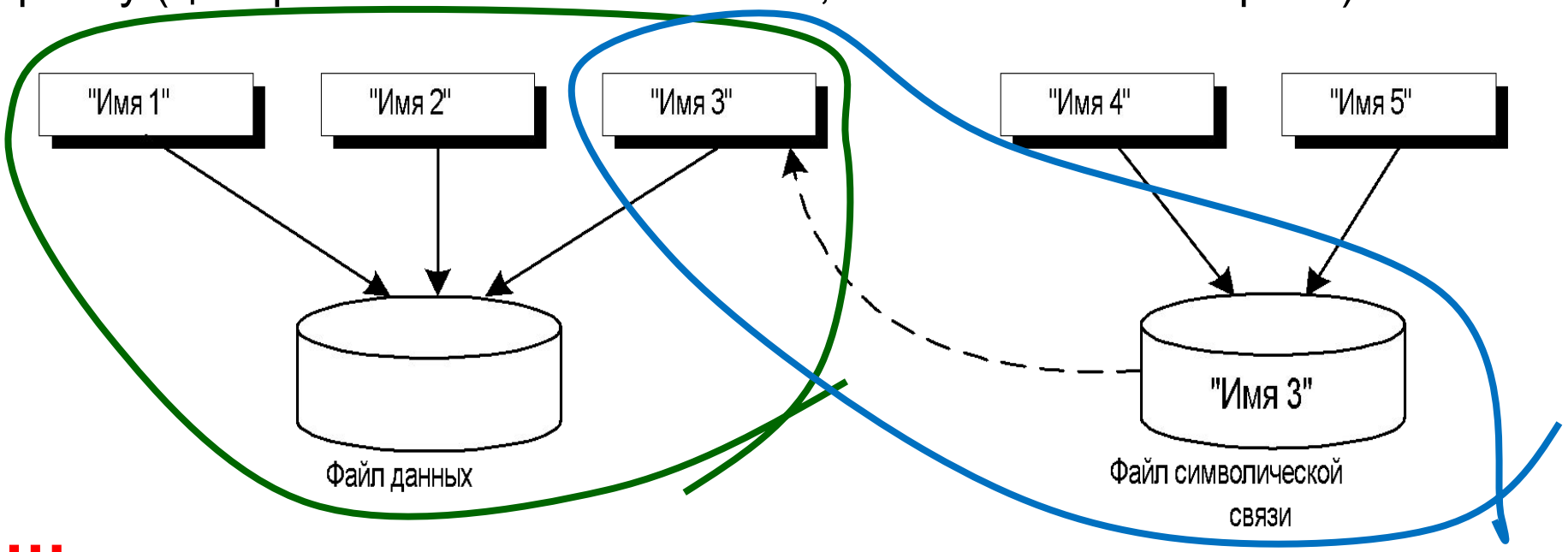
Жорсткі і символічні зв’язки

Hard Links

- один файл може мати декілька імен в одному каталозі або в декількох
- команда ***link*** – для створення жорсткого зв’язку.

Symbolic Links

Сиволічний зв’язок – це файл, який містить в собі повне ім’я іншого файлу (цей файл помічений як зв’язок, а не як текстовий файл)



!!! Жорсткий зв’язок вказує на сам файл, а символічний – на ім’я файлу

Монтування дисків

Сутність:

даний диск відображається на якийсь з каталогів основного тому (для цього використовуються порожні підкаталоги каталога ***/mount*** або ***/mnt***

Unix: монтування тому - ніби «щеплення» одного дерева до якого-небудь місця на іншому, основному дереві.

MS-DOS, Windows: допускають використання декількох окремих дерев.

Типи и атрибути файлів

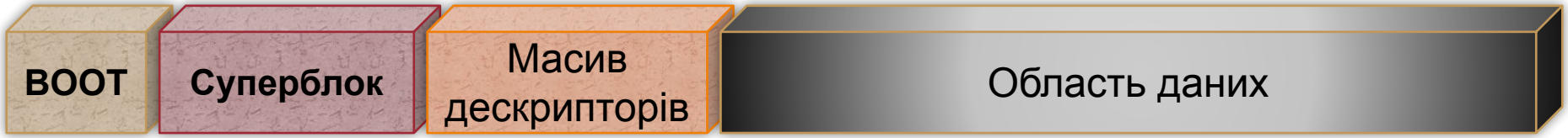
Тип файлу:

- – звичайний файл з даними
d – каталог;
c – символьний спеціальний файл – символьний пристрій
b – блочний спеціальний файл
l – символічний зв'язок
p – іменований канал
s – сокет – об'єкт для передачі даних по мережі

Атрибути:

- розмір в байтах
- кількість жорстких зв'язків
- «часові штампи»:
  дата/час останнього доступу до файлу,
  останньої модифікації файлу,
  останньої модифікації атрибутів файлу («дата створення файлу».
- атрибути доступу
  UID – власник (числовий ідентифікатор)
  GID – група-власник (числовий ідентифікатор)
- атрибути захисту: **r w x r - x - - x**
- інформація про розміщення файлу

Структура тому ОС UNIX



BOOT-сектор: його структура визначається не UNIX, а архітектурою комп'ютера;

суперблок - містить основні відомості про дисковий том:

- розмір логічного блоку
- кількість блоків
- розміри основних областей
- тип файлової системи
- можливі режими доступу
- дані про вільне місце на диску;

масив індексних дескрипторів – відомості про один з файлів, що зберігаються на диску (окрім імені файлу);

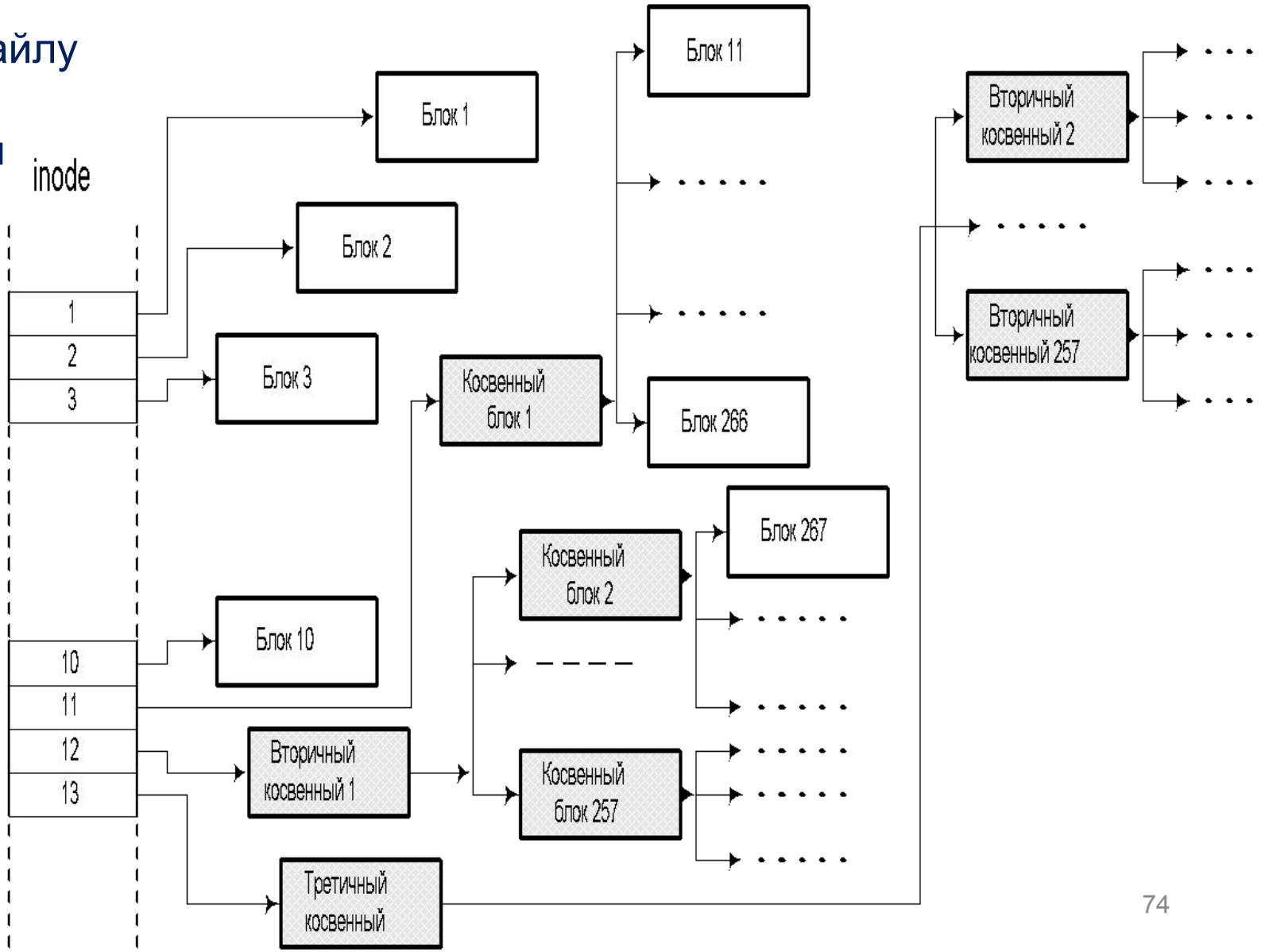
область даних – складається з логічних блоків (кластерів), які використовуються для зберігання файлів і каталогів (в UNIX використовується сегментоване розміщення файлів).

Каталоги Unix

- Каталоги не містять ніякої інформації про файл, окрім:
  - імені файлу
  - індексного дескриптора
- Кожний запис каталогу має змінний розмір (імена різної довжини)
- В кожному каталозі перших 2 записи (як в FAT):
  - «..» посилання на батьківський каталог
  - «.» посилання на даний каталог
- Запис з номером 0 – відповідає видаленому запису (файлу)

Індексний дескриптор (inode)

- лічильник жорстких зв’язків файлу
- тип файлу
- атрибути файлу
- дані про розміщення файлу



Інформація про вільний простір

- **Вільні дескриптори** – беруться з тих, що позначені 0, або з суперблоку (містить масиви для зберігання деякої кількості вільних блоків)
- **Вільні блоки даних**