

# Systemy plików: struktura partycji i właściwości FAT16, FAT32, exFAT oraz NTFS, ext2, ext3, ext4, Btrfs, XFS, ReiserFS

Przegląd typów systemów plików i ich cech technicznych

# Struktura partycji



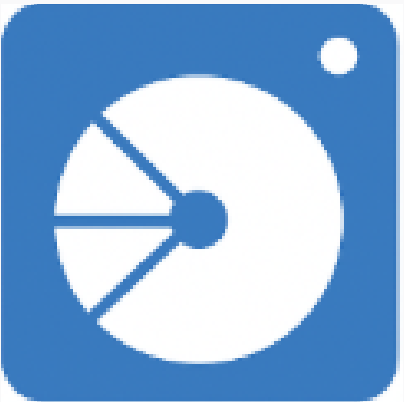
## Charakterystyka MBR

MBR obsługuje do czterech partycji z maksymalnym rozmiarem 2 TB i zawiera kod rozruchowy oraz sygnaturę.



## Cechy GPT

GPT obsługuje do 128 partycji, rozmiary powyżej 9 ZB, kopię zapasową tablicy i sumy kontrolne CRC.



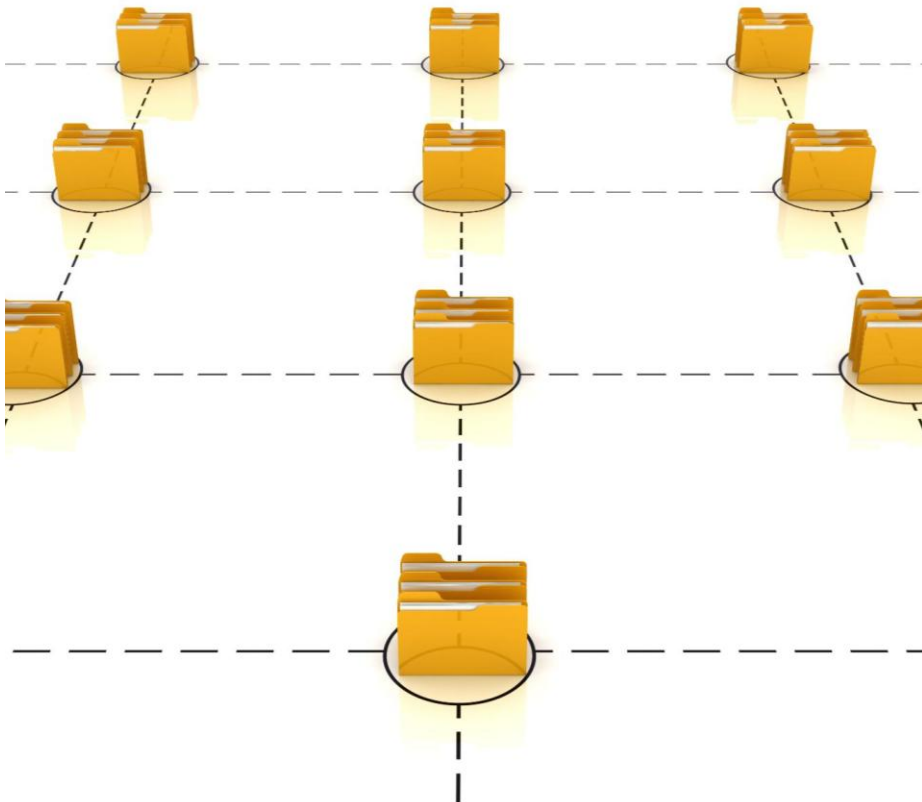
## Wybór schematu partycji

Wybór między MBR a GPT zależy od kompatybilności sprzętowej i wymagań dotyczących pojemności oraz bezpieczeństwa.

# Systemy plików

## FAT

# FAT16 – właściwości techniczne



## Prosta struktura systemu

FAT16 charakteryzuje się prostą strukturą i łatwą implementacją w systemach.

## Ograniczenia rozmiaru partycji

Obsługuje partycje do 2 GB i klastry od 2 KB do 64 KB, co ogranicza pojemność.

## Brak mechanizmu journalingu

Nie posiada journalingu ani systemu uprawnień, co zwiększa ryzyko utraty danych.

## Zastosowania w starszych systemach

Stosowany głównie w starszych systemach i urządzeniach wbudowanych ze względu na prostotę.

# FAT

## FAT – File Allocation Table

- **FAT** to jeden z najstarszych systemów plików, opracowany przez **Microsoft** w **1977 roku** przez Billa Gatesa i Marca McDonalda. [\[dosdays.co.uk\]](http://dosdays.co.uk)
- Początkowo przeznaczony dla **dyskietek** i systemu **Disk BASIC**, później zaadaptowany do **MS-DOS** i **Windows 9x**.
- Nazwa pochodzi od **tablicy alokacji plików (File Allocation Table)** – struktury, która przechowuje informacje o rozmieszczeniu plików na partycji.

## Zastosowanie FAT

- Stosowany w systemach:
  - **DOS**
  - **Windows 3.x / 95 / 98 / ME**
  - **Windows NT/2000** (częściowo)
- Obecnie używany głównie na:
  - **pendrive'ach**
  - **kartach pamięci (SD, MMC)**
  - **urządzeniach wbudowanych** – ze względu na prostotę i szeroką kompatybilność.
  - **systemy multimedialne** – np. odtwarzacze MP3, samochodowe systemy audio.

## Nieaktualne zastosowania

- **Dyskietki** – praktycznie nieużywane.
- **Stare wersje FAT (FAT12, FAT16)** – obecnie rzadko spotykane, głównie w muzealnych lub specjalistycznych zastosowaniach.

# Typy FAT

## Typy FAT – różnice techniczne

Podstawowa różnica między wersjami FAT polega na **liczbie bitów używanych do kodowania numerów klastrów** (jednostek alokacji plików). To właśnie ta liczba bitów jest zawarta w nazwie systemu:

| Typ FAT | Liczba bitów na wpis w tablicy FAT | Zakres klastrów  | Maks. rozmiar partycji                  | Maks. rozmiar pliku | Obsługa długich nazw        | Typowe zastosowanie                        |
|---------|------------------------------------|------------------|-----------------------------------------|---------------------|-----------------------------|--------------------------------------------|
| FAT12   | 12 bitów                           | < 4 087          | ~32 MB                                  | ~16 MB              | Nie                         | Dyskietki (historyczne)                    |
| FAT16   | 16 bitów                           | 4 087 – 65 526   | do 2 GB (do 4 GB z rozszerzeniem)       | 2 GB                | Nie                         | Starsze dyski, niektóre pendrive’y         |
| FAT32   | 32 bitów (używane 28 bitów)        | 65 526 – 268 mln | do 2 TB (lub więcej z małymi sektorami) | 4 GB – 1 B          | Tak (LFN – Long File Names) | Pendrive’y, karty SD, urządzenia przenośne |
| exFAT   | 64 bitów                           | do 2^32 klastrów | do 128 PB                               | do 16 EB            | Tak (Unicode)               | SDXC, SSD, nowoczesne pamięci flash        |

# Typy FAT

## Dlaczego to ważne?

- Liczba bitów określa **maksymalną liczbę klastrów**, a co za tym idzie - **rozmiar partycji** i **wydajność** systemu plików.
- FAT12 był używany głównie na **dyskietkach**.
- FAT16 – na **starszych dyskach twardych**.
- FAT32 – nadal używany na **pendrive’ach, kartach SD, urządzeniach przenośnych**.
- exFAT** (64-bitowy) – nowoczesna wersja dla dużych plików i nośników flash.

Dlaczego liczba bitów ma znaczenie w systemach plików FAT?

| System plików | Liczba bitów | Zastosowanie                               | Maks. rozmiar partycji                  | Uwagi                          |
|---------------|--------------|--------------------------------------------|-----------------------------------------|--------------------------------|
| FAT12         | 12-bitowy    | Dyskietki                                  | ~32 MB                                  | Przestarzały                   |
| FAT16         | 16-bitowy    | Starsze dyski HDD                          | ~2 GB (z rozszerzeniem do 4 GB)         | Ograniczenia w liczbie plików  |
| FAT32         | 32-bitowy    | Pendrive’y, karty SD, urządzenia przenośne | ~2 TB (z ograniczeniem do 4 GB na plik) | Szeroko stosowany              |
| exFAT         | 64-bitowy    | Duże pliki, nowoczesne nośniki flash       | ~128 PB                                 | Brak ograniczenia 4 GB na plik |

# FAT

## **Sektor w systemie FAT**

- **Sektor** to najmniejsza jednostka danych na nośniku (np. dysku, pendrive'ie).
- Standardowa wielkość sektora to **512 bajtów** (0,5 kB), choć nowoczesne dyski mogą używać większych sektorów (np. 4 kB – tzw. Advanced Format).
- Sektor jest zawsze **odczytywany i zapisywany w całości**.
- Współczesne systemy operacyjne i urządzenia nadal operują na sektorach 512 bajtowych, nawet jeśli logicznie traktują większe jednostki jako jeden sektor. [\[pjrc.com\]](http://pjrc.com)

## **Klaster w systemie FAT**

- **Klaster** to jednostka alokacji plików – czyli najmniejszy fragment przestrzeni dyskowej, który może być przypisany do pliku.
- Klaster składa się z **jednego lub więcej sektorów** – np. 8, 16, 32, 64 sektorów.
- W FAT32 spotykane są klastry od 4 kB do nawet 32 kB (czyli 8–64 sektorów). [\[cislinux2....shburn.edu\]](http://cislinux2....shburn.edu)
- **Numeracja klastrów** pozwala systemowi plików śledzić, gdzie znajdują się dane danego pliku.

## **Adresowanie danych w FAT**

- W przeszłości używano adresowania fizycznego: **Cylinder / Head / Sector (CHS)**.
- Obecnie stosuje się **LBA (Logical Block Addressing)** – sektory są numerowane logicznie od zera.
- System operacyjny przelicza numer klastra na numer sektora, a następnie lokalizuje dane na nośniku.

## **Zjawisko slack space**

- Jeśli plik zajmuje mniej niż cały klaster, **pozostała przestrzeń w klastrze jest niewykorzystana** – tzw. slack space.
- Przykład: plik o rozmiarze 1 bajta na klastrze 32 kB zajmie fizycznie 32 kB. [\[cislinux2....shburn.edu\]](http://cislinux2....shburn.edu)

# FAT

## ♦ Struktura tablicy FAT

- Tablica FAT to **lista powiązanych klastrów** – tworzy coś w rodzaju **łańcucha**.
- W katalogu pliku zapisany jest **numer pierwszego klastra**.
- W tablicy FAT pod tym numerem znajduje się **numer kolejnego klastra**, który zawiera dalszą część pliku.
- Proces powtarza się aż do napotkania **znacznika końca pliku (EOF)**, np. `0xFFFF` dla FAT16 lub `0xFFFFFFFF` dla FAT32. [\[www.on-time.com\]](http://www.on-time.com)

## ♦ Wartości w tablicy FAT

- **0x00000000** – wolny klaster.
- **0x00000001 – 0x000000FF** – zarezerwowane.
- **Numer kolejnego klastra** – jeśli plik zajmuje więcej niż jeden klaster.
- **Wartość EOF** – oznacza koniec pliku.
- **Wartość oznaczająca uszkodzony klaster** – np. `0xFFFFFFFF7`. [\[www.on-time.com\]](http://www.on-time.com)

## 🧠 Działanie systemu FAT – przykład

1. Plik `dane.txt` zaczyna się w klastrze **100**.
2. W tablicy FAT pod pozycją **100** znajduje się wartość **101** → kolejny klaster.
3. W pozycji **101**  
→ wartość **102**.
4. W pozycji **102**  
→ wartość **110**.
5. W pozycji **110**  
→ wartość **EOF**  
→ koniec pliku.

| Klaster                | FAT12       | FAT16           | FAT32                  |
|------------------------|-------------|-----------------|------------------------|
| Wolny                  | 0x000       | 0x0000          | 0x?0000000             |
| Zarezerwowany          | 0x001       | 0x0001          | 0x?0000001             |
| Używany                | 0x002–0xfeF | 0x0002–0xffef   | 0x?0000002–0x?ffffffef |
| Zarezerwowane wartości | 0xff0–0xff6 | 0xffff0–0xffff6 | 0x?ffffff0–0x?ffffff6  |
| Uszkodzony             | 0xff7       | 0xffff7         | 0x?ffffff7             |
| Ostatni                | 0xff8–0xfff | 0xffff8–0xfffff | 0x?ffffff8–0x?fffffff  |

# FAT



## FAT12 – przykład obliczenia wielkości klastra

### ♦ Założenia

- Dysk o pojemności **20 MB**.
- Standardowy rozmiar sektora: 512 bajtów.
- Liczba sektorów:

$$\frac{20 \text{ MB}}{512 \text{ B}} = \frac{20 \times 1024 \times 1024}{512} = 40960 \text{ sektorów}$$

### ♦ FAT12 – ograniczenia

- FAT12 używa **12-bitowych wpisów** w tablicy FAT.
- Maksymalna liczba klastrów: **4096** (czyli  $2^{12}$ , (ale:
  - **Kilka klastrów jest zarezerwowanych** (np. dla katalogu głównego, tablic FAT, itp.).
  - W praktyce dostępnych jest **mniej niż 4086 klastrów**. [\[www.on-time.com\]](http://www.on-time.com)

### ♦ Obliczenie wielkości klastra

Aby zmieścić 40960 sektorów w mniej niż 4086 klastrach:

$$\frac{40960 \text{ sektorów}}{4086 \text{ klastrów}} \approx 10 \text{ sektorów na klaster}$$

- Klaster = 10 sektorów × 512 B = 5120 B = 5 kB**

## Wnioski

- Dla dysku 20 MB w systemie FAT12, **rozmiar klastra musi wynosić co najmniej 5 kB**, aby liczba klastrów nie przekroczyła limitu FAT12.
- Liczba dostępnych klastrów** = maksymalna liczba plików, które można zapisać na partycji (jeśli każdy plik zajmuje jeden klaster).
- Im większy klaster**, tym mniej plików można zapisać, ale mniejsze zużycie tablicy FAT.

# Bloki partycji FAT

## Struktura partycji FAT – 4 główne regiony

System plików FAT (FAT12, FAT16, FAT32) dzieli partycję na **cztery logiczne obszary**:

### **1 Obszar zarezerwowany (Reserved Region)**

- Zawiera **Boot Sector** (sektor rozruchowy), czyli **Volume Boot Record (VBR)**.
- W boot sektorze znajduje się:
  - **BPB (BIOS Parameter Block)** – zawiera informacje o strukturze partycji: rozmiar sektora, liczba sektorów na klaster, liczba kopii FAT, lokalizacja katalogu głównego itd.
  - **Boot loader** – program ładujący system operacyjny (jeśli partycja jest rozruchowa).
- Może zawierać dodatkowe sektory informacyjne (np. FSInfo w FAT32).

### **2 Tablica alokacji plików (FAT Region)**

- Zawiera **tablicę FAT** – strukturę danych przechowującą informacje o stanie każdego klastra:
  - Wolny, zajęty, uszkodzony, koniec pliku (EOF).
  - Łańcuchy klastrów dla plików.
- Zazwyczaj istnieją **dwie kopie tablicy FAT** – dla bezpieczeństwa.
- Tablica FAT znajduje się **bezpośrednio za obszarem zarezerwowanym**.

### **3 Katalog główny (Root Directory Region)**

- W **FAT12/16** – znajduje się **w stałym miejscu** za tablicą FAT.
- Zawiera wpisy katalogowe:
  - Nazwa pliku (8.3 lub LFN), atrybuty, daty, wskaźnik na pierwszy klaster.
- W **FAT32** – katalog główny jest traktowany jak zwykły katalog i znajduje się w **regionie danych**.

# Bloki partycji FAT

## 📁 Struktura partycji FAT – 4 główne regiony

System plików FAT (FAT12, FAT16, FAT32) dzieli partycję na **cztery logiczne obszary**:

### ❷ Obszar danych (Data Region)

- Zajmowany przez **pliki i katalogi**.
- Podzielony na **klastry** – jednostki alokacji.
- System operacyjny korzysta z tablicy FAT, aby odnaleźć kolejne klastry pliku.

## Struktura partycji FAT

- Boot Sector z BPB i bootloaderem
- Tablica FAT (zwykle 2 kopie)
- Katalog główny (dla FAT12/16)
- Region danych (klastry z plikami i katalogami)



# Bloki partycji FAT



**Bootsector** - to zawsze pierwszy sektor partycji, zawiera podstawowe informacje o partycji, między innymi:

- wielkość partycji
- wielkość sektora (w bajatach)
- ilość sektorów w klastrze
- typ partycji (np. FAT32)
- ilość kopii tablicy FAT
- wskaźniki do innych danych
- kod ładujący system operacyjny

# Bloki partycji FAT



**Tablica FAT** - jest to struktura danych w której przechowywane są informacje o położeniu pliku na dysku. Każdemu klastrowi odpowiada dokładnie jeden wpis w tablicy, ten wpis to zazwyczaj informacja o tym że klaster jest jakiś specjalny bądź jest to numer klastra zawierający ciąg dalszy pliku. Szczegóły pokazane w tabelce poniżej:

| FAT12             | FAT16              | FAT32                   | Znaczenie                                                                                         |
|-------------------|--------------------|-------------------------|---------------------------------------------------------------------------------------------------|
| 0x000             | 0x0000             | 0x00000000              | Wolny klaster                                                                                     |
| 0x001             | 0x0001             | 0x00000001              | Zarezerwowany klaster                                                                             |
| 0x002 -<br>0xFE0  | 0x0002 -<br>0xFF0  | 0x00000002 -<br>0xFFFF0 | Używany klaster, gdy nie jest ostatnim klastrem pliku, wartość wskazuje na następny klaster pliku |
| 0xFF0 -<br>0xFF6  | 0xFFF0 -<br>0xFFF6 | 0xFFFF0 -<br>0xFFFF6    | Wartości zarezerwowane                                                                            |
| 0xFF7             | 0xFFF7             | 0xFFFF7                 | Uszkodzony klaster                                                                                |
| 0xFF8 -<br>0xFFFF | 0xFFF8 -<br>0xFFFF | 0xFFFF8 -<br>0xFFFFF    | Koniec pliku                                                                                      |

# Bloki partycji FAT



**Kopia FAT** - ze względu na dużą wartość informacji przechowywanych w tablicy warto zabezpieczyć się na wypadek zniszczenia oryginalnej tablicy (lub jej części) poprzez duplikację danych umieszczonych na niej. Niestety rozwiązanie nie zapewnia należytego bezpieczeństwa z co najmniej dwóch powodów. Ze względu na to że obie tablice są umieszczone obok siebie w przypadku wykrycia błąd sektorów w jednej jest duża szansa, że uszkodzona jest też kopia, ponadto usterka może być zduplikowana przed wykryciem błędu.

**Root directory** - jest to katalog główny systemu plików, występuje w ustalonym miejscu w partycji. W FAT12 i FAT16 ma sztywno ustaloną wielkość - mieści do 512 wpisów (przy długich nazwach jeszcze mniej). W FAT32 nie ma tego ograniczenia.

# Bloki partycji FAT



**Obszar danych** - pliki (i katalogi) umieszczone w klastrach jeden lub więcej klastrów przypadających na plik. Reprezentowane są jako ciąg bajtów, jeśli nie starczy klastra kolejne są w następnym itd. Za poprawność operacji następnika odpowiedzialna jest tablica FAT. Katalogi są to pliki specjalnej postaci. Katalog jest podzielony na małe struktury nazywane wpisami. Każde wpisy ma rozmiar 32 bajtów i jest informacją o katalogu bądź pliku, zawartym w naszym katalogu.

# Bloki partycji FAT

## Obszar danych - Wpis zawiera pola:

- 8 bajtów - nazwa pliku, (literki, cyferki i \$ % @ ' - \_ ( ) ~ ! # ^ & później również + , ; = [ ])
- 3 bajty - rozszerzenie,
- 1 bajt - atrybuty pliku:
  - read-only, tylko do odczytu
  - hidden, ukryty
  - system, plik systemowy
  - volume label, tak jest oznaczony jeden plik w katalogu głównym, jego nazwa jest nazwą dysku
  - subdirectory, czy jest to katalog
  - archive, plik, który trzeba skopiować przy następnej kopii zapasowej
  - device, plik reprezentuje urządzenie
  - unused, nie używane
- 1 bajt - wielkość liter nazwy i rozszerzenia pliku,
- 1 bajt - czas utworzenia w milisekundach,
- 2 bajty - czas utworzenia,
- 2 bajty - data utworzenia,
- 2 bajty - czas ostatniego dostępu,
- 2 bajty - zarezerwowane,
- 2 bajty - czas utworzenia lub ostatniej modyfikacji pliku,
- 2 bajty - data utworzenia lub ostatniej modyfikacji pliku,
- 2 bajty - numer klastra gdzie rozpoczyna się plik,
- 4 bajty - rozmiar pliku.

# Bloki partycji FAT



## Nazwy plików

Powyższy spis sugeruje, że nazwy plików zawierają mogą się składać tylko z 8 znaków + 3 literowe rozszerzenie, tak było na początku i tak jest np. w MS-DOS.

Mankament ten usunięto, jednakże na tyle ostrożnie by zachować wsteczną poprawność.

Tworzy się alias, w którym nazwa pliku przypomina nazwę oryginalną, a cała nazwa jest zapisywana w specjalnym katalogu o atrybutach (read-only, system, volume label, hidden).

Stare systemy nie przejmują się tymi ekstra katalogami i obowiązują zaliasowane nazwy.

# FAT / Zalety i wady tablicy alokacji

## ✓ Zalety systemu FAT i tablicy alokacji

- 1.Prosta struktura** – FAT jest lekki, łatwy do zaimplementowania i obsługiwany przez niemal wszystkie systemy operacyjne (Windows, Linux, macOS, Android, konsole). [\[includehelp.com\]](https://includehelp.com)
- 2.Szybkie odnajdywanie danych** – tablica FAT umożliwia szybkie odnalezienie pierwszego klastra pliku i kolejnych jego części.
- 3.Dwa egzemplarze tablicy FAT** – zwiększają odporność na uszkodzenia (druga kopia może być użyta w razie awarii pierwszej). [\[learn.microsoft.com\]](https://learn.microsoft.com)
- 4.Szeroka kompatybilność** – idealny dla pendrive'ów, kart pamięci, urządzeń wbudowanych.
- 5.Łatwość odzyskiwania danych** – prosta struktura ułatwia odzyskiwanie plików po awarii.

## ✗ Wady systemu FAT i tablicy alokacji

- 1.Brak wsparcia dla praw dostępu i szyfrowania** – FAT nie obsługuje mechanizmów bezpieczeństwa znanych z NTFS czy ext4. [\[pcpartsgeek.com\]](https://pcpartsgeek.com)
- 2.Fragmentacja danych** – pliki mogą być rozproszone po całej partycji, co prowadzi do spowolnienia odczytu/zapisu i konieczności defragmentacji. [\[includehelp.com\]](https://includehelp.com)
- 3.Slack space** – duże klastry na dużych partycjach powodują marnowanie miejsca (średnio pół klastra na plik).
- 4.Wrażliwość na awarie** – każda operacja (tworzenie, kasowanie, zmiana pliku) wymaga aktualizacji tablicy FAT. Jej uszkodzenie może prowadzić do **utraty wszystkich danych**. [\[learn.microsoft.com\]](https://learn.microsoft.com)
- 5.Brak organizacji katalogów** – pliki są zapisywane w pierwszym wolnym miejscu, co utrudnia zarządzanie.
- 6.Ograniczenie rozmiaru pliku** – maksymalny rozmiar pliku w FAT32 to **4 GB - 1 bajt**, co uniemożliwia zapis dużych plików (np. obrazów ISO DVD) bez dzielenia. [\[profolus.com\]](https://profolus.com)

# FAT32 – rozszerzenia i ograniczenia

## **Rozszerzenie FAT16**

FAT32 to ulepszenie FAT16, umożliwiające obsługę większych partycji i plików.

## **Większe partycje i pliki**

FAT32 obsługuje partycje do 32 GB i pliki do 4 GB, co zwiększa możliwości przechowywania.

## **Brak zabezpieczeń**

FAT32 nie obsługuje uprawnień ani journalingu, ograniczając bezpieczeństwo danych.

## **Zastosowanie w USB i SD**

FAT32 jest popularny w pamięciach USB i kartach SD ze względu na prostotę i kompatybilność.



# exFAT – nowoczesny FAT

## Optymalizacja dla pamięci flash

exFAT jest zoptymalizowany specjalnie dla pamięci flash, poprawiając wydajność zapisu i odczytu.

## Brak ograniczenia rozmiaru pliku

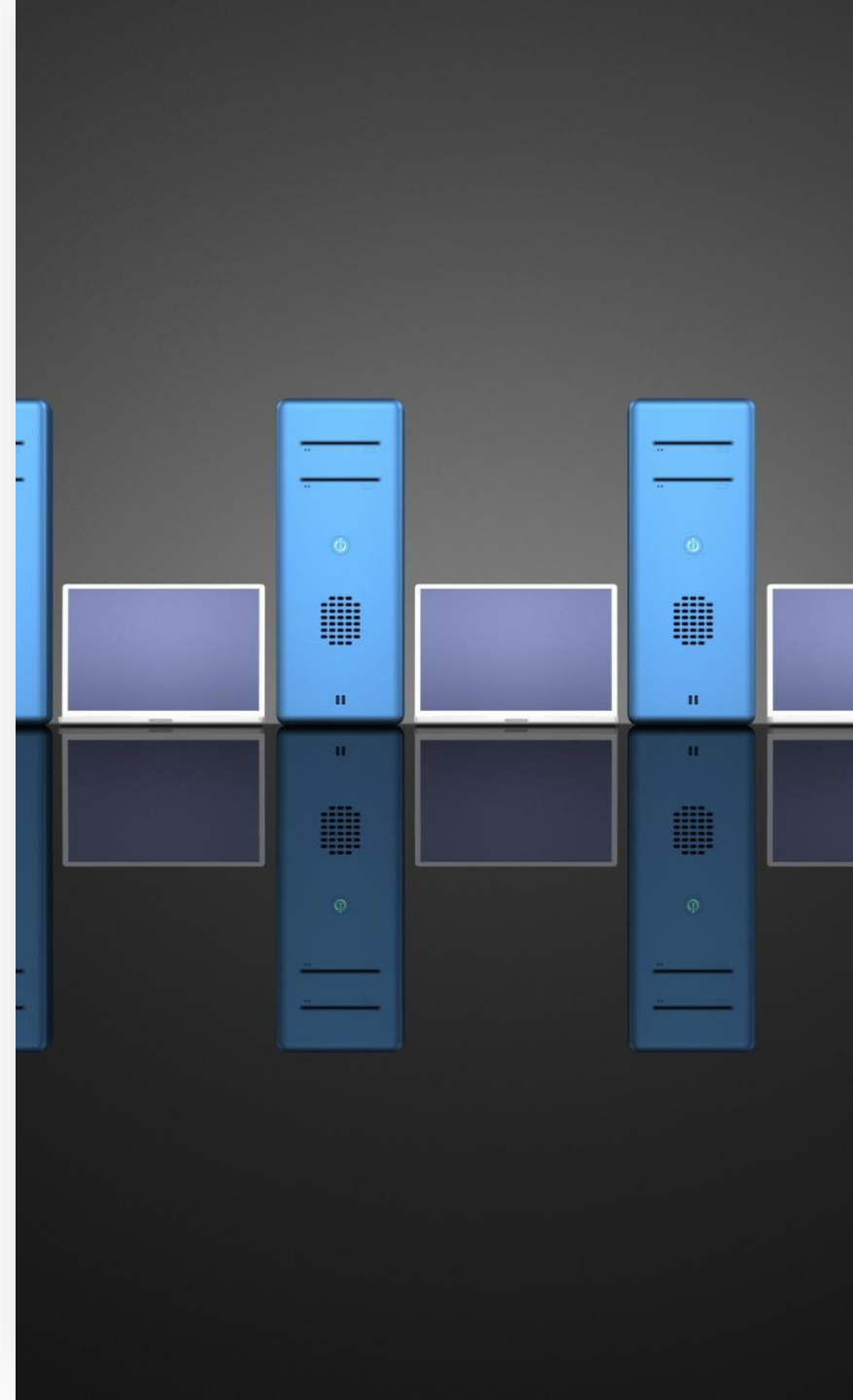
exFAT usuwa limit 4 GB na plik, umożliwiając przechowywanie dużych plików multimedialnych.

## Wsparcie wieloplatformowe

System obsługiwany jest przez Windows, macOS oraz nowoczesne jądra Linux od wersji 5.4.

## Idealny dla urządzeń mobilnych

Prostota i wydajność exFAT czynią go idealnym dla urządzeń przenośnych i mobilnych.



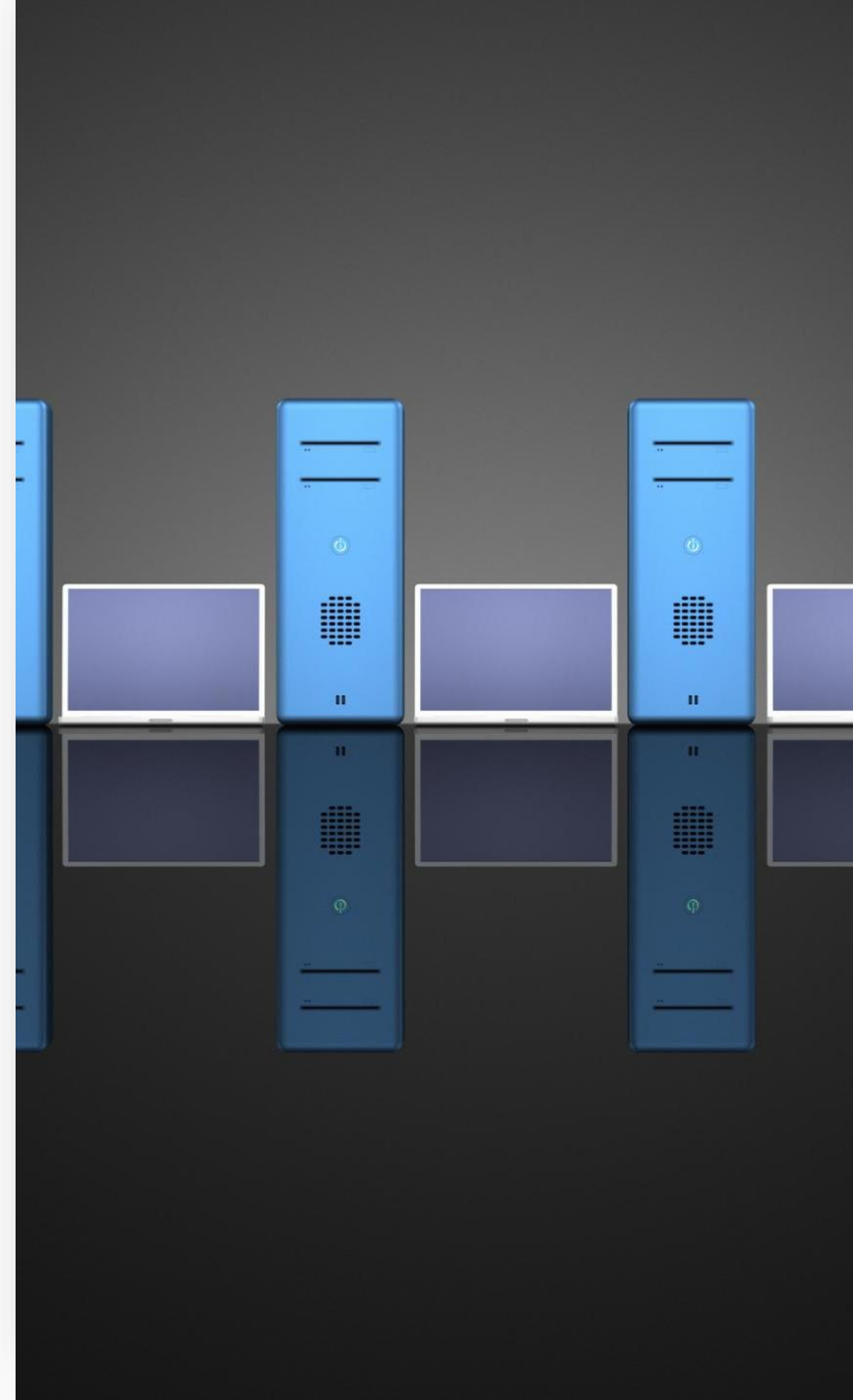
# exFAT – nowoczesny FAT

## Typowe zastosowania exFAT

**Nowoczesne pendrive'y i dyski SSD** – obsługa dużych plików (np. wideo 4K).

**Karty SDXC** – standardowo formatowane w exFAT.

**Współpraca między systemami** – Windows, macOS, Linux (od wersji 5.4 z natywnym wsparciem).



# System plików NTFS

# NTFS – architektura i funkcje

## **Zaawansowana struktura plików**

NTFS używa tabeli MFT do zarządzania wszystkimi plikami i katalogami systemu.

## **Obsługa dużych plików**

System pozwala na pliki i partycje o rozmiarze do 256 TB, idealny dla dużych danych.

## **Funkcje bezpieczeństwa**

NTFS oferuje ACL, szyfrowanie EFS oraz przydziały dyskowe dla ochrony danych.

## **Obsługa transakcji i linków**

Wspiera transakcyjność plików, punkty montowania oraz linki symboliczne dla elastyczności.



# NTFS – New Technology File System

## Dlaczego FAT przestał wystarczać?

System plików **FAT** był prosty i skuteczny w czasach, gdy nośniki danych miały niewielką pojemność, a wymagania użytkowników były ograniczone. Jednak z biegiem lat pojawiły się **nowe potrzeby**, którym FAT nie był w stanie sprostać:

### **Nowe wymagania wobec systemów plików**

- Obsługa **dużych plików** (powyżej 4 GB).
- **Zarządzanie uprawnieniami** i bezpieczeństwem (np. prawa dostępu, szyfrowanie).
- **Odporność na awarie** – np. journaling, ochrona przed utratą danych.
- **Wydajność** przy pracy z dużą liczbą plików i katalogów.
- Obsługa **metadanych** – daty utworzenia, modyfikacji, właściciel, atrybuty.
- **Skalowalność** – możliwość pracy na partycjach o rozmiarze wielu terabajtów.

### **Dlaczego nie rozbudowano FAT?**

- FAT miał **zbyt prostą strukturę**, która nie pozwalała na łatwe dodanie nowych funkcji.
- Rozbudowa FAT groziła **utrata kompatybilności** z istniejącymi systemami.
- Zamiast modyfikować FAT, zdecydowano się na **stworzenie nowego systemu plików** – np. **NTFS** dla Windows, **ext3/ext4** dla Linuxa, **APFS** dla macOS.

### **Korzyści dla użytkowników**

- **Zwykli użytkownicy** zyskali możliwość:
  - Przechowywania dużych plików (np. filmów HD, obrazów ISO).
  - Bezpieczniejszego przechowywania danych.
  - Lepszej organizacji plików i katalogów.
  - Szybszego dostępu do danych dzięki mniejszej fragmentacji.

# NTFS ograniczenia

## NTFS – ograniczenia rozmiaru pliku i partycji

### Maksymalny rozmiar pliku

- Teoretycznie:** do **16 EiB** (eksbibajtów), czyli  $2^{64}$  bajtów minus 1 KB.
- W praktyce (Windows):** ograniczenie do **256 TiB** (tebibajtów), zależnie od wersji systemu i rozmiaru klastra (np. 64 KB). [\[datto.com\]](https://datto.com)

### Maksymalny rozmiar partycji

- Teoretycznie:** do  $2^{64}$  klastrów.
- W praktyce (Windows):** do **256 TiB** – ograniczenie wynika z implementacji stosu systemowego w Windows. [\[superuser.com\]](https://superuser.com)

### Dlaczego te ograniczenia istnieją?

- NTFS został zaprojektowany z myślą o dużych wolumenach, ale **implementacje systemowe** (np. w Windows XP, 10, 11, Server) ograniczają maksymalne wartości.
- Rozmiar klastra (np. 4 KB, 64 KB) wpływa na maksymalną liczbę adresowalnych danych.
- Wersje 64-bitowe Windows pozwalają na większe partycje niż wersje 32-bitowe.

## Podsumowanie

| Parametr               | Teoretyczny limit | Praktyczny limit (Windows) |
|------------------------|-------------------|----------------------------|
| Rozmiar pliku          | 16 EiB            | 256 TiB                    |
| Rozmiar partycji       | $2^{64}$ klastrów | 256 TiB                    |
| Liczba plików na dysku | 4 294 967 295     | 4 294 967 295              |

# NTFS ograniczenia

## Wyjaśnienie tabeli NTFS – ograniczenia teoretyczne vs praktyczne

| Parametr               | Teoretyczny limit        | Praktyczny limit (Windows) | Wyjaśnienie                                                                                                                                             |
|------------------------|--------------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Rozmiar pliku          | 16 EiB (eksbibajtów)     | 256 TiB (tebibajtów)       | NTFS używa 64-bitowego adresowania, co pozwala na ogromne pliki, ale systemy Windows ograniczają to do 256 TiB ze względów wydajnościowych i zgodności. |
| Rozmiar partycji       | 2 <sup>64</sup> klastrów | 256 TiB                    | Teoretycznie NTFS może obsłużyć partycje o rozmiarze eksabajtów, ale w praktyce Windows ogranicza to do 256 TiB, zależnie od rozmiaru klastra.          |
| Liczba plików na dysku | 4 294 967 295            | 4 294 967 295              | NTFS używa 32-bitowego indeksowania plików, co daje ponad 4 miliardy możliwych plików na jednej partycji.                                               |

### Co oznaczają jednostki?

- TiB (tebibajt)** = 2<sup>40</sup> bajtów = 1 099 511 627 776 bajtów
- EiB (eksbibajt)** = 2<sup>60</sup> bajtów = 1 152 921 504 606 846 976 bajtów

# możliwości NTFS

## NTFS – journaling (księgowanie zmian)

### Czym jest journaling w NTFS?

- NTFS od wersji **3.0 (Windows 2000)** wykorzystuje mechanizm **journalingu**, czyli **wewnętrznego dziennika zmian**.
- Zmiany w strukturze systemu plików (np. dodanie pliku, zmiana nazwy, modyfikacja katalogu) są najpierw zapisywane w specjalnym pliku **\$LogFile** – dzienniku transakcji.
- Dzięki temu, w razie awarii (np. utraty zasilania), system może **odtworzyć lub cofnąć** niekompletne operacje, zachowując **spójność metadanych**. [\[aeanet.org\]](http://aeanet.org)

### Zakres ochrony journalingu w NTFS

| Element chroniony                                             | Status                                                                                                          |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <b>Metadane</b> (np. struktura katalogów, prawa dostępu, MFT) |  Chronione przez journaling  |
| <b>Dane użytkownika w plikach</b>                             |  Nie są objęte journalingiem |

- NTFS **nie chroni zawartości plików** – jeśli zapis danych zostanie przerwany (np. przez awarię zasilania), plik może zostać uszkodzony lub utracony.
- Mechanizm journalingu w NTFS jest **metadane-only**, co oznacza, że chroni tylko strukturę systemu plików, a nie same dane. [\[learn.microsoft.com\]](http://learn.microsoft.com)

# możliwości NTFS

## Korzyści journalingu w NTFS

- **Szybsze odzyskiwanie po awarii** – nie trzeba skanować całego dysku (jak w FAT32).
- **Lepsza niezawodność** – zmniejszone ryzyko uszkodzenia struktury plików.
- **Wsparcie dla narzędzi systemowych** – np. **CHKDSK** może szybciej naprawić błędy dzięki dziennikowi zmian.

## Ograniczenia

- **Brak ochrony danych użytkownika** – pliki mogą być częściowo zapisane lub uszkodzone.
- **Brak pełnego journalingu** – w przeciwieństwie do systemów takich jak ext4 (w trybie full journaling), NTFS nie zapisuje danych plików w dzienniku.

# możliwości NTFS

## EFS – Encrypting File System w NTFS

### Podstawowe informacje

- **EFS** to funkcja szyfrowania dostępna od wersji **NTFS 3.0** (czyli od **Windows 2000**).
- Umożliwia szyfrowanie **plików i katalogów** na poziomie systemu plików.
- Szyfrowanie odbywa się za pomocą **nakładek systemowych**, a dane są przechowywane w postaci zaszyfrowanej na dysku.

### Ograniczenia EFS

- **Nie można szyfrować plików systemowych** (np. plików rozruchowych, rejestru).
- **Dane użytkownika są szyfrowane**, ale nie są chronione przed usunięciem – EFS nie jest systemem backupu.
- **EFS nie szyfruje całych dysków** – do tego służy np. BitLocker.

## Dostępność EFS w systemach Windows

| Wersja Windows                   | Dostępność EFS                                      |
|----------------------------------|-----------------------------------------------------|
| Windows 2000                     | ✓ (pierwsza wersja, niekompatybilna z późniejszymi) |
| Windows XP Home                  | ✗ niedostępne                                       |
| Windows XP Professional          | ✓ dostępne                                          |
| Windows Vista Business/Ultimate  | ✓ dostępne                                          |
| Windows 7/8/10/11 Home           | ✗ niedostępne                                       |
| Windows 7/8/10/11 Pro/Enterprise | ✓ dostępne                                          |

# możliwości NTFS

- Od **Windows 11**, EFS jest dostępny **tylko w wersjach Professional i wyższych**.
- Nie jest przeznaczony do szyfrowania prywatnych danych w komputerach domowych** – jego głównym celem jest **ochrona danych w środowiskach korporacyjnych**, szczególnie w **domenach Windows Server** (np. 2003, 2008, 2012, 2016, 2019, 2022, 2025).



## Zgodność wersji

- Wersja EFS z **Windows 2000** jest **niekompatybilna** z późniejszymi wersjami Windows – pliki zaszyfrowane w Windows 2000 mogą nie być dostępne w nowszych systemach bez odpowiednich kluczy.

# możliwości NTFS

## Kompresja danych „w locie” w NTFS

- NTFS umożliwia **kompresję plików i folderów** bez tworzenia archiwów ZIP – dane są kompresowane i dekompresowane automatycznie podczas odczytu/zapisu.
- Kompresja działa na poziomie systemu plików, wykorzystując **algorytmy Lempel-Ziv (LZ)** – bezstratne i szybkie.

[\[elevenforum.com\]](https://elevenforum.com)

### •Wzajemne wykluczanie:

- Plik **nie może być jednocześnie skompresowany i zaszyfrowany** (EFS).
- Dotyczy to również **Windows 11** – próba aktywacji obu funkcji na tym samym pliku zakończy się błędem.

[\[elevenforum.com\]](https://elevenforum.com)

## Prawa dostępu (ACL – Access Control List)

- NTFS obsługuje **zaawansowane prawa dostępu** dla użytkowników i grup:

- **Read, Write, Modify, Full Control.**
- Możliwość **dziedziczenia i audytu** zmian.

### •Pełna funkcjonalność ACL dostępna jest tylko w wersjach:

- **Windows 11 Pro**
- **Enterprise**
- **Education**
- **Windows Server** [\[learn.microsoft.com\]](https://learn.microsoft.com)

### •W Windows 11 Home:

- Można ustawiać podstawowe uprawnienia NTFS.
- Brak dostępu do zaawansowanych funkcji inspekcji i zarządzania (np. audyt, edycja SACL).
- Brak wsparcia dla narzędzi domenowych i polityk grupowych. [\[tomsguide.com\]](https://tomsguide.com)

# możliwości NTFS

## Podsumowanie

| Funkcja                      | Windows 11 Home | Windows 11 Pro/Enterprise |
|------------------------------|-----------------|---------------------------|
| Kompresja NTFS               | ✓               | ✓                         |
| Szyfrowanie EFS              | ✗               | ✓                         |
| Kompresja + EFS jednocześnie | ✗               | ✗                         |
| Zaawansowane ACL             | ✗               | ✓                         |
| Audyty dostępu (SACL)        | ✗               | ✓                         |
| Zastosowanie w domenach      | ✗               | ✓                         |

# możliwości NTFS - transakcyjność

## TxF – Transactional NTFS

### Wprowadzenie

- Mechanizm **TxF (Transactional NTFS)** został wprowadzony w **Windows Vista** jako rozszerzenie NTFS.
- Umożliwia wykonywanie operacji plikowych w sposób **atomowy** – czyli albo cała operacja zakończy się sukcesem, albo zostanie całkowicie wycofana (rollback).
- Transakcje mogą obejmować **wiele plików** i **pozostawać otwarte przez dowolny czas**, co pozwala na złożone operacje na strukturze woluminu.

### Przykładowe operacje objęte transakcją

- Tworzenie pliku
- Zmiana rozmiaru pliku
- Modyfikacja metadanych (np. daty, atrybuty)
- Przenoszenie lub zmiana drzewa katalogów

### Status TxF w nowszych systemach

- Od **Windows 8** i **Windows 10/11**, TxF jest przestarzały (deprecated):
  - Microsoft nie zaleca jego używania w nowych aplikacjach.
  - API TxF nadal istnieje, ale może być **wyłączone lub nieobsługiwane** w przyszłych wersjach Windows.
  - Zamiast TxF zaleca się stosowanie **mechanizmów transakcyjnych na poziomie aplikacji** (np. baz danych, systemów wersjonowania).

### Dlaczego TxF był ważny?

- Zapewniał **spójność danych** w przypadku awarii.
- Ułatwiał tworzenie aplikacji, które musiały modyfikować wiele plików jednocześnie.
- Wspierał **model programowania transakcyjnego**, znany z baz danych.

# możliwości NTFS – transakcyjność

## Mechanizm działania TxF – Transactional NTFS

### Jak działa TxF?

- Transakcje są **optymalizowane** – ich zamknięcie odbywa się szybko i nie obciąża systemu w typowych warunkach.
- Składają się z **podoperacji**, które są grupowane i wykonywane jako **jedna całość**.
- NTFS gwarantuje **spójność danych** – jeśli którakolwiek z podoperacji się nie powiedzie, cała transakcja zostaje **wycofana** (rollback).

### Przykładowe operacje objęte transakcją

- Tworzenie pliku
- Zmiana rozmiaru pliku
- Modyfikacja metadanych (np. daty, atrybuty)
- Zmiana struktury katalogów (np. przenoszenie folderów)

### Status TxF w nowszych systemach

- TxF jest **przestarzały (deprecated)** od **Windows 10**.
- Microsoft **odradza jego użycie** w nowych aplikacjach, mimo że mechanizm nadal istnieje w systemie.
- API TxF może być **wyłączone** w przyszłych wersjach Windows.
- **Windows 11** również zawiera TxF, ale nie jest aktywnie rozwijany ani wspierany.

### Zalecenia dla współczesnych aplikacji

- Zamiast TxF należy stosować:
  - **Bazy danych** (np. SQL Server, SQLite) – z natywnym wsparciem transakcji.
  - **Systemy wersjonowania** (np. Git, SVN).
  - **Własne mechanizmy rollback** – np. tymczasowe pliki, kopie zapasowe.

# NTFS wersje

Numer wersji NTFS jest zapisywany w nagłówku woluminu i odpowiada wersji systemu operacyjnego, w której został użyty. Chronologiczna lista wersji NTFS:

| Wersja NTFS | System operacyjny                                                                                   | Rok        |
|-------------|-----------------------------------------------------------------------------------------------------|------------|
| 1.0         | Windows NT 3.1                                                                                      | 1993       |
| 1.1         | Windows NT 3.5                                                                                      | 1994       |
| 1.2         | Windows NT 3.51, Windows NT 4.0                                                                     | 1995–1996  |
| 3.0         | Windows 2000                                                                                        | 1997       |
| 3.1         | Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows 8, Windows 10, <b>Windows 11</b> | 2001–2021+ |

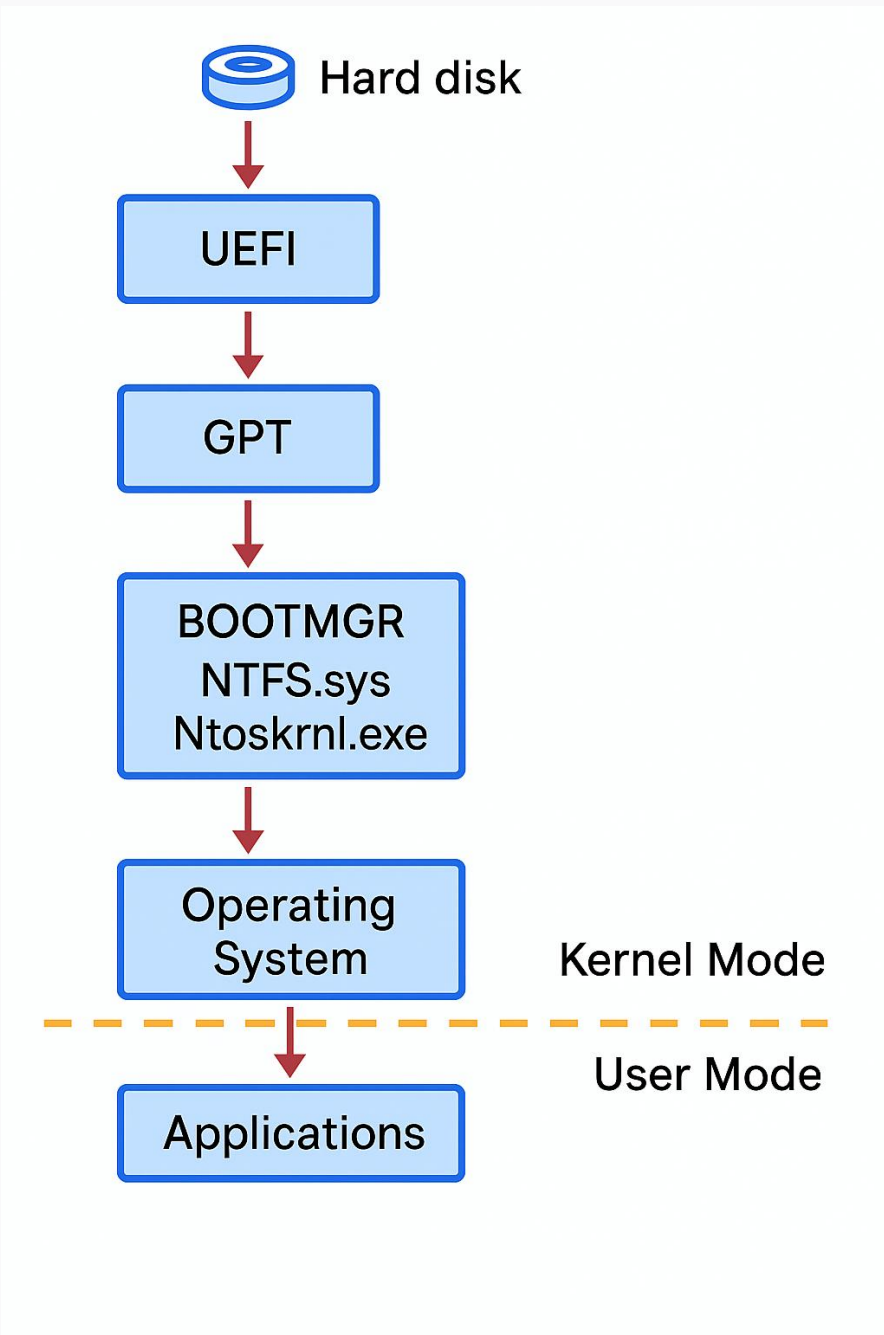
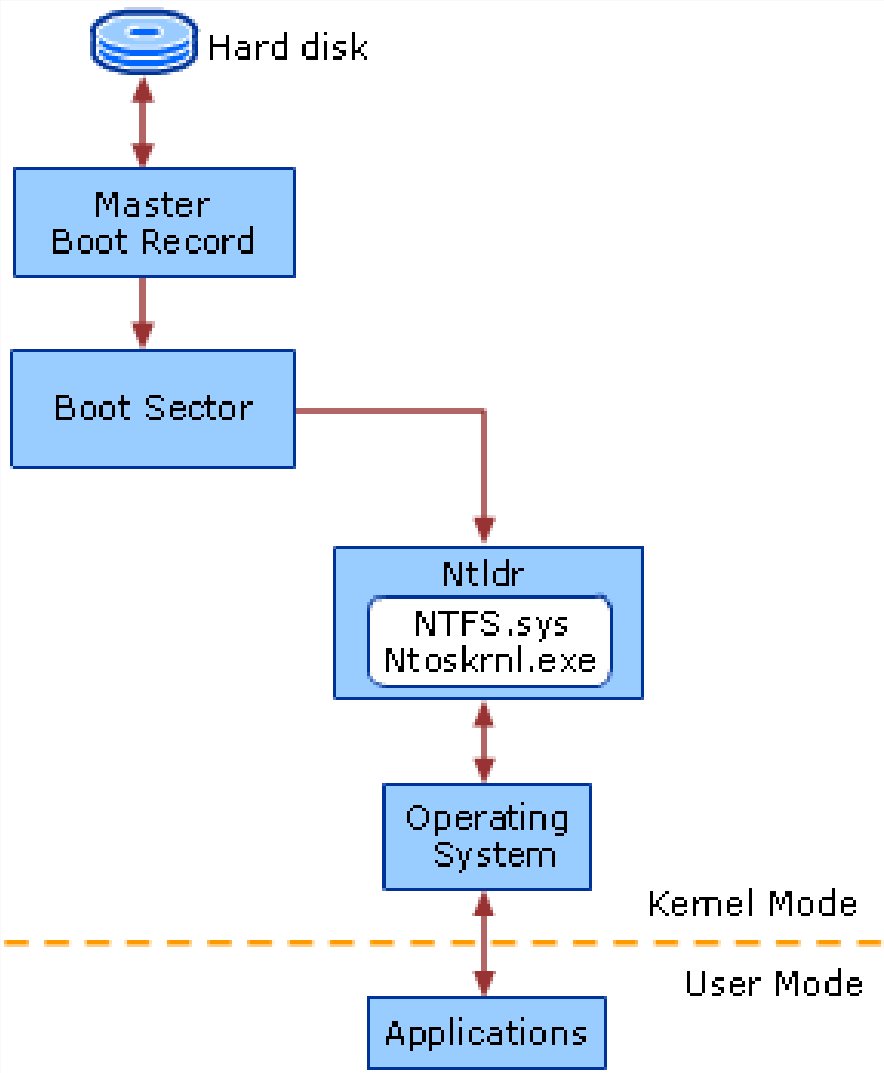
## Uwagi:

- **NTFS 3.1** jest nadal używany w **Windows 11**, mimo że system operacyjny przeszedł wiele zmian - sam format NTFS pozostał stabilny.
- W dokumentacji technicznej Microsoftu stosuje się **numerację NTFS 3.x**, ale w praktyce często spotyka się **nieformalne oznaczenia**:
  - NTFS 5.0 – dla Windows 2000, NTFS 5.1 – dla Windows XP, NTFS 6.x – dla Windows Vista i nowszych.

## Podsumowanie:

- NTFS 3.1 to najnowsza wersja używana w systemach Windows od XP do 11.
- Mimo braku zmiany wersji NTFS, systemy operacyjne wprowadzają nowe funkcje (np. EFS, TxF, kompresja, ACL), które są obsługiwane przez NTFS 3.1.

# Architektura NTFS



# Architektura rozruchu Windows XP / 2000 / NT – komponenty NTFS

## Komponenty rozruchowe:

- **Master Boot Record (MBR)** – ładowany przez BIOS, zawiera kod rozruchowy i tablicę partycji.
- **Boot sektor** – sektor partycji systemowej, zawiera dane o systemie plików i kod ładujący NTLDR.
- **NTLDR (ntldr.dll)** – ładowany z boot sektora, przełącza CPU w tryb chroniony, odczytuje plik **boot.ini** z opcjami rozruchu.

## Komponenty systemowe:

- **ntfs.sys** – sterownik systemu plików NTFS.
- **ntoskrnl.exe** – jądro systemu, ładuje sterowniki i zarządza zasobami systemowymi.

## Tryby pracy:

- **Kernel mode** – pełny dostęp do sprzętu i pamięci, używany przez jądro i sterowniki.
- **User mode** – ograniczony dostęp, używany przez aplikacje użytkownika.

# Architektura rozruchu Windows Vista, 7, 8, 10 i 11 – komponenty NTFS

## Komponenty rozruchowe:

### UEFI + GPT

- **UEFI** zastępuje BIOS – ładuje plik **BOOTMGR** z partycji EFI.
- **GPT (GUID Partition Table)** – nowoczesna struktura partycji, obsługuje wiele partycji bez ograniczeń MBR.

### Boot sektor (NTFS)

- Pierwszy sektor partycji systemowej.
- Zawiera:
  - dane o strukturze systemu plików NTFS (rozmiar klastra, lokalizacja MFT),
  - kod rozruchowy, który przekazuje kontrolę do **BOOTMGR**.

### BOOTMGR + BCD

- **BOOTMGR** – nowoczesny bootloader Windows (od Vista wzwyż).
- **BCD (Boot Configuration Data)** – baza danych z opcjami rozruchu (zastępuje boot.ini).

# Architektura rozruchu Windows 11 – komponenty NTFS

## **Komponenty systemowe:**

### **ntfs.sys**

- Sterownik systemu plików NTFS.
- Obsługuje operacje na plikach, katalogach, metadanych i strukturze MFT.

### **ntoskrnl.exe**

- Jądro systemu Windows.
- Ładuje sterowniki, zarządza pamięcią, procesami i bezpieczeństwem.

## **Tryby pracy:**

### **Kernel Mode**

- Tryb przetwarzania z pełnym dostępem do sprzętu i pamięci.
- Używany przez jądro systemu i sterowniki.

### **User Mode**

- Tryb przetwarzania dla aplikacji użytkownika.
- Ograniczony dostęp do zasobów systemowych – zapewnia izolację i bezpieczeństwo.

# Organizacja na woluminie NTFS

Kolejne slajdy

- Sektor i Klaster NTFS
- NTFS 5.1

# Sektor i Klaster NTFS

## Sektor

- Najmniejsza fizyczna jednostka dysku, na której zapisywane są dane w postaci cyfrowej (0 i 1).
- Standardowa wielkość sektora to **512 bajtów**.
- Sektory są jednostkami fizycznymi, widocznymi dla kontrolera dysku.

## Klaster

- Logiczna jednostka alokacji danych używana przez system plików.
- Klaster może składać się z jednego lub kilku sektorów.
- Tylko klaster o wielkości 512 B** składa się z jednego sektora.
- Przykład: **klaster 4 KB = 8 sektorów po 512 B**.
- System plików zapisuje dane w jednostkach klastrów, co wpływa na wydajność i efektywność wykorzystania przestrzeni dyskowej.

# Sektor i Klaster NTFS

Zależność między wielkością partycji a rozmiarem klastra i systemem plików

| <b>Wielkość partycji</b> | <b>FAT32 – klaster</b> | <b>FAT32 – wielkość FAT</b> | <b>NTFS – klaster</b> |
|--------------------------|------------------------|-----------------------------|-----------------------|
| 7 MB – 32 MB             | nieobsługiwane         | –                           | 512 B                 |
| 33 MB – 64 MB            | 512 B                  | 2 – 4 MB                    | 512 B                 |
| 65 MB – 128 MB           | 1 KB                   | 2 – 4 MB                    | 512 B                 |
| 129 MB – 256 MB          | 2 KB                   | 4 – 8 MB                    | 512 B                 |
| 257 MB – 512 MB          | 4 KB                   | 8 – 16 MB                   | 512 B                 |
| 513 MB – 1024 MB         | 4 KB                   | 8 – 16 MB                   | 1 KB                  |
| 1025 MB – 2 GB           | 4 KB                   | 8 – 16 MB                   | 2 KB                  |
| 2 GB – 4 GB              | 4 KB                   | 16 – 32 MB                  | 4 KB                  |
| 4 GB – 8 GB              | 4 KB                   | 32 – 64 MB                  | 4 KB                  |
| 8 GB – 16 GB             | 8 KB                   | 32 – 64 MB                  | 4 KB                  |
| 16 GB – 32 GB            | 16 KB                  | 32 – 64 MB                  | 4 KB                  |
| 32 GB – 2 TB             | – (nieobsługiwane*)    | –                           | 4 KB                  |

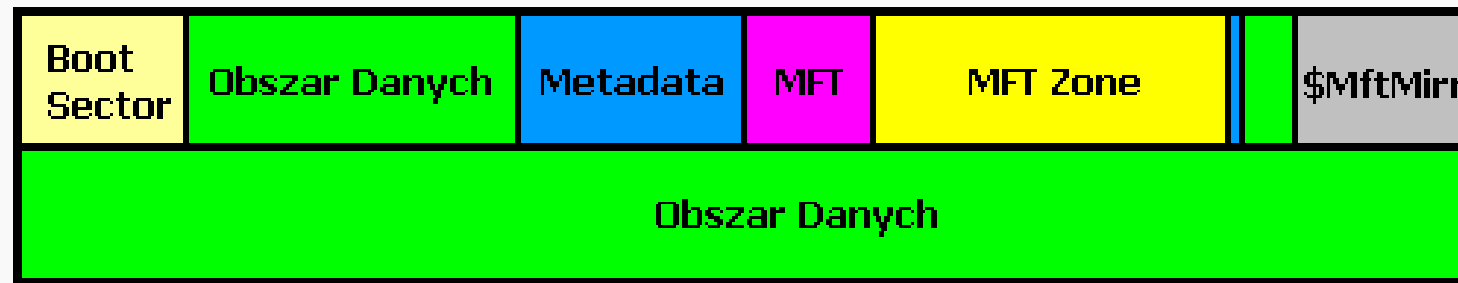
# Struktura partycji NTFS 5.1 w Windows 11 (od XP)

Na początku partycji jest **Boot Sector**, następnie obszar danych.

Po około 1/3 powierzchni partycji zaczyna się **Metadata** i **MFT**, po którym następuje jego rozszerzenie w postaci **MFT Zone**.

Dalsza część schematu przedstawia końcówkę Metadaty, fragment na dane, niewielki folder **\$MftMirr** i główny obszar danych, który zaczyna się mniej więcej od połowy partycji.

Jedyną różnicą między partycją dopiero co sformatowaną, a przechowującą dane jest brak katalogów i MFT, który dopiero powstanie na początku MFT Zone.



Dodatkowe informacje:

NTFS w Windows 11 korzysta z tych samych mechanizmów co w poprzednich wersjach: dziennikowanie, metadane, ACL, kompresja, szyfrowanie, itd.

Różnice między świeżo sformatowaną partycją a używaną nadal polegają na tym, że katalogi i rekordy MFT są tworzone dopiero przy zapisie danych.

# MFT

## Rozmiar rekordu MFT:

Każdy rekord w MFT ma **rozmiar 1 KB**. To jest standardowa wartość od wersji NTFS 5.1 (czyli od Windows XP) i obowiązuje również w Windows 11.

## Zawartość rekordu MFT:

Rekordy MFT przechowują **metadane plików i katalogów**, takie jak:

- nazwa pliku,
- lokalizacja na dysku,
- rozmiar,
- znaczniki czasowe,
- uprawnienia,
- atrybuty bezpieczeństwa.

## Małe pliki w MFT:

Jeśli plik jest **wystarczająco mały** (zwykle poniżej 700–800 bajtów, zależnie od długości nazwy i liczby atrybutów), jego zawartość może być **bezpośrednio zapisana w rekordzie MFT**. To przyspiesza dostęp do takich plików i oszczędza miejsce na dysku, zwłaszcza przy dużym rozmiarze klastra.

## Rozszerzenie rekordu MFT:

Jeśli jeden rekord MFT nie wystarcza do zapisania wszystkich atrybutów pliku (np. wielu strumieni danych, długich nazw, alternatywnych danych), NTFS tworzy **kolejne rekordy**, które są ze sobą powiązane.

|                           |                                                                            |
|---------------------------|----------------------------------------------------------------------------|
| Rekord pliku              | 000012                                                                     |
| Nazwa pliku dokument.txt  |                                                                            |
| Rozmiar danych 512 bajtów |                                                                            |
| Zmaczniki czasowe         | Utworzono:<br>2025-10-25 14:32:10<br>Zmodyfikowano:<br>2025-10-25 14:35:02 |
| Atrybuty                  | Podstawowe informacje<br>Nazwa pliku<br>Deskryptor zabezpieczeń            |
| Dane                      | Dane rezydentne (512 bajtów)                                               |

# MFT - Przykład rekordu MFT (1 KB)

| Atrybut               | Wartość / Opis                                                           |
|-----------------------|--------------------------------------------------------------------------|
| Typ rekordu           | Plik (File Record)                                                       |
| Numer rekordu         | 000012                                                                   |
| Nazwa pliku           | dokument.txt                                                             |
| Rozmiar pliku         | 512 bajtów                                                               |
| Data utworzenia       | 2025-10-25 14:32:10                                                      |
| Data modyfikacji      | 2025-10-25 14:35:02                                                      |
| Uprawnienia           | Użytkownik: IRENEUSZ, Grupa: Uczniowie, Dostęp: Odczyt/Zapis             |
| Atrybuty pliku        | Standard Information, File Name, Data, Security Descriptor               |
| Lokalizacja danych    | Dane zapisane bezpośrednio w rekordzie MFT (plik mały, < 1 KB)           |
| Zarezerwowane miejsce | Pozostałe bajty w rekordzie wypełnione zerami lub metadanymi systemowymi |

## Wyjaśnienie:

- Standard Information:** zawiera znaczniki czasowe, flagi systemowe.
- File Name:** nazwa pliku i jego identyfikator katalogu nadrzędnego.
- Data:** zawartość pliku (jeśli mała) lub wskaźnik do lokalizacji na dysku.
- Security Descriptor:** informacje o właścicielu i uprawnieniach.

Jeśli plik byłby większy niż 1 KB, jego dane byłyby zapisane poza rekordem MFT, a rekord zawierałby wskaźniki do lokalizacji danych na dysku.

# Pliki systemowe w MFT

| Nazwa pliku systemowego    | Nazwa pliku | Rekord MFT | Opis                                                                                                     |
|----------------------------|-------------|------------|----------------------------------------------------------------------------------------------------------|
| Master File Table (MFT)    | \$MFT       | 0          | Zawiera bazowy rekord dla każdego pliku w woluminie. Przy dużych plikach alokowane są rekordy dodatkowe. |
| Master File Table Mirror   | \$MFTMirr   | 1          | Kopia pierwszych 4 rekordów MFT. Zapewnia dostęp do MFT w razie awarii sektora.                          |
| Log File                   | \$LogFile   | 2          | Zapis transakcji na plikach. Umożliwia odtwarzanie danych.                                               |
| Volume Descriptor          | \$Volume    | 3          | Informacje o partycji: nazwa, wersja NTFS, czas utworzenia.                                              |
| Attribute Definition Table | \$AttrDef   | 4          | Opis typów atrybutów plików na partycji NTFS.                                                            |
| Root Directory / Folder    | .           | 5          | Folder główny woluminu.                                                                                  |
| Cluster Allocation Bitmap  | \$Bitmap    | 6          | Mapa zajętości klastrów na woluminie.                                                                    |
| Volume Boot Code           | \$Boot      | 7          | Kopia struktury Boot Sektor.                                                                             |
| Bad Cluster File           | \$BadClus   | 8          | Lista uszkodzonych klastrów na partycji.                                                                 |
| Quota Table                | \$Quota     | 9          | Deskryptory zabezpieczeń dla plików w woluminie.                                                         |
| Upper Case Table           | \$UpCase    | 10         | Informacje o konwersji nazw plików do formatu UNICODE (16-bitowego).                                     |

# Pliki systemowe w MFT

Podczas formatowania woluminu NTFS tworzonych jest zwykle 12 plików systemowych (nazywanych również plikami metadanych).

\$mft – główna tablica plików (MFT);

\$mftmirr – główna tablica plików 2 (MFT2); wykorzystywany do celów naprawczych \$badclust – plik uszkodzonych klastrów; zawiera wszystkie uszkodzone klastry woluminu

\$bitmap – mapa bitowa alokacji klastrów; zawiera mapę bitową całego woluminu, w której zaznaczony jest każdy zajęty klaster

\$boot – plik startowy; jeżeli wolumin jest woluminem startowym, zawiera program ładujący

\$attrdef – tablica definicji atrybutów; zawiera definicje wszystkich systemowych i określonych przez użytkownika atrybutów woluminu

\$logfile – plik dziennika; zawiera dziennik transakcji wykonywanych na plikach, wykorzystywany jest do celu odzyskiwania danych

\$quota – tablica przydziałów; tablica wykorzystywana do wykazywania użycia przydziałów dyskowych

\$upcase – tablica konwersji liter; tablica wykorzystywana do zamiany małych i dużych liter na odpowiadające im duże litery Unicode

\$volume – wolumin; zawiera informacje o woluminie, takie jak jego nazwa oraz wersja

\$extend – plik rozszerzeń NTFS; przechowuje dodatkowe rozszerzenia, takie jak przydziały, identyfikatory obiektów oraz dane punktu analizy ponownej

(bez nazwy) – indeks nazwy katalogu głównego; folder główny

# MFT Zone

Windows 11 chcąc zabezpieczyć MFT przed fragmentacją rezerwuje 12,5% przestrzeni partycji.

MFT Zone to tylko potencjalnie MFT, może zostać wykorzystany w kilku procentach, może też zostać wykorzystany w całości.

MFT Zone zajmuje dużo miejsca na partycji, ale został tak zaprojektowany, aby nie przeszkadzać w normalnym jej użytkowaniu.

Jeżeli skończy się wolna przestrzeń na nowe pliki, Windows automatycznie zmniejszy sektor MFT Zone o połowę.



# Metadata

- **Podział MFT:**

Microsoft w NTFS 5.1 (od Windows XP) rozdzielił strukturę MFT na:

- część zawierającą **rekordy plików**,
- oraz część zawierającą **pozostałe informacje alokacyjne**, np. bitmapę zajętości klastrów.

## Główne składniki Metadaty:

- **\$LogFile** – dziennik transakcji NTFS, używany do przywracania spójności danych po awarii.
- **\$Bitmap** – mapa zajętości klastrów, gdzie każdy klaster jest reprezentowany przez 1 bit (0 – wolny, 1 – zajęty).  
Dzięki temu NTFS bardzo szybko lokalizuje wolne miejsce.

## Fragmentacja:

- **MFT może ulec fragmentacji**, szczególnie na intensywnie używanych partycjach.
- **Metadata (np. \$Bitmap, \$LogFile)** są zwykle przechowywane w sposób ciągły i rzadko ulegają fragmentacji.

## Różnica względem FAT:

- W systemach FAT (np. FAT32) informacje o zajętości klastrów są przechowywane w jednej tablicy (File Allocation Table), która nie jest oddzielona od danych plików w tak modułarny sposób jak w NTFS.



# \$MftMirr

- \$MftMirr to specjalny plik systemowy NTFS, który zawiera kopię pierwszych 4 rekordów MFT:

- \$MFT (główna tablica plików),
- \$MftMirr (czyli sam siebie),
- \$LogFile (dziennik transakcji),
- \$Volume (informacje o woluminie)

- choć w niektórych źródłach podaje się, że może zawierać do 16 rekordów, zależnie od implementacji

i wersji systemu. [\[artefacts.help\]](http://artefacts.help)

- Lokalizacja:

\$MftMirr jest zazwyczaj umieszczony w logicznym środku partycji, co zwiększa szanse na jego dostępność w przypadku uszkodzenia początku lub końca dysku.

[\[serverfault.com\]](http://serverfault.com)

- Funkcja zabezpieczająca:

W połączeniu z mechanizmem journalingu (czyli zapisu zmian w \$LogFile przed ich fizycznym wykonaniem), \$MftMirr umożliwia odzyskanie MFT w przypadku jego uszkodzenia.

Narzędzia takie jak TestDisk potrafią przywrócić MFT właśnie na podstawie \$MftMirr.

[\[cgsecurity.org\]](http://cgsecurity.org)



# Katalogi (pliki katalogów)

- Struktura katalogów jako drzewa binarne:**

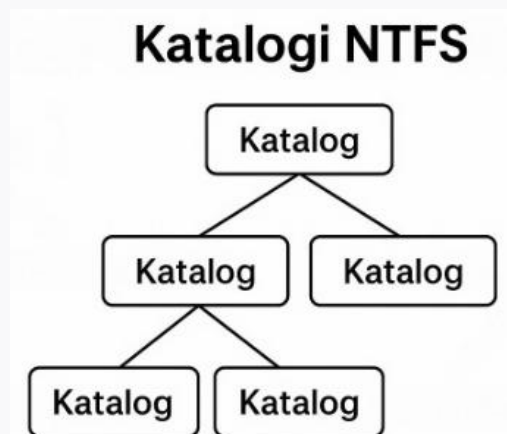
NTFS tworzy katalogi jako **specjalne pliki indeksowe**, które są organizowane w formie **drzew binarnych** (dokładniej: drzew B+), co umożliwia szybkie wyszukiwanie i dostęp do plików. Każdy wpis w katalogu zawiera odnośnik do odpowiedniego rekordu w MFT.

- Rozbudowana struktura względem FAT32:**

W porównaniu do FAT32, katalogi w NTFS są znacznie bardziej rozbudowane - zawierają metadane, uprawnienia, strumień danych, co czyni je **bardziej podatnymi na fragmentację**, zwłaszcza przy dużej liczbie plików i częstych operacjach. [\[ishell.pl\]](http://ishell.pl)

- Lokalizacja katalogów:**

Katalogi są domyślnie tworzone **po drugim fragmencie metadanych**, ale w miarę zapewniania partycji mogą być **rozmieszczone na całej jej powierzchni**, co jest zgodne z mechanizmem dynamicznego alokowania przestrzeni przez NTFS. [\[students.m...muw.edu.pl\]](http://students.m...muw.edu.pl)



# Katalog w systemie NTFS

- jest plikiem z odpowiednio posortowanym zbiorem tablic, w którym są zawarte informacje, które pliki z rekordu MFT należą do danego katalogu.
- ma postać drzewa. Pliki w katalogu są rozmieszczone w ten sposób, że do każdego wpisu w \$INDEX\_ROOT może być dołączony podwęzeł zawierający pliki znajdujące się po posortowaniu pomiędzy nim a poprzednim wpisem. Dzięki takiej strukturze przeszukiwanie katalogów jest szybkie.

| Rekord pliku katalogu                  |
|----------------------------------------|
| \$STANDARD_INFORMATION                 |
| \$FILE_NAME                            |
| \$ATTRIBUTE_LIST                       |
| \$SECURITY_DESCRIPTOR                  |
| \$INDEX_ROOT (o nazwie \$I30)          |
| \$INDEX_ALLOCATION<br>(o nazwie \$I30) |
| \$BITMAP                               |

Każdy katalog posiada trzy specyficzne atrybuty:

- \$INDEX\_ROOT - zawiera kilka wpisów do katalogu, tworzących wierzchołek drzewa.
- \$INDEX\_ALLOCATION - podaje, gdzie znajdują się pozostałe wpisy do katalogu.
- \$BITMAP - podaje, które z wpisów do katalogu są wypełnione.

# Obszar danych

- tu zapisywane są pliki użytkownika i systemu.

# Atrybuty plików

- **Atrybuty plików i katalogów** w NTFS są przechowywane w **rekordach MFT**.

Każdy plik lub katalog jest opisany zestawem atrybutów, takich jak:

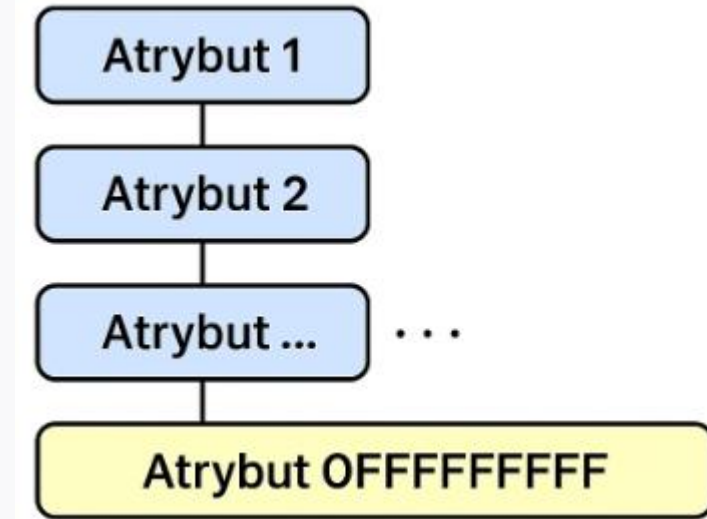
- nazwa pliku,
- data utworzenia, modyfikacji, dostępu,
- typ pliku,
- prawa dostępu,
- rozmieszczenie danych na dysku. [\[students.m...muw.edu.pl\]](https://students.muw.edu.pl)

- **Struktura atrybutów:**

- Atrybuty są zapisane **kolejno** w rekordzie MFT.
- Każdy atrybut ma nagłówek zawierający jego **typ**, **rozmiar** i **offset**.
- **Adres kolejnego atrybutu** jest obliczany jako suma offsetu bieżącego atrybutu i jego długości.

- **Zakończenie listy atrybutów:**

- Ostatni atrybut w rekordzie MFT ma **typ 0xFFFFFFFF**, co oznacza **koniec listy atrybutów**. System plików NTFS interpretuje ten wpis jako sygnał zakończenia odczytu informacji o pliku. [\[students.m...muw.edu.pl\]](https://students.muw.edu.pl)



# Atrybuty plików

## •Standard Information

Przechowuje podstawowe informacje o pliku (atrybut zawsze rezydentny).

## •Attribute List

Używany, jeśli sumaryczna długość wszystkich atrybutów jest większa od pojemności pojedynczego rekordu. Pozwala na umieszczenie części atrybutów w innych rekordach MFT.

## •File Name

Zawiera nazwę pliku (UNICODE), daty utworzenia, modyfikacji, ostatniego odczytu, atrybuty pliku oraz jego długość. Może zawierać alternatywne nazwy (np. DOS/Windows).

## •Security Descriptor

Określa zasady zabezpieczenia pliku.

## •Data

Zawiera właściwe dane pliku. Może być rezydentny lub nierezydentny. W NTFS może występować wiele strumieni danych.

## •Object ID

Unikalny identyfikator pliku. W starszych wersjach NTFS określał wersję partycji.

## •Logged Tool Stream

Strumień danych rejestrowany w \$LogFile, używany przez EFS (od NTFS 5.0).

## •Reparse Point

Opisuje zależności i właściwości rekordu zależne od flag. Może zawierać metadane adresu docelowego.

## •Index Root

Główny atrybut opisujący listę plików widoczną w katalogu. Zawsze rezydentny.

## •Index Allocation

Używany w strukturze B-drzewa dla dużych katalogów i indeksów.

## •Bitmap

Przechowuje informacje o zajętości klastrów w indeksie.

## •Volume Name

Zawiera nazwę woluminu.

## •Volume Information

Przechowuje dane o wersji systemu plików NTFS i jego właściwościach.

## •EA Information

Zawiera informacje o rozszerzonych atrybutach (Extended Attributes).

## •EA

Przechowuje rozszerzone atrybuty pliku.

## •Property Set

Używany przez aplikacje do przechowywania zestawów właściwości (np. metadanych dokumentów).

# Plik w NTFS

- **Każdy plik lub katalog w NTFS** jest reprezentowany przez **jeden rekord w pliku MFT** (Master File Table)
- Rekord MFT zawiera:
  - **nazwę pliku** (w formacie Unicode oraz skróconym 8.3),
  - **atrybuty pliku** (np. ukryty, systemowy, archiwalny),
  - **prawa dostępu** (ACL – Access Control List),
  - **informacje o położeniu danych na dysku** (lista klastrów).
- W przypadku **małych plików** (zwykle do ok. 700–800 bajtów), ich zawartość może być **bezpośrednio przechowywana w rekordzie MFT**, co przyspiesza dostęp i oszczędza miejsce. [\[ntfs.com\]](https://ntfs.com)

| Rekord pliku                 |
|------------------------------|
| \$STANDARD_INFORMATION       |
| \$FILE_NAME                  |
| \$ATTRIBUTE_LIST             |
| \$SECURITY_DESCRIPTOR        |
| \$DATA (bez nazwy)           |
| \$DATA (o nazwie strumienia) |

# NTFS / Mały plik

- **Mały plik w NTFS** może być zapisany **bezpośrednio w rekordzie MFT**, jeśli jego rozmiar (łącznie z metadanymi) nie przekracza około **700–800 bajtów**. [\[students.m...muw.edu.pl\]](#)
- Taki plik zawiera **minimalny zestaw atrybutów**, najczęściej:
  - **SA – Standard Information**: znaczniki czasowe, flagi systemowe (np. ukryty, archiwalny),
  - **FN – File Name**: nazwa pliku (Unicode i 8.3), wskaźnik do katalogu nadrzędnego,
  - **DATA – \$Data**: zawartość pliku (rezydentna),
  - **SD – Security Descriptor**: prawa dostępu (ACL). [\[students.m...muw.edu.pl\]](#)
- Atrybuty są zapisane **kolejno** w rekordzie MFT, a ich lokalizacja jest obliczana na podstawie offsetu i długości każdego atrybutu.



# NTFS / Przeciętny plik / Duży plik

## NTFS – Przeciętny plik

Plik o typowym rozmiarze, którego dane mieszczą się w jednym rekordzie MFT.

### Rekord MFT zawiera:

- **Nagłówek** – identyfikator rekordu, typ obiektu, flagi systemowe.
- **SA (Standard Information)** – daty utworzenia, modyfikacji, dostępności, flagi (np. ukryty, tylko do odczytu).
- **FN (File Name)** – nazwa pliku w formacie Unicode i 8.3, identyfikator katalogu nadrzędnego.
- **DATA (\$Data)** – zawartość pliku (jeśli mała, może być zapisana bezpośrednio w rekordzie).
- **SD (Security Descriptor)** – prawa dostępu (ACL), właściciel, grupa.

### 📁 Data Run

- Wskazuje lokalizację danych pliku na dysku (jeśli są zbyt duże, by zmieścić się w rekordzie).

### 📁 NTFS – Duży plik

Plik o większym rozmiarze, którego dane nie mieszczą się w jednym rekordzie MFT.

### 📁 Główny rekord MFT zawiera:

- **Nagłówek**
- **SA (Standard Information)**
- **EA (Extended Attributes)** – dodatkowe atrybuty, np. rozszerzenia aplikacji.
- **FN (File Name)**
- **SD (Security Descriptor)**

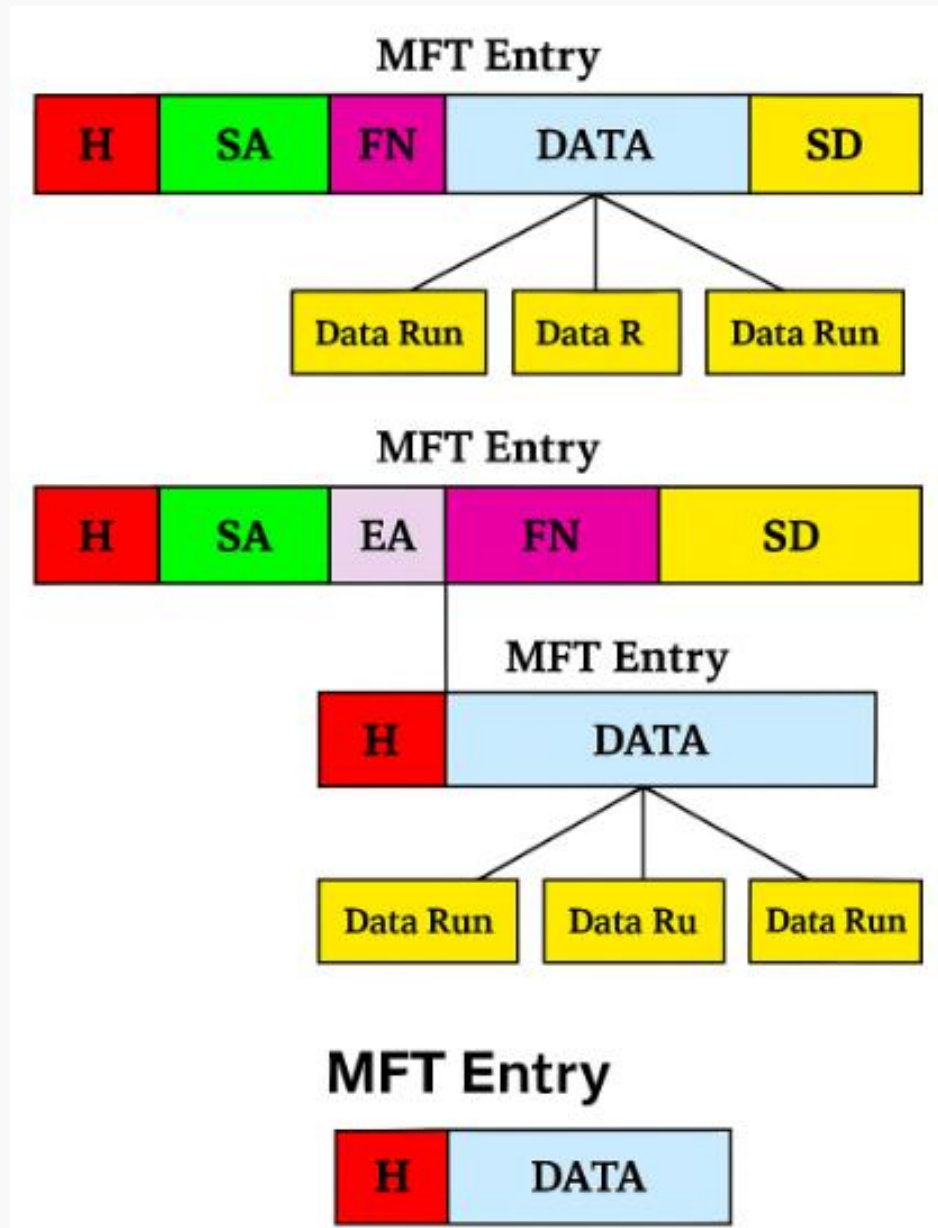
### 📁 Dodatkowe rekordy MFT:

- Zawierają kolejne fragmenty danych:

- **Nagłówek**
- **DATA (\$Data)** – wskazania do lokalizacji danych na dysku.

### 📁 Data Run (wielokrotne)

- Dane pliku są rozproszone w różnych lokalizacjach na dysku, wskazywane przez ciągi Data Run.



# NTFS / Bardzo duży plik

## Elementy diagramu NTFS / Bardzo duży plik

### ■ H - Nagłówek (Nagłówek rekordu MFT)

- Zawiera identyfikator rekordu, typ obiektu (plik/katalog), numer sekwencyjny, flagi systemowe.

- To początek każdego rekordu MFT.

### ■ SA – Standard Information

- Przechowuje metadane pliku: daty utworzenia, modyfikacji, ostatniego dostępu, flagi (np. tylko do odczytu, ukryty).

### ● EA – Extended Attributes

- Dodatkowe atrybuty pliku, np. informacje rozszerzone przez aplikacje lub system (rzadziej używane).

### ■ FN – File Name

- Nazwa pliku w formacie Unicode oraz skrócona wersja 8.3.

- Zawiera też identyfikator katalogu nadrzędnego.

### ■ SD – Security Descriptor

- Informacje o właścicielu pliku, grupie, listach kontroli dostępu (ACL).

- Określa, kto i w jaki sposób może korzystać z pliku.

### ■ DATA – \$Data

- Atrybut zawierający dane pliku.

- W przypadku dużych plików dane są przechowywane poza rekordem MFT, a rekord zawiera tylko wskaźniki (Data Runs).

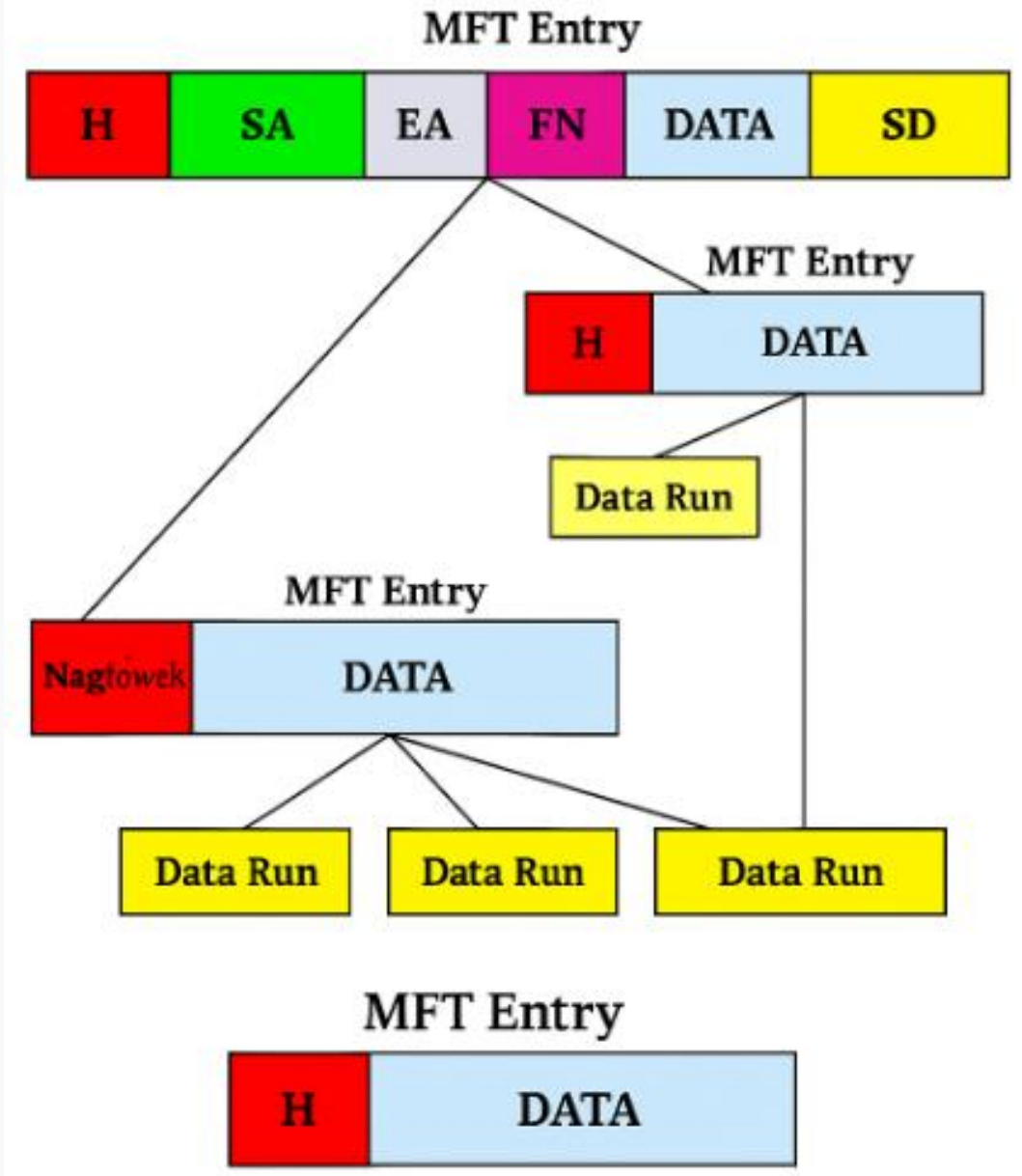
### ■ Data Run

- Fragmenty danych pliku zapisane na dysku.

- NTFS przechowuje informacje o lokalizacji danych w formie ciągów (runów), które wskazują konkretne klastry.

### ■ Dodatkowe rekordy MFT

- Gdy plik jest zbyt duży, aby zmieścić się w jednym rekordzie MFT, NTFS tworzy kolejne rekordy, które zawierają dalsze atrybuty (np. kolejne fragmenty \$Data).



# FAT32 różni się od NTFS

## Różnice między FAT32 a NTFS – struktura partycji

### Lokalizacja tablicy systemu plików

- FAT32**: Tablica alokacji plików (FAT) znajduje się **na początku partycji** (czasem również na końcu). Aby uzyskać dostęp do informacji o plikach, głowica dysku musi przemieścić się na sam początek – co może powodować opóźnienia.
- NTFS**: Główna tablica plików (**MFT – Master File Table**) znajduje się **w logicznym środku partycji**, co skraca czas dostępu do danych – głowica ma mniejszą drogę do pokonania.

### Rozmieszczenie danych na partycji

#### FAT32

- Obszar danych zajmuje większość powierzchni partycji.
- Metadane są ograniczone – system FAT32 nie przechowuje rozbudowanych informacji o plikach (np. uprawnień, właścicieli).

#### NTFS

- Partycja zawiera:

- **Metadata** – pliki systemowe NTFS (np. \$LogFile, \$Bitmap, \$Volume),
- **MFT** – główna tablica plików,
- **MFT Zone** – zarezerwowany obszar na rozszerzenie MFT,
- **Obszar danych** – pliki użytkownika.

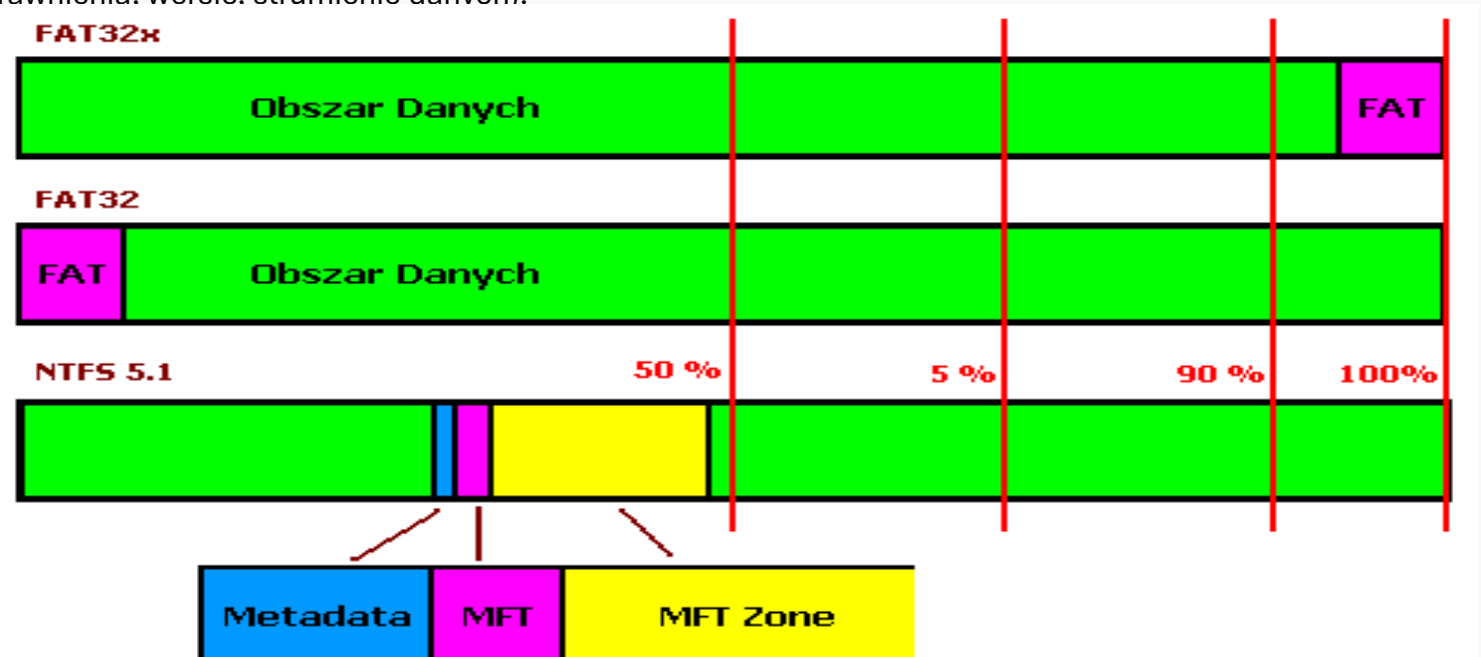
- NTFS przechowuje znacznie więcej informacji o plikach (atrybuty, uprawnienia, wersje, strumienie danych).

### Efektywność i niezawodność

- NTFS** oferuje:

- journaling (dziennik transakcji),
- lepsze zarządzanie przestrzenią dyskową,
- większe bezpieczeństwo danych,
- obsługę dużych plików i dysków.

- FAT32** jest prostszy, ale mniej wydajny i mniej odporny na błędy.



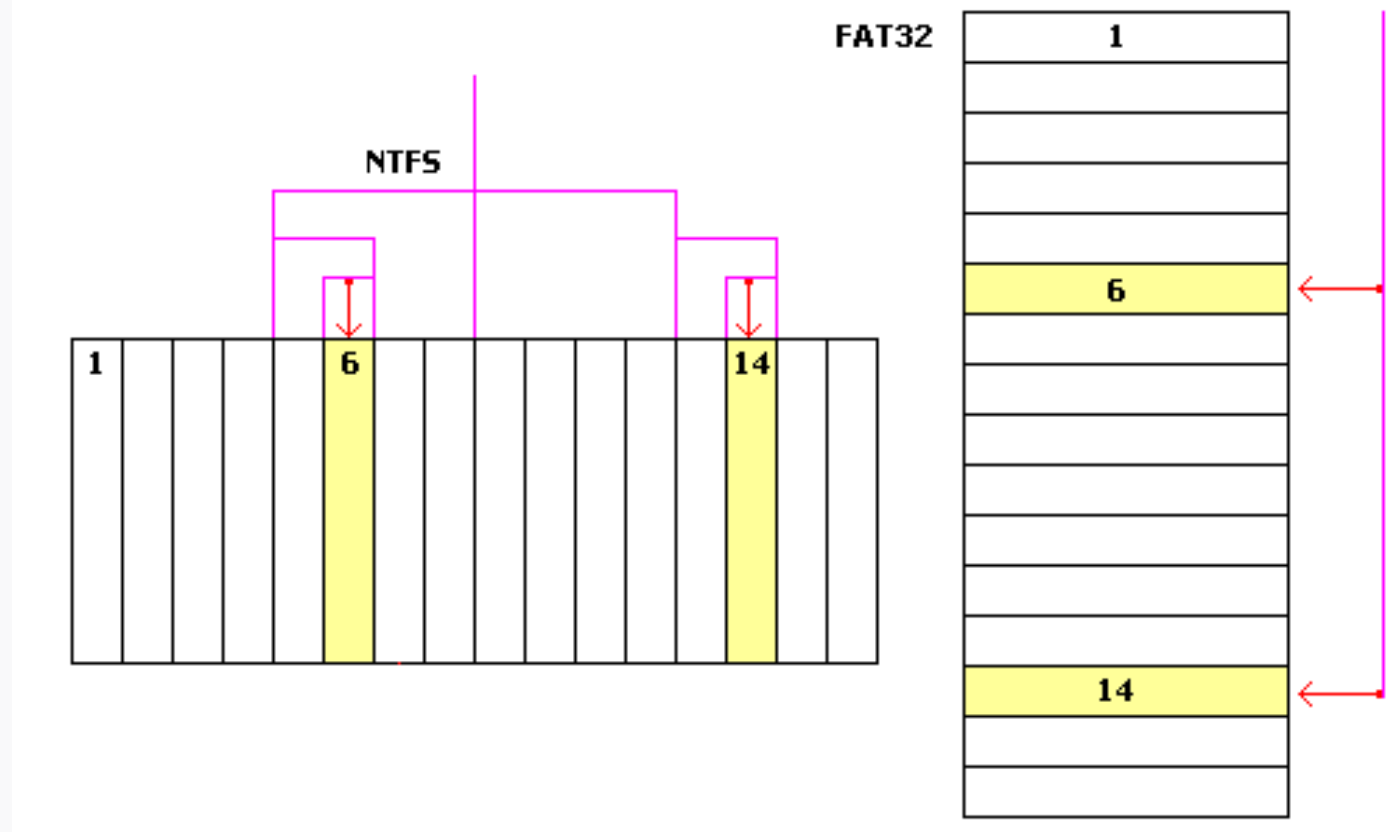
# Porównanie sposobu wyszukiwania plików w katalogach FAT32 i NTFS

- **FAT32 – katalogi linearne**

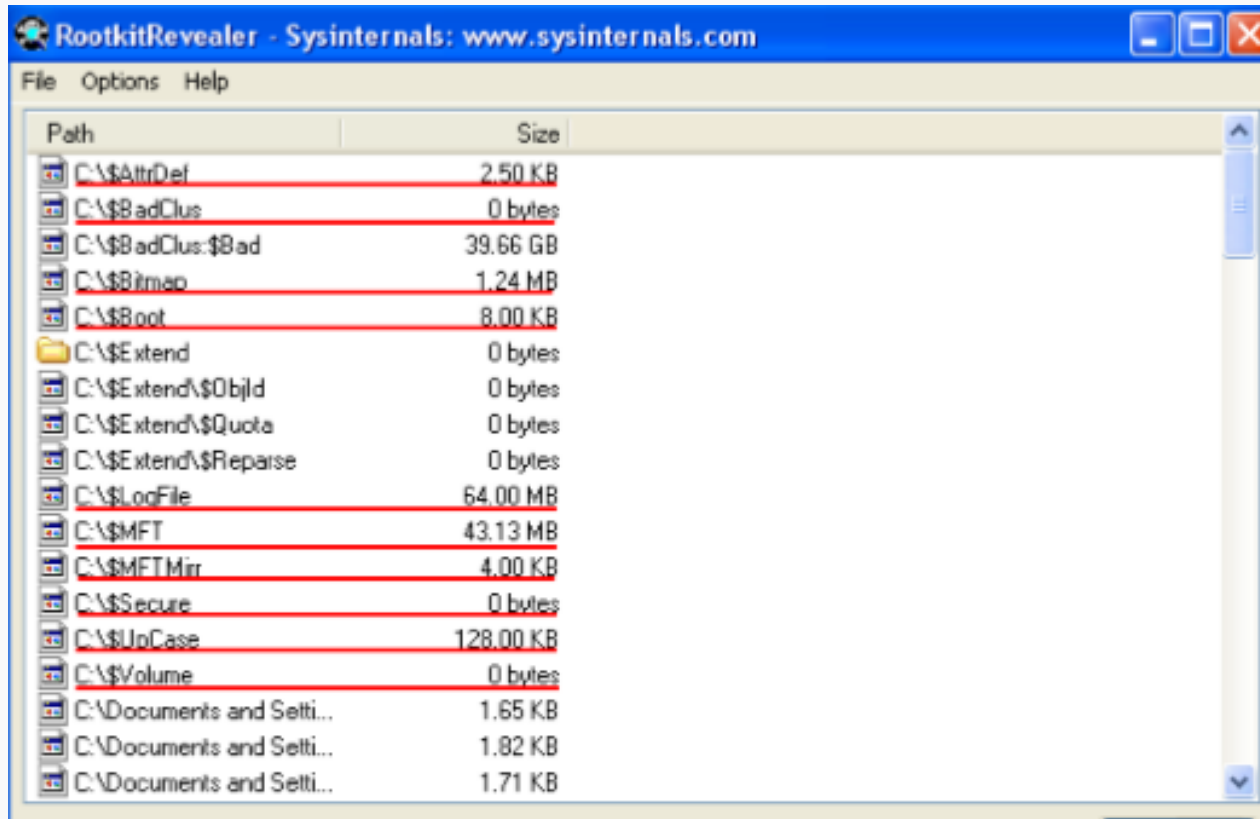
- Przeszukiwanie odbywa się **sekwencyjnie**.
- System przegląda **rekordy plików/katalogów jeden po drugim**, zaczynając od początku katalogu.
- Proces trwa do momentu odnalezienia właściwego rekordu.

- **NTFS – katalogi o strukturze drzewa binarnego**

- Katalogi są zorganizowane w **strukturę drzewa**.
- System dzieli zawartość katalogu na **dwie części**.
- Następnie wybiera tę część, w której może znajdować się plik i **ponownie dzieli ją na dwie**.
- Proces powtarza się **rekurencyjnie**, aż do odnalezienia właściwego pliku.

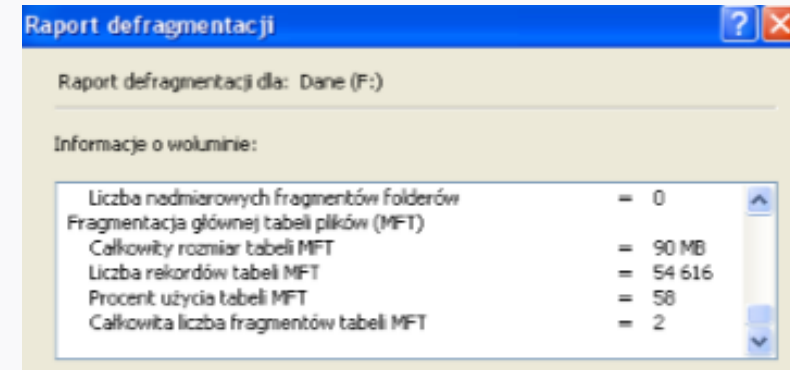


# Podgląd systemowych plików NTFS - Windows XP

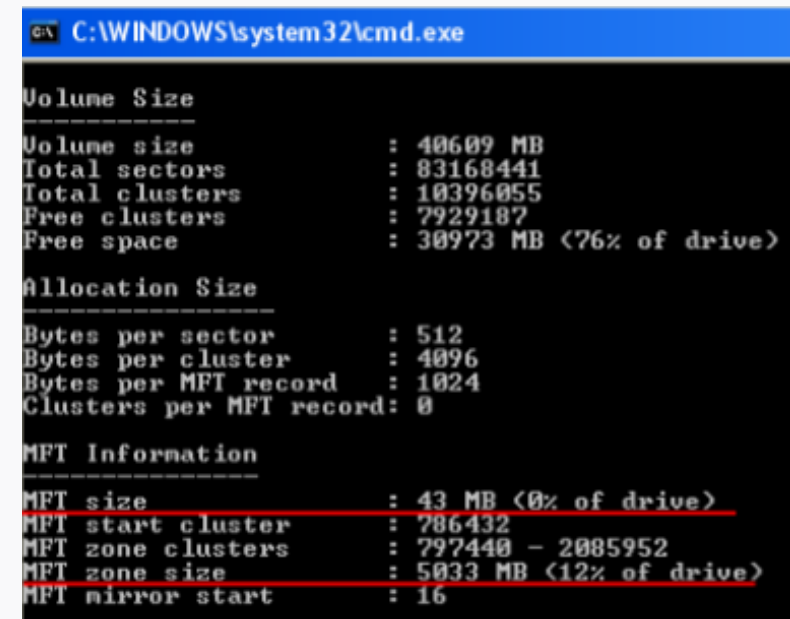


**RootkitRevealer** – narzędzie Sysinternals, które działało poprawnie tylko na Windows XP i Server 2003. Nie jest kompatybilne z Windows 10/11.

**Struktura folderów** – obecność folderów takich jak C:\Documents and Settings (widoczna w innych częściach narzędzia) typowa dla Windows XP



Format raportu defragmentacji.



Darmowe narzędzie NTFSInf

Darmowe narzędzie NTFSInf nie są kompatybilne z nowoczesnymi wersjami Windows.

# Podgląd systemowych plików NTFS - Windows 11

```
C:\Users\admin\Downloads\NTFSInfo>ntfsinfo C
```

```
NtfsInfo v1.2 - NTFS Information Dump
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com
```

## Volume Size

```
-----
Volume size          : 1299708 MB
Total sectors        : 2661804031
Total clusters       : 332725503
Free clusters        : 329232184
Free space           : 1286063 MB (98% of drive)
```

## Allocation Size

```
-----
Bytes per sector     : 512
Bytes per cluster    : 4096
Bytes per MFT record : 0
Clusters per MFT record: 0
```

## MFT Information

```
-----
MFT size              : 137 MB (0% of drive)
MFT start cluster     : 786432
MFT zone clusters     : 821696 - 848736
MFT zone size         : 105 MB (0% of drive)
MFT mirror start      : 2
```

## Meta-Data files

```
C:\Users\admin\Downloads\NTFSInfo>ntfsinfo C: > C:\Raport NTFSInfo.txt
```

```
C:\Users\admin\Downloads\NTFSInfo>fsutil fsinfo ntfsinfo C:
```

```
NTFS Volume Serial Number : 0xc5efb8a5efb6aba
NTFS Version : 3.1
LFS Version : 2.0
Total Sectors : 2 661 804 031 (1,2 TB)
Total Clusters : 332 725 503 (1,2 TB)
Free Clusters : 328 224 571 (1,2 TB)
Total Reserved Clusters : 1 771 534 (6,8 GB)
Reserved For Storage Reserve : 1 753 317 (6,7 GB)
Bytes Per Sector : 512
Bytes Per Physical Sector : 4096
Bytes Per Cluster : 4096 (4 KB)
Bytes Per FileRecord Segment : 1024
Clusters Per FileRecord Segment : 0
Mft Valid Data Length : 145,25 MB
Mft Start Lcn : 0x000000000000c0000
Mft2 Start Lcn : 0x0000000000000002
Mft Zone Start : 0x000000000000c89c0
Mft Zone End : 0x000000000000cf360
MFT Zone Size : 105,63 MB
Max Device Trim Extent Count : 4096
Max Device Trim Byte Count : 0xffffffff
Max Volume Trim Extent Count : 62
Max Volume Trim Byte Count : 0x40000000
Resource Manager Identifier : 1C98A9D4-8294-11F0-AE97-00155D01645B
```

```
C:\Users\admin\Downloads\NTFSInfo>fsutil fsinfo ntfsinfo C: > C:\Raport_NTFSInfo2.txt
```

Zaawansowany raport NTFS – struktura woluminu i metadane (Windows 11)

Struktura woluminu NTFS – analiza techniczna (Windows 11)

[NTFSInfo - Sysinternals | Microsoft Learn](#)

# Podgląd systemowych plików NTFS - Windows 11

## Narzędzie: FSUTIL (File System Utility)

- Wbudowane w systemy Windows (od Windows XP do Windows 11).
- Przeznaczone dla administratorów i zaawansowanych użytkowników.
- Wymaga **uruchomienia wiersza poleceń jako administrator**.

Co pokazuje polecenie `fsutil fsinfo ntfsinfo C:?`

## Informacje techniczne o woluminie NTFS:

- Rozmiar woluminu** (Volume size)
- Liczba sektorów i klastrów**
- Wolna przestrzeń**
- Rozmiary jednostek alokacji:**
  - bajty na sektor,
  - bajty na klaster,
  - bajty na rekord MFT,
  - klastry na rekord MFT.

- Informacje o MFT (Master File Table):**
  - rozmiar MFT,
  - lokalizacja MFT i MFT mirror,
  - zakres strefy MFT-Zone.

- Parametry TRIM** (dla dysków SSD):

- maksymalne rozmiary operacji TRIM dla urządzenia i woluminu.

- Resource Manager Identifier** – identyfikator menedżera zasobów NTFS (dla transakcyjnego systemu plików).

## Zastosowanie edukacyjne i administracyjne

- Analiza wydajności i struktury NTFS.
- Diagnostyka problemów z dyskiem.
- Porównanie konfiguracji różnych woluminów.
- Wykorzystanie w ćwiczeniach z zakresu systemów plików i bezpieczeństwa.

## Dodatkowe narzędzia (opcjonalnie)

- DiskExplorer for NTFS** – zaawansowany eksplorator sektorów NTFS.
- OSForensics File System Browser** – pokazuje ukryte pliki NTFS i ich strukturę.
- NTFSTool (GitHub)** – narzędzie forensyczne do analizy MFT i metadanych.

# Podgląd systemowych plików NTFS - Windows 11

defrag C: /V

- Parametr /V generuje szczegółowy raport, np.: Liczba fragmentów MFT,
- Procent utraty układu MFT,
- Liczba nadmiarowych fragmentów folderów.

defrag C: /A /V

gdzie **/A** oznacza analizę bez defragmentacji.

```
C:\Users\admin\Downloads\NTFSInfo>defrag C: /A /V
```

```
Invoking slab analysis on (C:)...
```

```
The operation completed successfully.
```

```
Post Defragmentation Report:
```

```
Volume Information:
```

|              |            |
|--------------|------------|
| Volume size  | = 1,23 TB  |
| Cluster size | = 4 KB     |
| Used space   | = 33,88 GB |
| Free space   | = 1,20 TB  |

```
Slab Consolidation:
```

|                          |         |
|--------------------------|---------|
| Space efficiency         | = 100%  |
| Potential purgable slabs | = 15055 |

## Interpretacja raportu analizy defragmentacji (Windows 11)

### Volume Information

- Volume size:** 1,23 TB – całkowita pojemność partycji C:.
- Cluster size:** 4 KB – rozmiar najmniejszego przydzielanego bloku danych.
- Used space:** 33,88 GB – ilość zajętego miejsca na dysku.
- Free space:** 1,20 TB – dostępna przestrzeń.

### Slab Consolidation

- Space efficiency: 100%** – oznacza, że dane są bardzo dobrze uporządkowane, nie ma istotnej fragmentacji.
- Potential purgable slabs: 15055** – liczba bloków danych (slabów), które mogą zostać zwolnione lub uporządkowane, ale nie wpływają znacząco na wydajność.

## Wnioski:

- Dysk jest **dobrze zoptymalizowany** – nie ma potrzeby przeprowadzania defragmentacji.
- Wysoka liczba „purgable slabs” może wskazywać na potencjalne możliwości optymalizacji, ale nie jest to krytyczne.
- Można rozważyć defragmentację tylko w przypadku zauważalnych spadków wydajności lub dużej liczby fragmentów w plikach.

# NTFS w systemie Linux

## Odczyt partycji NTFS

- Od wersji **jądra Linux 2.2.0 (1999)** możliwy jest **odczyt partycji NTFS**.
- Wszystkie współczesne dystrybucje Linuksa obsługują odczyt NTFS.

## Zapis na partycjach NTFS

- Przez wiele lat zapis był możliwy tylko przez **sterownik NTFS-3G**, działający w przestrzeni użytkownika (FUSE).
- **NTFS-3G**:
  - Stabilna wersja 1.0 została wydana w **lutym 2007**.
  - Umożliwia zapis, ale **nie zapewnia pełnej zgodności** z NTFS – np. ograniczenia w obsłudze praw dostępu i wydajności.

## Nowe sterowniki NTFS w jądrze Linux

- **NTFS3** – sterownik od Paragon Software:
  - Włączony do jądra Linux od wersji **5.15 (listopad 2021)**.
  - Obsługuje zapis, ale spotkał się z krytyką dotyczącą **utrzymania i braków funkcjonalnych**.
- **NTFSplus** – nowy sterownik (2025):
  - Opracowany przez **Namjae Jeon** (twórca sterownika exFAT).
  - Bazuje na klasycznym sterowniku NTFS (read-only), ale został **zmodernizowany**:
    - Obsługuje zapis.
    - Wykorzystuje nowoczesne mechanizmy jądra: **iomap**, **folio**, brak zależności od `buffer_head`.
    - Planuje pełne wsparcie dla **journalingu**.
    - Przeszedł **287 testów xfstests** (dla porównania NTFS3 – 218 testów). [\[altusintel.com\]](https://altusintel.com)

## Podsumowanie

**NTFS-3G** nadal działa i jest używany, ale ma ograniczenia.  
**NTFS3** zapewnia zapis w jądrze, ale jest słabo utrzymywany.  
**NTFSplus** to obiecujący nowy sterownik, który może zastąpić **NTFS3**, oferując lepszą wydajność i zgodność z NTFS.

# ISO 9660 – standardowy system plików dla płyt CD/DVD

## • ISO 9660 – standardowy system plików dla płyt CD/DVD

- **ISO 9660** to międzynarodowy standard opracowany przez **ISO** w 1988 roku, znany również jako **ECMA-119**.
- Obsługiwany przez wiele systemów operacyjnych: **Unix/Linux, Mac OS, Windows, DOS**.
- Często nazywany również **CDFS** (Compact Disc File System) lub **ISOFS**.
- Używany zarówno na **CD**, jak i **DVD**, choć na DVD częściej stosuje się **UDF**.

## 🔒 Ograniczenia ISO 9660:

- W wersji podstawowej **maksymalny rozmiar pliku to 2 GiB – 4 GiB** (w zależności od implementacji). [\[gameslearn...ociety.org\]](http://gameslearn.society.org)
- Ograniczenia nazw plików: 8.3 (8 znaków + rozszerzenie 3 znaki), tylko wielkie litery, brak spacji.

## 🧩 Joliet – rozszerzenie ISO 9660

- Opracowane przez **Microsoft** dla Windows 95.
- Dodaje obsługę:
  - **długich nazw plików** (do 64 znaków),
  - **rozdzielania wielkości liter**,
  - **Unicode** (np. polskie znaki).
- Jest **kompatybilne wstecznie** – płyta zawiera zarówno indeks ISO 9660, jak i Joliet. [\[learn.microsoft.com\]](http://learn.microsoft.com)

# ISO 9660 – standardowy system plików dla płyt CD/DVD

## UDF – Universal Disk Format

- Następca ISO 9660, opracowany przez **OSTA** (Optical Storage Technology Association).
- Standard dla **DVD**, **Blu-ray**, **flash media**, **CD-MRW**.
- Obsługuje:
  - **pliki do 2 TB**,
  - **długie nazwy plików (do 255 znaków)**,
  - **prawa dostępu**, **Unicode**, **pakietowe zapisywanie** (drag & drop).
- Wspierany przez większość nowoczesnych systemów operacyjnych.
- Często stosowany w połączeniu z ISO 9660 dla **kompatybilności wstecznej**. [\[confluence.si.edu\]](http://confluence.si.edu)

Podsumowanie:

| System plików | Maks. rozmiar pliku | Obsługa długich nazw | Unicode | Zapis | Kompatybilność |
|---------------|---------------------|----------------------|---------|-------|----------------|
| ISO 9660      | ~2–4 GiB            | Nie                  | Nie     | Nie   | Bardzo szeroka |
| Joliet        | ~2–4 GiB            | Tak (64 znaki)       | Tak     | Nie   | Szeroka        |
| UDF           | Do 2 TB             | Tak (255 znaków)     | Tak     |       |                |

# ISO 9660 – Poziomy zgodności:

## Poziom 1 (Level 1)

- **Nazwy plików:** format **8.3** – maksymalnie 8 znaków nazwy + 3 znaki rozszerzenia.
- **Znaki:** tylko **wielkie litery A–Z**, cyfry 0–9 i znak podkreślenia (\_).
- **Struktura katalogów:** maksymalnie **8 poziomów zagłębienia**.
- **Dodatkowo:** każdy plik musi mieć **wersję** (np. `plik.txt;1`). [\[pds.nasa.gov\]](https://pds.nasa.gov)

## Poziom 2 (Level 2)

- **Nazwy plików:** mogą być **dłuższe niż 8.3**, ale nadal ograniczone do znaków z zestawu ISO 9660.
- **Brak wersji plików.**
- **Dane:** zapisywane jako **ciągłe bajty**, bez podziału na sektory.
- **Większa elastyczność** w nazwach plików i strukturze katalogów. [\[diskinternals.com\]](https://diskinternals.com)

## Poziom 3 (Level 3)

- **Zapis danych:** umożliwia **dzielenie plików na fragmenty** (np. dla dużych plików).
- **Brak ograniczeń** co do długości nazw plików i struktury danych.
- **Obsługuje interleaving** – dane i audio mogą być przeplatane dla lepszej wydajności.
- **Usunięto limit 2 GiB** dla plików (w niektórych implementacjach). [\[learn.microsoft.com\]](https://learn.microsoft.com)

| Poziom | Nazwa pliku         | Głębokość katalogów | Obsługa wersji | Ograniczenia |
|--------|---------------------|---------------------|----------------|--------------|
| 1      | 8.3 (max 11 znaków) | max 8 poziomów      | Tak (;1)       | Duże         |
| 2      | Dłuższe nazwy       | max 8 poziomów      | Nie            | Mniejsze     |
| 3      | Dowolne             | Dowolna             | Nie            |              |

# ISO 9660 – ograniczenia nazw plików i folderów

## ✓ Zestaw dozwolonych znaków (wszystkie poziomy)

- **Duże litery:** A–Z
- **Cyfry:** 0–9
- **Znak podkreślenia:** \_
- **Kropka:** . (tylko jako separator nazwy i rozszerzenia)

## ✗ Zakazane w nazwach plików/folderów

- **Spacje** – niedozwolone
- **Więcej niż jedna kropka** – niedozwolone
- **Kropka na początku lub końcu nazwy** – niedozwolona
- **Kropki w nazwach folderów** – niedozwolone
- **Znaki specjalne** (np. !, @, #, -, ~) – niedozwolone
- **Małe litery** – niedozwolone (tylko wielkie litery są akceptowane) [\[pds.nasa.gov\]](https://pds.nasa.gov), [\[stackoverflow.com\]](https://stackoverflow.com), [\[www.ipdps.org\]](https://www.ipdps.org)

Podsumowanie zgodności nazw w ISO 9660

| Element | Dozwolone znaki      | Niedozwolone elementy                       |
|---------|----------------------|---------------------------------------------|
| Pliki   | A–Z, 0–9, _, jedna . | spacje, >1 kropka, początek/koniec z kropką |
| Foldery | A–Z, 0–9, _          | kropki, spacje, znaki specjalne             |

# UDF



## UDF – Universal Disk Format

- **UDF** to system plików opracowany przez **OSTA (Optical Storage Technology Association)** jako implementacja standardu **ISO/IEC 13346** (równoważnego z ECMA-167).
- Został zaprojektowany jako **następca ISO 9660**, oferując większą elastyczność i funkcjonalność.



## Obsługiwane nośniki danych

UDF jest używany na szerokiej gamie nośników, szczególnie tych o ograniczonych możliwościach ponownego zapisu:

- **Napędy magneto-optyczne**
- **Blu-ray (BD-R, BD-RE)**
- **DVD±R, DVD±RW, DVD-ROM**
- **CD-R, CD-RW**
- **DVD-RAM** [[ibm.com](http://ibm.com)]



## Cechy UDF

- Obsługuje **długie nazwy plików** (do 255 znaków) oraz **Unicode**.
- Umożliwia **zapis i modyfikację danych** na nośnikach wielokrotnego zapisu.
- Obsługuje **prawa dostępu** (POSIX), daty utworzenia, modyfikacji, dostępu.
- Maksymalny rozmiar pliku: **do 16 EiB** (w teorii).
- Obsługuje **hierarchiczną strukturę katalogów**.



## Zgodność

- UDF jest obsługiwany przez większość systemów operacyjnych: **Windows, macOS, Linux, Unix**.
- Wersje UDF (np. 1.02, 1.50, 2.01, 2.50) różnią się funkcjonalnością i kompatybilnością z różnymi typami nośników.

# UDF

## Obsługiwane nośniki danych

UDF jest stosowany nie tylko na klasycznych nośnikach optycznych, ale także na innych typach pamięci:

| Nośnik danych           | Obsługa przez UDF | Uwagi                                                                                                                          |
|-------------------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Napędy magneto-optyczne | ✓                 | Obsługa zapisu i modyfikacji danych                                                                                            |
| Blu-ray (BD-R, BD-RE)   | ✓                 | Standardowy system plików dla Blu-ray                                                                                          |
| DVD±R, DVD±RW, DVD-ROM  | ✓                 | Oficjalny system plików dla DVD-Video                                                                                          |
| CD-R, CD-RW             | ✓                 | Obsługa zapisu (CD-RW – wielokrotny zapis)                                                                                     |
| DVD-RAM                 | ✓                 | Obsługa pełnego zapisu i modyfikacji danych                                                                                    |
| CD-MRW (Mount Rainier)  | ✓                 | UDF wspiera ten format dla szybkiego zapisu <a href="https://learn.microsoft.com">[learn.microsoft.com]</a>                    |
| Dyski Iomega REV        | ✓                 | UDF obsługuje te dyski jako nośniki wymienne <a href="https://learn.microsoft.com">[learn.microsoft.com]</a>                   |
| Pamięci Flash (USB)     | ✓                 | UDF może być używany na pendrive'ach i innych pamięciach flash <a href="https://learn.microsoft.com">[learn.microsoft.com]</a> |

## Dlaczego UDF?

- Uniwersalność** – działa na wielu systemach operacyjnych: Windows, macOS, Linux.
- Zapis i modyfikacja** – idealny dla nośników wielokrotnego zapisu.
- Obsługa dużych plików** – do 16 EiB.
- Długie nazwy plików**, Unicode, prawa dostępu.

# UDF



## UDF – Universal Disk Format

- UDF to system plików opracowany przez **OSTA** jako implementacja standardu **ISO/IEC 13346** (ECMA-167).
- Został zaprojektowany jako **następca ISO 9660**, z myślą o nośnikach umożliwiającym zapis i modyfikację danych.



## Zastosowanie UDF



### Oficjalny system plików dla DVD-Video

- UDF został przyjęty przez **DVD Consortium** jako **standardowy system plików dla DVD-Video i DVD-Audio**.



### Obsługa przez systemy operacyjne

- Microsoft Windows Vista i nowsze** – natywna obsługa UDF.
- Linux od wersji 2.6** – obsługuje UDF, w tym zapis na nośnikach wielokrotnego zapisu (np. DVD±RW, CD-RW).

[\[kernel.org\]](http://kernel.org)



### Obsługa przez programy do pakietowego zapisu

- DirectCD, InCD, WriteUDF** – wykorzystują UDF do pakietowego zapisu danych na nośnikach optycznych.



## Ograniczenia i zalety

| Cecha                       | ISO 9660      | UDF                                                                                                                     |
|-----------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------|
| Maks. rozmiar pliku         | ~2 GiB        | <b>do 16 EiB</b> <a href="http://en.wikipedia.org">[en.wikipedia.org]</a>                                               |
| Obsługa długich nazw plików | Ograniczona   | Do 255 znaków (Unicode)                                                                                                 |
| Obsługa zapisu              | Tylko w sesji | Obsługuje zapis i modyfikację danych                                                                                    |
| Obsługa przez OS            | Szeroka       | Windows, macOS, Linux                                                                                                   |
| Obsługiwane nośniki         | CD/DVD        | CD-R/RW, DVD±R/RW, DVD-RAM, Blu-ray, Flash, Iomega REV, CD-MRW <a href="http://en.wikipedia.org">[en.wikipedia.org]</a> |

# Systemy plików Linux – ext

# ext2 – klasyczny system bez journalingu

## Prosty system plików

ext2 cechuje się prostą strukturą i brakiem journalingu, co zwiększa szybkość działania.

## Obsługa dużych partycji

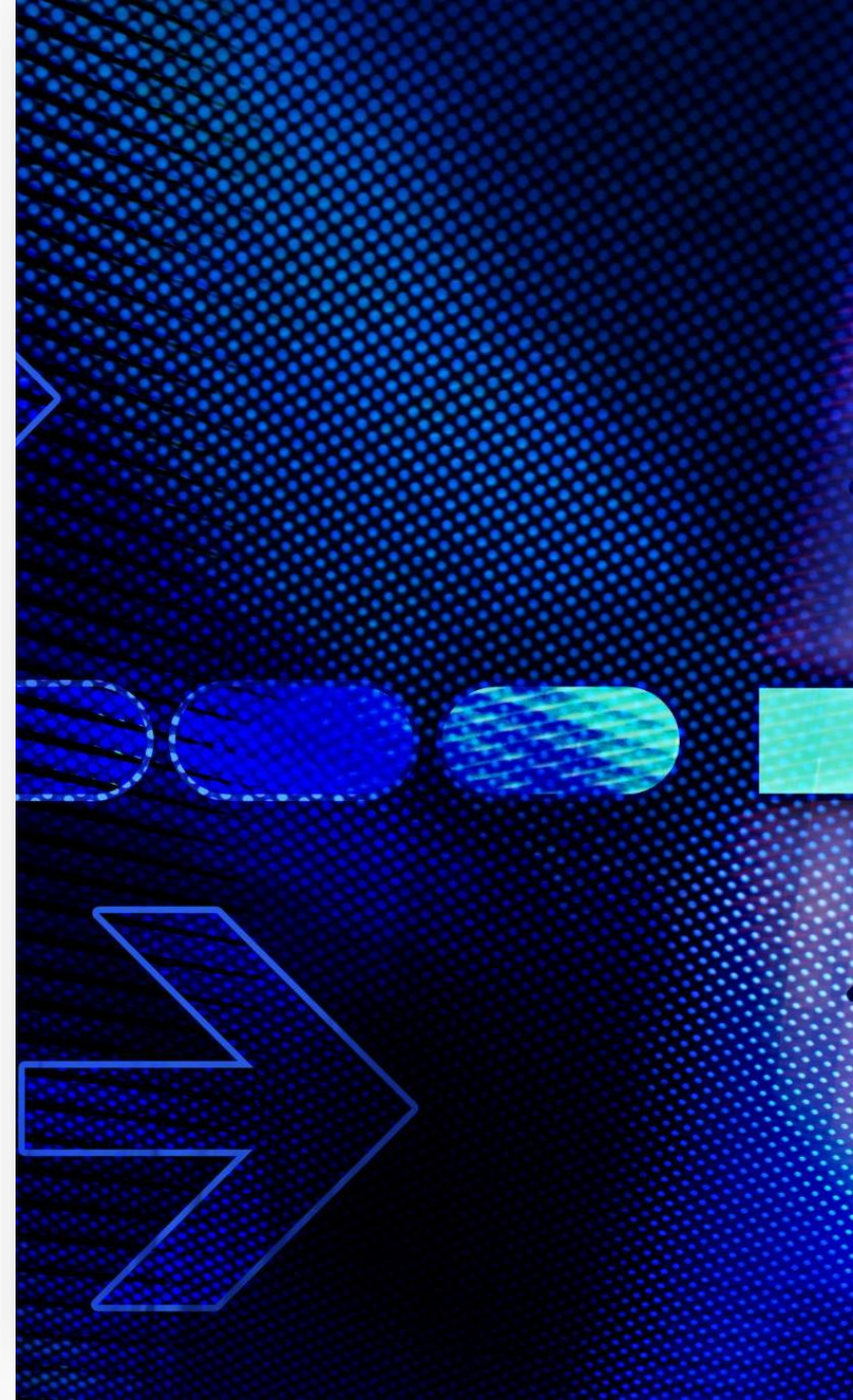
System obsługuje partycje do 32 TB i pliki o wielkości do 2 GB.

## Zarządzanie przestrzenią

Wykorzystuje bitmapy bloków i i-węzły do efektywnego zarządzania miejscem na dysku.

## Zastosowanie w systemach wbudowanych

Stosowany jest głównie w systemach wbudowanych i bootloaderach ze względu na prostotę i szybkość.



# ext2 – struktura i zastosowania

## 1. Struktura inode (i-węzłów)

- W systemie ext2 każdy plik jest reprezentowany przez **inode**, czyli strukturę danych zawierającą:
  - Rozmiar pliku
  - Właściciela (UID, GID)
  - Prawa dostępu (chmod)
  - Znaczniki czasu: utworzenia, modyfikacji, ostatniego dostępu
  - Liczbę dowiązań (link count)
  - Lokalizację bloków danych
- Inode **nie zawiera nazwy pliku** – ta jest przechowywana w katalogu, który wskazuje na odpowiedni inode.

## 2. Brak journalingu

- ext2 **nie posiada mechanizmu journalingu**, czyli zapisu zmian w dzienniku przed ich wykonaniem.
- Skutki:
  - Szybsze działanie (brak dodatkowego zapisu)
  - Większe ryzyko utraty danych po awarii systemu lub nagłym wyłączeniu zasilania
  - Konieczność użycia narzędzi typu `fsck` do naprawy systemu plików

## 3. Alokacja bloków

- Dane są przechowywane w **blokach** (np. 1 KB, 2 KB, 4 KB).
- ext2 używa **bitmap bloków i i-węzłów** do zarządzania wolną przestrzenią.
- Bitmapy pozwalają szybko określić, które bloki są wolne, a które zajęte.

# ext2 – struktura i zastosowania

## 3. Alokacja bloków

- Dane są przechowywane w **blokach** (np. 1 KB, 2 KB, 4 KB).
- ext2 używa **bitmap bloków i i-węzłów** do zarządzania wolną przestrzenią.
- Bitmapy pozwalają szybko określić, które bloki są wolne, a które zajęte.

## 4. Fragmentacja

- ext2 **nie obsługuje extents**, czyli ciągłych obszarów danych.
- Dane mogą być rozproszone po dysku, co prowadzi do **fragmentacji**.
- Fragmentacja wpływa na wydajność odczytu i zapisu, szczególnie przy dużych plikach.

## 5. Zastosowania ext2

- **Systemy wbudowane** – np. routery, urządzenia IoT, gdzie liczy się prostota i szybkość.
- **Bootloadery** – np. GRUB, które wymagają prostego systemu plików bez journalingu.
- **Nośniki flash** – gdzie nadmiar operacji zapisu (jak journaling) może skracać żywotność pamięci.
- **Środowiska edukacyjne** – do nauki podstaw działania systemów plików

# ext3 – stabilność i journaling

## Rozszerzenie ext2

ext3 to ulepszenie ext2, dodające mechanizm journalingu zwiększający niezawodność.

## Mechanizm journalingu

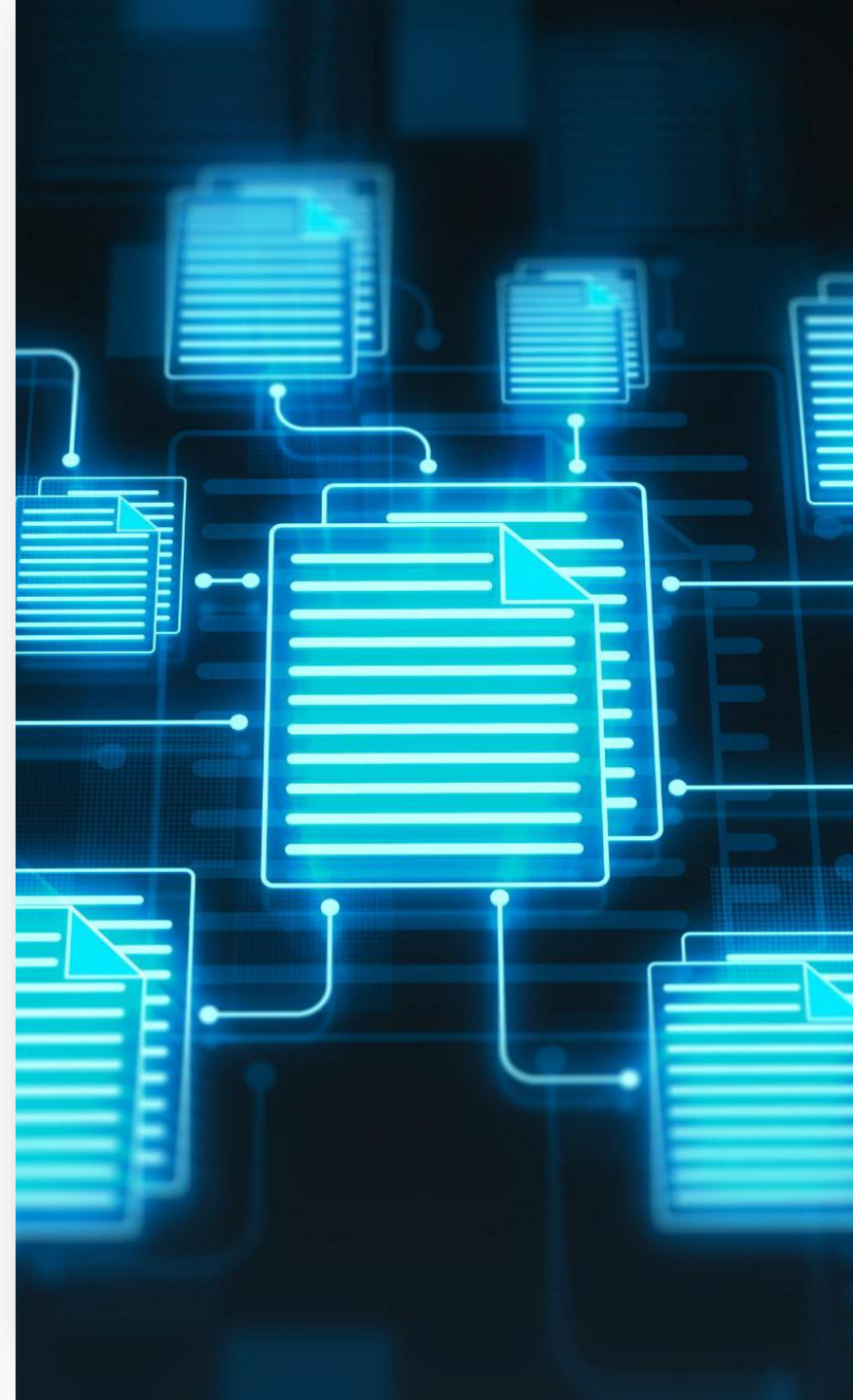
Journaling obejmuje metadane lub pełne dane, przyspieszając odzyskiwanie po awarii.

## Kompatybilność wsteczna

ext3 jest kompatybilny z ext2, co umożliwia łatwą migrację danych.

## Wysoka pojemność i stabilność

Obsługuje partycje do 32 TB i pliki do 2 GB, zapewniając stabilność systemu.



# ext3 – journaling i stabilność

## 1. Mechanizm journalingu

- ext3 wprowadza **dziennik operacji (journal)**, który rejestruje zmiany w systemie plików zanim zostaną one faktycznie zapisane.
- Domyślnie journaling obejmuje **metadane** (np. informacje o plikach, katalogach), co znacząco zwiększa **odporność na awarie**.
- W razie nagłego wyłączenia systemu, ext3 może szybko przywrócić spójność danych na podstawie dziennika, bez konieczności pełnego skanowania dysku (`fsck`).

## 2. Tryby journalingu

ext3 oferuje trzy tryby pracy journalingu, pozwalające dostosować poziom ochrony danych:

| Tryb             | Opis                                                                      | Zalety                               | Wady                            |
|------------------|---------------------------------------------------------------------------|--------------------------------------|---------------------------------|
| <b>Ordered</b>   | Domyślny tryb – najpierw zapisywane są dane, potem metadane do dziennika. | Dobra wydajność, spójność metadanych | Dane mogą zostać utracone       |
| <b>Writeback</b> | Metadane zapisywane są niezależnie od danych.                             | Najszybszy tryb                      | Najniższy poziom bezpieczeństwa |
| <b>Journal</b>   | Zarówno dane, jak i metadane trafiają do dziennika.                       | Najwyższe bezpieczeństwo             | Najniższa wydajność             |

# ext3 – journaling i stabilność

## 3. Kompatybilność z ext2

- ext3 jest **w pełni kompatybilny z ext2** – można łatwo dokonać migracji bez utraty danych.
- Wystarczy dodać dziennik (`tune2fs -j`) do istniejącej partycji ext2, by przekształcić ją w ext3.
- ext3 może być montowany jako ext2, jeśli nie jest wymagany journaling.

## 4. Fragmentacja

- ext3 **nie obsługuje extents**, czyli ciągłych obszarów danych.
- Dane mogą być rozproszone po dysku, co prowadzi do **fragmentacji**.
- Fragmentacja wpływa na wydajność, szczególnie przy dużych plikach i intensywnym zapisie.

## 5. Zastosowania ext3

- **Serwery** – dzięki journalingowi zapewnia większą niezawodność.
- **Systemy produkcyjne** – stabilność i kompatybilność z ext2.
- **Dystrybucje Linux do 2010 roku** – był domyślnym systemem plików w wielu popularnych dystrybucjach (np. Red Hat, Debian, Ubuntu).
- **Środowiska wymagające stabilności** – np. systemy bazodanowe, aplikacje krytyczne.

# ext4 – nowoczesność i wydajność

## Obsługa dużych partycji

ext4 obsługuje partycje do 1 eksabajta i pliki do 16 terabajtów, zapewniając skalowalność.

## Nowoczesne mechanizmy plików

Wprowadzenie extents i opóźnionych alokacji zwiększa wydajność systemu plików.

## Niezawodność i szybkie sprawdzanie

Sumy kontrolne metadanych i szybkie fsck minimalizują błędy i fragmentację.

## Kompatybilność z ext3

ext4 jest kompatybilny z ext3, co ułatwia migrację między systemami.



# ext4 – nowoczesność i wydajność

## 1. Extents – grupowanie bloków

- Ext4 wprowadza **extents**, czyli ciągłe zakresy bloków danych.
- Zamiast przechowywać każdy blok osobno (jak w ext3), ext4 zapisuje informacje o całych zakresach bloków.
- Zalety:
  - **Zmniejszenie fragmentacji** danych
  - **Szybszy dostęp** do plików
  - **Lepsza wydajność** przy dużych plikach

## 2. Opóźniona alokacja (Delayed Allocation)

- Dane nie są zapisywane natychmiast po wywołaniu operacji zapisu.
- System plików analizuje strukturę danych i dostępne miejsce, aby **optymalnie rozmieścić dane**.
- Zalety:
  - **Redukcja fragmentacji**
  - **Zwiększenie wydajności zapisu**
  - **Lepsze wykorzystanie przestrzeni dyskowej**

## 3. Sumy kontrolne metadanych

- Ext4 dodaje **sumy kontrolne** do metadanych (np. dziennika journalingu).
- Pozwala to na **wykrywanie uszkodzeń danych** spowodowanych błędami sprzętowymi lub awariami.
- Zwiększa **integralność systemu plików** i bezpieczeństwo danych.

# ext4 – nowoczesność i wydajność

## 4. Fsck – szybkie sprawdzanie systemu plików

- Ext4 wspiera **wielowątkowe sprawdzanie systemu plików** (`fsck`).
- Dzięki podziałowi na grupy alokacji, możliwe jest równoległe sprawdzanie różnych części dysku.
- Zalety:
  - **Znacznie szybsze uruchamianie systemu** po awarii
  - **Lepsza skalowalność** na dużych dyskach

## 5. Zastosowania ext4

- **Domyślny system plików** w większości dystrybucji Linux: Ubuntu, Debian, Fedora, Arch Linux itd.
- **Serwery i stacje robocze** – dzięki stabilności i wydajności
- **Systemy produkcyjne** – obsługa dużych plików i dysków
- **Środowiska desktopowe** – szybki dostęp do danych i niezawodność

# Nowoczesne systemy plików Linux

# Btrfs – zaawansowane funkcje i elastyczność

## **Zaawansowane funkcje Btrfs**

Btrfs obsługuje migawki, kompresję danych, sumy kontrolne i RAID dla niezawodności i wydajności.

## **Elastyczne zarządzanie przestrzenią**

System umożliwia łatwe rozszerzanie i zmniejszanie wolumenów bez przestojów.

## **Wsparcie dystrybucji Linux**

Btrfs jest wspierany przez popularne dystrybucje openSUSE, Fedora i Ubuntu 20.04 i nowsze.

## **Lepsze zarządzanie i odzyskiwanie**

Funkcje Btrfs umożliwiają szybkie odzyskiwanie danych i efektywne zarządzanie systemem plików.



# Btrfs – zaawansowane funkcje i elastyczność

## 1. Zaawansowane funkcje Btrfs

### •Migawki (snapshots)

Btrfs umożliwia tworzenie migawkowych kopii danych w czasie rzeczywistym, bez konieczności ich duplikowania. Migawki są szybkie i zajmują mało miejsca, ponieważ zapisują tylko zmiany względem oryginału.

### •Kompresja danych

Obsługuje kompresję w locie (np. algorytmy zlib, lzo, zstd), co pozwala zaoszczędzić miejsce na dysku i przyspieszyć odczyt danych.

### •Sumy kontrolne (checksums)

Każdy blok danych i metadanych jest zabezpieczony sumą kontrolną SHA256, co pozwala wykrywać i naprawiać błędy danych (np. bit rot).

### •RAID

Wbudowana obsługa RAID 0, 1, 10 (oraz eksperymentalnie RAID 5/6) bez potrzeby używania zewnętrznych narzędzi. Btrfs zarządza dyskami i ich redundancją samodzielnie.

## 2. Elastyczne zarządzanie przestrzenią

### •Dynamiczne wolumeny

Możliwość dodawania i usuwania urządzeń z systemu plików w locie, bez przestoju.

### •Podwolumeny (subvolumes)

Umożliwiają tworzenie logicznych jednostek w ramach jednego systemu plików, które można niezależnie zarządzać, montować i tworzyć migawki.

### •Automatyczne równoważenie danych (balancing)

Dane mogą być automatycznie przenoszone między urządzeniami w celu optymalizacji przestrzeni i wydajności.

# Btrfs – zaawansowane funkcje i elastyczność

## 3. Wsparcie dystrybucji Linux

- Btrfs jest wspierany przez wiele popularnych dystrybucji:
  - **openSUSE** – domyślny system plików z zaawansowaną integracją snapshotów.
  - **Fedora** – od wersji 33 domyślnie używa Btrfs.
  - **Ubuntu** – od wersji 20.04 dostępne wsparcie, choć nie jako domyślny system plików.

## 4. Lepsze zarządzanie i odzyskiwanie danych

### • Samonaprawa danych

Dzięki sumom kontrolnym i redundancji Btrfs może automatycznie wykrywać i naprawiać uszkodzone dane.

### • Szybkie odzyskiwanie

Migawki pozwalają na błyskawiczne przywrócenie systemu do wcześniejszego stanu – idealne przy aktualizacjach lub awariach.

### • Zintegrowane narzędzia administracyjne

Narzędzia takie jak `btrfs scrub`, `btrfs balance`, `btrfs check` umożliwiają kompleksowe zarządzanie i diagnostykę systemu plików.

## Podsumowanie

Btrfs to nowoczesny, zaawansowany system plików, który łączy funkcje typowe dla systemów klasy enterprise z elastycznością i wydajnością potrzebną w środowiskach desktopowych i serwerowych. Idealny dla administratorów, użytkowników wymagających snapshotów, kompresji i wysokiej niezawodności.

# XFS – wydajność dla dużych plików

## **Wysoka wydajność**

XFS jest zoptymalizowany do obsługi dużych plików i intensywnych operacji dyskowych.

## **Journaling metadanych**

System wykorzystuje journaling metadanych dla szybkiego odzyskiwania danych po awarii.

## **Struktura drzewa B+**

Struktura oparta na drzewach B+ zapewnia efektywne zarządzanie przestrzenią dyskową.

## **Zastosowania systemu**

XFS jest domyślnym systemem plików w CentOS i RHEL, idealnym dla serwerów i HPC.



# XFS – wydajność dla dużych plików

## 1. Wysoka wydajność

- **XFS** został zaprojektowany z myślą o obsłudze **dużych plików** i intensywnych operacji dyskowych.
- Doskonale sprawdza się w środowiskach, gdzie występuje **duża liczba operacji I/O**, np. serwery baz danych, systemy multimedialne, centra danych.
- Obsługuje **asynchroniczne operacje**, co pozwala na równoległe przetwarzanie danych i zwiększa przepustowość.

## 2. Journaling metadanych

- XFS stosuje **journaling metadanych**, co oznacza, że informacje o strukturze plików (np. katalogi, prawa dostępu, lokalizacja danych) są zapisywane w dzienniku przed ich faktycznym zapisaniem.
- Dzięki temu system może **szybko odzyskać spójność** po awarii, bez konieczności pełnego skanowania dysku.
- Journaling w XFS jest zoptymalizowany pod kątem wydajności i minimalizuje wpływ na operacje zapisu.

## 3. Struktura drzewa B+

- XFS wykorzystuje **drzewa B+** do zarządzania przestrzenią dyskową, co zapewnia:
  - **Szybki dostęp** do danych
  - **Efektywne indeksowanie** dużych katalogów
  - **Skalowalność** – dobrze radzi sobie z milionami plików
- Struktura B+ umożliwia dynamiczne rozszerzanie i reorganizację danych bez pogorszenia wydajności.

# XFS – wydajność dla dużych plików

## 4. Zastosowania systemu XFS

- **Domyślny system plików** w dystrybucjach takich jak **CentOS, RHEL (Red Hat Enterprise Linux)**.
- Idealny dla:
  - **Serwerów produkcyjnych**
  - **Systemów HPC (High Performance Computing)**
  - **Środowisk wymagających dużej przepustowości danych**
- Obsługuje **wolumeny o rozmiarze do 8 eksabajtów**, co czyni go odpowiednim dla zastosowań enterprise.

## Dodatkowe cechy XFS

- **Dynamiczne alokowanie przestrzeni** – brak fragmentacji dzięki inteligentnemu rozmieszczaniu danych.
- **Obsługa kwot dyskowych** – możliwość ograniczania przestrzeni dla użytkowników i grup.
- **Integracja z LVM** – łatwe zarządzanie wolumenami logicznymi.

# ReiserFS – optymalizacja dla małych plików

## Optymalizacja dla małych plików

ReiserFS jest zoptymalizowany do przechowywania dużej liczby małych plików, co zwiększa efektywność.

## Funkcje systemu

Obsługuje journaling i dynamiczne alokowanie przestrzeni dla lepszego zarządzania dyskiem.

## Obecny status systemu

ReiserFS jest rzadziej używany z powodu braku rozwoju, zastąpiony przez ext4 i Btrfs.

## Znaczenie edukacyjne

System ten jest przykładem specjalistycznego podejścia w nauce o systemach plików.



# ReiserFS – optymalizacja dla małych plików

## 1. Optymalizacja dla małych plików

- **ReiserFS** został zaprojektowany z myślą o efektywnym przechowywaniu **dużej liczby małych plików**, co czyni go wyjątkowym w porównaniu do innych systemów plików.
- Zamiast przydzielać każdemu plikowi osobny blok, ReiserFS potrafi **upakować wiele małych plików w jednym bloku**, co znacząco zmniejsza marnowanie przestrzeni.
- Dzięki temu system był szczególnie wydajny w środowiskach takich jak **serwery pocztowe**, katalogi z dużą liczbą plików konfiguracyjnych, czy systemy logowania.

## 2. Funkcje systemu

### •Journaling

ReiserFS obsługuje journaling metadanych, co zwiększa odporność na awarie i pozwala na szybkie odzyskiwanie spójności systemu plików po nieoczekiwanym wyłączeniu.

### •Dynamiczna alokacja przestrzeni

System potrafi elastycznie zarządzać przestrzenią dyskową, co w połączeniu z optymalizacją dla małych plików pozwala na **efektywne wykorzystanie dostępnego miejsca**.

### •Szybkie wyszukiwanie

ReiserFS wykorzystuje strukturę drzew (reiser tree), która umożliwia szybkie wyszukiwanie plików i katalogów, nawet w bardzo dużych strukturach.

# ReiserFS – optymalizacja dla małych plików

## 3. Obecny status systemu

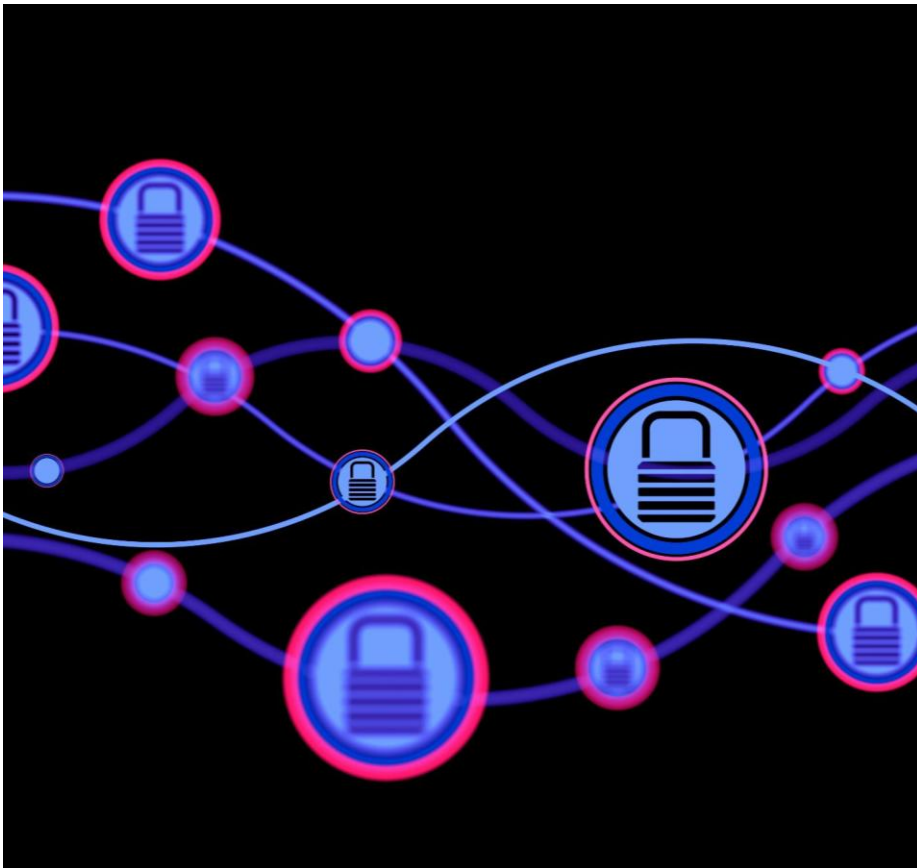
- **Brak aktywnego rozwoju** – projekt ReiserFS został wstrzymany, a jego twórca, Hans Reiser, został skazany za przestępstwo, co wpłynęło na dalsze wsparcie systemu.
- **Zastąpiony przez ext4 i Btrfs** – ze względu na brak aktualizacji i ograniczenia technologiczne, ReiserFS został wyparty przez nowocześniejsze systemy plików.
- **Obecnie rzadko używany**, choć nadal dostępny w niektórych starszych dystrybucjach Linuksa.

## 4. Znaczenie edukacyjne

- ReiserFS jest **doskonałym przykładem specjalistycznego podejścia** do projektowania systemów plików.
- Pokazuje, jak **konkretne potrzeby** (np. efektywne przechowywanie małych plików) mogą prowadzić do innowacyjnych rozwiązań.
- W edukacji technicznej może służyć jako **studium przypadku** dotyczące:
  - optymalizacji przestrzeni,
  - struktur danych w systemach plików,
  - wpływu journalingu na niezawodność,
  - ewolucji systemów plików w Linuksie.

# Porównanie systemów plików

# Zalety i ograniczenia różnych systemów plików



## Prostota systemów FAT

FAT16, FAT32 i exFAT oferują szeroką kompatybilność i prostotę, lecz brak im zaawansowanych funkcji bezpieczeństwa.

## Zaawansowane funkcje NTFS

NTFS zapewnia ochronę danych, journaling, szyfrowanie i kompresję, idealne dla systemów Windows.

## Ewolucja systemów Linux

ext2, ext3 i ext4 oferują rosnącą wydajność i niezawodność w środowisku Linux.

## Nowoczesne systemy plików

Btrfs, XFS i ReiserFS służą do specjalistycznych zastosowań, jak serwery i przechowywanie małych plików.

# Zmiana systemu plików

## Narzędzie `convert.exe` – aktualne informacje

- `convert.exe` to **wbudowane narzędzie systemu Windows**, dostępne w wersjach Windows 10, 11 oraz Windows Server.

- Służy do **konwersji partycji FAT16/FAT32 na NTFS** bez utraty danych.

### • Składnia polecenia:

- `convert [litera_dysku]: /FS:NTFS`

### • Przykład:

- `convert D: /FS:NTFS`

### • Parametry dodatkowe:

- `/V` – tryb szczegółowy
- `/NoSecurity` – dostęp do plików po konwersji dla wszystkich użytkowników
- `/X` – wymusza odmontowanie woluminu przed konwersją

## Charakterystyka konwersji FAT → NTFS

- **Jednokierunkowość**: konwersja z FAT do NTFS jest **nieodwracalna** – nie można użyć `convert.exe`, aby cofnąć proces.

- Aby przywrócić partycję do FAT, należy ją **sformatować**, co **usuwa wszystkie dane**.

- Zalecane jest wykonanie **kopii zapasowej** przed konwersją.

# Zmiana systemu plików

## Ograniczenia i wyjątki

• **Partycje systemowe na komputerach z procesorem RISC** (np. zgodne z ARC) **nie mogą być konwertowane na NTFS.**

- W takich przypadkach partycja systemowa **musi pozostać jako FAT.**
- Można utworzyć **małą partycję FAT (np. 1 MB)** jako systemową, a system Windows zainstalować na partycji NTFS (rozruchowej).

## Aktualność i zgodność

• Narzędzie `convert.exe` jest **wciąż aktualne i dostępne** w systemach Windows 10, 11 oraz Windows Server 2016–2025. [\[thewindowsclub.com\]](https://www.thewindowsclub.com)

• Obsługuje **FAT16 i FAT32**, ale **nie obsługuje exFAT** – w tym przypadku wymagane jest formatowanie. [\[techcommun...rosoft.com\]](https://www.techcommun.rosoft.com)

• Na systemach z architekturą RISC (np. RISC OS) nie jest możliwe użycie `convert.exe` do konwersji partycji systemowej. [\[riscosopen.org\]](https://www.riscosopen.org)

# Zmiana systemu plików

## Domyślne uprawnienia NTFS po konwersji

- **Domyślne uprawnienia NTFS nie są automatycznie stosowane** do partycji rozruchowej po konwersji z FAT.
- Aby zapewnić dostęp wszystkim użytkownikom, można użyć parametru:
  - `/NoSecurity`
- który **ustawia uprawnienia NTFS tak, aby wszyscy użytkownicy mieli dostęp** do plików i folderów po konwersji. [\[renenyffenegger.ch\]](https://renenyffenegger.ch)

## Dostępność `convert.exe` w Windows 11

- Narzędzie `convert.exe` jest **nadal dostępne i wspierane** w systemach Windows 10, 11 oraz Windows Server. [\[renenyffenegger.ch\]](https://renenyffenegger.ch)

# chkdsk

## ✓ CHKDSK – aktualne informacje (Windows 11/10)

- **CHKDSK (Check Disk)** to narzędzie firmy Microsoft dostępne od czasów **MS-DOS**, obecne również w **Windows 11**.
- Służy do **sprawdzania i naprawy błędów logicznych** systemu plików oraz **identyfikacji uszkodzonych sektorów** na dyskach HDD i SSD. [\[solveyourtech.com\]](https://solveyourtech.com)

## 🔧 Funkcje CHKDSK

- **Naprawa błędów systemu plików** – np. uszkodzone wpisy katalogów, błędne metadane, nieczytelne pliki. [\[pl.easeus.com\]](https://pl.easeus.com)
- **Wykrywanie i oznaczanie uszkodzonych sektorów:**
  - Soft bad sectors – logiczne, możliwe do naprawy.
  - Hard bad sectors – fizyczne, oznaczane jako nieużywalne. [\[elevenforum.com\]](https://elevenforum.com)
- **Działa również przed uruchomieniem systemu**, jeśli zaplanowano skanowanie partycji systemowej. [\[thewindowsclub.com\]](https://thewindowsclub.com)

## 💻 Uruchamianie CHKDSK

- Można uruchomić z:
  - **Wiersza poleceń (CMD)** jako administrator
  - **Windows Terminal**
  - **Eksploratora plików** (zakładka „Narzędzia” → „Sprawdź”)
  - **Windows Recovery Console** (np. z nośnika instalacyjnego) [\[allthings.how\]](https://allthings.how)

# chkdsk

## ⚠ Uwaga – uprawnienia NTFS

- **Domyślne uprawnienia NTFS nie są stosowane automatycznie** do partycji rozruchowej po konwersji z FAT.
- Aby zapewnić dostęp, można użyć parametru `/NoSecurity` przy konwersji. [\[youtube.com\]](https://www.youtube.com)

## vs CHKDSK vs SCANDISK

- **SCANDISK** był używany w starszych systemach (Windows 95–XP) i obsługiwał tylko FAT.
  - **CHKDSK** obsługuje zarówno FAT, jak i NTFS, ma więcej parametrów i działa w nowoczesnych systemach Windows.
- [\[darwinsdata.com\]](https://darwinsdata.com)

Wybierz Administrator: Wiersz polecenia

```
C:\Windows\System32>CHKDSK /?  
Checks a disk and displays a status report.
```

```
CHKDSK [volume[[path]filename]] [/F] [/V] [/R] [/X] [/I] [/C] [/L[:size]] [/B] [/scan] [/spotfix]
```

|          |                                                                                                                             |
|----------|-----------------------------------------------------------------------------------------------------------------------------|
| volume   | Specifies the drive letter (followed by a colon), mount point, or volume name.                                              |
| filename | FAT/FAT32 only: Specifies the files to check for fragmentation.                                                             |
| /F       | Fixes errors on the disk.                                                                                                   |
| /V       | On FAT/FAT32: Displays the full path and name of every file on the disk.<br>On NTFS: Displays cleanup messages if any.      |
| /R       | Locates bad sectors and recovers readable information (implies /F, when /scan not specified).                               |
| /L:size  | NTFS only: Changes the log file size to the specified number of kilobytes. If size is not specified, displays current size. |
| /X       | Forces the volume to dismount first if necessary. All opened handles to the volume would then be invalid (implies /F).      |

# fsck (File System Check)

- fsck to standardowe narzędzie w Linuksie do sprawdzania i naprawy błędów systemu plików.
- Działa na systemach plików takich jak ext2/ext3/ext4, XFS, Btrfs, FAT, NTFS (z ograniczeniami).

Przykładowe użycie:

**sudo umount /dev/sdX1**

**sudo fsck -y /dev/sdX1**

-y automatycznie zatwierdza naprawy.

Dla systemów NTFS można użyć:

**sudo ntfsfix /dev/sdX1**

`ntfsfix` naprawia podstawowe błędy NTFS i **wymusza uruchomienie CHKDSK przy następnym starcie Windows**. [\[hddfaqs.com\]](http://hddfaqs.com)

Dodatkowe narzędzia w Ubuntu:

| Narzędzie   | Zastosowanie                              |
|-------------|-------------------------------------------|
| fsck.ext4   | Sprawdzanie systemu plików ext4           |
| xfs_repair  | Naprawa systemu plików XFS                |
| btrfs check | Diagnostyka systemu plików Btrfs          |
| badblocks   | Wykrywanie uszkodzonych sektorów          |
| smartctl    | Monitorowanie stanu dysku (SMART)         |
| ddrescue    | Odzyskiwanie danych z uszkodzonych dysków |

# Pytania