Systemy zarządzania plikami

Ostatnia modyfikacja: 07.05.2021

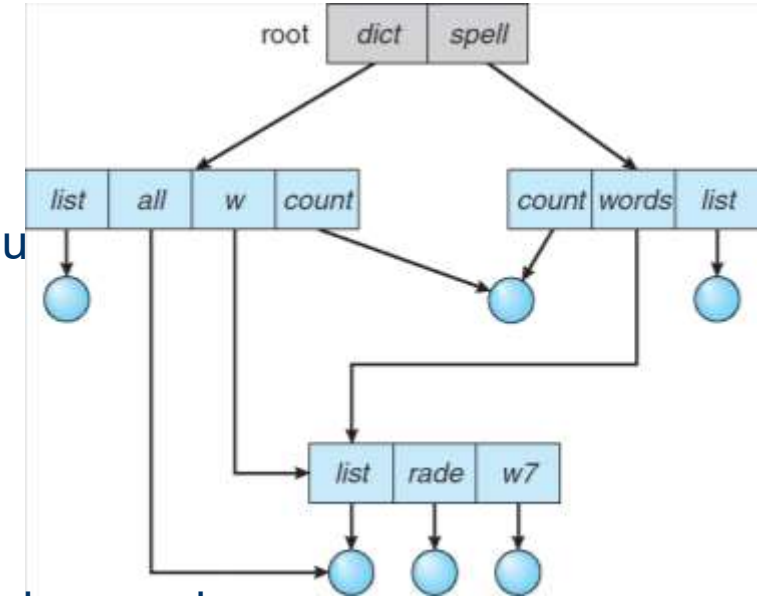
Spis treści

Pliki i systemy plików

- **Plik**:
  - Nazwana logiczna jednostka magazynowania informacji, przechowywana zwykle w pamięci (pomocniczej).
  - POSIX: „ **File** - an object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation”
- Pliki są zazwyczaj zorganizowane w **system plików**, pozwalający na zarządzanie przechowywaniem informacji na nośnikach oraz na wykonywanie operacji plikowych, np.: utwórz, usuń, otwórz dostęp, odczytaj, zapisz, zmień domyślną pozycję w pliku.
- System plików tworzy odwzorowanie nazwy pliku w położenie jego składowych na nośniku za pomocą struktury typu **Blok Kontrolny Pliku** *(File control block)*.
  - Systemy UNIX-owe: numer i-węzła, prawa dostępu, rozmiar, znaczniki czasu
  - MS Windows: NTFS przechowuje atrybuty pliku (w tym zawartość) w rekordach głównej tablicy systemu plików (*Master File Table*, MFT).
- Istnieją systemy plików (np. proc), które nie są odwzorowana na żaden nośnik danych; operacje plikowe umożliwiają dostęp do prywatnych zasobów jądra, czy informacji o własnościach systemu. W innych systemach plików, dostęp do plików specjalnych (UNIX: FIFO, /dev/*) daje plikowy dostęp do pewnej funkcjonalności jądra.

System plików

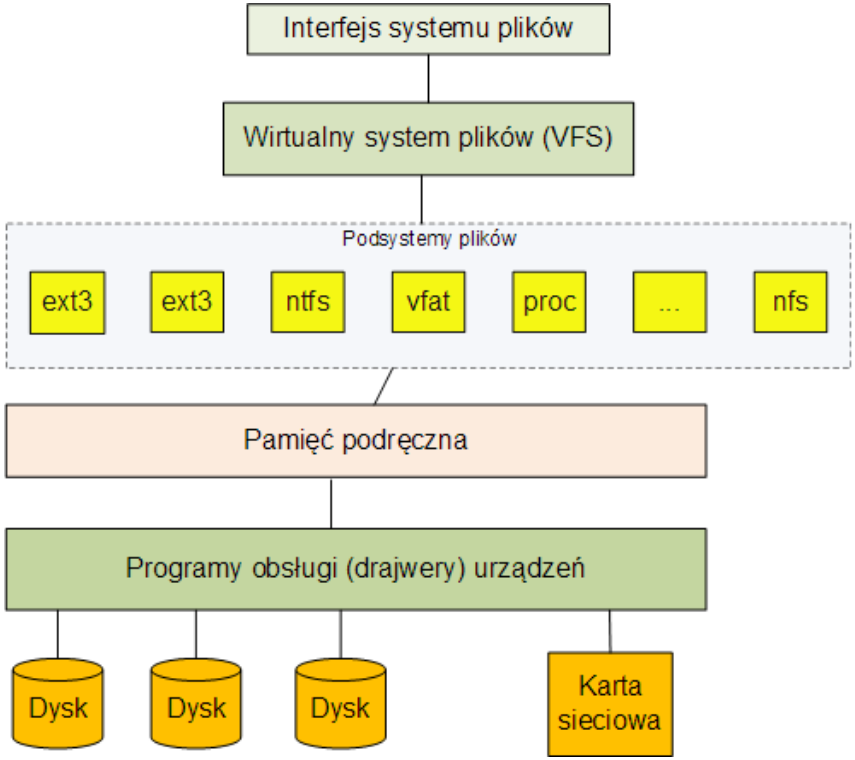
- System plików przechowuje zawartość plików (ew. forki pliku) oraz ich atrybuty (obowiązkowe,rozszerzone), a także listy (katalogi) plików – pozwalające na logiczne grupowanie plików.
- Logiczne relacje pomiędzy plikami i katalogami są reprezentowane (API) za pomocą grafu skierowanego, np. acyklicznego, z wyróżnionym węzłem katalogu głównego. Wskazywanie pliku w grafie – za pomocą nazw ścieżkowych (względem korzenia). Realizacja takiego grafu wymaga wprowadzania pomocniczych obiektów systemu pliku, np. dowiązań.



- Zadania ochrony, czy zachowywania spójności są wspierane przez specjalne obiekty systemu plików (np. ACL, dziennik).
- Przed użyciem konkretnego systemu plików w aktywnym systemie operacyjnym, musi on zostać logicznie dołączony do systemu (montaż systemu plików). W ten sposób możliwe jest wskazanie, za pomocą bezwzględnej ścieżki plikowej, każdego obiektu plikowego przechowywanego przez zamontowane (pod)systemy plików.

Zadania systemu operacyjnego

- Udostępnienie interfejsu operacji plikowych (API).
- Efektywna realizacja operacji plikowych przy zachowaniu założonych zasad bezpieczeństwa i spójności systemu plików.

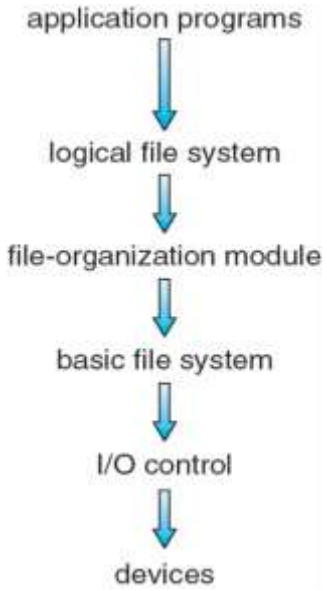


Warstwowa struktura podsystemu zarządzania systemami plików

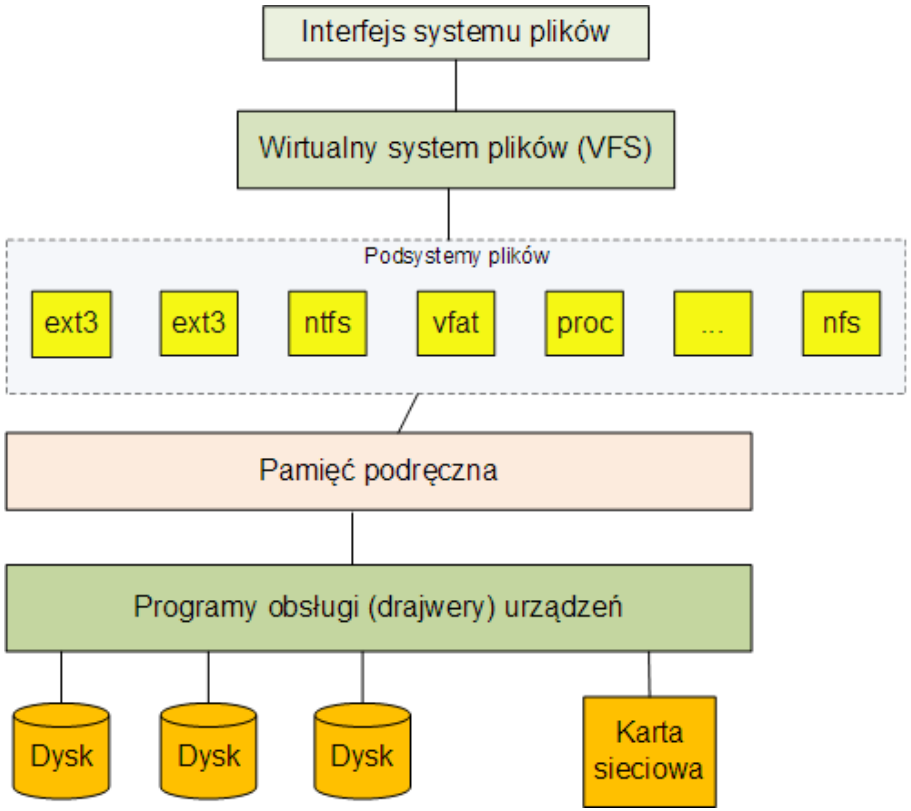
- Udostępnienie narzędzi do
  - badania spójności systemu plików
  - archiwizacji (backup)
  - rekonstrukcji systemu plików.

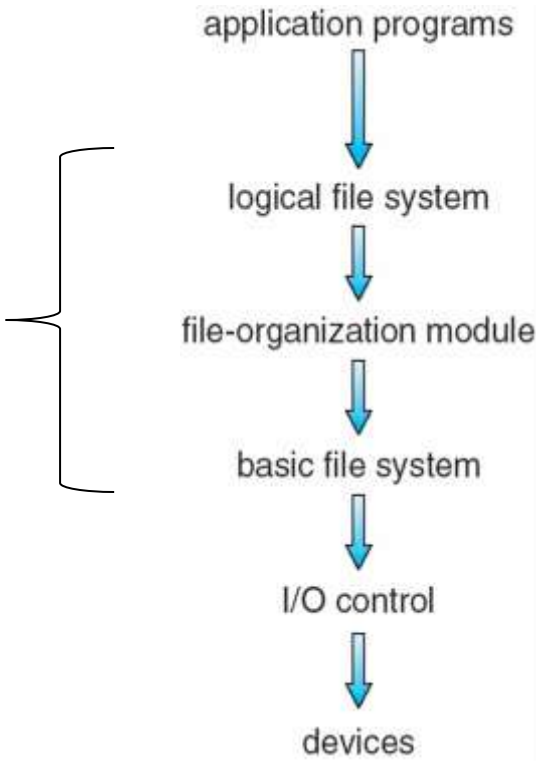
Warstwowa budowa (dyskowego) systemu plików

- **Moduł logicznego systemu plików** zarządza metadanymi: w katalogach i blokach kontrolnych plików (FCB). Utrzymuje odwzorowanie nazwy ścieżkowej pliku na położenie fragmentów na dysku. Zajmuje się ochroną.
- **Moduł organizacji plików** tłumaczy logiczne numery bloków dyskowych na numery fizyczne. Zarządza przydziałem pamięci i pamięcią wolną.
- **Podstawowy system plików** tłumaczy komendy postaci “retrieve block 123” na wywołania funkcji drajwera dysku. Ponadto zarządza buforami (przechowują przesyłane dane) i pamięcią podręczną (przechowuje często używane dane)
- **Drajwer dysku** (software) steruje dyskiem za pomocą **kontrolera dysku** (hardware).
  - Tłumaczy komendy typu: “read drive1, cylinder 72, track 2, sector 10, into memory location 1060” na sekwencje sterujące kontrolerem.
- **Dysk** zapewnia odczyt/zapis w trybie bezpośrednim (blokowo).
  - Tradycyjną jednostką magazynowania/przesyłania jest **sektor** (512B), obecnie spotykane są sektory 4kB
  - Możliwy podział na **partycje/woluminy** z niezależnymi systemami plików
  - **Blok kontrolny partycji** (*superblock, master file table*) zawiera informacje o rozmiarach bloków, liczbie i położeniu wolnych bloków i gł. katalogu, itd.)



Podsystem zarządzania plikami

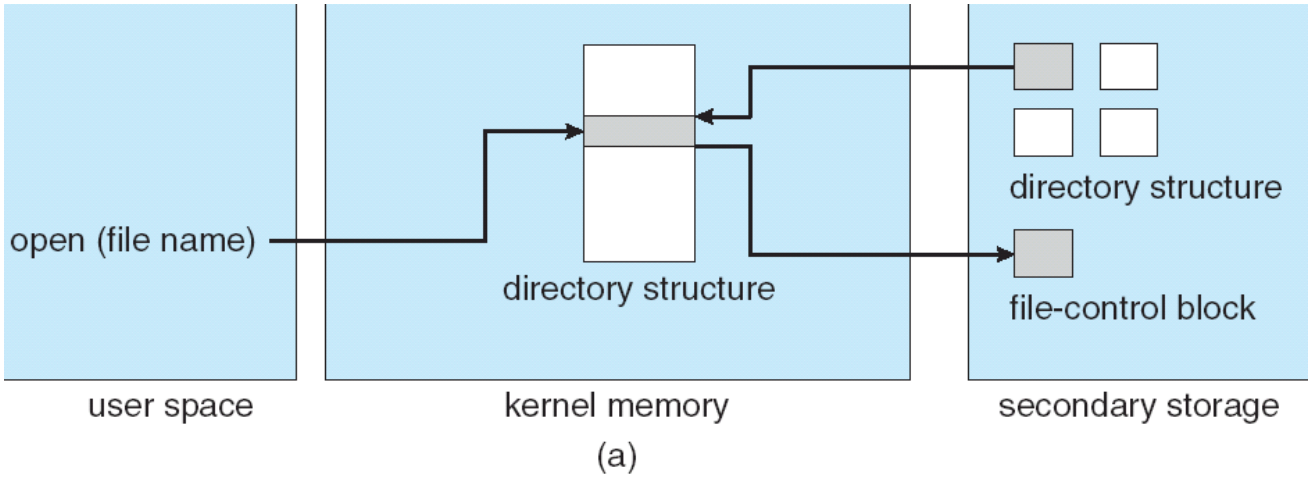




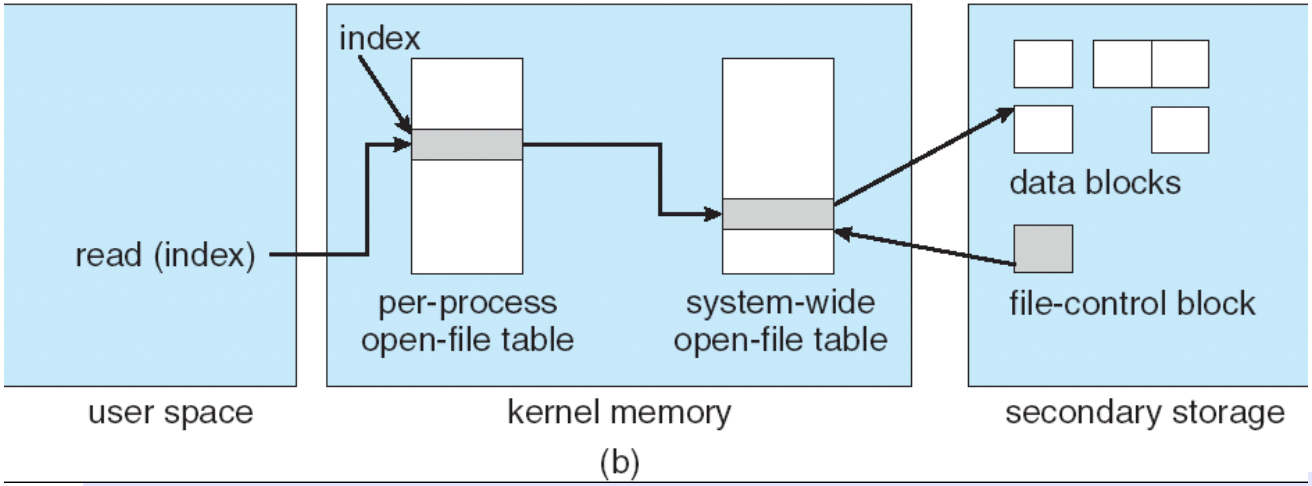
Hierarchiczna organizacja podsystemu zarządzania plikami

Warstwy obsługi operacji na pliku konkretnego (pod)systemu plików

Interfejs systemu plików



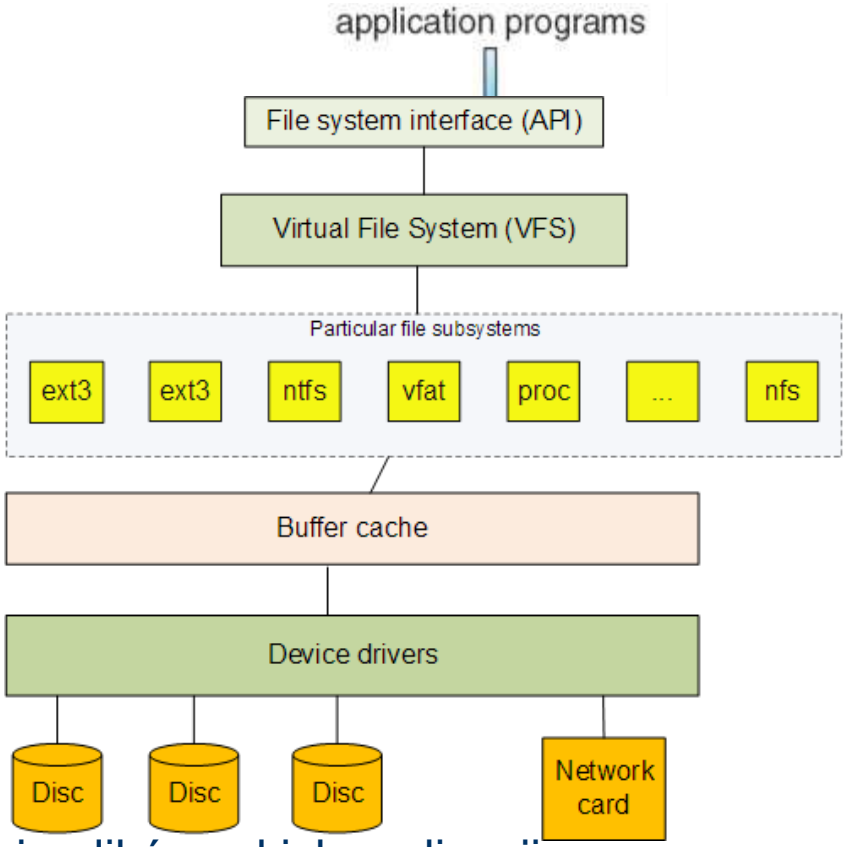
a) otwarcie pliku



b) odczyt pliku

Zarządzanie wieloma systemami plików

- System operacyjny
  - może wspierać ustaloną, niewielką liczbę systemów plików (np. FAT, NTFS); brak rozszerzania ułatwia efektywną implementację.
  - może wspierać nieograniczoną (?) liczbę systemów plików (FS) – o ile każdy wspierany system plików jest konkretyzacją pewnego **wirtualnego systemu plików** (VFS)



- Wirtualne systemy plików (Virtual File Systems, VFS) realizują zorientowaną obiektowo (*object-oriented*) metodę implementacji systemu plików.
  - Oddzielone są ogólne operacje na systemie plików od ich realizacji (specyficznych dla konkretnego FS).
  - Jądro reprezentuje wszystkie obiekty plikowe za pomocą takich samych struktur (Linux: vnodes), przekazując strukturę do procedury wykonawczej konkretnego FS – w razie potrzeby wykonania operacji na obiekcie plikowym.
  - API FS jest interfejsem VFS, a nie żadnego konkretnego systemu plików.

VFS systemu Linux

- Użytkownik widzi drzewo systemu plików z jednym korzeniem (/)
- Jądro ukrywa szczegóły realizacyjne w tym to, że widoczne drzewo plików jest zbudowane z podsystemów plików przy pomocy abstrakcji wirtualnego systemu plików (virtual file system, VFS)
- VFS systemu Linux obsługuje 4 typy obiektów:
  - struktura **inode** reprezentuje plik
  - struktura **file** reprezentuje otwarty plik
  - struktura **superblock** reprezentuje system plików
  - struktura **dentry object** reprezentuje pojedynczy obiekt katalogowy (plik, katalog), buforowany dla szybszego dostępu do obiektu.
- Przykładowo dla obiektu file zdefiniowano następujące operacje (patrz **struct file_operations w /usr/include/linux/fs.h**)

  int open(. . .) — Open a file

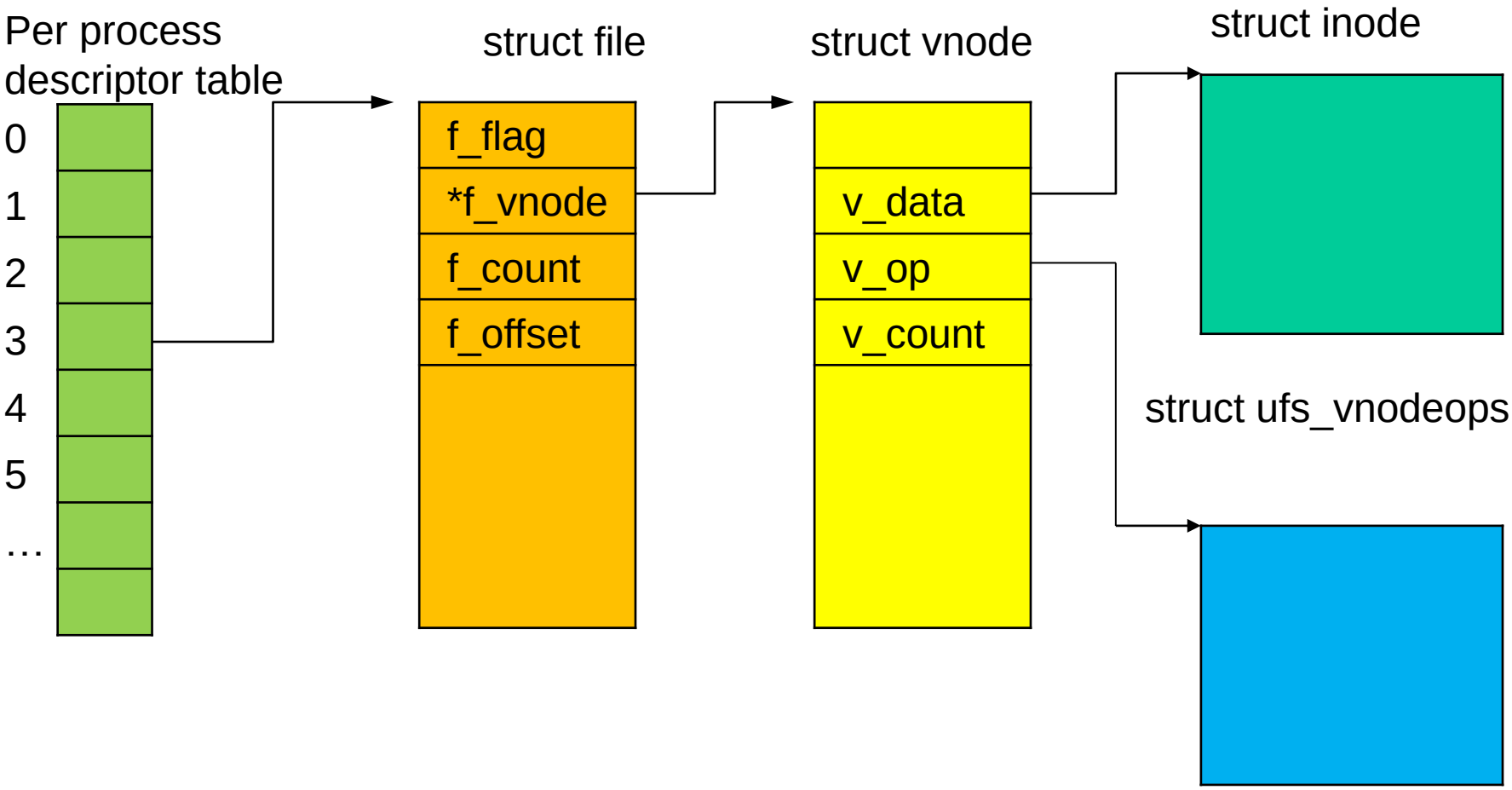
  ssize t read(. . .) — Read from a file

  ssize t write(. . .) — Write to a file

  int mmap(. . .) — Memory-map a file

Struktury danych

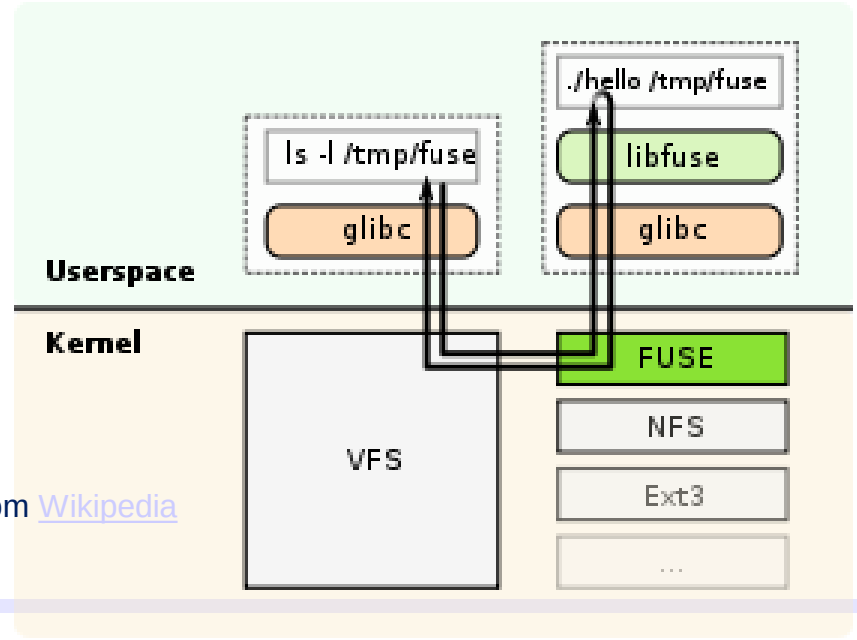
- Struktury danych, które są użyte w SO Linux do wiązania deskryptora pliku z plikiem (tu: z systemu plików UFS)



Rozszerzanie VFS - FUSE

- **F**ilesystem in **User**space (FUSE) jest programistycznym interfejsem dostępnym w systemach Unix/Linux, który umożliwia użytkownikom nieuprzywilejowanym tworzenie i użytkowanie własnych systemów plików bez konieczności modyfikacji jadra systemu. Kod obsługujący system plików działa w trybie użytkownika. FUSE pozwala na podłączenie tego kodu do interfejsów jądra obsługujących systemowe funkcje wejścia/wyjścia.

  System plików wykorzystujący FUSE jest zazwyczaj realizowany jako program, który korzysta z biblioteki libfuse. libfuse dostarcza dwa interfejsy (API): synchroniczny i asynchroniczny.

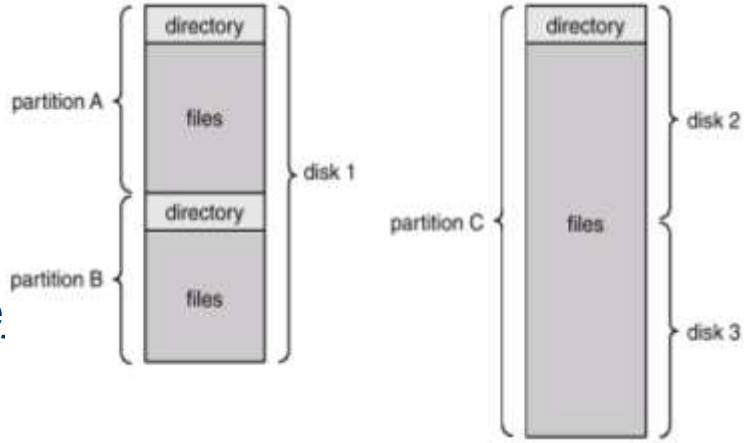


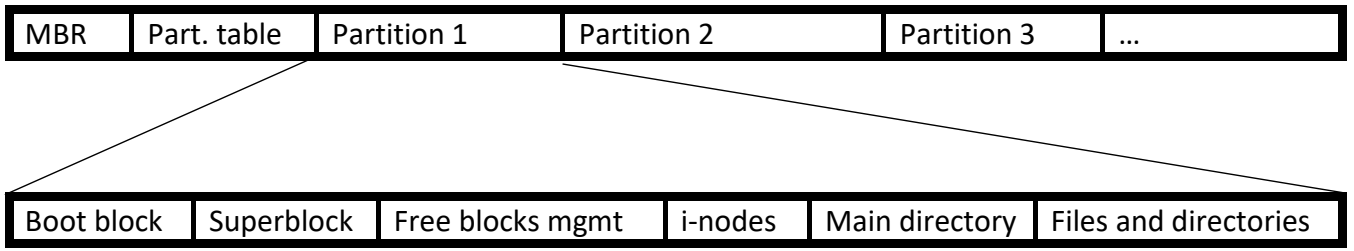
From Wikipedia

- **Zagrożenia**
  - dostarczony system plików nie musi spełniać żadnych warunków, a współpracujące programy mogą nie być na to przygotowane
  - Demon systemu plików może gromadzić informacje o swojej pracy -> potencjalny wyciek informacji.

Typowa organizacja dysku

- Typowa organizacja (MBR) dysku w systemie zarządzania plikami
  - MBR (*Master Boot Record*) – główny rekord startowy, kod z MBR czyta tablicę partycji i inicjuje ładowanie systemu z partycji aktywnej
  - *Boot block* – blok startowy systemu partycji, ładuje SO z tej partycji
  - Pozostała pamięć urządzeń podzielona jest logicznie na strefy (partycje, wolumeny, tomy,…).
- Sformatowana strefa zawiera nagłówek (*superblock*), główny katalog oraz miejsce na podkatalogi i pliki. Budowa strefy (określona przy formatowaniu) jest związana z realizacją funkcjonalności konkretnego systemu plików (np. ext4, NTFS). Parametry strefy (np. rozmiar grona) są przechowywane w nagłówku strefy.





Przydział bloków strefy logicznej

- Sformatowany obszar nośnika, reprezentujący strefę logicznej (partycji), przechowuje następujące typy obiektów
  - Katalogi
  - Pliki, zwykłe i specjalne (charakterystyczne dla danego systemu plików)
  - Zbiory bloków pamięci:
    - bloków wolnych (nieprzydzielonych) i
    - bloków uszkodzonych i rezerwowych
  - Dziennik (opcjonalnie)

- System plików może posługiwać się jednostką alokacji bloków, która jest inna niż fizyczna jednostka alokacji pamięci nośnika.
- Maksymalna liczba obiektów systemu pliku każdego typu może być ograniczana oddzielnie, a ograniczenia ustalane w trakcie formatowania strefy (partycji).

Organizacja katalogów

- Liniowa **lista** nazw plików ze wskaźnikami (*pointer*) do bloków danych.
  - Prosta do zaprogramowania
  - Czasochłonna w działaniu – dla bardzo dużych list.

- **Tablica rozproszona** (*hash hable*).
  - Zmniejsza czas przeszukiwania katalogu
  - *kolizje* – sytuacje gdy dwie nazwy plików są odwzorowane na tę samą wartość funkcji haszującej.
  - Stała wielkość
- **Drzewo,** np. B+
  - Zmniejsza się czas przeszukiwania dużych katalogów, ale częste usuwanie/dodawanie pozycji psuje efektywność.

➢Katalogi mogą być reprezentowane jako pliki specjalne, bezpośrednio dostępne jedynie dla systemu. W każdym przypadku procesy powinny wykonywać operacje na katalogach za pomocą funkcji systemowych.

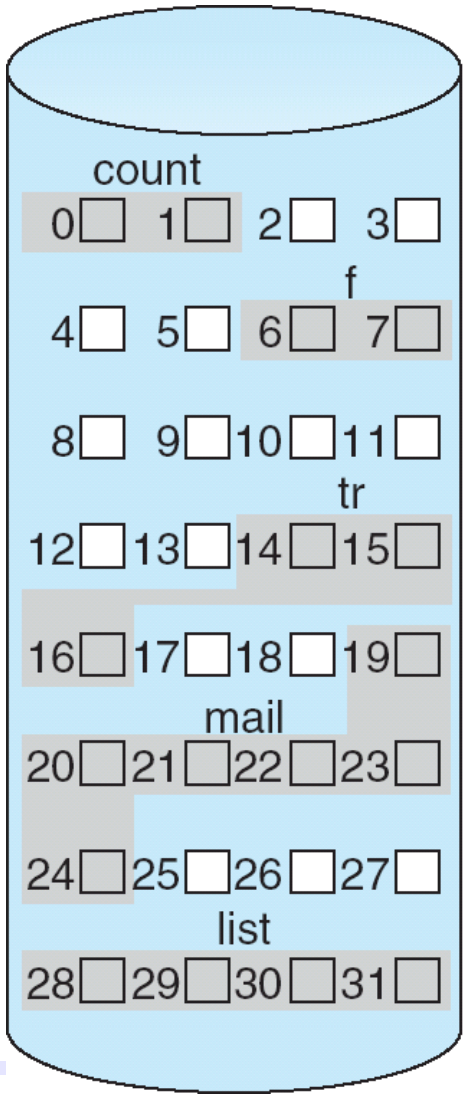
Metody przydziału miejsca dla plików

- Metoda przydziału (alokacji) określa sposób przydziału bloków dyskowych dla zapamiętania informacji składających się na plik. Dalej omówiono podstawowe metody:
  - przydział ciągły (*Contiguous allocation*)
  - przydział listowy (*Linked allocation*)
  - przydział indeksowy (*Indexed allocation*)

- Rzeczywiste systemy plików mieszają podejścia podstawowe. Pomocniczo wykorzystywane są też takie organizacje, jak:
  - **sterta** (dane są gromadzone bardzo szybko i przechowywane przed ich przetworzeniem – w oparciu o zawartość)
  - **obszar** (*extent*, fragment pliku/katalogu; sekwencja przylegających do siebie bloków dyskowych), poprawiająca przepustowość I/O oraz zmniejszająca narzut na zarządzanie.
  - **płyty pamięci (*slabs*)** – patrz materiały dot. zarządzania pamięcią jądra (alokator płytkowy).

Przydział ciągły

- Każdy plik zajmuje zbiór przylegających do siebie bloków dysku.
- Prostota – przydział określa położenie początku pliku (nr bloku) i długość (liczba bloków).
- Dostęp bezpośredni.
- Rozrzutne gospodarowanie dyskiem (problem dynamicznego przydziału, fragmentacja zewn.).
- Pliki nie mogą się dowolnie powiększać.



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Przydział ciągły – c.d.

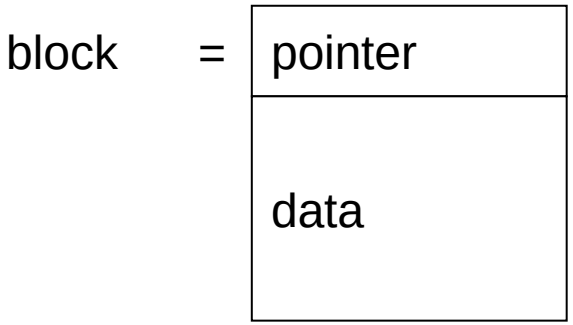
- Szereg nowych systemów plików (np. Veritas File System, ext4, NTFS, ISO9660, UDF,XFS) używa zmodyfikowanego schematu alokacji ciągłej, przydzielającego plikowi jeden lub więcej **obszarów** (*extents*), zwanych też **porcjami nadmiarowymi**.

- **Porcja nadmiarowa** (*extent*): zbiór bloków tworzących ciągły obszar. Informacja o porcji nadmiarowej: położenie pierwszego bloku i liczba bloków (a nie adresy wszystkich bloków składowych) -> mniejsza zajętość Bloku Kontrolnego Pliku, szybszy przydział/zwalnianie pamięci.

- **Przykład**. W systemie plików UDF, wykorzystywanych na płytach DVD, długość pliku reprezentuje 30-bitowa liczba; ogranicza to długość plików do 1GB. Każdy film (pliki logiczny) jest zazwyczaj zapisywany w postaci kilku obszarów, z których każdy jest ciągły.

Przydział listowy

- Każdy plik tworzy lista bloków dyskowych, które mogą być rozrzucone po całym dysku.

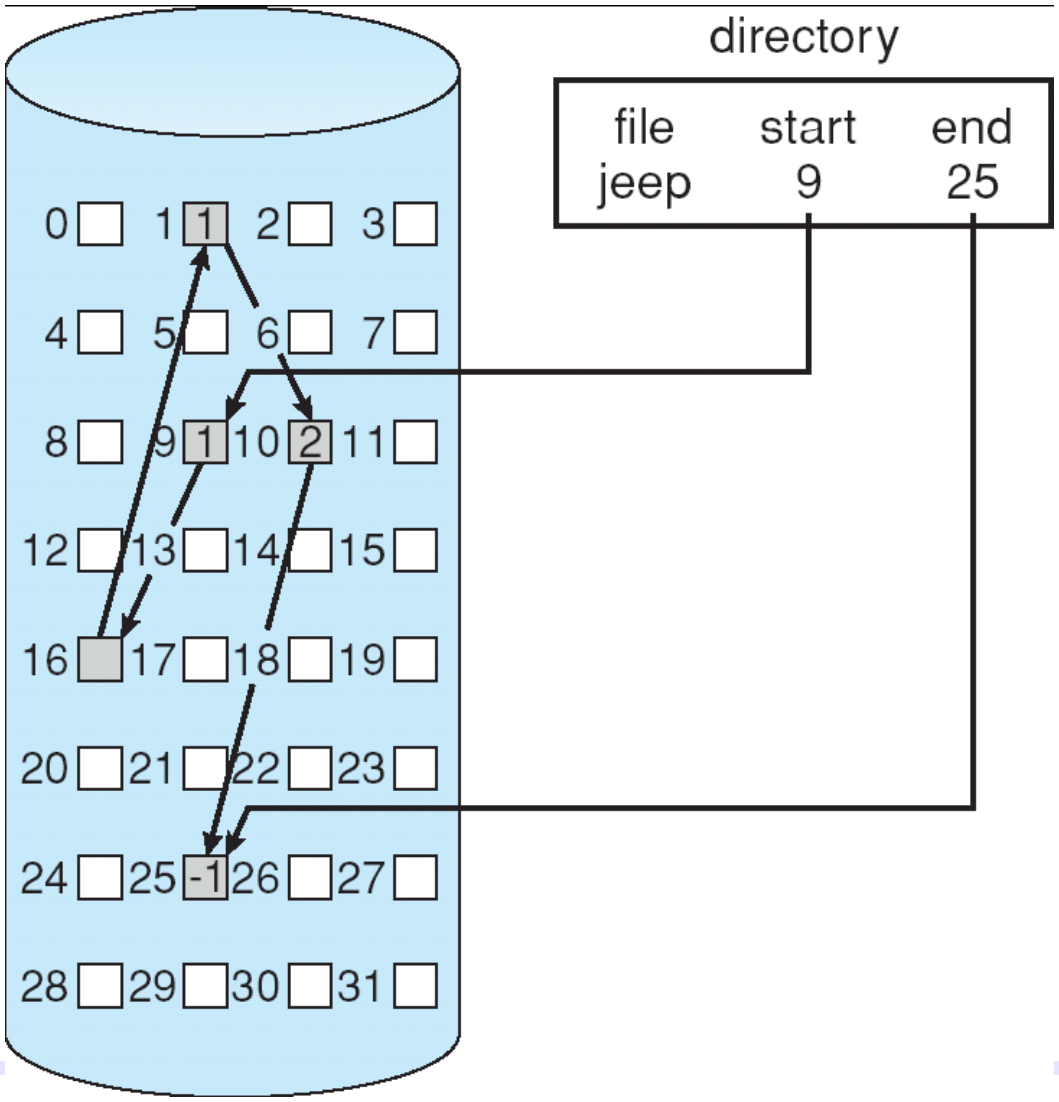


Cechy charakterystyczne przydziału listowego

- prostota
- nie ma zewnętrznej fragmentacji pamięci
- nie ma dostępu swobodnego (*random access*)

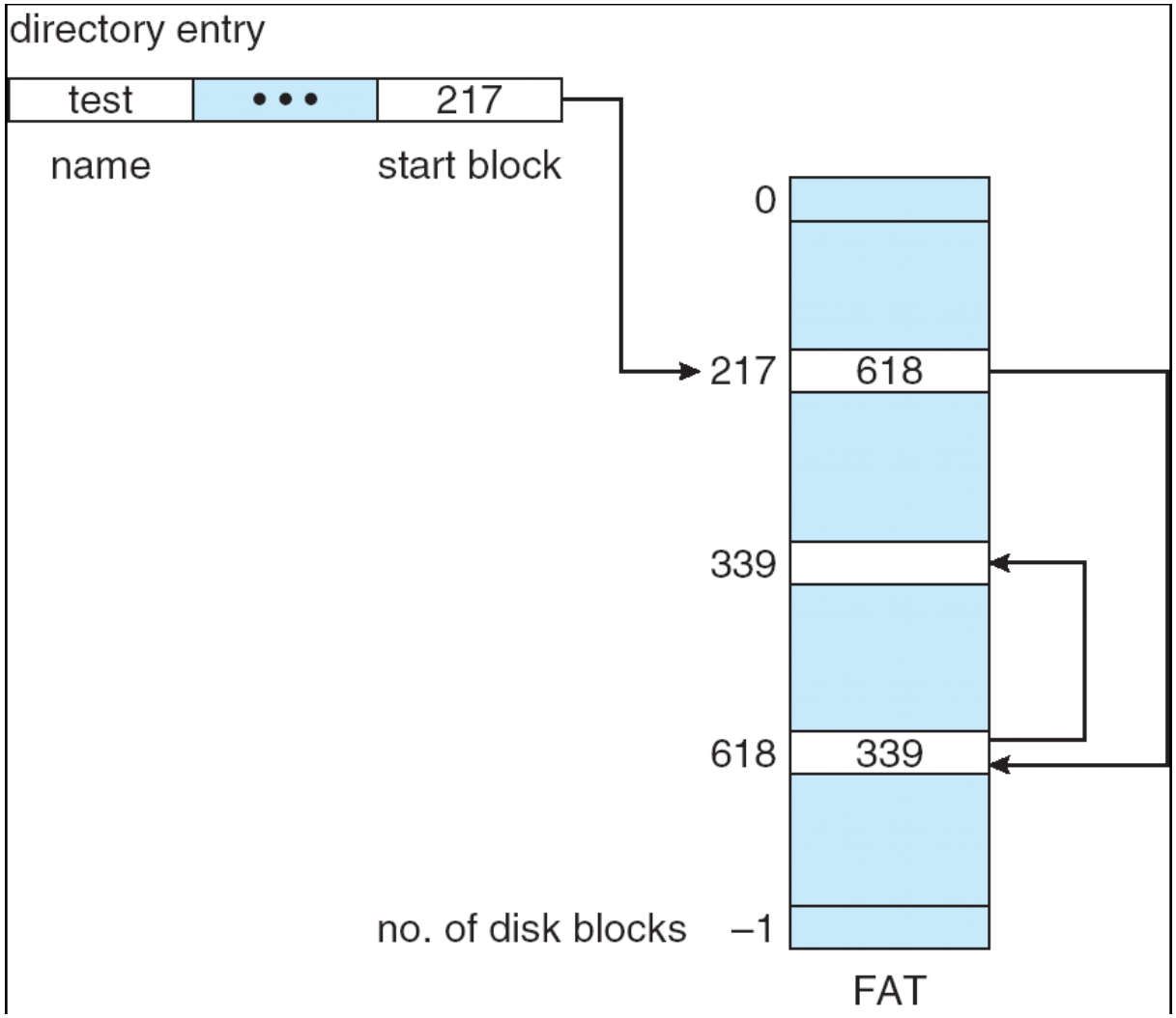
Dla zmniejszenia rozmiaru przestrzeni zajętej przez wskaźniki, bloki danych grupuje się w grona (*clusters*).

Przydział listowy - przykład



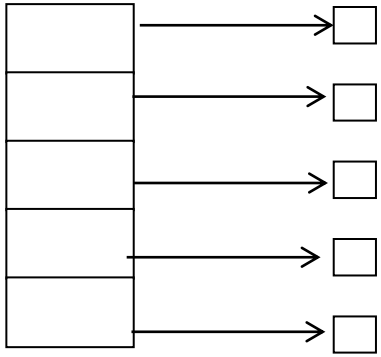
Zmodyfikowany przydział listowy – FAT

- **File-allocation table** (FAT) – tablica ulepszonej metody przydziału listowego, wykorzystywanej np. w systemach MS-DOS and OS/2. Lista numerów bloków każdego pliku przechowywana jest w FAT, a nie przy bloku danych, co poprawia szybkość dostępu, a także przydziału nowych bloków.



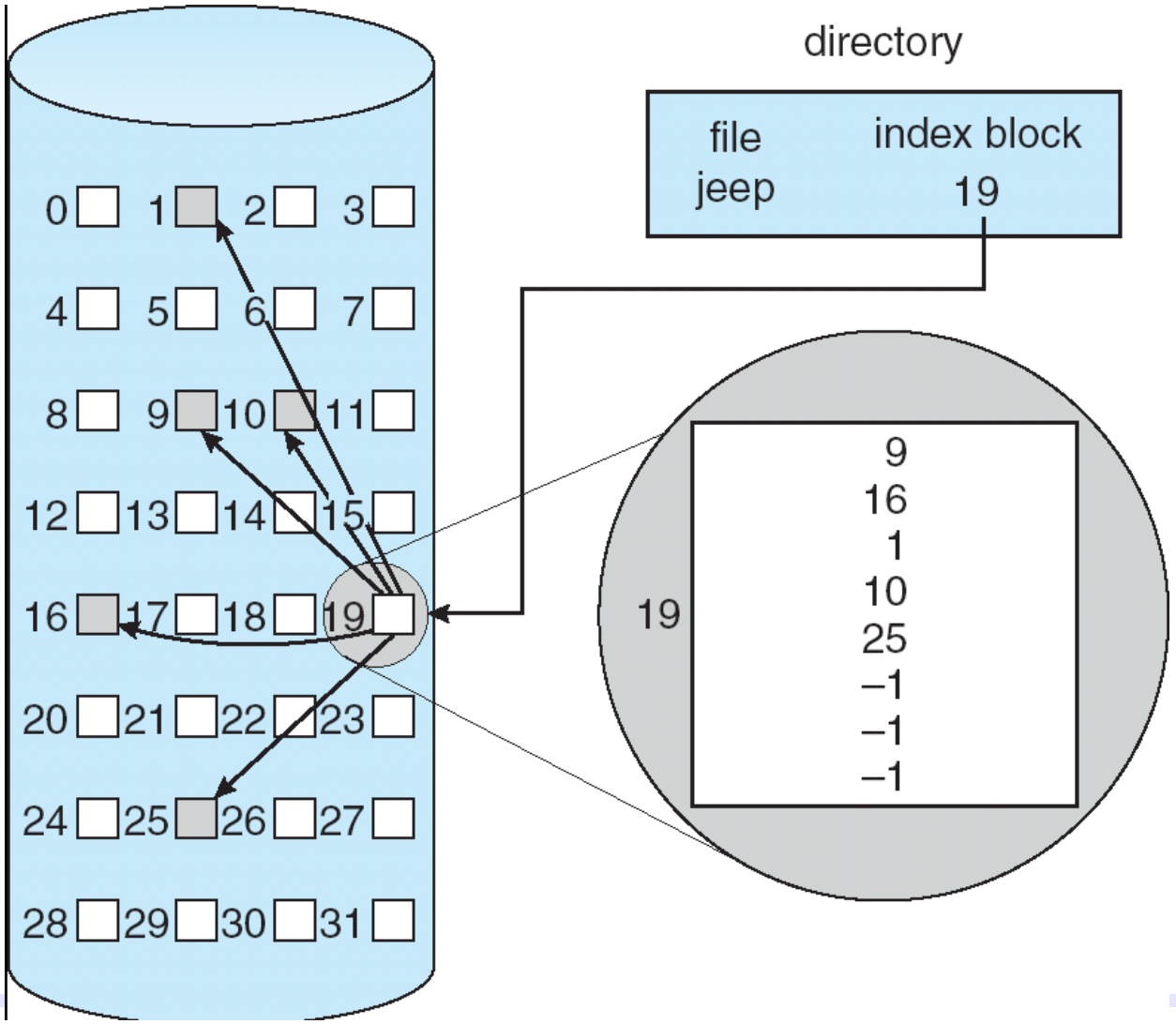
Przydział indeksowy

- Wskaźniki bloków pliku są skupione w bloku indeksowym (*index block.)*



index table

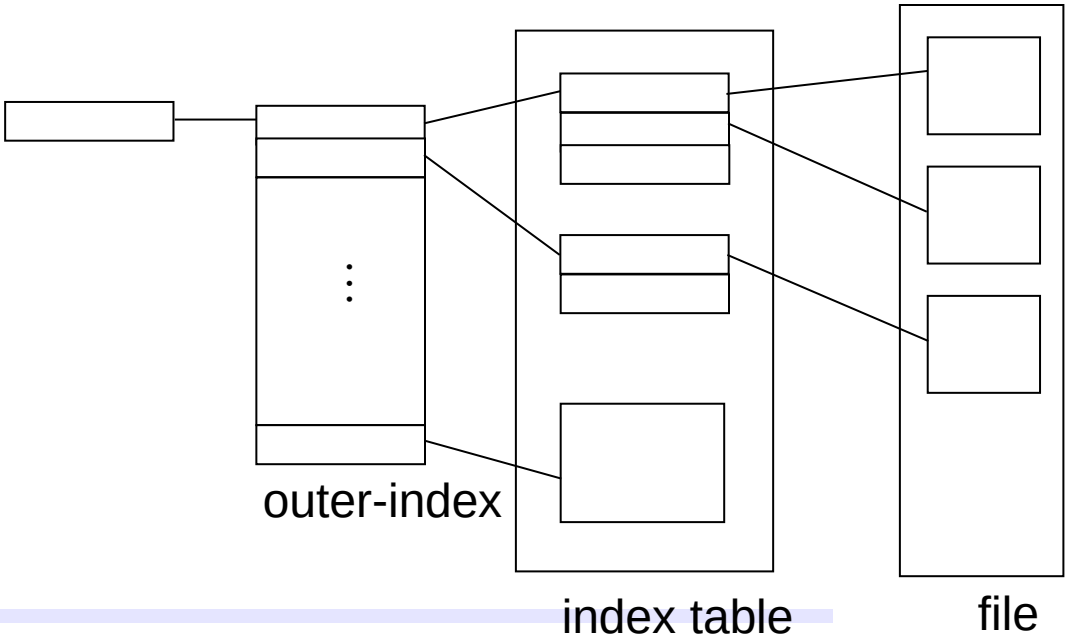
Przydział indeksowy - przykład



Przydział indeksowy (c.d.)

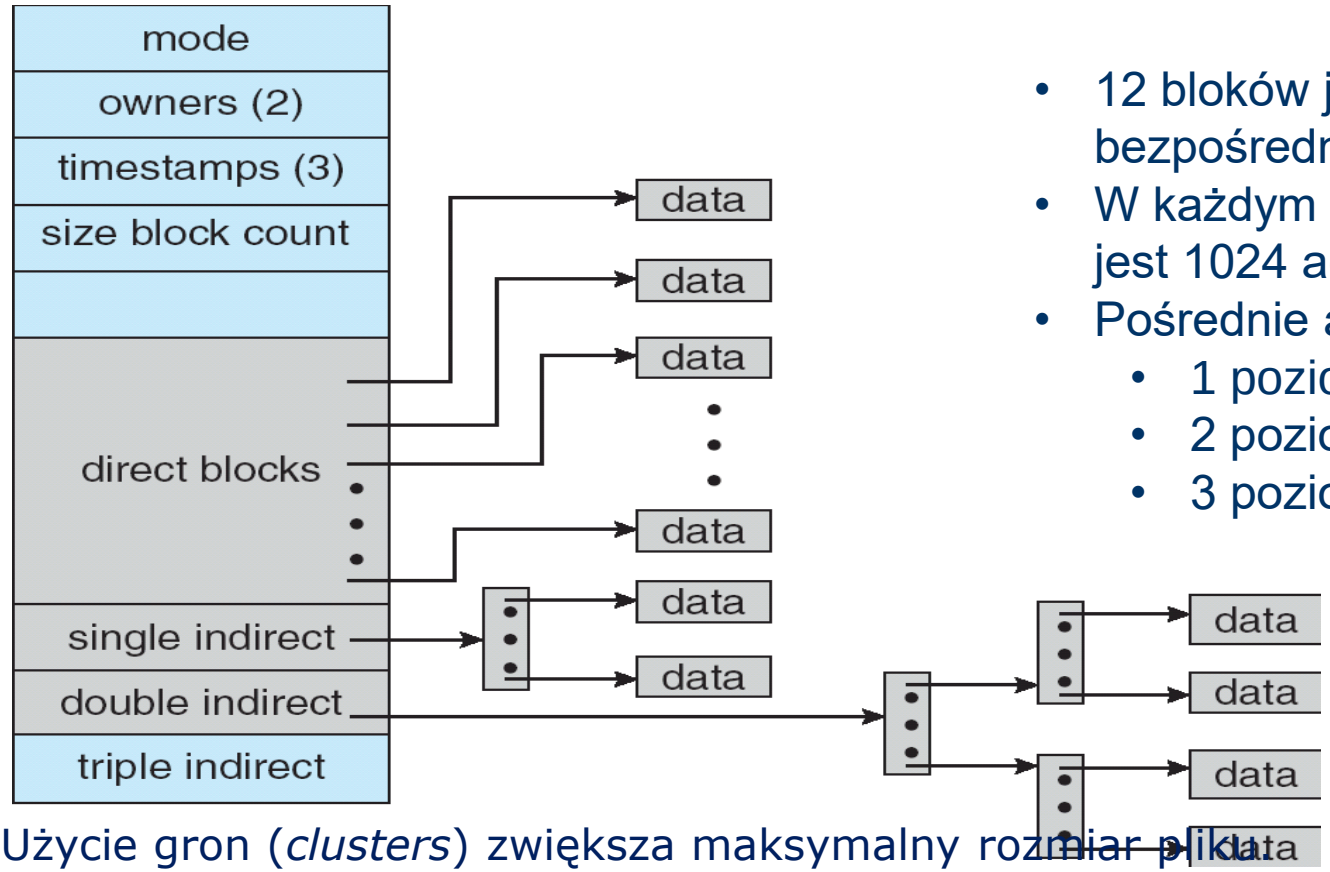
Cechy przydziału indeksowego.

- Możliwy jest dostęp swobodny
- Narzuty zwiazane z **blokiem indeksowym** (*index table*)
  - Pamięć
  - Dodatkowy czas, potrzebny na znalezienie adresu szukanego bloku danych w bloku indeksowym
- Maksymalna wielkość pliku jest iloczynem rozmiaru bloku indeksowego oraz rozmiaru bloku pliku.
- Dwupoziomowy indeks pozwala na znaczne zwiększenie maksymalnej długości pliku
- Odwzorowanie adresu logicznego w fizyczny dla pliku o nieograniczonej długości jest możliwe np. przy użyciu **listy bloków indeksowych**



Przydział mieszany: UNIX/Linux

Przykładowo: rozmiar bloku 4KiB, adresowanie 32-bitowe

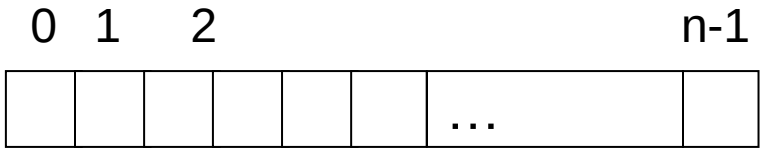


- 12 bloków jest dostępnych bezpośrednio
- W każdym bloku indeksowym jest 1024 adresów bloków
- Pośrednie adresowanie:
  - 1 poziom: 1024 adresy
  - 2 poziom: 1024^2 adresów
  - 3 poziom: 1024^3 adresów

- Użycie gron (*clusters*) zwiększa maksymalny rozmiar pliku.
- Systemy FFS/UFS wykorzystują fragmenty bloków (x512B) albo bloki o kilku rozmiarach (ext2) dla zmniejszenia fragmentacji wewnętrznej.
- Powiązane i-węzły/katalogi/pliki mogą być przechowywane w jednej grupie bloków/cylindrów - dla przyspieszenia dostępu .

Zarządzanie wolną przestrzenią

- Wektor bitowy (*n* blocks)



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

- Obliczenie numeru bloku:

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

Procesory mają instrukcje wyznaczające pozycję najmłodszego bitu 1

Zarządzanie wolną przestrzenią (c.d.)

- Mapa bitowa wymaga dodatkowego miejsca na dysku, np.:

  block size = 2^{12} bytes

  disk size = 2^{30} bytes (1 gigabyte)

  $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)

  - Łatwo tworzyć pliki z przylegającymi blokami (*contiguous files*)
  - W systemach plików z grupami cylindrów/bloków – każda grupa ma swoją pulę i-węzłów, bloków danych oraz mapy bloków wolnych. Poszukiwanie wolnych bloków rozpoczyna się od grupy, w którym jest i-węzeł pliku – co poprawia lokalność odniesień, a więc czas dostępu.

- Lista powiązana wolnych bloków (*free list*)
  - Nie można łatwo znaleźć przylegających bloków
  - Metoda wydajna, gdy potrzeba dowolnego wolnego bloku

- Grupowanie - pierwszy wolny blok zawiera adresy następnych n-1 bloków i adres następnego bloku z adresami.

- Zliczanie: bloki są przydzielane i zwalniane w grupach; pierwszy blok grupy zawiera liczność grupy (organizacja jak w porcjach nadmiarowych, *extents*)

Zarządzanie wolną przestrzenią (c.d.)

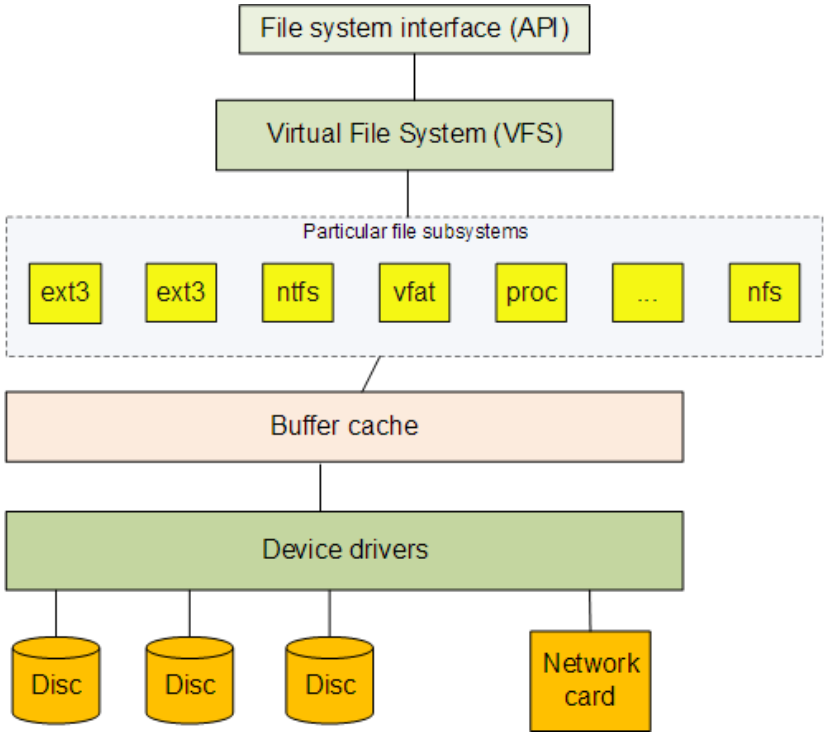
- Mapy wykorzystania obszaru pamięci (Space Maps)
  - Wykorzystywane w systemie plików **ZFS**
  - W przypadku bardzo dużych systemów plików wszystkie metadane (również mapy bitowe) nie mieszczą się w pamięci -> perspektywa dużej liczby operacji I/O w przypadku alokacji/usuwania dużych plików
  - W ZFS obszar pamięci urządzenia dyskowego dzielona jest na „płyty” (**metaslab** units)
    - Jeden wolumin może zawierać setki „płyt”
    - W każdym obszarze jest oddzielna mapa wykorzystania
    - Przydział wielu bloków naraz; wykorzystywana technika zliczania (wskaźnik początku i licznik bloków grupy) do reprezentacji przestrzeni wolnej
    - Zapis aktywności (przydział/zwalnianie bloków) jest zapisywany w porządku czasowym do logu, a nie na dysk
    - Dla uzyskania aktualnego stanu wolnych bloków log jest wczytywany tworząc w pamięci strukturę drzewo. Jest wówczas przeprowadzane scalanie sąsiednich bloków oraz aktualizacja mapy na dysku (w trybie transakcyjnym). Kolejne zmiany puli bloków wolnych są logowane itd..
    - Zbiór bloków wolnych określa więc drzewo i log

Efektywność i wydajność

- Efektywność wykorzystania miejsca na dysku zależy od:
  - algorytmów przydziału miejsca i algorytmów obsługi katalogów
  - rodzaju danych przechowywanych w pozycjach katalogowych (np. data ostatniego dostępu, czy tylko data modyfikacji)
  - alokacji pamięci dla matadanych: wstępna, czy na żądanie
  - tego, czy struktury danych mają stałą, czy zmienną długość
- Wydajność
  - Przechowywanie danych i ich metadanych w pobliżu
  - **Pamięć podręczna** bloków dyskowych (*disk cache*) – część pamięci operacyjnej zawierająca najczęściej używane bloki dyskowe.
  - **Zapis asynchroniczny** (popularny), z wykorzystaniem pamięci podręcznej, czy **zapis synchroniczny** (dane zapisane na dysku bez pośrednictwa pamięci podręcznej przed potwierdzeniem OK operacji)?
  - **Wczesne zwalnianie bufora** (*free-behind*) i **czytanie z wyprzedzeniem** (*read-ahead*) – dla optymalizacji dostępu sekwencyjnego
  - W PC można przeznaczyć część pamięci operacyjnej na **dysk wirtualny** (RAM disk); konieczny jest stosowny drajwer.

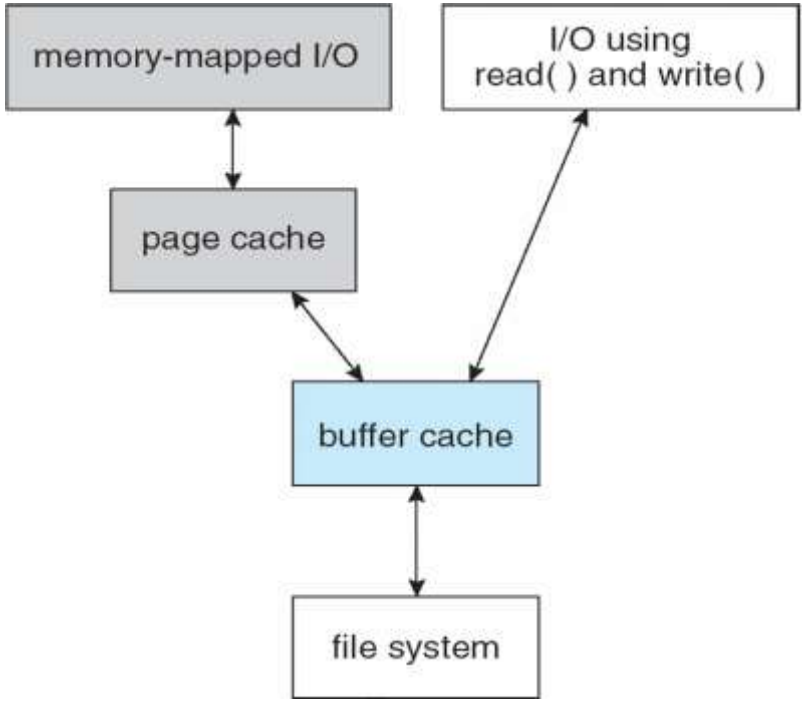
Pamięć podręczna I/O

- **Pamięć podręczna bloków dyskowych** (*disk cache*) – część pamięci operacyjnej zawierająca najczęściej używane bloki dyskowe.
  - zmniejsza liczbę operacji dyskowych
- Pamięć bloków dyskowych może być użyteczna
  - Do obsługi podsystemów wejścia/wyjścia
  - Do obsługi stronicowania/segmentacji na żądanie (*demand paging / segmentation*) oraz do wymiany (procesów, *swapping*).



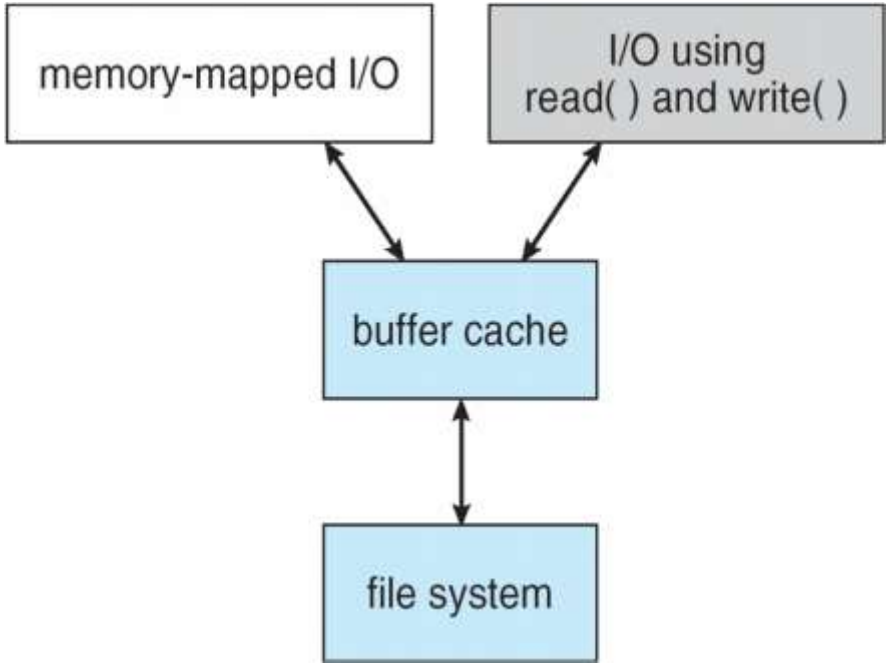
I/O bez jednolitej pamięci podręcznej

- Pamięć podręczna stron (*page cache*) przechowuje strony VM raczej niż bloki dyskowe
- Pamięć podręczna stron jest również wykorzystywana przez techniki odwzorowywania plików/urządzeń wej/wyj w pamięci (*memory-mapped I/O*)
- Funkcje read()/write() typowo wykorzystują bufory bloków dyskowych (pośredniczą w przesyłaniu danych pomiędzy dyskiem a pamięcią główną)



I/O z jednolitą pamięcią podręczną

- Jednolita pamięć podręczna, to wykorzystywanie tej samej puli bloków pamięci do przechowywania stron pamięci wirtualnej jak i kopii bloków dyskowych.
- Unika się podwójnego buforowania części bloków.
- Powstaje nowy problem wyboru priorytetów oraz konstrukcji algorytmów wymiany.



Uwagi

- Warto rozważyć dodanie wielu instrukcji kodu, by zaoszczędzić operacji wejścia wyjścia. Uzasadnienie:
  - Intel Core i7 Extreme Edition 990x (2011) at 3.46Ghz = 159,000 MIPS
    - http://en.wikipedia.org/wiki/Instructions_per_second
  - Dla typowego dysku twardego (HD) wykonującego 250 operacji wejścia/wyjścia w ciągu sekundy (I/O operations per second IOPS)
    - 159,000 MIPS / 250 = 630 milionów instrukcji procesora w czasie jednej operacji wejścia/wyjścia
  - Szybki dysk półprzewodnikowy (SSD) może wykonać ponad 60,000 IOPS ((https://en.wikipedia.org/wiki/IOPS)
    - 159,000 MIPS / 60,000 = 2.65 milionów instrukcji procesora w czasie jednej operacji wejścia/wyjścia
- Dla niektórych HDD kolejkowanie operacji przez kontroler dysku może podwoić IOPS (optymalizacja ruchu głowicy i kolejności operacji).
- Typowo, SSD wykazują duże IOPS przez ograniczony czas – przy stałym obciążeniu.

Przykład rachunkowy.

Wyznacz średni czas, potrzebny do wykonania operacji I/O na 4KiB bloku danych. Załóż średni czas pozycjonowania głowicy (*seek time*) 5ms dysku HDD o obrotach 7200/min; szybkość transmisji 1Gib/s, opóźnienie wprowadzane przez kontroler: 0.1ms. Odp.: 9.3ms.

Dziennik systemu plików

- **Systemy plików z dziennikiem/kroniką (***journaling file systems*) zapisują każdą modyfikację systemu plików jako **transakcję**.
- Wszystkie operacje transakcji są zapisywane do **dziennika/kroniki (*log file*)**. Transakcję uważa się za **zatwierdzoną** (*committed*) jeśli tylko zostanie zapisana do dziennika, chociaż system plików może nie być jeszcze uaktualniony. Takie zatwierdzenie pozwala zakończyć pomyślnie wywołanie funkcji systemowej.
- Transakcje z dziennika są asynchroniczne zapisywane do systemu plików. Kiedy system plików jest uaktualniony – daną transakcję usuwa się z dziennika.
- W przypadku awarii systemu należy uaktualnić system plików przy wykorzystaniu dziennika. Jeśli jednak transakcja nie została zatwierdzona (tzn. nie wszystkie operacje zostały zapisane w dzienniku) – musi zostać **wycofana** (*undone*)
- Dziennik może być plikiem specjalnym systemu pliku.
- Niektóre systemy plików pozwalają określić które operacje na systemie plików są realizowane w trybie transakcyjnym.

Sprawdzanie i rekonstrukcja systemu plików

- Program sprawdzania spójności (*consistency checker*) – porównuje dane w strukturze katalogowej z blokami danych na dysku i próbuje usunąć wszelkie napotkane niezgodności.
  - Wykorzystuje się wewnętrzną nadmiarowość informacji przechowywanej przez system plików do sprawdzania spójności
  - Sprawdzanie może być powolne i zawodne

  Przykłady programów sprawdzania spójności : fsck, chkdsk, sfc

- Składowanie (*backup*) plików dyskowych na innych urządzeniach do magazynowania informacji (dyskach elastycznych, taśmach magnetycznych, dyskach optycznych) pozwala na odtworzenie danych *(restoring)* w przypadku awarii.

- Backup może być:
  - fizyczny (urządzenia lub partycji) bądź logiczny (katalogów, plików).
  - całkowity, bądź przyrostowy,