

Wykład 10

Systemy plików

Interfejs i implementacja

Systemy plików

Przechowują olbrzymie ilości informacji

Gigabajty => Terabajty => Petabajty

Przechowywana informacja nie jest tracona po zakończeniu procesu.

Czas życia od sekund do lat.

Potrzebny jest sposób odnajdywania informacji

Wiele procesów musi być w stanie wykorzystywać tę samą informację współbieżnie.

Nazwy plików

Konieczność **odnalezienia** pliku, po tym jak został utworzony.

Każdy plik ma co najmniej jedną nazwę.

Nazwa może być:

Łatwa dla zrozumienia dla człowieka: np. “Notatka.txt”, “program3.cpp”.

Odczytywana bezpośrednio przez maszynę: np. 65231

Duże i małe litery mogą być rozróżniane lub nie

Nazwa może zawierać informacje o przeznaczeniu pliku.

Wygodne z punktu widzenia użytkownika.

Wykorzystywane przez programy użytkowe

Przykład (Nazwa z rozszerzeniem)

| | |
|---------|-----------------------|
| abc.jpg | Obraz w formacie jpeg |
| p1.c | Program w języku C |
| f.txt | Ogólny plik tekstowy |

Struktura pliku

Brak struktury: plik jest ciągiem bajtów.

Plik rekordów

Rekordy o stałej lub zmiennej długości.

Plik o skomplikowanej strukturze

Tekst sformatowany

Program wykonywalny.

Każdą strukturę pliku można zasymulować za pomocą ciągu bajtów i wykorzystując znaki kontrolne.

Dlatego często system operacyjny traktuje plik jako ciąg bajtów.

Struktura pliku jest ustalana na poziomie programu użytkownika (ale nie zawsze - przykład. pliki wykonywalne .exe)

Atrybuty pliku

Nazwa pliku

Rozmiar

Właściciel

Prawa dostępu

Czasy (utworzenia/ostatniego dostępu/ostatniej modyfikacji).

Hasło

Inne informacje, z reguły niedostępne dla programu użytkownika

Na przykład informacja o położeniu pliku na dysku

Operacje na plikach

Utworzenie (ang. create)

Usunięcie (ang. delete)

Otwarcie (ang. open) – przygotowanie pliku do dostępu.

Zamknięcie (ang. close) – wskazanie, że dostęp do pliku nie będzie dalej potrzebny.

Odczyt (ang. read) (do bufora w pamięci procesu)

Zapis (ang. write) (z bufora w pamięci procesu)

Przesunięcie wskaźnika bieżącej pozycji (ang. seek).

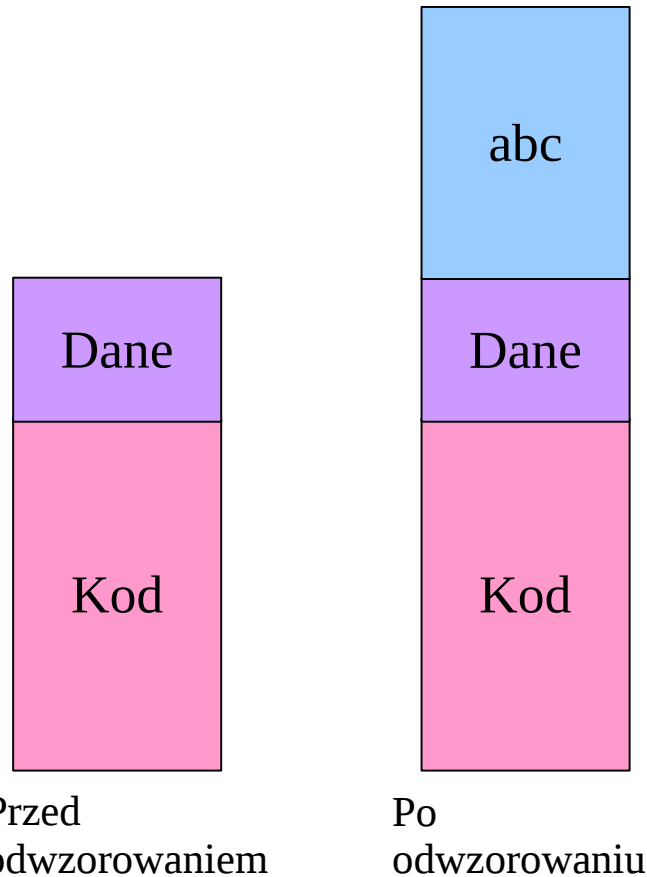
Operacje read oraz write wykonują odczyt oraz zapis z miejsca wskazywanego przez wskaźnik bieżącej pozycji; wartość wskaźnika jest zwiększana o liczbę odczytanych lub zapisanych bajtów.

Operacja seek pozwala na **dostęp swobodny** do pliku (ang. random access), jeżeli nie jest zaimplementowana mówimy o **dostępie sekwencyjnym**.

Dołączenie (ang. append) – zapis na końcu pliku

Odczyt/Zmiana atrybutów (w tym nazwy)

Pliki odwzorowane w pamięci (ang. memory mapped files)



Po otwarciu pliku wykonywana jest operacja mmap.

Plik staje się częścią przestrzeni adresowej procesu.

Wygodna abstrakcja

Unikamy podwójnego kopiowania danych

Implementacja na ogół wykorzystuje mechanizm pamięci wirtualnej

Problemy:

Plik współdzielony przez kilka procesów.

Próba dostępu “za końcem” pliku.

Katalogi

Proste nadawanie nazw nie wystarcza w przypadku tysięcy plików na dysku.

Problem: wiele plików o identycznej nazwie, np. różne wersje tego samego programu

Ludzie mają tendencję do grupowania informacji związanych ze sobą.

Systemy plików umożliwiają to przy pomocy katalogów (ang. directory), zwanych także folderami.

Grupowanie pozwala na.

Łatwiejsze znalezienie plików.

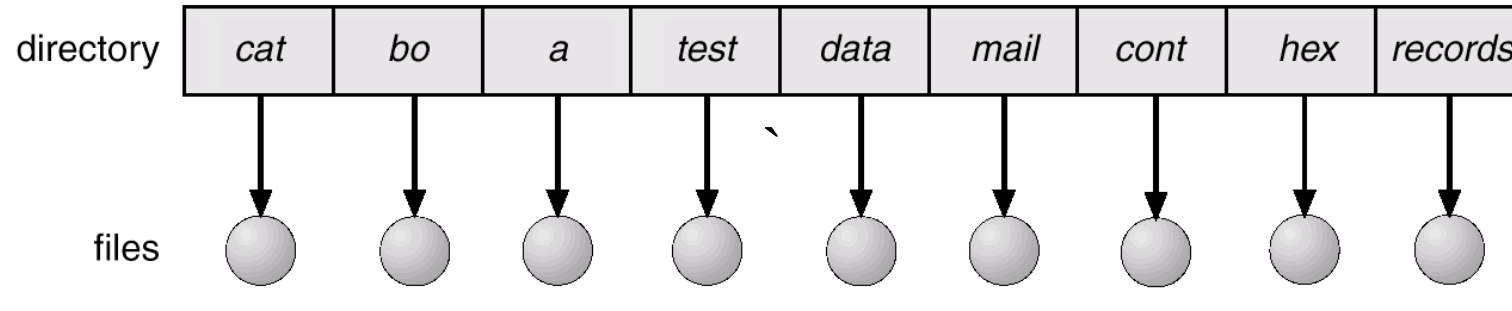
Określenie, które pliki są ze sobą związane.

Operacje na katalogach:

Głównie odczyt i przeszukiwanie katalogu.

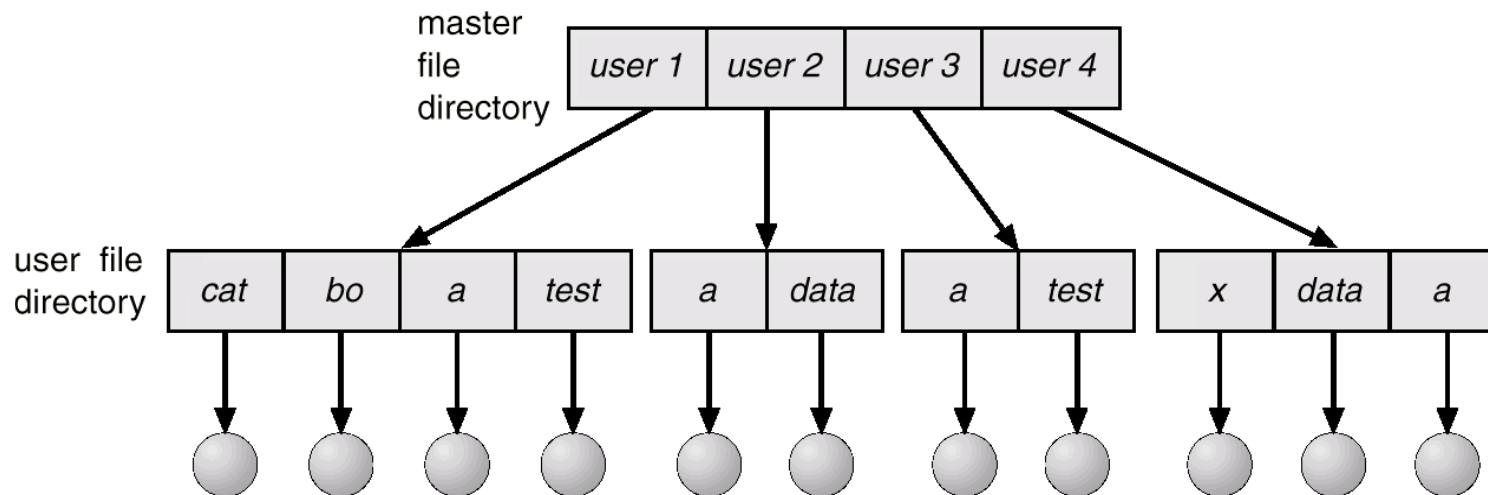
Także tworzenie nowych dowiązań.

Katalog o strukturze jednopoziomowej i dwupoziomowej



Jeden katalog dla wszystkich użytkowników.

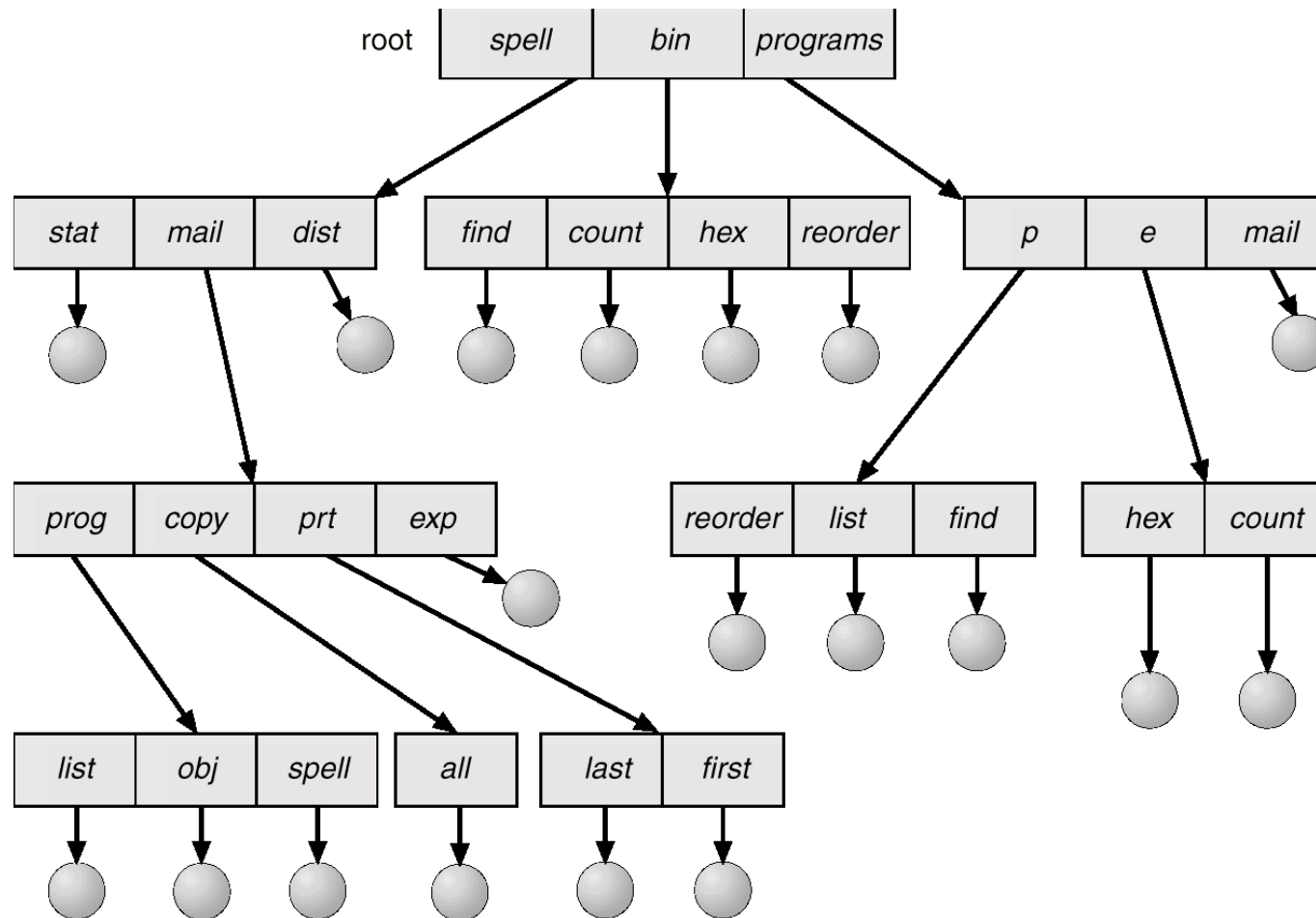
Problemy: (a) Nie pozwala na grupowania (b) Konflikty nazw plików.



Katalog główny + Odrębny katalog dla każdego z użytkowników

Nadal nie pozwala na grupowania plików

Katalog o strukturze drzewa



Umożliwia grupowanie.

Katalog aktualny (ang. current directory).

Lokalizacja pliku podana przez ścieżkę.

Ścieżka bezwzględna: początek w korzeniu drzewa.

Ścieżka względna: początek w katalogu aktualnym.

/bin/hex albo ./p/list (jeżeli /programs jest katalogiem głównym)

Katalog o strukturze grafu acyklicznego

Brak cykli w grafie chroni przed ścieżkami o nieskończonej długości.

Plik może mieć dwie różne nazwy (aliasy)

Problem wiszących wskaźników

Gdy usuniemy *list*

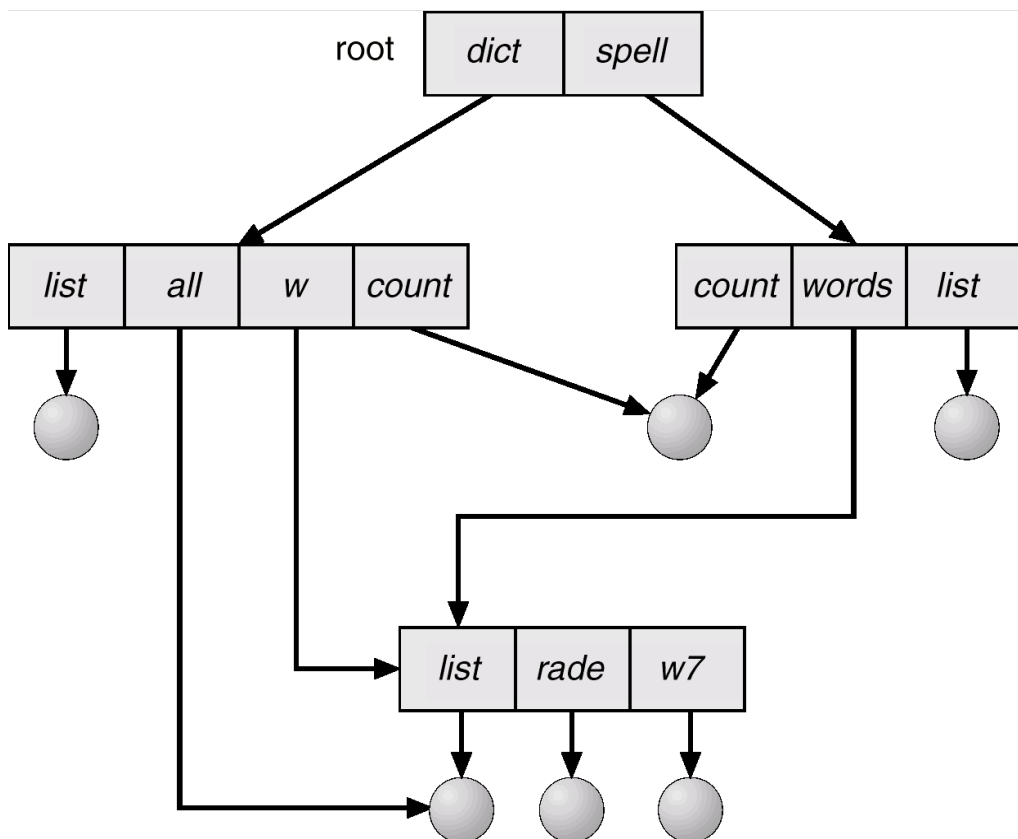
Jak zapewnić acykliczność grafu ?

W Uniksie hard links i symbolic links.

twarde dowiązania to kolejna nazwa tego samego pliku.

symboliczne dowiązania to specjalny plik wskazujący na inny.

Tylko super-użytkownik może utworzyć twarde dowiązanie do katalogu.



Interfejs systemu plików a jego implementacja

Użytkownik, programista aplikacji jest zainteresowany interfejsem do systemu plików udostępnianym przez system operacyjny.

Operacje na plikach i katalogach.

Projektanta systemu interesuje, jak te operacje mogą zostać zaimplementowane.

Jak przechowywać informację o blokach dyskowych zajętych przez plik.

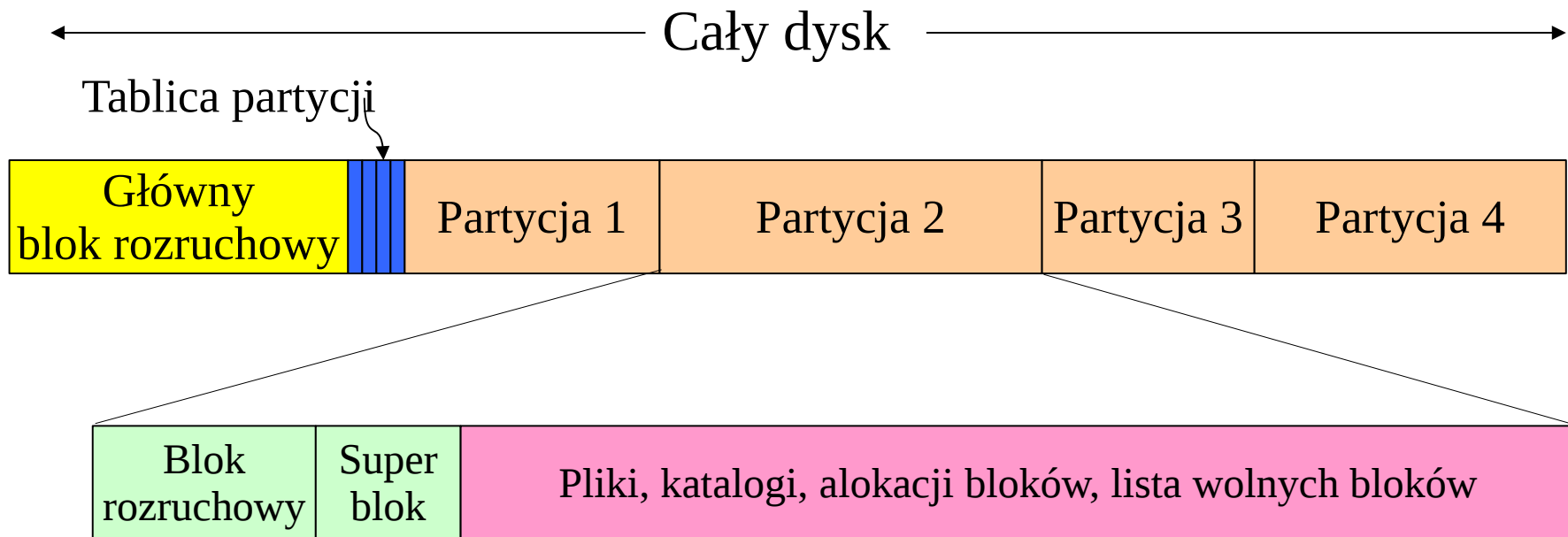
Jak szybko odnaleźć i-ty blok danego pliku (operacja *seek*)

Jak przechowywać informacje o wolnych blokach (blokach nie zajętych przez żaden z plików, ale które mogą być przydzielone w przyszłości)

Gdzie przechowywać informacje o atrybutach pliku

Gdzie przechowywać informacje o strukturze (np. drzewiastej) systemu plików..

Podział dysku na partycje

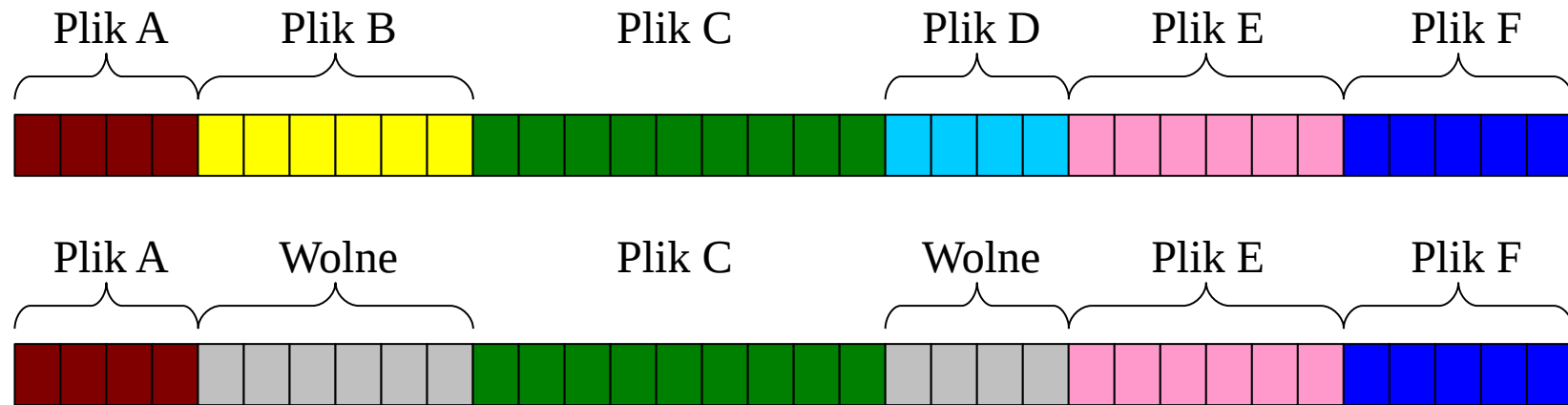


Bloki rozruchowe (boot blocks) zawierają kod ładujący system operacyjny do pamięci.

Każda partycja zawiera odrębny system plików (mogą to być systemy różnych typów). Tablica partycji zawiera informacje o podziale dysku na partycje (początek i koniec)

Super blok zawiera informacje ogólne o systemie (np. całkowita liczba bloków danych, całkowita liczba plików)

Ciągła alokacja bloków danych



Każdy plik zajmuje nieprzerwany ciąg bloków.

Bardzo szybki odczyt.

Bardzo proste zarządzanie informacją o blokach danego pliku – wystarczy tylko pamiętać numer pierwszego bloku oraz liczbę bloku.

Bardzo szybka operacja *seek*.

Wady ciągłej alokacji

Może powstać fragmentacja (podobnie jak w przypadku pamięci RAM) – a kompakcja w pamięci dyskowej jest **bardzo** wolna.

Musimy z góry znać maksymalny rozmiar pliku przy jego tworzeniu – problemy przy zwiększaniu rozmiaru pliku.

Powyższe wady sprawiają że ciągła alokacja jest stosowana w systemach plików tylko do odczytu – np. ISO9600 dla pamięci CD-ROM.

W tym przypadku kompletny system plików, wraz ze wszystkimi plikami jest tworzony przy wypalaniu (wyłaczaniu) płytki.

Raz utworzony system plików nie będzie już modyfikowany (w płytkach wielosesyjnych każda sesja to odrębny system plików).

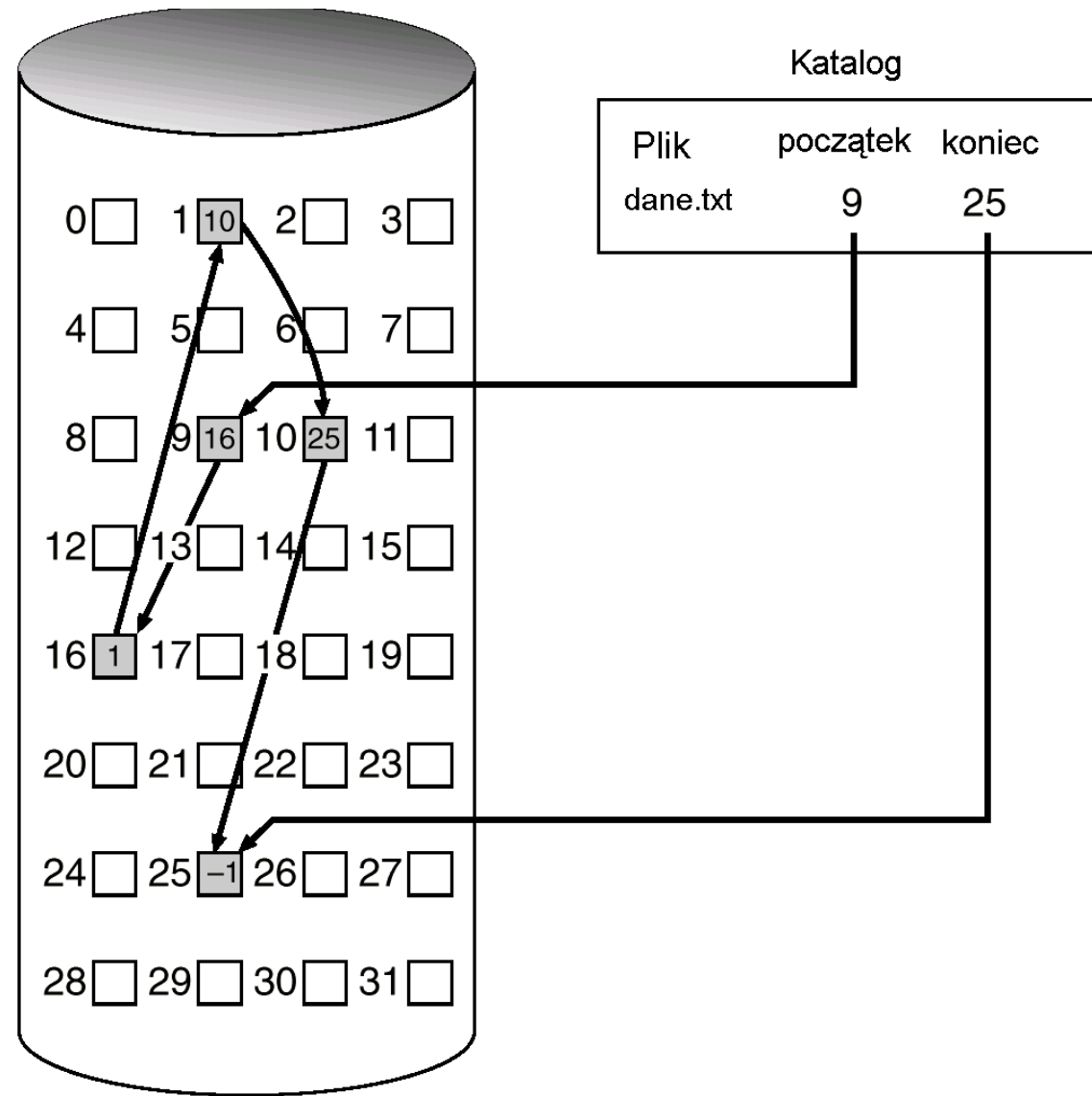
Nie ma potrzeby zarządzania wolnymi blokami, uwzględnienia możliwości zwiększenia rozmiaru plików etc.

Alokacja listowa

Ostatnie (dwa, cztery) bajty bloku danych są zarezerwowane na numer następnego bloku.

W katalogu przechowywany jest numer pierwszego bloku + numer ostatniego (aby umożliwić rozrost pliku).

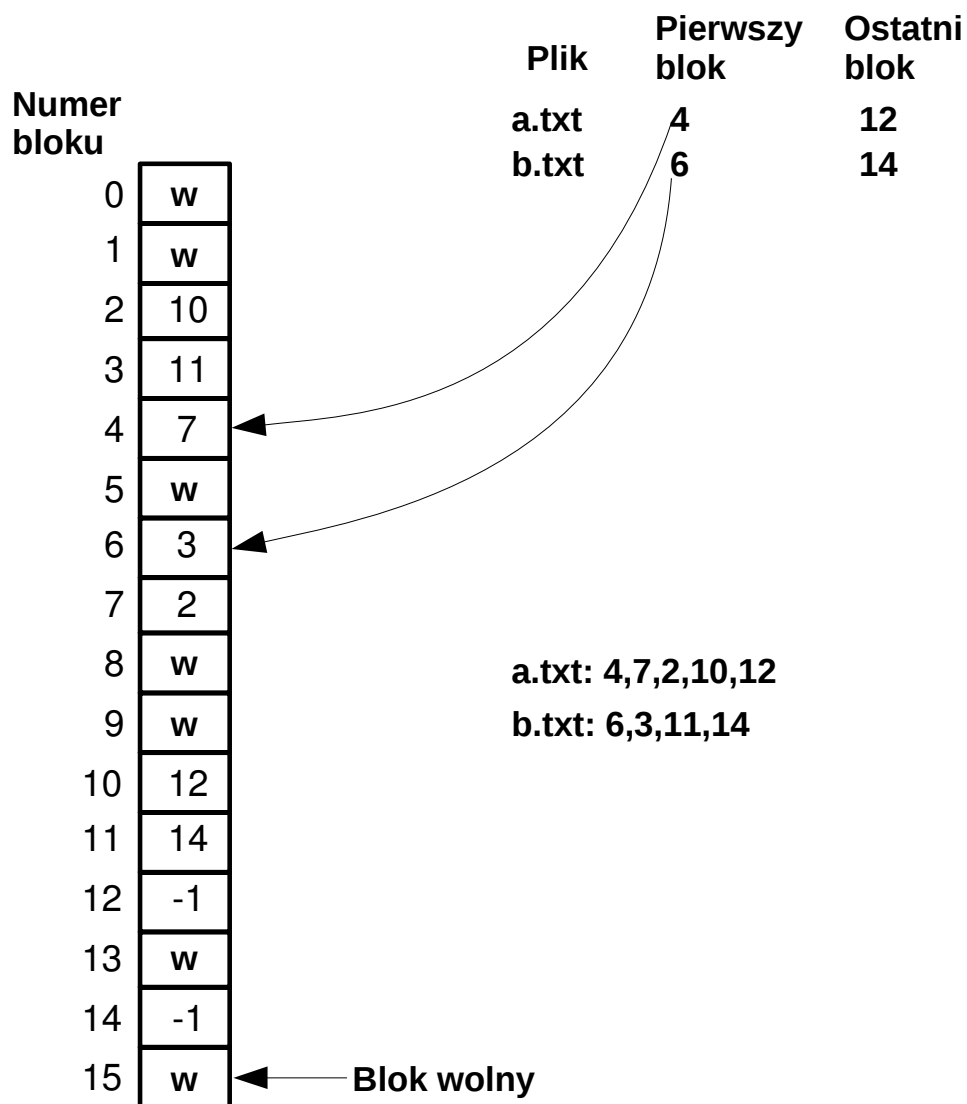
-1 oznacza ostatni blok pliku.



Zaleta: Plikowi możemy przydzielić dowolny blok danych na dysku.

Wada: Nie nadaje się do dostępu swobodnego, ponieważ aby wykonać operację *seek* musimy przeczytać wiele bloków na liście.

Tablica alokacji plików – FAT (ang. File Allocation Table)



Odmiana alokacji listowej, w której numery następnych bloków przechowywane są w odrębnej tablicy (FAT).

Specjalne znaczniki na blok wolny (w) i ostatni blok pliku (-1).

Rozwiązany problem zarządzania wolnymi blokami danych.

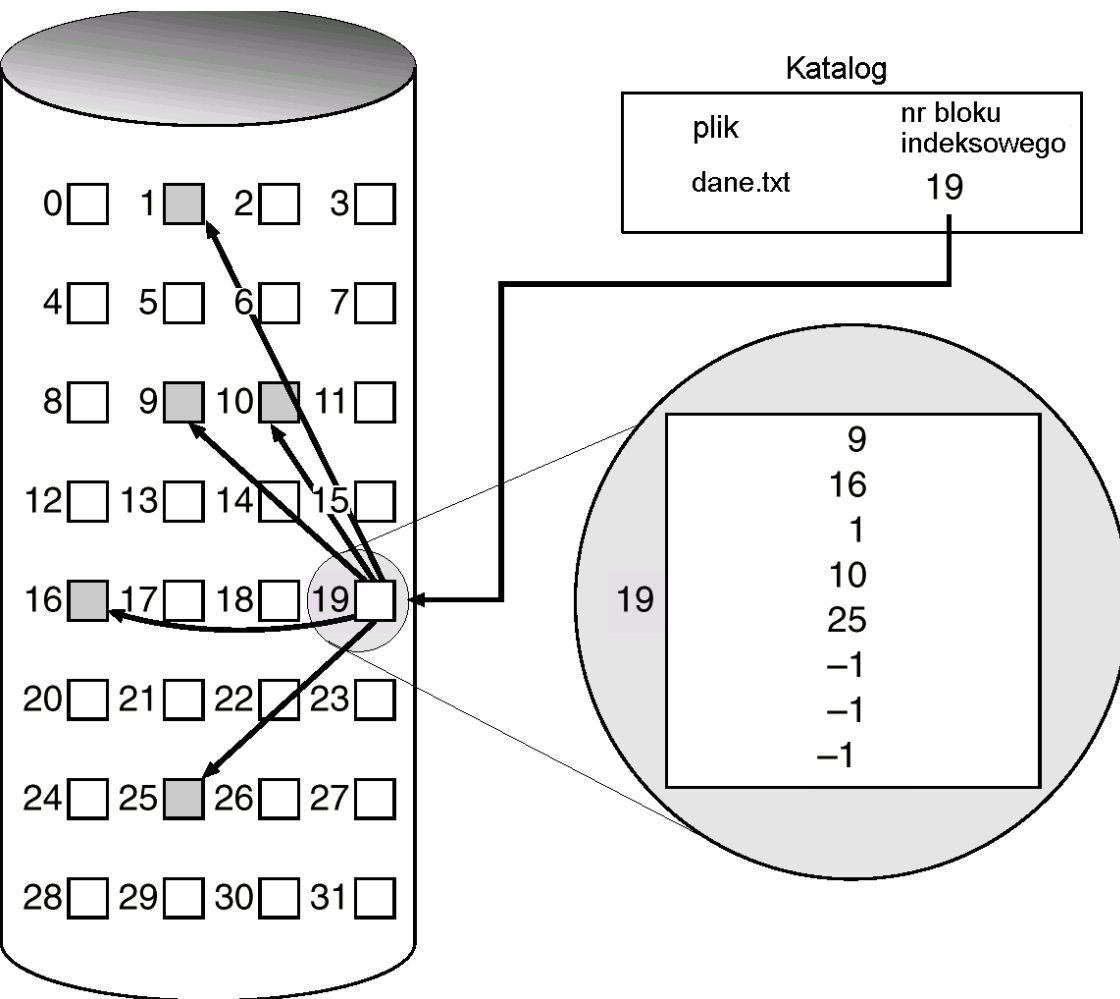
Tablica może znajdować się (w części lub w całości) w pamięci RAM, co zwiększa wydajność, zwłaszcza operacji *seek*.

Problem zapewnienia spójności pomiędzy kopią tablicy w pamięci RAM, a oryginałem w pamięci dyskowej.

Uszkodzenie tablicy FAT może prowadzić do utraty całego systemu plików.

Potencjalnie bardzo duży rozmiar tablicy np. 80MB dla dysku 20GB

Alokacja indeksowa



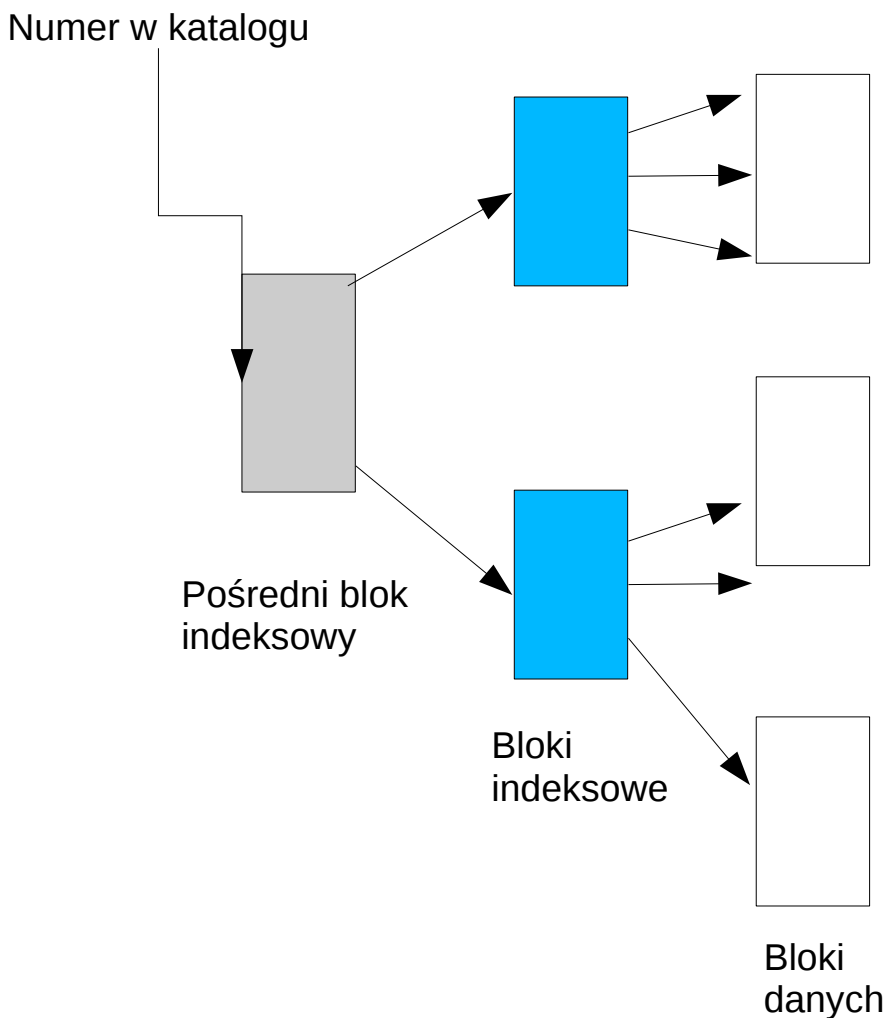
Oddzielny blok indeksowy poświęcony na numery bloków z danymi.

Dobra wydajność.

Zakładając 1KB bloki i 32-bitowe numery w bloku indeksowym zmieści się 256 numerów. Co zrobić gdy rozmiar pliku jest większy od 256 KB ?

Rozwiązanie: Indeksowanie pośrednie

Indeksowanie pośrednie



Katalog zawiera numer bloku pośredniego, który z kolei zawiera numery bloków indeksowych.

Plik może mieć maksymalną długość $256 \times 256 \text{ KB} = 64 \text{ MB}$.

Potrzebujemy indeksowania podwójnie pośredniego (maks 16GB) lub potrójnie pośredniego.

Problem utraty miejsca na bloki indeksowe

Zarządzanie wolnym blokami

W systemie plików, w miarę jak pliki są tworzone i usuwane musimy przydzielać i zwalniać bloki danych (i być może indeksowe).

W systemach FAT wystarczy wykorzystać specjalny znacznik w tablicy

System musi być w stanie szybko odnaleźć wolny blok.

Najprostszy sposób: mapy bitowe

| 0 | 1 | 2 | | | | n-1 | |
|---|---|---|---|---|---|-----|---|
| 1 | 0 | 1 | 0 | 0 | 1 | ... | 1 |

Liczba bitów w mapie jest równa liczbie dynamicznie alokowanych bloków

Pozycja 0 oznacza blok wolny

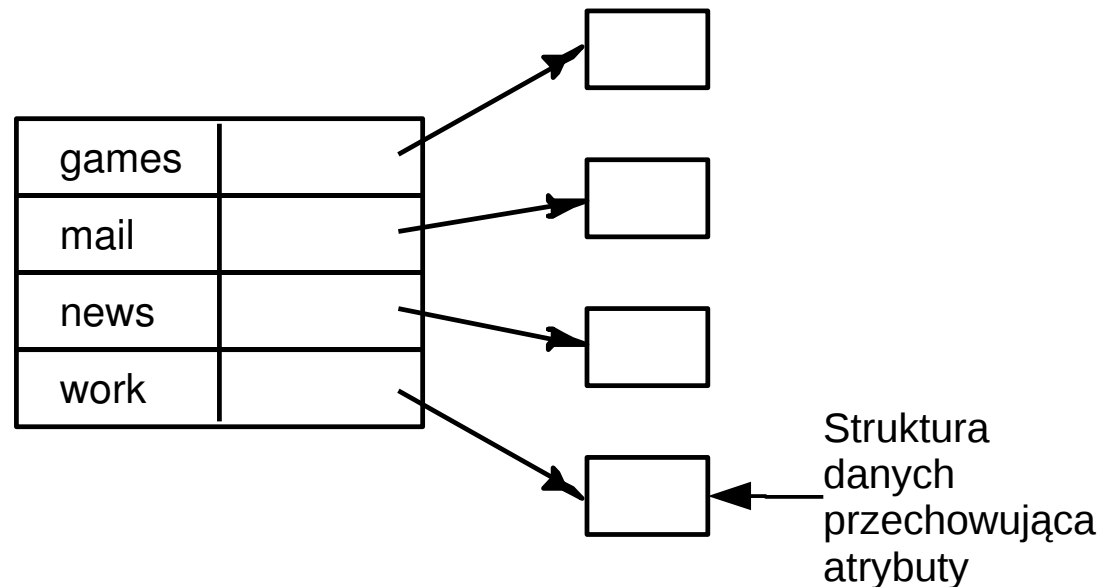
Pozycja 1 oznacza blok wykorzystywany.

Znowu powstaje problem zachowania spójności struktur w pamięci i na dysku

Inne rozwiązanie: Lista wolnych obszarów

Przechowywanie atrybutów pliku

| | |
|-------|----------|
| games | atrybuty |
| mail | atrybuty |
| news | atrybuty |
| work | atrybuty |

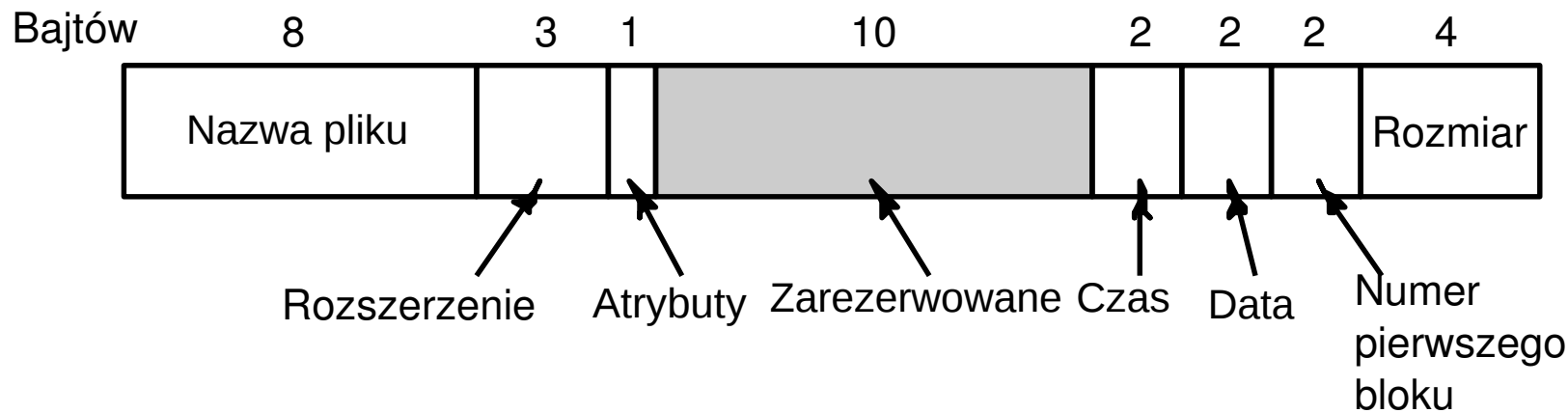


Atrybuty przechowują dane zarówno widoczne dla użytkownika (długość, prawa dostępu, daty) jak i niewidoczne np. numer(y) bloków.

Rozwiązanie z systemu MS-DOS wszystkie atrybuty przechowywane są w katalogu.

Rozwiązanie z systemu UNIX atrybuty przechowywane są w odrębnej strukturze zwanej i-węzłem (ang. i-node) Każdy plik oraz katalog ma swój i-węzeł. Pewna część bloków na dysku jest poświęcona na tablicę i-węzłów. Pozycja katalogu zawiera jedynie numer i-węzła.

Pozycja katalogu w systemie MS-DOS



MSDOS wykorzystuje tablicę FAT. Numer bloku w tablicy może mieć 12 (FAT12) 16 (FAT16) albo 32 (FAT32 Windows95SE) bity.

Pozycja katalogu ma stały rozmiar (32bajty) i przechowuje wszystkie atrybuty pliku.

Katalog jest szczególnym typem pliku.

W kolejnych wersjach wykorzystano zarezerwowane bajty (np. dodatkowe 16-bitów numeru pierwszego bloku w systemie FAT32)

Sektor startowy przechowuje informacje o liczbie kopii i położeniu tablic FAT oraz o położeniu głównego katalogu.

Klasyczny system plików Uniksa

| | | | | | |
|---------------|------------|----------------------|------------------|--------------------|--------------------------|
| Blok startowy | Super blok | Mapa bitowa i-węzłów | Tablica i-węzłów | Mapa bitowa bloków | Bloki danych i indeksowe |
|---------------|------------|----------------------|------------------|--------------------|--------------------------|

Uniks wykorzystuje alokację indeksową (szczegóły za chwilę).

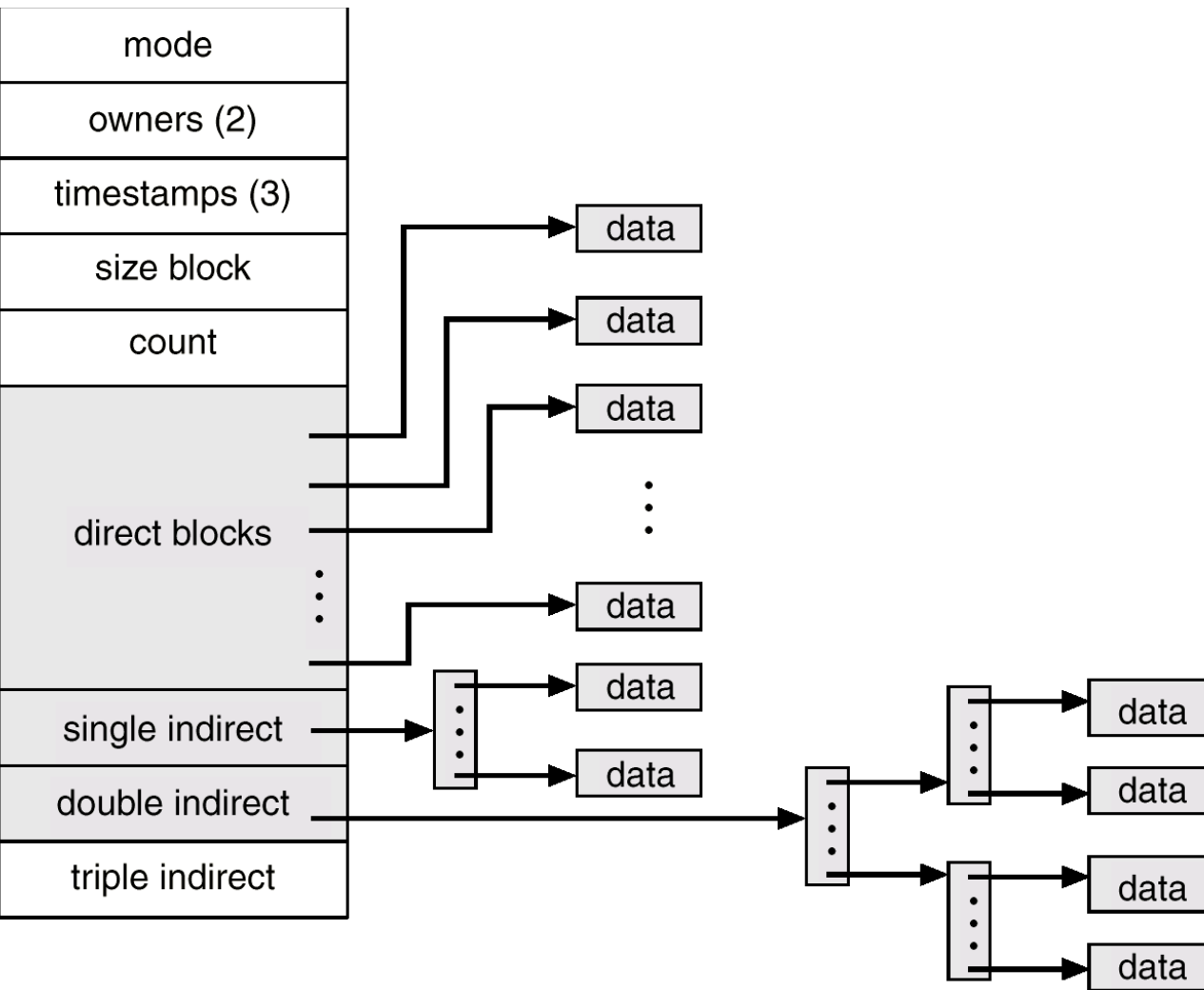
Katalog traktowany jest jako szczególny plik.

Wszelkie atrybuty pliku przechowywane są w i-węźle. I-węzły przechowywane są w tablicy na dysku.

Potrzebujemy odrębnej mapy bitowej do zarządzania wolnym miejscem w tablicy i-węzłów. (pliki są tworzone i usuwane).

Całkowita liczba plików i katalogów w systemie nie może być większa od rozmiaru tablicy i-węzłów.

Postać i-węzła w Uniksie

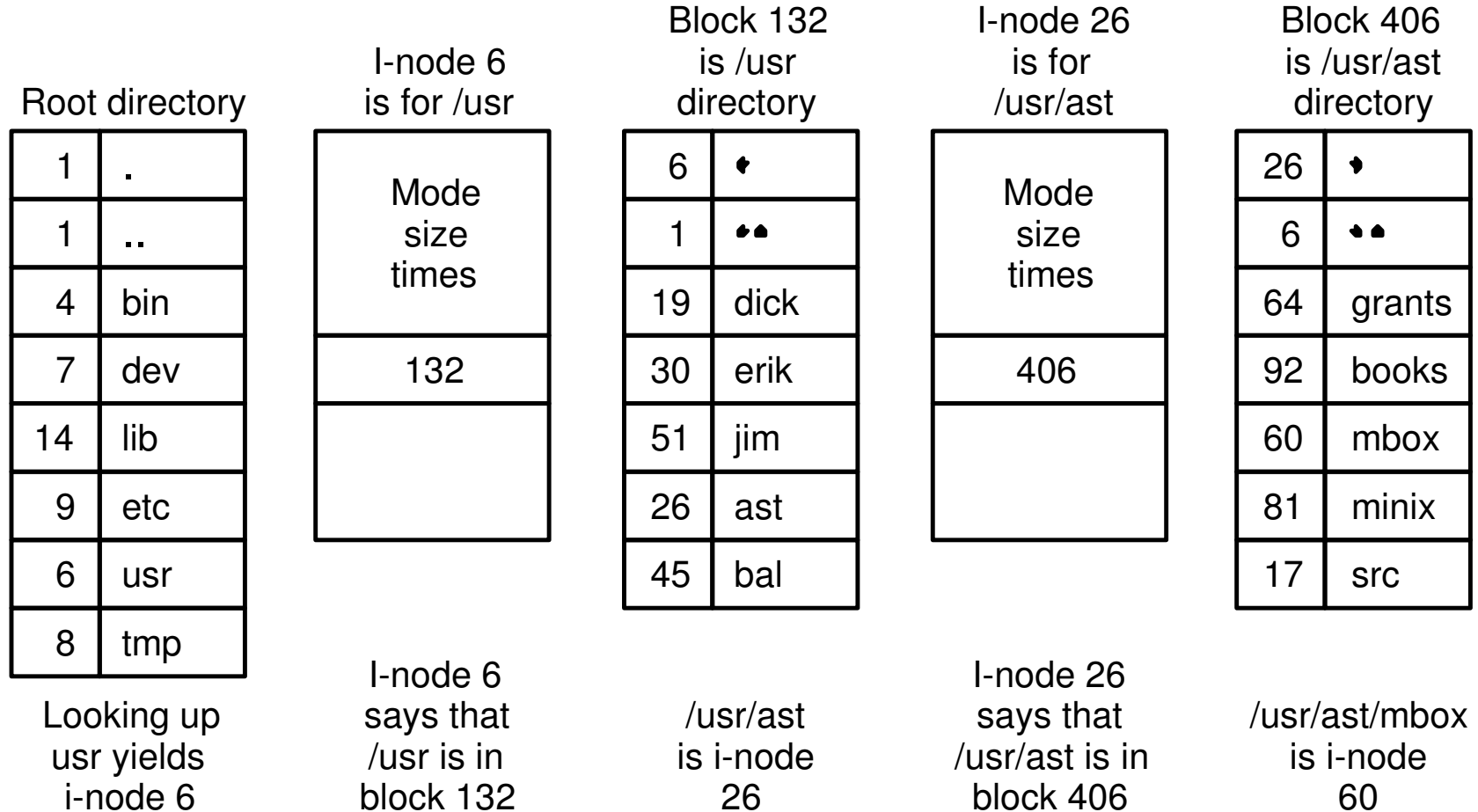


Dla małych plików (np. Do 14 bloków numery są przechowywane bezpośrednio w i-węźle).

Dla większych wykorzystuje się blok indeksowy (single indirect).

Dla jeszcze większych indeksowanie podwójnie i potrójnie pośrednie.

Katalogi w Uniksie



Tłumaczenie nazwy ścieżki /usr/ast/mbox na numer i-węzła

Pozycja katalogu zawiera wyłącznie numer i-węzła

Systemy plików z kroniką (ang. log)

Cel - minimalizacja prawdopodobieństwa katastrofalnej awarii systemu plików, skrócenie czasu naprawy systemu po utracie zasilania. Uwaga: systemy z kronikowaniem nie zastępują UPSa !!!

Specjalny obszar zwany kroniką (ang. journal).

Zmiana w systemie plików przebiega w sposób następujący: (a) zapisz (commit) bloki dyskowe które zmieniasz do kroniki - zapisywane są bloki składające się na **transakcję** (b) jeżeli pierwszy commit powiódł się wykonaj właściwy commit do systemu plików (c) jeżeli drugi zapis się powiódł to możesz (niekoniecznie od razu) usunąć bloki z kroniki (kronika działa jak trochę podobnie jak kolejka FIFO).

Jeżeli awaria nastąpiła przed zakończeniem (a) to nic się nie dzieje - nastąpi utrata danych, ale system plików pozostanie w stanie spójnym.

Jeżeli awaria nastąpi pomiędzy (a) - (b), to przy naprawie systemu plików wykonaj ponowny commit transakcji z kroniki.

Różne możliwości funkcjonowania kroniki: (a) kronikowane są bloki z danymi jak i metadanymi (np. tablica i-węzłów, bitmapy i-węzłów i bloków, bloki indeksowe), (b) wyłącznie bloki z metadanymi, po awarii zasilania system plików zachowa spójność ale jest duża szansa na utratę danych plików zapisywanych w momencie awarii.

Uwaga: gwarancja pozostawienie systemu plików w stanie spójnym nie oznacza gwarancji że dane przechowywane w pliku będą spójne z punktu widzenia aplikacji.

Strategie optymalizacji wydajności

Generalnie powinniśmy dążyć do tego aby kolejne bloki danych tego samego pliku zostały umieszczone w kolejnych blokach na dysku.

Programy typu Defrag – mogą być bardzo czasochłonne.

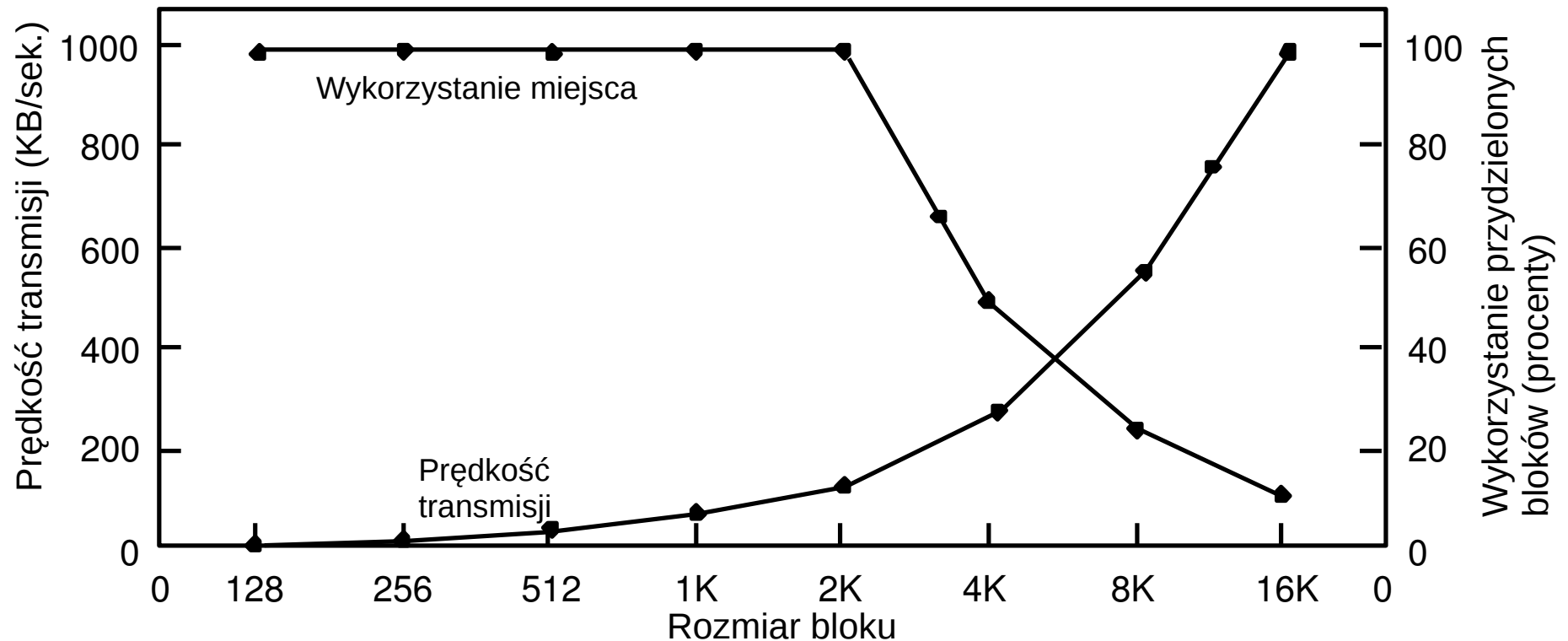
Inne strategie

Prealokacja bloków: jeżeli przy zapisie do pliku potrzebujesz jednego nowego bloku, to przydziel od razu k (typowo $k=8,16$) bloków zajmujących ciągły obszar dysku, w nadziei, że za chwilę będą potrzebne nowe bloki. Jeżeli dodatkowe bloki nie zostaną wykorzystane, to są zwalniane przy zamykaniu pliku.

Wykorzystaj pamięć podręczną

Zwiększ rozmiar bloku

Wybór rozmiaru bloku (Tannenbaum, 2004)



Pamięć masowa jest podzielona na 512-bajtowe sektory (2048 bajtów w CD-ROMie). Blok dyskowy jest wielokrotnością sektora. Typowe rozmiary bloku to 1,2,4,8 sektortów.

Większy rozmiar bloku => większa prędkość transmisji (mniejsza fragmentacja, czyli mniej ruchów głowicą).

Większy rozmiar bloku => mniej efektywne wykorzystanie miejsca na dysku. Przeciętnie z każdym plikiem wiąże się utrata miejsca równa połowie długości bloku

Pamięć podręczna (ang. cache) dysku

Idea: Przeznaczyć część pamięci RAM na przechowywanie najczęściej używanych bloków w celu poprawienia wydajności systemu.

Musimy dostarczyć mechanizm (np. tablice mieszające) pozwalający szybko sprawdzić czy blok o numerze i jest w pamięci podręcznej.

Jeżeli bloku nie ma to jest on wczytywany z dysku i dodawany do pamięci podręcznej.

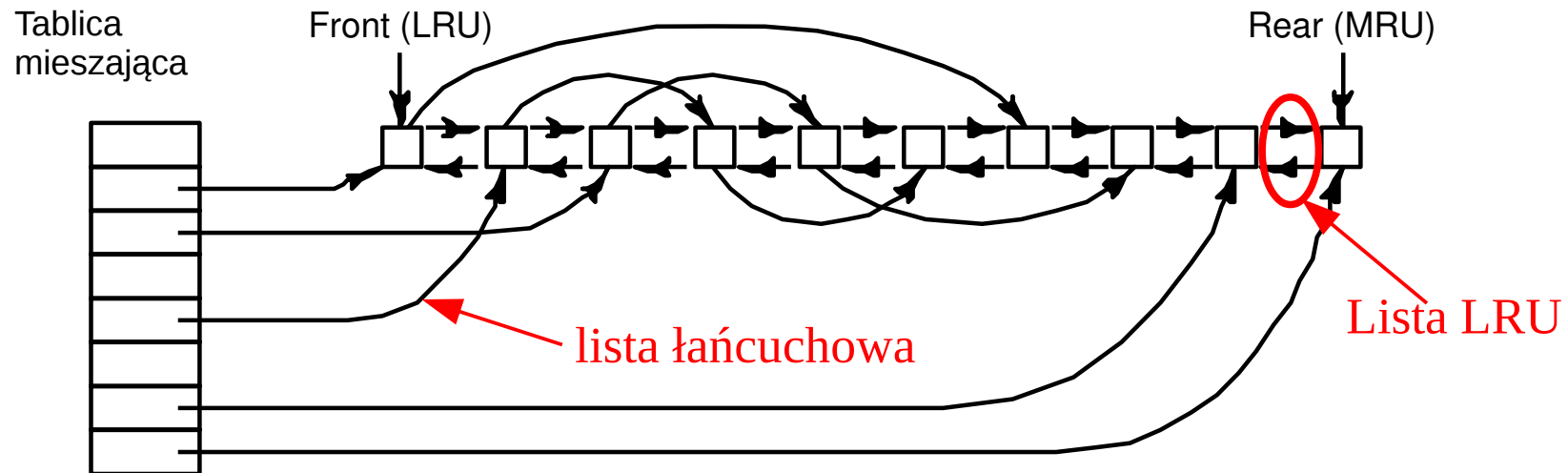
Musimy określić algorytm wyboru bloku usuwanego z pamięci podręcznej (np. LRU).

Musimy określić strategię obsługi zapisów:

Write-through – zapisywany blok trafia jednocześnie do pamięci podręcznej i na dysk.

Blok jest zapisywany z opóźnieniem – większa wydajność, ale większe niebezpieczeństwo uszkodzenia systemu. Większość systemów wprowadza maksymalne opóźnienie np. 20 s dla bloków danych zwykłych plików (nie katalogów) i 2 sekundy dla pozostałych bloków

Przykład implementacji pamięci podręcznej



Tablica mieszająca: pozwala na szybkie sprawdzenie, czy blok jest w pamięci. Każdy element odpowiada jednej wartości funkcji mieszającej.

Każdy blok jest elementem dwóch list: dwukierunkowej listy LRU, oraz listy łańcuchowej na której przechowywane są bloki o identycznej wartości funkcji mieszającej.

W przypadku trafienia (ang. cache hit) blok pobrany z pamięci jest przesuwany na koniec listy LRU .

Nowy blok jest wczytywany w miejsce bloku na pierwszej pozycji listy i przesuwany na koniec.