Organizacja systemu plików

- organizacja logiczna pliku: rekordy o stałej lub zmiennej długości

  np. w systemie Unix typowo pliki zorganizowane są jako sekwencje bajtów, zatem są to rekordy o stałej długości jednego bajta

- organizacja fizyczna pliku: zbiór bloków dyskowych — prowadzi do zjawiska fragmentacji wewnętrznej

  kompromis: wielkość bloku/efektywność oraz fragmentacja

- operacje na plikach: tworzenie, kasowanie, otwieranie, zamykanie, odczyt sekwencyjny, zapis sekwencyjny, przesuwanie kursora

- zarządzanie plikami: tworzenie i odczytywanie atrybutów, kontrolowanie praw dostępu

- organizacja wielu plików na jednym urządzeniu: katalogi plików

- operacje na katalogach: tworzenie, przeszukiwanie, sortowanie

- operacje na systemach plików: tworzenie, sprawdzanie spójności

Operacje na plikach

Podstawowe:

- tworzenie
- zapis
- odczyt
- pozycjonowanie (*seek*), inaczej: przesuwanie kursora
- usuwanie pliku
- zerowanie pliku (zapis od zera)

Dodatkowe:

- dopisywanie na końcu pliku
- zmiana nazwy pliku

Zauważmy, że np. kopiowanie pliku można zrealizować przez utworzenie nowego i następnie sekwencję odczytów i zapisów. Jednak nie chcielibyśmy np. zmieniać nazwy pliku przez utworzenie nowego pod nową nazwą i przepisanie zawartości.

Atrybuty plików

- zabezpieczenia - kto może uzyskać dostęp i w jaki sposób
- hasło
- twórca
- właściciel
- flaga tylko do odczytu
- flaga ukrycia - pliki nie wyświetlane w normalnym listingu
- flaga systemowa - pliki systemowe
- flaga archiwum - pliki wymagające archiwizacji
- flaga tekstowy/binarny
- flaga dostępu losowego/tylko sekwencyjnego
- flaga pliku tymczasowego
- flaga blokady
- długość rekordu - liczba bajtów w rekordzie
- pozycja klucza - offset klucza w rekordzie
- długość klucza
- czas utworzenia
- czas ostatniej modyfikacji
- czas ostatniego dostępu
- bieżący rozmiar - liczba bajtów w pliku
- maksymalny rozmiar - liczba bajtów do jakiej plik może się rozrosnąć

Typy plików

System operacyjny może obsługiwać pliki różnych **typów**. Typami plików mogą być: plik tekstowy, plik binarny, plik archiwizacji, plik skompresowany, plik zaszyfrowany, itp. Z każdym z tych typów plików związane są pewne operacje charakterystyczne, na przykład linkowanie i ładowanie do pamięci pliku binarnego, wyświetlanie zawartości pliku archiwizacji, itp.

Ponadto, z różnymi typami mogą być związane pewne charakterystyczne atrybuty, np. *encoding* dla plików tekstowych, algorytm kompresji dla plików skompresowanych, albo identyfikator klucza dla plików zaszyfrowanych.

System operacyjny może znać różne typy plików, i posiadać procedury niezbędne do ich obsługi. Alternatywnie, system może traktować różne typy plików identycznie, zapewniając tylko podstawowe procedury ich obsługi. W takim przypadku obowiązkiem użytkownika pozostaje uruchamianie aplikacji niezbędnych dla posługiwania się poszczególnymi plikami.

Struktury plików

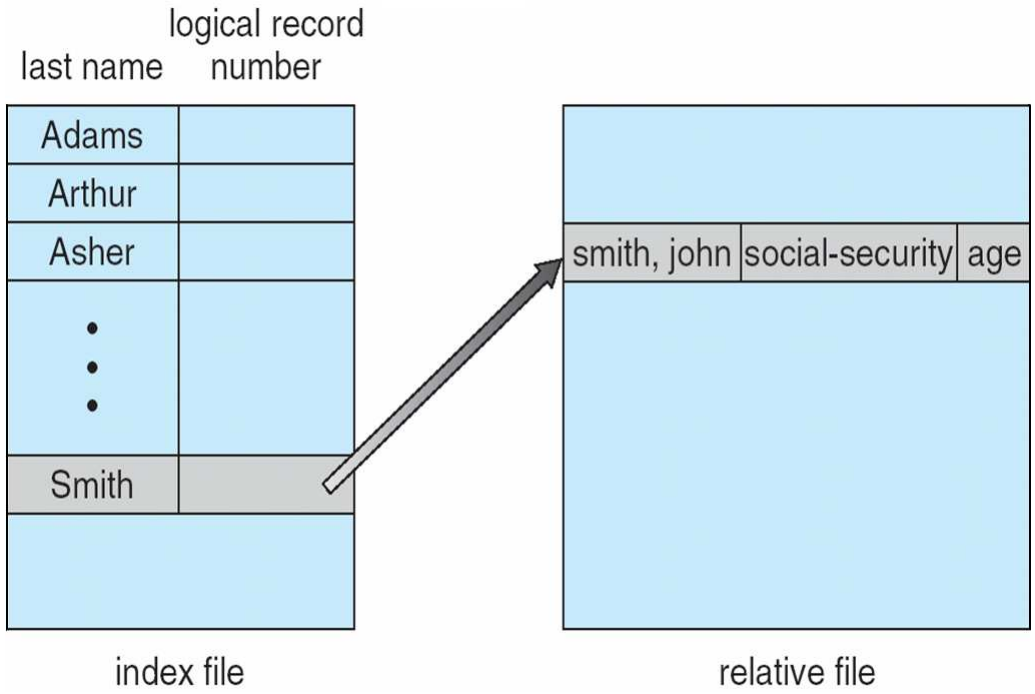
Z typem pliku związana jest często **struktura**, albo **inaczej organizacja logiczna**. Przykładem prostej struktury pliku jest **ciąg bajtów**. Taką strukturę obsługują takie systemy jak Unix albo DOS. W ramach tej struktury obsługiwane są wszystkie pliki w systemie, np. pliki tekstowe, pliki binarne zawierające program binarny skompilowany na maszynę 64-bitową, pliki binarne zawierające obraz cyfrowy w formacie JPEG w kodowaniu 24-bitowym, itp.

Przykładem innej struktury jest **ciąg rekordów** o stałej lub zmiennej długości. W tym przypadku elementarne operacje wejścia/wyjścia na pliku zapisują lub odczytują pojedynczy rekord o odpowiedniej długości, lub jego wielokrotność. Przykładowo, rekord może mieć stałą długość 36 bitów, i posługiwanie się takimi plikami miałoby sens dla komputera o długości słowa maszynowego również 36-bitów, ponieważ naturalną jednostką transferu byłoby wtedy 36 bitów.

(Trochę kłopotliwe byłoby przechowywanie w takich plikach danych tekstowych zakodowanych w 7-bitowym kodzie ASCII, aczkolwiek możnaby w pojedynczym słowie przechować dokładnie 5 znaków tekstowych, ze stratą tylko jednego bitu na słowo.)

Pliki indeksowane

Przykładem bardziej złożonej struktury może być **struktura indeksowana**.



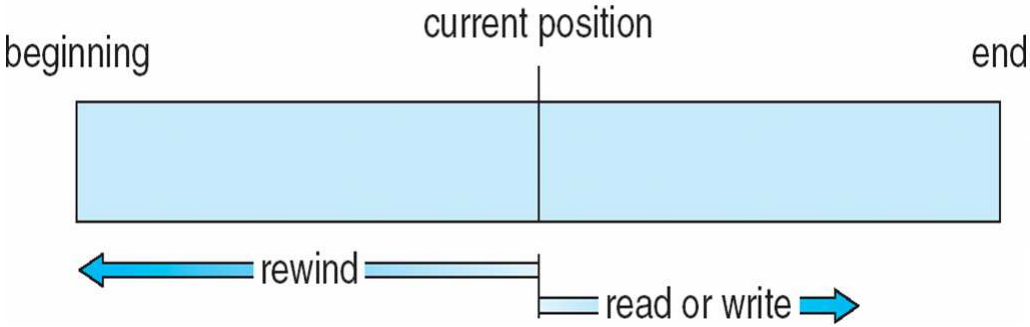
Pliki indeksowane składają się z dwóch części, z których jedna zawiera indeks, a druga właściwe dane. Takie pliki można traktować jako rodzaj prostej bazy danych, gdzie dane posiadają klucze według których są zapamiętywane i wyszukiwane.

Dostęp sekwencyjny

Kolejnym aspektem obsługi plików przez system jest sposób dostępu. Podstawowymi sposobami dostępu są:

- dostęp sekwencyjny,
- dostęp bezpośredni,
- dostęp indeksowany.

Dostęp sekwencyjny:



Terminologia stosowana w odniesieniu do dostępu sekwencyjnego przywodzi skojarzenia z taśmą magnetyczną (np. przewijanie do początku). Rzeczywiście, istniały implementacje systemów plików zapisane na taśmach magnetycznych, i dostęp sekwencyjny jak najbardziej ma sens na takich nośnikach.

Dostęp bezpośredni

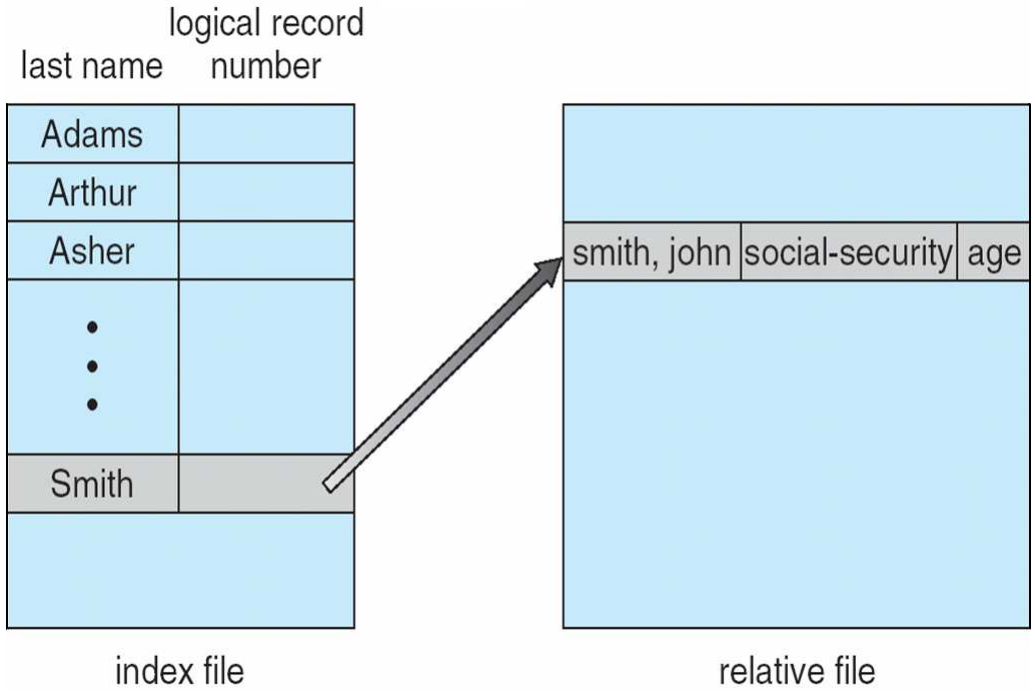
Dla plików przechowywanych na dyskach zwykle możliwe jest przejście do dowolnej pozycji (bloku) pliku bez sekwencyjnego czytania wszystkich bloków poprzedzających. Ten sposób dostępu nazywany jest **dostępem bezpośrednim**, a operację dostępu do wybranej sekcji pliku w tym trybie nazywa się **pozycjonowaniem** (*seek*), albo czasami **przesuwaniem kursora**.

Możliwość dostępu bezpośredniego i wykonalność operacji pozycjonowania wynika z implementacji plików na dyskach magnetycznych (oraz SSD), która może umożliwiać obliczenie adresu bloku dyskowego zawierającego poszukiwaną daną, i wykonanie operacji dokładnie na tym bloku.

Jednak jak wkrótce zobaczymy, w zależności od sposobu implementacji systemu plików, nawet na urządzeniach dyskowych, dostęp bezpośredni może nie być możliwy do realizacji.

Dostęp indeksowany

Pliki indeksowane zapewniają szybki dostęp do rekordów uporządkowanych według klucza, lecz wymagają dwustopniowego dostępu. Alternatywą jest wczytanie i trzymanie indeksu w pamięci, jednak jest to sensowne tylko do pewnej wielkości. Możliwe jest również stosowanie trzystopniowej struktury, gdzie indeks główny trzymany jest w pamięci RAM, natomiast indeks drugiego poziomu jest już przechowywany na dysku.



Krótkie podsumowanie — pytania sprawdzające

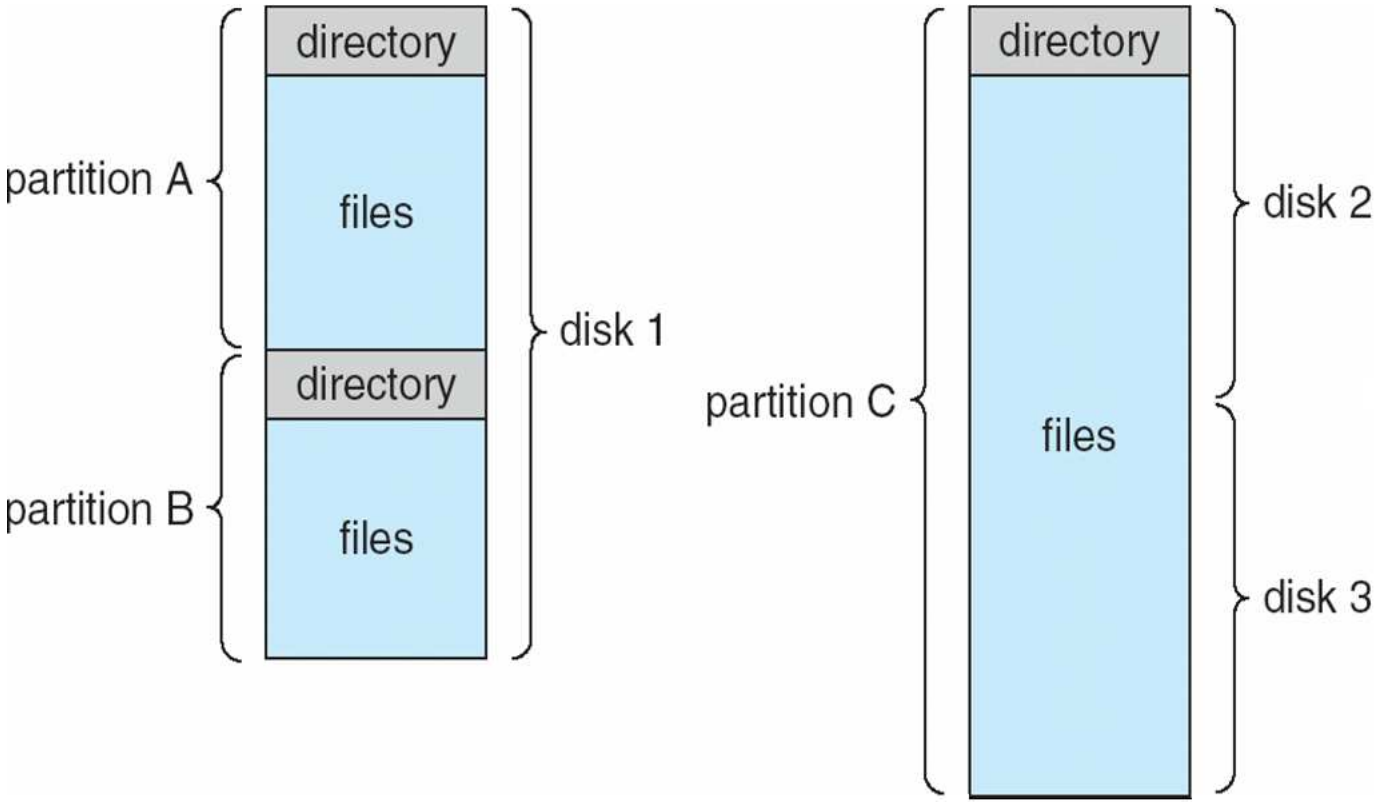
1. Jakie są podstawowe operacje wykonywane na plikach?

2. Co to jest struktura pliku i jakie są możliwe struktury?

3. Czym różni się dostęp sekwencyjny do pliku od dostępu bezpośredniego?

4. Co to jest operacja pozycjonowania (*seek*)?

Organizacja logiczna systemu plików: partycje



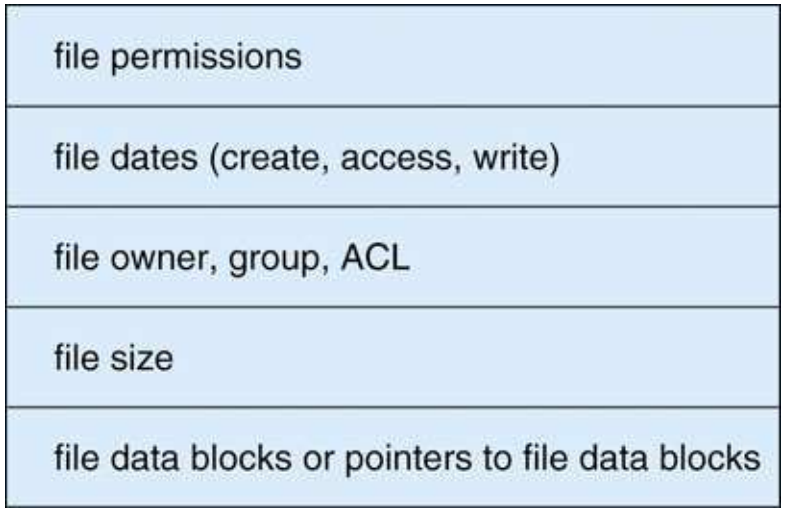
Partycjonowanie dysków umożliwia łatwiejszą organizację rozmieszczenia różnych elementów systemu na dysku. W ramach partycji mogą być przechowywane systemy plików, które składają się z katalogów oraz poszczególnych plików. Partycja może również być wykorzystywana w inny sposób, na przykład jako obszar wymiany systemu pamięci wirtualnej *swap*. Partycja może obejmować część jednego dysku, cały dysk, lub wiele dysków.

Organizacja logiczna systemu plików: katalogi

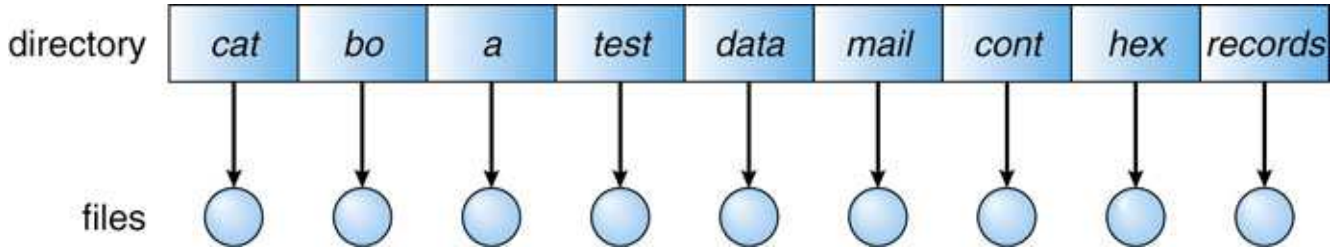
Systemy operacyjne posługują się wieloma plikami, zarówno dla organizacji samego systemu, jak i w celu udostępnienia operacji programom. Systemy plików mogą mieć różną organizację, ale dla umożliwienia tworzenia, przechowywania, i dostępu do wielu plików posiadają na ogół strukturę zwaną **katalogiem**.

Katalog w ogólności jest listą plików zawartych w systemie, z ich nazwami i informacją o alokacji pliku na dysku, jak adres **bloku kontrolnego pliku FCB** (*File Control Block*).

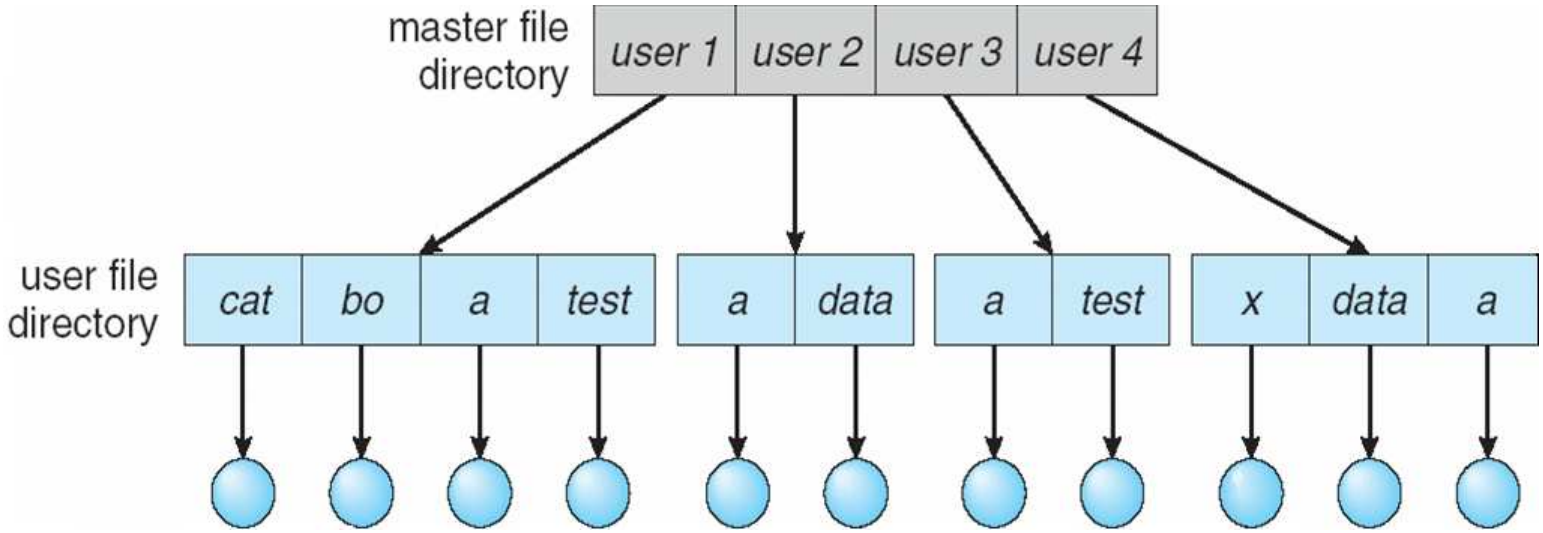
Blok kontrolny pliku w systemie Unix, tzw. *i-node*:



Katalogi jednopoziomowe

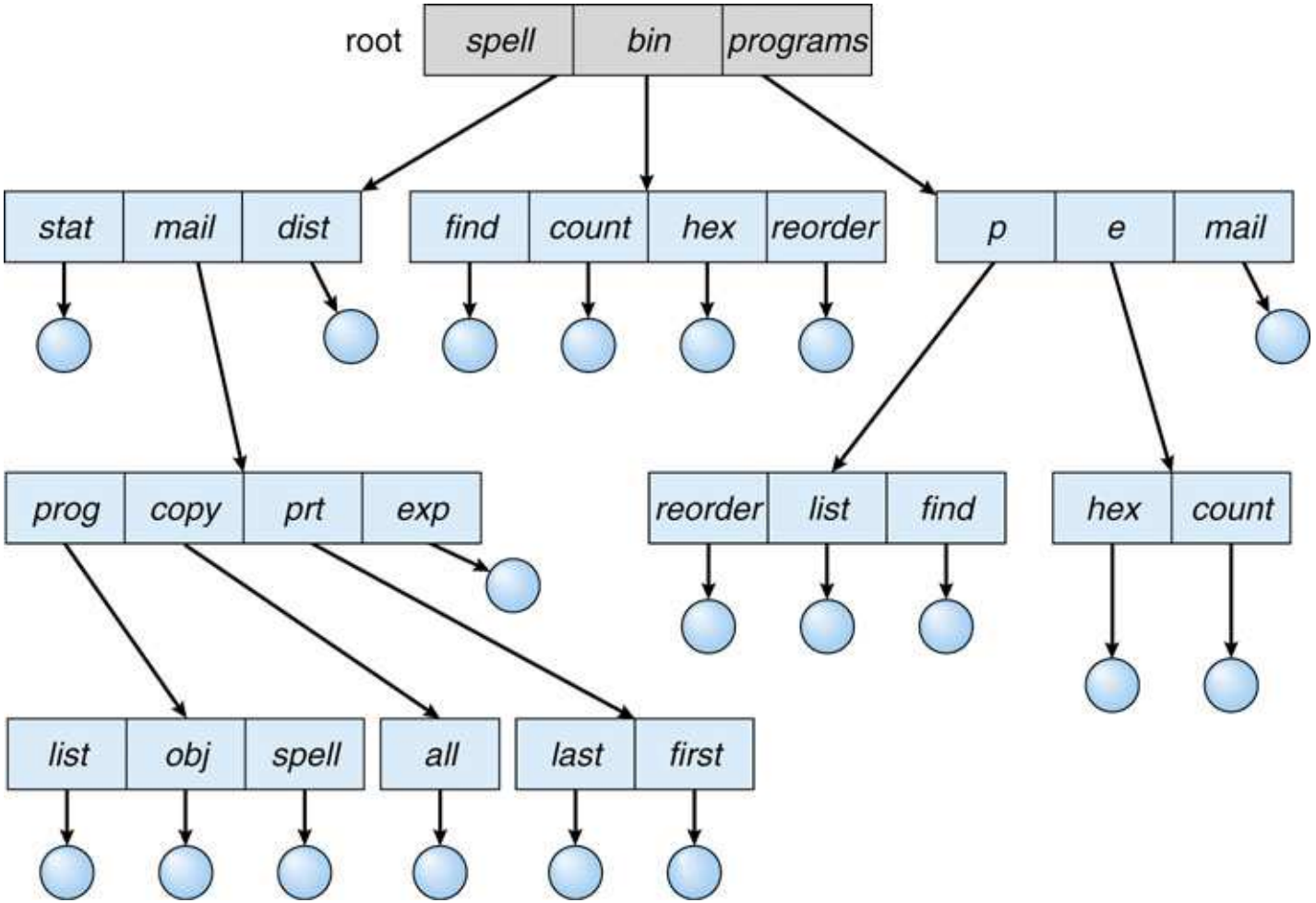


Katalogi dwupoziomowe



Przewaga katalogów dwupoziomowych nad jednopoziomowymi: umożliwiają odseparowanie przestrzeni nazw dla różnych projektów, programistów, wersji, itp.

Katalogi wielopoziomowe



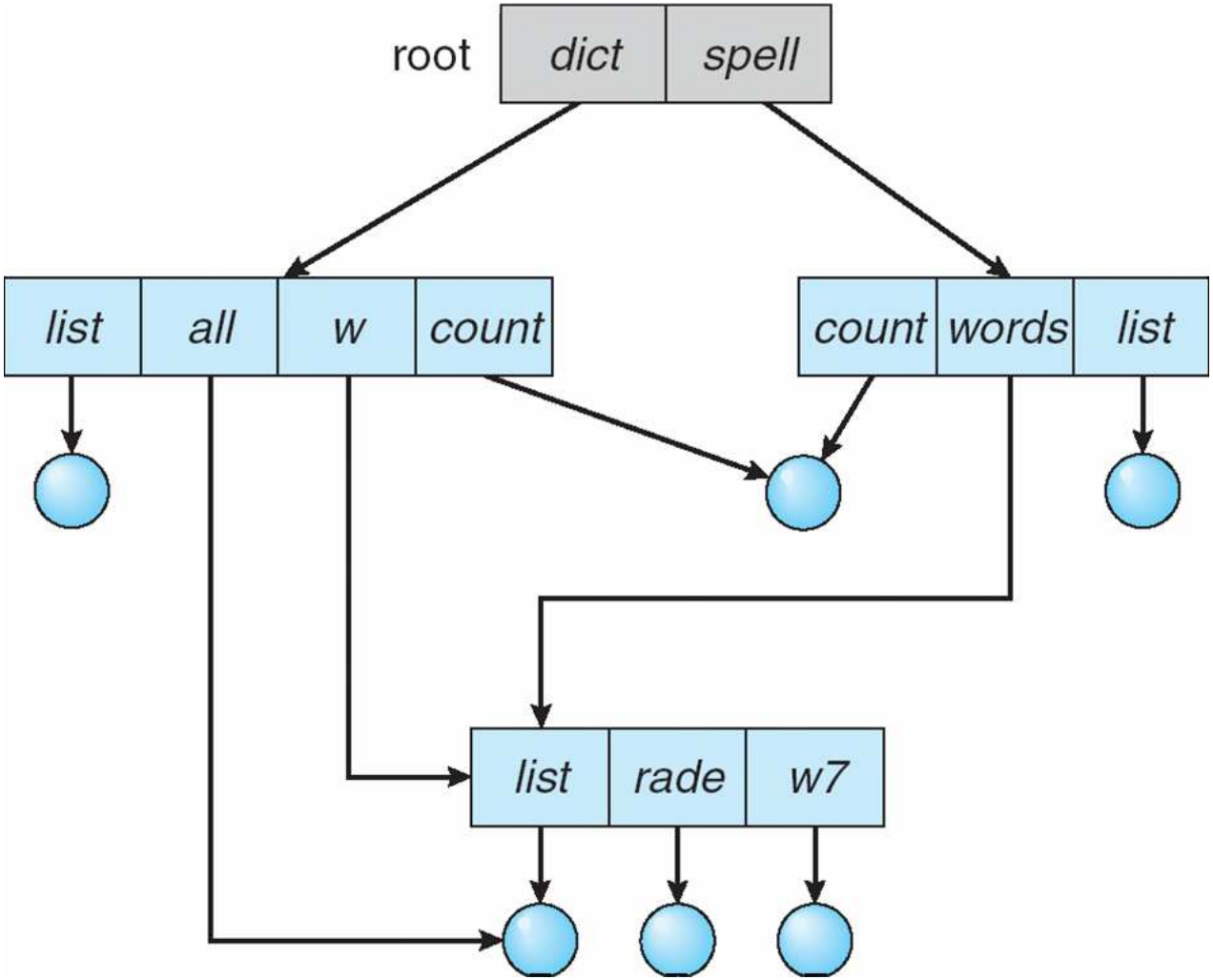
Linki w systemie plików — drzewiasta struktura katalogów

Podstawową strukturą katalogów jest hierarchiczna struktura drzewiasta, z katalogiem głównym stanowiącym korzeń struktury, i katalogami zawierającymi na kolejnych poziomach — obok plików — podkatalogi.

Pozycja w katalogu reprezentująca plik lub podkatalog zawiera nazwę danego obiektu i wskaźnik do niego, zwany **dowiązaniem** lub *link*iem. Tworzenie nowego pliku lub katalogu związane jest z utworzeniem nowego linku w odpowiednim katalogu.

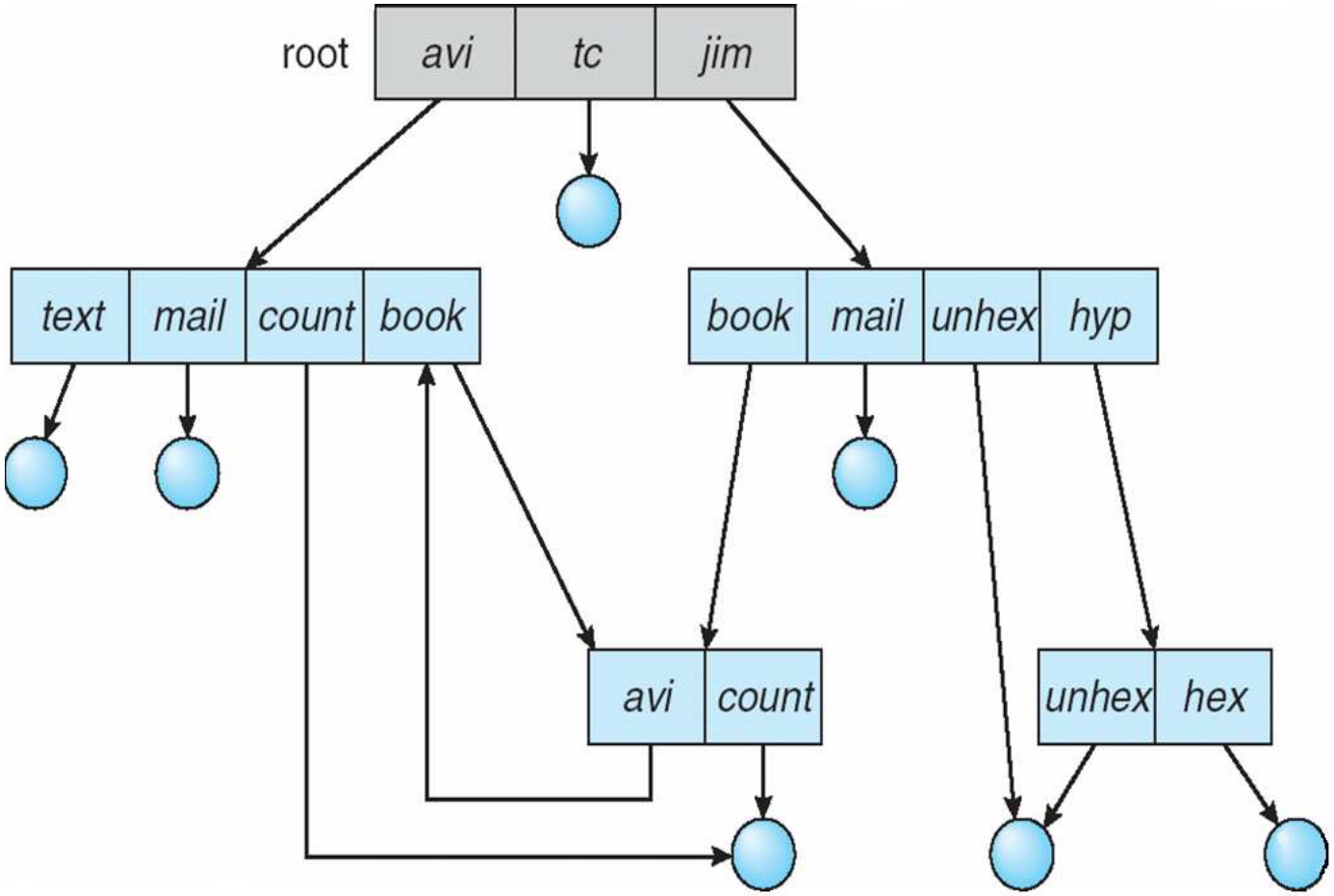
Możliwe jest również tworzenie dodatkowych — nadmiarowych — linków do już istniejących plików oraz katalogów.

Acykliczna struktura katalogów



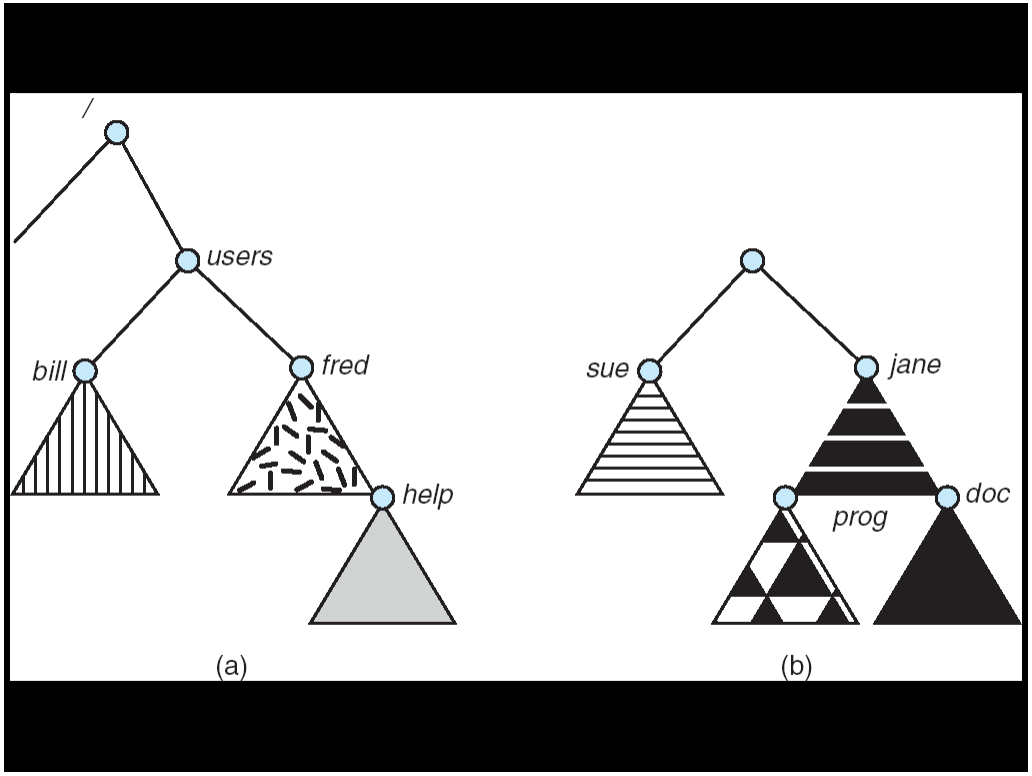
Acykliczna struktura katalogów pozwala na większą ogólność niż struktura ściśle drzewiasta, np. na istnienie jednego pliku w różnych katalogach. Pozwala to na unikanie kopiowania, ale wymaga specjalnych procedur np. przy usuwaniu plików.

Ogólna struktura katalogów



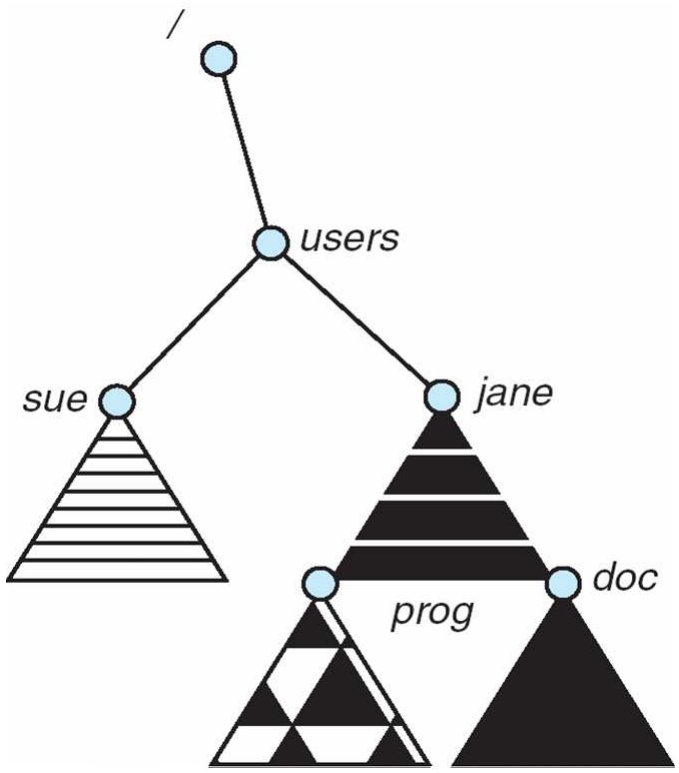
Ogólna struktura katalogów może zawierać cykle między katalogami. Procedury, które zagłębiają się rekurencyjnie w strukturze katalogów mogą w ten sposób wpadać w nieskończone pętle.

Niezależne systemy plików



Niektóre systemy dopuszczają operacje na wielu systemach plików jednocześnie. Na przykład, dyski C:, D:, itd., w systemie Windows.

Zamontowane systemy plików



Unix (i inne podobne systemy) operują tylko na jednym logicznym systemie plików, i wymagają, by wszystkie używane fizyczne systemy plików były włączone w jeden system logiczny. W systemie Unix jest to realizowane przez operację **montowania** jednego systemu plików w jakimś miejscu drzewa katalogów drugiego.

Jeśli montowany system plików zostanie włączony do niepustego katalogu, to tymczasowo przesłania część podstawowej struktury katalogów.

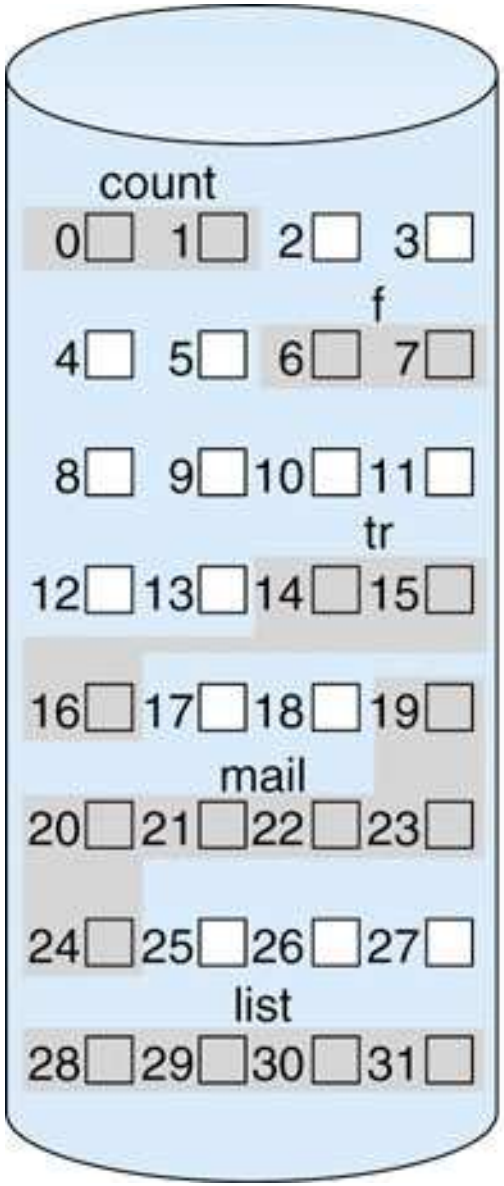
Krótkie podsumowanie — pytania sprawdzające

1. Jaką strukturę katalogów implementuje system Unix: drzewiastą, acykliczną, czy ogólną?

2. Jaką strukturę katalogów implementuje system Windows: drzewiastą, acykliczną, czy ogólną?

3. Czym różni się acykliczna od ogólnej struktury katalogów?

Metody alokacji: ciągła alokacja bloków

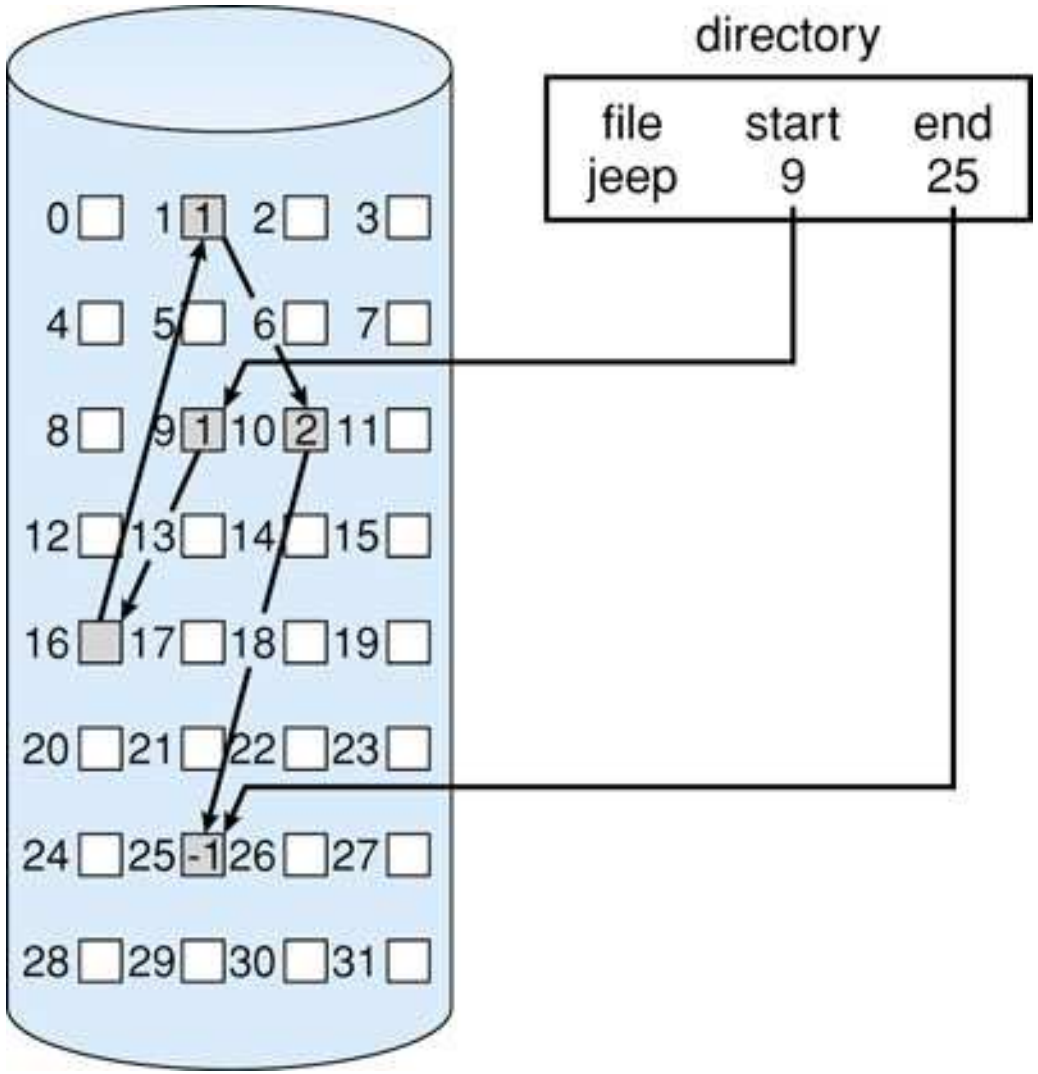


directory

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

Uwaga na fragmentację zewnętrzną.

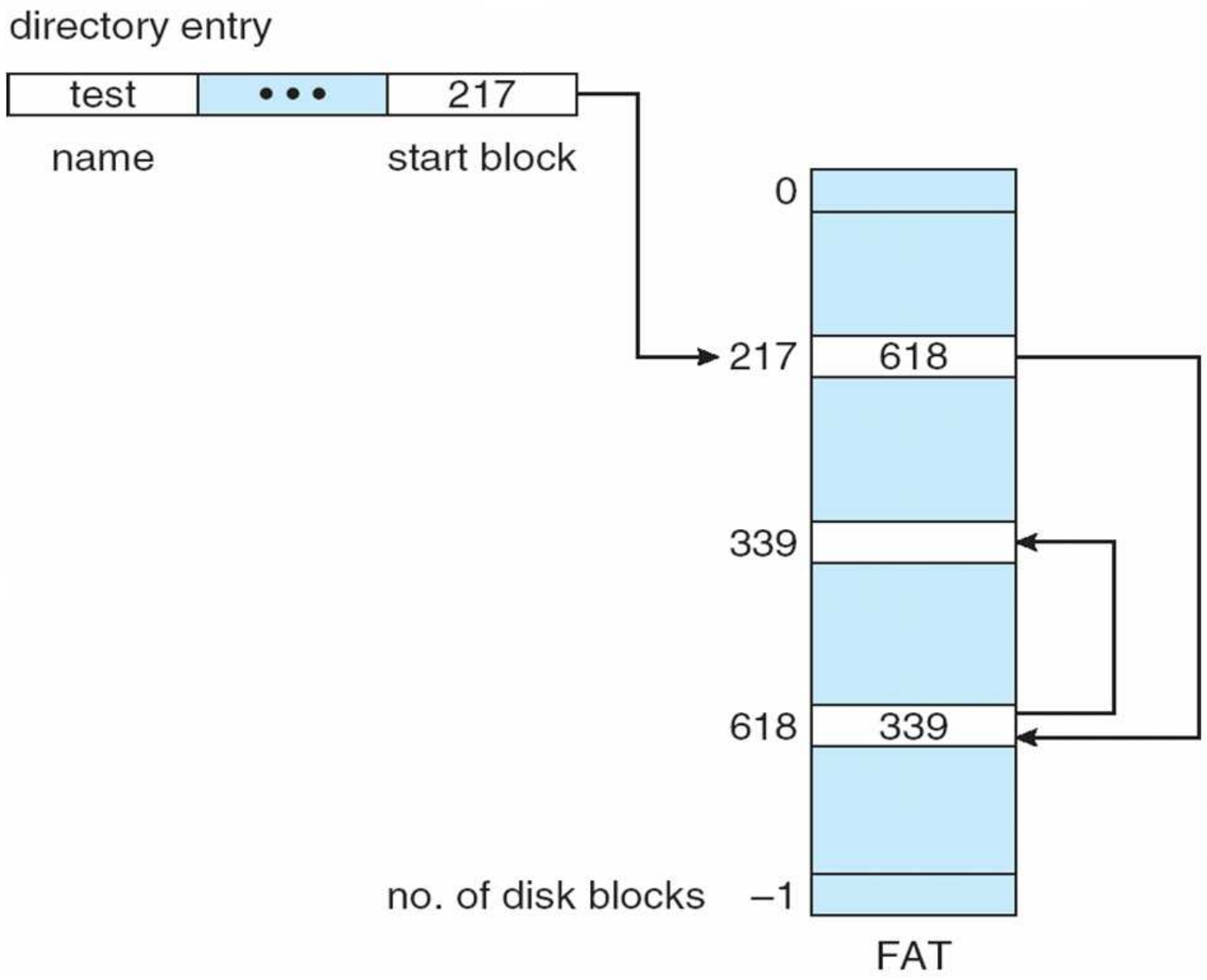
Metody alokacji: listowa alokacja bloków



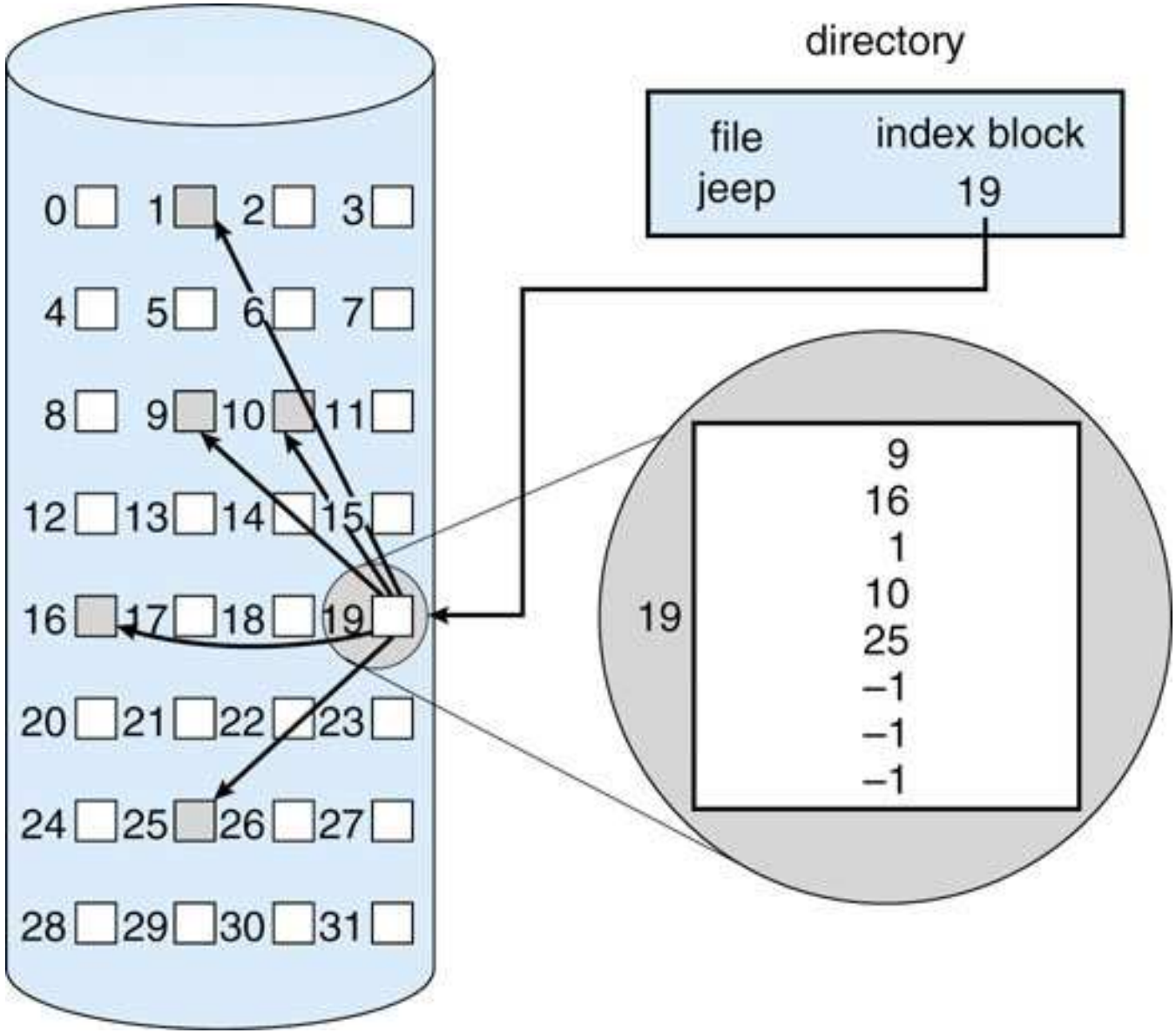
Uwaga: praktycznie dostęp tylko sekwencyjny.

Tablica alokacji FAT

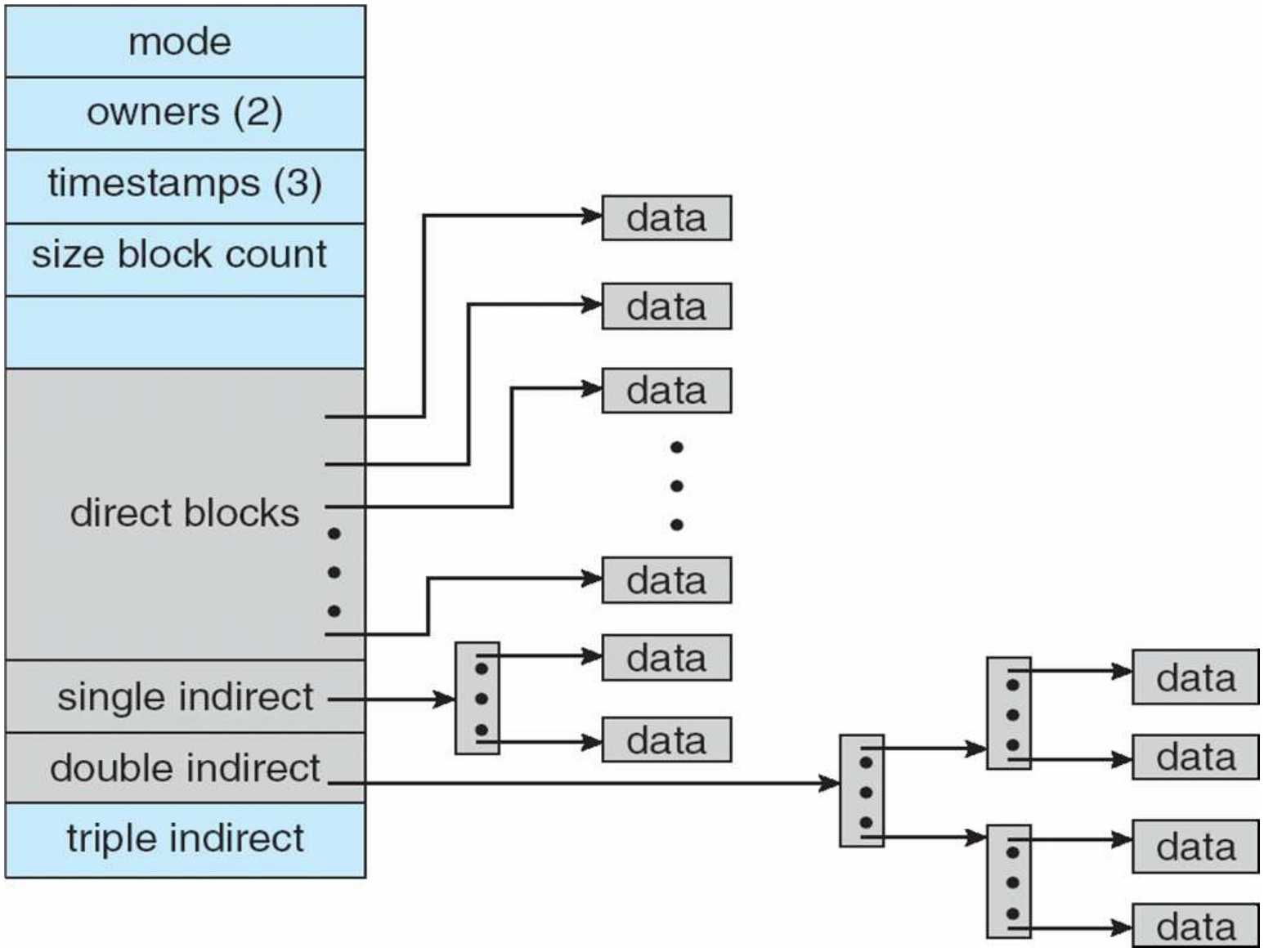
Wykorzystywana w systemach DOS i Windows tablica alokacji FAT (*File Allocation Table*) wykorzystuje alokację listową z tablicą bloków przechowywaną w pamięci.



Indeksowana alokacja bloków



Struktura indeksowana w systemie UNIX



Implementacja katalogów

Jakkolwiek operacje wykonywane na katalogach są proste (przeszukiwanie, wpisywanie, usuwanie), to wykonywane są często, zatem algorytmy i struktury danych wykorzystywane do implementacji katalogów są również istotne.

Najczęściej wykorzystuje się albo strukturę prostą, taką jak lista lub tablica nieuporządkowana (system Unix). Taka struktura pozwala na proste i szybkie implementacje dodawania i usuwania plików, ale wyszukiwanie jest bardziej kosztowną operacją. Dodatkowo, prezentacja zawartości katalogu w postaci uporządkowanej wymaga każdorazowego sortowania.

Jednak zastosowanie tablicy lub listy sortowanej komplikuje operacje tworzenia i usuwania plików. Usuwanie można uprościć przez zaznaczanie pozycji katalogu jako usuniętych, ale to ma swoje oddzielne konsekwencje.

Można również do implementacji katalogu wykorzystać strukturę bardziej zaawansowaną, taką jak tablica haszowa. Ma ona przewagę błyskawicznego dostępu do dowolnej pozycji nawet przy bardzo dużej liczbie pozycji, ale typowo marnuje dużo więcej miejsca, oraz wymaga zaimplementowania złożonych algorytmów obsługi kolizji funkcji haszowych.

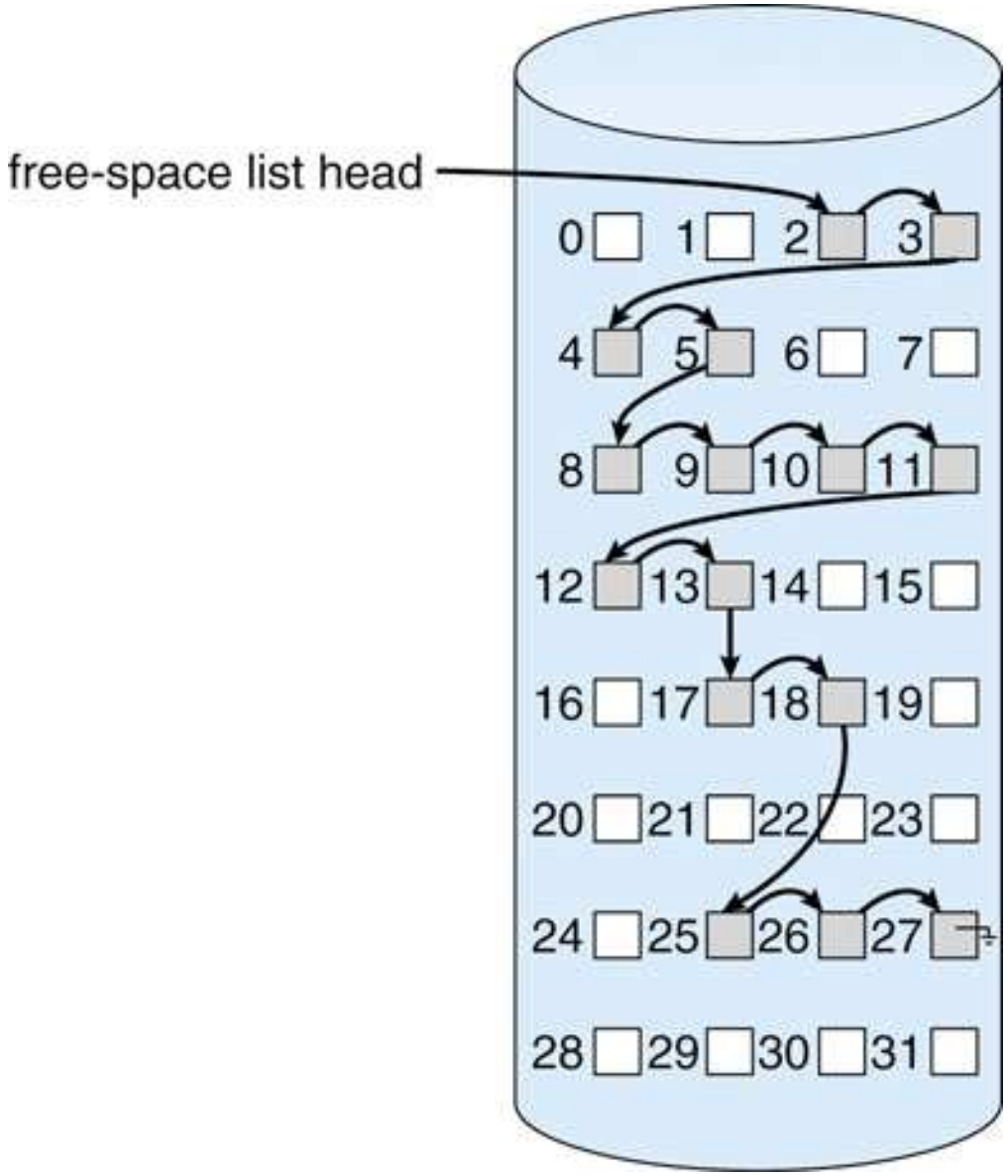
Zarządzanie wolną przestrzenią: mapa bitowa

Osobnym mechanizmem funkcjonowania systemu plików jest zarządzanie pulą wolnych bloków dyskowych.

Prostą metodą jest zastosowanie **mapy bitowej**, albo **wektora bitów**, gdzie każdy pojedynczy bit reprezentuje blok dyskowy i oznacza jego stan: 1 jeśli wolny, i 0 jeśli zajęty.

Ta prosta w implementacji metoda pozwala zwalniać bloki w czasie stałym. Przydział nowego bloku wymaga znalezienia na mapie ustawionego bitu, co wiąże się z przeszukiwaniem. Z kolei mapa bitowa pozwala łatwo znajdować ciągłe bloki wolnej przestrzeni o wymaganej wielkości.

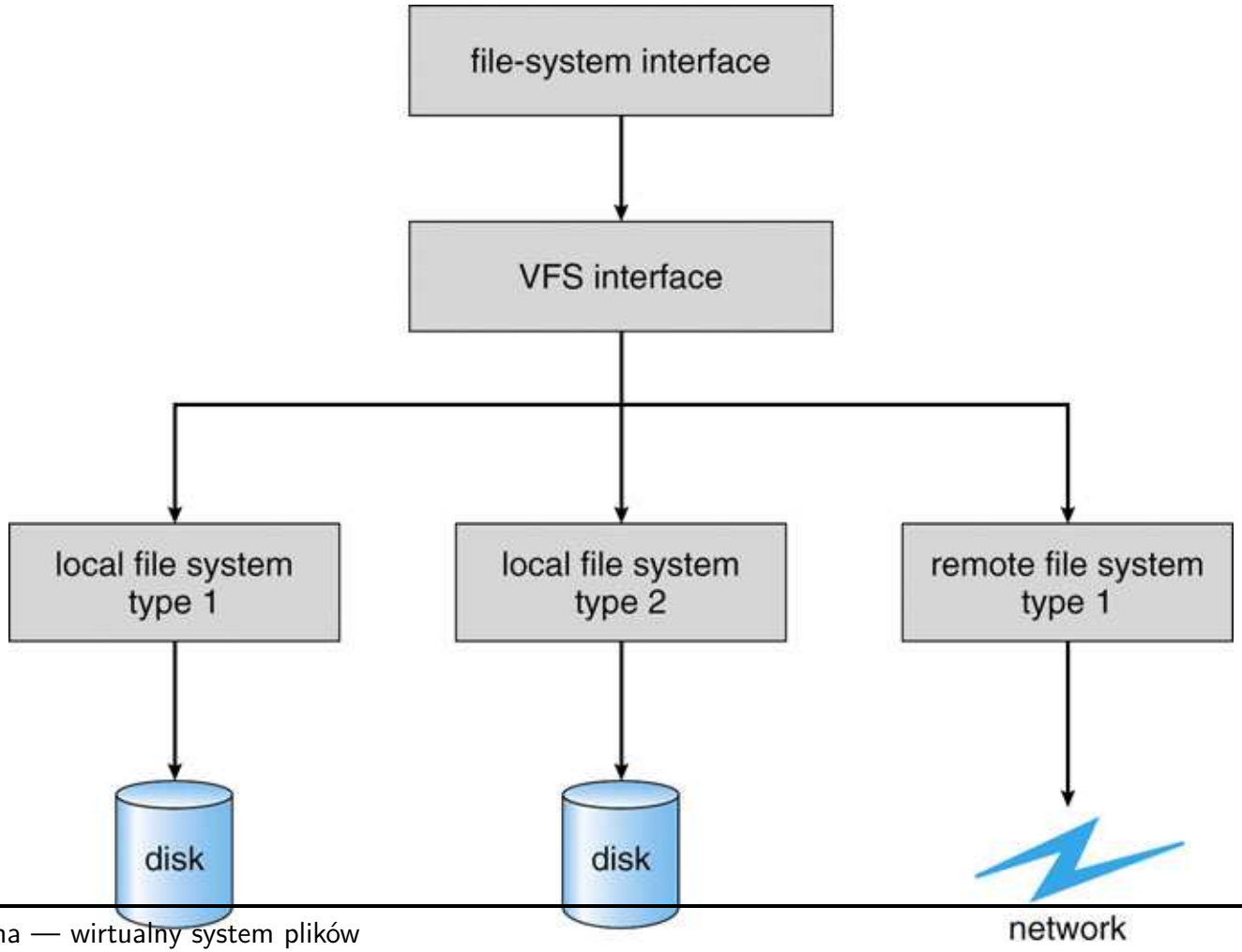
Zarządzanie wolną przestrzenią: lista wolnych bloków



Zarządzanie pulą wolnych bloków przez listę wskaźnikową: pobieranie i oddawanie bloków jest możliwe w stałym czasie.

Wirtualny system plików

Wirtualny system plików (VFS) jest uogólnionym interfejsem do wielu różnych systemów plików. Dostarcza on np. unikalnych identyfikatorów plików wspólnych dla wielu różnych systemów plików. Zauważmy, że jeśli w danym systemie istnieje kilka systemów plików, to ich wewnętrzne identyfikatory, takie jak bloki *i-node* w systemach Unix odnoszą się wyłącznie do jednego systemu plików.



System VFS w Linuksie posługuje się następującymi obiektami:

- obiekt *inode* reprezentujący blok kontrolny indywidualnego pliku,
- obiekt *file* reprezentujący otwarty plik,
- obiekt *superblock* reprezentujący system plików,
- obiekt *dentry* reprezentujący pozycję katalogu w systemie plików.

Zestaw operacji dostarczanych przez linuksowy VFS obejmuje: open, read, write i mmap.

Sieciowy system plików

Najstarszą i najprostszą metodą współdzielenia plików w sieci było ich kopiowanie, na przykład wykorzystując protokół **ftp** albo jedną z jego bardziej prymitywnych, lub bardziej wyrafinowanych wersji.

Podejściem alternatywnym jest **rozproszony system plików** DFS (*Distributed File System*), który umożliwia dostęp do katalogów zdalnych systemów plików tak jakby były obecne w lokalnym systemie. NFS (*Network File System*) umożliwia współdzielenie między komputerami systemów plików, albo poddrzew katalogów:

- serwer NFS **eksportuje** strukturę dyskową (zwykle: system plików),
- klient NFS **montuje** strukturę w wybranym katalogu, tak jakby to był system plików na własnym dysku fizycznym,
- użytkownik klienta operuje na plikach, w ramach swoich uprawnień, związanych z plikiem na systemie serwera,
- klient NFS odmontowuje strukturę, jeśli chce zakończyć użytkowanie.

Prawa dostępu systemu NFS są oparte na identyfikatorach użytkownika, co wymaga stosowania jednolitego systemu identyfikatorów w całej jednostce.

Krótkie podsumowanie — pytania sprawdzające

1. Jakie są wady i zalety metod ciągłej i listowej alokacji bloków w systemie plików?

2. Jakie są wady i zalety metody indeksowanej alokacji bloków w systemie plików?

3. Jakie są możliwe metody implementacji katalogów?

4. Czy wiesz jaka jest implementacja systemu plików oraz katalogów w jakimś znanym Ci systemie (Windows, Android, ios, Linux)?

5. Dlaczego lista wskaźnikowa jest odpowiednią strukturą do implementacji puli wolnych bloków pamięci dyskowej?

Uszkodzenia systemu plików

Zarówno pojedyncze pliki jak i systemy plików na dysku mogą ulegać uszkodzeniom. Jest to niezależne od posiadania redundatnych struktur dyskowych typu RAID, które też mogą ulegać awariom i nie gwarantują 100% bezpieczeństwa zapisanych danych.

W przypadku pojedynczych plików, waga i konsekwencje takiego zdarzenia są zwykle mniejsze, a jednocześnie możliwości naprawy uszkodzenia pliku są zwykle ograniczone. Należą do nich: odtworzenie pliku z kopii zapasowej, odtworzenie brakującej części danych z części zachowanej, i ... niewiele więcej.

Jednak uszkodzenie systemu plików może skutkować niemożnością odtworzenia wszystkich plików, pomimo iż albo przeważająca ich większość, albo nawet wszystkie, mogą nadal istnieć na dysku nieuszkodzone. Odtworzenie plików w takiej sytuacji jest zwykle możliwe przy pomocy specjalistycznych narzędzi, które jednak wymagają dodatkowego nakładu specjalistycznej pracy manualnej, co jest bardzo kosztowne.

Dlatego systemy plików są konstruowane w taki sposób, aby zapewnić dużą odporność na uszkodzenia wskutek błędów w pojedynczych blokach, oraz możliwość maksymalnie automatycznej naprawy systemu, gdy uszkodzenie powstanie. Niekiedy przy takiej naprawie potrzebna jest interwencja operatora, ale ma ona charakter decyzji czy dany plik można bez szkody usunąć, i nie wymaga długotrwałej pracy specjalistów.

Taka odporna struktura systemu plików może być również automatycznie sprawdzana pod względem wewnętrznej spójności.

Księgujące systemy plików

Najczęstsze przypadki uszkodzenia systemu plików nie wynikają z awarii dysku, ale z wyłączenia zasilania w momencie, gdy trwał zapis na dysku. Dlatego odporne systemy plików konstruuje się w taki sposób, by maksymalnie zabezpieczyć się przed takimi zdarzeniami. Jeden taki mechanizm odporności nazywa się księgującym systemem plików (*journaling file system*).

Księgujący system plików wykonuje każdą operację na systemie plików w ten sposób, że przed faktycznym wykonaniem operacje są zapisywane do dziennika (*log file*). Ze względu na istniejące buforowanie, i opóźnione realizacje operacji dyskowych, wpis operacji do dziennika musi być doprowadzony do końca, i zweryfikowany, zanim rozpocznie się rzeczywista realizacja operacji na systemie. Po faktycznym wykonaniu tej operacji, jej wpis jest usuwany z dziennika.

Ta procedura gwarantuje, że jeśli praca systemu zostanie przerwana na którymkolwiek etapie, to awaria może objąć albo wpis do dziennika, albo realizację operacji na dysku, ale nie jedno i drugie.

W takim przypadku, po przywróceniu działania systemu, może on sprawdzić, czy operacja została wykonana w całości czy nie, i dokończyć niewykonane etapy.

Idempotentne i niepodzielne operacje

Działanie mechanizmu księgującego wymaga aby operacje wpisane do dziennika były **idempotentne**, to znaczy, żeby mogły być wykonane ponownie, nawet wiele razy, z tym samym skutkiem.

Na przykład, operacja usunięcia wpisu pliku o danej nazwie z katalogu — np. `prog1.c` — jest idempotentna. Jeśli została już wykonana wcześniej, to może być wykonana ponownie bez wprowadzania błędów. Ponowna próba usunięcia wpisu nie znajdzie go w katalogu, więc nie będzie miała efektu.

Ale operacja dodawania bloków kasowanego pliku do listy wolnych bloków może nie być idempotentna, ponieważ jeśli była wykonana wcześniej, to te bloki już są na liście, i ich ślepe dodanie utworzyłoby niepoprawną listę. Księgujący system plików musi wykonywać wszystkie operacje idempotentnie, zatem dodawanie bloków musi odbyć się poprzez przeszukanie listy wolnych bloków, i dodanie tylko tych, których jeszcze tam nie ma.

Krótkie podsumowanie — pytania sprawdzające

1. Co to jest księgujący system plików i w jakim celu jest stosowany?

2. Dlaczego w księgującym systemie plików operacje dyskowe są najpierw zapisywane do dziennika, a dopiero potem wykonywane?

3. Co to są operacje idempotentne?