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: przeszukiwanie, sortowanie, sprawdzanie spójności systemu plików

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.

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

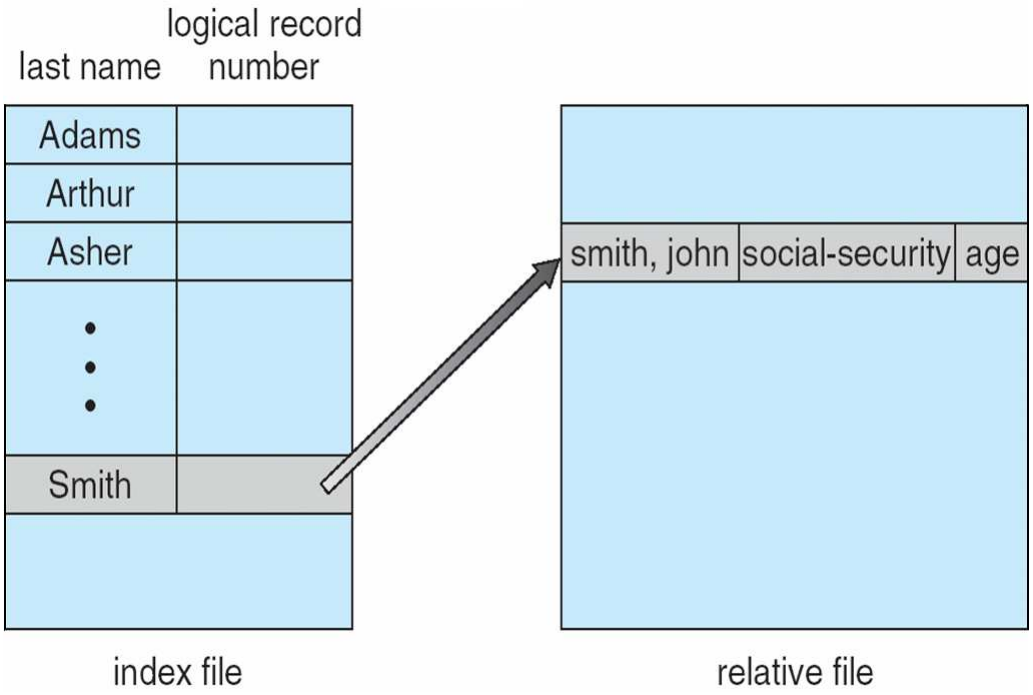
Z typem pliku związana jest często **struktura**, albo **inaczej organizacja logiczna**. Przykładem prostej struktury pliku jest **ciąg bajtów**. Taką strukturę obsługują takie systemy jak Unix albo DOS. W ramach tej struktury obsługiwane są wszystkie pliki w systemie, np. pliki tekstowe, pliki binarne 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, ze stratą tylko jednego bitu na słowo.)

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

Pliki indeksowane



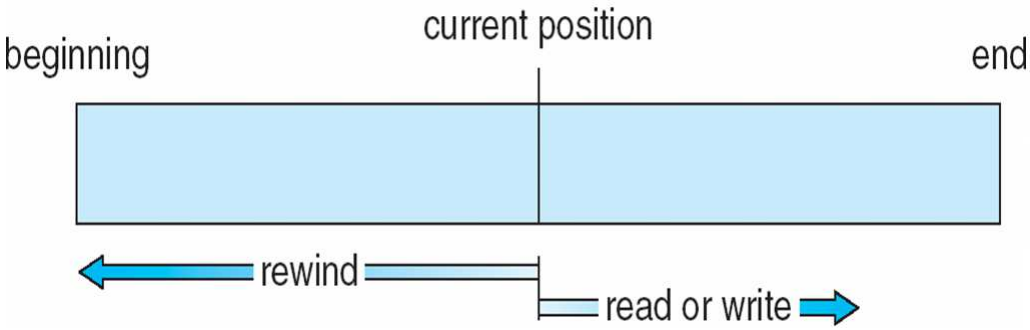
Pliki indeksowane składają się z dwóch części, z których jedna zawiera indeks, a druga właściwe dane. Takie pliki można traktować jako rodzaj prostej bazy danych, gdzie dane posiadają klucze według których są zapamiętywane i wyszukiwane.

Struktura wewnętrzna

Do tego dochodzi zagadnienie struktury wewnętrznej, która jest związana z implementacją na typowych dyskach, oferujących na ogół operacje na blokach dyskowych o stałym rozmiarze. W zależności od struktury pliku, rekordy są w jakiś sposób pakowane do bloków dyskowych. Wielkość bloku wpływa na efektywność operacji dyskowych, ale również prowadzi do **fragmentacji wewnętrznej**.

Dostęp sekwencyjny

Kolejnym aspektem obsługi plików przez system jest sposób dostępu.



Dostęp bezpośredni

Dla plików przechowywanych na dyskach możliwe jest przejście do dowolnego bloku pliku bez sekwencyjnego czytania wszystkich bloków poprzedzających. Operację tę nazywa się **pozycjonowaniem** (*seek*), albo czasami **przesuwaniem kursora**.

Możliwość dostępu bezpośredniego i wykonalność operacji pozycjonowania wynika z implementacji plików na dyskach magnetycznych, które w naturalny sposób umożliwiają obliczenie adresu bloku dyskowego zawierającego poszukiwaną daną, i pobranie dokładnie tego bloku.

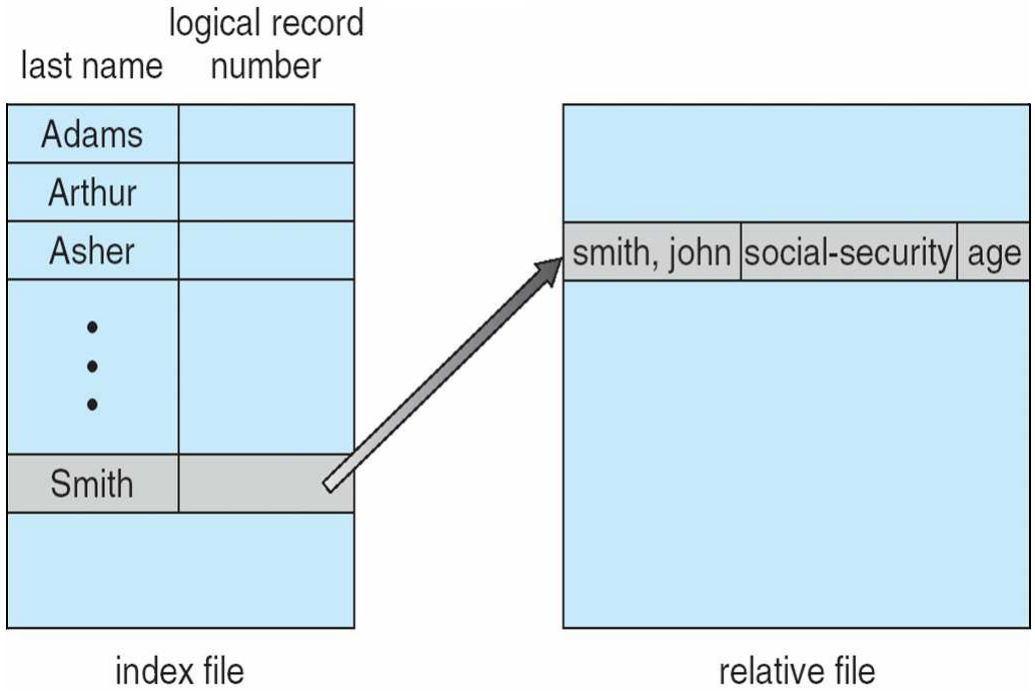
Jak zobaczymy nieco później, w niektórych implementacjach systemów plików, nawet na urządzeniach dyskowych, nie jest możliwy taki sposób dostępu.

Symulacja dostępu sekwencyjnego

sequential access	implementation for direct access
reset	$cp = 0;$
read next	*read cp*; $cp = cp + 1;$
write next	*write cp*; $cp = cp + 1;$

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

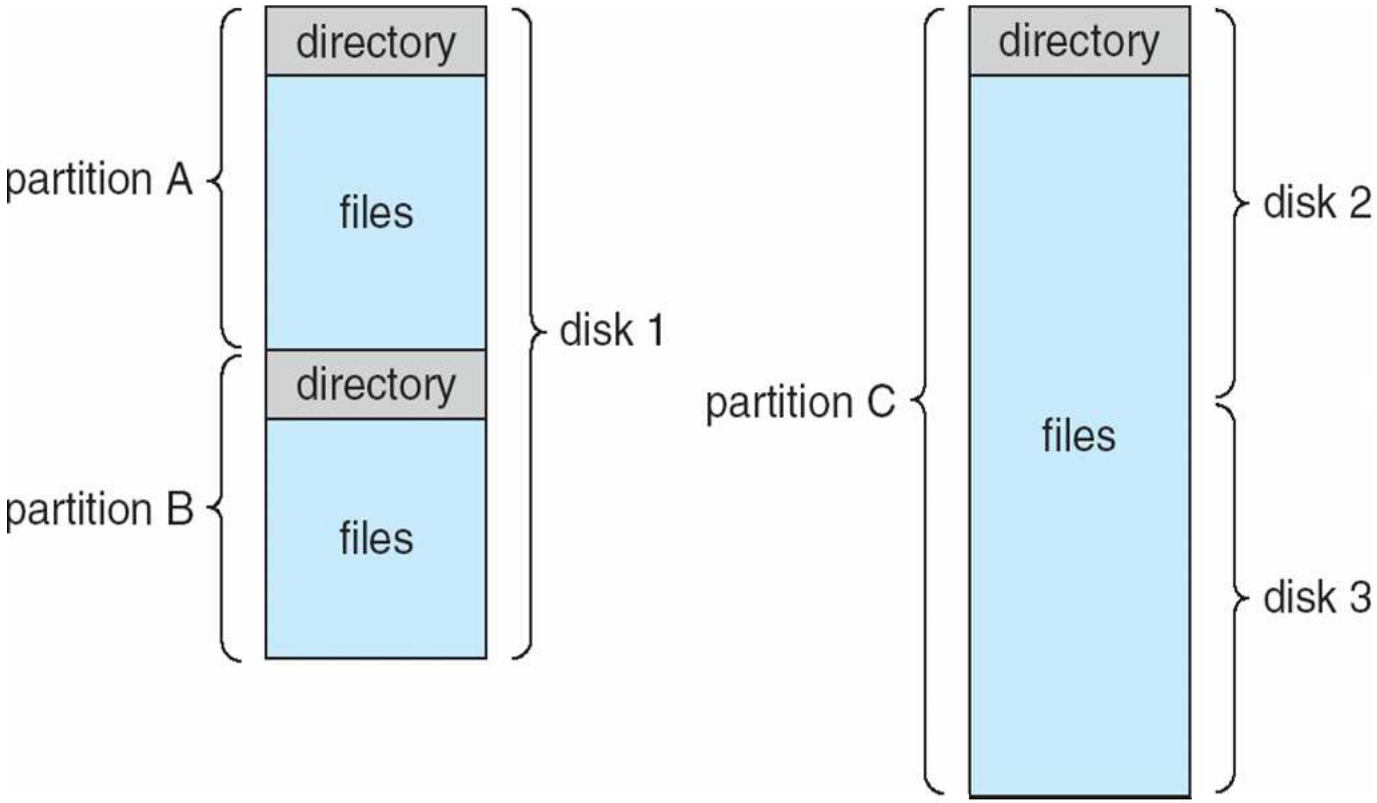
1. Jakie są podstawowe operacje wykonywane na plikach?

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



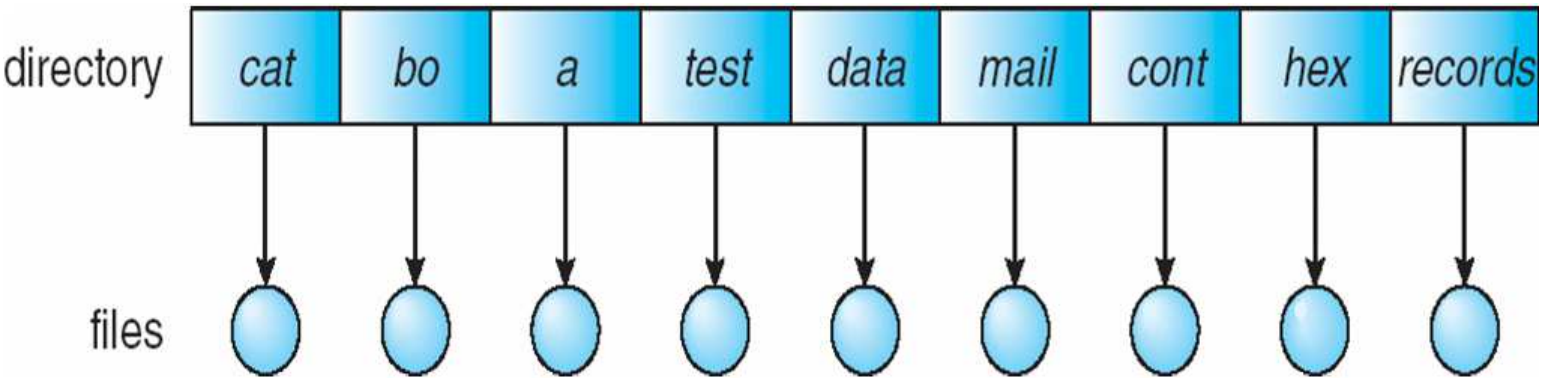
Partycjonowanie dysków umożliwia łatwiejszą organizację rozmieszczenia różnych elementów systemu na dysku. Na przykład: jedna partycja może zawierać system operacyjny, a inna być używana jako obszar wymiany *swap*. Partycja może obejmować część jednego dysku, cały dysk, lub wiele dysków. System plików składa się z **katalogu** oraz obszaru alokacji poszczególnych plików.

Organizacja logiczna systemu plików: katalogi

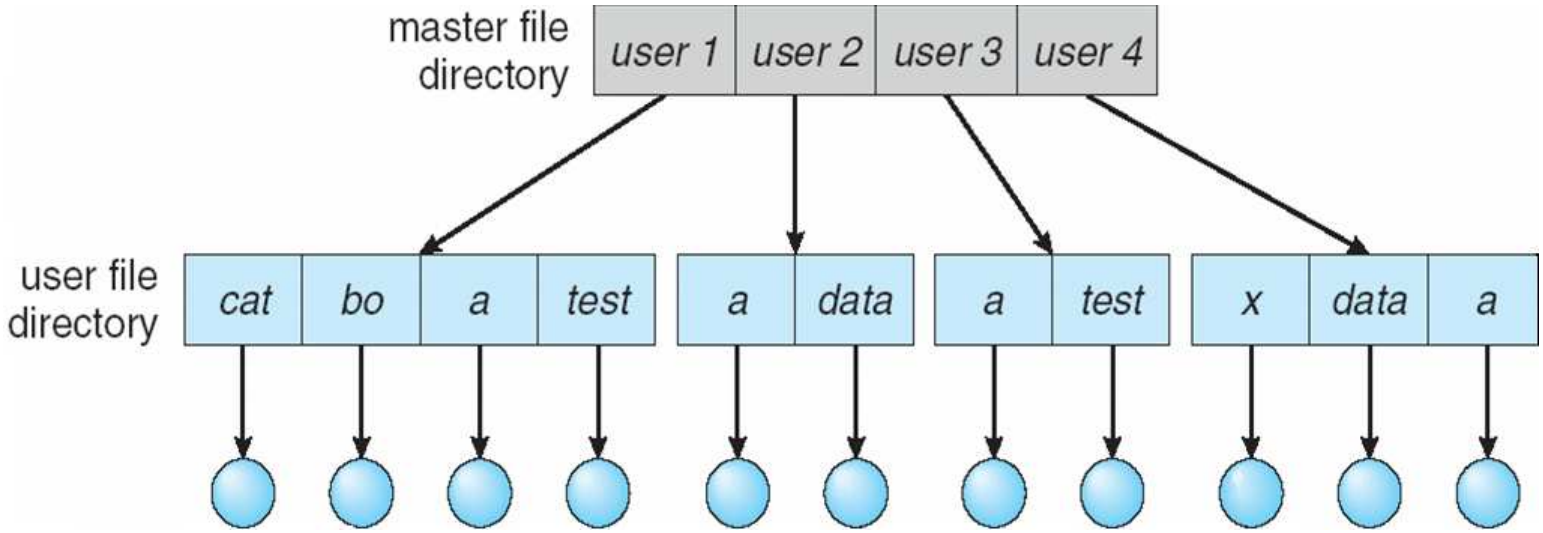
Systemy operacyjne posługują się wieloma plikami zarówno dla organizacji samego systemu, jak i w celu udostępnienia operacji programom. Systemy plików mogą mieć różną organizację, ale dla umożliwienia tworzenia, przechowywania, i dostępu do wielu plików posiadają na ogół strukturę zwaną **katalogiem**.

Katalog w ogólności jest listą plików zawartych w systemie, z ich nazwami i informacją o alokacji pliku na dysku, jak adres bloku kontrolnego pliku (patrz niżej), lub bezpośrednio adres pierwszego bloku alokacji.

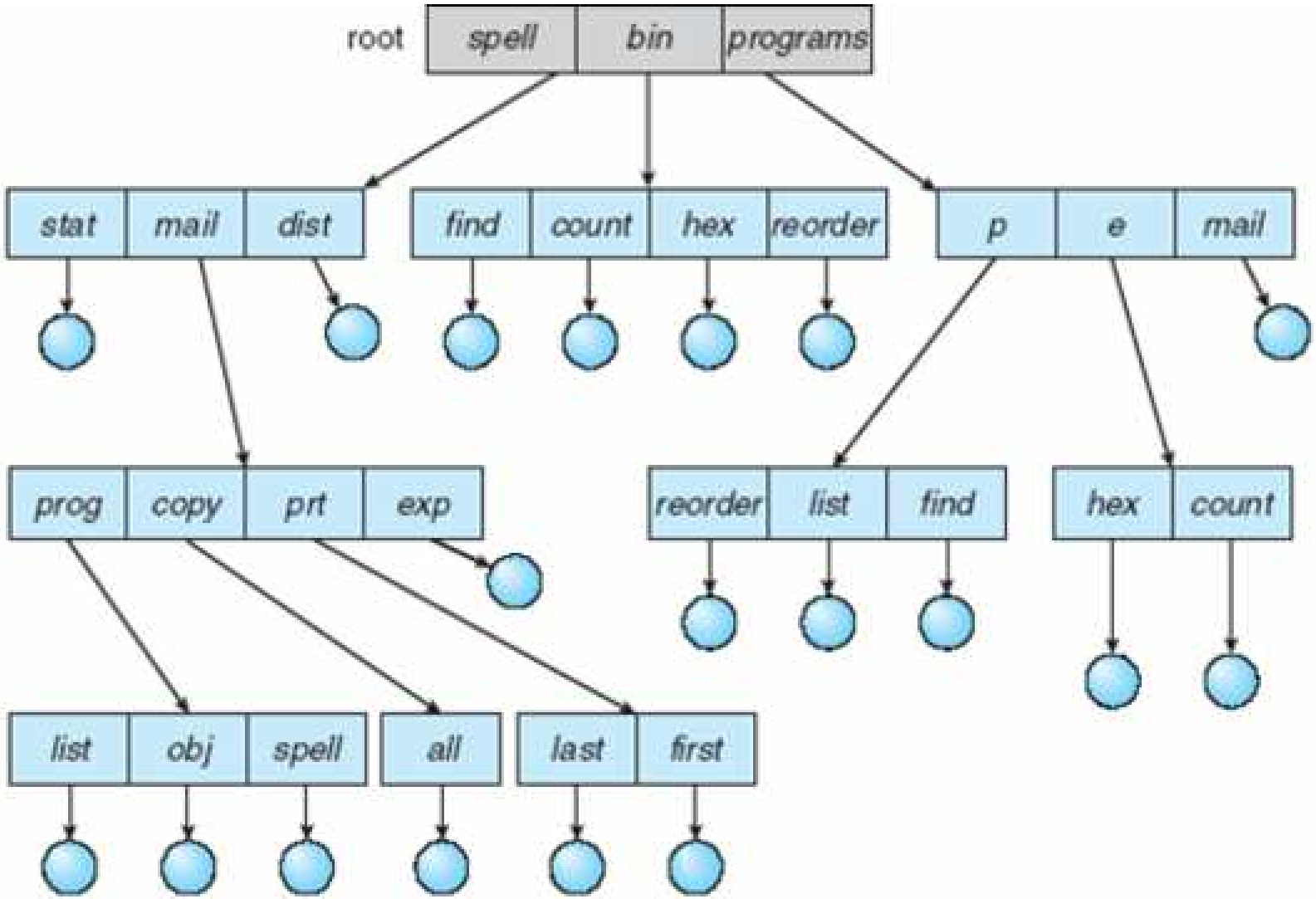
Katalogi jednopoziomowe



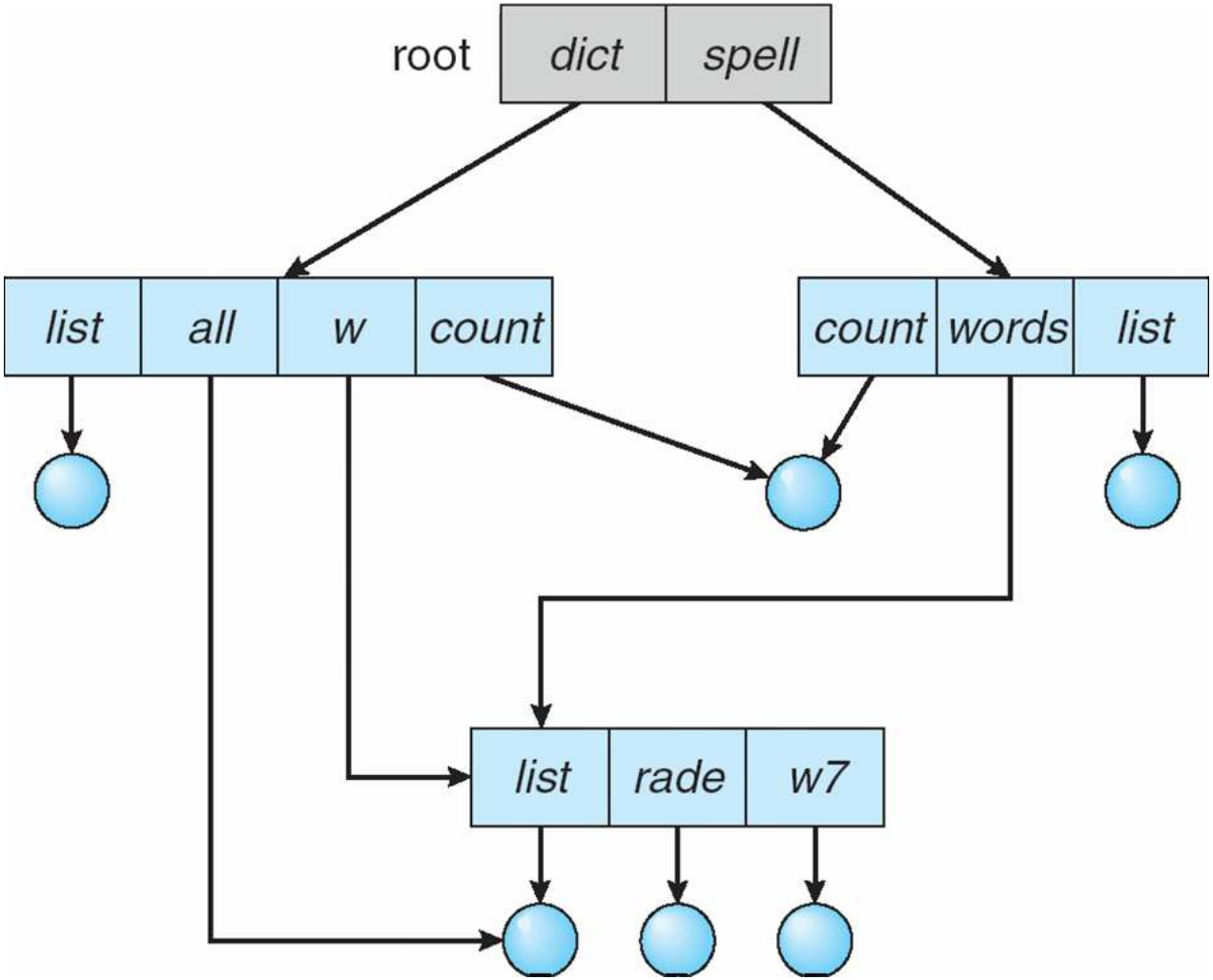
Katalogi dwupoziomowe



Katalogi wielopoziomowe

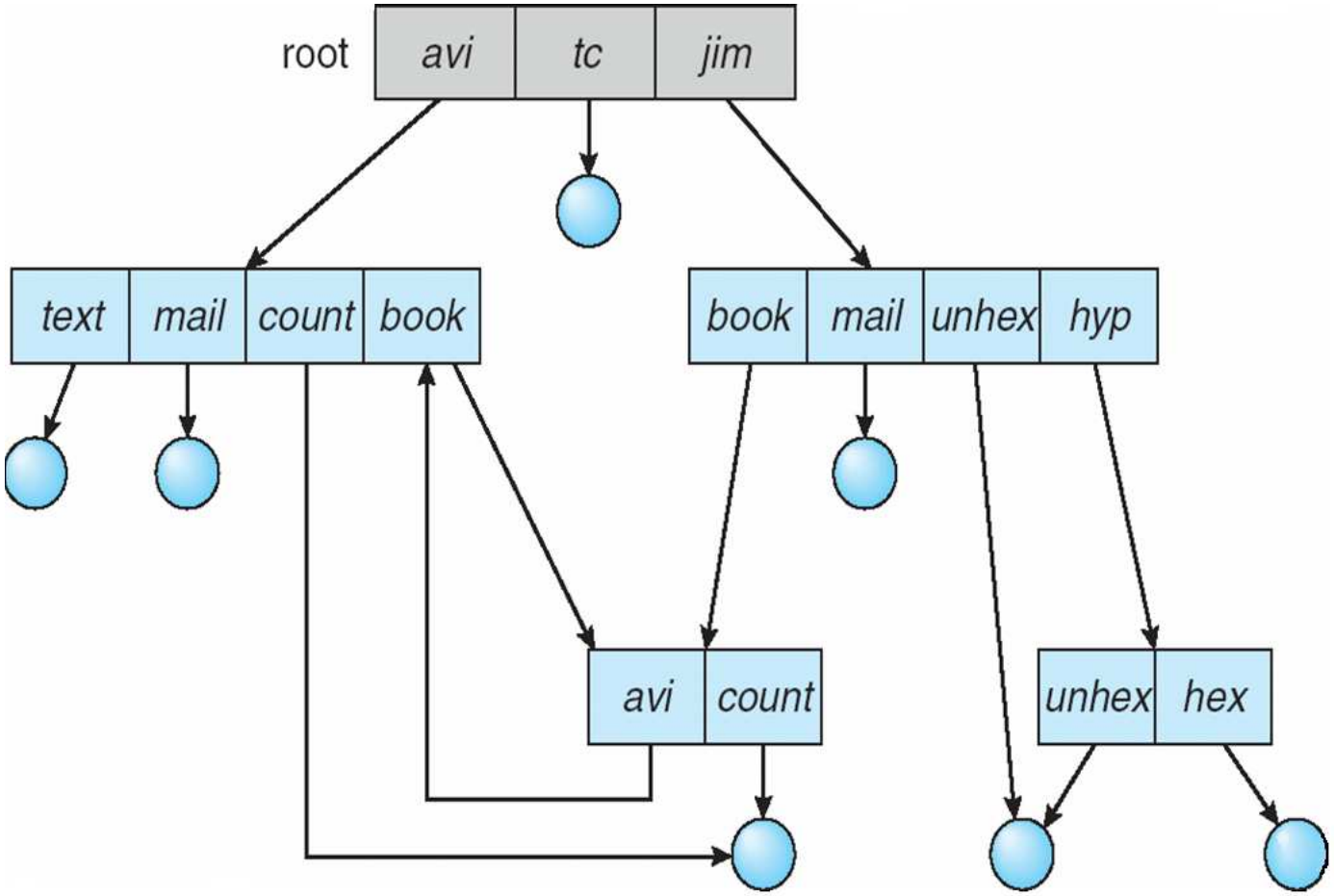


Acykliczna struktura katalogów



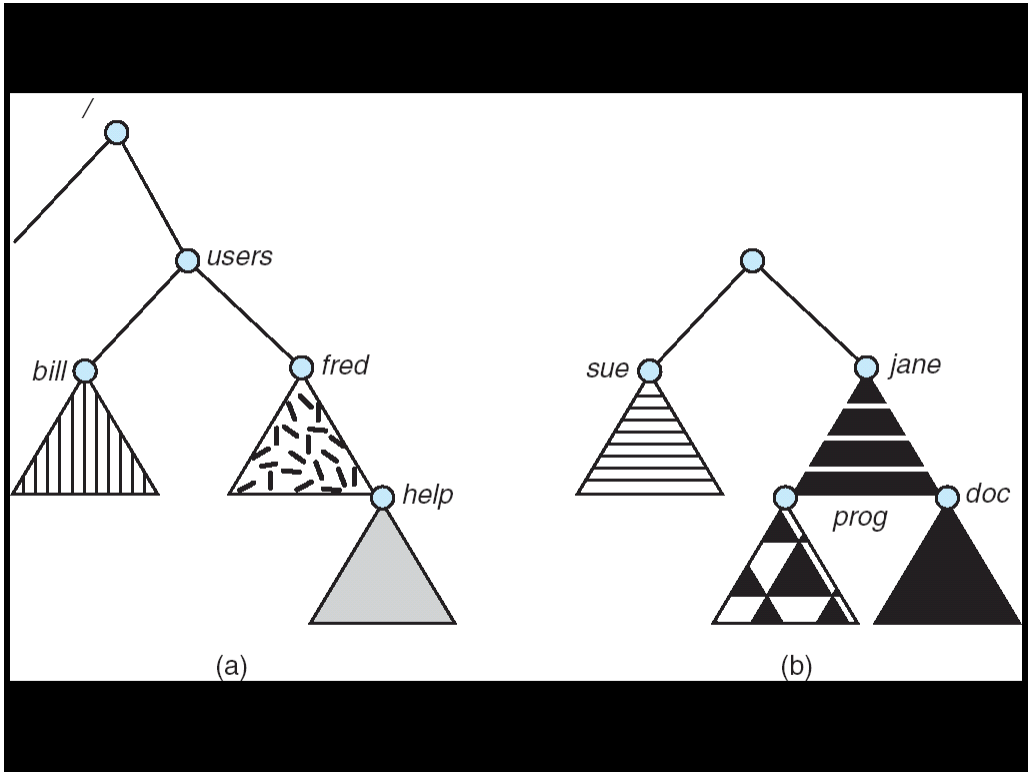
Acykliczna struktura katalogów pozwala na większą ogólność niż struktura ściśle drzewiasta, np. na istnienie jednego pliku w różnych katalogach. Pozwala to na unikanie kopiowania, ale wymaga specjalnych procedur np. przy usuwaniu plików.

Ogólna struktura katalogów



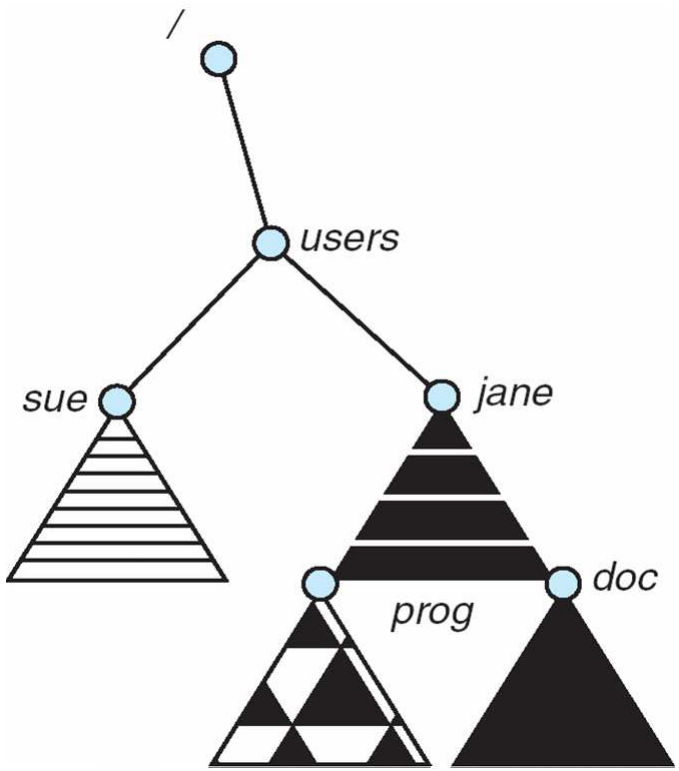
Ogólna struktura katalogów może zawierać cykle między katalogami. Procedury, które zagłębiają się rekurencyjnie w strukturze katalogów mogą w ten sposób wpadać w nieskończone pętle.

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.

Sieciowe systemy 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.

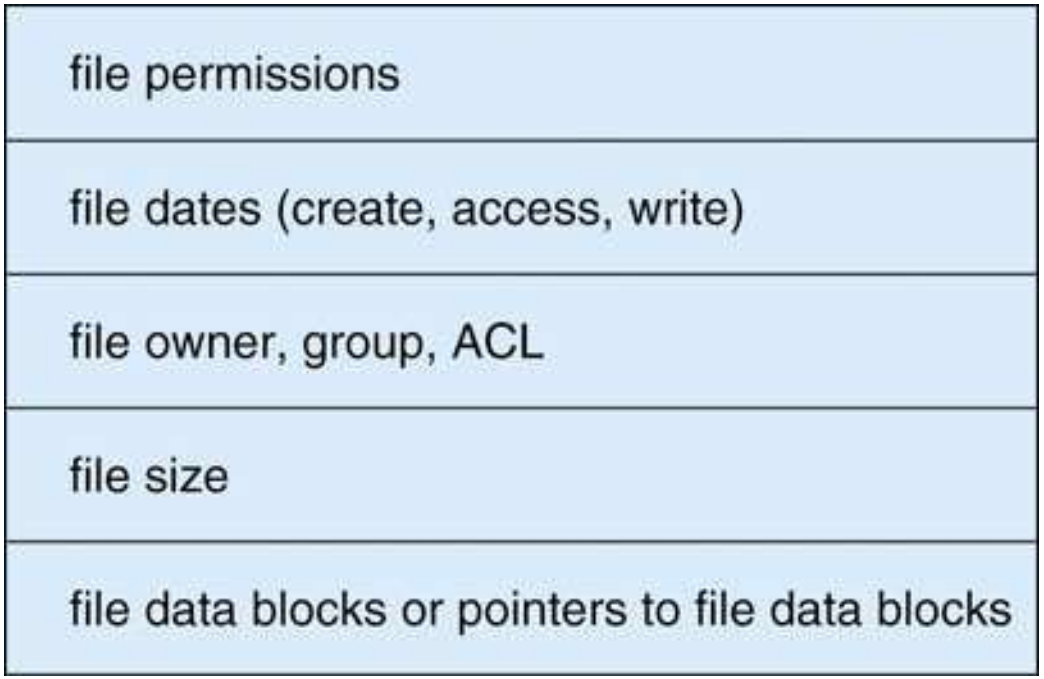
Krótkie podsumowanie — pytania sprawdzające

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

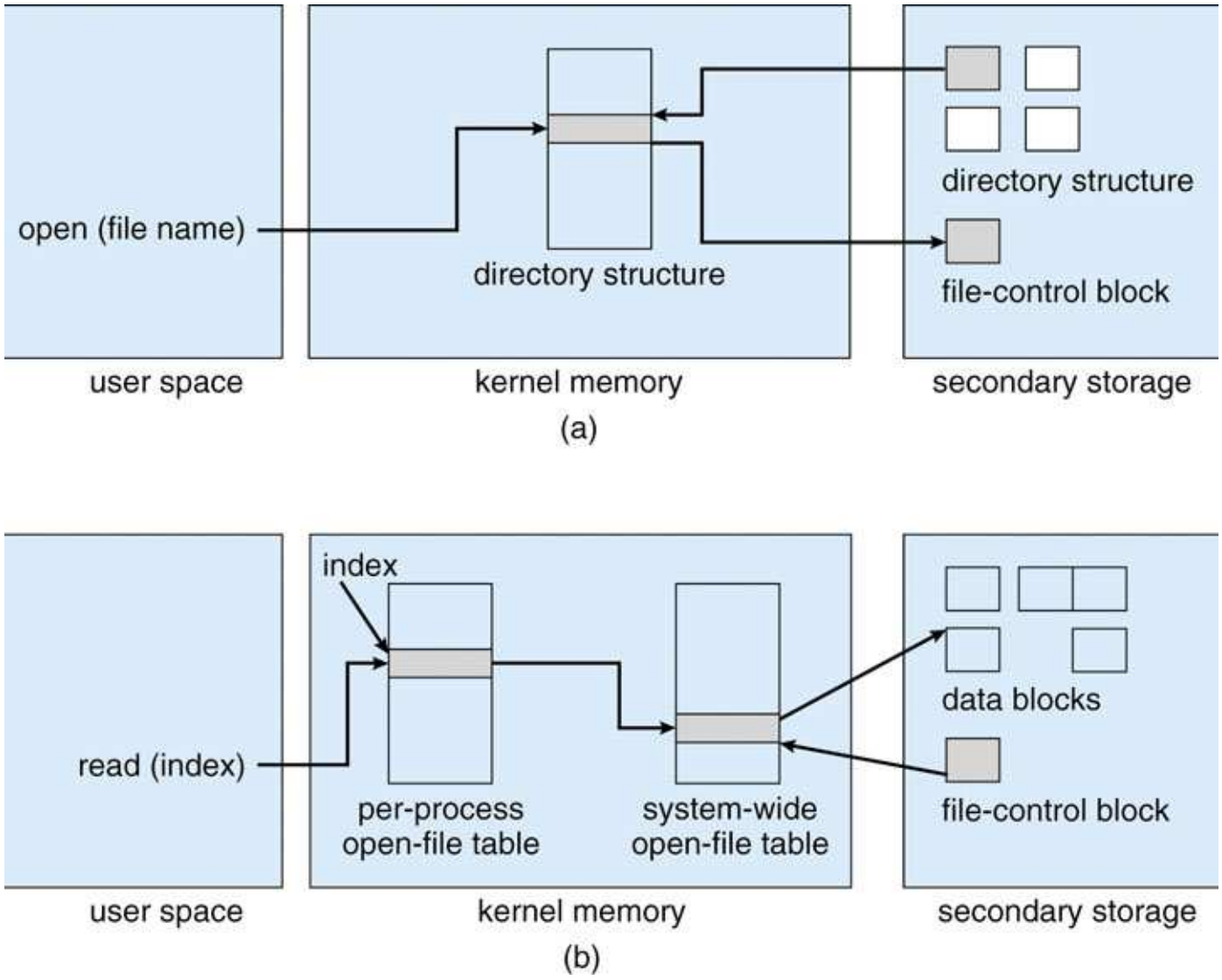
2. Jaką strukturę katalogów implementuje system Windows: drzewiastą, acykliczną, czy ogólną?

3. Czym różni się acykliczna od ogólnej struktury katalogów?

Blok kontrolny pliku

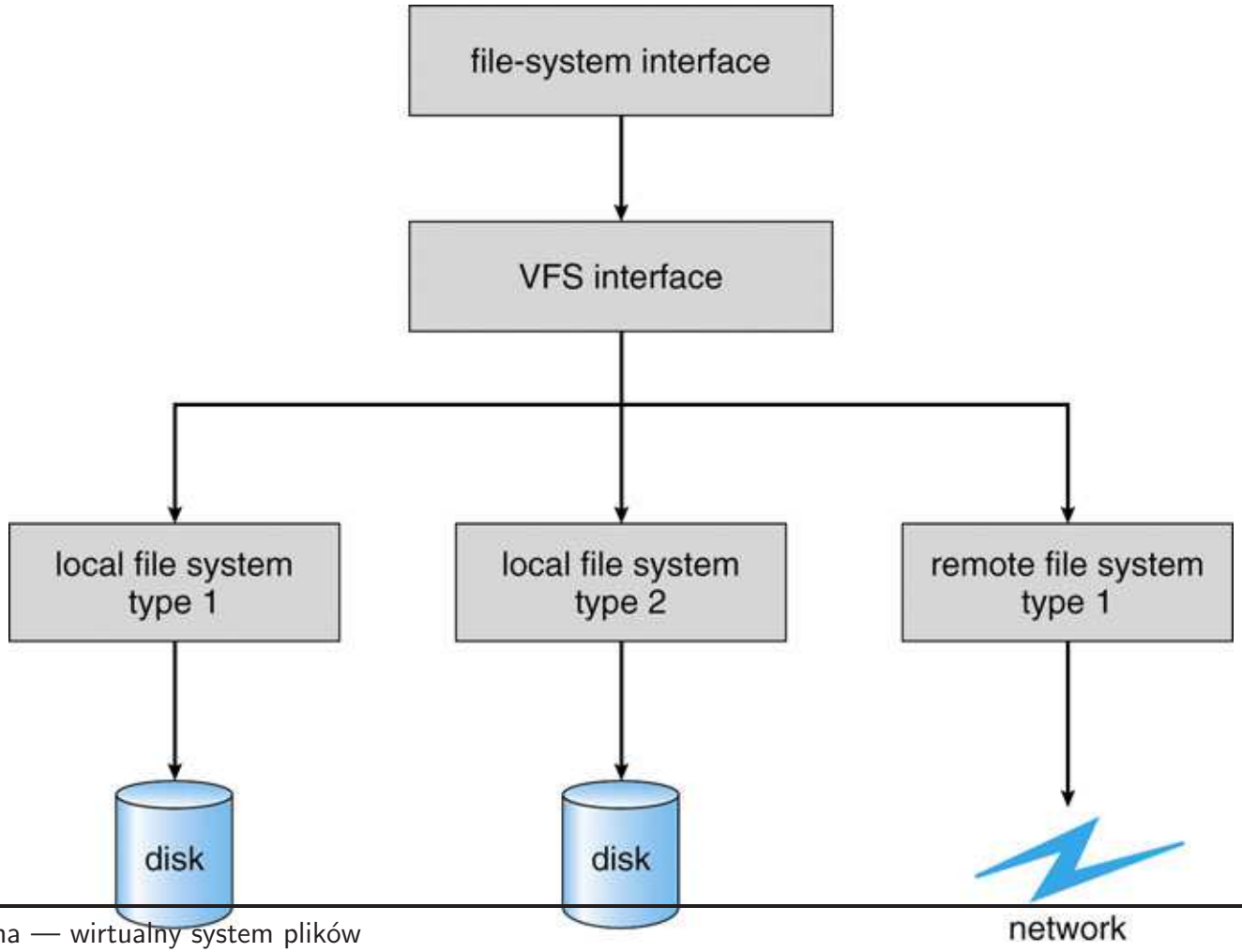


Operacje otwarcia i odczytu plików



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.

Implementacja katalogów

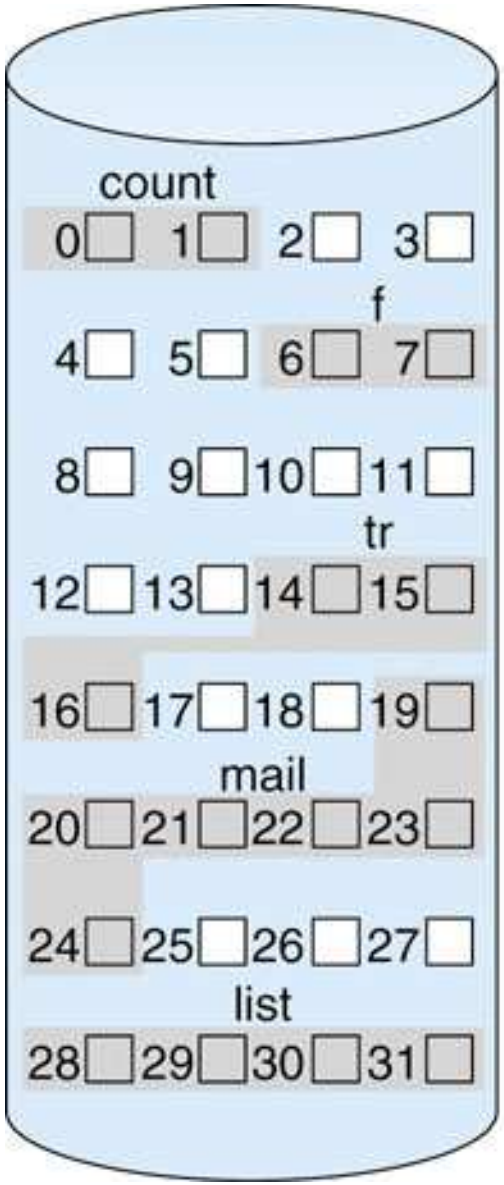
Jakkolwiek operacje wykonywane na katalogach są proste (przeszukiwanie, wpisywanie, usuwania), to wykonywane są często, zatem algorytmy i struktury danych wykorzystywane do implementacji katalogów są również istotne.

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, wielokrotnego 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.

Metody alokacji: ciągła alokacja bloków

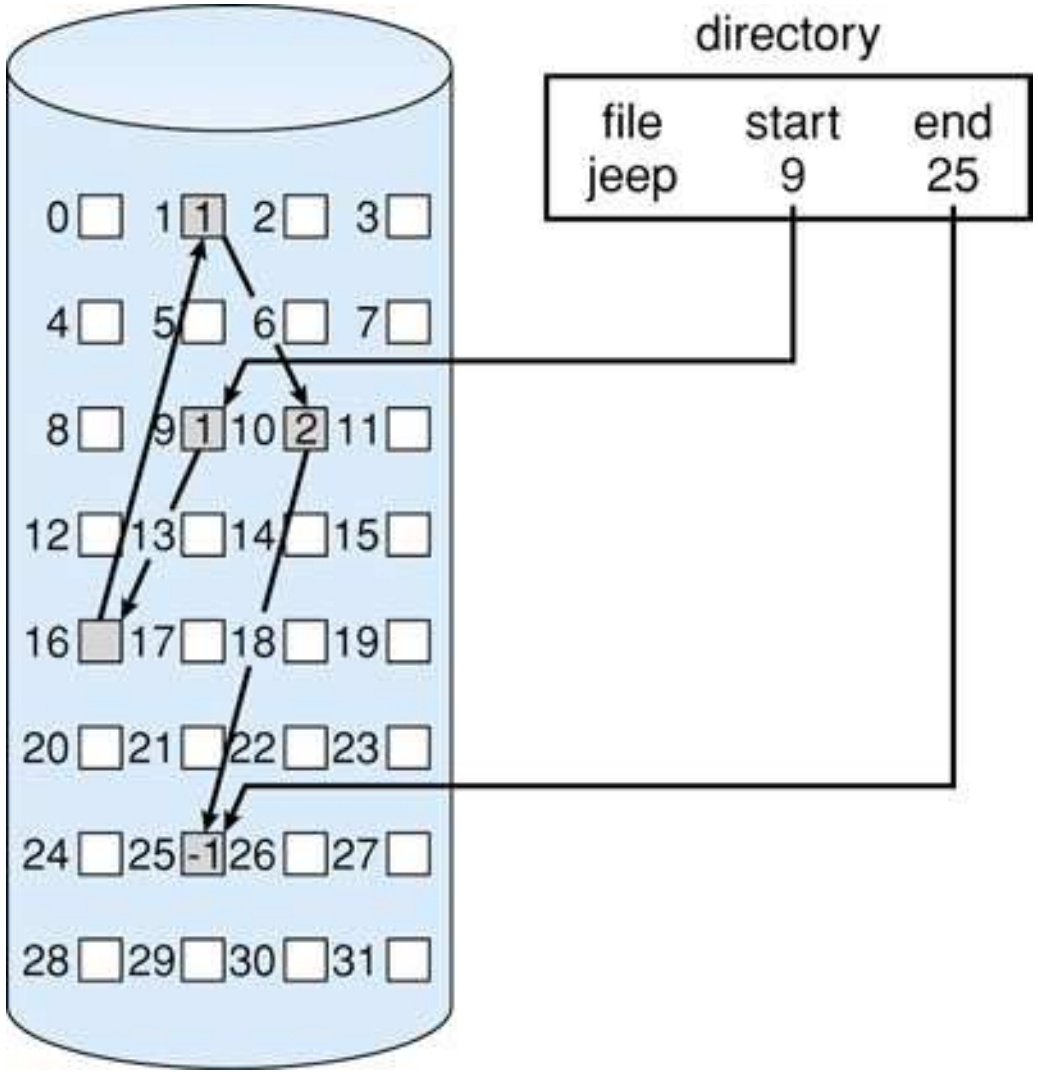


directory

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

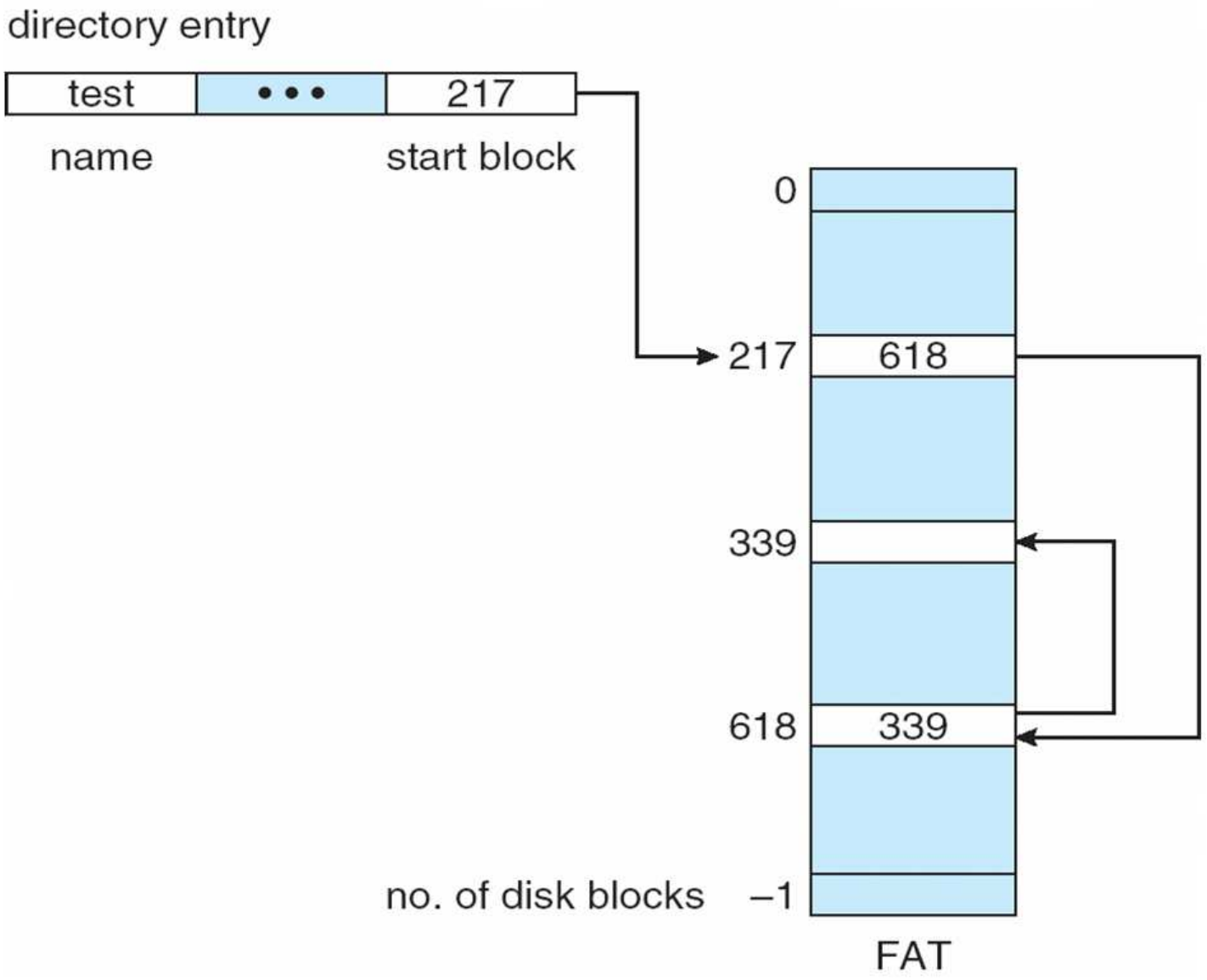
Uwaga na fragmentację zewnętrzną.

Metody alokacji: listowa alokacja bloków

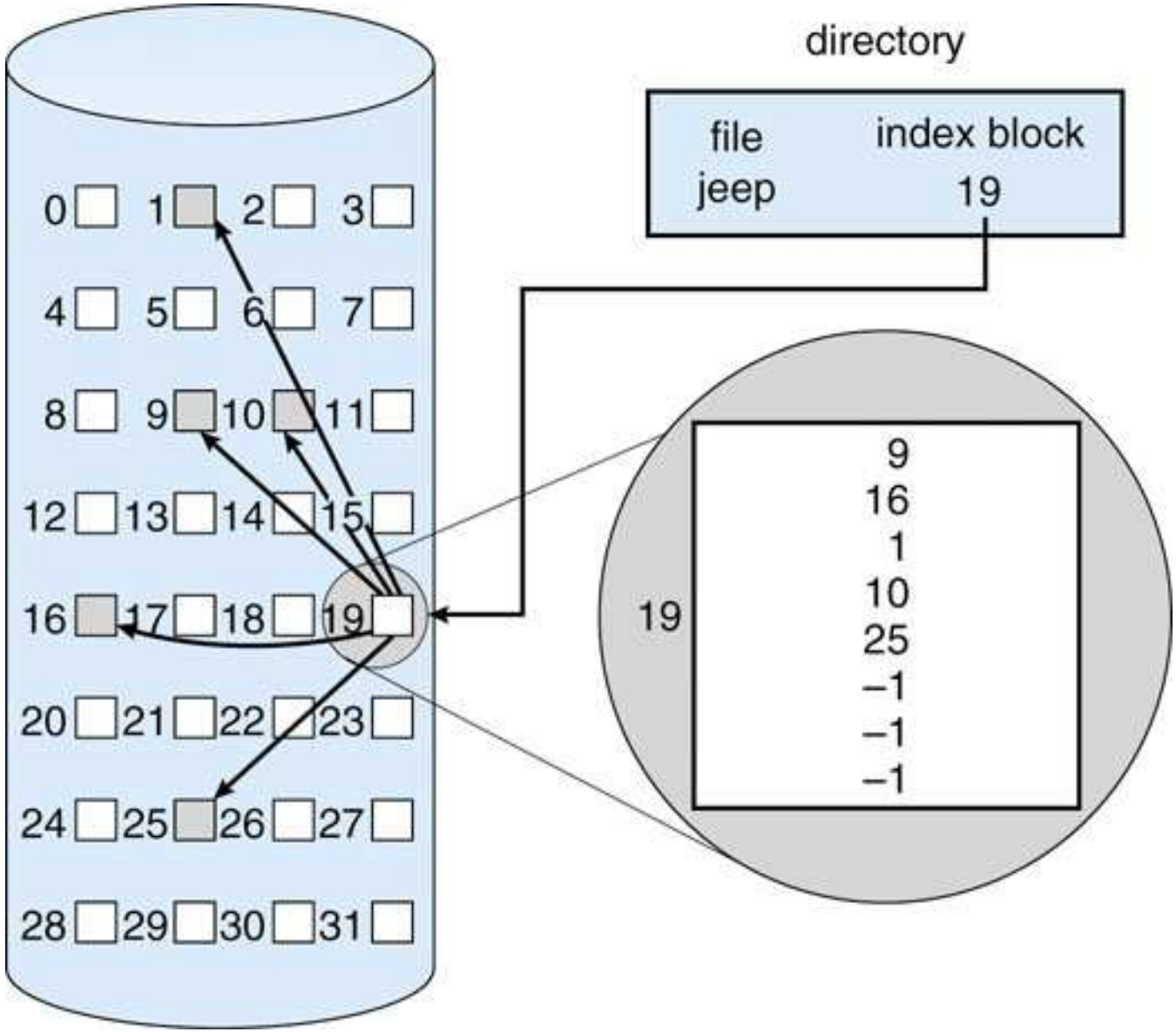


Uwaga: praktycznie dostęp tylko sekwencyjny.

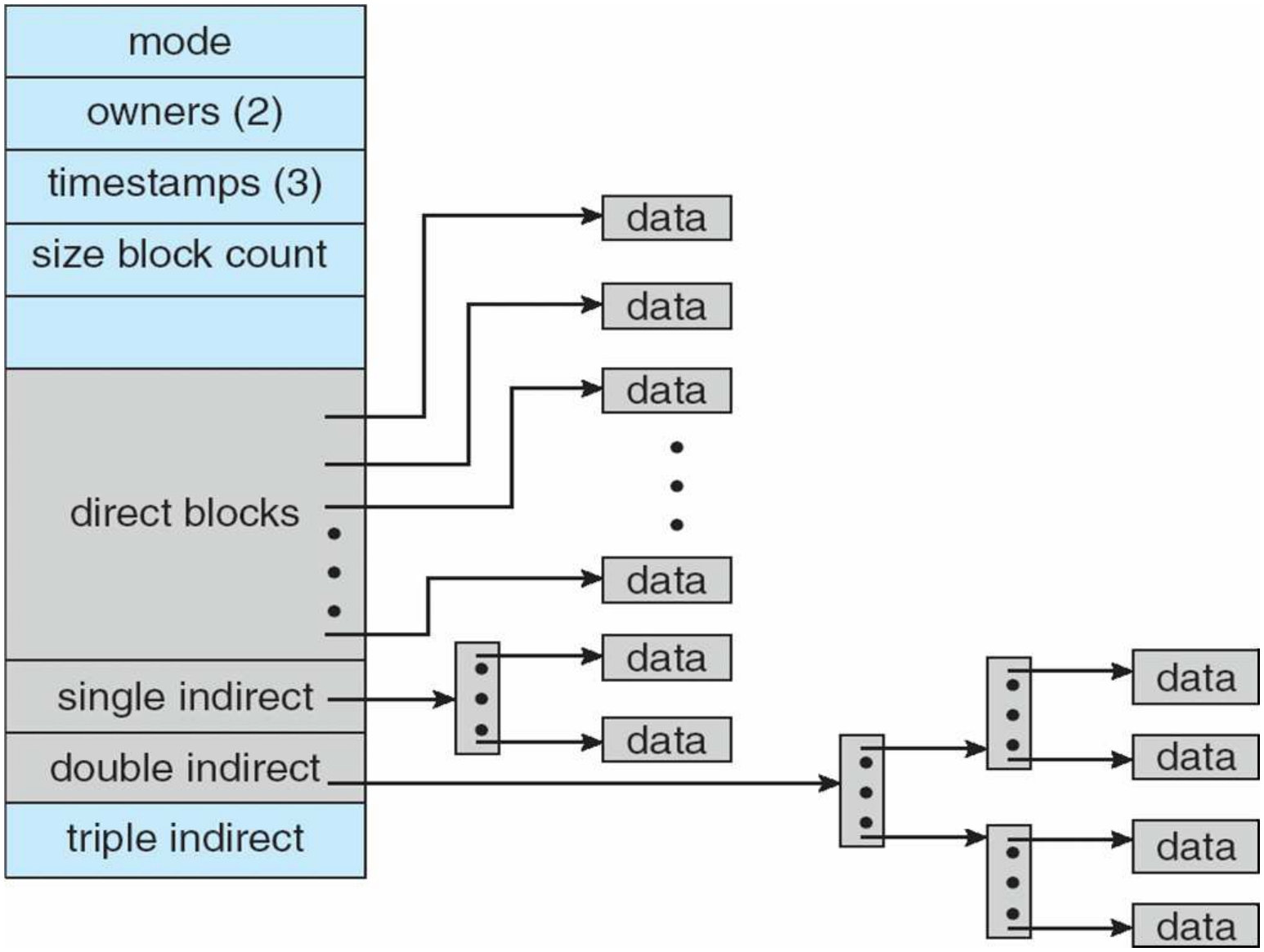
Tablica alokacji FAT (alokacja listowa)



Indeksowana alokacja bloków



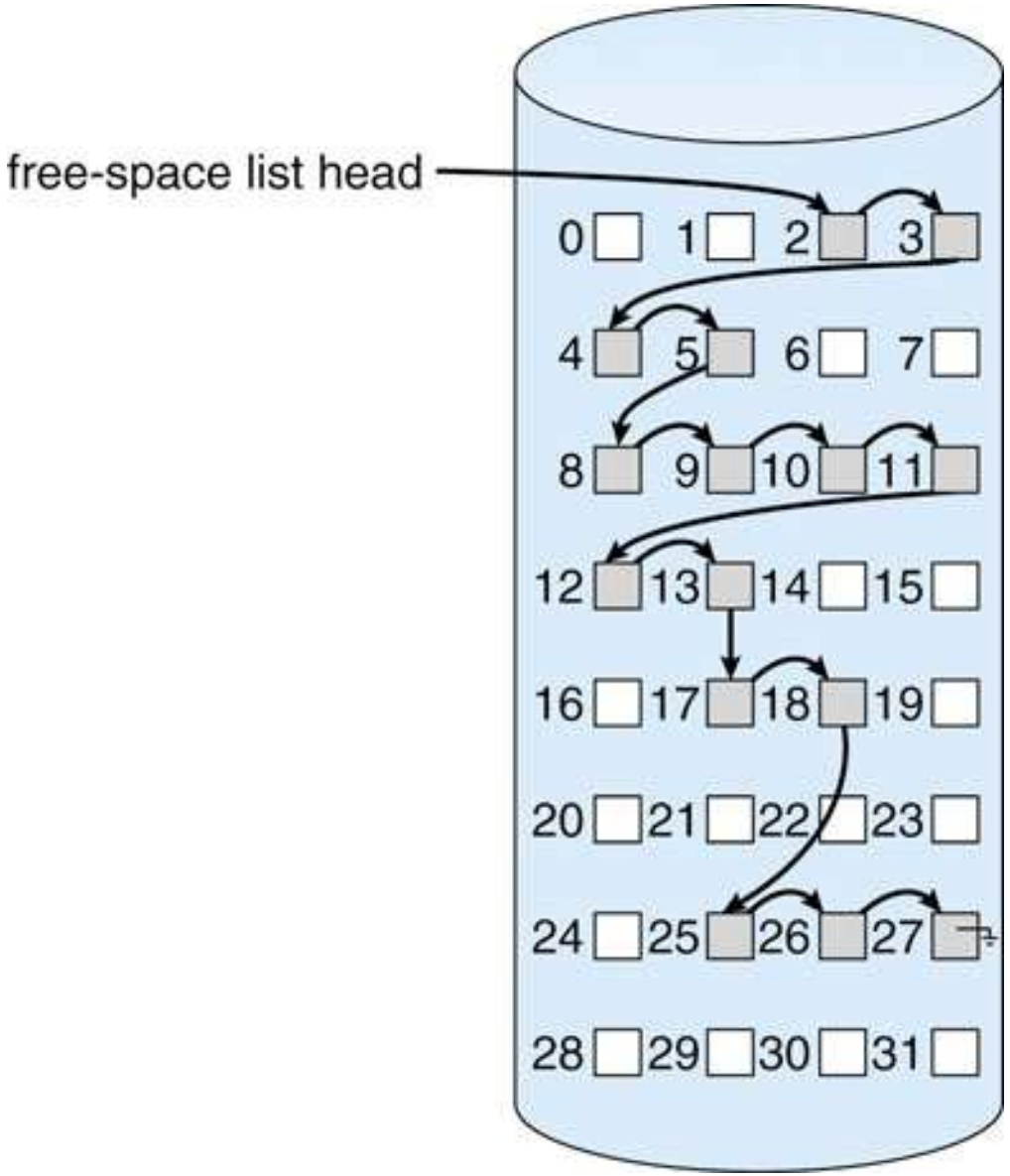
Schemat alokacji w systemie UNIX



Zarządzanie wolną przestrzenią: mapa bitowa

Dodatkowym 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 metoda jest prosta w implementacji i pozwala łatwo znajdować ciągłe bloki wolnej przestrzeni o wymaganej wielkości.

Zarządzanie wolną przestrzenią: lista wolnych bloków



Krótkie podsumowanie — pytania sprawdzające

1. Jakie są możliwe metody implementacji katalogów?

2. Czy wiesz jaką metodą zaimplementowane są katalogi w jakimś znanym Ci systemie?

3. Jakie są wady i zalety metod ciągłej i listowej alokacji bloków dla plików?

4. Jakie są wady i zalety metody indeksowanej alokacji bloków dla plików?

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