Implementacja Systemu plików

Implementacja System plików

- Struktura Systemu plików
- Implementacja systemu plików
- Implementacja kartotek
- Metody przydziału pamięci dyskowj
- Zarządzanie wolną przestrzenią pamięci

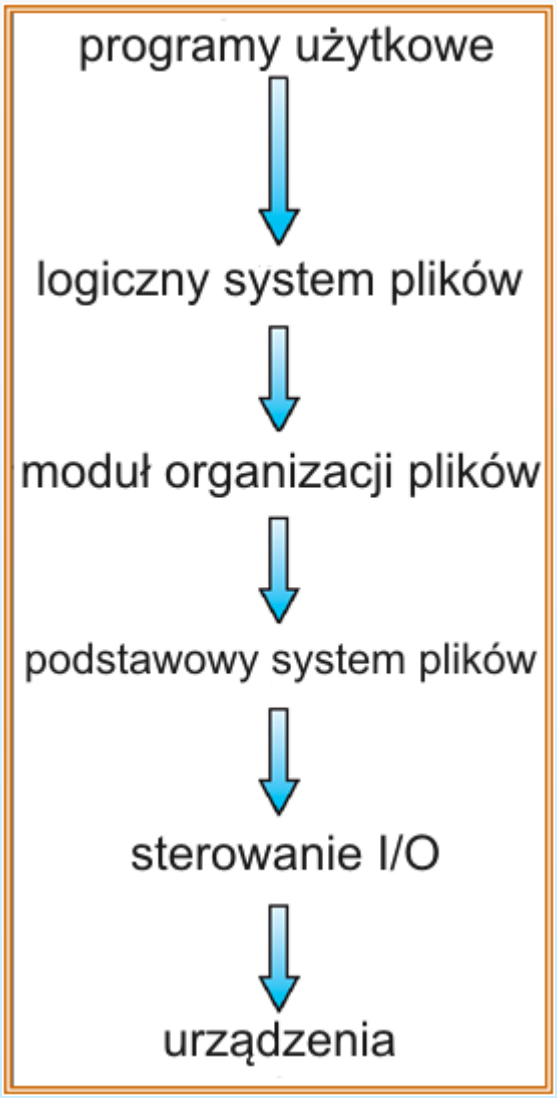
Tematyka wykładu

- Przedstawić implementację lokalnych systemów plików oraz struktur y katalogów
- Omówić algorytmy przydziału pamięci dyskowej

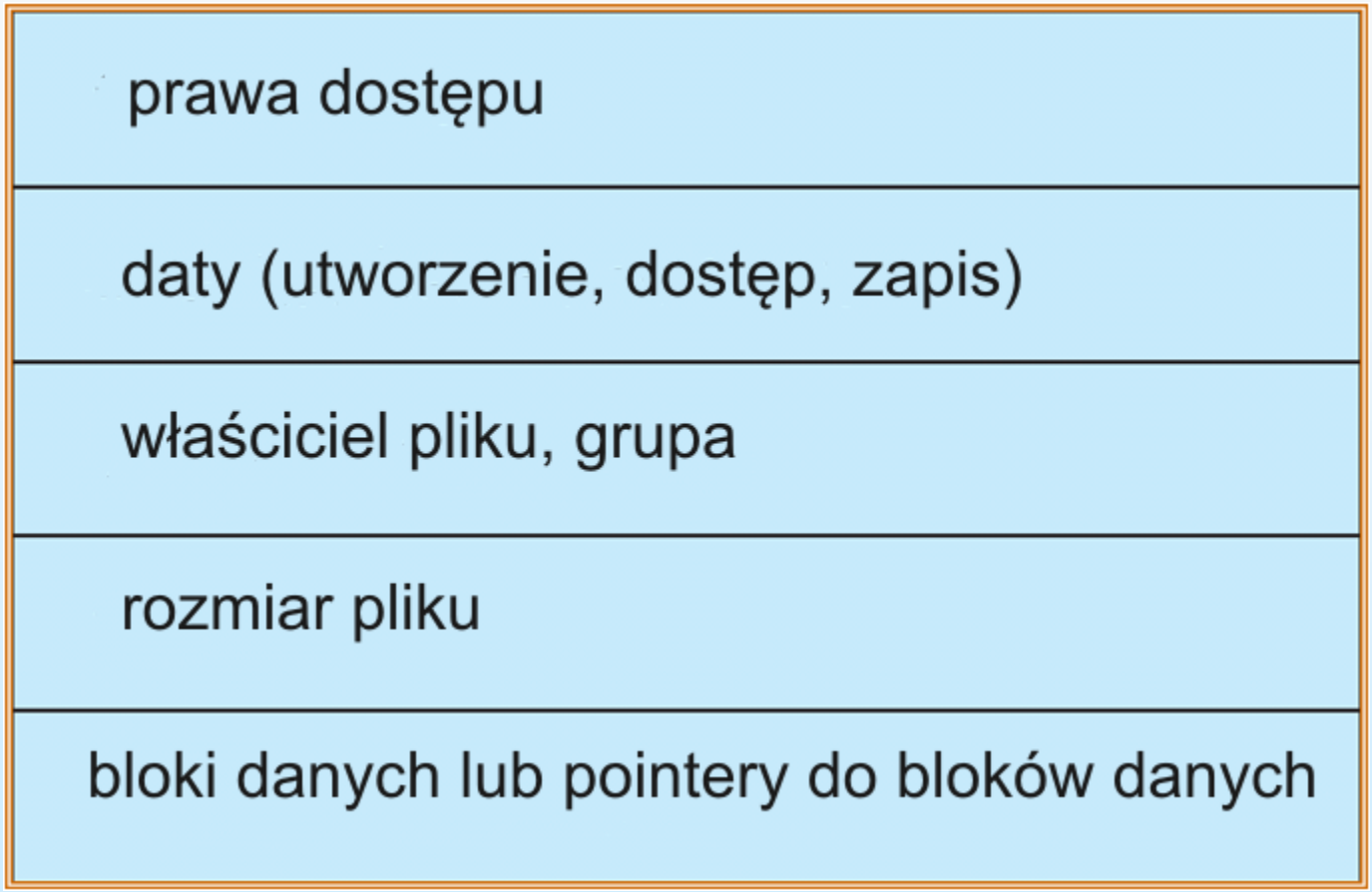
Struktura systemu plików

- Struktura pliku
  - Logiczna jednostka pamięci
  - Kolekcja powiązanej ze sobą informacji
- System plików rezyduje w pamięci zewnętrznej (dyski)
- Warstwowy system plików
- **Blok kontrolny pliku** – struktura zawierająca informację o pliku

Warstwowy system plików



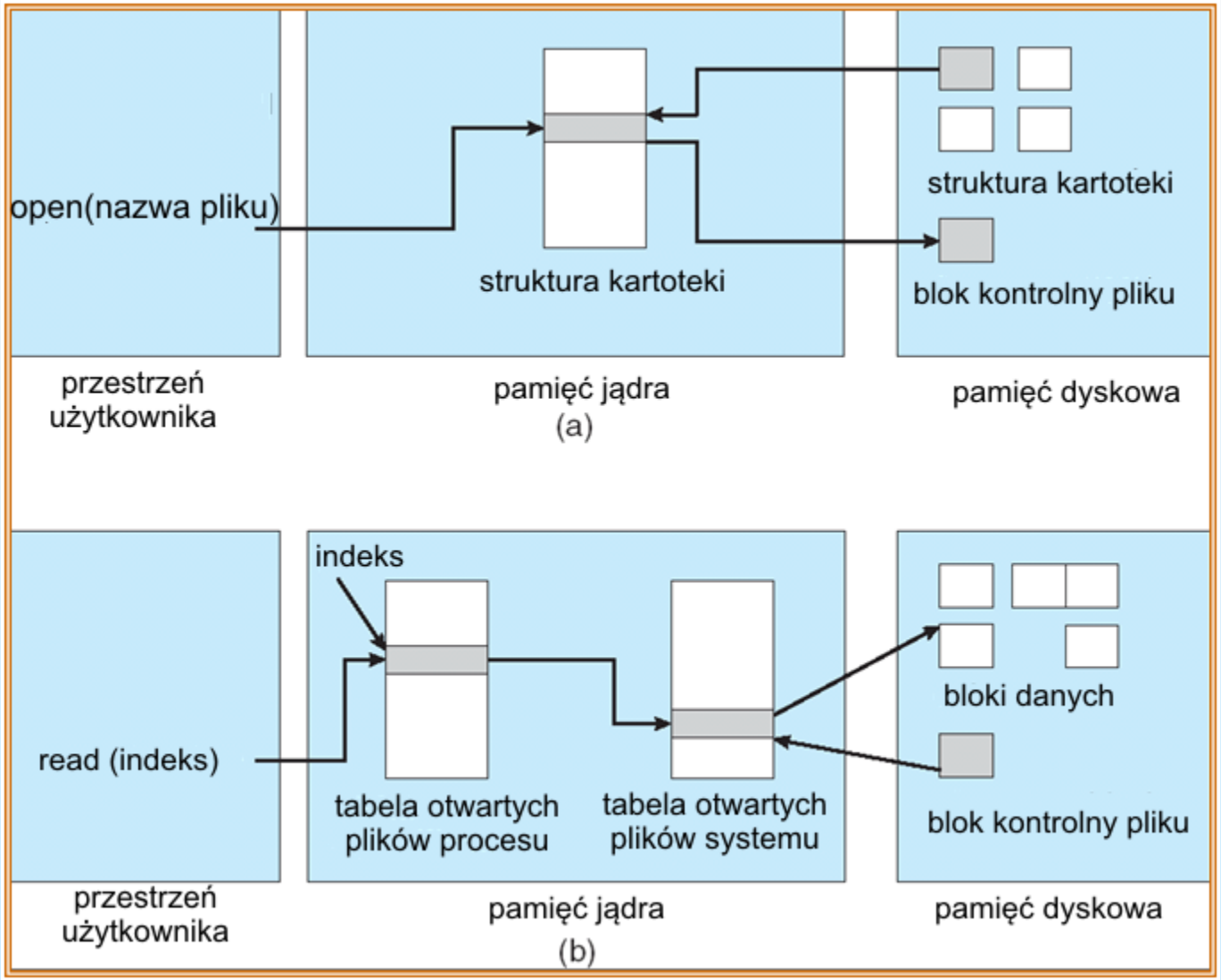
Typowy Blok kontrolny pliku FCB



Struktury danych systemu plików

- Następujące rysunki przedstawiają struktury danych systemu plików dostarczane przez systemy operacyjne.
- Rysunek 12-3(a) dotyczy otwierania pliku.
- Rysunek 12-3(b) dotyczy czytania pliku.

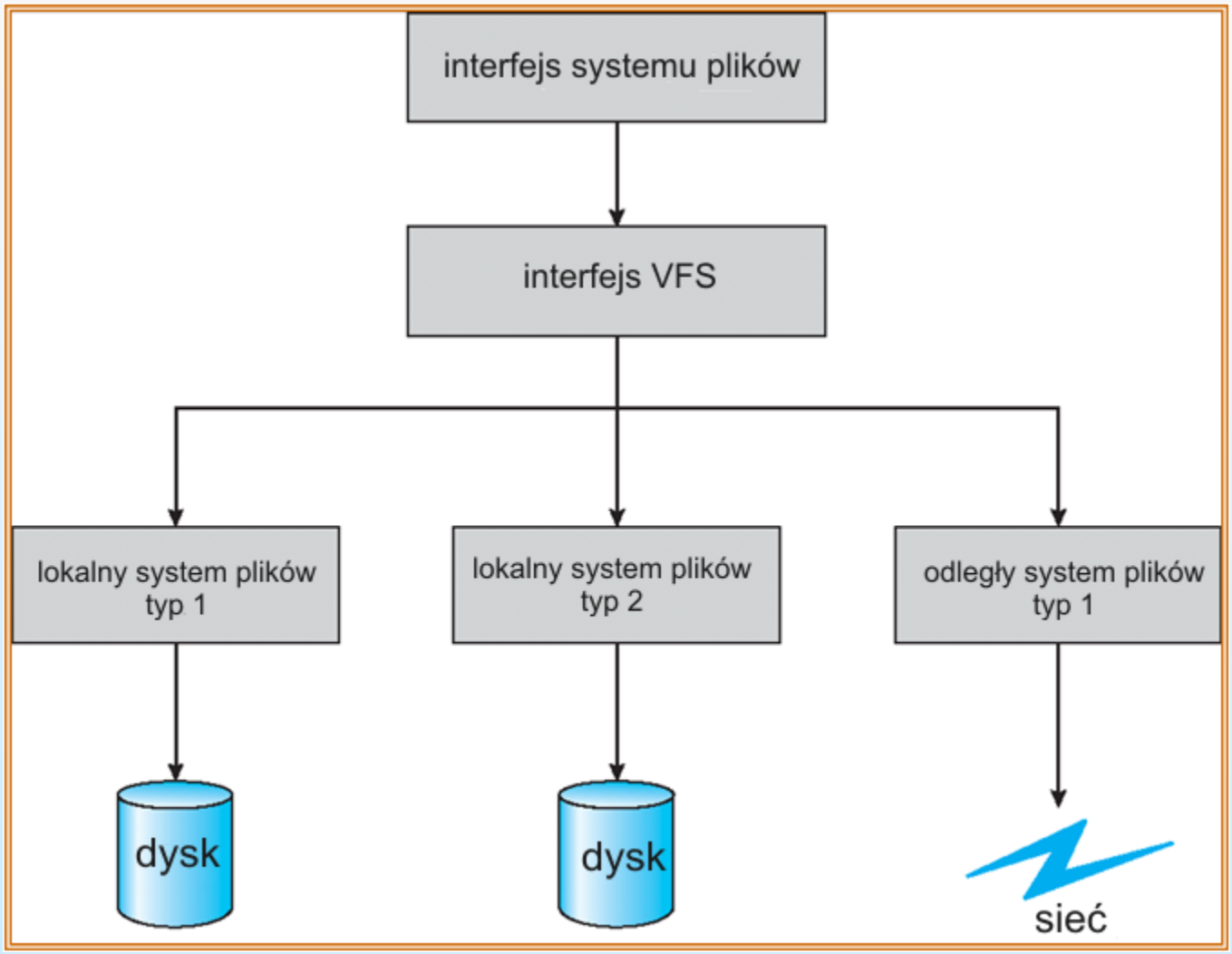
Struktury danych systemu plików (cd.)



Wirtualny system plików

- Wirtualne systemy plików (VFS) stanowią obiektowo zorientowany sposób implementacji systemów plików.
- VFS umożliwia aby taki sam interfejs odwołań do systemu (API – *Application Programming Interface*)) mógł być używany do różnych typów systemów plików.
- API jest interfejsem do VFS a nie do konkretnego typu systemu plików.

Ogólna struktura Wirtualnego systemu plików



Implementacja kartoteki

- **Liniowa lista** nazw plików z pointerami do bloków danych.
  - prostota programowania
  - Czasochłonność działania

- **Tablica haszująca** – liniowa lista z haszującą strukturą danych.
  - Zmniejsza czas przeszukiwania kartoteki
  - **kolizje** – sytuacje w których dwie nazwy pliku odnoszą się do tej samej lokacji
  - Stały rozmiar

Metody przydziału miejsca na dysku

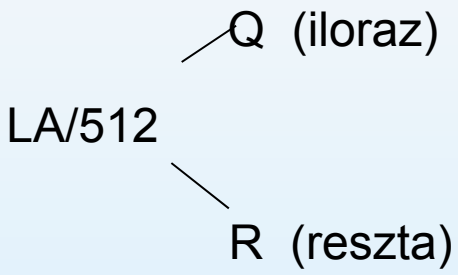
- Metody przydziału dotyczą sposobu w jaki bloki dyskowe przydzielane są plikom:
- **Przydział ciągły**
- **Przydział listowy**
- **Przydział indeksowy**

Przydział ciągły

- Każdy plik składa się z fciągłych (sąsiednich) bloków dyskowych
- Prosta metoda – wymagana jest znajomość tylko początkowego bloku (blok #) oraz długość (liczba bloków)
- Dostęp bezpośredni
- Strata przestrzeni pamięci (problem dynamicznego przydziału pamięci)
- Trudność zwiększania objętości plików

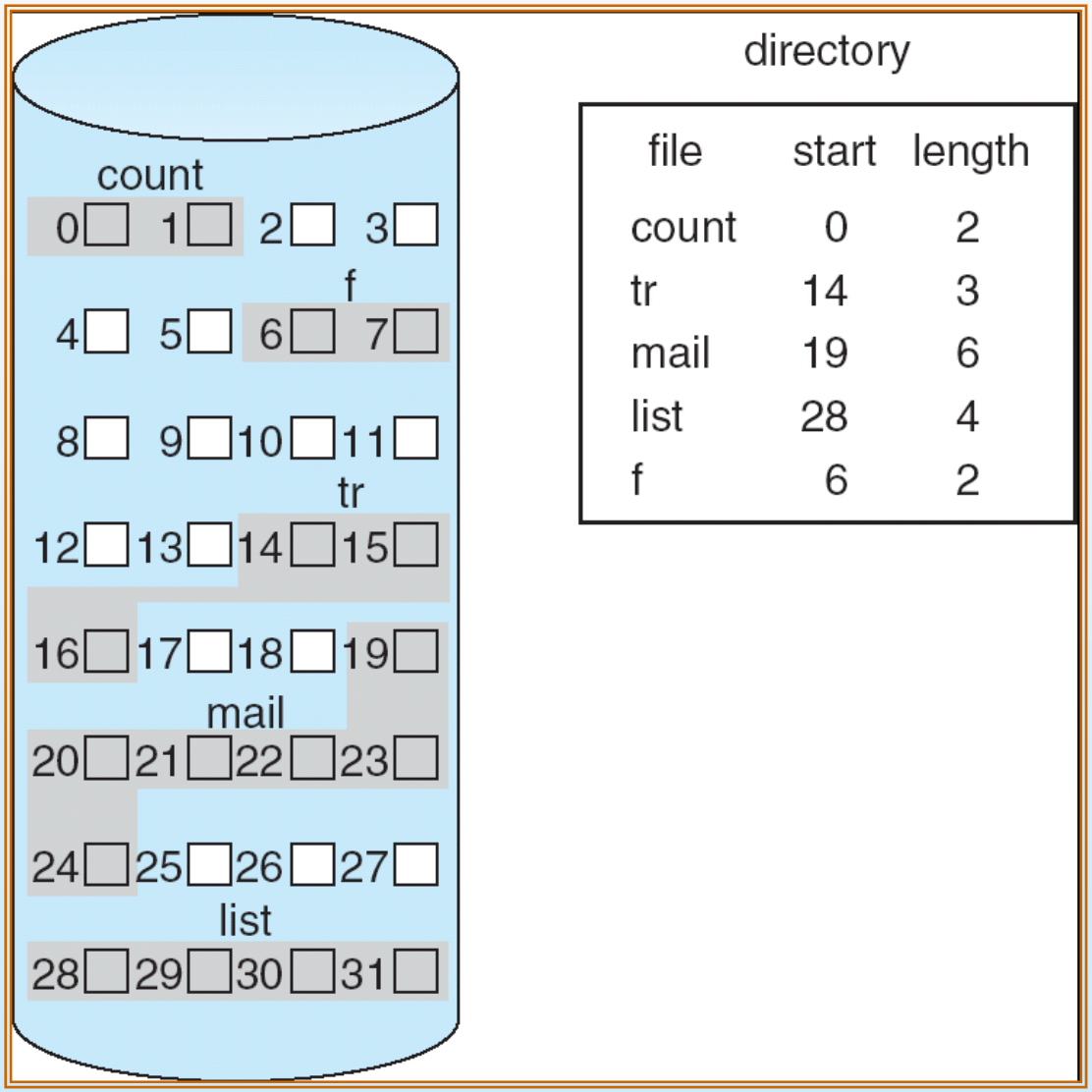
Przydział ciągły (cd.)

- Mapowanie z logicznego na fizyczny



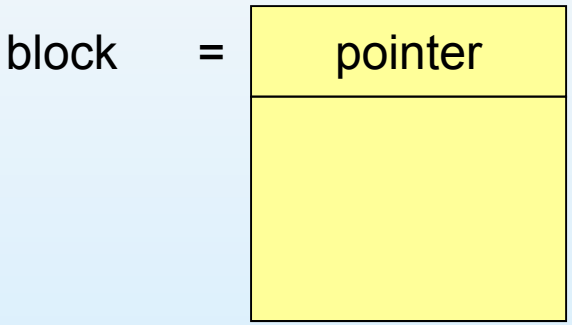
Blok dostępny = Q + adres początkowy
Przesunięcie w bloku = R

Przydział ciągły pamięci dyskowej



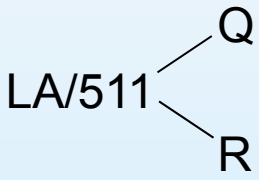
Przydział listowy

- Każdy plik jest listą powiązanych ze sobą bloków dyskowych.:bloki mogą być rozmieszczone dowolnie na dysku..



Przydział listowy (cd.)

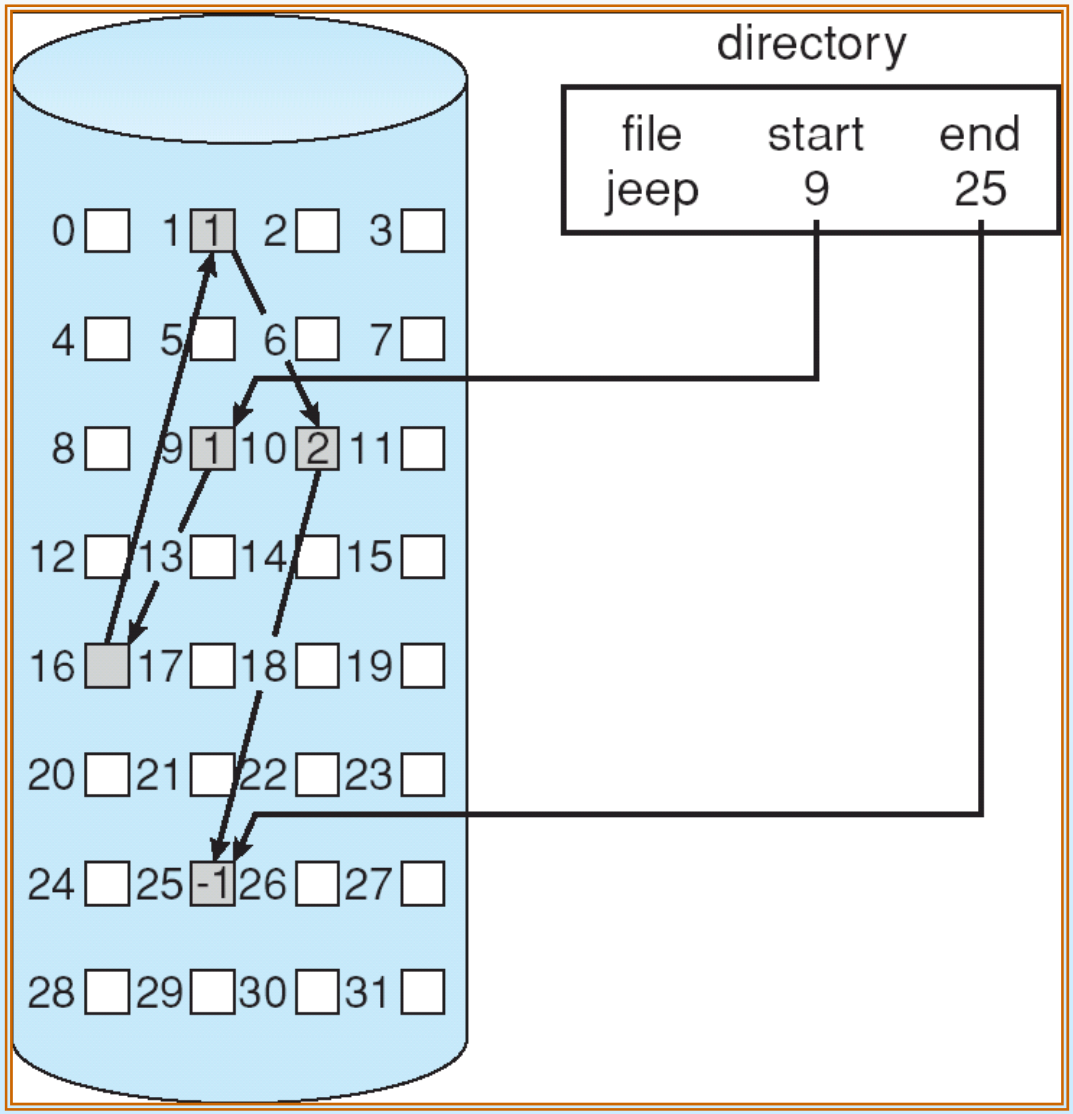
- Prostota – wymaga tylko adresu początkowego
- Free-space management system – no waste of space
- No random access
- Mapping



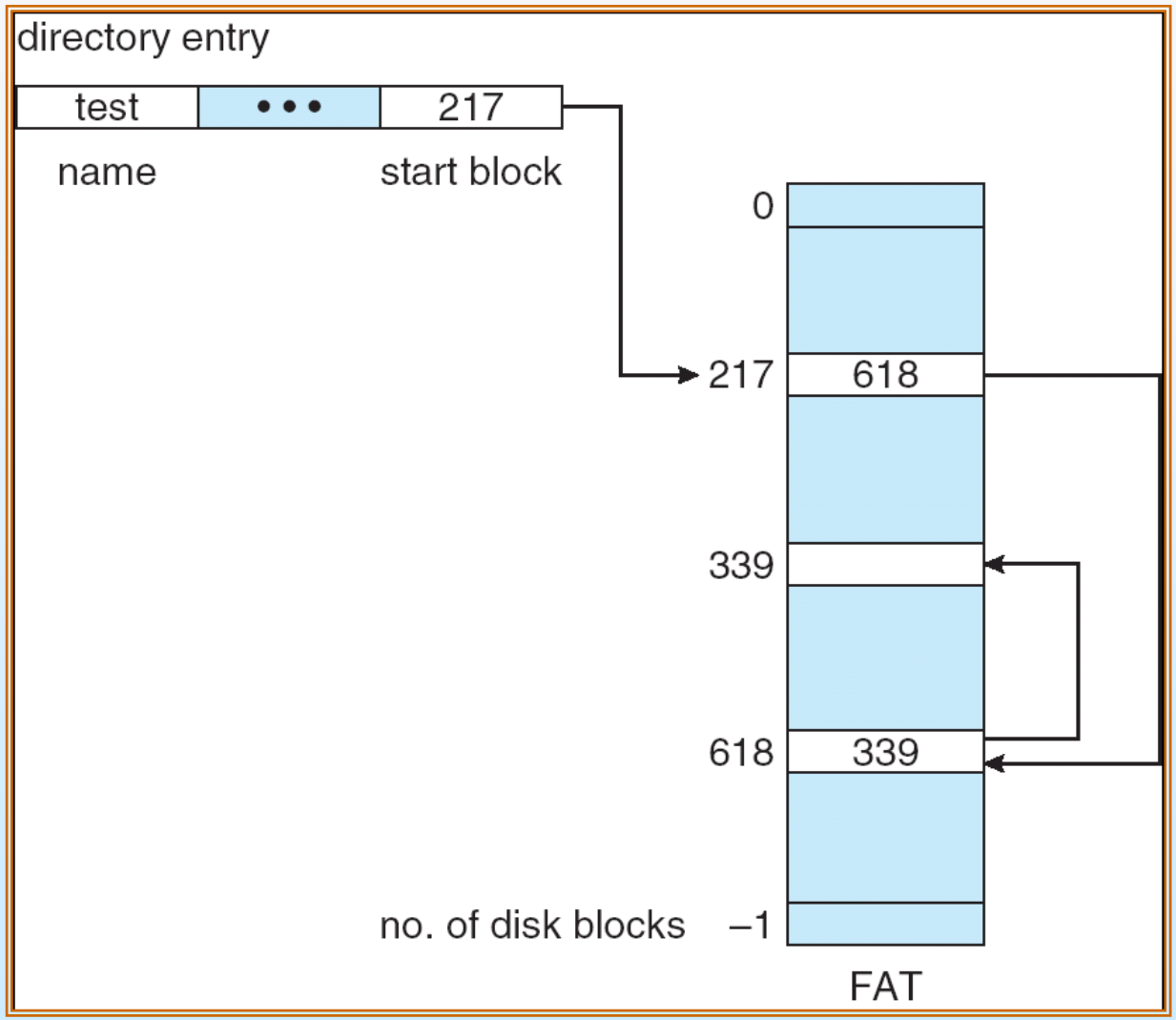
Q jest blokiem który ma być dostępny w liście bloków reprezentującej plik.
Przesunięcie w bloku = R + 1

File-allocation table (FAT) – przydział pamięci stosowany w systemach MS-DOS, MS-Windows oraz OS/2.

Przydział listowy (cd.)



Tablica alokacji FAT



Przydział indeksowy

- Wszystkie wskaźniki zgromadzone są w *bloku indeksowym.*
- Logical view.

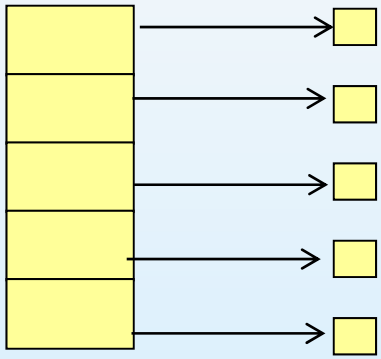
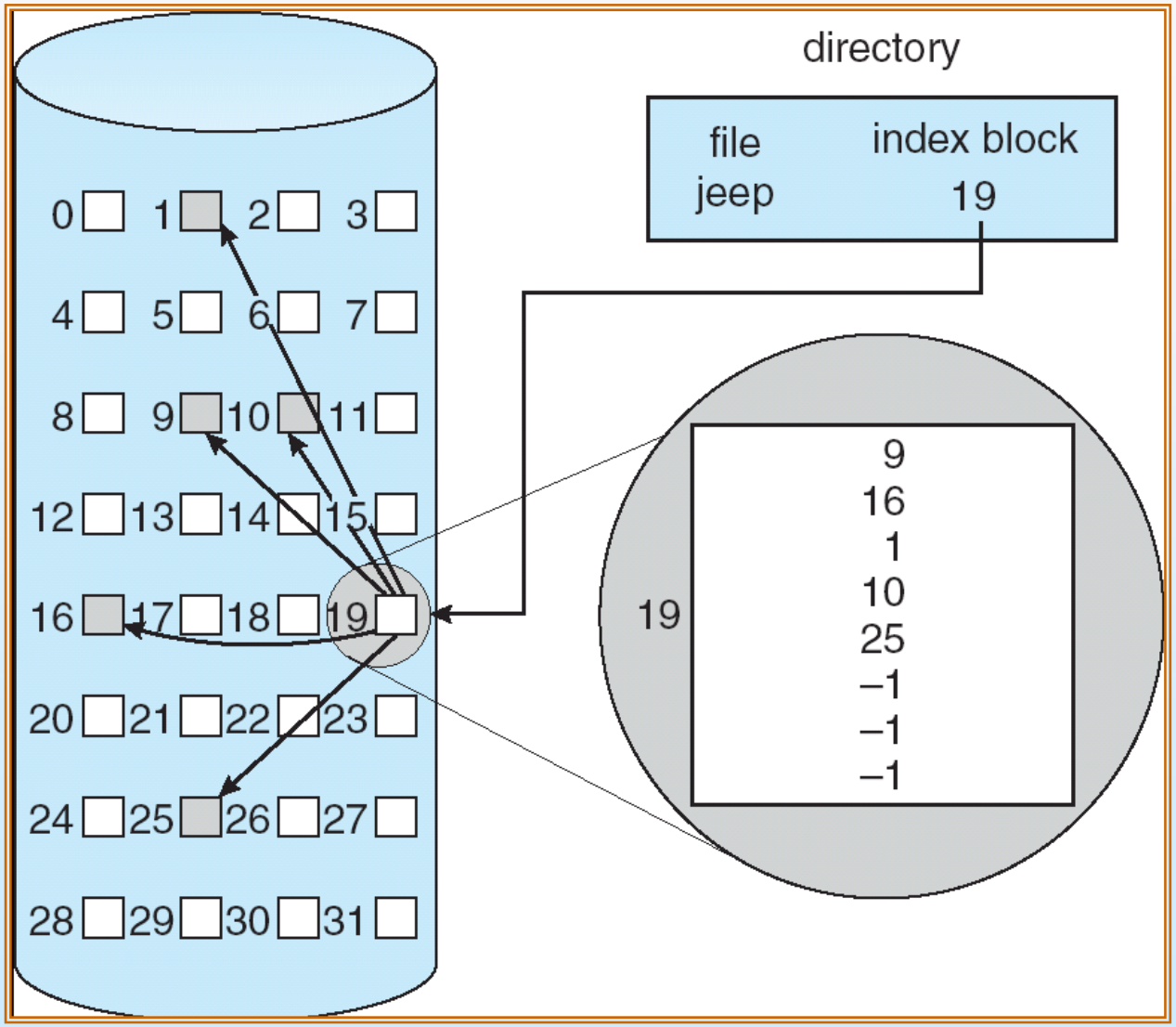


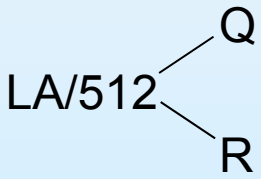
Tabela indeksów

Przykład przydziału indeksowego



Przydział indeksowy (cd.)

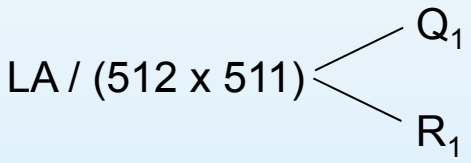
- Wymagana tabela indeksów
- Dostęp bezpośredni
- Dynamiczny dostęp bez zewnętrznej fragmentacji ale występuje narzut bloków indeksów.
- Mapowanie adresu logicznego na fizyczny w pliku o maksymalnym rozmiarze 256K słów i rozmiarze bloku 512 słów. Potrzebny jest tylko 1 blok na tabelę indeksów.



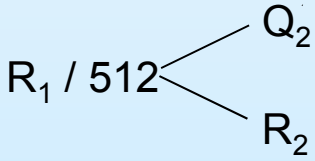
Q = przesunięcie w tablicy indeksów
R = przesunięcie w bloku

Przydział indeksowy – Mapowanie (cd.)

- Mapowanie adresu logicznego na fizyczny w pliku o nieograniczonej długości (blok rozmiaru 512 słów).
- Schemat połączeń – Lista bloków indeksów (bez ograniczenia długości).



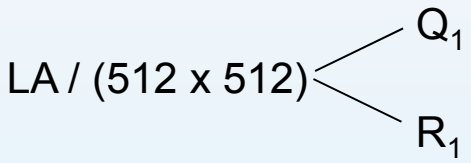
Q_1 = blok indeksów (tablica indeksów)
R_1 wykorzystany następująco:



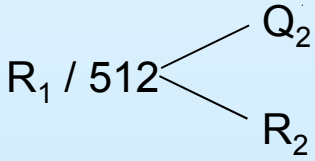
Q_2 = przesunięcie w bloku indeksów
R_2 przesunięcie w bloku danych:

Przydział indeksowy – mapowanie (cd.)

- Dwupoziomowy indeks (rozmiar maksymalnego pliku wynosi 512^3)

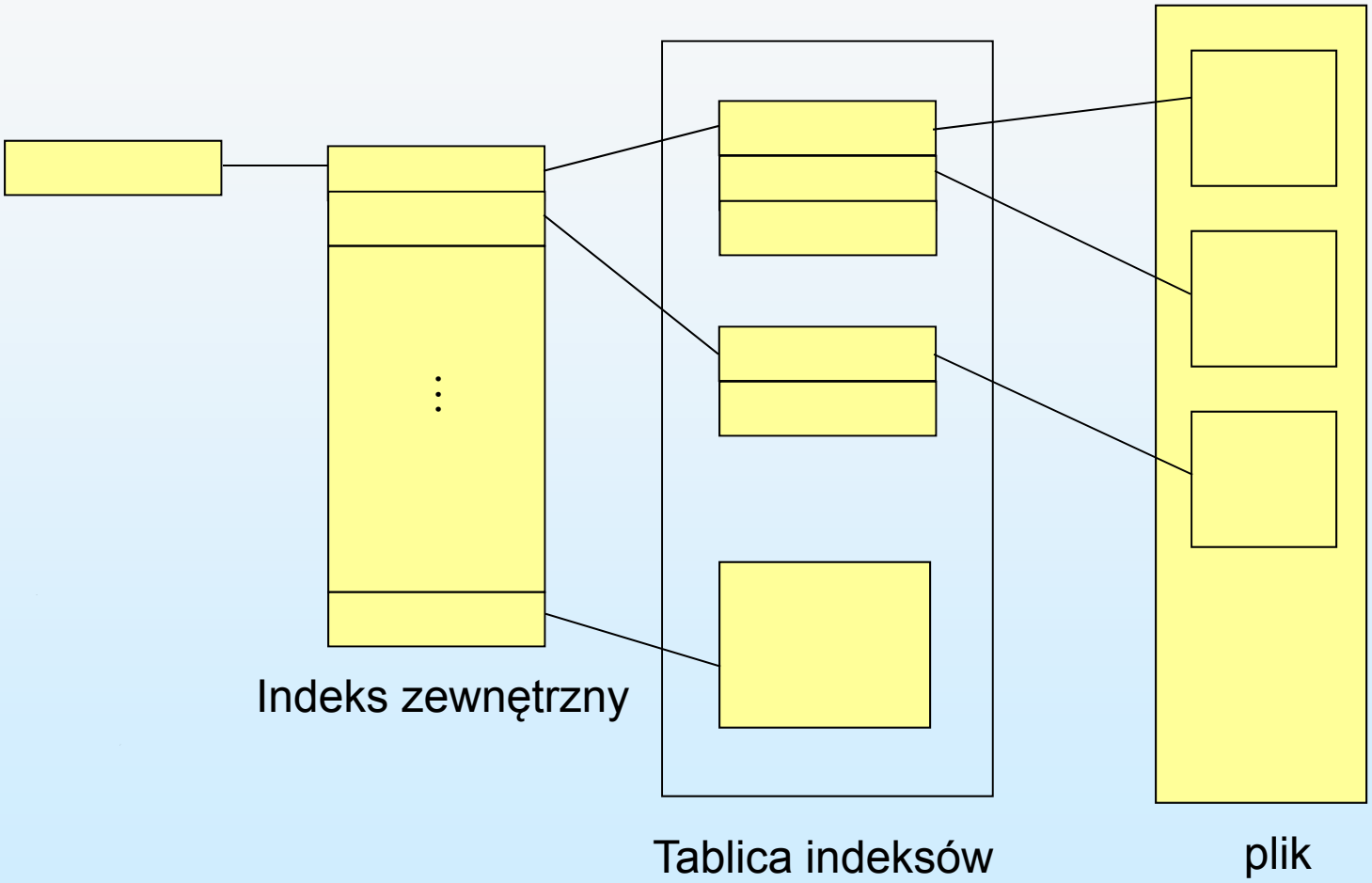


Q_1 = zastąpienie indeksem zewnętrznym
R_1 wykorzystany następująco:

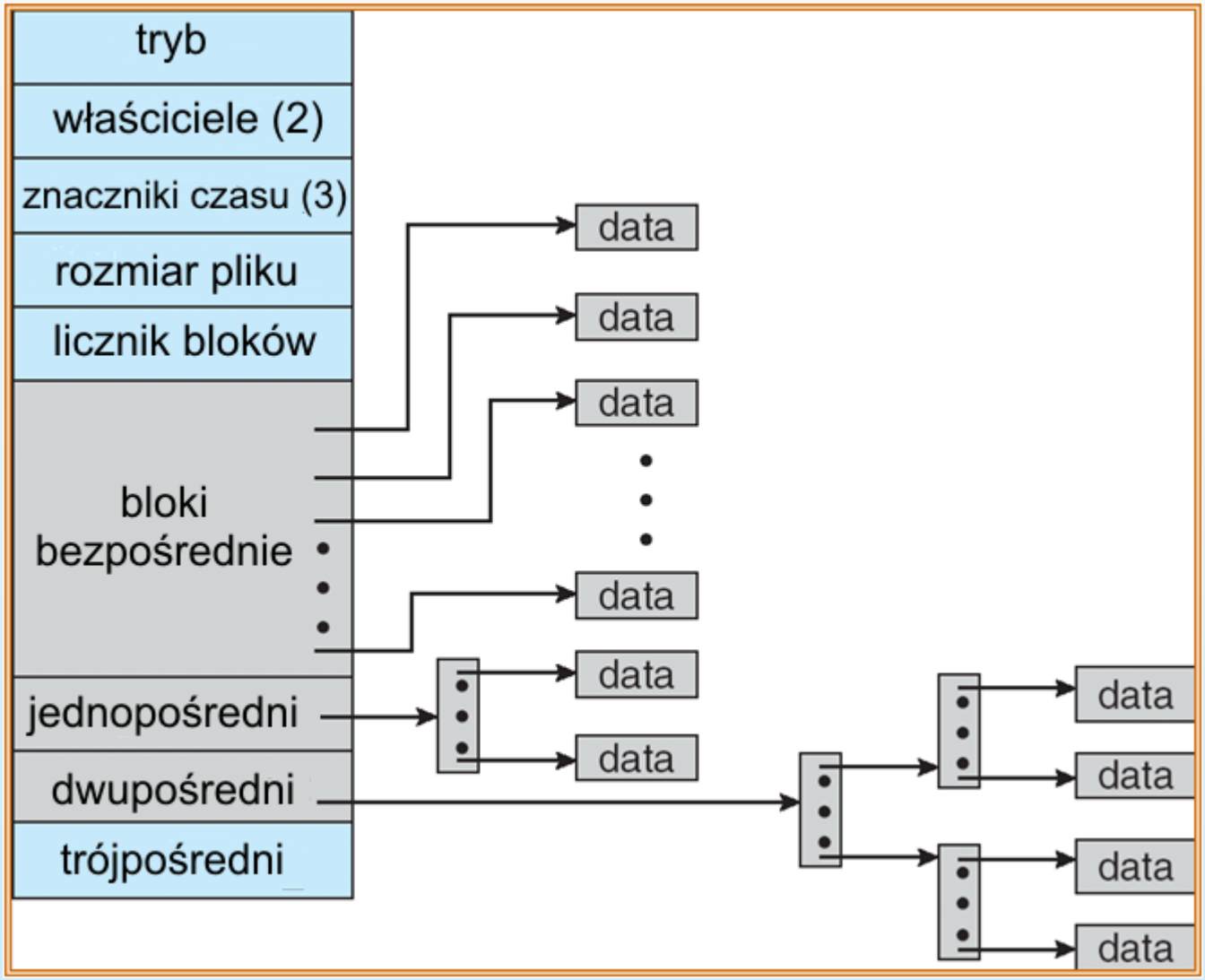


Q_2 = zastąpienie blokiem indeksów
R_2 zastąpienie blokiem danych:

Przydział indeksowy – mapowanie (cd.)

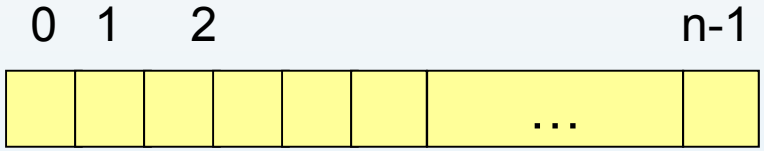


Schemat kombinowany (struktura i-węzła)



Zarządzanie wolną przestrzenią

- Wektor bitowy (*n* bloków)



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{blok}[i] \text{ wolny} \\ 1 \Rightarrow \text{bloc}[i] \text{ zajęty} \end{cases}$$

Obliczanie numeru bloku

(liczba bitów w słowie) *
(liczba wyzerowanych słów) +
Pozycja pierwszego bitu 1 w niezerowym słowie

Zarządzanie wolną przestrzenią (cd.)

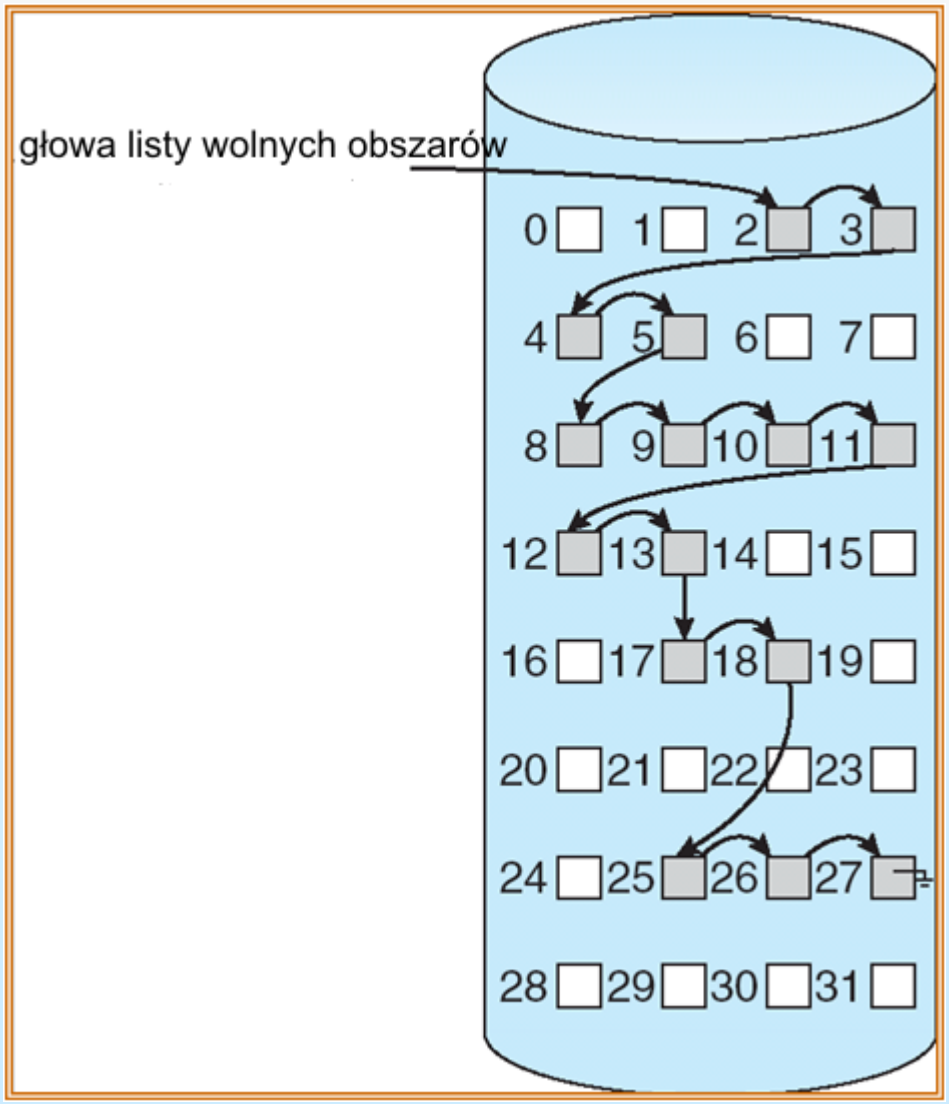
- Mapa bitowa wymaga dodatkowej przestrzeni
  - Przykład:

    rozmiar bloku = 2^{12} bajtów

    rozmiar dysku = 2^{30} bajtów (1 gigabajt)

    $n = 2^{30}/2^{12} = 2^{18}$ bitów (lub 32K bajtów)
- Łatwy sposób w przypadku przydziału ciągłego
- Lista bloków wolnych
  - Niemożność tworzenia obszarów ciągłych
  - Brak straty przestrzeni pamięci
- Grupowanie
- Zliczanie

Lista wolnych bloków na dysku



Zarządzanie wolną przestrzenią (cd.)

- Potrzeba ochrony:
  - Pointer do listy wolnych bloków
  - Mapa bitowa
    - Musi być przechowywana na dysku
    - Copia w pamięci i na dysku mogą się różnić
    - Nie można dopuścić aby dla blok [*i*] twystąpił taka sytuacja kiedy bit[*i*] = 1 w pamięci a biit[*i*] = 0 na dysku
  - Rozwiązanie:
    - Ustaw bit[*i*] = 1 na dysku
    - Przydziel blok[*i*]
    - Ustaw bit[*i*] = 1 w pamięci

Koniec