Ch09 dan Ch10
Konsep dan Implementasi File Systems

IKI 20250 – Sistem Operasi
Fakultas Ilmu Komputer UI

Revisi: 16 April 2011

Tujuan Pembelajaran

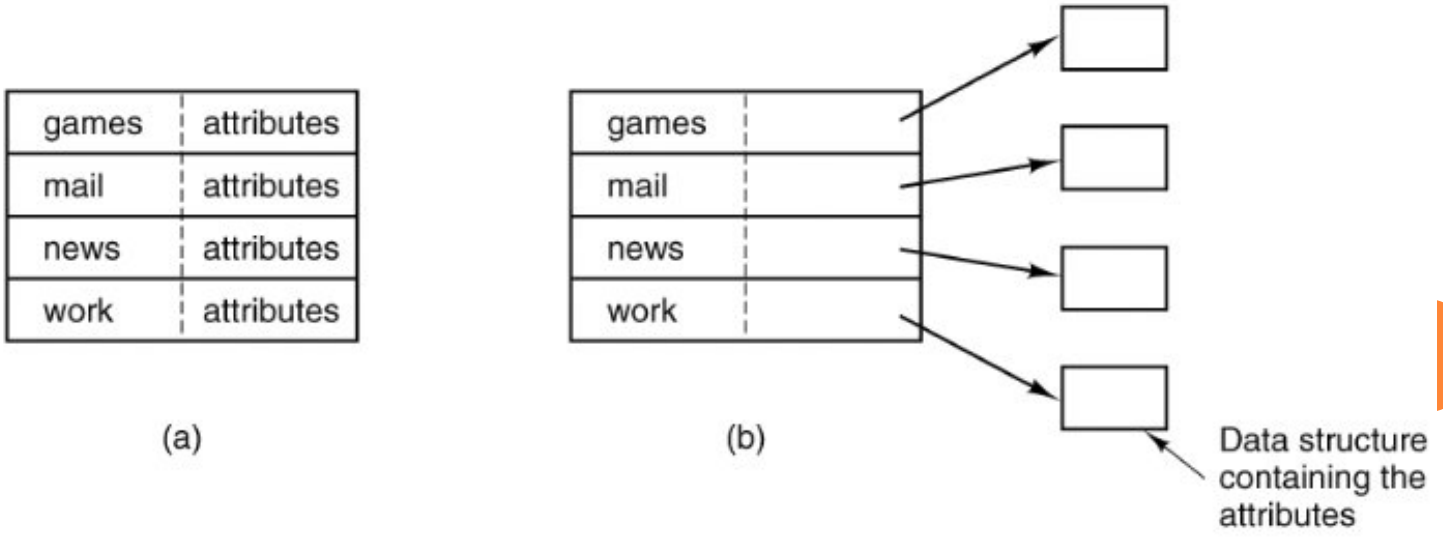
- Memahami konsep dan implementasi file, direktori dan filesystems
- Memahami metode alokasi blok untuk file
- Memahami mekanisme pengaturan blok kosong

File

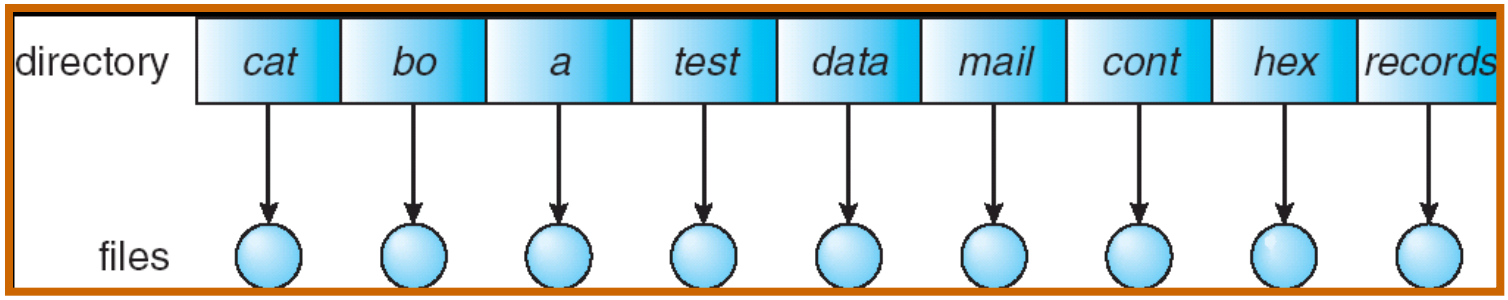
- File – unit informasi yang disimpan pada penyimpanan permanen
  - File Attribute -- informasi tentang file (metadata)
    - Name, identifier, type, location, size, protection, time, date, owner dan lainnya
  - Metada file disimpan pada sebuah FCB (File Control Block) atau inode (UNIX)
- File-system – manajemen file pada disk
  - Mendukung operasi create, delete, open, close, read, write, append, dan lainnya

DIREKTORI

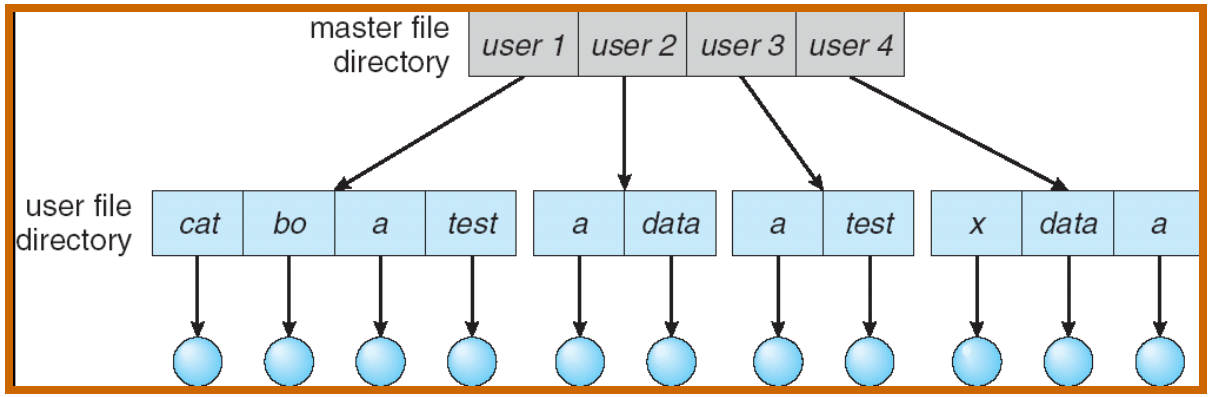
- Direktori/folder – menyimpan informasi file atau sub direktori yang tergabung didalamnya. Kemungkinan implementasi:
  - Gambar (a) : Direktori berisikan nama, atribut dan alamat file pada disk
  - Gambar (b) : Direktori berisikan nama dan pointer ke struktur data file (FCB/Inodes).



Single and Two Level Directory

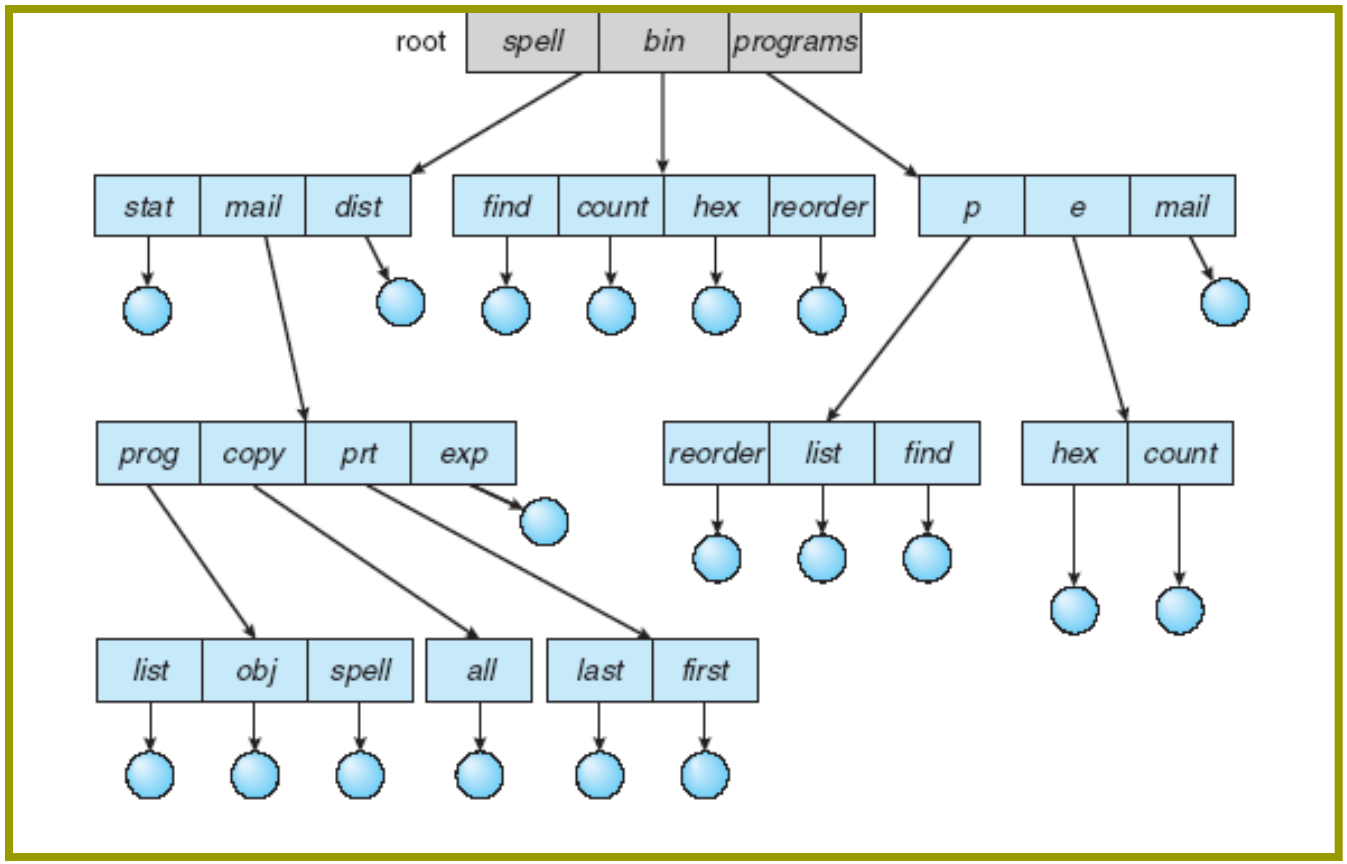


Single level directory



Two level directory

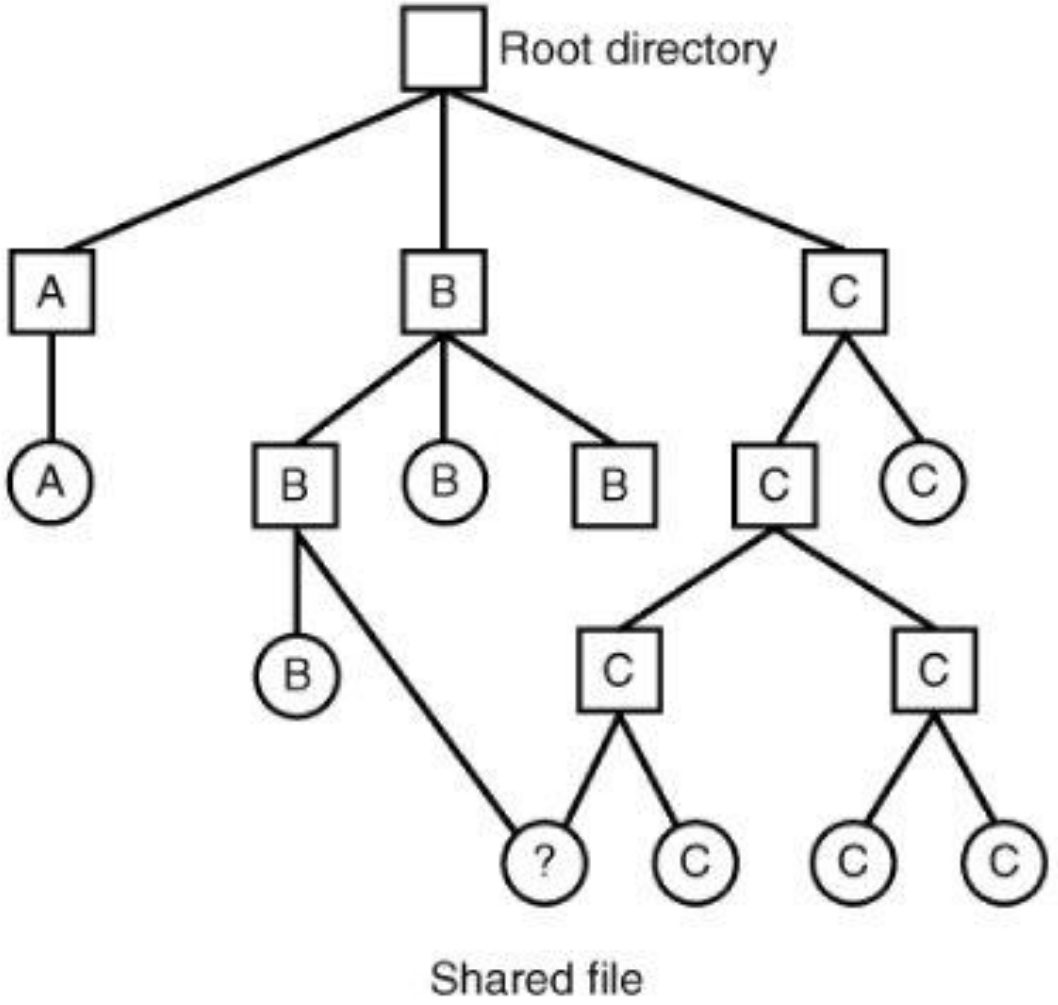
Tree-Structured Directory



Path

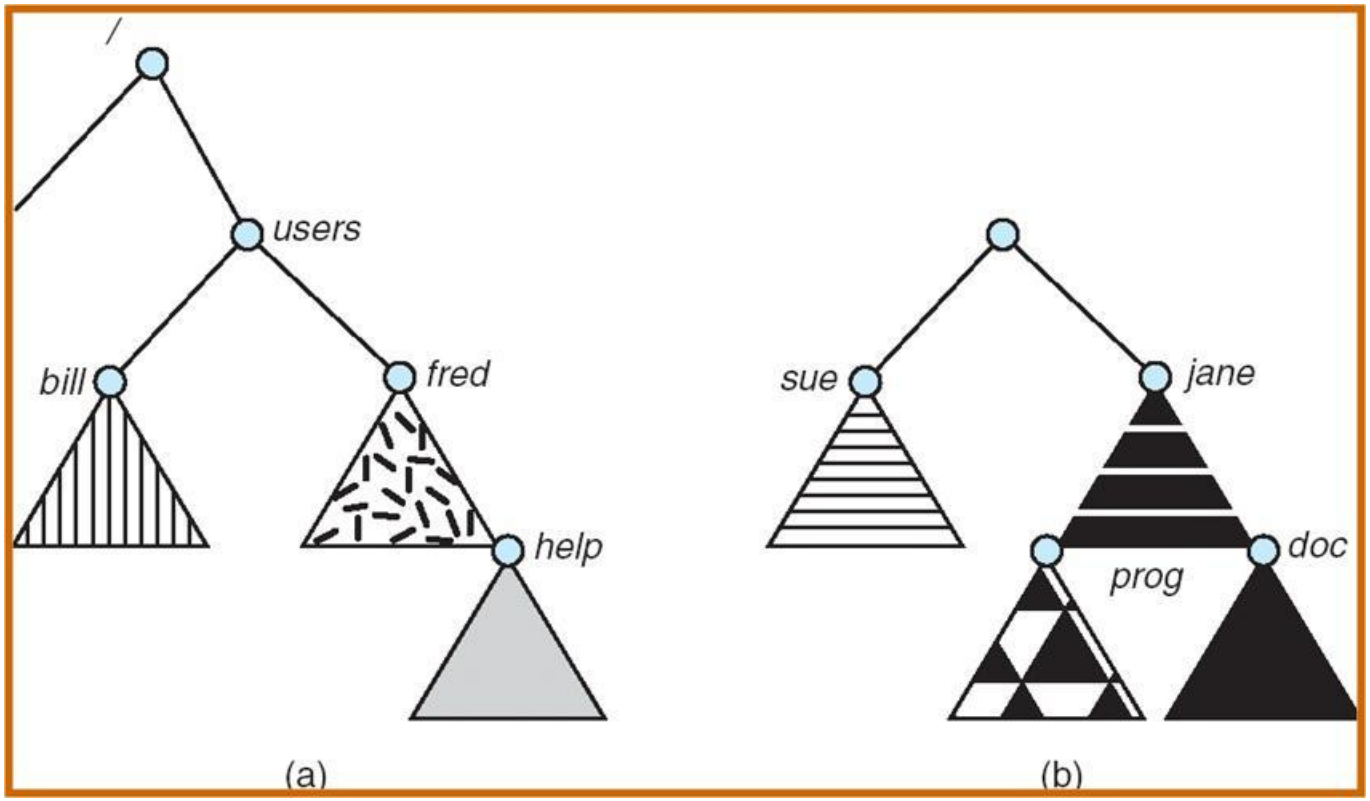
- Absolute Path : path dimulai dari direktori root (/) atau volume (C:\, D:\)
- Relative Path : path dimulai dari *current directory*

File Sharing



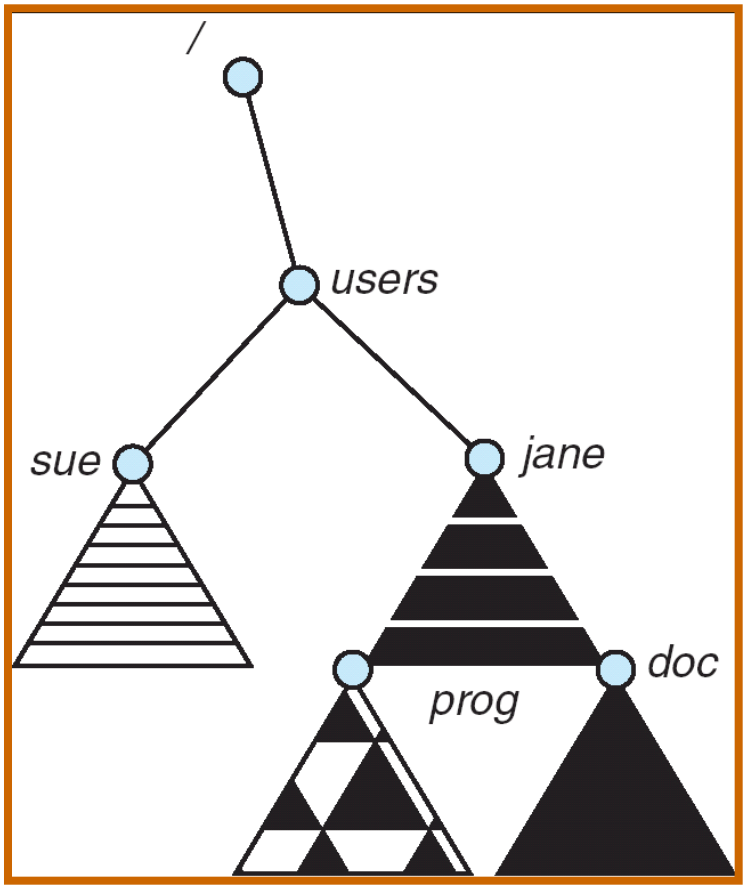
Shared file

FileSystem Mounting



a) Existing systems b) Unmounted volume

File System Mounting

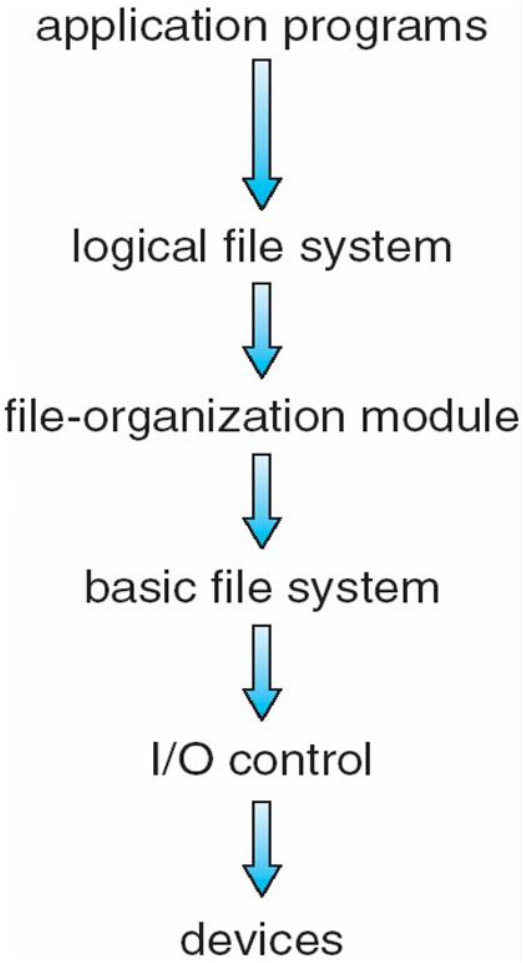


Mounted volume

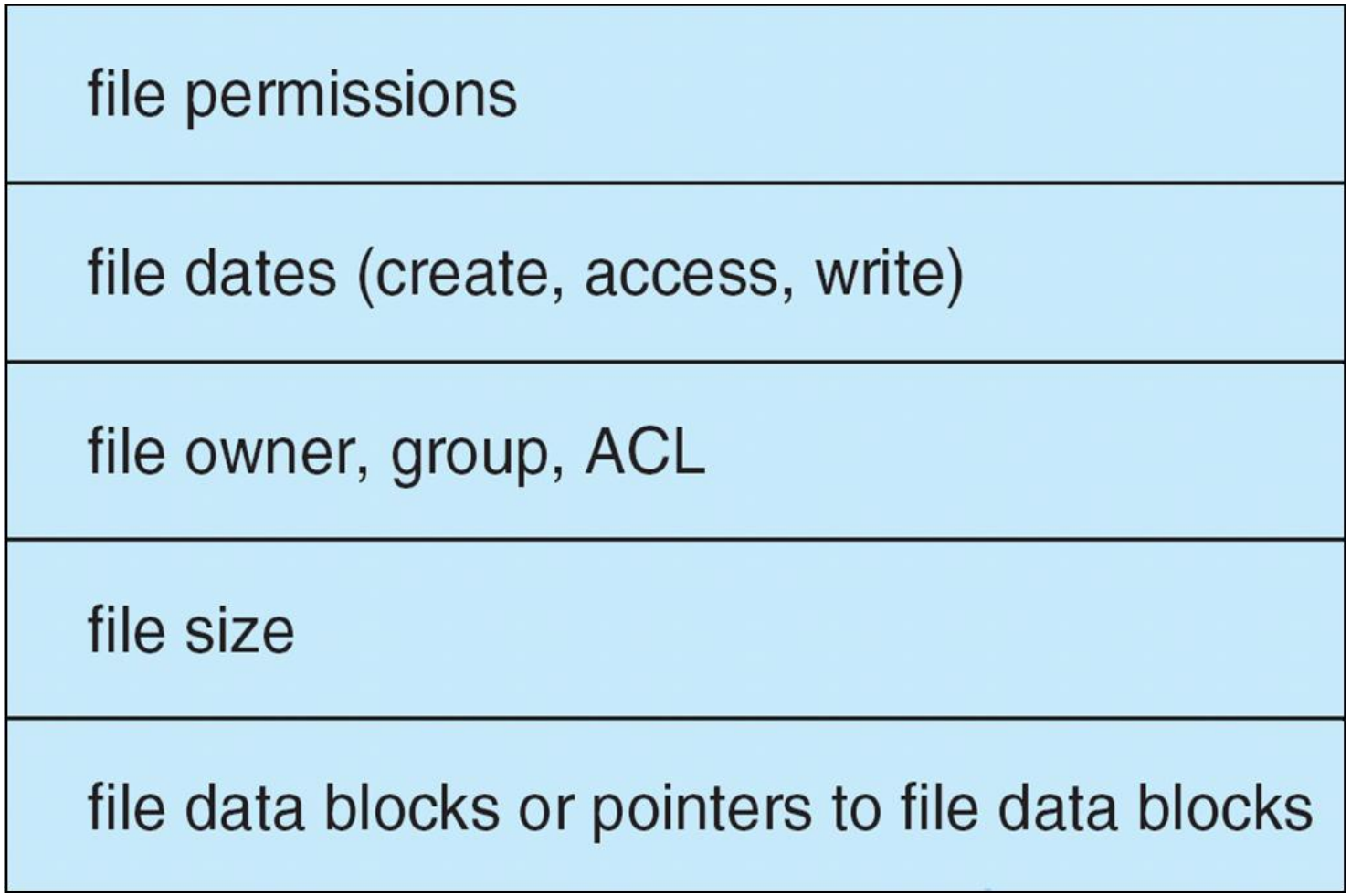
Struktur file system

- Blok – unit penyimpanan logic file
  - Besar 1 block = 1 sektor atau lebi
  - Ukuran 1 sektor =32 byte sampai 4 KiB, umumnya 512 byte
- **File system** memberikan kemudahan dan efisiensi pengaksesan, penyimpanan dan peraihan data pada disk.
- **File control block** – menyimpan struktur file-system sebuah file
- **Device driver** mengontrol penyimpanan fisik

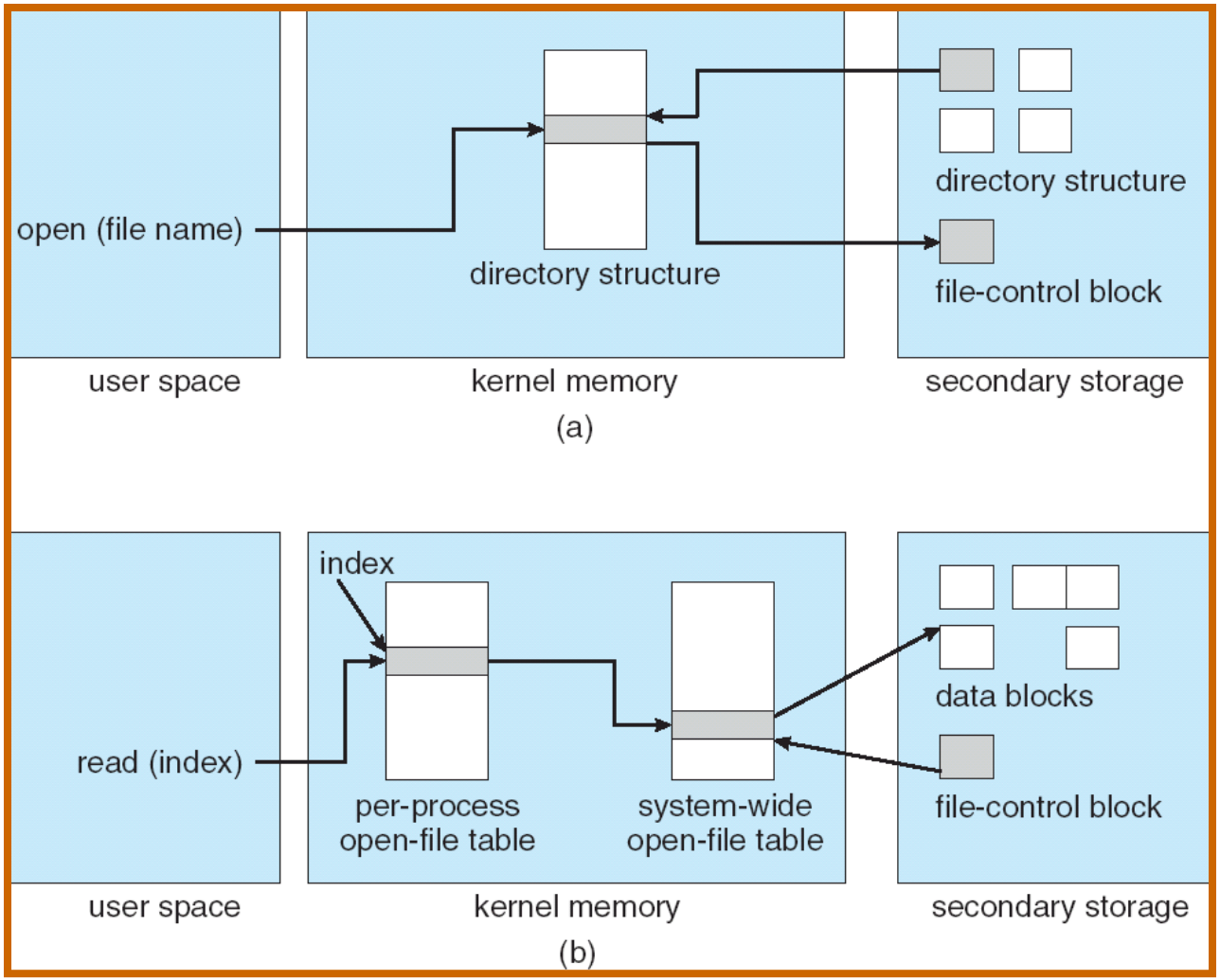
Lapisan File System



Layout Umum File Control Block

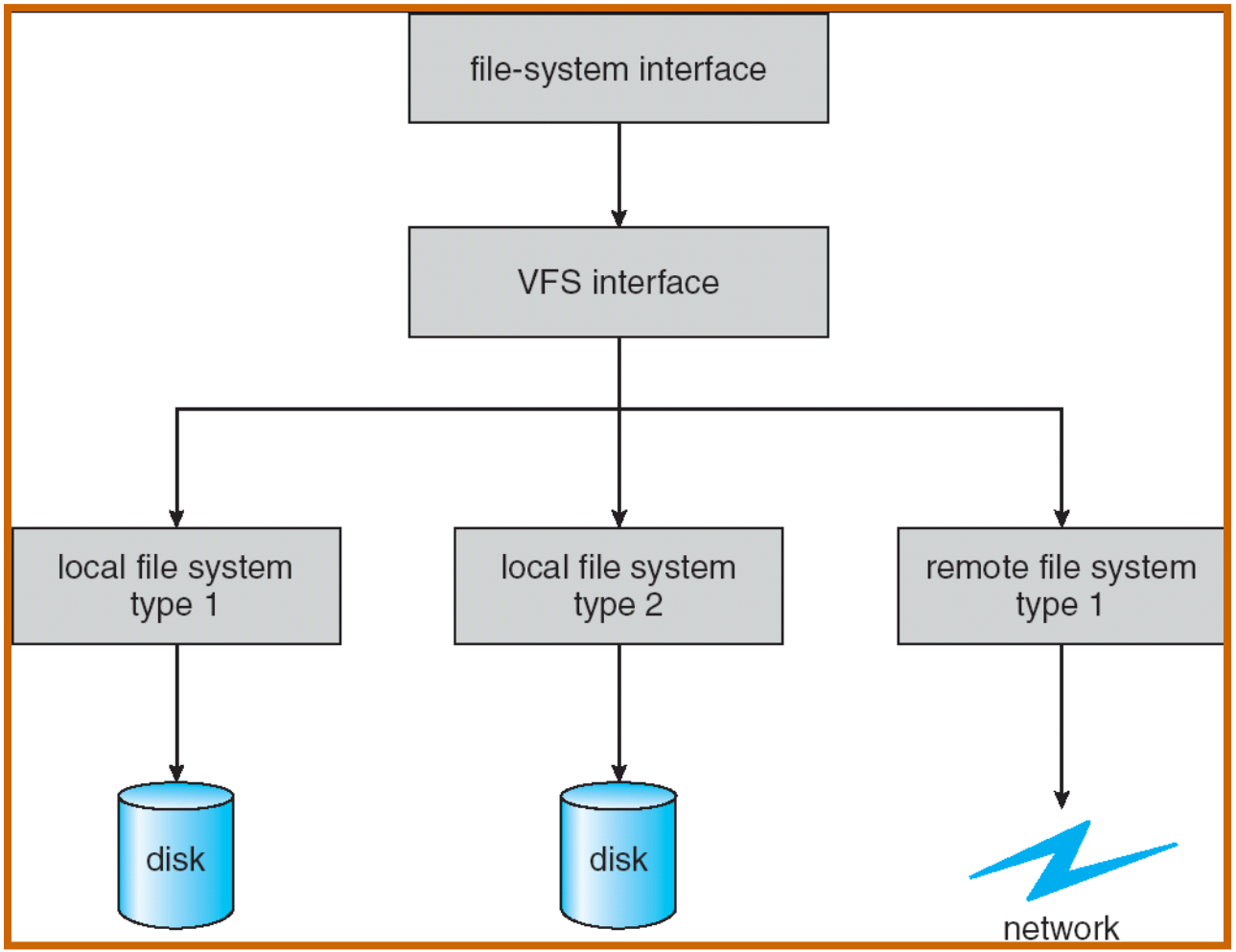


Struktur File System pada Memori



a) Buka File b) Baca File

Virtual File System



Akses File

- Sequential
  - Akses dari awal blok secara berurutan sampai ke blok terakhir yang ingin dibaca
  - Contoh: tape
- Direct/random
  - Akses blok secara acak.
  - Contoh: disk, CD/DVD

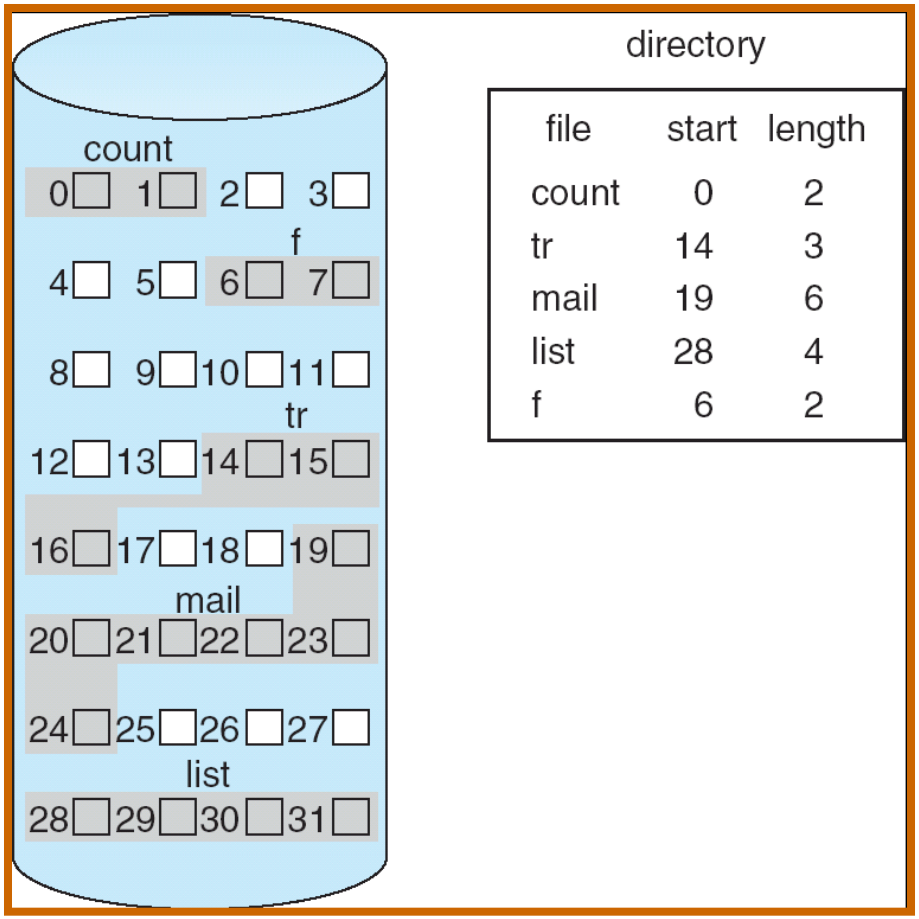
Metode Alokasi Blok untuk File

1. Contiguous allocation
2. Linked allocation
3. Indexed allocation

Contiguous Allocation (cont.)

- Setiap file menempati blok disk secara berurutan
- Keuntungan
  - Pergerakan head sedikit
  - Aksesnya mudah – hanya membutuhkan lokasi awal blok (b) dan panjang blok (n)
    - Mendukung Sequential dan Direct (random) Access
    - Lokasi blok : b, b+1, b+2, ..., b+n-1

Contiguous Allocation (cont.)

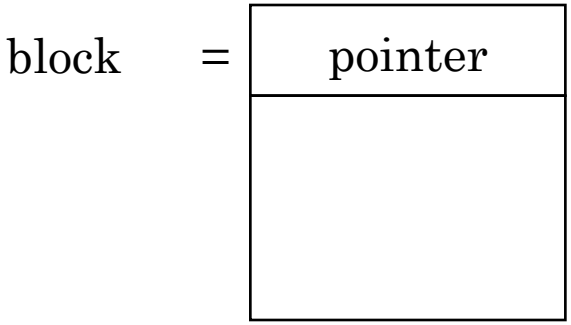


Contiguous Allocation

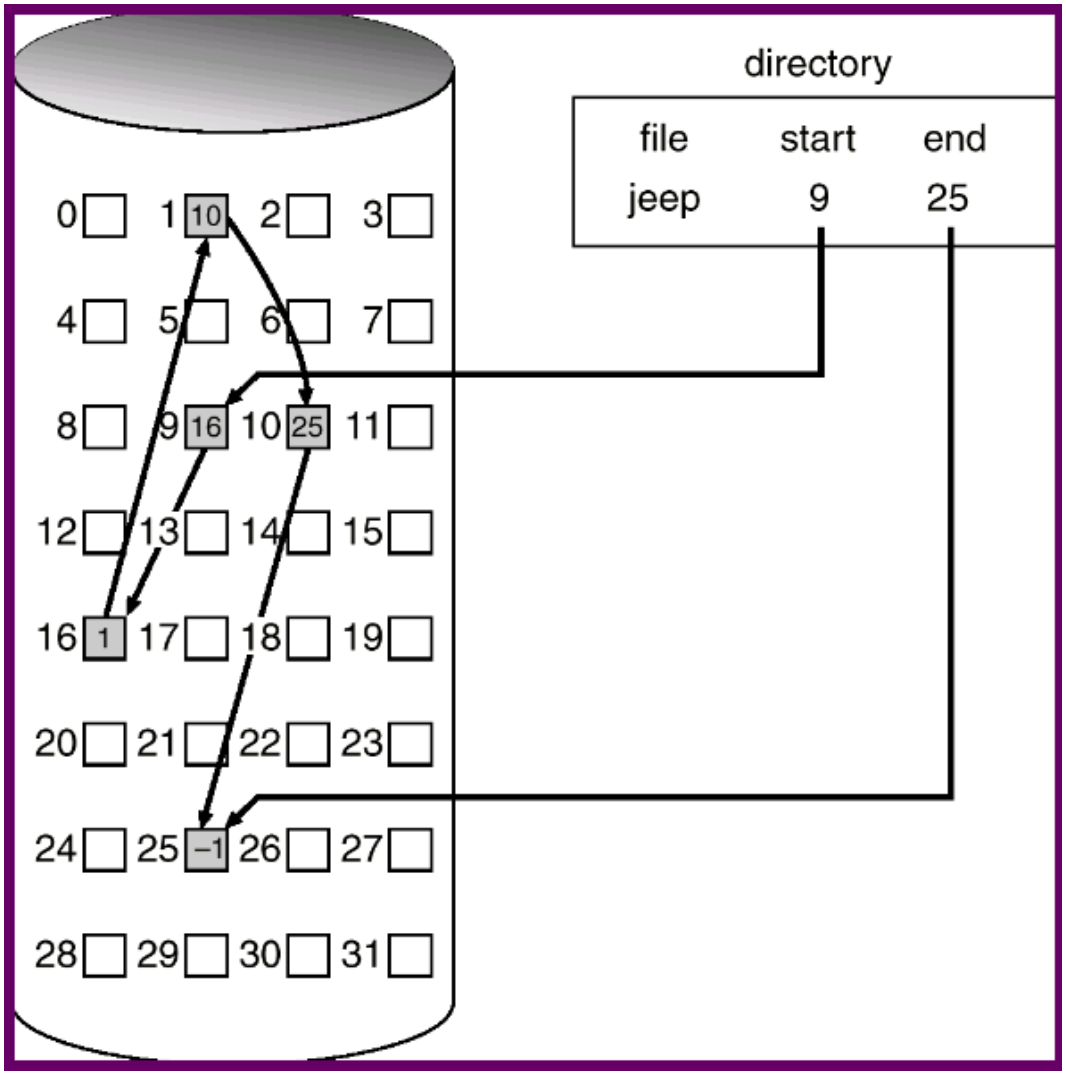
- Permasalahan
  - Fragmentasi eksternal
    - Solusi : Compaction
    - Butuh waktu lama untuk disk besar
  - Besaran blok yang dibutuhkan untuk file yang bersifat dinamis sulit diketahui
    - Contoh: file log, file data karyawan, dan lainnya
    - Solusi : block extension (tambahan blok)
    - Masih bisa terjadi fragmentasi external dan internal

Linked Allocation (cont.)

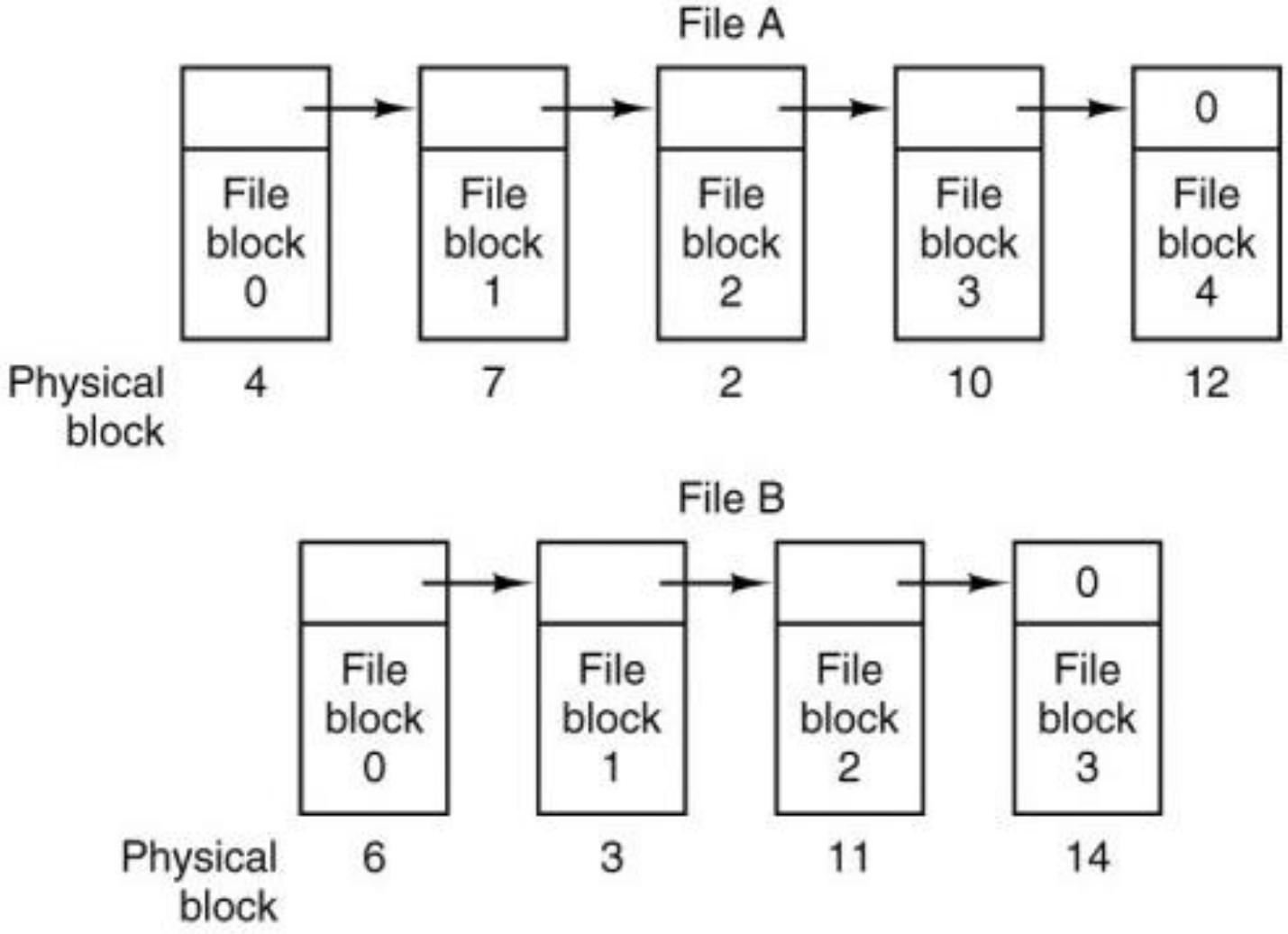
- Blok setiap file tersebar pada disk dan disatukan dengan menggunakan pointer.
- Keuntungan
  - Tidak ada fragmentasi eksternal
  - Fragmentasi internal hanya terjadi pada akhir blok setiap file
  - Effektif untuk sequential access



Linked Allocation (cont.)



Linked Allocation (cont.)



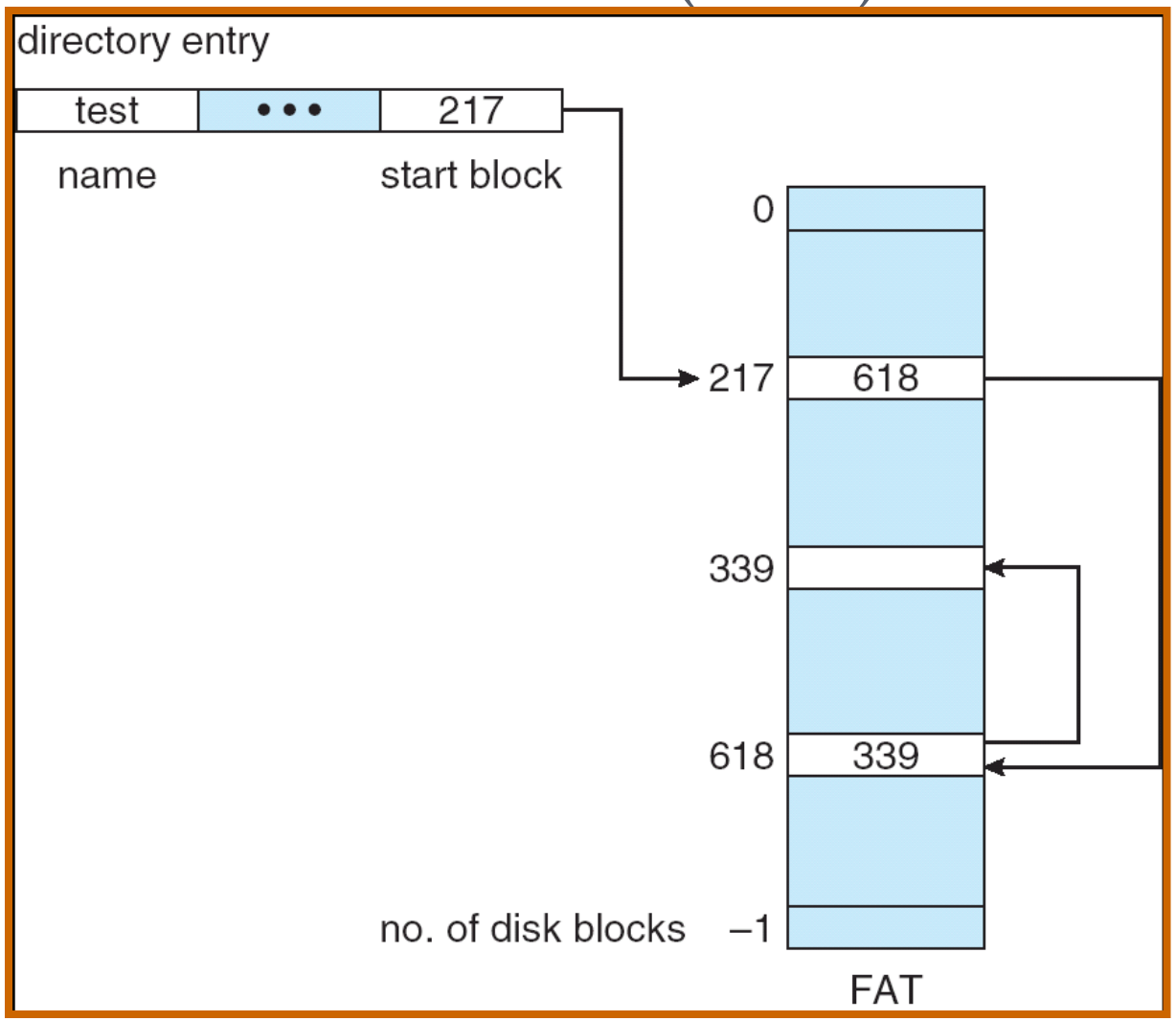
Linked Allocation (cont.)

- Permasalahan
  - Butuh alokasi ruang untuk pointer
    - 1 blok = 512 byte
    - 1 blok = pointer (4 byte) + sisa blok (508 byte)
    - Solusi ? buat cluster block
  - Kurang efektif untuk random access
    - Lambat
    - Baca blok i, harus dari blok i-1
  - Reliabilitas
    - Pointer hilang
    - Solusi: pakai dua pointer, prev dan next (double linked-list)

File Allocation Table (FAT)

- FAT: Menggunakan tabel pointer yang merujuk ke alamat blok
- Tabel disimpan pada blok awal sebuah volume
- Untuk mempercepat akses, FAT disimpan pada cache atau memori

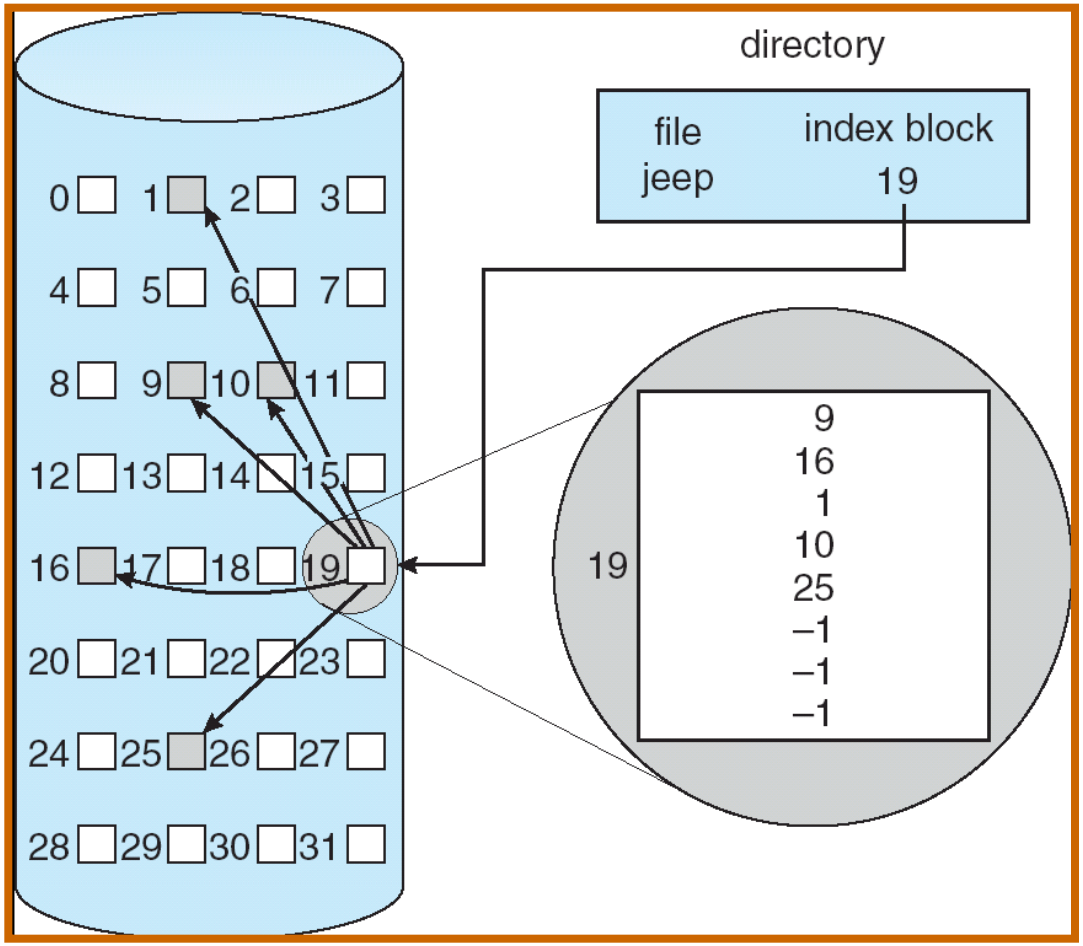
File Allocation Table (FAT)



Indexed Allocation

- Blok digunakan untuk menampung indeks
- Alamat blok suatu file ditempatkan pada blok indeks.
- Direktori berisikan nama file dan alamat indeks blok.
- Mendukung pengaksesan secara sequential & direct (random)

Indexed Allocation

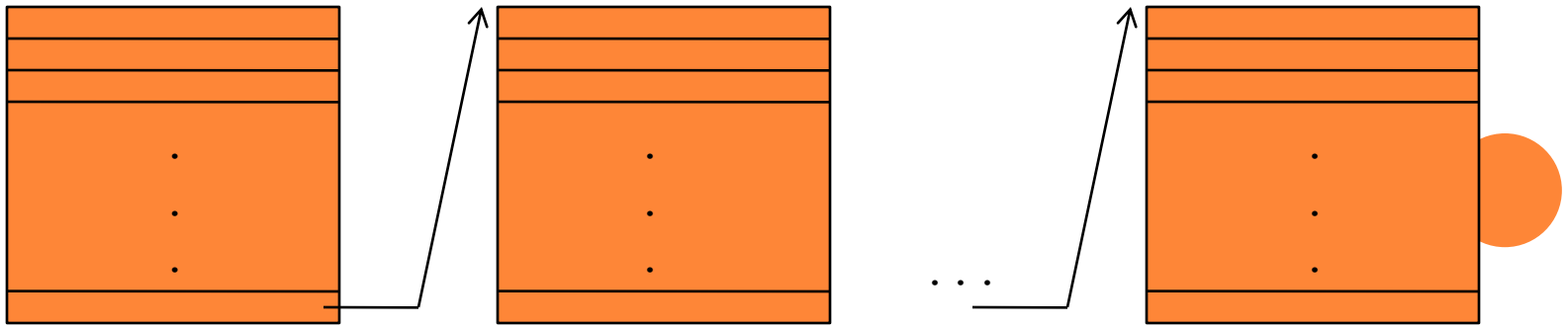


Indexed Allocation

- Permasalahan
  - Jika file mempunyai sedikit blok, maka akan banyak ruang kosong yang tersisa pada blok indeks (fragmentasi internal)
- Bagaimana jika blok indeks tidak cukup untuk menampung entri alamat blok suatu file?
  - Linked Scheme
  - Multilevel Index
  - Combined Scheme

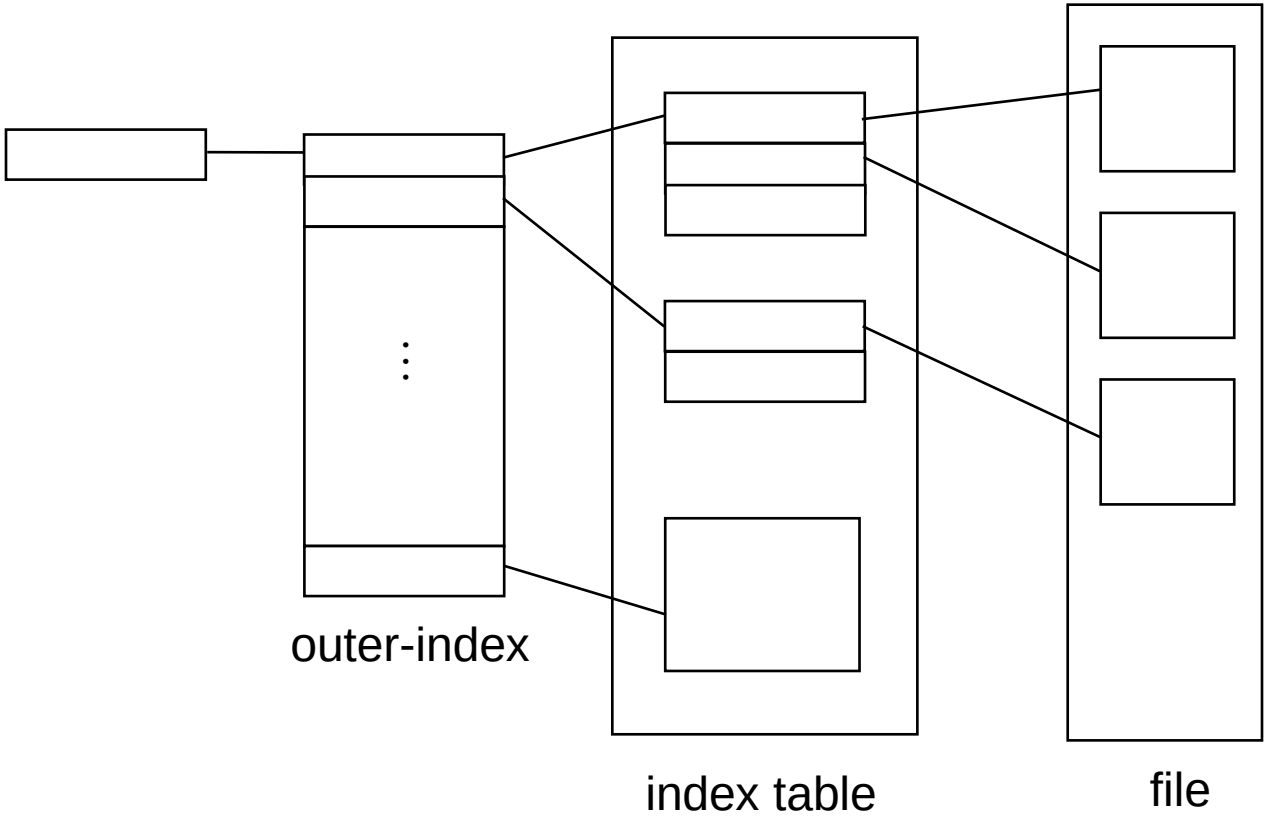
Linked Scheme

- Link beberapa blok indeks
- Contoh:
  - 1 blok indeks dapat menyimpan 100 alamat blok.
  - Jika sebuah file membutuhkan blok data sebanyak 100 blok atau lebih, maka jumlah blok indeks akan berjumlah lebih dari satu blok



Multilevel Index

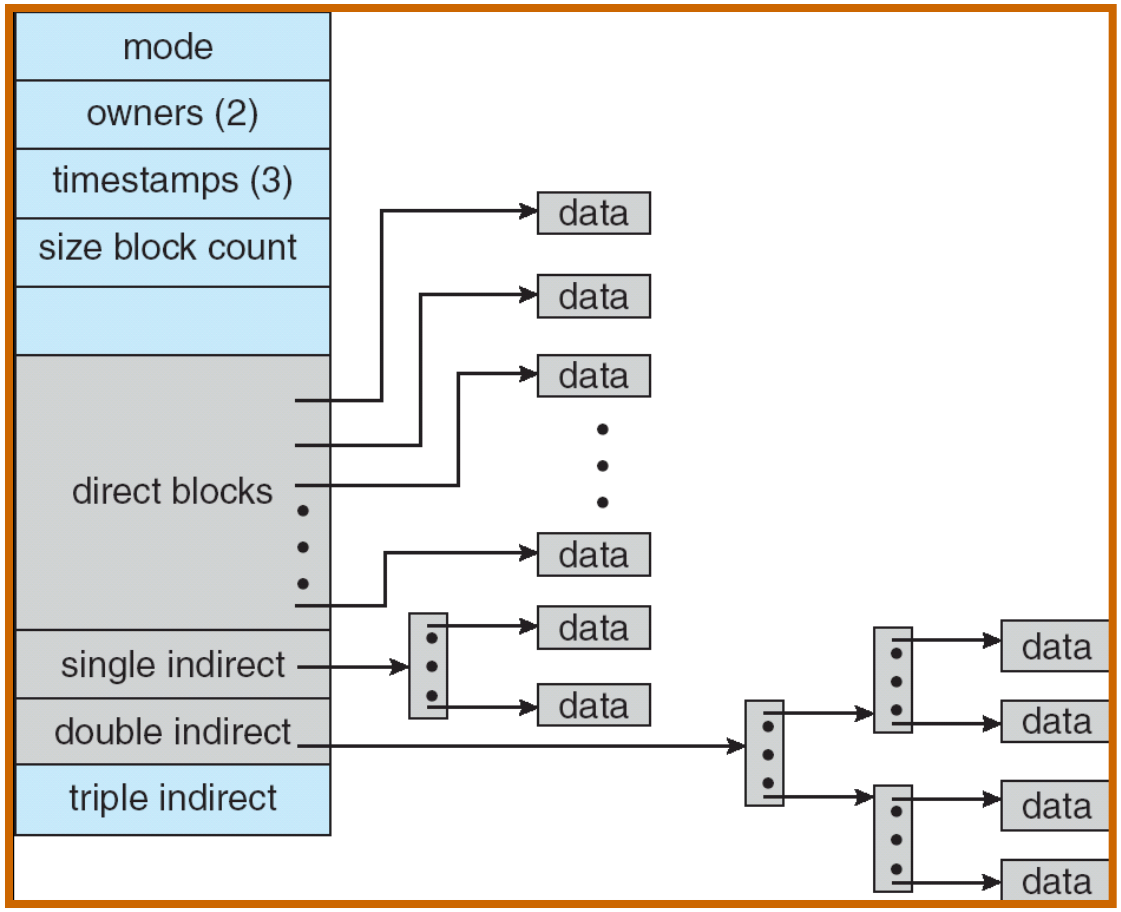
- Indeks bertingkat : first level index, two level index



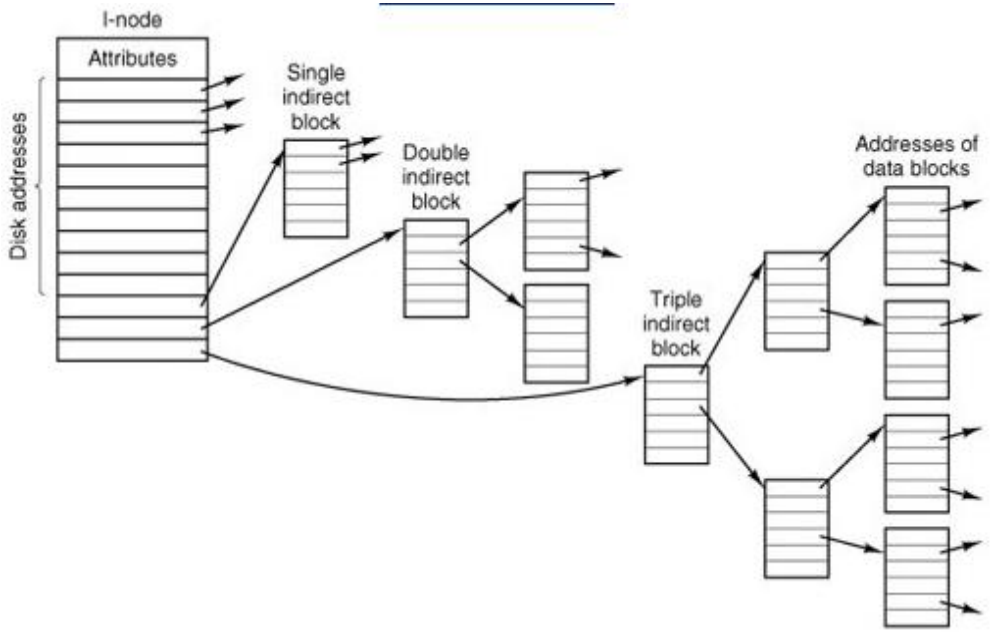
Combined Scheme

- Combined scheme
  - Digunakan pada UNIX File System (UFS)
  - Blok indeks disimpan pada inode (UFS)
  - Pointer file pada inode = 15 entri
    - 12 entri pertama mengacu ke direct block
    - 1 entri single indirect block
    - 1 entri double indirect block
    - 1 entri triple indirect block

Combined Scheme: Inode (UFS)



INODE DENGAN UKURAN 1 BLOK = 4KIB



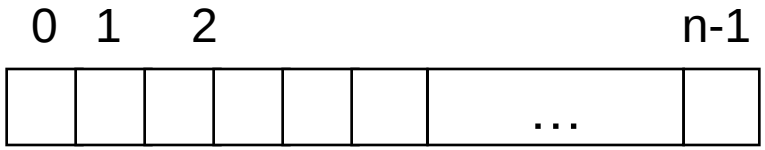
Level	Number of Blocks	Number of Bytes
Direct	12	48K
Single Indirect	512	2M
Double Indirect	$512 \times 512 = 256K$	1G
Triple Indirect	$512 \times 256K = 128M$	512G

Manajemen Ruang Kosong

- Blok kosong dapat dicari pada *free-space list*, yang berisikan daftar blok yang kosong.
- Sebelum file dibuat:
  - Berapa jumlah blok kosong yang dibutuhkan?
  - Cari blok kosong yang tersedia pada *free-space list*
- Mekanisme pembentukan *free-space list*: Bit Vector, linked list, grouping dan counting

BIT VECTOR (CONT.)

- Menggunakan sejumlah n bit untuk mengidentifikasi sejumlah n blok
  - Bit 0 : blok terisi
  - Bit 1 : blok kosong



$$\text{bit} \begin{cases} 0 \Rightarrow \text{blok}[i] \text{ terisi} \\ 1 \Rightarrow \text{blok}[i] \text{ kosong} \end{cases}$$

Blok kosong → 2, 3, 8, 9

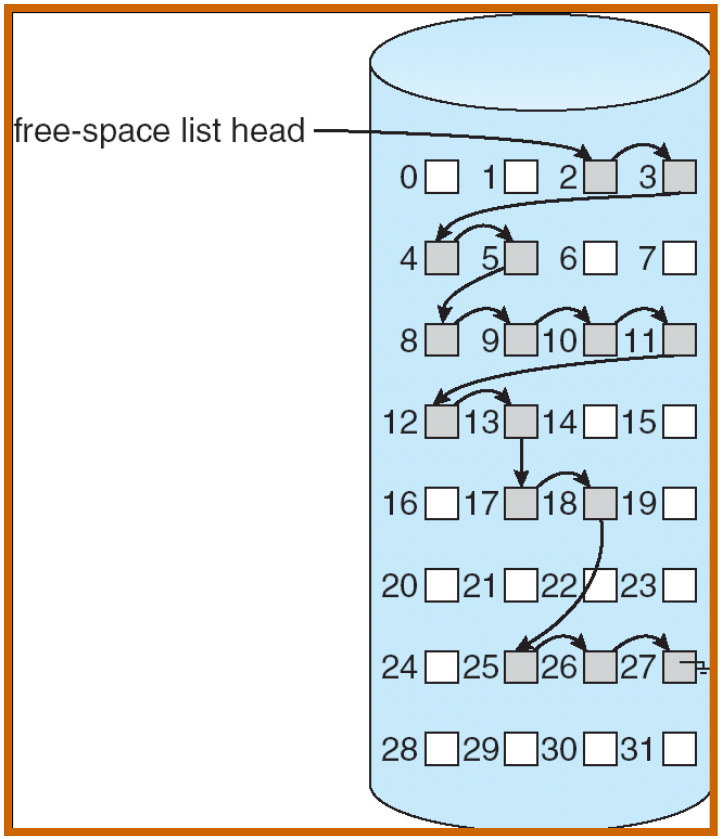
→ 00110000110...

BIT VECTOR

- Keuntungan
  - Pencarian blok mudah dilakukan
- Lambat, bila bit vector tidak berada dimemori
- Butuh sejumlah byte untuk menampung bit vector. Contoh:
  - Ukuran blok = 2^{12} byte
  - Ukuran disk = 2^{30} byte (1 gigabyte)
  - Dibutuhkan bit sebanyak $2^{30}/2^{12} = 2^{18}$ bit
  - Ruang untuk menampung bit vector = $2^{18}/2^{3} = 2^{15}$ byte (32 kbyte)

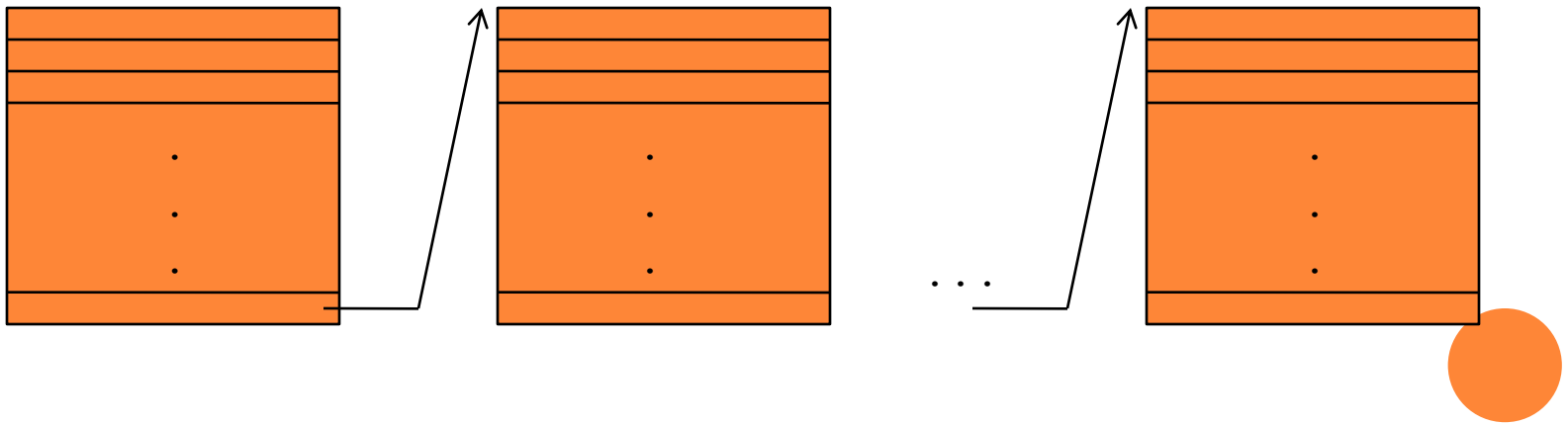
Linked List

- Blok disk kosong saling terhubung dengan pointer



GROUPING

- Menyimpan sejumlah *n-1* alamat blok kosong pada sebuah blok.
- Pada entri ke *n* terdapat pointer yang mengacu ke blok yang juga berisikan daftar *n* blok kosong



Counting

- Memanfaatkan perilaku sistem yang menghapus dan menempatkan file secara sekuensial
- Format : <lokasi awal alamat blok kosong, n>
- n = Jumlah blok kosong yang berurutan

Selesai