

# IF3191- Manajemen File

Henny Y. Zubir  
Departemen Teknik Informatika  
Institut Teknologi Bandung



## Ikhtisar

- Konsep File
- Metode Akses
- Implementasi File
- Implementasi Direktori
- Manajemen Media Penyimpanan
- Efisiensi & Performansi
- Pemulihan



## Manajemen File

- **File:** koleksi informasi bernama
- File manager mengelola kumpulan dengan cara:
  - Menyimpan informasi pada perangkat
  - Pemetaan blok pada media penyimpanan dengan view logik
  - Alokasi/dealokasi media penyimpanan
  - Menyediakan direktori file

## Kebutuhan Penyimpanan Informasi

- Dapat menyimpan data dalam jumlah besar dan jangka panjang
- Informasi harus tetap disimpan meskipun proses yang menggunakannya telah berakhir
- Informasi harus dapat diakses secara konkuren oleh banyak proses

## Konsep File: Penamaan

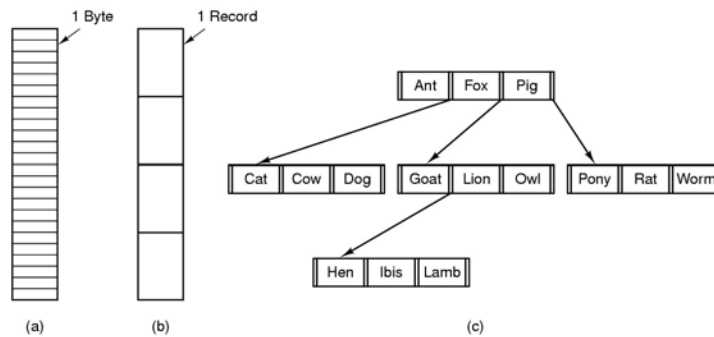
- Terdiri dari nama dan ekstensi
- Contoh:

| Extension | Meaning   |
|-----------|---|
| file.bak  | Backup file                                       |
| file.c    | C source program                                  |
| file.gif  | CompuServe Graphical Interchange Format image     |
| file.hlp  | Help file   |
| file.html | World Wide Web HyperText Markup Language document |
| file.jpg  | Still picture encoded with the JPEG standard      |
| file.mp3  | Music encoded in MPEG layer 3 audio format        |
| file.mpg  | Movie encoded with the MPEG standard              |
| file.o    | Object file (compiler output, not yet linked)     |
| file.pdf  | Portable Document Format file                     |
| file.ps   | PostScript file                                   |
| file.tex  | Input for the TEX formatting program              |
| file.txt  | General text file                                 |
| file.zip  | Compressed archive                                |

## Konsep File: Struktur (1)

- Tidak ada struktur – sekuens of words atau bytes
- Struktur rekord sederhana
  - Baris
  - Panjang tetap
  - Panjang bervariasi
- Struktur kompleks
  - Dokumen terformat
  - Relocatable load file
- Siapa yang menentukan struktur?
  - OS
  - Program

## Konsep File: Struktur (2)

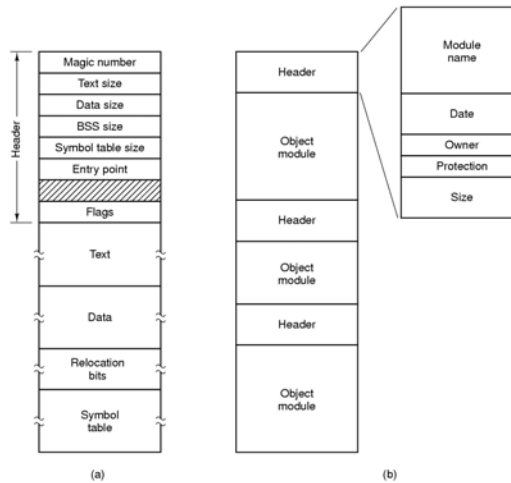


## Konsep File: Atribut

- Nama – informasi untuk pengacuan file, disimpan dalam bentuk yg dapat dibaca user
- Tipe – diperlukan untuk sistem yang mendukung tipe berbeda
- Lokasi – pointer ke lokasi file pada perangkat
- Size – ukuran file saat ini
- Proteksi – mengontrol siapa yang bisa membaca, menulis, atau mengeksekusi file
- Time, date, dan user identification – data untuk proteksi, security, dan monitoring penggunaan
- Informasi mengenai file disimpan pada struktur direktori, yang dikelola pada disk

## Konsep File: Tipe

- a) File Executable
- b) File archive



## Konsep File: Metode Akses

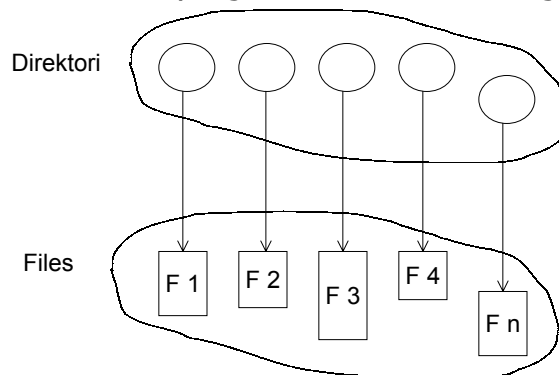
- Akses sekuensial
  - Membaca semua bytes/records dari awal
  - Tidak dapat melompat, hanya bisa melakukan rewind atau back up
  - Baik jika digunakan pada media pita magnetik
- Akses random
  - bytes/records dibaca tidak terurut
  - penting untuk sistem basisdata
  - pembacaan dapat berupa:
    - memindahkan penanda file (seek), kemudian baca, atau ...
    - baca dan kemudian pindahkan penanda file

## Konsep File: Operasi (1)

- create
- delete
- read
- write
- $\text{open}(F_i)$  – mencari entri  $F_i$  pada struktur direktori pada disk dan memindahkan isi entri tsb ke memori
- $\text{close}(F_i)$  – memindahkan isi entri  $F_i$  di memori ke struktur direktori pada disk
- seek – reposisi pada file
- truncate
- append
- get attributes
- set attributes
- rename

## Konsep Direktori: Struktur

- Kumpulan nodes yang berisi informasi mengenai semua file



- Baik struktur direktori maupun file berada pada disk

## Konsep Direktori: Atribut

- Nama
- Tipe
- Alamat
- Panjang saat ini
- Panjang maksimum
- Tanggal terakhir diakses (untuk arsip)
- Tanggal terakhir diupdate (untuk dump)
- ID pemilik
- Informasi proteksi

## Konsep Direktori: Operasi

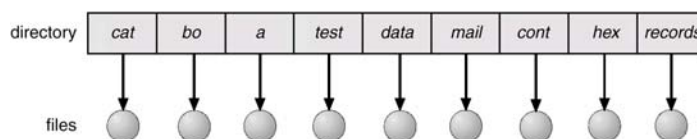
- Pencarian file
- Pembuatan file
- Penghapusan file
- List isi direktori
- Rename file
- Menjelajah (traverse) sistem file

## Konsep Direktori: Issue

- Efisiensi – pencarian file dgn cepat
- Penamaan – menyenangkan bagi user
  - Dua user bisa memiliki nama yg sama utk file yg berbeda
  - File yg sama bisa memiliki beberapa nama yg berbeda
- Grouping – pengelompokan logik dari file berdasarkan jenisnya (mis. Program Pascal, game, ...)

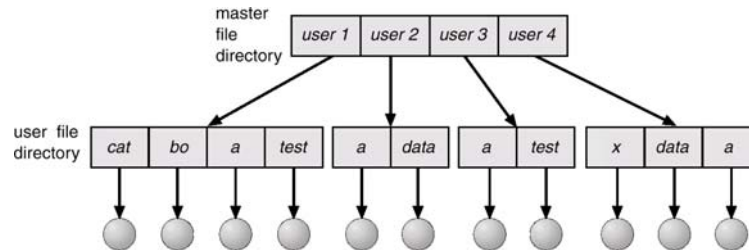
## Konsep Direktori: Struktur (1)

- Direktori Satu Tingkat:
  - Satu direktori tunggal untuk semua user
  - Kelebihan/kekurangan:
    - Penamaan
    - Pengelompokan (grouping)



## Konsep Direktori: Struktur (2)

- Direktori Dua Tingkat:



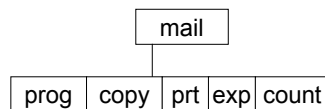
## Konsep Direktori: Struktur (3)

- Direktori Dua Tingkat (Lanj.):
  - Direktori terpisah untuk tiap user
  - Kelebihan/kekurangan:
    - Nama path
    - Bisa memiliki nama file sama utk user yg berbeda
    - Pencarian efisien
    - Tidak memiliki kemampuan grouping



## Konsep Direktori: Struktur (6)

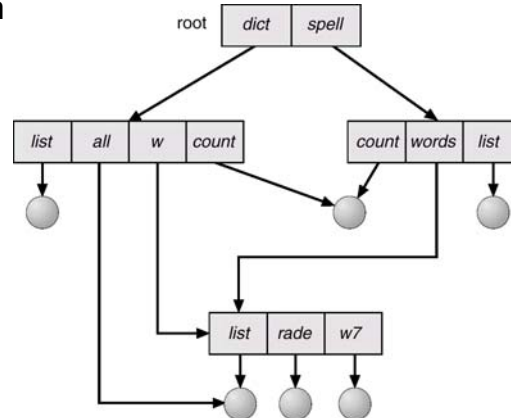
- Menghapus file: `rm <nama-file>`
- Pembuatan sub-direktori baru dilakukan pada current directory
  - `mkdir <nama-dir>`
  - contoh: jika current directory `/spell/mail`  
`mkdir count`



- Menghapus "mail" ⇒ menghapus keseluruhan subtree yang akarnya "mail"

## Konsep Direktori: Struktur (6)

- Graf asiklik:
  - Memiliki sub-direktori dan file yang dapat digunakan bersama



## Konsep Direktori: Struktur (7)

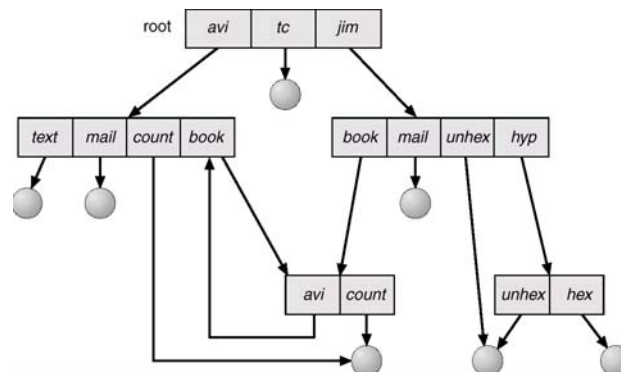
- Graf asiklik (Lanj.):
  - Dua nama yg berbeda (alias)
  - jika *dict* menghapus *list*  $\Rightarrow$  dangling pointer

Solusi:

- Backpointer, sehingga kita dapat menghapus semua pointer  
backpointer menggunakan model daisy chain
- Solusi entry-hold-count

## Konsep Direktori: Struktur (8)

- Direktori Graf Umum:



## Konsep Direktori: Struktur (9)

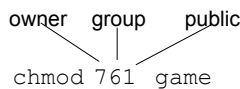
- Direktori Graf Umum (Lanj.):
  - Bagaimana menjamin tidak terdapat cycle?
    - Hanya mengizinkan link ke file, bukan sub-direktori
    - Melakukan garbage collection
    - Setiap kali link baru ditambahkan, gunakan algoritma pendeteksian cycle untuk menentukan OK/tidaknya

## Proteksi

- Pemilik/pembuat file harus dapat mengontrol:
  - Akses apa yg dapat dilakukan
  - Oleh siapa
- Tipe akses
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

## Proteksi: Access List dan Grup

- Mode akses: read, write, execute (r, w, x)
- Tiga kelas user
  - a) akses pemilik RWX 7 ⇒ 1 1 1
  - b) akses group 6 RWX ⇒ 1 1 0
  - c) akses publik 1 RWX ⇒ 0 0 1
- Manager dpt membuat group (unik), dan menambahkan user untuk group tsb
- Definisikan mode akses pada file atau sub-direktori



- Berikan group ke file: **chgrp** G game



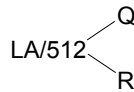
## Implementasi File: Struktur

- Struktur file
  - Unit penyimpanan logik
  - Kumpulan informasi yg saling terkait
- Sistem file berada media penyimpanan sekunder (disk)
- Sistem file diorganisasikan dalam beberapa layer
- *File control block* – struktur penyimpanan yg berisi informasi mengenai file



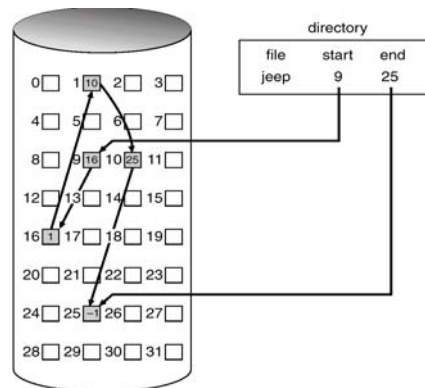
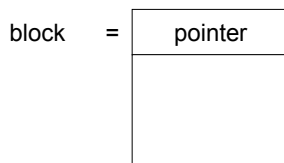
## Implementasi File: Alokasi Kontigu

- Tiap file menempati satu set blok kontigu pada disk
- Sederhana – hanya memerlukan lokasi awal (block #) dan panjang (banyaknya blok)
- Akses random
- Pemborosan ruang (masalah alokasi penyimpanan dinamis)
- File tidak bisa bertambah besar
- Pemetaan dari logik ke fisik
- Blok yg akan diakses = ! + alamat awal
- Displacement ke blok = R

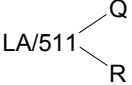


## Implementasi File: Alokasi Berkait (1)

- Tiap file merupakan list berkait dari blok disk: blok berada tersebar pada disk
- Alokasikan sebanyak yang diperlukan

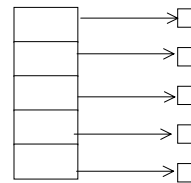


## Implementasi File: Alokasi Berkait (2)

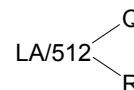
- Sederhana – hanya perlu alamat awal
- Sistem pengelolaan ruang kosong – tidak ada pemborosan ruang
- Tidak ada akses random
- Pemetaan 
  - Blok yg akan diakses adalah blok ke-Q pada rantai blok berkait yg merepresentasikan file
  - Displacement ke blok =  $R + 1$
- *File-allocation table (FAT)* – alokasi ruang disk yg digunakan oleh MS-DOS and OS/2

## Implementasi File: Alokasi Berindeks (1)

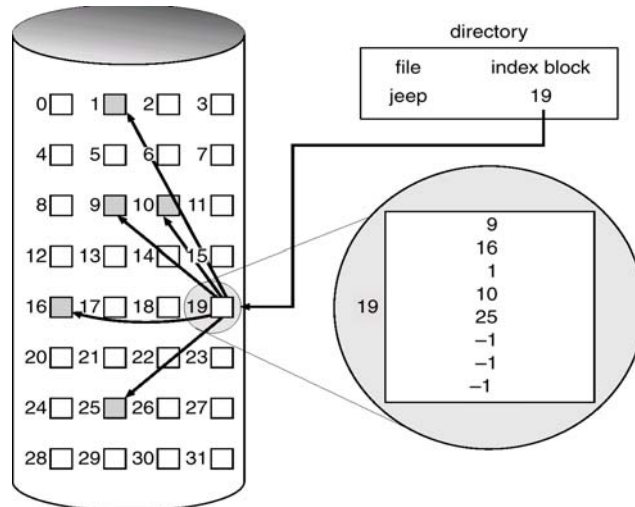
- Bawa semua pointer ke blok indeks
- View lojik
- Perlu tabel indeks
- Akses random
- Akses dinamis tanpa fragmentasi eksternal, tapi memiliki overhead dari blok indeks
- Pemetaan dari lojik ke fisik pada file dgn ukuran maksimum 256K words dan ukuran blok 512 words. Perlu 1 blok utk tabel indeks
  - $Q$  = displacement ke tabel index
  - $R$  = displacement ke blok



Tabel indeks



## Implementasi File: Alokasi Berindeks (2)



## Implementasi File: Alokasi Berindeks (3)

- Pemetaan

- Pemetaan dari logik ke fisik pd file dgn panjang tak terbatas (ukuran blok 512 words)
- Skema berkait – blok link dari tabel indeks (tidak ada batasan pada ukuran)

$$LA / (512 \times 511) \begin{cases} Q_1 \\ R_1 \end{cases}$$

- $Q_1$  = blok dari tabel indeks
- $R_1$  digunakan sbb:

$$R_1 / 512 \begin{cases} Q_2 \\ R_2 \end{cases}$$

- $Q_2$  = displacement ke blok pada indeks tabel
- $R_2$  displacement ke blok file:

## Implementasi File: Alokasi Berindeks (4)

– Indeks two-level (ukuran file maksimum  $512^3$ )

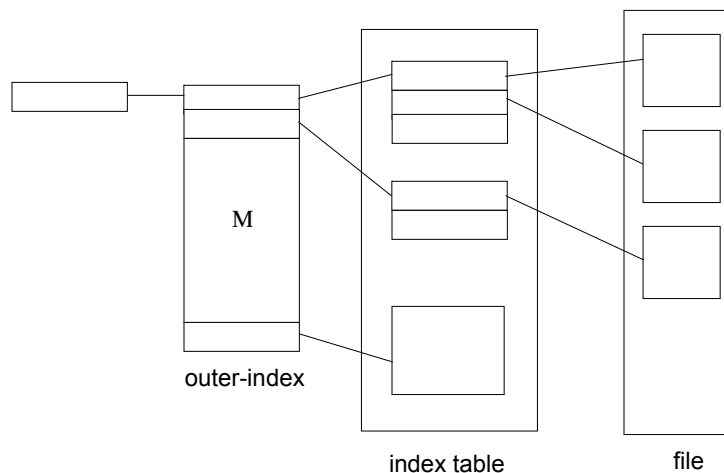
$$LA / (512 \times 512) \begin{cases} Q_1 \\ R_1 \end{cases}$$

- $Q_1$  = displacement ke indeks luar
- $R_1$  digunakan sbb:

$$R_1 / 512 \begin{cases} Q_2 \\ R_2 \end{cases}$$

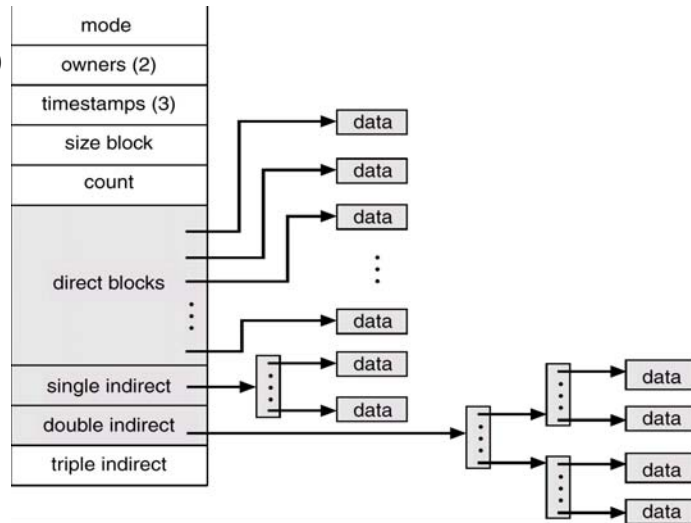
- $Q_2$  = displacement ke blok pd tabel indeks
- $R_2$  displacement blok file:

## Implementasi File: Alokasi Berindeks (5)



## Implementasi File: Kombinasi

Unix:  
(4 Kbyte per blok)



## Manajemen Ruang Kosong (1)

- Bit vector ( $n$  blok)



$\text{bit}[j] = \begin{cases} 0 & \Rightarrow \text{block}[j] \text{ kosong} \\ 1 & \Rightarrow \text{block}[j] \text{ ditempati} \end{cases}$

- Kalkulasi nomor blok

(banyaknya bit per word) \*  
(banyaknya 0-value words) +  
offset bit 1 pertama

## Manajemen Ruang Kosong (2)

- Bit map memerlukan ruang tambahan.
  - Contoh:
    - ukuran blok size =  $2^{12}$  bytes
    - ukuran disk =  $2^{30}$  bytes (1 GByte)
    - $n = 2^{30}/2^{12} = 2^{18}$  bits (or 32KBytes)
- Mudah untuk memperoleh file kontigu
- List berkait (list kosong)
  - Sukar memperoleh ruang kontigu
  - Tidak ada pemborosan ruang
- Grouping
- Counting

## Manajemen Ruang Kosong (3)

- Perlu memproteksi:
  - Pointer ke list kosong
  - Bit map
    - Harus disimpan pada disk
    - Salinan pada memori dan disk mungkin berbeda
    - Tidak memungkinkan blok[*i*] berada pd situasi dimana bit[*i*] = 1 pada memori dan bit[*i*] = 0 pada disk
  - Solusi:
    - Set bit[*i*] = 1 pada disk
    - Alokasikan block[*i*]
    - Set bit[*i*] = 1 pada memori

## Implementasi Direktori

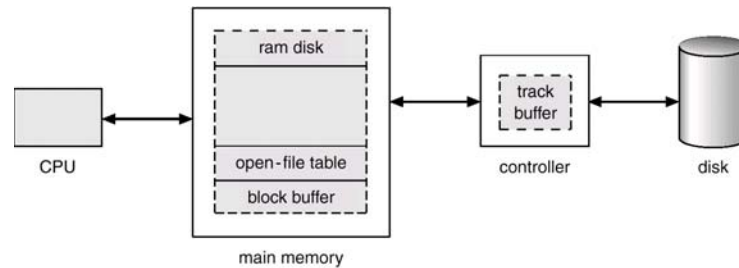
- List linier dari nama file dgn pointe ke blok data
  - Sederhana utk diprogram
  - Lama dalam eksekusi
- Hash Table – list linier dgn struktur data hash
  - mengurangi waktu pencarian direktori
  - *collitions* – situasi dimana dua nama file hash ke lokasi yg sama
  - ukuran fixed

## Efisiensi & Performansi

- Efisiensi tergantung pada:
  - Algoritma alokasi disk dan direktori
  - Tipe data yg disimpan pada entri direktori file
- Performansi
  - disk cache – bagian memori utama yg menyimpan isi blok yg sering diakses
  - free-behind and read-ahead – teknik utk optimasi akses sekuensial
  - meningkatkan performansi PC dgn menjadikan sebagian memori untuk virtual disk atau RAM disk

## Disk Caching

- Berbagai alternatif lokasi disk cache



## Pemulihan

- Consistency checker – membandingkan data pada struktur direktori dgn blok data pada disk, dan mencoba memperbaiki inkonsistensi
- Menggunakan program utk mem-backup data dari disk ke perangkat penyimpanan lainnya (floppy disk, magnetic tape)
- Memulihkan kehilangan file atau disk dgn mengambil data dari backup