Bab 9

Sistem File

POKOK BAHASAN:

- ✓ Konsep File
- ✓ Metode Akses
- ✓ Struktur Direktori
- ✓ File System Mounting
- ✓ File Sharing
- ✓ Proteksi

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami konsep file
- ✓ Memahami metode akses file dan penyimpan sekunder
- ✓ Memahami struktur file dan direktori
- ✓ Memahami proses mounting pada sistem file
- ✓ Memahami proteksi pada file

9.1 KONSEP FILE

File adalah unit penyimpan logika yang diabstraksi sistem operasi dari perangkat penyimpan. File berisi informasi yang disimpan pada penyimpan sekunder (seperti magnetic disk, magnetic tape dan optical disk). Informasi dalam file didefinisikan oleh pembuatnya. Sebuah file mempunyai struktur tertentu tergantung

tipenya. Tipe file terdiri dari data baik data numeric, karakter maupun binary serta program misalnya source program, object program dan executable program.

9.1.1 Atribut File

Sebuah file mempunyai atribut yg berbeda antara sistem operasi satu dengan lainnya, tetapi secara umum terdiri dari :

- **Nama**, informasi disimpan dalam bentuk yang dapat dibaca manusia
- **Tipe**, diperlukan sistem yang mendukung tipe yang berbeda.
- **Lokasi**, pointer ke lokasi file pada perangkat.
- **Ukuran**, ukuran file saat ini.
- **Proteksi**, mengontrol siapa yang dapat membaca, menulis dan mengeksekusi.
- **Waktu**, **tanggal** dan **identifikasi user**, data untuk monitoring proteksi, sekuriti dan penggunaan.

Informasi file disimpan dalam struktur direktori yang diatur oleh disk.

9.1.2 Operasi pada File

Sebagai tipe data abstrak, perlu didefinisikan operasi yang dapat dibentuk oleh file. Terdapat enam operasi dasar yg disediakan sebagai sistem call yaitu :

- Membuat file (*create*)
- Menulis file (*write*)
- Membaca file (*read*)
- Reposisi dalam file (*file seek*)
- Menghapus file (*delete*)
- Memotong file (*truncate*)
- Open(F_i) mencari struktur direktori untuk entry F_i dan memindahkan isi entry ke memori.
- Close (*Fi*) memindahkan isi entry *Fi* di memori ke struktur direktori pada disk.

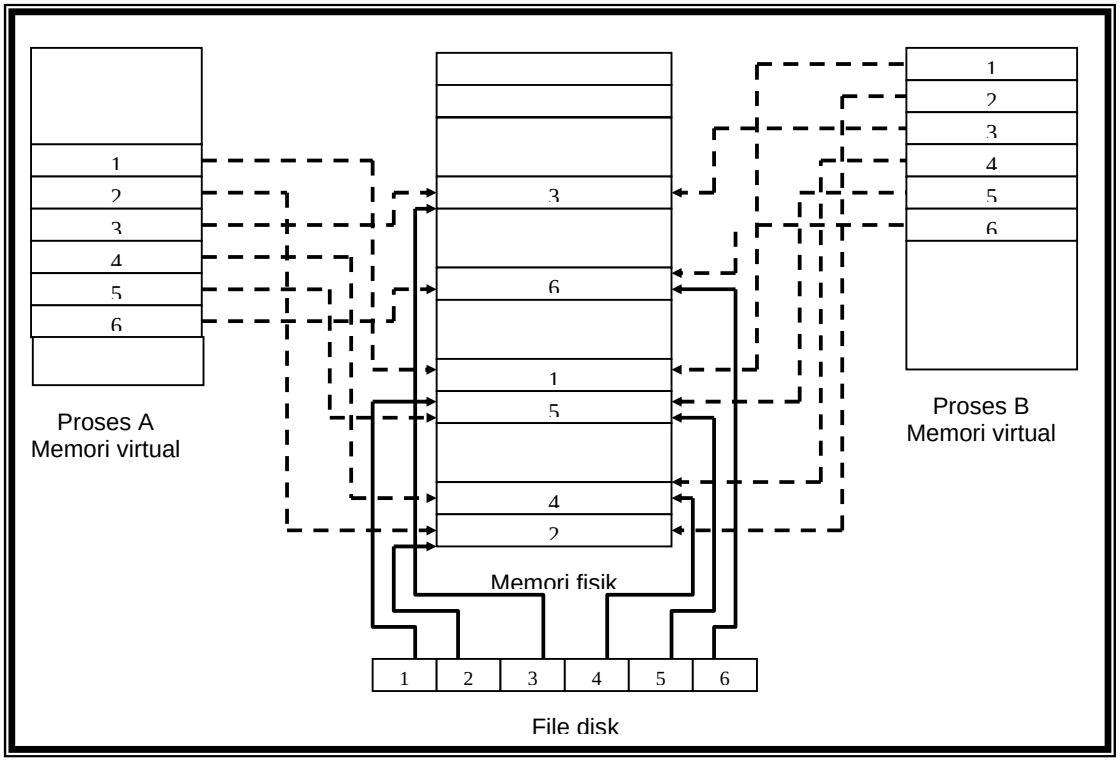
Operasi tambahan yang biasanya dilakukan terhadap file adalah :

- Menambah (*append*) informasi baru pada akhir file yang sudah ada
- Mengubah nama (*rename*) file yang sudah ada
- Membuat duplikasi (*copy*) file

Kebanyakan operasi file melibatkan pencarian direktori untuk masukan yang berhubungan dengan file. Untuk menghindari pencarian tetap, beberapa sistem akan membuka file bila file tersebut aktif pertama kali. Sistem operasi menyimpan tabel kecil yang berisi informasi tentang semua file yang terbuka (*open-file table*). Bila file tidak digunakan lagi, dilakukan penutupan oleh proses dan sistem operasi memindahkan file dari *open-file table.*

Beberapa informasi yang berkaitan dg pembukaan file yaitu

- Pointer file.
- Jumlah file yang dibuka.
- Lokasi file pada disk.



Gambar 9-1 : Pemetaan file ke memori

9.1.3 Tipe File

Salah satu pertimbangan penting dalam merancang sistem file dan keseluruhan sistem operasi adalah apakah sistem operasi mengenali dan mendukung sistem file. Bila sistem operasi mengenali tipe suatu file, maka dapat dilakukan operasi terhadap file

dengan cara yang rasional. Misalnya user yang mencoba mencetak file executable dapat dicegah oleh sistem operasi karena file adalah program binary.

Teknik yg umum untuk implementasi tipe file adalah memasukkan tipe file sebagai bagian dari nama file. Nama file dibagi menjadi dua bagian yaitu nama dan extension (seperti pada MS-DOS) seperti pada Gambar 9-2. Setiap file mempunyai atribut pembuat berisi nama dari program yang membuatnya (seperti pada MS-Windows / Apple Macintosh). Atribut ini di-set oleh sistem operasi saat menggunakan system call *create*. Bila user membuka file tersebut dengan melakukan *double-clicking* mouse pada icon dari file tsb, program yang dibuat ditampilkan otomatis.

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

Gambar 9-2 : Tipe File

UNIX menggunakan *magic number* yang disimpan pada awal file untuk mengindikasikan tipe file berupa program executable, batch file (shell script), file postscript dan lain-lain. Tidak semua file mempunyai *magic number*, sehingga informasi tipe tidak dapat digambarkan. UNIX tidak menyimpan nama dari program pembuatnya. UNIX juga mengijinkan nama extension dari file tersembunyi, sehingga user dapat menentukan tipe file sendiri dan tidak tergantung pada sistem operasi.

9.1.4 Struktur File

Tipe file juga digunakan untuk untuk menunjukkan struktur internal dari file. File tertentu harus konfirmasi ke struktur yang dibutuhkan yang dimengerti oleh sistem operasi. Misalnya sistem operasi membutuhkan file executable yang mempunyai struktur khusus sehingga dapat menentukan dimana letak memory dan lokasi dari instruksi pertama.

Beberapa sistem operasi menggunakan sekumpulan sistem pendukung struktur file dg sejumlah operasi khusus untuk manipulasi file dengan struktur tersebut. Hal ini menjadi kelemahan pada sistem operasi yang mendukung struktur file lebih dari satu. Jika sistem operasi menentukan 10 struktur file berbeda, maka perlu menyertakan kode untuk mendukung struktur file tersebut. Setiap file perlu dapat didefinisikan sebagai satu dari tipe file yang didukung oleh sistem operasi.

Beberapa sistem operasi seperti UNIX dan MS-DOS hanya mendukung sejumlah struktur file. UNIX menentukan setiap file merupakan deret 8 bit byte dan bit tersebut tidak di terjemahkan oleh sistem operasi. Skema ini mempunyai fleksibilitasi maksimum, tetapi sedikit dukungan. Setiap program aplikasi harus mnyertakan kode sendiri untuk menterjemahkan file input ke dalam struktur yang tepat. Setidaknya semua SO harus mendukung sedikitnya satu struktur file executable sehingga sistem dapat *load* dan menjalankan program

9.1.5 Struktur File Internal

Secara internal, sistem disk mempunyai ukuran blok yang ditentukan oleh ukuran sebuah sector. Semua disk I/O dibentuk dalam unit satu blok (*physical record*) yang berukuran sama. Ukuran *physical record* tidak tepat dg panjang *logical record*. *Logical record* mempunyai panjang yang bervariasi. Solusinya adalah dengan

mengirim sejumlah logical record ke blok fisik. Ukuran *logical record*, ukuran blok fisik dan teknik pengiriman menentukan berapa banyak *logical record* yang berada pada blok fisik. Pengiriman dapat dilakukan oleh program aplikasi user atau sistem operasi.

File merupakan deretan blok-blok. Semua fungsi dasar I/O dioperasikan pada blok-blok tersebut. Konversi dari *logical record* ke blok fisik berhubungan dg perangkat lunak sederhana.

9.2 METODE AKSES

File menyimpan informasi. Bila digunakan, informasi tersebut harus diakses dan dibaca ke memory. Terdapat beberapa cara mengakses informasi pada file yaitu akses berurutan (*sequential access*), akses langsung (*Direct access* atau *relative access*) dan metode akses lain.

9.2.1 Akses Berurutan (*Sequential Access*)

Akses berurutan merupakan metode akses paling sederhana. Informasi pada file diproses secara berurutan, satu record diakses setelah record yang lain. Metode akses ini berdasarkan model *tape* dari suatu file yang bekerja dengan perangkat *sequential-access* atau *random-access*.

Operasi pada akses berurutan terdiri dari :

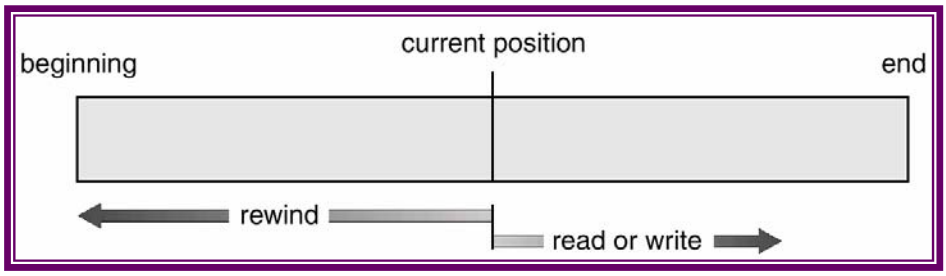
read next

write next

reset

no *read after last write (rewrite)*

Operasi *read* membaca bagian selanjutnya dari file dan otomatis menambah file pointer yang melacak lokasi I/O. Operasi *write* menambah ke akhir file dan ke akhir material pembacaan baru (*new end of file*). File dapat di-*reset* ke awal dan sebuah program untuk meloncat maju atau mundur ke *n* record.



Gambar 9-3 : Akses file berurutan

9.2.2 Akses Langsung (*Direct Access*)

File merupakan *logical record* dengan panjang tetap yang memungkinkan program membaca dan menulis record dengan cepat tanpa urutan tertentu. Metode akses langsung berdasarkan model disk dari suatu file, memungkinkan acak ke sembarang blok file, memungkinkan blok acak tersebut dibaca atau ditulis.

Operasi pada akses langsung terdiri dari :

read n

write n

position to n

read next

write next

rewrite n

Operasi file dimodifikasi untuk memasukkan nomor blok sebagai parameter. Nomor blok ditentukan user yang merupakan **nomor blok relatif**, misalnya indeks relatif ke awal dari file. Blok relatif pertama dari file adalah 0, meskipun alamat disk absolut aktual dari blok misalnya 17403 untuk blok pertama. Metode ini mengijinkan sistem operasi menentukan dimana file ditempatkan dan mencegah user mengakses posisi dari sistem file yang bukan bagian dari file tersebut.

Tidak semua sistem operasi menggunakan baik akses berurutan atau akses langsung untuk file. Beberapa sistem hanya menggunakan akses berurutan, beberapa sistem lain menggunakan akses langsung. Untuk mengubah akses berurutan ke akses langsung bukan sesuatu hal yang sulit seperti pada Gambar 9-4.

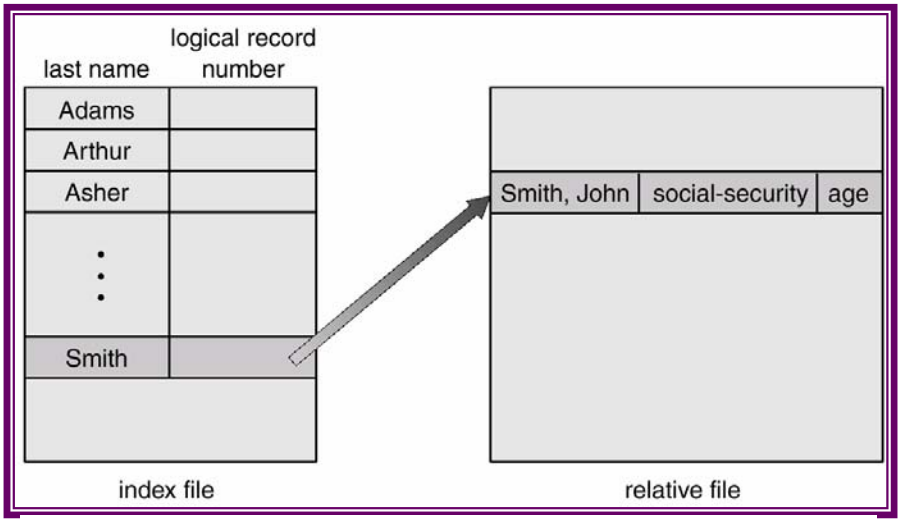
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp*+1;
write next	*write cp*; *cp* = *cp*+1;

Gambar 9-4 : Mengubah akses berurutan menjadi akses langsung

9.2.3 Metode Akses Lain

Metode akses lain dapat dibangun berpedoman pada metode *direct access*. Metode tambahan ini biasanya melibatkan konstruksi indeks untuk file. Indeks, seperti indeks pada bagian akhir buku, berisi pointer ke blok-blok tertentu. Untuk menentukan masukan dalam file, pertama dicari indeks, dan kemudian menggunakan pointer untuk mengakses file secara langsung dan menemukan masukan yang tepat.

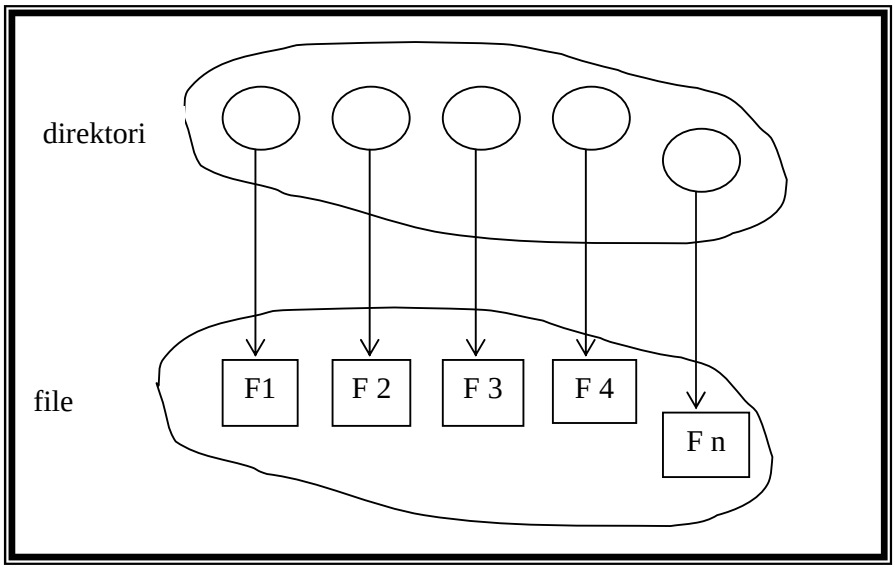
File indeks dapat disimpan di memori. Bila file besar, file indeks juga menjadi terlalu besar untuk disimpan di memori. Salah satu pemecahan nya adalah membuat indeks untuk file indeks. File indeks primer berisi pointer ke file indeks sekunder, yang menunjuk ke data item aktual. Bentuk pengaksesan secara berindeks diilustrasikan pada Gambar 9-5.



Gambar 9-5 : Contoh indeks dan file relatif

9.3 STRUKTUR DIREKTORI

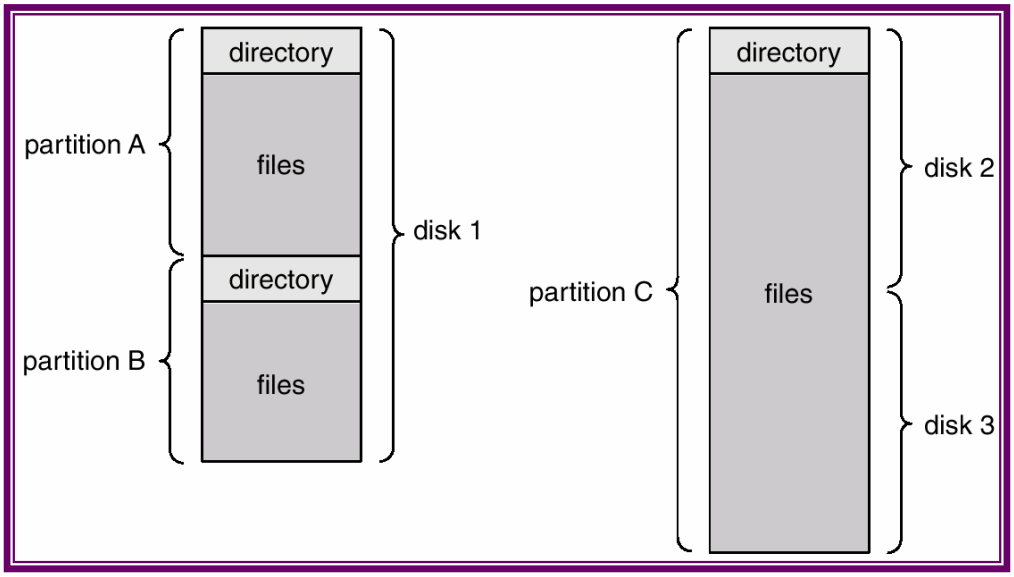
Direktori adalah kumpulan titik yang berisi informasi tentang semua file (Gambar 9-6). Beberapa sistem menyimpan ratusan file pada disk ratusan gigabyte. Untuk mengatur semua data menggunakan organisasi yg dilakukan dalam dua bagian.



Gambar 9-5 : Direktori

Pertama, system file dipecah ke dalam partisi, yang disebut juga "*minidisk*" (pada mesin IBM) atau "*volume*" (pada mesin PC dan Macintosh). Setiap disk pada sistem berisi sedikitnya satu partisi, merupakan struktur low-level dimana file dan direktori berada. Terkadang, partisi digunakan untuk menentukan beberapa daerah terpisah dalam satu disk, yang diperlakukan sebagai perangkat penyimpan yang terpisah. Sistem lain menggunakan partisi yang lebih besar dari sebuah disk untuk mengelompokkan disk ke dalam satu struktur logika.

Kedua, setiap partisi berisi informasi mengenai file di dalamnya. Informasi ini disimpan pada entry dalam "*device directory* atau *volume table of contents*". Perangkat direktori (atau direktori) menyimpan informasi seperi nama, lokasi, ukuran dan tipe untuks semua file dari partisi tersebut. Organisasi file yang umum dapat dilihat pada Gambar 9-7.



Gambar 9-7 : Organisasi sistem file

Informasi yang terdapat pada direktori adalah

- Nama
- Tipe
- Alamat
- Panjang saat ini
- Panjang maksimum
- Tanggal akses terakhir
- Tanggal perubahan terakhir
- ID pemilik
- Informasi proteksi

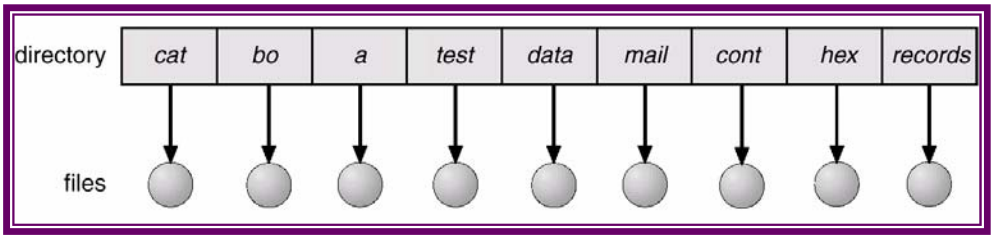
Beberapa operasi yang dibentuk pada direktori adalah :

- Mencari file (*search*)
- Membuat file (*create*)
- Menghapus file (*delete*)
- Mendaftar suatu direktori (*list*)
- Mengubah nama file (*rename*)
- Melintasi sistem file (*traverse*)

Organisasi file dan direktori disarankan yang seefisien mungkin sehingga dapat menempatkan file dengan cepat. Selain itu dalam penamaan file dan direktori harus nyaman untuk user. Dua user dapat memberikan nama file yang sama. File yang sama dapat mempunyai beberapa nama. Dalam organisasi file dan direktori juga perlu dilakukan pengelompokan file berdasarkan property, misalnya semua program Java, game dan lain-lain.

9.3.1 Direktori Satu Level

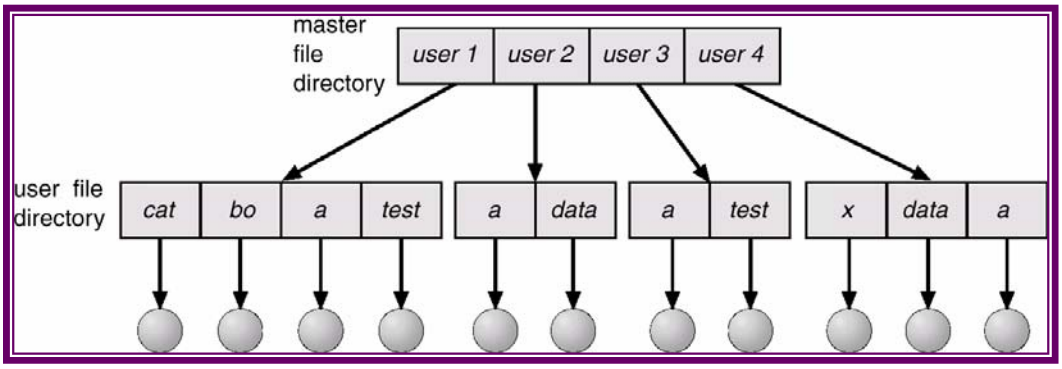
Direktori ini hanya terdiri dari satu direktori untuk setiap user (Gambar 9-8). Pada direktori jenis ini terjadi permasalahan penamaan dan pengelompokan berdasarkan user.



Gambar 9-8 : Direktori satu level

9.3.2 Direktori Dua Level

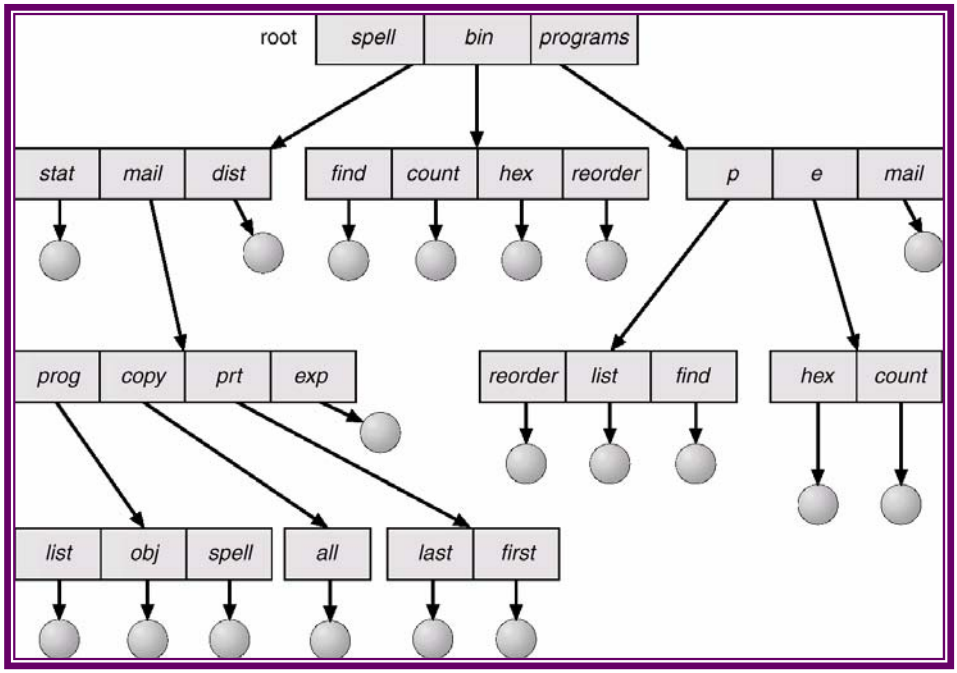
Direktori ini terdiri dari dua level yang memisahkan direktori untuk setiap user (Gambar 9-9). Setiap file diberi nama path, dapat mempunyai nama file yang sama untuk user yang berbeda, mempunyai kapabilitas pencarian, tetapi belum dilakukan pengelompokan.



Gambar 9-9 : Direktori dua level

9.3.3 Direktori Berstruktur Pohon

Direktori berstruktur pohon merupakan struktur direktori yang biasa digunakan. Pohon mempunyai direktori root. Setiap file pada sistem mempunyai nama path yang unik (Gambar 9-10).



Gambar 9-10 : Direktori berstruktur pohon

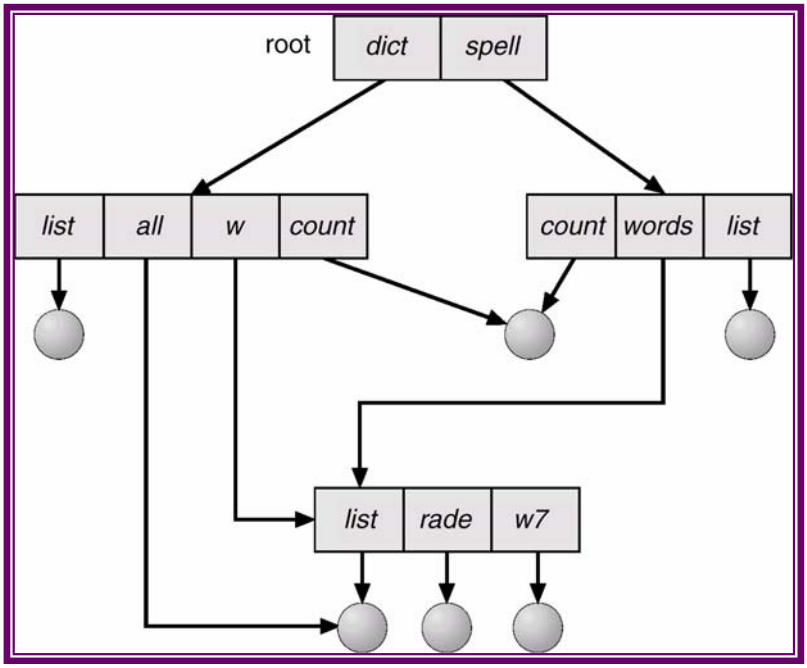
Pada direktori ini pencarian file dan direktori lebih efisien, mengelompokkan file dan dapat mengakses direktori dan sub direktori.

Sebuah direktori atau subdirektori berisi kumpulan file atau sub direktori. Sebuah direktori merupakan file yang diperlakukan dengan cara khusus. Semua direktori mempunyai format internal yang sama. Satu bit dalam setiap masukan direktori merupakan masukan sebagai file (0) atau sebagai subdirektori (1).

Sistem call khusus digunakan untuk membuat dan menghapus direktori. Nama path dapat dibagi menjadi dua tipe yaitu nama path “absolut” dan “relatif”. Pada saat membuat file baru akan dilakukan pada *current directory*. Demikian juga pada saat membuat direktori baru.

9.3.4 Direktori Acyclic Graph

Struktur tree melarang menggunakan bersama-sama file dan direktory. Pada direktori *acyclic graph* memungkinkan direktori mempunyai subdirektori dan file yang digunakan bersama-sama. File dan subdirektori yang sama mungkin berada pada dua direktori yang berbeda (Gambar 9-11).



Gambar 9-11 : Direktori acyclic graph

Direktori *acyclic graph* diimplementasikan dalam beberapa cara. Cara yang umum, pada beberapa system UNIX, adalah membuat entry direktori baru yang disebut *link*. Sebuah *link* berupa sebuah pointer ke file atau subdirektori lain. Sebuah *link* dapat diimplementasikan sebagai nama path absolut atau relatif (sebuah *symbolic link*). Sebuah *link* berbeda dari direktory asal. *Link* diindentifikasi oleh format pada direktory entry dan memberi nama pointer secara tak langsung. Pendekanan lain dengan menduplikasi semua informasi pada direktori yang digunakan bersama-sama sehingga kedua entri identik dan sama. Cara ini menyebabkan informasi asli dan duplikasi tidak dapat dibedakan

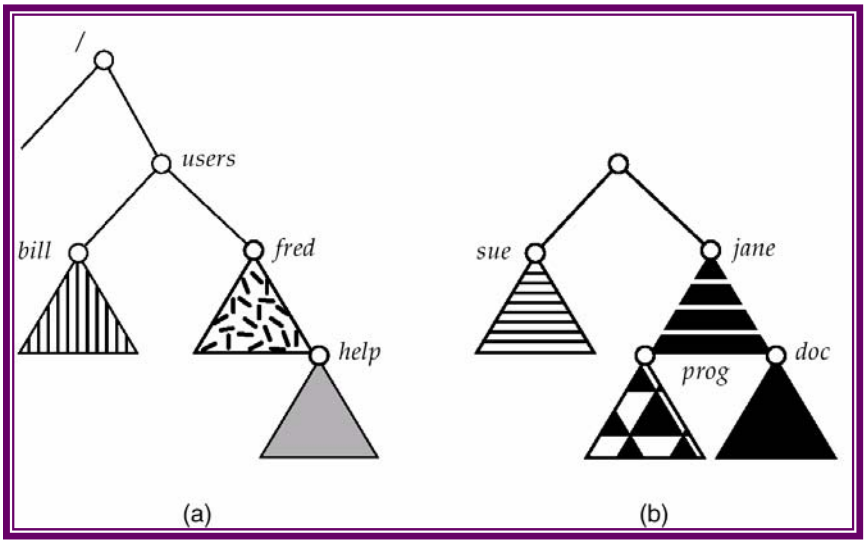
Permasalahan mendasar adalah memelihara konsistensi jika file dimodifikasi. Struktur direktori *acyclic-graph* lebih fleksibel daripada struktur pohon, tetapi lebih

kompleks. Sebuah file mungkin mempunyai lebih dari satu nama path, konsekuensinya, nama file yang berbeda harus merujuk ke file yang sama. Jika mencoba melintasi keseluruhan sistem file (misalnya untuk akumulasi statistik pada semua file) struktur sharing tidak boleh dilintasi lebih dari satu kali.

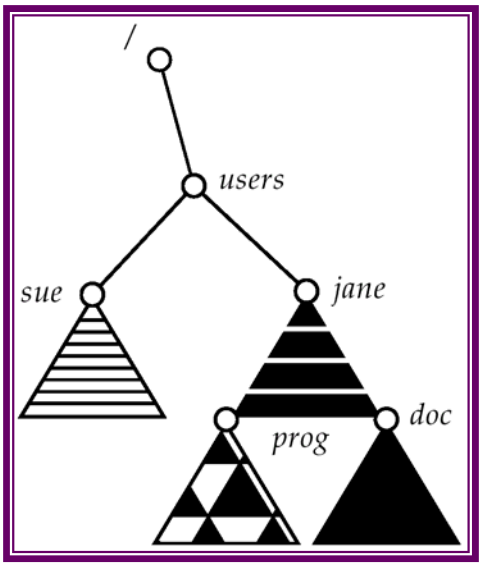
Masalah lainnya melibatkan penghapusan. Kapan ruang yang sudah dialokasikan untuk file yang digunakan bersama-sama dapat di dealokasi dan digunakan lagi. Pendekatan lain untuk penghapusan adalah menyediakan file sampai semua acuan dihapus.

9.4 FILE SYSTEM MOUNTING

Suatu sistem file harus di-*mount* sebelum diakses. File yang tidak di-*mount* seperti Gambar 9-12 akan dilakukan proses mounting pada *mount point* (Gambar 9-13)



Gambar 9-12: (a) Sistem eksis (b) partisi yang tidak di-mount



Gambar 9-13 : Mount point

9.5 PROTEKSI

Informasi yang disimpan dalam system komputer harus diproteksi dari kerusakan fisik (*reliability*) dan akses yang tidak benar (*protection*).

Reliability biasanya dilakukan dengan duplikasi copy dari file. Beberapa sistem komputer mempunyai sistem yang secara otomatis (atau melalui intervensi operator komputer) menduplikasi file ke tape secara regular dari sistem file yang secara tiba-tiba dihapus. *Protection*, sebaliknya, dapat dilakukan dalam beberapa cara.

9.5.1 Tipe Akses

Mekanisme proteksi dengan tipe akses file terbatas yang dapat dibuat. Akses diperbolehkan atau tidak tergantung beberapa faktor, satu diantaranya permintaan tipe akses. Beberapa operasi yang disediakan :

- Membaca dari file (*read*)
- Menulis ke file (*write*)
- Menjalankan file (*execute*)
- Menambah isi file (*append*)
- Menghapus file (*delete*)

- Melihat nama dan atribut file (*list*)

Operasi yang lain, seperti pemberian nama, meng-copy atau mengubah file, juga harus dikontrol. Untuk beberapa alasan, fungsi level lebih tinggi (seperti mengcopy) diimplementasikan oleh system program yang menggunakan system call level lebih rendah. Proteksi disediakan hanya pada level lebih rendah. Sebagai contoh, meng-copy file diimplementasikan dengan deretan permintaan membaca. Dalam hal ini user dengan akses read dapat menyebabkan file di-copy, dicetak dan lain-lain.

9.5.2 Access List dan Group

Pendekatan permasalahan proteksi yang sering digunakan adalah dengan membuat akses secara dependent pada identifikasi user. Skema umum untuk implementasi akses *identity-dependent* dengan menghubungkan masing-masing file dan direktori dg sebuah *access list* yang menentukan nama user dan tipe akses yang diijinkan untuk setiap user.

Bila user meminta akses ke file khusus, sistem operasi memeriksa *access list.* Jika user tersebut terdaftar, akses diijinkan, sebaliknya terjadi *protection violation* dan dilarang mengakses file.

Masalah pokok dengan *access list* adalah ukuran. Jika ingin mengijinkan user membaca file, harus didaftar semua user dengan akses read. Teknik ini mempunyai dua konsekuensi yaitu membangun sebuah daftar mungkin kesulitan dan directory entry yang sebelumnya mempunyai ukuran tetap sekarang menjadi ukuran bervariasi, sehingga muncul permasalahan manajemen ruang.

Masalah ini dipecahkan dengan melakukan pengetatan terhadap *access list.* Beberapa system memperkenalkan tiga klasifikasi user :

- **Owner**. User yang membuat file
- **Group**. Kumpulan user yang menggunakan file bersama-sama dan memerlukan akses yang sama
- **Universe**. Semua user lain dalam system.

Agar sistem diatas bekerja dg baik, keanggotaan group harus dikontrol secara ketat. Sebagai contoh, dalam sistem UNIX, group dapat dibuat dan dimodifikasi hanya oleh manager (superuser).

9.5.3 Contoh Proteksi : UNIX

Pada sistem UNIX, proteksi direktori ditangani sama dengan proteksi file, misalnya, diasosiasikan dengan setiap subdirektory menggunakan owner, group dan universe (others) sebagai 3 bit RWX.

Informasi yang terdapat pada file dari kiri ke kanan terdiri dari proteksi file atau direktori, jumlah link ke file, nama pemilik, nama group, ukuran file dalam byte, tanggal membuat, nama file (Gambar 9-14).

```
-rw-rw-r--   1   pbg   staff   31200  Sep   3   08:30  intro.ps
drwx------   5   pbg   staff     512  Jul   8   09:33  private/
drwxrwxr-x   2   pbg   staff     512  Jul   8   09:35  doc/
drwxrwx---   2   pbg   student   512  Aug   3   14:13  student-proj/
-rw-r—-r--   1   pbg   staff    9423  Feb  24   1993   program.c
-rwxr-xr-x   1   pbg   staff   20471  Feb  24   1993   program
drwx—-x--x   4   pbg   faculty   512  Jul  31   10:31  lib/
drwx------   3   pbg   staff    1024  Aug  29   06:52  mail/
drwxrwxrwx   3   pbg   staff     512  Jul   8   09:35  test/
```

Gambar 9-14 : Proteksi file dan direktori pada UNIX

RINGKASAN:

LATIHAN SOAL :

1. Apakah keuntungan dan kerugian menyimpan nama pembuat program pada atribut file (seperti pada SO Machintosh)
2. Terdapat beberapa metode akses misalnya sequential access dan direct access. Jelaskan !
3. Sebutkan dan jelaskan Tree-structured directory dan acyclic-graph directory
4. Diketahui sebuah system mendukung 5000 user. Misalnya akan mengijinkan 4990 user dapat mengakses sebuah file. Bagaimana spesifikasi proteksi pada UNIX ?