Bab 10

Implementasi Sistem File

POKOK BAHASAN:

- ✓ Struktur Sistem File
- ✓ Implementasi Direktori
- ✓ Metode Alokasi
- ✓ Manajemen Ruang Bebas
- ✓ Efisiensi dan Performansi
- ✓ Perbaikan
- ✓ Sistem File Berstruktur Log
- ✓ Network File Sistem

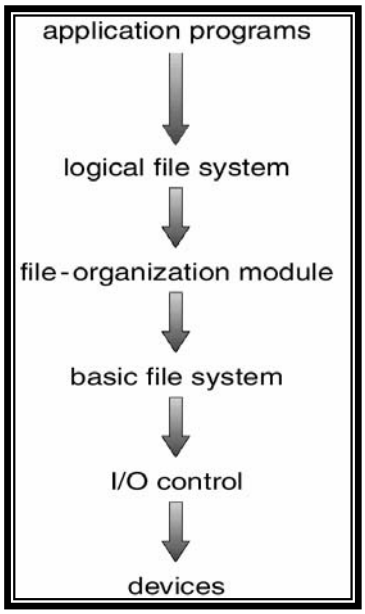
TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami bagaimana implementasi sistem file dan direktori
- ✓ Memahami metode alokasi file dan direktori
- ✓ Memahami bagaimana manajemen ruang bebas
- ✓ Memahami bagaimana efisiensi dan performansi dari sistem file
- ✓ Memahami perbaikan sistem file dari kegagalan sistem
- ✓ Mengetahui beberapa sistem file seperti Log dan Network File Sistem

10.1 STRUKTUR SISTEM FILE

File adalah unit penyimpan logika yang berisi sekumpulan informasi yang berhubungan. Sistem file berada pada penyimpan sekunder (disk). Sistem file diorganisasi ke dalam layer-layer seperti Gambar 10-1.

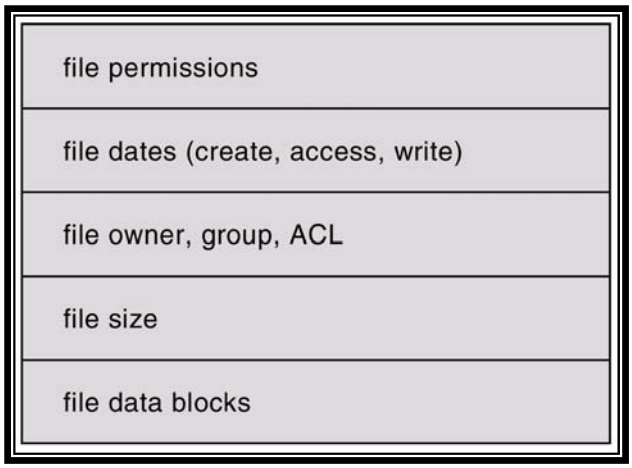


Gambar 10-1 : Layersistem file

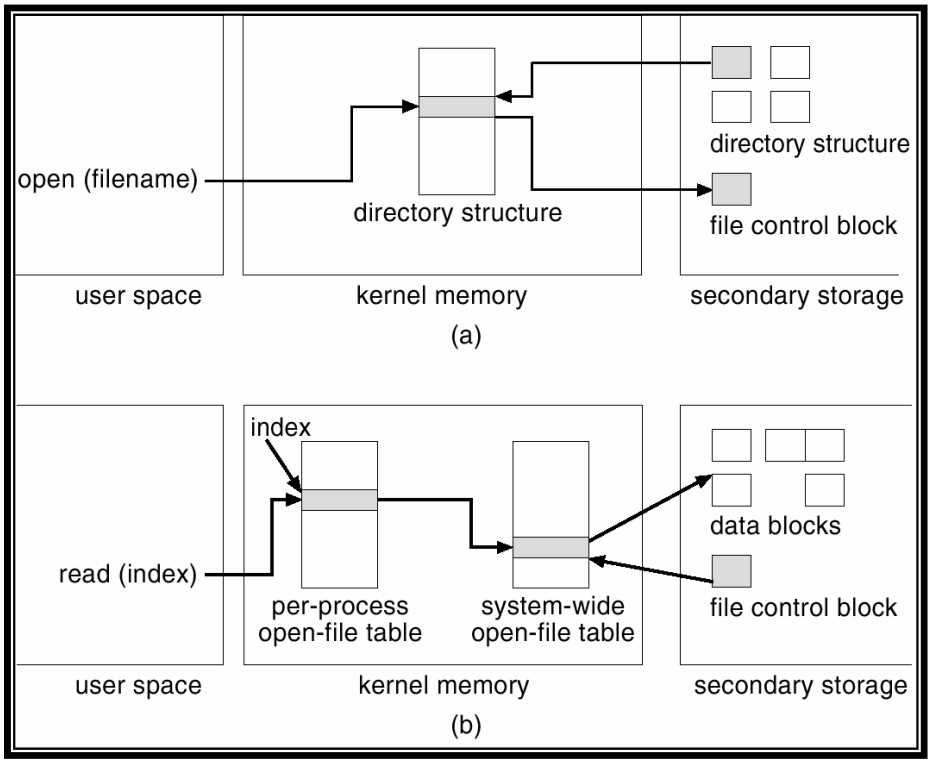
Pada level terendah, *I/O control* berisi *device driver* dan *interrupt handler* untuk mengirim informasi antara memori dan sistem disk. *Basic file system* berisi perintah bagi *device driver* untuk membaca dan menulis blok fisik pada disk. *File organization module* berisi modul untuk mengetahui blok logika pada blok fisik. *Logical file system* menggunakan struktur direktori untuk memberikan ke *file organization module* informasi tentang kebutuhan terakhir.

Informasi mengenai sebuah file disimpan pada struktur penyimpan yang disebut *file control block* seperti Gambar 10-2.

Gambar 10-3 mengilustrasikan pentingnya struktur sistem file disediakan oleh sistem operasi. Pada saat membuka file (dengan menjalankan perintah *open*) blok-blok dari struktur direktori disimpan pada struktur direktori di memori dan mengubah *file control block.* Pada saat membaca file (dengan menjalankan perintah *read*), indeks yang dibaca di cari lokasi blok pada disk melalui tabel open file yang berada di memori.

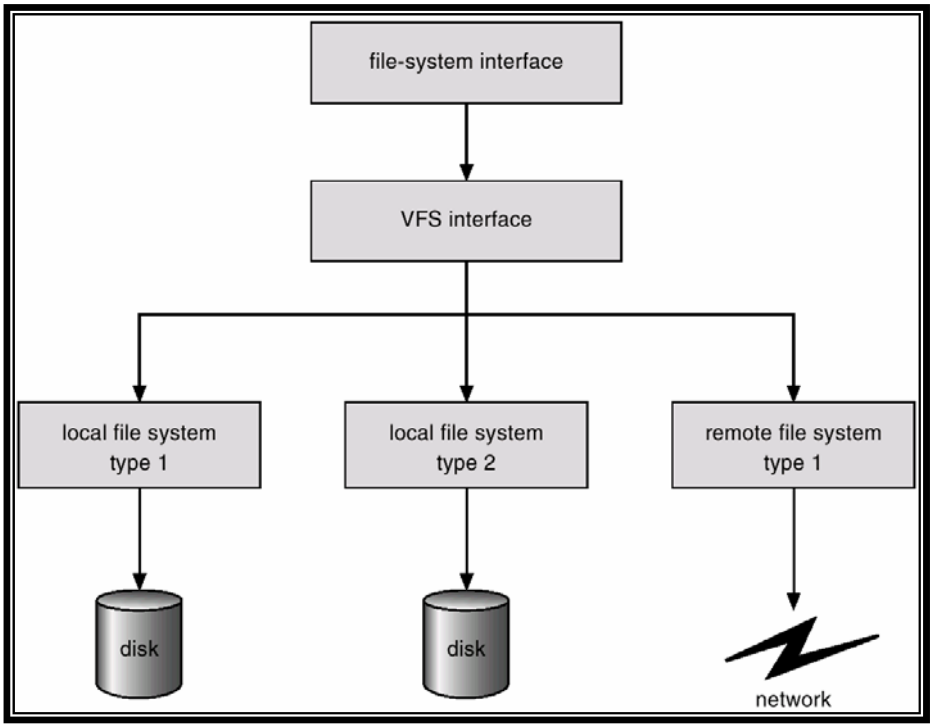


Gambar 10-2 : File control block



Gambar 10-3 : (a) membuka file (b) membaca file

Virtual File Systems (VFS) merupakan implementasi sistem file yang berorientasi obyek. VFS memungkinkan antarmuka system call (API) yang sama digunakan untuk sistem file yang berbeda. API adalah lebih sebagai antarmuka VFS dan bukan untuk tipe sistem file tertentu. Skema VFS dapat dilihat pada Gambar 10-4.



Gambar 10-4 : Skema Virtual File System

10.2 IMPLEMENTASI DIREKTORI

Implementasi direktori menggunakan daftar nama file linier dengan pointer ke blok data. Hal ini berdampak pada pemrograman yang mudah tetapi memerlukan waktu yang lama untuk eksekusi.

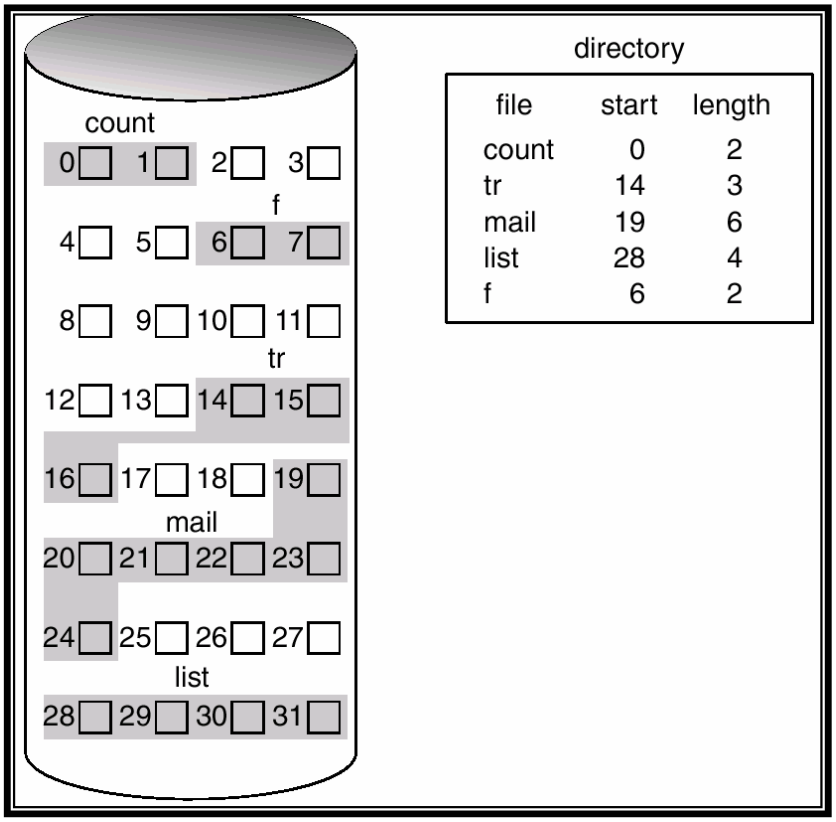
Untuk mempercepat waktu eksekusi digunakan Tabel Hash berupa daftar linier dengan struktur data hash. Dengan struktur data hash akan mengurangi waktu pencarian direktori. Tetapi struktur hash mempunyai resiko bertabrakan apabila terjadi situasi dimana dua nama file hash yang berbeda berada pada lokasi yang sama. Struktur hash berukuran tetap.

10.3 METODE ALOKASI

Metode alokasi berhubungan dengan bagaimana blok-blok pada disk dialokasikan untuk file. Terdapat beberapa metode alokasi antara lain alokasi berurutan (*contiguous allocation*), alokasi berhubungan (*linked allocation*) dan alokasi berindeks (*indexed allocation*).

10.3.1 Alokasi Berurutan (Contiguous Allocation)

Pada alokasi berurutan, setiap file menempati sekumpulan blok yang berurutan pada disk (Gambar 10-5). Model ini sangat sederhana karena hanya membutuhkan lokasi awal (block #) dan panjang (jumlah blok). Akses pada blok disk dilakukan secara random dan memakan banyak ruang (permasalahan dynamic storage-allocation). File yang disimpan secara berurutan tidak dapat berkembang.

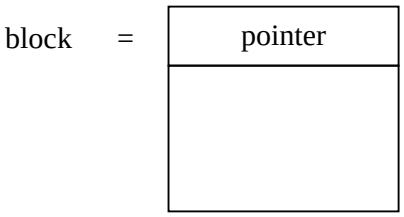


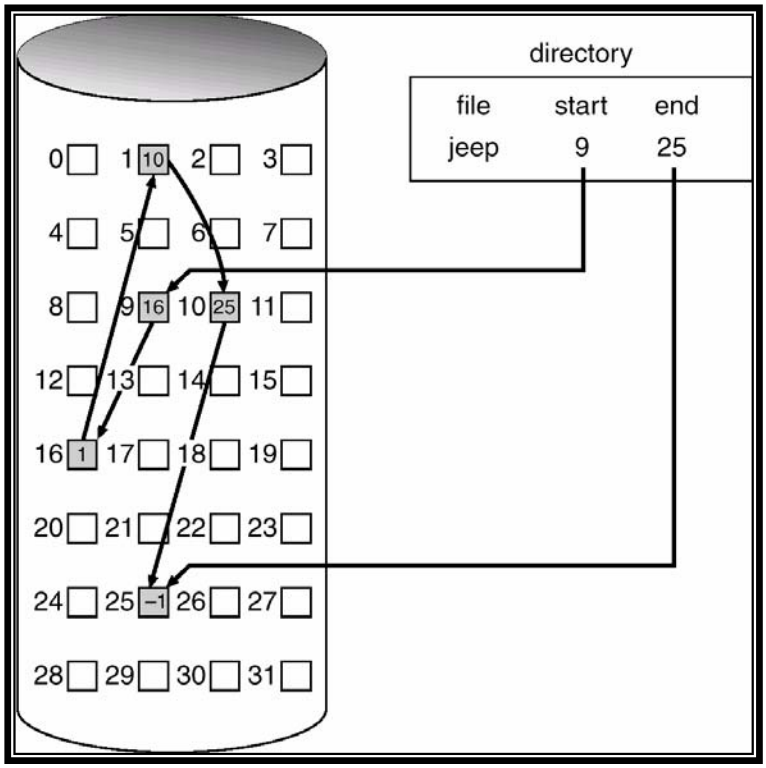
Gambar 10-5 : Alokasi Berurutan

Beberapa sistem file yang baru (misalnya Veritas File System) menggunakan skema alokasi berurutan yang dimodifikasi. File sistem Extent-based mengalokasikan blok pada disk secara berkembang (*extent*). *Extent* adalah blok berurutan pada disk. *Extent* dialokasikan untuk alokasi file. Sebuah file terdiri dari satu atau lebih *extent*.

10.3.2 Alokasi Berhubungan (Linked Allocation)

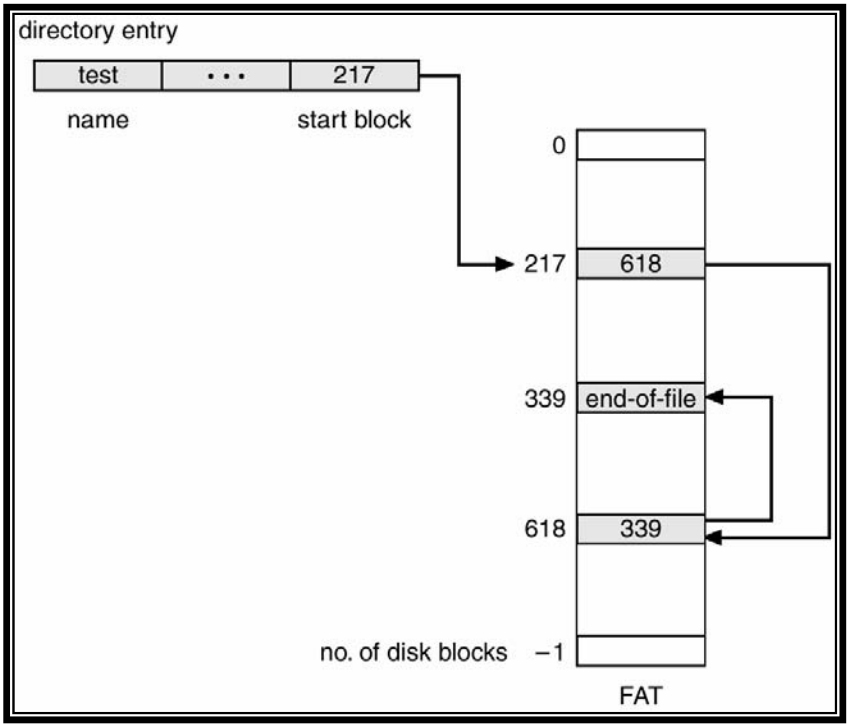
Pada alokasi berhubungan, setiap file adalah sebuah linked list dari blok-blok terpisah pada disk (Gambar 10-6). Pada setiap blok terdapat satu pointer yang menunjuk ke blok lain.





Gambar 10-6 : Alokasi Berhubungan

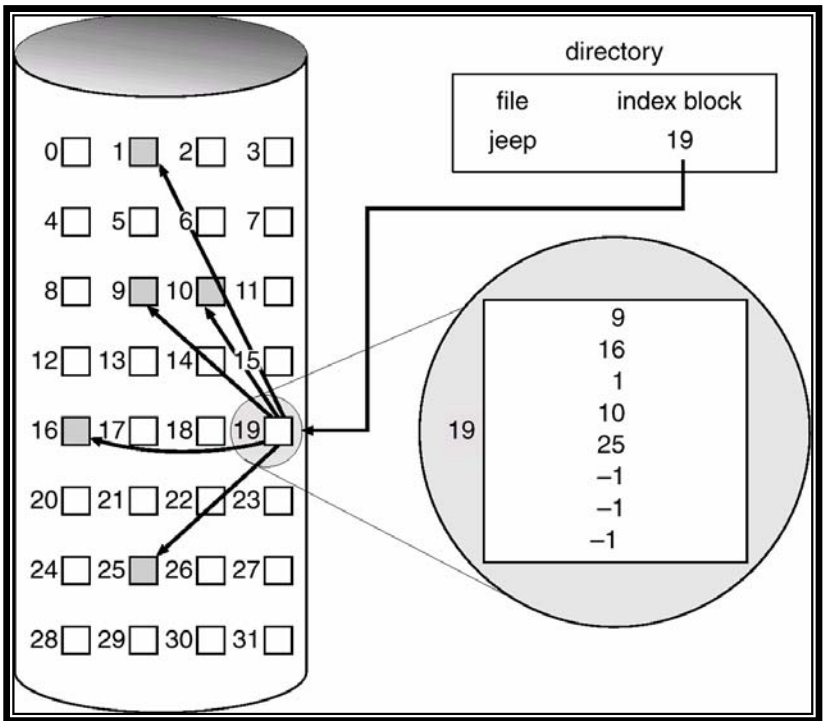
Alokasi berhubungan mempunyai bentuk yang sederhana, hanya memerlukan alamat awal. Sistem manajemen ruang bebas pada alokasi berhubungan tidak memakan banyak ruang. Model ini tidak menggunakan random access. Blok yang diakses adalah blok ke-*Q* pada rantai link dari blok pada file. Perpindahan ke blok = *R* + 1. Contoh sistem file yang menggunakan alokasi berhubungan adalah file-allocation table (FAT) yang digunakan MS-DOS dan OS/2. Bentuk file allocation tabel dapat dilihat pada Gambar 10-7.



Gambar 10-6 : File allocation table

10.3.3 Alokasi Berindeks (Indexed Allocation)

Pada alokasi berindeks, terdapat satu blok yang berisi pointer ke blok-blok file (Gambar 10-7). Alokasi berindeks berupa bentuk logika.

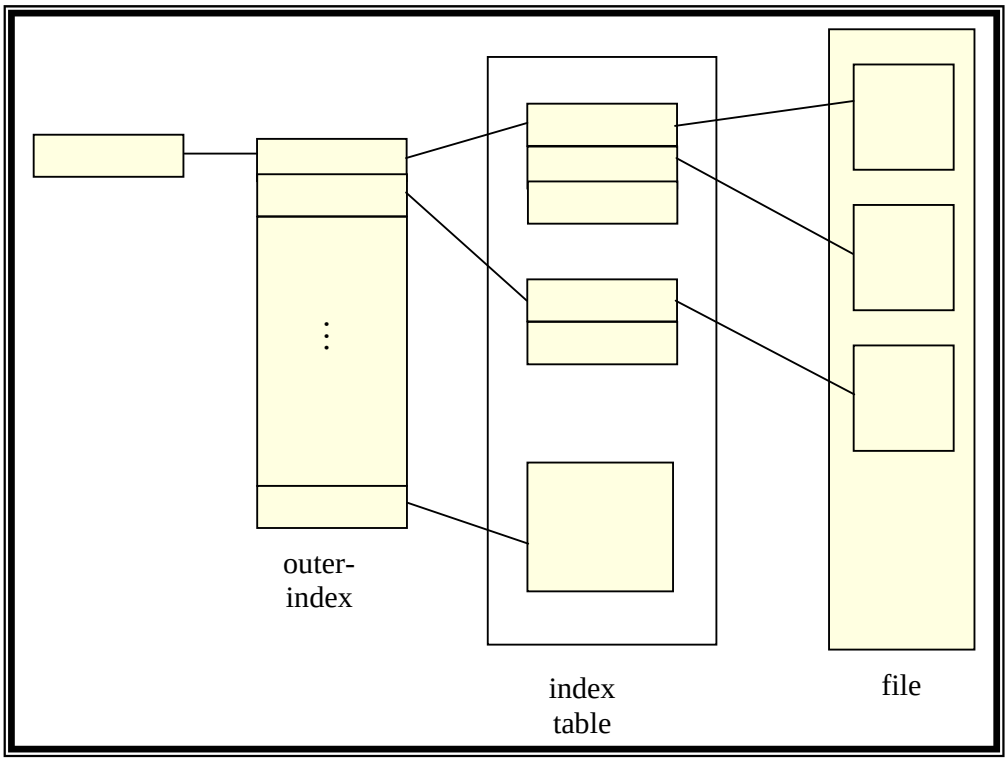


Gambar 10-7 : Alokasi berindeks

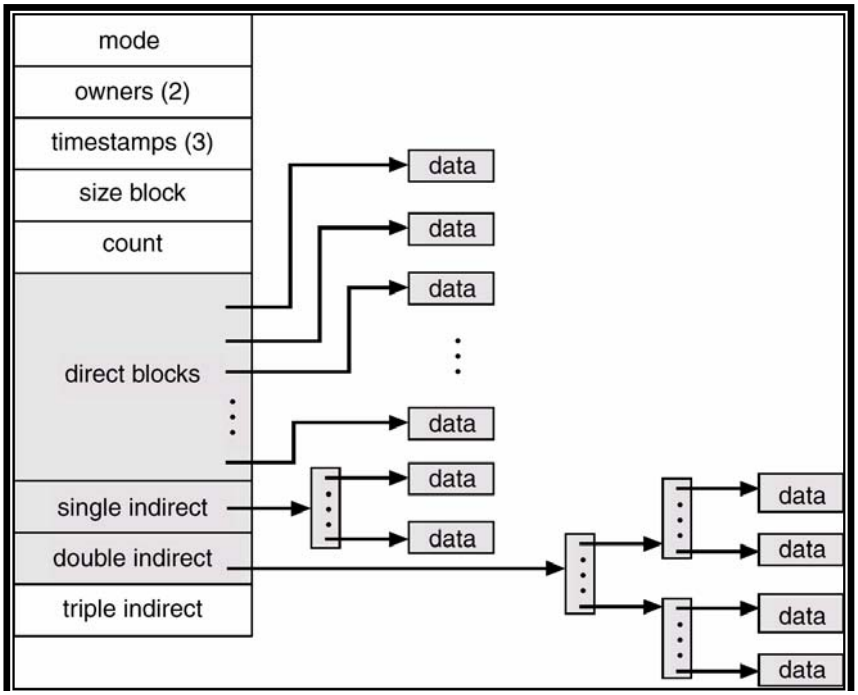
Pada alokasi berindeks, memerlukan tabel indeks yang membawa pointer ke blok-blok file yang lain. Akses dilakukan secara random. Merupakan akses dinamis tanpa fragmentasi eksternal, tetapi mempunyai blok indeks yang berlebih. Pemetaan dari logika ke fisik dalam file ukuran maksimum 256K word dan ukuran blok 512 word hanya memerlukan 1 blok untuk tabel indeks.

Apabila pemetaan dari logika ke fisik dalam sebuah file dari ukuran tak hingga (ukuran blok adalah 512 word) maka digunakan skema menghubungkan blok link dari tabel indeks (ukuran tak terbatas). Untuk ukuran file maksimum 512^3 digunakan skema two-level indeks (Gambar 10-8). Pada skema two-level indeks terdapat tabel indeks luar dan dalam. Indeks dipetakan ke tabel indeks luar kemudian dipetakan ke tabel indeks dalam setelah itu mengakses blok file yang dimaksud.

Sistem operasi UNIX mengimplementasikan kombinasi alokasi berurutan dan alokasi berindeks seperti pada Gambar 10-9.



Gambar 10-8 : Skema two level indeks



Gambar 10-9 : Alokasi pada UNIX

10.4 MANAJEMEN RUANG BEBAS

Daftar ruang bebas biasanya diimplementasikan sebagai *bit map* atai *bit vector* (vektor bit). Setiap blok direpresentasikan dengan 1 bit. Jika blok bebas, maka bit bernilai 1, sebaliknya jika blok dialokasikan, bit bernilai 0. Sebagai contoh, misalnya disk dengan blok 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 dan 27 bebas dan sisanya dialokasikan. Maka bit map dari ruang bebas adalah :

001111001111110001100000011100000…

Perhitungan nomor blok yang bebas adalah sebagai berikut :

(jumlah bit per word) * (jumlah nilai-0 word) + offset dari bit 1 pertama

Pemetaan bit biasanya membutuhkan ruang tambahan, misalnya ukuran blok = 2^{12} byte, ukuran disk = 2^{30} byte (1 gigabyte) maka

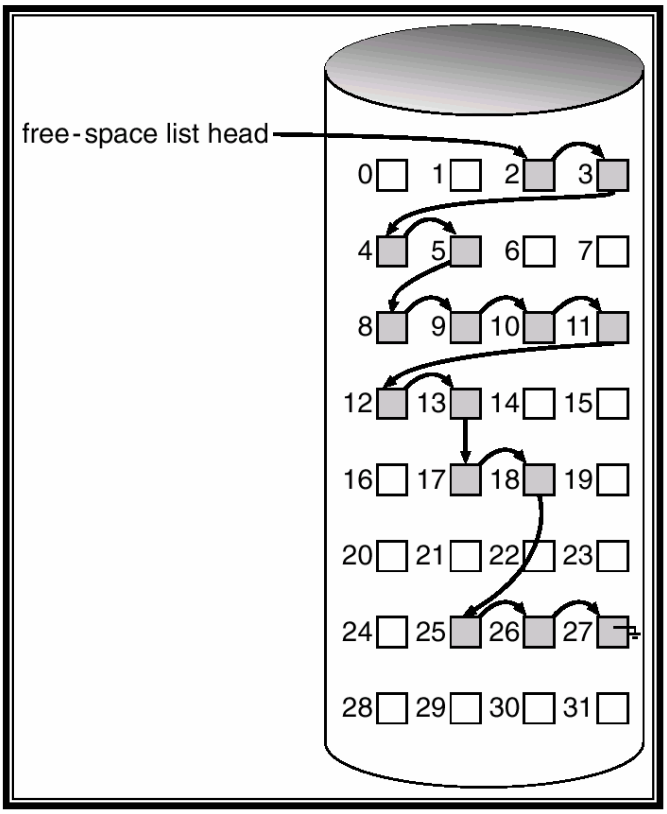
$n = 2^{30}/2^{12} = 2^{18}$ bit (atau 32K byte)

Dengan menggunakan vector bit mudah untuk mendapatkan file yang berurutan. Pengembangan dari vector bit adalah dengan menggunakan linked list (free list) seperti Gambar 10-10. Akan tetapi model ini tidak mendapatkan ruang berurutan dengan mudah meskipun tidak memakan tempat. Modifikasi berikutnya melakukan pengelompokan blok yang bebas agar lebih mudah untuk mendapakan ruang yang berurutan.

10.5 EFISIENSI DAN PERFORMANSI

Efisiensi penggunaan ruang disk tergantung pada alokasi disk dan algoritma directori serta tipe data disimpan pada entry direktory dari file.

Untuk meningkatkan performansi penggunaan ruang disk digunakan disk cache yang digunakan pada bagian terpisah dari main memory untuk penggunaan blok yang sering. Selain itu juga menggunakan teknik untuk optimasi akses berurutan yang disebut *free-behind dan read-ahead*. – teknik untuk optimasi akses berurutan. Untuk meningkatkan performansi PC juga dapat menggunakan bagian tertentu dari memory sebagai virtual disk atau RAM disk.



Gambar 10-10 : Menghubungkan daftar ruang bebas pada disk

10.6 PERBAIKAN

Untuk memperbaiki sistem file dilakukan dengan memeriksa konsistensi dengan cara membandingkan data pada struktur direktori dengan blok data pada disk dan mencoba memperbaiki inkonsistensi. Selain itu juga dapat menggunakan program sistem untuk *back up* data dari disk ke penyimpan lain (floppy disk, magnetic tape). Perbaikan akan Recover menghilangkan file atau disk dengan *restoring* data dari backup.

10.7 SISTEM FILE LOG STRUCTURED

Sistem file **Log structured** (atau journaling) menyhimpan semua update ke file sistem sebagai **transaksi.** Semua transaksi ditulis ke **log**. Sebuah transaksi dijadikan **committed** jika ditulis ke log. Tetapi, kemngkinan sistem file tidak diupdate

Transaksi pada log ditulis secara tidak beraturan ke sistem file. Jika sistem file dimodifikasi, transaksi dihapus dari log. Jika file bertabrakan, semua transaksi yang tersisa pada log harus dibentuk.

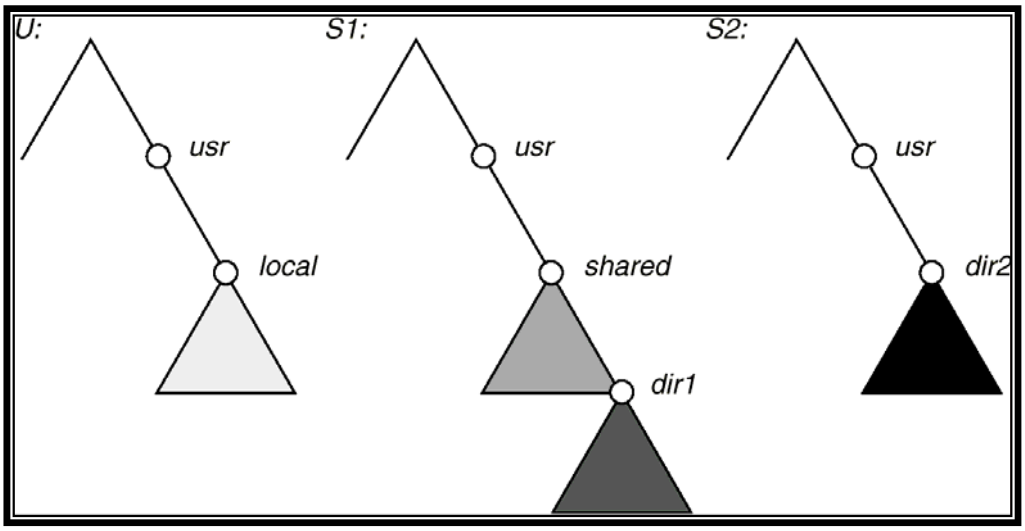
10.8 SUN NETWORK FILE SYSTEM (NFS)

Network file system adalah implementasi dan spesifikasi dari sistem perangkat lunak untuk mengakses remote files melalui LAN (atau WAN). NFS merupakan bagian dari Solaris and SunOS yang berjalan pada Sun workstations menggunakan unreliable datagram protocol (UDP/IP) protocol dan Ethernet.

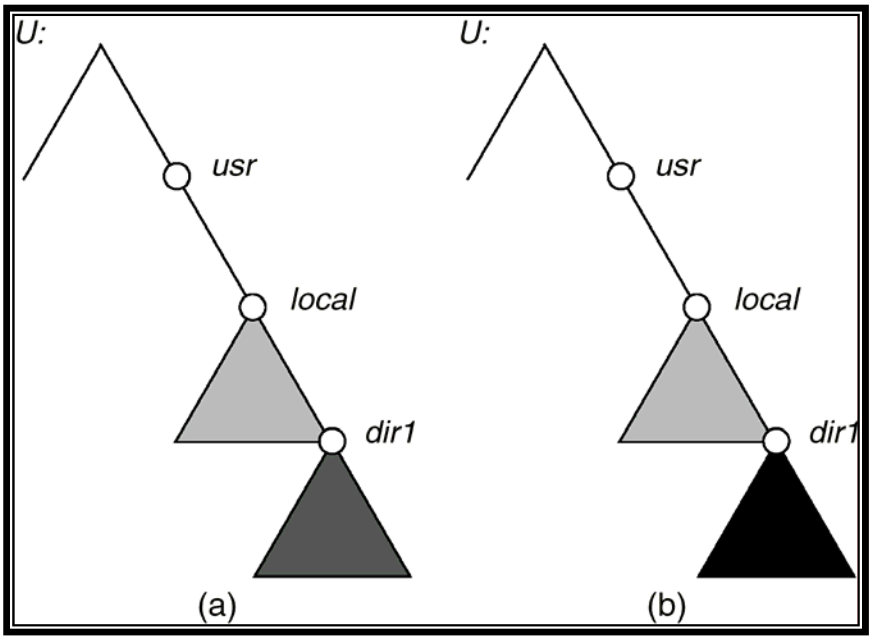
Workstation yang saling berhubungan dipandang sebagai mesin independent dengan file sistem yang independent, memungkinkan sharing diantara sistem file secara transparent. Directory remote di-mount ke directory sistem file lokal. Mounted directory terlihat sebagai subtree dari sistem file lokal, mengubah subtree secara descending dari directory lokal. Spesifikasi dari remote directory untuk operasi mount tidak transparant; host name dari remote directory harus disediakan. File pada remote directory dapat diakses secara transparant. Subyek ke akreditasi akses yang benar, sembarang sistem file (atau directory dalam sistem file), dapat di-mount secara remote ke top dari sembaran directory lokal.

NFS didesain untuk operasi pada lingkungan heterogen dari mesin, SO dan arsitektur network yang berbeda; spesifikasi NFS tidak tergantung dari media tersebut. Ketidak tergantungan dilakukan melalui penggunaan RPC pada bagian tertinggi dari protokol External Data Representation (XDR) yang digunakan antara 2 antarmuka independent. Spesifikasi NFS berbeda antara layanan tersedian dengan mekanisme mount dan layanan akses file remote actual.

Misalnya terdapat tiga file sistem yang independent seperti Gambar 10-11. Kemudian dilakukan mount dengan NFS maka file sistem hasil seperti Gambar 10-12.



Gambar 10-11 : Tiga sistem file yang independen



Gambar 10-12 : Mounting pada NFS

LATIHAN SOAL :

1. Sistem file biasanya diimplementasikan dalam struktur layer atau modular. Jelaskan struktur layer pada system file.
2. Ada beberapa cara file dialokasikan pada ruang disk, yaitu contiguous, linked atau berindeks. Jelaskan ketiga cara alokasi file diatas dan berikan contoh.
3. Sebutkan dan jelaskan cara untuk memperbaiki sistem dari kegagalan sehingga tidak kehilangan data atau data inconsistency.
4. Apakah permasalahan yang timbul bila sebuah system memperbolehkan system file di-mount secara simultan lebih dari satu lokasi ?