BAB II

LANSADASAN TEORI

2.1 Konsep File

File adalah sekumpulan informasi yang berhubungan dan tersimpan dalam *secondary storage*, tipe file dibagi menjadi 2 yaitu data (karakter, numeric, binary, dll) dan program. File juga mempunyai atribut yang meliputi nama file, tipe file, lokasi, ukuran, waktu pembuatan dan identitas pembuat, proteksi dan informasi tentang file.

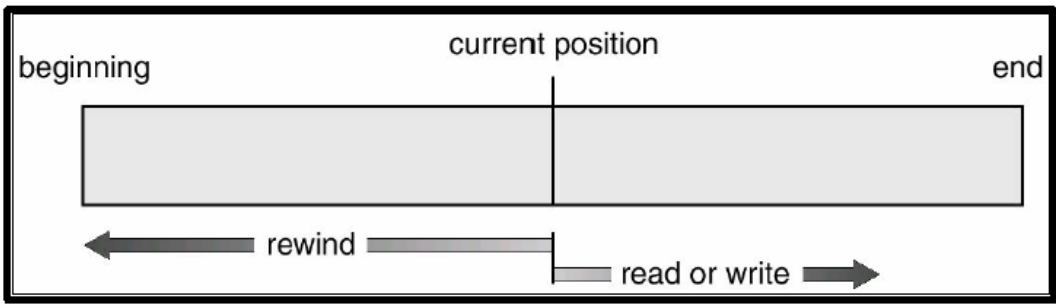
Operasi – operasi pada file antara lain membuat, menulis, membaca, menghapus, mencari, membuka, menutup, menghapus dengan menyisakan atribut

2.1.1 Struktur File

a. Sistem operasi membutuhkan struktur file tertentu untuk menjalankan atau mengakses suatu file.
b. Semua sistem operasi diharuskan mampu mengenal sedikitnya satu jenis struktur file.
c. Jika sistem operasi semakin banyak mengenal struktur file, maka semakin luas aplikasi yang dapat dijalankan namun ukuran sistem operasi akan semakin membengkak. Sebaliknya jika semakin sedikit struktur file, maka sistem operasi hanya dapat menjalankan aplikasi dalam jumlah yang sedikit pula

2.1.2 Metode Akses File

2.1.2.1. *Sequential Access*



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

Gambar 2.1 Metode *Sequential Access*

Dimana n adalah *relative block number*

2.1.2.2. *Direct Access*

a. Sangat berguna unutk pengaksesan langsung informasi dalam jumlah besar, seperti database
b. File dapat dilihat sebagai deretan blok yang berindeks
c. Relative Block Number digunakan agar memungkinkan sistem operasi untuk memutuskan dimana suatu file dapat ditempatkan, dan mencegah user untuk mengakses sebagian dari sistem file yang bukan merupakan bagian dari miliknya

Metode akses lainnya dibangun / dikembangkan berdasarkan *direct access method*

biasa melibatkan proses pembuatan indexks dari file. Untuk mencari suatu bagian dari file, pertama – tama cari indeksnya, kemudian dengan *pointer* tersebut kita mengakses file secara langsung, lalu mencati bagian file yang dibutuhkan

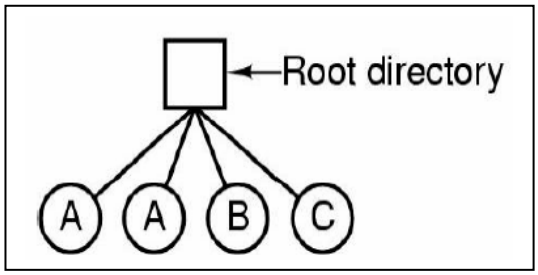
2.2 Konsep Direktori

Struktur direktori dapat dibagi menjadi 5 bagian sebagai berikut

a. *Single – Level Directory*

Semua file terdapat didalam direktori yang sama

Tiap file memiliki nama yang unik

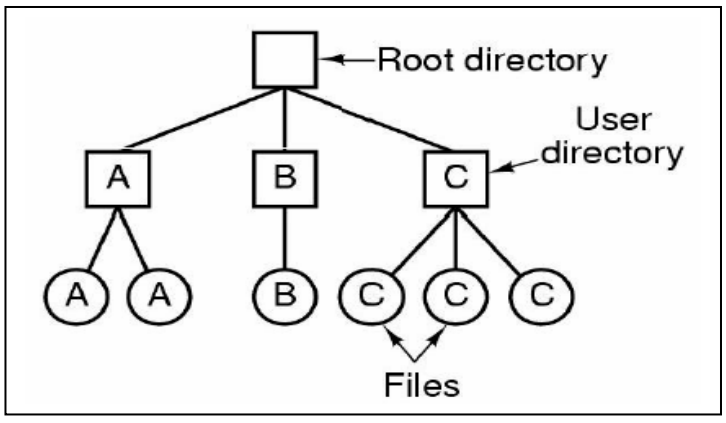


Gambar 2.2 *Single – Level Directory*

b. *Two – Level Director*

Membuat direktori yang terpisah untuk tiap user

Terdapat *User File Directory* dan *Master File Directory*



Gambar 2.3 *Two – Level Directory*

c. *Tree – Structured Directory*

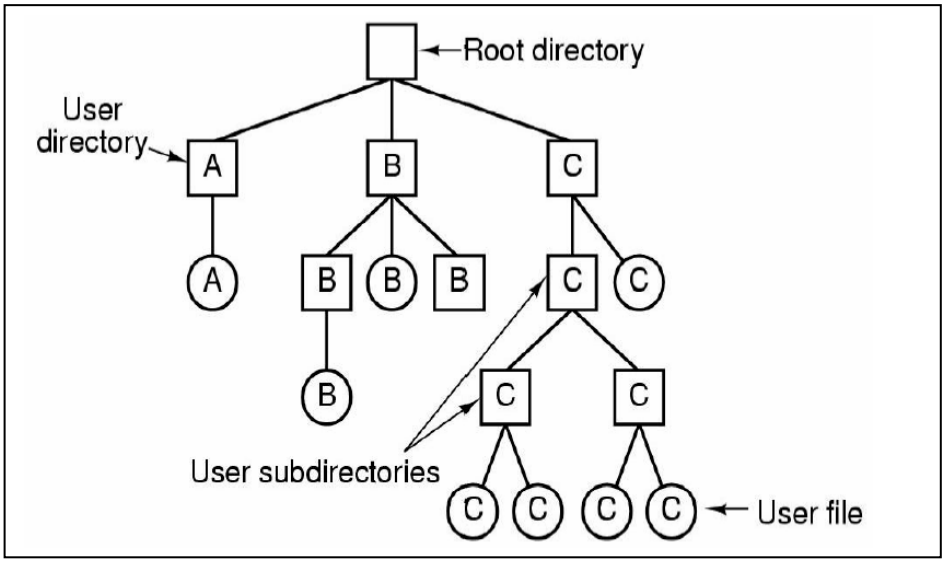
Tiap direktori dapat mengandung file dan subdirektori

Path (*absolute path*) adalah urutan direktori yang berasal dari *MFD (Master File Directory).*

Working Dir (relative path) adalah path yang berasal dari *current path*

Current Directory adalah direktori yang baru – baru ini digunakan

Contoh absolute path : UserZ/Word/UnitD/XYZ



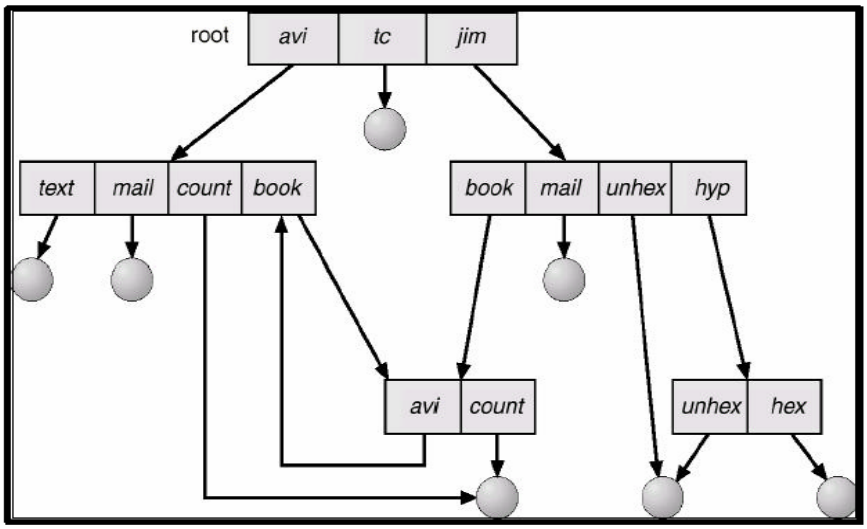
Gambar 2.4 *Tree – Structured Directory*

d. *Acyclic – Graph Directory*

Satu file dapat memiliki *absolute* path yang berbeda

Masalah yang akan timbul dalam struktur ini adalah *dangling pointer*, solusinya adalah backpointers, agar kita dapat menghapus semua pointer, backpointers menggunakan struktur *daisy chain*

e. *General – Graph Director*



Gambar 2.5 *General – Graph Directory*

Meyakinkan tidak adanya siklus, hanya mengizinkan link ke file, menggunakan alogoritma siklik dalam mendeteksi setiap ada link baru yang ditambahkan

2.3 Kontrol Akses File

a. *Access Control List (ACL),* yaitu suatu file atau direktori berasosiasi dengan suatu username dan tipe data
b. *Owner*, *Group* dan *Universe* Kontrol Akses
c. Klasifikasi User dalam mengakses suatu file
d. *Owner* : user yang menciptakan file tersebut
e. *Group* : sekelompok user yang saling berbagi file dan tergabung dalam sebuah kelompok kerja
f. *Universe* : semua user yang saling terhubung dalam sistem
g. Implementasi kontrol akses yang sering digunakan merupakan kombinasi keduanya

2.4 Struktur dan Implementasi File *System*

2.4.1 Struktur file *system*

Karakteristik penting dari disk :

a) Disk tersebut dapat ditulis ulang di disk tersebut, hal ini memungkinkan untuk membaca, memodifikasi dan menulis di disk tersebut
b) Dapat diakses langsung ke setiap *block* di disk, hal ini memeudahkan untuk mengakses setisp file baik secara berurut maupun tidak berurut, dan berpindah dari satu file ke file yang lain dengan hanya mengangkat head disk dan menunggu disk berputar

Organisasi file *system*

a) Masalah desain dalam membangun file *system*
b) Definisi dari file *system* mencangkup definisi file dan atribunya, operasi ke file dan struktur direktori dalam mengorganisasikan file
c) Membuat algoritma dan struktur data yang memetakkan struktur *logical file system* ke tempat penyimpanan sekunder
d) *I/O Contol* (driver *device dan interrupt handler*), driver *device* adalah komunikasi antara sistem operasi dengan perangkat keras
e) *Basic File System*, mengeluarkan perintah *generic* ke *device* driver baca dan tulis pada suatu *block* dalam disk
f) *File Organization Module*, Informasi tentang *logical address* dan *physical address* dari file tersebut, mengatur juga sisa disk dengan melacak alamat yang belum dialokasikan dan menyediakan alamat tersebut saat user ingin menulis file kedalam disk

g) *Logical File System*, tingkat ini berisi informasi tentang symbol nama file, struktur dari direktori, proteksi dan sekuriti dari file tersebut

2.5 Linux

Linux adalah *Unix clone*, kernel yang ditulis oleh Linus Torvalds dan dikembangkan dengan bantuan programmer dan *hacker*, dari seluruh dunia. Linux mempunyai semua fitur yang dimiliki oleh Unix, termasuk *multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write esestubles,* proper memory management dan TCP/IP *networking*

Linux merupakan sistem operasi yang baik dan banyak digunakan sebagai server dikarenakan sistem operasi ini telah teruji aman, cepat dan tangguh

Direktori root Linux memiliki beberapa direktori yang merupakan standar direktori pada banyak distro Linux. Direktori – direktori tersebut antara lain :

1 /bin, mengandung perintah-perintah yang dapat digunakan baik itu oleh sistem administrasi dan oleh users. Dapat juga mengandung perintah-perintah yang dipakai secara tidak langsung oleh scripts./dev berisi file sistem khusus, yang merupakan refleksi *device hardware* yang dikenali dan digunakan sistem

2 /boot, mengandung semua yang diperlukan untuk proses boot kecuali konfigurasi file dan map installer. /boot menyimpan data yang digunakan sebelum kernel mulai mengeksekusi program user-mode. /boot juga berisi sektor saved master boot, sektor map files, dan data lain yang tidak di-edit langsung dengan tangan./etc berisi file – file konfigurasi sistem, biasanya hanya boleh dirubah oleh super *user*

3 /home berisi direktori – direktori yang merupakan direktori home untuk *user* biasa dan aplikasi tertentu

4 /lib adalah direktori yang mengandung *shared library* image yang diperlukan untuk mem-boot sistem dan menjalankan perintah dalam root file system, contoh: dengan binari dalam /bin dan /sbin /mnt direktori khusus yang disediakan untuk mounting (mengaitkan) *device disk storage* ke sistem dalam bentuk direktori

5 /proc berisikan file – file sistem khusus yang menunjukkan data – data *kernel* setiap saat

6 /root direktori *home* untuk *user* root

7 /sbin, utilitas yang digunakan untuk sistem administrasi (dan perintah root-only yang lain) disimpan di /sbin/, /usr/sbin, dan /usr/local/sbin. /sbin mengandung binari esensial untuk boot, meyimpan kembali, mengembalikan seperti semula, dan/atau membetulkan sistem dalam penambahan ke binari dalam /bin./temp berisi file – file sementara yang dibutuhkan oleh sebuah aplikasi yang sedang berjalan

8 /usr adalah *second major section file system*. /usr adalah data *read-only* yang dapat digunakan bersama-sama. Hal ini menandakan bahwa /usr harus bisa dipakai bersamasama oleh various FHS-compliant hosts dan tidak boleh ditimpa. Informasi yang host-specific atau bervariasi dengan waktu disimpan dimana saja./var berisikan file –file *log, mailbox* dan data – data aplikasi

9 /dev adalah lokasi dari suatu file khusus piranti keras.

10 /etc mengandung konfigurasi file dan direktori yang spesifik ke current system, tidak ada binari yang bolah diletakkan di bawah /etc

11 /mnt, direktori ini disediakan agar sistem administrasi bisa secara sementara me-mount filesystem sesuai dengan yang dibutuhkan. Isi dari direktori ini adalah isu lokal dan tidak boleh mempengaruhi manner dimana program sedang berjalan.. Direktori ini tidak boleh digunakan untuk instalasi program: direktori sementara yang tepat, yang sedang tidak digunakan oleh sistem, harus digunakan.

12 /opt, disediakan untuk instalasi dari tambahan untuk paket aplikasi piranti lunak. Paket yang akan diinstalasi di /opt harus menempatkan file statis di direktori tree /opt/<paket> yang berbeda, dimana <paket> adalah nama yang mendeskripsikan paket piranti lunak.

13 /tmp, direktori /tmp harus dibuat siap pakai untuk program yang membutuhkan file sementara. Program tidak boleh berasumsi bahwa file atau direktori dalam /tmp disediakan diantara invokasi dari program.

14 /var berisi variabel file data. Variabel file data ini mengandung direktori spool dan file, administratif dan logging data, dan file sementara.

2.5.1 Operasi pada File dan Direktori

2.5.1.1 Operasi File

Tabel 2.1 Operasi pada File

Operasi	Fungsi
cat	Menampilkan isi file
cp	Menyalin satu atau beberapa file
find	Mencari file tertentu pada suatu direktori

ls	Menampilkan informasi dari file
more	Menampilkan isi suatu file teks per layar
mv	Memindahkan file ke direktori lain atau mengubahkan nama file
pg	Menampilkan isi suatu file teks per layar
rn	Menghapus file

2.5.1.2 Operasi Direktori

Tabel 2.2 Operasi pada Direktori

Operasi	Fungsi
cd	Mengaktifkan suatu direktori sebagai direktori kerja.
copy	Menyalin seluruh struktur
mkdir	Membuat direktori baru
pwd	Menampilkan nama direktori kerja.
rmdir	Menghapus direktori.
mv	Mengubah nama direktori

2.5.2 Struktur File

Adapun struktur dari file dapat dilihat seperti berikut ini:

```
struct file {
        struct list_head        f_list;
        struct dentry           *f_dentry;
        struct file_operations  *f_op;
        atomic_t                f_count;
        unsigned int            f_flags;
        mode_t                  f_mode;
        loff_t                  f_pos;
        unsigned long           f_reada,f_ramax,f_raend,f_ralen,f_rawin;
        struct fown_struct      f_owner;
        unsigned int            f_uid,f_gid;
        int                     f_error;
```

```
        unsigned long           f_version;
        void                    *private_data;
};
```

Penjelasan dari struktur file diatas:

- f_list : menyambungkan semua file ke dalam beberapa list
- f_dentry : pointer yang menunjuk pada objek dentry yang berkaitan
- f_op : pointer yang menunjuk pada tabel operasi tabel
- f_mode : mode proses akses
- f_pos : current file offset (pointer file)
- f_count : counter penggunaan obyek file
- f_flags : flag yang dibuat saat file dibuka
- f_reada : flag yang digunakan untuk *read* head
- f_ramax : jumlah maksimum page yang akan di-*read ahead*
- f_raend : pointer file setelah *read ahead* terakhir
- f_ralen : jumlah byte yang di-*read ahead*
- f_rawin : jumlah page yang di-*read ahead*
- f_owner : data untuk I/O asynchronous melalui signal
- f_uid : ID user pemilik file
- f_gid : ID grup user pemilik file
- f_error : kode kesalahan untuk operasi *write* network

2.5.3 *Virtual File System* pada Linux

Kernel Linux telah mengembangkan VFS (*Virtual File System*) yang dapat mengenali data yang menggunakan File Sistem lain. File Sistem yang dikenali oleh Virtual File System Linux terbagi menjadi 3 jenis, yaitu:

1. *Disk Based File System*

   Tipe file sistem ini mengatur *space* memori yang bisa digunakan pada partisi disk *local*. Tipe file sistem ini yang umum adalah Ext2. Tipe lain yang dikenal dengan baik oleh

   VFS adalah :

   1. File *system* bagi varian Unix seperti system V dan BSD.
   2. Microsoft *file system* seperti MS-DOS, VFAT (Windows 98) dan NTFS (Windows NT).
   3. *File system* ISO96660 CD-ROM.
   4. *File ssystem* lain seperti HPFS (IBM's, OS/2), HFS (Apple Machintosh), FFS (Amiga's Fast Filesystem) dan ADFS (Acorn's machines).

2. *Network File System*

   Tipe sistem file ini memungkinkan akses yang mudah ke suatu file yang terdapat pada jaringan komputer lain. Beberapa *file system* jenis ini yang dikenal dengan baik oleh VFS adalah : NFS, Coda, AFS (*Andrews File System*), SMB (*Microsoft's Windows dan IBM's OS/2 LAN Manager*) dan NCP (Novell's *NetWare Core Protocol*).

3. *Special File System*

Tipe ini tidak mengijinkan mengatur space disk. Pada direktori /proc menyediakan interface yang mengijinkan user untuk mengakses struktur data kernel. Direktori /dev/pts digunakan sebagai pendukung terminal semu. Seperti yang digambarkan pada standar Open Group's Unix98.

2.5.4 *Mounting* dan *Unmounting*

Agar suatu file system dapat dikenali oleh Virtual File System Linux, perlu dilakukan suatu proses yang disebut mounting.

Proses mounting sebenarnya adalah merepresentasikan file yang terdapat pada device eksternal (misal: disket) yang menggunakan file sistem lain menjadi inode sementara agar dapat dibaca seperti layaknya file lainnya oleh Virtual File System Linux.

Jika kita melakukan perubahan pada file (menambah, mengurangi, mengganti), hal tersebut tidak dilakukan langsung pada device tapi disimpan dalam media sementara. Untuk meyimpan perubahan, harus dilakukan proses unmounting, yaitu menghapus inode sementara yang dipakai sebelumnya dan menyimpan perubahan (jika ada) yang telah dilakukan pada device. Karena itu proses unmounting sangat perlu dilakukan.

2.5.5 Linux *Extented File System*

Sejarah perkembangan Extended File System

Versi mLinux yang pertama berbasis pada file sistem Minix. Setelah Linux semakin berkembang, *Extended File System* (*Ext FS*) diperkenalkan. Ada beberapa perubahan signifikan tetapi kinerjanya masih kurang memuaskan. Pada

tahun 1994 *Second Extended Filesystem (Ext2)* diperkenalkan. Di samping adanya beberapa fitur baru, Ext2 sangat efisien, handal dan fleksibel sehingga menjadi file sistem Linux yang paling banyak digunakan.

2.5.5.1 Linux *Second Extended File System* (Ext2FS)

Untuk sebagian besar user dan system administration tasks yang umum, file dan direktori mudah untuk diterima seperti struktur pohon. Komputer bagaimanapun tidak bisa melihat hal tersebut seperti struktur pohon.

Setiap partisi memiliki sistem file sendiri. Dengan membayangkan sistem file bersamaan, kita dapat membentuk sebuah ide mengenai struktur pohon dari seluruh sistem, tapi tidak sesederhana itu. Dalam sebuah sistem file, file direpresentasikan dengan *inode*, sejenis nomor seri unik yang berisi informasi tentang data sebenarnya yang membentuk sebuah file: milik siapa file tersebut, dan dimana file tersebut terletak pada harddisk.

Setiap partisi memiliki himpunan *inode* tersendiri, pada sistem yang memiliki banyak partisi, bisa terdapat beberapa file dengan nomor *inode* yang sama.

Setiap *inode* menggambarkan struktur data pada harddisk, menyimpan properti dari file, termasuk lokasi fisik dari data file. Ketika harddisk disiapkan untuk menerima peyimpanan data, biasanya selama proses instalasi sistem awal atau ketika menambahkan disket tambahan ke dalam sistem yangada, sejumlah *inode* per partisi yang pasti diciptakan. Jumlah ini akan menjadi jumlah maksimum file, dari berbagai tipe (termasuk direktori, file khusus, link, dll.) yang

dapat muncul pada saat yang sama pada sebuah partisi. Pada umumnya terdapat 1 *inode* setiap 2 sampai 8 KB.

Mekanisme Second Extended File System (Ext2FS)

Ext2fs menggunakan mekanisme yang mirip dengan BSD Fast File System (ffs) dalam mengalokasikan blok-blok data dari file, yang membedakan adalah :

a. Pada ffs, file dialokasikan ke disk dalam blok sebesar 8KB, dan blok-blok itu dibagi menjadi fragmen-fragmen 1KB untuk menyimpan file-file berukuran kecil atau blok-blok yang terisi secara parsial di bagian akhir file.
b. Ext2fs tidak menggunakan fragmen, pengalokasian dalam unit-unit yang lebih kecil. Ukuran blok secara default pada ext2fs adalah 1KB, meskipun mendukung juga pengalokasian 2KB dan 4KB.
c. Alokasi pada Ext2fs didesain untuk menempatkan blok-blok lojik dari file ke dalam blok-blok fisik pada disk, dengan demikian I/O request untuk beberapa blok-blok disk secagai operasi tunggal.

Kehandalan *Second Extended File System* Ext2FS

a. Administrator sistem dapat memilih ukuran blok yang optimal (dari 1024 sampai 4096 bytes), tergantung dari panjang file rata-rata, saat membuat file sistem.
b. Administrator dapat memilih banyak *inode* dalam setiap partisi saat membuat file sistem.
c. Strategi update yang aman dapat meminimalisasi dari *system crash*.

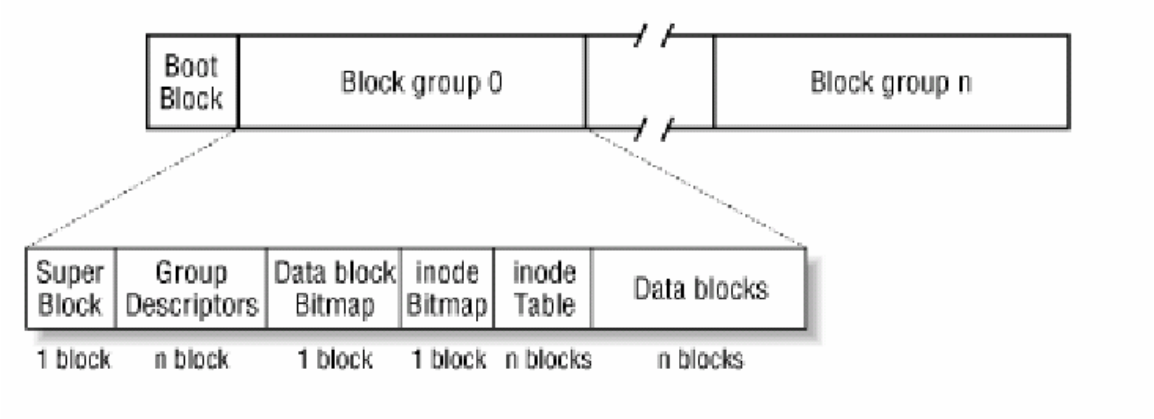
d. Mendukung pengecekan kekonsistensian otomatis saat booting.
e. Mendukung file *immutable* (file yang tidak dapat dimodifikasi)dan *append-only* (file yang isinya hanya dapat ditambahkan pada akhir file tersebut).

Informasi yang Disimpan pada Inode

1 Device tempat inode berada
2 Mode file
3 Locking information
4 Pemilik dan grup pemilik dari file tersebut.
5 Jenis file (regular, direktori, dll.)
6 Hak akses atas file.
7 Waktu pembuatan, pembacaan, dan perubahan terakhir.
8 Waktu perubahan informasi pada *inode*.
9 Jumlah link yang menunjuk ke file ini.
10 Ukuran file.
11 Alamat yang menunjukan lokasi sebenarnya dari data file.

Satu – satunya informasi yang tidak tersimpan pada *inode* adalah nama file dan direktori. Informasi ini tersimpan pada file direktori khusus. Dengan membandingkan nama file dan nomor *inode*, sistem dapat membangun struktur pohon yang dapat dimengerti user. User dapat melihat nomor *inode* dengan menggunakan opsi –i pada perintah ls. Masing-masing *inode* memiliki ruang memori yang terpisah pada disk.

Pembagian Blok



Gambar 2.6 Pembagian Blok

Layout dari partisi dan group block Ext2FS
Keterangan:

Setiap partisi terbagi menjadi:

1 *Boot block*, yang merupakan blok pertama, dipakai untuk *booting*, sehingga tidak diurusi oleh Ext2FS.
2 *Block group* sebanyak n buah. Setiap *block group* berukuran sama dan terdiri dari:
    a) *Super Block*, disimpan dalam struktur ext2_super_block
    b) *Group Descriptor*, disimpan dalam bentuk xt2_group_desc
    c) Data Block Bitmap
    d) Inode Bitmap
    e) *Inode Table*, terdiri dari kumpulan block yang berurutan, dan masing-masing blok mengandung sejumlah inode yang terlah terdefinisi sebelumnya. Semua inode memiliki ukuran sama.
    f) *Data Blocks*, menyimpan data sebenarnya dari file.

Struktur *inode* dalam Linux diimplementasikan sebagai berikut :

```
struct inode {
        struct list_head i_hash;
        struct list_head i_list;
        struct list_head i_dentry;
        unsigned long i_ino;
        unsigned int i_count;
        kdev_t i_dev;
        umode_t i_mode;
        nlink_t i_nlink;
        uid_t i_uid;
        gid_t i_gid;
        kdev_t i_rdev;
        off_t i_size;
        time_t i_atime;
        time_t i_mtime;
        time_t i_ctime;
        unsigned long i_blksize;
        unsigned long i_blocks;
        unsigned long i_version;
        unsigned long i_nrpages;
        struct semaphore i_sem;
        struct inode_operations *i_op;
        struct super_block *i-sb;
        wait_queue_head_t i_wait;
        struct file_lock *i_flock;
        struct vm_area_struct *i_mmap;
        struct page *i_pages;
        spinlock_t i_shaerd_lock;
        struct dquot *i_dquot(MAXQUOTAS);
        struct pipe_inode_info *i_pipe;
        unsigned long i_state;
```

```
        unsigned long i_flags;
        unsigned char i_sock;
        atomic_t i_writecount;
        unsigned int i_attr_flags;
        _u32 i_generation;
        union {
        ...
        struct extfs_inode_info ext2_i;
        ...
        struct socket socket_i;
        void *generic)ip;
        }u;
};
```

2.5.5.2 Linux *Third Extended File System* (Ext3FS)

Ext3FS merupakan pengembangan dari Ext2FS. Ext3FS memiliki beberapa kelebihan antara lain:

1 Optimasi waktu pengecekan jika terjadi kegagalan sumber daya, kerusakan sisem atau *unclean shutdown*.

2 Setelah mengalami kegagalan sumber daya, *unclean shutdown*, atau kerusakan sistem, Ext2FS harus melalui proses pengecekan. Proses inidapat membuang waktu sehingga proses *booting* menjadi sangat lama, khususnya untuk *disk* besar yang mengandung banyak sekali data. Dalam proses ini, semua data tidak dapat diakses. Jurnal yang disediakan oleh *EXT3* menyebabkan tidak perlu lagi dilakukan pengecekan data setelah kegagalan sistem. *EXT3* hanya dicek bila ada kerusakan *hardware* seperti kerusakan *hard disk*, tetapi kejadian ini sangat jarang. Waktu yang

diperlukan *EXT3 file* sistem setelah terjadi *unclean shutdown* tidak tergantung dari ukuran *file* sistem atau banyaknya *file*, tetapi tergantung dari besarnya jurnal yang digunakan untuk menjaga konsistensi. Besar jurnal *default* memerlukan waktu kira-kira sedetik untuk pulih, tergantung kecepatan *hardware.*

3 Integritas data dan kecepatan akses yang fleksibel.

4 Ext3FS menjamin adanya integritas data setelah terjadi kerusakan atau *unclean shutdown*. Ext3FS memungkinkan kita memilih jenis dan tipe proteksi dari data.

5 Mudah melakukan migrasi dari Ex2FS.

6 Kita dapat berpindah dari *EXT2* ke sistem *EXT3* tanpa melakukan format ulang.

7 Cepat

8 Daripada menulis data lebih dari sekali, *EXT3* mempunyai *throughput* yang lebih besar daripada *EXT2* karena *EXT3* memaksimalkan pergerakan *head hard disk*. Kita bisa memilih tiga jurnal mode untuk memaksimalkan kecepatan, tetapi integritas data tidak terjamin.

2.6 Protokol

Protokol adalah suatu tata cara atau aturan berkomunikasi antar dua pihak. Dalam konteks teknologi informasi, protokol berarti suatu standar komunikasi antar sistem atau komputer dalam suatu jaringan. Dengan adanya protokol maka suatu data dapat ditransmisikan ke tujuan dan setelah sampai di tujuan maka data tersebut bisa diterima dengan benar. Tata cara komunikasi ini sangatlah kompleks,

meliputi cara meng-*handle error*, cara mengirimkan data, cara si pengirim memberi informasi bahwa data sudah selesai ditransmisikan, cara si penerima mengetahui bahwa data sudah diterima dan lain sebagainya.

Protokol bisa dibagi dua macam, yaitu:

Low-level protocol, protocol ini menentukan aturan transmisi data dari sisi elektronik dan *physical*, misalnya aturan mengenai *bit* dan *byte ordering*, mekanisme deteksi *error* dan koreksi terhadap error tersebut. Contoh dari protokol ini misalnya adalah TCP/IP.

High-level protocol, protocol ini berurusan dengan format data, misalnya seperti sintaks dari pesan yang dikirim, urutan pesan yang dikirim (*handshaking*) dan karakter set yang digunakan. Contoh dari protokol ini sudah kita rasakan sehari-hari, misalnya SMTP dan POP3 untuk e-mail. HTTP untuk *web browsing*. FTP untuk transfer *file* dan lain sebagainya.

Sekarang kita ambil contoh misalnya SMTP. SMTP adalah singkatan dari *Simple Mail Transfer Protocol*, adalah suatu protokol standar yang digunakan untuk mengirimkan e-mail dari pengirim ke sebuah *mail server* dan dari *mail server* akhirnya pesan tersebut dikirimkan ke alamat e-mail tujuan. Jadi secara teknis maka akan terasa bahwa semua digerakkan dengan suatu protokol yang rapi dan *solid*, yaitu SMTP.

Protokol haruslah bersifat standar agar bisa digunakan oleh semua pihak. Andapun bisa membuat suatu protokol jika mau. Pada komunitas internet dikenal yang namanya RFC. Singkatan dari *Request For Comment*. RFC adalah semacam dokumen yang berisi ketentuan protokol, baik yang sudah menjadi standar

ataupun yang masih dalam tahap *experiment* atau *draft*. Semua protokol standar di internet selalu ditulis dalam bentuk dokumen RFC dan diberi nomor tertentu.

2.7 Protokol SSH

SSH memungkinkan kita untuk *login* ke *sistem host* secara *remote*, tidak seperti *rlogin* atau *telnet*, *SSH* mengenkripsi *session login*, yang membuat *password* tidak dapat dibaca oleh orang yang tidak berwenang.

Dengan adanya SSH ini kita dapat *login* ke sistem yang lain melalui *command shell*. Menggunakan *secure method* untuk login secara *remote* ke dalam sistem lain akan mengurangi bahaya keamanan baik itu di sistem anda maupun *remote* sistem. SSH (*Secure Shell*) adalah sebuah protokol untuk menghubungkan dua sistem. Didalam protokol SSH, komputer klien memulai koneksi dengan komputer server. Berikut ini adalah beberapa keuntungan yang diberikan oleh protokol SSH :

a) Setelah terkoneksi, komputer klien memeriksa koneksi ke server yang sama selama session berikutnya
b) Komputer klien mengirimkan autentifikasi ke server, seperti *username*, *password* kedalam bentuk format yang sudah dienkripsi
c) Semua informasi baik itu yang dikirm maupun yang diterima selama koneksi dikirim menggunakan enkripsi 128bit, yang membuat informasi itu sulit untuk didekripsi kembali
d) Komputer klien mempunyai kemampuan untuk menggunakan program X11 yang dapat dipanggil dari *command shell*

2.8 Concurrent Version System (CVS)

CVS adalah singkatan dari *Concurrent Version System*, sebuah sistem yang dapat membantu untuk mengontrol perubahan – perubahan file. CVS bermula sebagai serangkaian *script shell* yang ditulis oleh Dick Grune, yang dikirimkan ke *newsgroup comp.source.unix* dalam rilis 6 volume pada Desember 1986. Meski tidak ada kode yang berasal dari *script* ini dalam versi baru CVS, banyak algoritma resolusi konflik berasal dari script tersebut. Bulan April 1989, Brian Berliner merancang dan mengkodekan CVS. Jeff Polk kemudian membantu Brian dengan desain modul CVS dan dukungan cabang vendor (*vendor branch support*). CVS ini biasanya dipakai oleh programmer untuk membantunya bekerja, walaupun sebenarnya CVS ini bisa dimanfaatkan tidak hanya oleh programmer. Contohnya CVS ini bisa juga dimanfaatkan oleh webmaster atau bisa juga dimanfaatkan oleh orang – orang yang ingin membuat dokumen, dan banyak lagi lainnya.

Sebenarnya ada banyak sistem yang menawarkan pengontrolan versi, tapi biasanya sistem itu hanya berlaku untuk satu lingkungan tertentu. Contohnya saja seperti *Microsoft Word*, aplikasi keluaran *Microsoft* menyediakan fasilitas pengontrolan versi untuk dokumen yang diolahnya, kemudian ada contoh lain misalnya *Delphi*, produk keluaran *Borland* ini dulu juga pernah menyertakan sistem pengontrolan versi yang dikenal dengan nama *Version Control System* (VCS), tetapi pada akhirnya fasilitas ini malah dihilangkan. Masih banyak lagi contoh sistem pengontrolan versi yang lain, tetapi kebanyakan hanya bisa berjalan untuk aplikasi tertentu.

Pada prinsipnya, CVS sangat membantu bagi para pengembang aplikasi dalam pengolahan file, terutama bagi *development team* yang memiliki banyak programmer yang bekerja dalam satu *group*.

Salah satu manfaat CVS dapat dilihat dari contoh kasus dibawah ini :

Seseorang pelanggan memesan aplikasi kepada programmer. Pelanggan tersebut ingin aplikasi yang dia pesan selesai dalam 2 minggu. Setelah melakukan persetujuan maka programmer mulai melakukan pengembangan aplikasi yang telah dipesan. Ditengah proses pengembangan ketika kita menyerahkan *prototype* aplikasi ke pelanggan untuk diuji coba, ternyata ada beberapa bagian yang tidak cocok, akhirnya programmer harus menyesuaikan aplikasi lagi. Setelah programmer telah melakukan penyesuaian ulang ternyata pelanggan mengiginkan *prototype* aplikasi yang sebelumnya. Seandainya programmer masih menyimpan versi yang sebelumnya tentu saja persoalan semacam ini bukan menjadi suatu masalah dan sebagai programmer yang baik seharusnya memang memiliki *source code* program yang dibuatnya dari awal hingga akhir beserta dengan dokumentasinya.

Dengan CVS, kita dapat mengambil kembali versi sebelumnya dari aplikasi yang sudah kita buat. Dengan CVS kita dapat pula menyimpan semua versi dari tiap file yang kita buat. CVS juga sangat bermanfaat jika adalah anggota sekelompok orang yang bekerja pada proyek yang sama. Dengan proyek yang dikerjakan oleh banyak orang, maka kemungkinan terjadi konflik pada file – file *source code* sangat besar. CVS mengatasi masalah ini dengan cara membatasi para pengembang satu sama lain. Setiap pengembang bekerja di direktorinya masing – masing.

2.8.1 Fitur CVS (*Concurrent Version System*)

a) Dengan CVS kita dapat kembali ke versi sebelumnya dari software anda

b) Dengan CVS kita dapat pula menyimpan semua versi dari tiap file yang anda buat

c) CVS juga bermanfaat jika anda adalah anggota sekelompok orang yang bekerja pada proyek yang sama. Dengan proyek yang dikerjakan oleh banyak orang, maka kemungkinan terjadinya konflik pada file-file *source code* sangat besar, misalnya satu orang merubah file yang telah dirubah oleh orang lainnya. CVS mengatasi masalah ini dengan cara meng-*insulate* para pengembang satu sama lain. Setiap pengembang bekerja di direktorinya, dan CVS menggabungkan pekerjaannya ketika mereka telah selesai.

2.9 Repository

Repository merupakan tempat penyimpanan salinan semua direktori dan file – file yang berada didalam sistem kendali versi. Ketika seorang programmer mengakses sebuah file yang berada didalam CVS, sebenarnya dia tidak langsung mengakses ke file tersebut. Sebagai gantinya programmer akan menggunakan CVS *command* untuk melakukan penyalinan file dari repository dan bekerja pada file salinan tersebut. Setelah selesai melakukan perubahan file tersebut maka programmer melakukan penyimpanan kembali ke *repository* deangan menggunakan CVS *Command*. Dan *repository* kini dapat akan menyimpan perubahan file yang telah dilakukan.

Repository dapat diakses oleh komputer lokal maupun komputer yang terhubung dalam jaringan. *Repository* dinagi menjadi beberapa komponen yaitu : CVSROOT yang berisikan file – file *administrative* untuk CVS itu sendiri, dan komponen lainnya berisi modul – modul yang digunakan oleh pengguna.

2.10 Putty

Putty adalah sebuah program *client* untuk *32-bit Windows systems*. Putty mendukung 3 metode login, yaitu *Telnet, R-Login* dan *SSH*. Dari ketiga metode tersebut yang dipakai penulis adalah *SSH*, karena SSH mempunyai tingkat keamanan yang lebih tinggi dari kedua metode tersebut. Dengan menggunakan putty kita dapat melakukan login / autentifikasi ke sistem lain (misalnya dari windows ke linux) secara *remote*