KONSEP MOUNTING, SHARING DAN PROTEKSI

Mounting

Mounting adalah proses mengkaitkan sebuah sistem berkas yang baru ditemukan pada sebuah piranti ke struktur direktori utama yang sedang dipakai. Piranti-piranti yang akan di-*mount* dapat berupa *cd-rom*, disket atau sebuah *zip-drive*. Tiap-tiap sistem berkas yang akan di-*mount* akan diberikan sebuah *mount point*, atau sebuah direktori dalam pohon direktori sistem Anda, yang sedang diakses.

Sistem berkas yang dideskripsikan di /etc/fstab (fstab adalah singkatan dari *filesystem tables*) biasanya akan di-*mount* saat komputer baru mulai dinyalakan, tapi dapat juga me-*mount* sistem berkas tambahan dengan menggunakan perintah:

mount [nama piranti]

atau dapat juga dengan menambahkan secara manual *mount point* ke berkas /etc/fstab. Daftar sistem berkas yang di-*mount* dapat dilihat kapan saja dengan menggunakan perintah *mount*. Karena izinnya hanya diatur *read-only* di berkas fstab, maka tidak perlu khawatir pengguna lain akan mencoba mengubah dan menulis *mount point* yang baru.

Seperti biasa saat ingin mengutak-atik berkas konfigurasi seperti mengubah isi berkas fstab, pastikan untuk membuat berkas cadangan untuk mencegah terjadinya kesalahan teknis yang dapat menyebabkan suatu kekacauan. Kita dapat melakukannya dengan cara menyediakan sebuah disket atau *recovery-disk* dan mem-*back-up* berkas fstab tersebut sebelum membukanya di editor teks untuk diutak-atik.

Red Hat Linux dan sistem operasi lainnya yang mirip dengan UNIX mengakses berkas dengan cara yang berbeda dari MS-DOS, Windows dan Macintosh. Di linux, segalanya disimpan di dalam sebuah lokasi yang dapat ditentukan dalam sebuah struktur data. Linux bahkan menyimpan perintah-perintah sebagai berkas. Seperti sistem operasi modern lainnya, Linux memiliki struktur *tree*, hirarki, dan organisasi direktori yang disebut sistem berkas.

Semua ruang kosong yang tersedia di *disk* diatur dalam sebuah pohon direktori tunggal. Dasar sistem ini adalah direktori *root* yang dinyatakan dengan sebuah garis miring ("/"). Pada linux, isi

sebuah sistem berkas dibuat nyata tersedia dengan menggabungkan sistem berkas ke dalam sebuah sistem direktori melalui sebuah proses yang disebut *mounting*.

Sistem berkas dapat di-*mount* mau pun di-*umount* yang berarti sistem berkas tersebut dapat tersambung atau tidak dengan struktur pohon direktori. Perbedaannya adalah sistem berkas tersebut akan selalu di-*mount* ke direktori *root* ketika sistem sedang berjalan dan tidak dapat di-*mount*. Sistem berkas yang lain di-*mount* seperlunya, contohnya yang berisi *hard drive* berbeda dengan *floppy disk* atau CD-ROM.

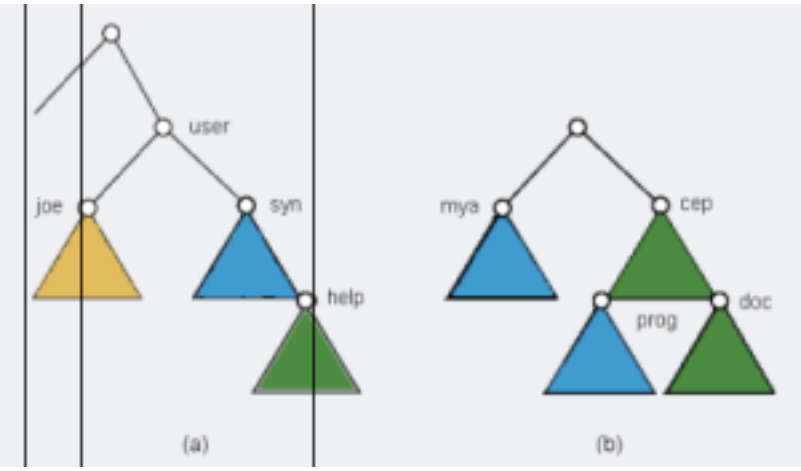
Mounting Overview

Mounting membuat sistem berkas, direktori, piranti dan berkas lainnya menjadi dapat digunakan di lokasi-lokasi tertentu, sehingga memungkinkan direktori itu menjadi dapat diakses. Perintah mount menginstruksikan sistem operasi untuk mengkaitkan sebuah sistem berkas ke sebuah direktori khusus.

Memahami Mount Point

Mount point adalah sebuah direktori dimana berkas baru menjadi dapat diakses. Untuk me-mount suatu sistem berkas atau direktori, titik mount-nya harus berupa direktori, dan untuk me-mount sebuah berkas, mount point-nya juga harus berupa sebuah berkas.

Biasanya, sebuah sistem berkas, direktori, atau sebuah berkas di-mount ke sebuah mount point yang kosong, tapi biasanya hal tersebut tidak diperlukan. Jika sebuah berkas atau direktori yang akan menjadi mount point berisi data, data tersebut tidak akan dapat diakses selama direktori/berkas tersebut sedang dijadikan mount point oleh berkas atau direktori lain. Sebagai akibatnya, berkas yang di-mount akan menimpa apa yang sebelumnya ada di direktori/berkas tersebut. Data asli dari direktori itu dapat diakses kembali bila proses mounting sudah selesai.



Gambar 9-1. Mount Point

Saat sebuah sistem berkas di-mount ke sebuah direktori, izin direktori root dari berkas yang di-mount akan mengambil alih izin dari mount point. Pengecualiannya adalah pada direktori induk akan memiliki atribut .. (double dot). Agar sistem operasi dapat mengakses sistem berkas yang baru, direktori induk dari mount point harus tersedia.

Untuk segala perintah yang membutuhkan informasi direktori induk, pengguna harus mengubah izin dari direktori mounted-over. Kegagalan direktori mounted-over untuk mengabulkan izin dapat menyebabkan hasil yang tidak terduga, terutama karena izin dari direktori mounted-over tidak dapat terlihat. Kegagalan umum terjadi pada perintah pwd. Tanpa mengubah izin direktori mounted-over, akan timbul pesan error seperti ini:

```
pwd: permission denied
```

Masalah ini dapat diatasi dengan mengatur agar izin setidaknya di-set dengan 111.

Mounting Sistem Berkas, Direktori, dan Berkas

Ada dua jenis mounting: remote mounting dan mounting lokal. Remote mounting dilakukan dengan sistem remote dimana data dikirimkan melalui jalur telekomunikasi. Remote sistem berkas seperti Network File Systems (NFS), mengharuskan agar file diekspor dulu sebelum di-mount. mounting lokal dilakukan di sistem lokal.

Tiap-tiap sistem berkas berhubungan dengan piranti yang berbeda. Sebelum kita menggunakan sebuah sistem berkas, sistem berkas tersebut harus dihubungkan dengan struktur direktori yang ada (dapat root atau berkas yang lain yang sudah tersambung).
Sebagai contoh, kita dapat me-mount dari /home/server/database ke mount point yang dispesifikasikan sebagai /home/user1, /home/user2, and /home/user3:

- /home/server/database /home/user1
- /home/server/database /home/user2
- /home/server/database /home/user3

Sharing

Kita dapat berbagi berkas dengan pengguna lainnya yang teregistrasi. Hal pertama yang harus kita lakukan adalah menentukan dengan siapa berkas tersebut akan dibagi dan akses seperti apa yang akan diberikan kepada mereka. Berbagi bekas berguna bagi pengguna yang ingin bergabung dengan pengguna lain dan mengurangi usaha untuk mencapai sebuah hasil akhir.

Saat sebuah sistem operasi dibuat untuk *multiple user*, masalah berbagi berkas, penamaan berkas dan proteksi berkas menjadi sangat penting. Oleh karena itu, sistem operasi harus dapat mengakomodasikan/mengatur pembagian berkas dengan memberikan suatu struktur direktori yang membiarkan pengguna untuk saling berbagi.

Berkaitan dengan permasalahan akses berkas, kita dapat mengijinkan pengguna lain untuk melihat, mengedit atau menghapus suatu berkas. Proses mengedit berkas yang menggunakan *web-file system* berbeda dengan menggunakan aplikasi seperti Windows Explorer. Untuk mengedit sebuah file dengan *web-file system*, kita harus menduplikasi berkas tersebut dahulu dari *web-file system* ke komputer lokal, mengeditnya di komputer lokal, dan mengirim file tersebut kembali ke sistem dengan menggunakan nama berkas yang sama.

Sebagai contoh, kita dapat mengizinkan semua pengguna yang terdaftar untuk melihat berkas-berkas yang ada di direktori (tetapi mereka tidak dapat mengedit atau menghapus berkas tersebut). Contoh lainnya, kita dapat mengijinkan satu pengguna saja untuk melakukan apa pun terhadap sebuah direktori dan segala isinya (ijin untuk melihat semua berkas, mengeditnya,

menambah berkas bahkan menghapus isi berkas). Kita juga dapat memberikan kesempatan bagi pengguna untuk mengubah izin dan kontrol akses dari sebuah isi direktori, namun hal tersebut biasanya di luar kebiasaan, sebab seharusnya satu-satunya pengguna yang berhak mengubah izin adalah kita sendiri.

Sistem berkas web memungkinkan kita untuk menspesifikasikan suatu akses dalam tingkatan berkas. Jadi, kita dapat mengijinkan seluruh orang untuk melihat isi dari sebuah direktori atau mengijinkan sebagian kecil pengguna saja untuk mengakses suatu direktori. Bahkan, dalam kenyataannya, kita dapat menspesifikasikan jenis akses yang berbeda dengan jumlah pengguna yang berbeda pula.

Kebanyakan pada sistem banyak pengguna menerapkan konsep direktor berkas *owner/user* dan group.

- *Owner*: pengguna yang dapat mengubah atribut, memberikan akses, dan memiliki sebagian besar kontrol di dalam sebuah berkas atau direktori.
- *Group*: sebagian pengguna yang sedang berbagi berkas.

Proteksi

Ketika kita menyimpan informasi dalam sebuah sistem komputer, ada dua hal yang harus menjadi perhatian utama kita. Hal tersebut adalah :

1. Reabilitas dari sebuah sistem

   Maksud dari reabilitas sistem adalah kemampuan sebuah sistem untuk melindungi informasi yang telah tersimpan agar terhindar dari kerusakan, dalam hal ini adalah perlindungan secara fisik pada sebuah berkas. Reabilitas sistem dapat dijaga dengan membuat cadangan dari setiap berkas secara manual atau un otomatis, sesuai dengan layanan yang dari sebuah sistem operasi.

2. Proteksi (Perlindungan) terhadap sebuah berkas

   Perlindungan terhadap berkas dapat dilakukan dengan berbagai cara. Pada bagian ini, kita akan membahas secara detail mekanisme yang diterapkan dalam melindungi sebuah berkas.

Dalam pembahasan mengenai proteksi berkas, kita akan berbicara lebih mengenai sisi keamanan dan mekanisme bagaimana menjaga keutuhan suatu berkas dari gangguan akses luar yang tidak dikehendaki. Sebagai contoh bayangkan saja Anda berada di suatu kelompok kerja dimana masing-masing staf kerja disediakan komputer dan mereka saling terhubung membentuk suatu jaringan; sehingga setiap pekerjaan/dokumen/berkas dapat dibagi-bagikan ke semua pengguna dalam jaringan tersebut. Misalkan lagi Anda harus menyerahkan berkas RAHASIA.txt ke atasan Anda, dalam hal ini Anda harus menjamin bahwa isi berkas tersebut tidak boleh diketahui oleh staf kerja lain apalagi sampai dimodifikasi oleh orang yang tidak berwenang. Suatu mekanisme pengamanan berkas mutlak diperlukan dengan memberikan batasan akses ke setiap pengguna terhadap berkas tertentu.

Tipe Akses

Proteksi berkaitan dengan kemampuan akses langsung ke berkas tertentu. Panjangnya, apabila suatu sistem telah menentukan secara pasti akses berkas tersebut selalu ditutup atau selalu dibebaskan ke setiap pengguna lain maka sistem tersebut tidak memerlukan suatu mekanisme proteksi. Tetapi tampaknya pengimplementasian seperti ini terlalu ekstrim dan bukan pendekatan yang baik. Kita perlu membagi akses langsung ini menjadi beberapa jenis-jenis tertentu yang dapat kita atur dan ditentukan (akses yang terkontrol).

Dalam pendekatan ini, kita mendapatkan suatu mekanisme proteksi yang dilakukan dengan cara membatasi jenis akses ke suatu berkas. Beberapa jenis akses tersebut antara lain:

- Read/Baca: membaca berkas
- Write/Tulis: menulis berkas
- Execute/Eksekusi: memasukkan berkas ke memori dan dieksekusi
- Append/Sisip: menulis informasi baru pada baris akhir berkas
- Delete/Hapus: menghapus berkas
- List/Daftar: mendaftar nama dan atribut berkas

Operasi lain seperti *rename*, *copying*, atau *editing* yang mungkin terdapat di beberapa sistem merupakan gabungan dari beberapa jenis kontrol akses diatas. Sebagai contoh, menyalin sebuah berkas dikerjakan sebagai runtutan permintaan baca dari pengguna. Sehingga dalam hal ini,

seorang pengguna yang memiliki kontrol akses *read* dapat pula meng-*copy*, mencetak dan sebagainya.

Kontrol Akses

Pendekatan yang paling umum dipakai dalam mengatasi masalah proteksi berkas adalah dengan membiarkan akses ke berkas ditentukan langsung oleh pengguna (dalam hal ini pemilik/pembuat berkas itu). Sang pemilik bebas menentukan jenis akses apa yang diperbolehkan untuk pengguna lain. Hal ini dapat dilakukan dengan menghubungkan setiap berkas atau direktori dengan suatu daftar kontrol-akses (*Access-Control Lists/ACL*) yang berisi nama pengguna dan jenis akses apa yang diberikan kepada pengguna tersebut.

Sebagai contoh dalam suatu sistem VMS, untuk melihat daftar direktori berikut daftar kontrol-akses, ketik perintah "DIR/SECURITY", atau "DIR/SEC". Salah satu keluaran perintah itu adalah daftar seperti berikut ini:

```
WWW-HOME.DIR;1      [HMC2000,WWART]          (RW,RWED,,E)
    (IDENTIFIER=WWW_SERVER_ACCESS,OPTIONS=DEFAULT,ACCESS=READ)
(IDENTIFIER=WWW_SERVER_ACCESS,ACCESS=READ)
```

Baris pertama menunjukkan nama berkas tersebut WWW-HOME.DIR kemudian disebelahnya nama grup pemilik HMC2000 dan nama pengguna WWART diikuti dengan sekelompok jenis akses RW, RWED,,E (R=Baca, W=Tulis, E=Eksekusi, D=Hapus). Dua baris dibawahnya itulah yang disebut daftar konrol-akses. Satu-satu baris disebut sebagai masukan kontrol-akses (*Access Control Entry/ACE*) dan terdiri dari 3 bagian. Bagian pertama disebut sebagai IDENTIFIER/Identifikasi, menyatakan nama grup atau nama pengguna (seperti [HMC2000, WWART]) atau akses khusus (seperti WWW_SERVER_ACCESS). Bagian kedua merupakan daftar OPTIONS/Plihan-pilihan. Dan terakhir adalah daftar ijin ACCESS/akses, seperti *read* atau *execute*, yang diberikan kepada siapa saja yang mengacu pada bagian Identifikasi.

Cara kerjanya: apabila seorang pengguna meminta akses ke suatu berkas/direktori, sistem operasi akan memeriksa ke daftar kontrol-akses apakah nama pengguna itu tercantum dalam daftar tersebut. Apabila benar terdaftar, permintaan akses akan diberikan dan sebaliknya bila tidak, permintaan akses akan ditolak.

Pendekatan ini memiliki keuntungan karena penggunaan metodologi akses yang kompleks sehingga sulit ditembus sembarangan. Masalah utamanya adalah ukuran dari daftar akses tersebut. Bayangkan apabila kita mengijinkan semua orang boleh membaca berkas tersebut, kita harus mendaftar semua nama pengguna disertai ijin akses baca mereka. Lebih jauh lagi, tehnik ini memiliki dua konsekuensi yang tidak diinginkan:

- Pembuatan daftar semacam itu merupakan pekerjaan yang melelahkan dan tidak efektif.
- Entri direktori yang sebelumnya memiliki ukuran tetap, menjadi ukuran yang dapat berubah-ubah, mengakibatkan lebih rumitnya managemen ruang kosong.

Masalah ini dapat diselesaikan dengan penggunaan daftar akses yang telah disederhanakan.

Untuk menyederhanakan ukuran daftar kontrol akses, banyak sistem menggunakan tiga klasifikasi pengguna sebagai berikut:

- *Owner*: pengguna yang telah membuat berkas tersebut.
- *Group*: sekelompok pengguna yang saling berbagi berkas dan membutuhkan akses yang sama.
- *Universe*: keseluruhan pengguna.

Pendekatan yang dipakai belum lama ini adalah dengan mengkombinasikan daftar kontrol-akses dengan konsep kontrol- akses pemilik, grup dan semesta yang telah dijabarkan diatas. Sebagai contoh, Solaris 2.6 dan versi berikutnya menggunakan tiga klasifikasi kontrol-akses sebagai pilihan umum, tetapi juga menambahkan secara khusus daftar kontrol-akses terhadap berkas/direktori tertentu sehingga semakin baik sistem proteksi berkasnya.

Contoh lain yaitu sistem UNIX dimana konrol-aksesnya dinyatakan dalam 3 bagian. Masing-masing bagian merupakan klasifikasi pengguna (yi.pemilik, grup dan semesta). Setiap bagian kemudian dibagi lagi menjadi 3 bit jenis akses -*rwx*, dimana *r* mengontrol akses baca, *w* mengontrol akses tulis dan *x* mengontrol eksekusi. Dalam pendekatan ini, 9 bit diperlukan untuk merekam seluruh informasi proteksi berkas.

Berikut adalah keluaran dari perintah "ls -al" di sistem UNIX:

```
-rwxr-x---  1  david karyawan 12210 Nov 14 20:12 laporan.txt
```

Baris di atas menyatakan bahwa berkas laporan.txt memiliki akses penuh terhadap pemilik berkas (yi.david), grupnya hanya dapat membaca dan mengeksekusi, sedang lainnya tidak memiliki akses sama sekali.

Pendekatan Pengamanan Lainnya

Salah satu pendekatan lain terhadap masalah proteksi adalah dengan memberikan sebuah kata kunci (*password*) ke setiap berkas. Jika kata-kata kunci tersebut dipilih secara acak dan sering diganti, pendekatan ini sangatlah efektif sebab membatasi akses ke suatu berkas hanya diperuntukkan bagi pengguina yang mengetahui kata kunci tersebut.

Meski pun demikian, pendekatan ini memiliki beberapa kekurangan, diantaranya:

- Kata kunci yang perlu diingat oleh pengguna akan semakin banyak, sehingga membuatnya menjadi tidak praktis.
- Jika hanya satu kata kunci yang digunakan di semua berkas, maka jika sekali kata kunci itu diketahui oleh orang lain, orang tersebut dapat dengan mudah mengakses semua berkas lainnya. Beberapa sistem (contoh: TOPS-20) memungkinkan seorang pengguna untuk memasukkaan sebuah kata kunci dengan suatu subdirektori untuk menghadapi masalah ini, bukan dengan satu berkas tertentu.
- Umumnya, hanya satu kata kunci yang diasosiasikan dengan semua berkas lain. Sehingga, pengamanan hanya menjadi semua-atau-tidak sama sekali. Untuk mendukung pengamanan pada tingkat yang lebih mendetail, kita harus menggunakan banyak kata kunci.

Recovery

Karena semua direktori dan berkas disimpan di dalam memori utama dan disk, maka kita perlu memastikan bahwa kegagalan pada sistem tidak menyebabkan hilangnya data atau data menjadi tidak konsisten.

Pemeriksaan Rutin

Informasi direktori pada memori utama pada umumnya lebih up to date daripada informasi yang terdapat di disk dikarenakan penulisan dari informasi direktori cached ke disk tidak langsung terjadi pada saat setelah peng-update-an terjadi. Consistency checker membandingkan data yang

terdapat di struktur direktori dengan blok-blok data pada disk, dan mencoba memperbaiki semua ketidak konsistensian yang terjadi akibat crash-nya komputer. Algoritma pengalokasian dan management ruang kosong menentukan tipe dari masalah yang ditemukan oleh checker dan seberap sukses dalam memperbaiki masalah-masalah tersebut.

Back Up and Restore

Karena kadang-kadang magnetik disk gagal, kita harus memastikan bahwa datanya tidak hilang selamanya. Karena itu, kita menggunakan program sistem untuk mem-back up data dari disk ke alat penyimpanan yang lain seperti floppy disk, magnetic tape atau optical disk. Pengembalian berkas-berkas yang hilang hanya masalah menempatkan lagi data dari backup data yang telah dilakukan.

Untuk meminimalisir penyalinan, kita dapat menggunakan informasi dari setiap masukan direktori berkas. Umpamanya, jika progam back up mengetahui bahwa back up terakhir dari berkas sudah selesai dan penulisan terakhir pada berkas dalam direktori menandakan berkas tidak terjadi perubahan maka berkas tidak harus disalin lagi. Penjadualan back up yang umum sebagai berikut :

Hari 1 : Salin ke tempat penyimpanan back up semua berkas dari disk, disebut sebuah full backup.
Hari 2 : Salin ke tempat penyimpanan lain semua berkas yang berubah sejak hari 1, disebut incremental backup.
Hari 3 :Salin ke tempat penyimpanan lain semua berkas yang berubah sejak hari 2.
Hari N : Salin ke tempat penyimpanan lain semua berkas yang berubah sejak hari N-1, lalu kembali ke hari 1.

Keuntungan dari siklus backup ini adalah kita dapat menempatkan kembali berkas mana pun yang tidak sengaja terhapus pada waktu siklus dengan mendapatkannya dari back up hari sebelumnya. Panjang dari siklus disetujui antara banyaknya tempat penyimpanan backup yang diperlukan dan jumlah hari ke belakang dari penempatan kembali dapat dilakukan.

Ada juga kebiasaan untuk mem-backup keseluruhan dari waktu ke waktu untuk disimpan selamanya daripada media backupnya digunakan kembali. Ada bagusnya menyimpan backup-backup permanent ini di lokasi yang jauh dari backup yang biasa, untuk menghindari kecelakaan seperti kebakaran dan lain-lain. Dan jangan menggunakan kembali media backup terlalu lama karena media tersebut akan rusak jika terlalu sering digunakan kembali.