CHAPTER 13

Mengajukan - SistemAntarmuka

Bagi sebagian besar pengguna, sistem file adalah aspek yang paling terlihat dari tujuan umum Pengoperasian sistem. Dia menyediakan itu mekanisme untuk on line penyimpanan dari Dan mengakses untuk data dan program sistem operasi dan semua pengguna komputer sistem. Itu mengajukan sistem terdiri dari dua berbeda bagian: A koleksi dari file, setiap menyimpan terkait data, Dan A direktori struktur, yang mengatur Dan memberikan informasi tentang semua file dalam sistem. Sebagian besar sistem file tetap hidup penyimpanan perangkat, yang Kami dijelaskan di dalam Bab 11 Dan akan melanjutkan ke membahas di dalam itu Berikutnya bab. Di dalam ini bab, Kami mempertimbangkan itu bermacam-macam aspek dari file Dan struktur direktori utama. Kami juga membahas semantik berbagi file antara beberapa proses, pengguna, dan komputer. Terakhir, kami mendiskusikan cara untuk melakukannya menangani perlindungan file, diperlukan ketika kita memiliki banyak pengguna dan menginginkannya kontrol WHO mungkin mengakses file Dan Bagaimana file mungkin menjadi diakses.

CHAPTER OBJECTIVES

- Explain the function of file systems.
- Describe the interfaces to file systems.
- Discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures.
- Explore file-system protection.

13.1 Mengajukan Konsep

Komputer dapat menyimpan informasi di berbagai media penyimpanan, seperti NVM perangkat, HDD , bersifat magnetis kaset, Dan optik disk. Jadi itu itu komputer sistem akan nyaman digunakan, sistem operasi menyediakan logika yang seragam tampilan informasi yang disimpan. Sistem operasi mengabstraksi dari fisik properti dari -nya penyimpanan perangkat ke mendefinisikan A logis penyimpanan satuan, itu **file** . File adalah dipetakan oleh itu Pengoperasian sistem ke fisik perangkat. Ini penyimpanan perangkat adalah biasanya tidak mudah menguap, Jadi itu isi adalah gigih di antara sistem reboot.

File adalah kumpulan informasi terkait bernama yang direkam pada detik-detik. penyimpanan ondary. Dari sudut pandang pengguna, file adalah bagian terkecil dari logis sekunder penyimpanan; itu adalah, data tidak bisa ditulis ke sekunder penyimpanan kecuali mereka adalah di dalam A mengajukan. Biasanya, file mewakili program (keduanya sumber dan bentuk objek) dan data. File data mungkin numerik, abjad, alfanumerik, merik, atau biner. File mungkin menjadi bebas membentuk, seperti sebagai teks file, atau mungkin menjadi diformat dengan kaku. Secara umum, file adalah urutan bit, byte, baris, atau catatan, yang berarti yang ditentukan oleh pembuat dan pengguna file. Konsep file adalah dengan demikian sangat umum.

Karena file adalah **metode** yang digunakan pengguna dan aplikasi untuk menyimpan dan mengambil data, Dan Karena mereka adalah Jadi umum tujuan, milik mereka menggunakan memiliki membentang melampaui batas aslinya. Misalnya UNIX , Linux, dan beberapa operasi lainnya. sistem makan menyediakan sistem file proc yang menggunakan antarmuka sistem file untuk menyediakan mengakses ke sistem informasi (seperti proses detailnya).

Informasi dalam file ditentukan oleh pembuatnya. Banyak jenis yang berbeda informasi mungkin menjadi disimpan di dalam A mengajukan-sumber atau dapat dieksekusi program, numerik atau data teks, foto, musik, video, dan sebagainya. Sebuah file memiliki struktur tertentu yang ditentukan masa depan, yang tergantung pada jenisnya. **File** teks adalah urutan karakter yang terorganisir menjadi beberapa baris (dan mungkin halaman). File **sumber** adalah urutan fungsi, masing-masing yang adalah lebih jauh terorganisir sebagai deklarasi diikuti oleh dapat dieksekusi pernyataan. File **yang dapat dieksekusi** adalah serangkaian bagian kode yang dapat dibawa oleh pemuat Penyimpanan dan mengeksekusi.

13.1.1 Mengajukan Atribut

Sebuah file diberi nama, untuk kenyamanan pengguna manusia, dan dirujuk oleh namanya. Sebuah nama biasanya berupa serangkaian karakter, seperti example.c . Beberapa sistem membedakan antara karakter huruf besar dan kecil dalam nama, sedangkan sistem lain tidak. Ketika sebuah file diberi nama, file tersebut menjadi independen proses, pengguna, dan bahkan sistem yang menciptakannya. Misalnya, satu mungkin pengguna membuat mengajukan contoh.c , dan lain mungkin pengguna edit itu mengajukan dengan menyebutkan namanya. Pemilik file mungkin menulis file tersebut ke drive USB , kirimkan sebagai lampiran email, atau salin melalui jaringan, dan masih bisa dipanggil example.c pada sistem tujuan. Kecuali ada sharing dan metode sinkronisasi, salinan kedua itu sekarang tidak bergantung pada salinan pertama dandapat berubah terpisah.

Atribut file bervariasi dari satu sistem operasi ke sistem operasi lainnya, tetapi biasanyaterdiri atas ini:

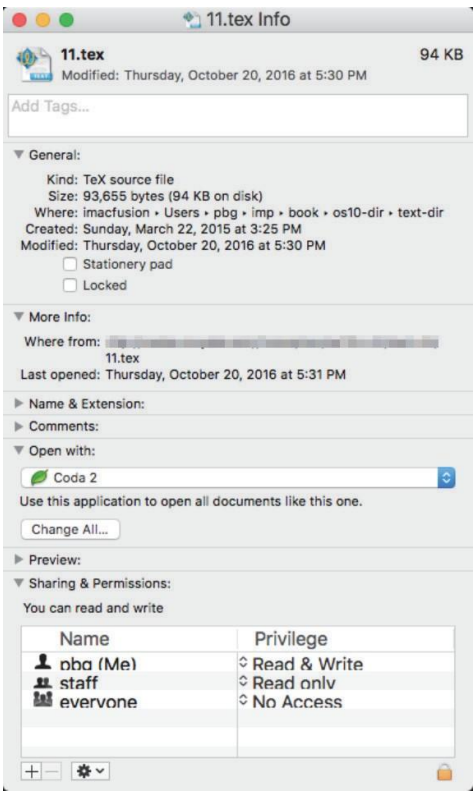
- **Nama** . Itu simbolis mengajukan nama adalah itu hanya informasi disimpan di dalam manusia-dapat dibaca membentuk.
- **Identifikasi** . Ini unik menandai, biasanya A nomor, mengidentifikasi itu mengajukan di dalam itumengajukan sistem; dia adalah itu nama yang tidak dapat dibaca manusia untuk itu mengajukan.
- **Jenis** . Ini informasi adalah diperlukan untuk sistem itu mendukung berbeda jenisdari file.

- **Lokasi** . Ini informasi adalah A penunjuk ke A perangkat Dan ke itu lokasi dariitu mengajukan perangkat itu.

- **Ukuran** . Ukuran file saat ini (dalam byte, kata, atau blok) dan mungkin itu maksimum diizinkan ukuran adalah termasuk di dalam atribut ini.
- **Perlindungan** . Informasi kontrol akses menentukan siapa yang dapat melakukan membaca, menulis, mengeksekusi, Dan Jadi pada.
- **Stempel waktu dan identifikasi pengguna** . Informasi ini mungkin disimpan untuk pembuatan, modifikasi terakhir, dan penggunaan terakhir. Data ini dapat berguna untuk pro- teks, keamanan, Dan penggunaan pemantauan.

Beberapa lebih baru mengajukan sistem Juga mendukung **diperpanjang mengajukan atribut** , termasuk arang-aktor pengkodean dari itu mengajukan Dan keamanan fitur seperti sebagai A mengajukan checksum. Angka 13.1 mengilustrasikan A **file info jendela** pada MacOS itu ditampilkan A file atribut.

Informasi tentang semua file disimpan dalam struktur direktori, yaitu berada pada itu sama perangkat sebagai itu file diri. Khas, A direktori pintu masuk terdiri dari nama file dan pengenal uniknya. Pengidentifikasi pada gilirannya menemukan itu lainnya mengajukan atribut. Dia mungkin mengambil lagi dibandingkan A kilobita ke catatan informasi ini untuk setiap file. Dalam sistem dengan banyak file, ukuran direktori itu sendiri mungkin megabyte atau gigabyte. Karena direktori harus cocok volatilitas file, seperti file, mereka harus disimpan di perangkat dan berada biasanya telah membawa ke dalam memori sedikit demi sedikit, sebagai diperlukan.



Angka 13.1 A info berkas jendela pada macOS.

13.1.2 Mengajukan Operasi

File adalah tipe data abstrak. Untuk mendefinisikan file dengan benar, kita perlu mempertimbangkan operasi yang dapat dilakukan pada file. Sistem operasi dapat menyediakan panggilan sistem untuk membuat, menulis, membaca, memposisikan ulang, menghapus, dan memotong file. Ayo periksa apa yang harus dilakukan sistem operasi untuk melakukan ketujuh hal ini operasi file dasar. Maka akan mudah untuk melihat bagaimana operasi serupa lainnya, seperti sebagai mengganti nama a mengajukan, dapat dilaksanakan.

- **Membuat file** . Dua langkah diperlukan untuk membuat file. Pertama, ruang di sistem file harus ditemukan untuk file tersebut. Kami mendiskusikan bagaimana mengalokasikan ruang untuk itu mengajukan di dalam Bab 14. Kedua, sebuah pintu masuk untuk itu baru mengajukan harus menjadi dibuat di dalam Adirektori.
- **Membuka file** . Daripada membuat semua operasi file menentukan nama file, menyebabkan sistem operasi mengevaluasi nama, memeriksa izin akses- sions, dan seterusnya, semua operasi kecuali membuat dan menghapus memerlukan file membuka() Pertama. Jika sukses, itu membuka panggilan kembali A mengajukan menangani itu adalah digunakan sebagaisebuah argumen di dalam yang lain panggilan.
- **Menulis file** . Untuk menulis file, kami membuat panggilan sistem yang menentukan keduanya buka pegangan file dan informasi yang akan ditulis ke file. Sistem harus menyimpan **penunjuk tulis** ke lokasi di file tempat penulisan berikutnya akan terjadi jika berurutan. Penunjuk tulis harus diperbarui kapan pun A menulis terjadi.
- **Membaca file** . Untuk membaca dari suatu file, kami menggunakan panggilan sistem yang menentukan pegangan file dan di mana (dalam memori) blok file berikutnya seharusnya berada meletakkan. Lagi, itu sistem kebutuhan ke menyimpan A **membaca penunjuk** ke itu lokasi di dalam file tempat pembacaan berikutnya dilakukan, jika berurutan. Setelah dibaca telah terjadi, penunjuk baca diperbarui. Karena suatu proses biasanya baik membaca dari atau menulis ke file, lokasi operasi saat ini bisa disimpan sebagai **posisi file saat ini per proses penunjuk** . Baik yang dibaca maupun menulis operasi menggunakan ini sama penunjuk, penghematan ruang angkasa Dan mengurangi sistem kompleksitas.
- **Memposisikan ulang di dalam A mengajukan** . Itu posisi file saat ini penunjuk dari itu membuka file diposisikan ulang ke nilai tertentu. Reposisi dalam file tidak perlu melibatkan ada yang sebenarnya masukan/keluaran . Ini operasi berkas adalah juga dikenal sebagai mengajukan **mencari** .
- **Menghapus A file** . Ke menghapus A mengajukan, Kami mencari itu direktori untuk itu bernama mengajukan. Setelah menemukan entri direktori terkait, kami melepaskan semua ruang file, jadi itu dia Bisa menjadi digunakan kembali oleh lainnya file, Dan menghapus atau tanda sebagai bebas itu direktori pintu masuk. Perhatikan bahwa beberapa sistem memperbolehkan **tautan keras** — beberapa nama (langsung entri tory) untuk file yang sama. Dalam hal ini isi file sebenarnya tidak dihapus sampai itu tautan terakhir adalah dihapus.
- **Memotong file** . Pengguna mungkin ingin menghapus konten file tetapi

menjaga atributnya. Daripada memaksa pengguna untuk menghapus file lalu membuatnya kembali, fungsi ini memungkinkan semua atribut tetap tidak berubah— kecuali untuk panjang file. File kemudian dapat diatur ulang ke panjang nol, dan ruang filenya Bisa dilepaskan.

Ketujuh operasi dasar ini terdiri dari kumpulan minimal file yang diperlukan operasi. Umum lainnya operasi meliputi menambahkan yang baru informasi ke akhir file yang ada dan mengganti nama file yang ada. Ini primitif operasi kemudian dapat digabungkan untuk melakukan operasi file lainnya. Contohnya, Kami Bisa membuat A menyalin dari A mengajukan oleh menciptakan A baru mengajukan Dan Kemudian membaca dari itu tua Dan menulis ke itu baru. Kami Juga ingin ke memiliki operasi itu mengizinkan A pengguna untuk mendapatkan dan mengatur berbagai atribut file. Misalnya, kita mungkin ingin memilikinya operasi yang memungkinkan pengguna untuk menentukan status file, seperti file panjang, Dan ke mengatur mengajukan atribut, seperti sebagai itu file pemilik.

Sebagai tersebut, paling dari itu mengajukan operasi tersebut melibatkan mencari itu direktori untuk entri yang terkait dengan file bernama. Untuk menghindari hal ini secara terus-menerus pencarian, banyak sistem mengharuskan panggilan sistem open() dilakukan sebelum a mengajukan adalah Pertama digunakan. Itu Pengoperasian sistem terus A meja, ditelepon itu **file terbuka meja** , mengandung informasi tentang semua membuka file. Kapan A mengajukan operasi adalah diminta, file ditentukan melalui indeks ke dalam tabel ini, jadi tidak diperlukan pencarian. Ketika file tidak lagi digunakan secara aktif, file tersebut ditutup oleh proses, dan sistem operasi berpotensi menghapus entri dari tabel file terbuka melepaskan kunci. create() dan delete() adalah panggilan sistem yang berfungsi dengan fungsi tertutup lebih tepatnya dibandingkan membuka file.

Beberapa sistem secara implisit membuka file ketika referensi pertama dibuat. File tersebut secara otomatis ditutup ketika pekerjaan atau program yang dibuka file berakhir. Namun sebagian besar sistem mengharuskan pemrogram membuka a file secara eksplisit dengan panggilan sistem open() sebelum file tersebut dapat digunakan. Itu operasi open() mengambil nama file dan mencari direktori, menyalin entri direktori ke dalam tabel file terbuka. Panggilan open() juga dapat menerima akses- mode informasi — membuat, hanya baca, Baca tulis, hanya tambahan, Dan Jadi pada. Mode ini diperiksa berdasarkan izin file. Jika mode permintaannya adalah diizinkan, itu mengajukan adalah dibuka untuk itu proses. Itu membuka() sistem panggilan khas kembali A penunjuk ke itu pintu masuk di dalam itu membuka file meja. Ini penunjuk, bukan itu sebenarnya mengajukan nama, adalah digunakan di dalam semua masukan/keluaran operasi, menghindari setiap lebih jauh mencari Dan menyederhanakan itu panggilan sistem antarmuka.

Implementasi operasi open() dan close() lebih rumit. rumit di dalam sebuah lingkungan Di mana beberapa proses mungkin membuka itu mengajukan secara simultan- secara baru. Ini mungkin terjadi di dalam A sistem Di mana beberapa berbeda aplikasi membuka itu sama mengajukan pada itu sama waktu. Khas, itu Pengoperasian sistem kegunaan dua tingkat dari intern tabel: A per proses meja Dan A seluruh sistem meja. Itu per proses meja trek semua file itu A proses telah terbuka. Disimpan di dalam ini meja adalah informasi tentang itu proses menggunakan dari itu mengajukan. Untuk contoh, itu saat ini mengajukan penunjuk untuk setiap mengajukan adalah ditemukan Di Sini. Mengakses hak ke itu mengajukan Dan akuntansi informasi Bisa Juga termasuk.

Setiap entri dalam tabel per-proses pada gilirannya menunjuk ke file terbuka di seluruh sistem meja. Tabel seluruh sistem berisi informasi yang tidak bergantung pada proses, misalnya seperti lokasi file pada disk, tanggal akses, dan ukuran file. Setelah file telah dibuka oleh satu proses, tabel seluruh sistem menyertakan entri untuk file tersebut. Ketika proses lain mengeksekusi

open() panggilan, entri baru ditambahkan begitu saja ke itu proses membuka file meja menunjuk ke itu sesuai pintu masuk di dalam itu sistem- meja lebar. Biasanya, tabel file terbuka juga memiliki **jumlah terbuka** yang terkait dengannya setiap file untuk menunjukkan berapa banyak proses yang membuka file tersebut. Setiap penutupan() berkurang ini membuka menghitung, Dan Kapan itu membuka menghitung mencapai nol, itu mengajukan adalah TIDAK lebih lama di dalam menggunakan, Dan itu file pintu masuk adalah DIHAPUS dari itu membuka file meja.

FILE LOCKING IN JAVA

In the Java API, acquiring a lock requires first obtaining the FileChannel for the file to be locked. The lock() method of the FileChannel is used to acquire the lock. The API of the lock() method is

```
FileLock lock(long begin, long end, boolean shared)
```

where begin and end are the beginning and ending positions of the region being locked. Setting shared to true is for shared locks; setting shared to false acquires the lock exclusively. The lock is released by invoking the release() of the FileLock returned by the lock() operation.

The program in Figure 13.2 illustrates file locking in Java. This program acquires two locks on the file file.txt. The lock for the first half of the file is an exclusive lock; the lock for the second half is a shared lock.

Di dalam ringkasan, beberapa bagian-bagian dari informasi adalah terkait dengan sebuah membuka mengajukan.

- **Penunjuk berkas** . Pada sistem yang tidak menyertakan file offset sebagai bagian dari membaca() Dan menulis() sistem panggilan, itu sistem harus melacak itu terakhir membaca- tulis lokasi sebagai penunjuk posisi file saat ini. Penunjuk ini unik untuk setiap proses yang beroperasi pada file dan oleh karena itu harus dipisahkan darinya itu di disk mengajukan atribut.
- **Jumlah pembukaan file** . Saat file ditutup, sistem operasi harus menggunakan kembali file tersebut entri tabel file terbuka, atau bisa kehabisan ruang di tabel. Banyak proses mungkin telah membuka file, dan sistem harus menunggu hingga file terakhir mengajukan ke menutup sebelum menghapus itu membuka file meja pintu masuk. Itu buka file menghitung melacak jumlah pembukaan dan penutupan dan mencapai nol pada penutupan terakhir. Itu sistem lalu bisa menghapus itu pintu masuk.
- **Lokasi file** . Sebagian besar operasi file memerlukan sistem untuk membaca atau menulis data di dalam itu mengajukan. Itu informasi diperlukan ke menemukan itu mengajukan (di manapun dia terletak, baik di penyimpanan massal, di server file di seluruh jaringan, atau di A penggerak RAM) adalah disimpan di dalam Penyimpanan Jadi itu itu sistem melakukan bukan memiliki ke membaca dia dari itu direktori struktur untuk setiap operasi.
- **Hak akses** . Setiap proses membuka file dalam mode akses. Informasi ini- tion adalah disimpan pada itu per proses meja Jadi itu Pengoperasian sistem Bisa mengizinkan ataumembantah I/O berikutnya permintaan.

Beberapa sistem operasi menyediakan fasilitas untuk mengunci file yang terbuka (atau file detik). tions dari sebuah file). Kunci file memungkinkan satu proses mengunci file dan mencegah proses lainnya proses dari memperoleh mengakses ke dia. Mengajukan kunci adalah berguna untuk file itu adalah bersama oleh beberapa proses— misalnya, file log sistem yang dapat diakses dan diubah oleh a jumlah proses di dalam itu sistem.

Kunci file menyediakan fungsionalitas yang mirip dengan kunci pembaca–

penulis, tercakup dalam Bagian 7.1.2. Kunci **bersama** mirip dengan kunci pembaca yang dapat dilakukan oleh beberapa proses memperoleh kunci secara bersamaan. Kunci **eksklusif** berperilaku seperti kunci penulis; hanya satu proses pada A waktu Bisa mendapatkan seperti A kunci. Dia adalah penting ke catatan itu bukan

```
impor java.io.*;
impor java.nio.saluran.*;

publik kelas Contoh Penguncian{
  publik statis terakhir boolean EKSKLUSIF = PALSU;
  publik statis terakhir boolean DIBAGIKAN = BENAR;

  publik statis ruang kosong utama (String argumen[]) melempar{
    Pengecualian IOKunci File kunci bersama = batal;
    Kunci File Kunci eksklusif = batal;

    mencoba{
      File Akses Acak raf = baru RandomAccessFile("file.txt","rw");

      // mendapatkan itu saluran untuk
      itu mengajukan Saluran File bab =
      raf.getChannel();

      // ini kunci itu Pertama setengah dari itu mengajukan -
      eksklusif Kunci eksklusif = ch.lock(0, raf.panjang()/2,
      EKSKLUSIF);

      /** Sekarang memodifikasi itu data . . . */

      // melepaskan itu kunci
      eksklusifLock.release();

      // ini kunci itu Kedua setengah dari itu mengajukan -
      bersama kunci bersama =
      ch.lock(raf.length()/2+1,raf.length(),BERBAGI);

      /** Sekarang membaca itu data . . . */

      // lepaskan kunci
      sharedLock.rilis();
    } menangkap (java.io.IOException  {
      yaitu)Sistem.err.println(ioe);
    }
    Akhirnya{
      jika (Kunci eksklusif != batal)
              eksklusifLock.release();
      jika (Kunci bersama != batal)
              sharedLock.rilis();
    }
  }
}
```

Angka 13.2 Penguncian file contoh di dalam Jawa.

semua Pengoperasian sistem menyediakan keduanya jenis dari kunci: beberapa sistem menyediakan hanyaeksklusif mengajukan mengunci.

Selain itu, sistem operasi dapat memberikan perintah **wajib** atau **saran. maaf** mekanisme penguncian file. Dengan penguncian wajib, setelah suatu proses diperoleh sebuah eksklusif kunci, itu Pengoperasian sistem akan mencegah setiap lainnya proses dari

mengakses itu terkunci mengajukan. Untuk contoh, menganggap A proses memperoleh sebuah eksklusif mengunci file system.log . Jika kami mencoba membuka system.log dari yang lain proses— misalnya, editor teks— sistem operasi akan mencegah akses sampai itu eksklusif kunci adalah dilepaskan. Kalau tidak, jika itu kunci adalah penasehat, Kemudian itu Pengoperasian sistem akan bukan mencegah itu teks editor dari memperoleh mengakses ke sistem- tem.log . Lebih tepatnya, itu teks editor harus menjadi tertulis Jadi itu dia secara manual memperoleh itu kunci sebelum mengakses file. Dengan kata lain, jika skema penguncian bersifat wajib, Sejarahnya, sistem operasi memastikan integritas penguncian. Untuk penguncian penasehat, itu terserah pengembang perangkat lunak untuk memastikan bahwa kunci diperoleh dengan benar Dan dilepaskan. Sebagai A umum aturan, jendela Pengoperasian sistem mengambil wajib mengunci, Dan UNIX sistem mempekerjakan penasehat kunci.

Penggunaan kunci file memerlukan tindakan pencegahan yang sama seperti sinkronisasi proses biasa. kronik. Untuk contoh, pemrogram mengembangkan pada sistem dengan mandat- penguncian tory harus berhati-hati untuk menahan kunci file eksklusif hanya saat berada mengakses itu mengajukan. Jika tidak, mereka akan mencegah lainnya proses dari mengakses itu mengajukan sebagai Sehat. Lebih-lebih lagi, beberapa Pengukuran harus menjadi diambil ke memastikan itu dua atau lebih proses tidak mengalami kebuntuan saat mencoba memperolehnya mengajukan kunci.

13.1.3 Mengajukan Jenis

Kapan Kami desain A mengajukan sistem— memang, sebuah seluruh Pengoperasian sistem— kita selalu pertimbangkan apakah sistem operasi harus mengenali dan mendukung jenis file. Jika sistem operasi mengenali jenis file, maka sistem operasi dapat beroperasi pada itu mengajukan di dalam wajar cara. Untuk contoh, A umum kesalahan terjadi Kapan A pengguna mencoba mengeluarkan bentuk objek biner dari suatu program. Upaya ini biasanya menghasilkan sampah; Namun, upaya tersebut dapat berhasil jika sistem operasinya memiliki pernah diberi tahu itu berkas adalah A objek biner program.

A umum teknik untuk menerapkan mengajukan jenis adalah ke termasuk itu jenis sebagai bagian dari nama file. Itu nama dibagi menjadi dua bagian — sebuah nama Dan sebuah perpanjangan, biasanya terpisah oleh A periode (Angka 13.3). Di dalam ini jalan, itu pengguna dan sistem operasi dapat mengetahui dari namanya saja apa jenis filenya adalah. Paling Pengoperasian sistem mengizinkan pengguna ke menentukan A mengajukan nama sebagai A urutan karakter diikuti dengan titik dan diakhiri dengan perpanjangan yang dibuat hingga karakter tambahan. Contohnya termasuk resume.docx , server.c , dan ReaderThread.cpp .

Sistem menggunakan ekstensi untuk menunjukkan jenis file dan jenisnya operasi yang dapat dilakukan pada file itu. Hanya file dengan .com , .exe , atau .sh ekstensi dapat dijalankan, misalnya. File .com dan .exe adalah dua bentuk dari biner dapat dieksekusi file, sedangkan itu .SH mengajukan adalah A **kerang naskah** mengandung, di dalam Format ASCII , perintah ke sistem operasi. Program aplikasi juga menggunakan ekstensi ke menunjukkan mengajukan jenis di dalam yang mereka adalah tertarik. Untuk contoh, Kompiler Java mengharapkan file sumber memiliki ekstensi .java , dan Microsoft Kata kata prosesor mengharapkan -nya file ke akhir dengan A .dokter atau .docx perpanjangan. Ini ekstensi adalah bukan selalu diperlukan, Jadi A pengguna mungkin menentukan A mengajukan tanpa

ekstensinya (untuk menyimpan pengetikan), dan aplikasi akan mencari file dengan nama yang diberikan dan ekstensi yang diharapkan. Karena ekstensi ini tidak didukung oleh sistem operasi, hal ini dapat dianggap sebagai " petunjuk " bagiaplikasi itu beroperasi pada mereka.

Mempertimbangkan, juga, itu MacOS Pengoperasian sistem. Di dalam ini sistem, setiap mengajukan memiliki A jenis, seperti sebagai .aplikasi (untuk aplikasi). Setiap mengajukan Juga memiliki A pencipta atribut

mengajukan jenis	biasa perpanjangan	fungsi
dapat dieksekusi	exe, com, bin atau tidak ada	mesin siap dijalankan-bahasa program
obyek	keberatan, Hai	dikompilasi, mesin bahasa, bukan terhubung
sumber kode	c, cc, java, perl, asm	kode sumber dalam berbagaibahasa
kelompok	kelelawar, sh	perintah ke perintah penerjemah
markup	xml, html, teks	tekstual data, dokumen
kata prosesor	xml, rtf, dok	berbagai pengolah kata format
perpustakaan	lib, a, jadi, dll	perpustakaan rutinitas untukpemrogram
mencetak atau melihat	gif, pdf, jpg	ASCII atau file biner dalam aformat untuk mencetak atau melihat
arsip	rar, zip, tar	file terkait dikelompokkan menjadi satu file, terkadang com-ditekan, untuk pengarsipan atau penyimpanan
multimedia	mpeg, bergerak, mp3,mp4, avi	file biner yang berisi audio atau A/V informasi

Angka 13.3 Umum mengajukan jenis.

berisi nama program yang membuatnya. Atribut ini diatur oleh sistem operasi selama panggilan `create()` , sehingga penggunaannya diberlakukan dan didukung oleh sistem. Misalnya, file yang dihasilkan oleh pengolah kata memiliki itu kata prosesor nama sebagai -nya pencipta. Kapan itu pengguna terbuka itu mengajukan, oleh mengklik dua kali itu mouse pada itu ikon mewakili itu mengajukan, itu kata prosesor adalah dipanggil secara otomatis, Dan itu mengajukan adalah sarat, siap ke menjadi diedit.

Itu UNIX sistem kegunaan A **sihir nomor** disimpan pada itu awal dari beberapa file biner untuk menunjukkan jenis data dalam file (misalnya format dari file gambar). Demikian pula, ia menggunakan angka ajaib teks di awal teks file ke menunjukkan itu jenis dari mengajukan (yang kerang bahasa A naskah adalah tertulis di dalam) dan seterusnya. (Untuk detail lebih lanjut tentang angka ajaib dan jargon komputer lainnya, melihat http://www.catb.org/esr/jargon/ .) Bukan semua file memiliki sihir angka, Jadi fitur sistem tidak dapat hanya didasarkan pada informasi ini. UNIX tidak catat juga nama program pembuatnya. UNIX mengizinkan nama file-perpanjangan petunjuk, Tetapi ini ekstensi adalah juga tidak diberlakukan juga bukan tergantung pada oleh itu Pengoperasian sistem; mereka adalah dimaksudkan sebagian besar ke bantuan pengguna di dalam menentukan Apa jenis dari isi itu mengajukan mengandung. Ekstensi Bisa menjadi digunakan atau diabaikan oleh A diberikan aplikasi, Tetapi itu adalah ke atas ke itu aplikasi programmer.

13.1.4 Mengajukan Struktur

Jenis file juga dapat digunakan untuk menunjukkan struktur internal file. Sumber dan file objek memiliki struktur yang sesuai dengan harapan program itu membaca mereka. Lebih jauh, yakin file harus sesuai ke A diperlukan struktur itu

dipahami oleh sistem operasi. Misalnya saja sistem operasi memerlukan itu sebuah dapat dieksekusi mengajukan memiliki A spesifik struktur Jadi itu dia Bisa menentukan di mana di memori untuk memuat file dan di mana lokasi instruksi pertama adalah. Beberapa sistem operasi memperluas gagasan ini ke dalam serangkaian sistem yang didukung mengajukan struktur, dengan set dari spesial operasi untuk memanipulasi file dengan itustruktur.

Poin ini membawa kita pada salah satu kelemahan memiliki pengoperasian sistem mendukung banyak struktur file: membuat sistem operasi menjadi besar dan tidak praktis. Jika sistem operasi mendefinisikan lima struktur file yang berbeda, itu kebutuhan ke berisi itu kode ke mendukung ini mengajukan struktur. Di dalam tambahan, dia mungkin menjadi perlu untuk mendefinisikan setiap file sebagai salah satu jenis file yang didukung oleh operasi sistem. Ketika aplikasi baru memerlukan informasi yang terstruktur dengan cara yang tidak didukung oleh itu Pengoperasian sistem, berat masalah mungkin hasil.

Misalnya, asumsikan bahwa suatu sistem mendukung dua jenis file: file teks (terdiri dari karakter ASCII yang dipisahkan oleh carriage return dan line feed) Dan dapat dieksekusi biner file. Sekarang, jika Kami (sebagai pengguna) ingin ke mendefinisikan sebuah terenkripsi file untuk melindungi konten agar tidak dibaca oleh orang yang tidak berwenang, kami mungkin tidak menemukan jenis file yang sesuai. File terenkripsi bukan baris teks ASCII melainkan (tampaknya) bit acak. Meskipun tampaknya biner mengajukan, dia adalah bukan dapat dieksekusi. Sebagai A hasil, Kami mungkin memiliki ke mengelakkan atau penyalahgunaan itu Pengoperasian sistem tipe file mekanisme atau meninggalkan kita enkripsi skema.

Beberapa sistem operasi memberlakukan (dan mendukung) jumlah file minimal struktur. Pendekatan ini telah diadopsi di UNIX , Windows, dan lainnya. UNIX menganggap setiap file sebagai urutan byte 8-bit; tidak ada interpretasi dari bit-bit ini dibuat oleh sistem operasi. Skema ini memberikan hasil maksimal fleksibilitas tetapi sedikit dukungan. Setiap program aplikasi harus menyertakan programnya sendiri kode untuk menafsirkan file input ke struktur yang sesuai. Namun, semuanya sistem operasi harus mendukung setidaknya satu struktur— yaitu struktur yang dapat dieksekusi mengajukan-Jadi bahwa sistem adalah mampu untuk memuat dan lari program.

13.1.5 Intern Mengajukan Struktur

Secara internal, menemukan sebuah mengimbangi di dalam A mengajukan Bisa menjadi rumit untuk itu Pengoperasian sistem. Sistem disk biasanya memiliki ukuran blok yang ditentukan dengan baik ukuran suatu sektor. Semua I/O disk dilakukan dalam satuan satu blok (fisik catatan), Dan semua blok adalah itu sama ukuran. Dia adalah tidak sepertinya itu itu fisik catatan ukuran akan tepat cocok itu panjang dari itu diinginkan logis catatan. Logis catatan bahkan panjangnya mungkin berbeda-beda. Mengemas sejumlah catatan logis menjadi fisik blok adalah umum larutan ke ini masalah.

Misalnya UNIX sistem operasi mendefinisikan semua file secara sederhana aliran byte. Setiap byte dapat dialamatkan secara individual berdasarkan offsetnya dari awal (atau akhir) dari itu mengajukan. Di dalam ini kasus, itu logis catatan ukuran adalah 1 byte. Itu mengajukan sistem secara otomatis paket Dan membongkar byte ke dalam fisik disk blok— mengatakan, 512 byte per memblokir-sebagai diperlukan.

Ukuran catatan logis, ukuran blok fisik, dan teknik pengepakan menentukan milikku Bagaimana banyak logis catatan adalah di dalam setiap fisik memblokir. Itu sedang mengemas Bisa menjadi dilakukan baik oleh

program aplikasi pengguna atau oleh sistem operasi. Di dalam dalam kasus apa pun, file tersebut dapat dianggap sebagai rangkaian blok. Semua I/O dasar fungsi beroperasi dalam bentuk blok. Konversi dari catatan logis ke fisik blok adalah relatif sederhana perangkat lunak masalah.

Karena ruang disk selalu dialokasikan dalam blok, sebagian terakhir blok setiap file umumnya terbuang sia-sia. Jika setiap blok berukuran 512 byte, misalnya, maka file sebesar 1.949 byte akan dialokasikan empat blok (2.048 byte); yang terakhir 99 byte akan menjadi sia-sia. Itu limbah terjadi ke menyimpan semuanya di dalam unit blok (bukan byte) adalah fragmentasi internal. Semua sistem file menderita dari fragmentasi internal; semakin besar ukuran blok, semakin besar internalnyafragmentasi.

13.2 Mengakses Metode

File toko informasi. Kapan dia adalah digunakan, ini informasi harus menjadi diakses Dan membaca ke dalam memori komputer. Informasi dalam file dapat diakses di beberapa cara. Beberapa sistem hanya menyediakan satu metode akses untuk file. Yang lain (seperti sistem operasi mainframe) mendukung banyak metode akses, dan memilih itu Kanan satu untuk A tertentu aplikasi adalah A besar desain masalah.

13.2.1 Sekuensial Mengakses

Metode akses yang paling sederhana adalah **akses sekuensial** . Informasi dalam file tersebut adalah diproses secara berurutan, satu demi satu catatan. Mode akses ini sejauh ini yang paling umum; misalnya, editor dan kompiler biasanya mengakses file di ini mode.

Membaca dan menulis merupakan sebagian besar operasi pada file. Sebuah bacaan_operasi— membaca next() — membaca bagian file selanjutnya secara otomatis memajukan penunjuk file, yang melacak lokasi I/O . Begitu pula dengan tulisannya operasi— tulis next() — ditambahkan ke akhir file dan dilanjutkan ke akhir dari itu baru saja tertulis bahan (itu baru akhir dari mengajukan). Seperti A mengajukan Bisa menjadi mengatur ulang ke itu awal, Dan pada beberapa sistem, A program mungkin menjadi mampu ke melewati maju atau mundur n catatan untuk beberapa bilangan bulat n —mungkin hanya untuk $n = 1$. Berurutan akses, yang digambarkan pada Gambar 13.4, didasarkan pada model tape dari file dan bekerja sebagai Sehat pada akses berurutan perangkat sebagai dia melakukan pada akses acak yang.

13.2.2 Langsung Mengakses

Lain metode adalah **langsung mengakses** (atau **relatif akses**). Di Sini, A mengajukan adalah dibuat ke atas **catatan logis** dengan panjang tetap yang memungkinkan program membaca dan menulis catatan dengan cepat tanpa urutan tertentu. Metode akses langsung didasarkan pada disk model file, karena disk memungkinkan akses acak ke blok file mana pun. Untuk langsung mengakses, itu mengajukan adalah dilihat sebagai A bernomor urutan dari blok atau catatan. Dengan demikian,

Angka 13.4 Akses berurutan mengajukan.

Kami mungkin membaca memblokir 14, Kemudian membaca memblokir 53, Dan Kemudian menulis memblokir 7. Di sana adalah TIDAK pembatasan pada itu memesan dari membaca atau menulis untuk A akses langsung mengajukan.

File akses langsung sangat berguna untuk akses langsung ke jumlah besar informasi. Basis data sering kali berjenis ini. Ketika pertanyaan tentang a subjek tertentu tiba, kita hitung blok mana yang berisi jawaban dan Kemudian membaca itu memblokir secara langsung ke menyediakan itu diinginkan informasi.

Sebagai contoh sederhana, pada sistem reservasi maskapai penerbangan, kita mungkin menyimpan semuanya informasi tentang penerbangan tertentu (misalnya, penerbangan 713) di blok tersebut diidentifikasi berdasarkan nomor penerbangan. Dengan demikian, jumlah kursi yang tersedia untuk penerbangan 713 disimpan di blok 713 dari file reservasi. Untuk menyimpan informasi tentang a kumpulan yang lebih besar, seperti orang, kita mungkin menghitung fungsi hash pada kumpulan orang nama atau cari indeks kecil di memori untuk menentukan blok yang akan dibaca dan mencari.

Untuk metode akses langsung, operasi file harus dimodifikasi termasuk itu memblokir nomor sebagai A parameter. Dengan demikian, Kami memiliki baca(n) , Di mana *n* adalah nomor blok, bukan read next() , dan write(n) daripada menulis next() . Pendekatan alternatif adalah dengan mempertahankan read next()_ dan tulis next() dan untuk menambahkan file posisi operasi(n) di mana *n* adalah memblokir nomor. Kemudian, ke memengaruhi A baca(n) , Kami akan posisi berkas(n) DanKemudian membaca Berikutnya() .

Itu memblokir nomor asalkan oleh itu pengguna ke itu Pengoperasian sistem adalah biasanya nomor **blok relatif** . Nomor blok relatif adalah indeks relatif terhadap awal dari itu mengajukan. Dengan demikian, itu Pertama relatif memblokir dari itu mengajukan adalah 0, itu Berikutnya adalah 1, Dan Jadi pada, bahkan meskipun itu mutlak disk alamat mungkin menjadi 14703 untuk itu blok pertama dan 3192 untuk blok kedua. Penggunaan nomor blok relatif memungkinkan sistem operasi untuk memutuskan di mana file harus ditempatkan (disebut **alokasi masalah** , sebagai Kami membahas di dalam Bab 14) Dan membantu ke mencegah itu pengguna dari mengakses porsi dari itu mengajukan sistem itu mungkin bukan menjadi bagian dari dia mengajukan. Beberapa sistem awal milik mereka relatif memblokir angka pada 0; yang lain awal pada 1.

Bagaimana, Kemudian, melakukan itu sistem memuaskan A meminta untuk catatan *N* di dalam A mengajukan? Asumsikan- Jika kita memiliki panjang rekaman logis *L*, permintaan untuk rekaman *N* diubah menjadi permintaan I/O untuk *L* byte dimulai dari lokasi $L * (N)$ di dalam file (dengan asumsi catatan pertama adalah $N = 0$). Karena catatan logis berukuran tetap, ini juga mudah ke membaca, menulis, atau menghapus A catatan.

Tidak semua sistem operasi mendukung akses sekuensial dan langsung file. Beberapa sistem hanya mengizinkan akses file berurutan; yang lain hanya mengizinkan langsung mengakses. Beberapa sistem memerlukan itu A mengajukan menjadi didefinisikan sebagai sekuensial atau langsung Kapan itu dibuat. File seperti itu hanya dapat diakses dengan cara yang konsisten dengan file tersebut pernyataan. Kami Bisa dengan mudah mensimulasikan sekuensial mengakses pada A akses langsung mengajukan oleh cukup simpan variabel *cp* yang menentukan posisi kita saat ini, seperti yang ditunjukkan pada Angka 13.5. Simulasi A akses langsung mengajukan pada A akses berurutan mengajukan, Namun, adalah sangat tidak efisien Dan

kikuk.

13.2.3 Lainnya Mengakses Metode

Metode akses lainnya dapat dibangun di atas metode akses langsung. Ini metode umumnya melibatkan itu konstruksi dari sebuah indeks untuk itu mengajukan. Itu **indeks** , menyukai sebuah indeks di dalam itu kembali dari A buku, mengandung petunjuk ke itu bermacam-macam blok. Ke menemukan catatan dalam file, pertama-tama kita mencari indeks dan kemudian menggunakan penunjuk untuk mengakses itu mengajukan secara langsung Dan ke menemukan itu diinginkan catatan.

sekuensial mengakses	penerapan untuk akses langsung
mengatur ulang	cp = 0;
baca_berikutnya	membaca cp; cp = cp + 1;
tulis_berikutnya	menulis cp; cp = cp + 1;

Angka 13.5 Simulasi dari sekuensial mengakses pada A akses langsung mengajukan.

Untuk contoh, A harga eceran mengajukan mungkin daftar itu universal produk kode (UPC)untuk item, dengan itu terkait harga. Setiap catatan terdiri dari A 10 angka UPC DanA 6 angka harga, untuk A 16 byte catatan. Jika kita disk memiliki 1.024 byte per memblokir, KamiBisa toko 64 catatan per memblokir. A mengajukan dari 120.000 catatan akan menempati tentang 2.000 blok (2 juta byte). Oleh penyimpanan itu mengajukan diurutkan oleh UPC , Kami Bisa mendefinisikansebuah indeks terdiri dari itu Pertama UPC di dalam setiap memblokir. Ini indeks akan memiliki 2.000entri dari 10 angka setiap, atau 20.000 byte, Dan dengan demikian bisa menjadi disimpan di dalam Penyimpanan. Kemenemukan itu harga dari A tertentu barang, Kami Bisa membuat A biner mencari dari itu indeks.Dari ini mencari, Kami mempelajari tepat yang memblokir mengandung itu diinginkan catatan Dan mengakses itu memblokir. Ini struktur memungkinkan kita ke mencari A besar mengajukan sedang mengerjakan kecil masukan/keluaran . Dengan besar file, itu indeks mengajukan sendiri mungkin menjadi terlalu besar ke menjadi disimpan di dalam Penyimpanan. Satu larutan adalah ke membuat sebuah indeks untuk itu indeks mengajukan. Itu utama indeksmengajukan mengandung petunjuk ke sekunder indeks file, yang titik ke itu sebenarnya data item.

Misalnya, metode akses sekuensial terindeks (ISAM) IBM menggunakan metode akses sekuensial kecil indeks master yang menunjuk ke blok disk dari indeks sekunder. Yang sekunder indeks blok titik ke itu sebenarnya mengajukan blok. Itu mengajukan adalah disimpan diurutkan pada A didefinisikan kunci. Untuk menemukan item tertentu, pertama-tama kita melakukan pencarian biner pada indeks master, yang memberikan nomor blok indeks sekunder. Blok ini dibaca di, dan sekali lagi pencarian biner digunakan untuk menemukan blok yang berisi apa yang diinginkan catatan. Akhirnya, ini memblokir adalah dicari secara berurutan. Di dalam ini jalan, setiap catatan Bisa menjadi terletak dari -nya kunci oleh pada paling dua akses langsung membaca. Angka 13.6 menunjukkan A serupa situasi sebagai dilaksanakan oleh O pena VMS indeks Dan relatif file.

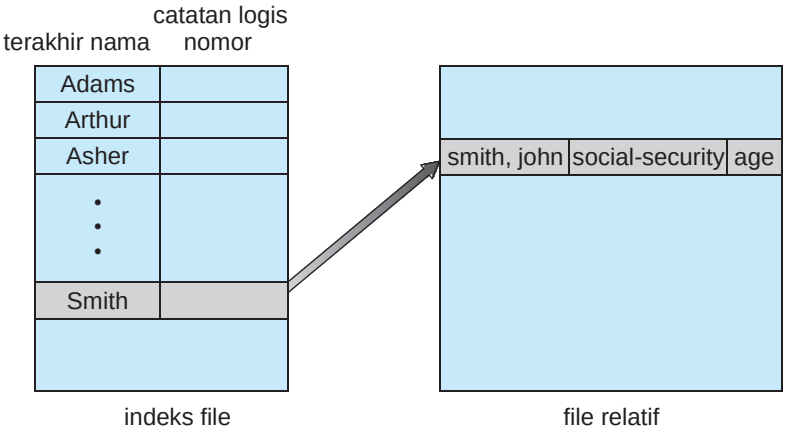
13.3 Direktori Struktur

Direktori dapat dilihat sebagai tabel simbol yang menerjemahkan nama file menjadi milik mereka mengajukan kontrol blok. Jika Kami mengambil seperti A melihat, Kami melihat itu itu direktori diri dapat diatur dalam banyak cara. Organisasi harus mengizinkan kita untuk memasukkan entri, ke menghapus entri, ke mencari untuk sebuah bernama pintu masuk, dan

untuk daftar semua entri di direktori. Pada bagian ini, kita memeriksa beberapa skema untuk mendefinisikan logis struktur dari itu direktori sistem.

Kapan mempertimbangkan A tertentu direktori struktur, Kami membutuhkan ke menyimpan di dalam pikiran itu operasi itu adalah menjadi dilakukan pada sebuah direktori:

- **Mencari untuk A file** . Kami membutuhkan ke menjadi mampu ke mencari A direktori struktur ke menemukanitu pintu masuk untuk A tertentu mengajukan. Sejak file memiliki simbolis nama, Dan serupa



Angka 13.6 Contoh dari indeks Dan relatif file.

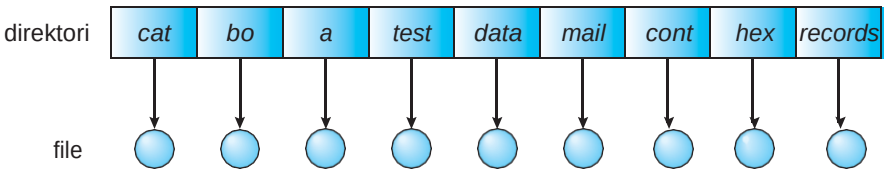
nama mungkin menunjukkan A hubungan di antara file, Kami mungkin ingin ke menjadi mampu ke menemukan semua file yang nama cocok A tertentu pola.

- **Membuat A file** . Baru file membutuhkan ke menjadi dibuat Dan ditambahkan ke itu direktori.
- **Menghapus A mengajukan** . Ketika sebuah mengajukan adalah TIDAK lebih lama diperlukan, Kami ingin ke menjadi mampu ke menghapus dia dari itu direktori. Catatan A menghapus daun-daun A lubang di dalam itu direktori struktur dan sistem file mungkin memiliki metode untuk mendefrag direktori struktur.
- **Daftar direktori** . Kita harus dapat membuat daftar file dalam direktori dan isi dari itu direktori pintu masuk untuk setiap mengajukan di dalam itu daftar.
- **Ganti nama file** . Karena nama file mewakili isinya kepada penggunanya, kita harus bisa mengganti nama saat isi atau penggunaan file tersebut perubahan. Mengganti nama file juga memungkinkan posisinya di dalam direktoristruktur untuk diganti.
- **Melintasi sistem file** . Kami mungkin ingin mengakses setiap direktori dan setiap direktori mengajukan di dalam A direktori struktur. Untuk keandalan, dia adalah A Bagus ide ke menyimpan itu isi dan struktur dari itu seluruh mengajukan sistem pada reguler interval. Sering, kami melakukan ini dengan menyalin semua file ke pita magnetik, penyimpanan sekunder lainnya, atau melintasi jaringan ke sistem lain atau cloud. Teknik ini menyediakan A cadangan menyalin di dalam kasus dari sistem kegagalan. Di dalam tambahan, jika A mengajukan adalah TIDAK lebih lama di dalam menggunakan, itu mengajukan Bisa menjadi disalin itu cadangan target Dan itu disk ruang angkasa dari itu mengajukan dilepaskan untuk digunakan kembali oleh file lain.

Di dalam itu mengikuti bagian, Kami menggambarkan itu paling umum skema untuk mendefinisikanitu logis struktur tentu saja direktori.

13.3.1 Direktori Tingkat Tunggal

Struktur direktori yang paling sederhana adalah direktori satu tingkat. Semua file dikon- tertahan di dalam itu sama direktori, yang adalah mudah ke mendukung Dan memahami (Angka13.7).



Angka 13.7 Tingkat tunggal direktori.

Namun, direktori satu tingkat memiliki keterbatasan yang signifikan ketika nomor dari file meningkat atau Kapan itu sistem memiliki lagi dibandingkan satu pengguna. Sejak semua file adalah di dalam itu sama direktori, mereka harus memiliki unik nama. Jika dua pengguna panggilan milik mereka data mengajukan `tes.txt` , Kemudian itu nama yang unik aturan adalah dilanggar. Untuk contoh, dalam satu kelas pemrograman, 23 siswa memanggil program tersebut untuk kedua kalinya tugas `prog2.c` ; 11 lainnya menyebutnya `tugas2.c` . Untungnya, sebagian besar file sistem mendukung nama file hingga 255 karakter, sehingga relatif mudah Pilih unik mengajukan nama.

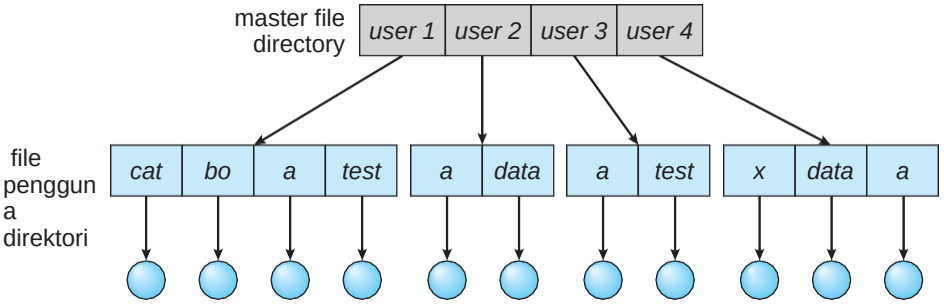
Bahkan satu pengguna pada direktori satu tingkat mungkin merasa sulit untuk mengingatnya. ber nama semua file seiring bertambahnya jumlah file. Hal ini biasa terjadi bagi pengguna untuk memiliki ratusan file pada satu sistem komputer dan setara nomor dari tambahan file pada lain sistem. Penyimpanan melacak dari Jadi banyak file adalahA tugas yang menakutkan.

13.3.2 Dua Tingkat Direktori

Sebagai Kami memiliki terlihat, A tingkat tunggal direktori sering mengarah ke kebingungan dari mengajukan nama di antara pengguna yang berbeda. Solusi standarnya adalah membuat direktori terpisah untuk setiap pengguna.

Dalam struktur direktori dua tingkat, setiap pengguna memiliki **file penggunanya sendiri arah- cerita** (**UFD**). UFD memiliki struktur serupa, namun masing-masing daftar saja file dari A lajang pengguna. Kapan A pengguna pekerjaan dimulai atau A pengguna log di dalam, itu sistem **menguasai file direktori** (**MFD**) dicari. MFD diindeks berdasarkan nama pengguna atau akun nomor, Dan setiap pintu masuk poin ke UFD tersebut untuk itu pengguna (Angka 13.8).

Kapan A pengguna merujuk ke A tertentu mengajukan, hanya miliknya memiliki UFD adalah mencari. Dengan demikian, berbeda pengguna mungkin memiliki file dengan itu sama nama, sebagai panjang sebagai semua itu mengajukan nama dalam setiap UFD bersifat unik. Untuk membuat file untuk pengguna, sistem operasi pencarian hanya itu milik pengguna UFD ke memastikan apakah lain mengajukan dari itu nama



Angka 13.8 Dua tingkat direktori struktur.

ada. Untuk menghapus file, sistem operasi membatasi pencariannya pada UFD lokal ; dengan demikian, dia tidak bisa secara tidak sengaja menghapus lain milik pengguna mengajukan itu memiliki itu sama nama.

Itu pengguna direktori diri harus menjadi dibuat Dan dihapus sebagai diperlukan. A spesial sistem program adalah dijalankan dengan yang sepantasnya pengguna nama dan akun informasi. Program ini membuat UFD baru dan menambahkan entri ke dalamnya MFD . Itu eksekusi dari ini program mungkin menjadi terbatas ke sistem administrasi- tor. Alokasi ruang disk untuk direktori pengguna dapat ditangani dengan teknik dibahas di dalam Bab 14 untuk file diri.

Meskipun itu dua tingkat direktori struktur memecahkan itu tabrakan nama masalah- lem, dia tetap memiliki kekurangan. Ini struktur secara efektif terisolasi satu pengguna dari lain. Isolasi adalah sebuah keuntungan Kapan itu pengguna adalah sama sekali mandiri tetapi merupakan kerugian ketika pengguna ingin bekerja sama dalam beberapa tugas dan melakukan hal tersebut mengakses satu milik orang lain file. Beberapa sistem secara sederhana Mengerjakan bukan mengizinkan lokal pengguna file ke menjadi diakses oleh pengguna lain.

Jika akses diizinkan, satu pengguna harus memiliki kemampuan untuk memberi nama file di direktori pengguna lain. Untuk memberi nama file tertentu secara unik dalam dua tingkat direktori, kita harus memberikan nama pengguna dan nama file. Dua tingkat direktori dapat dianggap sebagai pohon, atau pohon terbalik, dengan tinggi 2. Akar pohon adalah MFD . Keturunan langsungnya adalah UFD . Keturunan dari itu UFD s adalah itu file diri. Itu file adalah itu daun-daun dari itu pohon. Menentukan nama pengguna dan nama file mendefinisikan jalur di pohon dari root (MFD) ke daun (file yang ditentukan). Jadi, nama pengguna dan nama file menentukan **jalur nama** . Setiap file dalam sistem memiliki nama jalur. Untuk memberi nama file secara unik, pengguna harus mengetahui jalur nama dari itu mengajukan diinginkan.

Misalnya, jika pengguna A ingin mengakses file pengujiannya sendiri yang bernama test.txt , dia cukup merujuk ke `test.txt` . Untuk mengakses file bernama `test.txt` pengguna B (dengan entri direktori nama `pengguna`), Namun, dia mungkin memiliki ke merujuk ke `/userb/test.txt` . Setiap sistem memiliki sintaksnya sendiri untuk memberi nama file secara langsung. cerita lainnya dibandingkan itu milik pengguna memiliki.

Sintaks tambahan diperlukan untuk menentukan volume file. Contohnya, di Windows, volume ditentukan dengan huruf diikuti dengan titik dua. Jadi, sebuah spesifikasi file mungkin `C: \ userb \ test` . Beberapa sistem melangkah lebih jauh Dan memisahkan itu volume, direktori nama, Dan mengajukan nama bagian dari itu spesifik- fiksi. Di dalam Wahai pena VMS , untuk contoh, itu mengajukan `login.com` mungkin menjadi ditentukan sebagai: `u:[sst.crissmeyer]login.com;1` , Di mana `kamu` adalah itu nama dari itu volume, `sst` adalah itu nama dari itu direktori, `crissmeyer` adalah itu nama dari itu subdirektori, Dan `1` adalah itu Versi: kapan nomor. Lainnya sistem— seperti itu sebagai UNIX Dan Linux — sederhananya merawat itu volume nama sebagai bagian dari itu direktori nama. Itu Pertama nama diberikan adalah itu dari itu volume, Dan itu istirahat adalah itu direktori Dan mengajukan. Untuk contoh,

`/u/pgalvin/test` mungkin menentukan volume `u` , direktori `pgalvin` , dan file `test` .A spesial contoh dari ini situasi terjadi dengan itu sistem file. Program asalkan sebagai bagian dari itu sistem-pemuat, perakit, kompiler, kegunaan rute-gigi, perpustakaan, Dan Jadi pada- adalah umumnya

didefinisikan sebagai file. Kapan itu sesuaiperintah adalah diberikan ke itu Pengoperasian sistem, ini file adalah membaca oleh itu pemuatDan dieksekusi. Banyak memerintah penerjemah secara sederhana merawat seperti A memerintah sebagai itu nama dari A mengajukan ke memuat Dan menjalankan. Di dalam itu direktori sistem sebagai Kami didefinisikan diadi atas, ini mengajukan nama akan menjadi dicari untuk di dalam itu saat ini UFD . Satu larutanakan menjadi ke menyalin itu sistem file ke dalam setiap UFD . Namun, penyalinan semua itusistem file akan limbah sebuah sangat besar jumlah dari ruang angkasa. (Jika itu sistem file

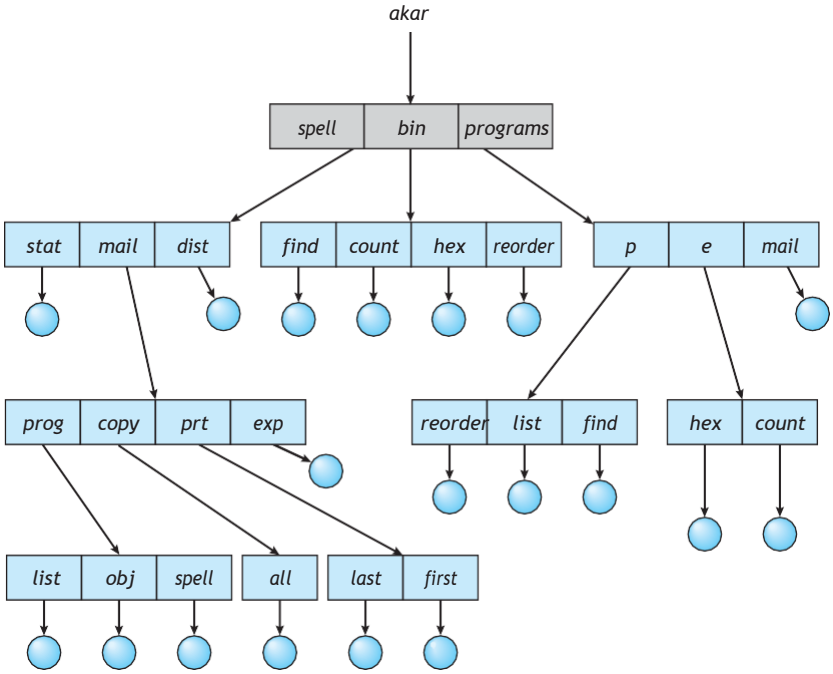
membutuhkan 5 MB , maka mendukung 12 pengguna akan membutuhkan 5 × 12 = 60 MB saja salinan dari sistem file.)

Solusi standarnya adalah dengan sedikit mempersulit prosedur pencarian. A direktori pengguna khusus didefinisikan untuk berisi file sistem (misalnya, user 0). Setiap kali nama file diberikan untuk dimuat, sistem operasi terlebih dahulu mencari UFD lokal . Jika file ditemukan, maka file tersebut akan digunakan. Jika tidak ditemukan, sistem secara otomatis pencarian itu spesial pengguna direktori itu mengandung itu sistem file. Urutan pencarian direktori ketika sebuah file diberi nama disebut **pencarian jalur** . Jalur pencarian dapat diperluas untuk memuat daftar direktori yang tidak terbatas untuk mencari ketika nama perintah diberikan. Cara ini merupakan cara yang paling banyak digunakan di UNIX dan Windows. Sistem juga dapat dirancang sedemikian rupa sehingga setiap pengguna memiliki miliknya sendirimemiliki mencari jalur.

13.3.3 Terstruktur Pohon Direktori

Sekali Kami telah melihat caranya ke lihat a direktori dua tingkat sebagai A pohon dua tingkat, generalisasi alaminya adalah memperluas struktur direktori ke pohon ketinggian sewenang-wenang (Gambar 13.9). Generalisasi ini memungkinkan pengguna untuk membuat sendiri memiliki subdirektori dan mengatur file-filenya sesuai dengan itu. Sebatang pohon adalah yang paling banyak umum direktori struktur. Itu pohon memiliki A akar direktori, Dan setiap mengajukan di dalam itu sistem memiliki keunikan jalur nama.

Direktori (atau subdirektori) berisi sekumpulan file atau subdirektori. Di dalam banyak implementasi, A direktori adalah secara sederhana lain mengajukan, Tetapi dia adalah diperlakukan di dalam cara khusus. Semua direktori memiliki format internal yang sama. Masing-masing satu bit direktori pintu masuk mendefinisikan itu pintu masuk sebagai A mengajukan (0) atau sebagai A subdirektori (1). Spesial



Angka 13.9 Terstruktur pohon direktori struktur.

sistem panggilan adalah digunakan ke membuat Dan menghapus direktori. Di dalam ini kasus itu Pengoperasian sistem (atau kode sistem file) mengimplementasikan format file lain, yaitu a direktori.

Di dalam normal menggunakan, setiap proses memiliki A saat ini direktori. Itu **saat ini direktori** harus berisi sebagian besar file yang saat ini menarik untuk proses tersebut. Kapan referensi dibuat ke file, direktori saat ini dicari. Jika file diperlukan yang tidak ada dalam direktori saat ini, maka pengguna biasanya harus menentukan a jalur nama atau mengubah itu saat ini direktori ke menjadi itu direktori memegang itu mengajukan. Untuk mengubah direktori, panggilan sistem dapat disediakan yang mengambil direktori nama sebagai A parameter Dan kegunaan dia ke mendefinisikan ulang itu saat ini direktori. Dengan demikian, itu pengguna dapat mengubah direktori saat ini kapan pun dia mau. Sistem lain meninggalkannya ke aplikasi (misalnya, shell) untuk melacak dan beroperasi pada direktori saat ini, seperti setiap proses bisa memiliki berbeda saat ini direktori.

Direktori awal shell login pengguna saat ini ditetapkan ketika pekerjaan pengguna dimulai atau pengguna login. Sistem operasi mencari akuntansi file (atau lokasi lain yang telah ditentukan sebelumnya) untuk menemukan entri untuk pengguna ini (untuk tujuan akuntansi). Dalam file akuntansi ada penunjuk ke (atau nama) direktori awal pengguna. Pointer ini disalin ke variabel lokal untuk pengguna ini itu menentukan itu milik pengguna awal saat ini direktori. Dari itu kerang, lainnya proses Bisa menjadi melahirkan. Itu saat ini direktori dari setiap subproses adalah biasanya itu saat ini direktori dari orang tua Kapan dia dulu melahirkan.

Jalur nama Bisa menjadi dari dua jenis: mutlak Dan relatif. Di dalam UNIX Dan Linux, nama **jalur absolut** dimulai dari akar (yang ditandai dengan inisial " / ") Dan ikuti aku A jalur turun ke itu spesifikasi c jika saya mengedit jika aku pergi, memberi itu direktori ayolah itu jalur. A **relatif jalur nama** mendefinisikan A jalur dari itu saat ini direktori. Untuk Misalnya, dalam sistem file terstruktur pohon pada Gambar 13.9, jika direktori saat ini adalah `/spell/mail` , maka nama jalur relatif `prt/first` merujuk ke file yang sama sebagai melakukan itu mutlak jalur nama `/mantra/mail/prt/pertama` .

Mengizinkan A pengguna ke mendefinisikan dia memiliki subdirektori izin dia ke memaksakan struktur pada filenya. Struktur ini mungkin menghasilkan direktori terpisah untuk file yang terkait dengan topik berbeda (misalnya, subdirektori telah dibuat untuk menampung teks buku ini) atau berbagai bentuk informasi. Misalnya, `program` direktori mungkin berisi program sumber; direktori `bin` mungkin menyimpan semua itu biner. (Sebagai A samping catatan, dapat dieksekusi file adalah dikenal di banyak orang sistem sebagai " biner " yang dipimpin ke mereka makhluk disimpan di dalam itu direktori `bin` .)

Sebuah menarik kebijakan keputusan di dalam A terstruktur pohon direktori kekhawatiran Bagaimana untuk menangani penghapusan direktori. Jika suatu direktori kosong, entri tersebut ada di direktori yang memuatnya dapat dengan mudah dihapus. Namun, misalkan arahan- cerita ke menjadi dihapus adalah bukan kosong Tetapi mengandung beberapa file atau subdirektori. Satu dari dua pendekatan Bisa menjadi diambil. Beberapa sistem akan bukan menghapus A direktori kecuali dia adalah kosong. Dengan demikian, ke menghapus A direktori, itu pengguna harus Pertama menghapus semua itu file di direktori itu. Jika ada subdirektori, prosedur ini harus diterapkan secara rekursif kepada mereka, sehingga mereka juga dapat dihapus. Pendekatan ini dapat membuahkan hasil dalam jumlah pekerjaan yang besar. Pendekatan alternatif, seperti yang dilakukan oleh UNIX `rm` memerintah, adalah memberikan pilihan: kapan a permintaan

dibuat untuk menghapus direktori, semua file dan subdirektori direktori tersebut juga harus dihapus dihapus. Pendekatan mana pun cukup mudah diterapkan; pilihannya adalah salah satu kebijakan. Itu yang terakhir kebijakan adalah lagi nyaman, Tetapi dia adalah Juga lagi berbahaya, Karena sebuah seluruh direktori struktur Bisa menjadi DIHAPUS dengan satu memerintah. Jika itu memerintah

dikeluarkan karena kesalahan, sejumlah besar file dan direktori perlu dipulihkan (asumsi cadangan ada).

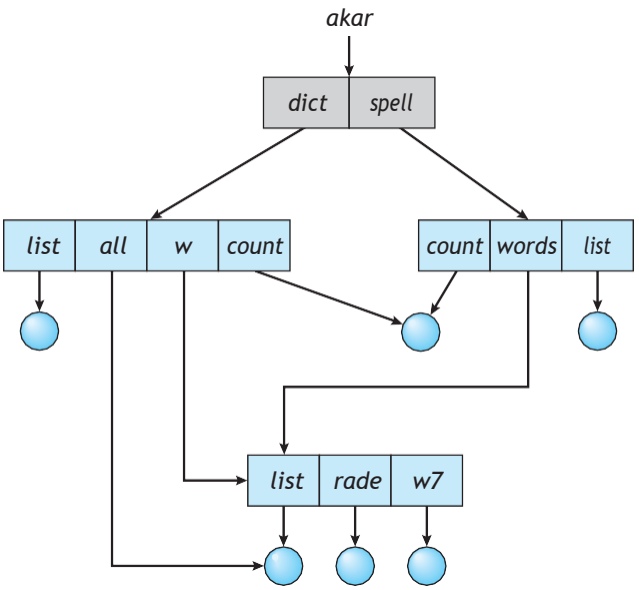
Dengan A terstruktur pohon direktori sistem, pengguna Bisa menjadi diizinkan ke mengakses, di dalam tambahan ke milik mereka file, itu file dari lainnya pengguna. Untuk contoh, pengguna B Bisa mengakses A mengajukan dari pengguna A oleh menentukan -nya jalur nama. Pengguna B Bisa menentukan salah satu sebuah mutlak atau A relatif jalur nama. Kalau tidak, pengguna B Bisa mengubah dia saat ini direktori ke menjadi pengguna Sebagai direktori Dan mengakses itu mengajukan oleh -nya mengajukan nama.

13.3.4 Grafik Asiklik Direktori

Mempertimbangkan dua pemrogram WHO adalah bekerja pada A persendian proyek. Itu file juga- terkait dengan proyek itu dapat disimpan dalam subdirektori, memisahkannya dari proyek dan file lain dari kedua pemrogram. Tapi karena keduanya programmer adalah sama bertanggung jawab untuk itu proyek, keduanya ingin itu subdirektori ke menjadi di dalam milik mereka memiliki direktori. Di dalam ini situasi, itu umum subdirektori sebaiknya menjadi ***bersama.*** Direktori atau file bersama ada dalam sistem file di dua (atau lebih) tempat di sekali.

Struktur pohon melarang berbagi file atau direktori. Sebuah **asiklik graph** — yaitu, grafik tanpa siklus — memungkinkan direktori untuk berbagi subdirektori cerita dan file (Gambar 13.10). File atau subdirektori yang sama mungkin ada dua direktori yang berbeda. Grafik asiklik adalah generalisasi alami dari grafik pohon- tersusun direktori skema.

Penting untuk dicatat bahwa file (atau direktori) bersama tidak sama dengan dua salinan dari itu mengajukan. Dengan dua salinan, setiap programmer Bisa melihat itu menyalin lebih tepatnya dibandingkan itu asli, Tetapi jika satu programmer perubahan itu mengajukan, itu perubahan akan bukan muncul di dalam itu yang lain menyalin. Dengan A bersama mengajukan, hanya satu sebenarnya mengajukan ada, Jadi setiap perubahan dibuat oleh satu orang adalah langsung bisa dilihat ke itu lainnya. Membagikan adalah



Angka 13.10 Grafik asiklik direktori struktur.

khususnya penting untuk subdirektori; A baru mengajukan dibuat oleh satu orang akansecara otomatis muncul di dalam semua itu bersama subdirektori.

Kapan rakyat adalah bekerja sebagai A tim, semua itu file mereka ingin ke membagikan Bisa menjadi dimasukkan ke dalam satu direktori. Direktori home setiap anggota tim dapat berisi ini direktori dari bersama file sebagai A subdirektori. Bahkan di dalam itu kasus dari A lajang pengguna, organisasi file pengguna mungkin mengharuskan beberapa file ditempatkan di tempat yang berbeda subdirektori. Untuk contoh, A program tertulis untuk A tertentu proyek sebaiknya menjadi keduanya di dalam itu direktori dari semua program Dan di dalam itu direktori untuk itu proyek.

File dan subdirektori bersama dapat diimplementasikan dalam beberapa cara. A umum jalan, dicontohkan oleh UNIX sistem, adalah ke membuat A baru direktori pintu masuk ditelepon A tautan. A **tautan** adalah secara efektif A penunjuk ke lain mengajukan atau subdirektori. Untuk Misalnya, tautan dapat diimplementasikan sebagai nama jalur absolut atau relatif. Ketika referensi ke suatu file dibuat, kami mencari direktorinya. Jika direktori entri ditandai sebagai tautan, lalu nama file sebenarnya dicantumkan dalam tautan tersebut informasi. Kami **menyelesaikan** tautan tersebut dengan menggunakan nama jalur itu untuk menemukan yang asli mengajukan. Tautan mudah diidentifikasi berdasarkan formatnya di entri direktori (atau dengan memiliki tipe khusus pada sistem yang mendukung tipe) dan secara efektif tidak langsung petunjuk. Sistem operasi mengabaikan link ini ketika melintasi direktori pohon ke melestarikan itu struktur asiklik dari itu sistem.

Pendekatan umum lainnya untuk mengimplementasikan file bersama adalah dengan melakukan duplikat semua informasi tentang mereka di kedua direktori berbagi. Jadi, keduanya entri adalah identik Dan setara. Mempertimbangkan itu perbedaan di antara ini mendekati dan pembuatan tautan. Tautannya jelas berbeda dengan direktori aslinya pintu masuk; dengan demikian, itu dua adalah bukan setara. Duplikat direktori entri, Namun, membuat yang asli dan salinannya tidak dapat dibedakan. Masalah besar dengan duplikat direktori entri adalah mempertahankan konsistensi kapan A mengajukan adalah diubah.

Struktur direktori grafik asiklik lebih fleksibel dibandingkan pohon sederhana strukturnya, tetapi juga lebih kompleks. Beberapa masalah harus dipertimbangkan dengan hati-hati. Sebuah file sekarang mungkin memiliki beberapa nama jalur absolut. Akibatnya, nama file yang berbeda mungkin merujuk ke file yang sama. Situasi ini mirip dengan masalah aliasing untuk bahasa pemrograman. Jika kita mencoba melintasi seluruh mengajukan sistem- ke menemukan A mengajukan, ke mengumpulkan statistik pada semua file, atau ke menyalin semua file ke penyimpanan cadangan — masalah ini menjadi signifikan, karena kami tidak melakukannya ingin melintasi bersama struktur lagi dari sekali.

Masalah lainnya melibatkan penghapusan. Kapan ruang dapat dialokasikan untuk a file yang dibagikan dibatalkan alokasinya dan digunakan kembali? Salah satu kemungkinannya adalah menghapus file tersebut kapan pun siapa pun menghapus dia, Tetapi ini tindakan mungkin meninggalkan teruntai petunjuk ke itu file yang sekarang tidak ada. Lebih buruk lagi, jika sisa penunjuk file berisi disk sebenarnya alamat, Dan itu ruang angkasa adalah kemudian digunakan kembali untuk lainnya file, ini teruntaipetunjuk mungkin menunjuk ke dalam tengah dari lainnya file.

Di dalam A sistem Di mana membagikan adalah dilaksanakan oleh simbolis tautan, ini situasi adalah agak lebih mudah ke menangani. Itu penghapusan dari A tautan membutuhkan bukan memengaruhi itu asli mengajukan; hanya itu tautan adalah DIHAPUS. Jika itu mengajukan

pintu masuk diri adalah dihapus, itu ruang angkasa untuk file tersebut dibatalkan alokasinya, sehingga tautannya menggantung. Kita dapat mencari tautan ini dan menghapusnya juga, kecuali jika daftar tautan terkait tetap disimpan setiap file, pencarian ini bisa mahal. Alternatifnya, kita dapat meninggalkan tautannya sampai ada upaya untuk menggunakannya. Pada saat itu, kita dapat menentukan bahwa file dengan nama yang diberikan oleh link tidak ada dan bisa gagal menyelesaikannya tautan nama; itu mengakses adalah diperlakukan hanya sebagai dengan setiap lainnya liar mengajukan nama. (Di dalam ini kasus, itu sistem perancang sebaiknya mempertimbangkan dengan hati-hati Apa ke Mengerjakan Kapan A mengajukan adalah

dihapus Dan lain mengajukan dari itu sama nama adalah dibuat, sebelum A simbolis tautan ke itu asli mengajukan adalah digunakan.) Di dalam itu kasus dari UNIX , simbolis tautan adalah kiri Kapan A mengajukan adalah dihapus, dan itu adalah ke atas ke itu pengguna ke menyadari itu asli mengajukan adalah hilang atau memiliki pernah diganti. Microsoft penggunaan Windows itu sama mendekati.

Pendekatan lain untuk menghapus adalah dengan menyimpan file sampai semua referensinya itu dihapus. Untuk menerapkan pendekatan ini, kita harus memiliki mekanisme tertentu untuk menentukan bahwa referensi terakhir ke file tersebut telah dihapus. Kita bisa menyimpan A daftar dari semua referensi ke A mengajukan (direktori entri atau simbolis tautan). Kapan tautan atau salinan entri direktori dibuat, entri baru ditambahkan daftar referensi file. Ketika sebuah tautan atau entri direktori dihapus, kami menghapusnya pintu masuk pada itu daftar. Itu mengajukan adalah dihapus Kapan -nya referensi file daftar adalah kosong.

Itu masalah dengan ini mendekati adalah itu variabel Dan berpotensi besar ukuran dari itu referensi file daftar. Namun, Kami Sungguh Mengerjakan bukan membutuhkan ke menyimpan itu seluruh daftar — kita hanya perlu menghitung jumlah referensi. Menambahkan yang baru tautan atau direktori pintu masuk peningkatan itu referensi menghitung. Menghapus A tautan atau pintu masuk penurunan itu menghitung. Kapan itu menghitung adalah 0, itu mengajukan Bisa menjadi dihapus; di sana adalah TIDAK referensi yang tersisa untuk itu. Sistem operasi UNIX menggunakan pendekatan ini untuk tautan nonsimbolis (atau **tautan keras**), menjaga jumlah referensi dalam informasi file blok mation (atau inode; lihat Bagian C.7.2). Dengan secara efektif melarang banyak hal referensi ke direktori, Kami menjaga sebuah grafik asiklik struktur.

Untuk menghindari masalah seperti yang baru saja dibahas, beberapa sistem sederhana Mengerjakan tidak mengizinkan bersama direktori atau tautan.

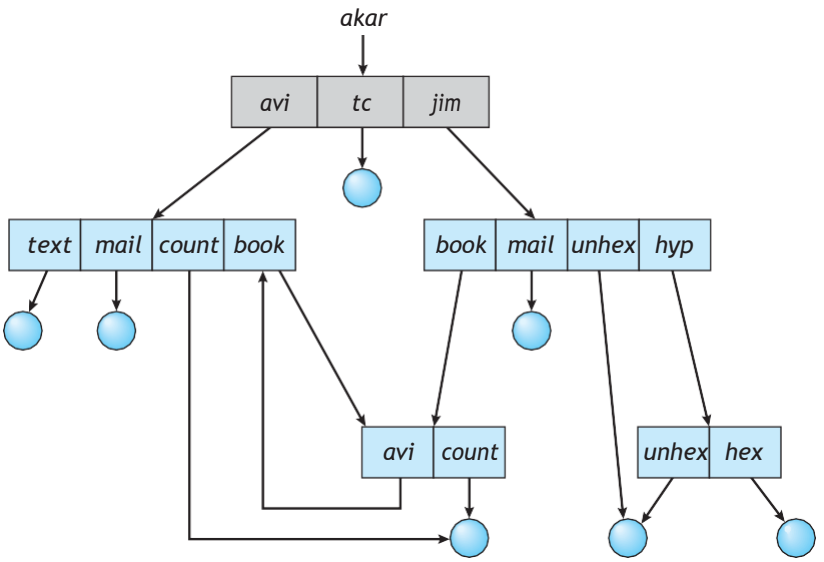
13.3.5 Umum Grafik Direktori

Masalah serius dalam menggunakan struktur grafik asiklik adalah memastikan hal itu tidak ada siklus. Jika kita memulai dengan direktori dua tingkat dan mengizinkan pengguna untuk membuatnya subdirektori, hasil direktori terstruktur pohon. Seharusnya cukup mudah untuk dilihat yang hanya menambahkan file dan subdirektori baru ke struktur pohon yang sudah ada direktori mempertahankan sifat terstruktur pohon. Namun, saat kami menambahkan tautan, struktur pohon dihancurkan, menghasilkan struktur grafik sederhana (Gambar 13.11).

Itu utama keuntungan dari sebuah asiklik grafik adalah itu relatif kesederhanaan dari itualgoritma ke melintasi itu grafik Dan ke menentukan Kapan di sana adalah TIDAK lagireferensi ke A mengajukan. Kami ingin ke menghindari melintasi bersama bagian dari sebuah asiklik grafik dua kali, terutama untuk pertunjukan alasan. Jika Kami memiliki hanya dicari A besarbersama subdirektori untuk A tertentu mengajukan tanpa temuan dia, Kami ingin ke menghindarimencari itu subdirektori lagi; itu Kedua mencari akan menjadi A limbah dari waktu.Jika siklus diperbolehkan untuk ada di direktori, kami juga menginginkannya ke menghindari mencari setiap komponen dua kali, untuk alasan dari ketepatan sebagai Sehat sebagai melakukan-mance. Sayangnya dirancang algoritma mungkin hasil di dalam sebuah tak terbatas lingkaran terus menerus mencari melalui itu siklus Dan tidak pernah mengakhiri. Satu larutan

adalah ke membatasi sewenang-wenang itu nomor dari direktori itu akan menjadi diakses selama A mencari.

Masalah serupa terjadi ketika kami mencoba menentukan kapan file dapat dihapus. Dengan struktur direktori grafik asiklik, nilai 0 di referensi menghitung cara itu di sana adalah TIDAK lagi referensi ke itu mengajukan atau direktori, dan file tersebut dapat dihapus. Namun, jika ada siklus, referensinya dihitung mungkin bukan 0 meskipun tidak mungkin lagi merujuk ke direktori atau file. Ini anomali hasil dari kemungkinannya dari referensi diri (atau a siklus) di dalam itu direktori struktur. Di dalam ini kasus, Kami umumnya membutuhkan ke menggunakan A **sampah koleksi**



Angka 13.11 Umum grafik direktori.

skema ke menentukan Kapan itu terakhir referensi memiliki pernah dihapus Dan itu diskruang angkasa Bisa menjadi dialokasikan kembali. Sampah koleksi melibatkan melintasi itu seluruh mengajukansistem, menandai semuanya itu Bisa menjadi diakses. Kemudian, A Kedua lulus mengumpulkansemuanya itu adalah bukan ditandai ke A daftar dari bebas ruang angkasa. (A serupa menandaiprosedur Bisa menjadi digunakan ke memastikan itu A penjelajahan atau mencari akan menutupi semuanyadi dalam itu mengajukan sistem sekali Dan hanya sekali.) Sampah koleksi untuk A berbasis disk mengajukansistem, Namun, adalah sangat waktu memakan Dan adalah dengan demikian jarang mencoba.Sampah koleksi adalah diperlukan hanya Karena dari mungkin siklus di dalam itu grafik. Dengan demikian, struktur grafik asiklik lebih mudah untuk dikerjakan. Kesulitannya adalah untuk menghindari siklus sebagai baru tautan adalah ditambahkan ke itu struktur. Bagaimana Mengerjakan Kami tahu Kapan A baru tautan akan menyelesaikan A siklus? Di sana adalah algoritma ke mendeteksi siklus di dalam grafik; Namun, mereka adalah secara komputasi mahal, khususnya Kapan itu grafik adalah pada disk penyimpanan. A lebih sederhana algoritma di dalam itu spesial kasus dari direktori Dan tautan adalah untuk melewati tautan selama traversal direktori. Siklus dihindari, dan tidak ada tambahanatas terjadi.

13.4 Perlindungan

Ketika informasi disimpan dalam sistem komputer, kami ingin menyimpannya dengan aman dari kerusakan fisik (masalah keandalan) dan akses yang tidak tepat (masalahdari perlindungan).

Keandalan umumnya diberikan melalui duplikat salinan file. Banyak komputer- ers memiliki sistem program itu secara otomatis (atau melalui operator komputer intervensi) menyalin disk file ke tape pada reguler interval (sekali per hari atau pekan atau bulan) untuk menyimpan salinan jika sistem file rusak secara tidak sengaja. Mengajukan sistem Bisa menjadi rusak oleh perangkat keras masalah (seperti sebagai kesalahan di dalam membaca

atau menulis), kekuatan melonjak atau kegagalan, kepala mogok, kotoran, suhu ekstrem,

Dan vandalisme. File mungkin menjadi dihapus secara tidak sengaja. Bug di dalam itu berkas sistem lembut- ware juga dapat menyebabkan isi file hilang. Keandalan dibahas lebih lanjut detail di dalam Bab 11.

Perlindungan Bisa menjadi asalkan di dalam banyak cara. Untuk A laptop sistem berlari sistem operasi modern, kami mungkin memberikan perlindungan dengan mewajibkan pengguna nama Dan kata sandi autentikasi ke mengakses dia, mengenkripsi itu sekunder penyimpanan- usia sehingga bahkan seseorang yang membuka laptop dan melepas drive pun akan melakukannya A sulit waktu mengakses -nya data, Dan firewall jaringan mengakses Jadi itu Kapan sedang digunakan sulit untuk dibobol melalui koneksi jaringannya. Dalam multipengguna sistem, bahkan akses sistem yang valid memerlukan mekanisme yang lebih maju untuk melakukannya mengizinkan hanya sah akses dari itu data.

13.4.1 Jenis Akses

Kebutuhan untuk melindungi file adalah akibat langsung dari kemampuan mengakses file. Sistem itu Mengerjakan bukan izin mengakses ke itu file dari lainnya pengguna Mengerjakan bukan membutuhkan perlindungan. Dengan demikian, Kami bisa menyediakan menyelesaikan perlindungan oleh melarang mengakses. Kalau tidak, kita bisa menyediakan bebas mengakses dengan TIDAK perlindungan. Keduanya pendekatan adalah juga ekstrim untuk umum menggunakan. Apa adalah diperlukan adalah dikendalikan mengakses.

Mekanisme perlindungan memberikan akses terkendali dengan membatasi jenis akses file yang dapat dilakukan. Akses diizinkan atau ditolak tergantung pada beberapa faktor, salah satunya adalah jenis akses yang diminta. Beberapa perbedaan jenis operasi mungkin menjadi dikendalikan:

- **Membaca** . Membaca dari itu mengajukan.
- **Menulis** . Menulis atau menulis kembali itu mengajukan.
- **Jalankan** . Memuat itu mengajukan ke dalam Penyimpanan Dan menjalankan dia.
- **Menambahkan** . Menulis baru informasi pada itu akhir dari itu mengajukan.
- **Menghapus** . Menghapus itu mengajukan Dan bebas -nya ruang angkasa untuk mungkin penggunaan kembali.
- **Daftar** . Daftar itu nama Dan atribut dari itu mengajukan.
- **Atribut mengubah** . Berubah itu atribut dari itu mengajukan.

Operasi lain, seperti mengganti nama, menyalin, dan mengedit file, juga dapat dilakukan dikendalikan. Namun, bagi banyak sistem, fungsi tingkat yang lebih tinggi ini mungkin saja terjadi diimplementasikan oleh program sistem yang membuat panggilan sistem tingkat rendah. Perlindungan adalah asalkan pada hanya itu lebih rendah tingkat. Untuk contoh, penyalinan A mengajukan mungkin menjadi dilaksanakan secara sederhana oleh A urutan dari membaca permintaan. Di dalam ini kasus, A pengguna dengan membaca mengakses Bisa Juga menyebabkan itu mengajukan ke menjadi disalin, dicetak, Dan Jadi pada.

Banyak mekanisme perlindungan telah diusulkan. Masing-masing

memiliki kelebihan dan kekurangannya serta harus sesuai dengan tujuan penerapannya. A sistem komputer kecil yang hanya digunakan oleh beberapa anggota kelompok penelitian, misalnya, mungkin tidak memerlukan jenis perlindungan yang sama seperti perusahaan besar komputer yang digunakan untuk penelitian, keuangan, dan operasi personalia. Kami membahas beberapa pendekatan ke perlindungan di dalam itu mengikuti bagian Dan hadiah A lagi menyelesaikan perlakuan di dalam Bab 17.

13.4.2 Mengakses Kontrol

Pendekatan yang paling umum terhadap masalah perlindungan adalah membuat akses bergantung pada penyok pada identitas pengguna. Pengguna yang berbeda mungkin memerlukan jenis yang berbeda akses ke file atau direktori. Skema paling umum untuk mengimplementasikan identitas-bergantung mengakses adalah ke rekan dengan setiap mengajukan Dan direktori sebuah **kontrol akses daftar** (**ACL**) menentukan pengguna nama Dan itu jenis dari mengakses diizinkan untuk setiap pengguna. Ketika pengguna meminta akses ke file tertentu, sistem operasi akan memeriksanya daftar akses yang terkait dengan file itu. Jika pengguna itu terdaftar untuk yang diminta mengakses, itu mengakses adalah diizinkan. Jika tidak, A perlindungan pelanggaran terjadi, Dan itu pengguna pekerjaan adalah ditolak mengakses ke mengajukan.

Pendekatan ini memiliki keuntungan dalam memungkinkan metodologi akses yang kompleks ya. Masalah utama dengan daftar akses adalah panjangnya. Jika kita ingin mengizinkan setiap orang untuk membaca file, kita harus mencantumkan semua pengguna dengan akses baca. Teknik ini memiliki dua yang tidak diinginkan konsekuensi:

- Membangun seperti A daftar mungkin menjadi A membosankan Dan tidak bermanfaat tugas, khususnyajika Kami Mengerjakan bukan tahu di dalam maju itu daftar pengguna di dalam itu sistem.
- Itu direktori pintu masuk, sebelumnya dari tetap ukuran, Sekarang harus menjadi dari variabel ukuran,dihasilkan di dalam lagi rumit ruang angkasa pengelolaan.

Masalah ini dapat diatasi dengan menggunakan versi akses yang ringkas daftar.

Untuk menyingkat panjang daftar kontrol akses, banyak sistem yang mengenalinya tiga klasifikasi dari pengguna di dalam hubungan dengan setiap mengajukan:

- **Pemilik** . Itu pengguna WHO dibuat itu mengajukan adalah itu pemilik.
- **Kelompok** . Sekumpulan pengguna yang berbagi file dan memerlukan akses serupa adalah akelompok, atau kelompok kerja.
- **Lainnya** . Semua lainnya pengguna di dalam itu sistem.

Pendekatan terbaru yang paling umum adalah menggabungkan daftar kontrol akses dengan itu lagi umum (Dan lebih mudah ke melaksanakan) pemilik, kelompok, Dan semesta mengakses- kontrol skema hanya dijelaskan. Untuk contoh, Solaris kegunaan itu tiga kategori dari akses secara default tetapi memungkinkan daftar kontrol akses ditambahkan ke file tertentu dan direktori Kapan lagi berbutir halus mengakses kontrol adalah diinginkan.

Sebagai ilustrasi, perhatikan seseorang, Sara, yang sedang menulis buku baru. Dia memiliki mempekerjakan tiga mahasiswa pascasarjana (Jim, Dawn, dan Jill) untuk membantu proyek tersebut. Itu teks dari itu buku adalah disimpan di dalam A mengajukan bernama `buku.tex`. Itu perlindungan terkaitdengan ini mengajukan adalah sebagai berikut:

- Sara sebaiknya menjadi mampu ke memohon semua operasi pada itu mengajukan.
- Jim, Dawn, dan Jill seharusnya hanya bisa membaca dan menulis file;

mereka sebaiknya tidak menjadi diizinkan ke menghapus itu mengajukan.

- Semua pengguna lain harus dapat membaca, tetapi tidak dapat menulis, file tersebut. (Sara adalah tertarik untuk membiarkan sebanyak mungkin orang membaca teks tersebut sehingga dia Bisa mendapatkan umpan balik.)

IZIN DI DALAM A UNIX SISTEM

Dalam sistem UNIX , proteksi direktori dan proteksi file ditangani demikian pula. Terkait dengan setiap file dan direktori ada tiga bidang— pemilik, grup, Dan semesta- setiap terdiri dari itu tiga bit rwx , Di mana R kontrol akses **baca** , w mengontrol akses **tulis** , dan x mengontrol **eksekusi** . Jadi, pengguna bisa daftar itu isi dari A subdirektori hanya jika itu R sedikit adalah mengatur di dalam itu sesuai bidang. Demikian pula, A pengguna Bisa mengubah miliknya saat ini direktori ke lain saat ini direktori (katakanlah, foo) hanya jika bit x yang terkait dengan subdirektori foo diatur di sesuai bidang.

A direktori sampel daftar dari A UNIX lingkungan adalah ditampilkan di bawah ini:

```
-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 jwg   student     512   Aug 3 14:13    student-proj/
-rw-r--r--   1 pbg   staff      9423   Feb 24 2017    program.c
-rwxr-xr-x   1 pbg   staff     20471   Feb 24 2017    program
drwx--x--x   4 tag   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/
```

Itu bidang pertama menjelaskan itu perlindungan dari itu mengajukan atau direktori. A D sebagai itu Pertama karakter menunjukkan A subdirektori. Juga ditampilkan adalah itu nomor dari tautan ke itu mengajukan, itu milik pemilik nama, itu grup nama, itu ukuran dari itu mengajukan di dalam byte, itu tanggal dari terakhir modifikasi, Dan Akhirnya itu file nama (dengan opsional perpanjangan).

Untuk mencapai perlindungan tersebut, kita harus membuat grup baru — misalnya, teks — dengan anggota jim, Fajar, Dan jill. Itu nama dari itu kelompok, teks, harus Kemudian dikaitkan dengan file book.tex, dan hak akses harus diatur sesuai dengan itu kebijakan Kami memiliki diuraikan.

Sekarang mempertimbangkan A pengunjung ke yang Sara akan menyukai ke menganugerahkan sementara mengakses ke Bab 1. Itu pengunjung tidak bisa menjadi ditambahkan ke itu teks kelompok Karena itu akan memberi dia mengakses ke semua bab. Karena A mengajukan Bisa menjadi di dalam hanya satu kelompok, Sara tidak dapat menambahkan grup lain ke Bab 1. Dengan tambahan kontrol akses- fungsionalitas daftar, pengunjung dapat ditambahkan ke daftar kontrol akses Bab 1.

Agar skema ini berfungsi dengan baik, izin dan daftar akses harus dikonfigurasi. dikendalikan rapat. Ini kontrol Bisa menjadi ahli di dalam beberapa cara. Untuk contoh, dalam sistem UNIX , grup hanya dapat dibuat dan dimodifikasi oleh pengelola dari itu fasilitas (atau oleh setiap pengguna super). Dengan demikian, kontrol adalah dicapai melalui manusia interaksi. Mengakses daftar adalah dibahas lebih jauh di dalam Bagian 17.6.2.

Dengan itu lagi terbatas perlindungan klasifikasi, hanya tiga bidang adalah diperlukan ke mendefinisikan perlindungan. Sering, setiap bidang adalah A koleksi dari bit, Dan setiap sedikit salah satu memungkinkan atau mencegah

itu mengakses terkait dengan dia. Untuk contoh, itu UNIX sistem mendefinisikan tiga bidang yang masing-masing terdiri dari tiga bit — rwx , di mana r mengontrol akses baca, w kontrol menulis akses, dan x mengontrol eksekusi. A memisahkan bidang adalah disimpan untuk itu

mengajukan pemilik, untuk itu file kelompok, Dan untuk semua lainnya pengguna. Di dalam ini skema, sembilan bit per file diperlukan untuk mencatat informasi perlindungan. Jadi, sebagai contoh kita, perlindungan bidang untuk itu mengajukan buku.tex adalah sebagai berikut: untuk itu pemilik Sara, semua bit adalah mengatur; untuk itu kelompok teks , itu R Dan w bit adalah mengatur; Dan untuk itu semesta, hanya itu R sedikit adalah mengatur.

Satu kesulitan di dalam menggabungkan pendekatan datang di dalam itu pengguna antarmuka. Pengguna harus dapat mengetahui kapan izin ACL opsional ditetapkan pada file. Dalam Solaris contoh, A " + " adalah ditambahkan ke itu reguler izin, sebagai di dalam:

```
19 -rw-r--r--+  1  jim  staf  130 Mungkin 25 22:13 file1
```

A memisahkan mengatur dari perintah, setfacl Dan getfacl , adalah digunakan ke mengelola itu ACL s.

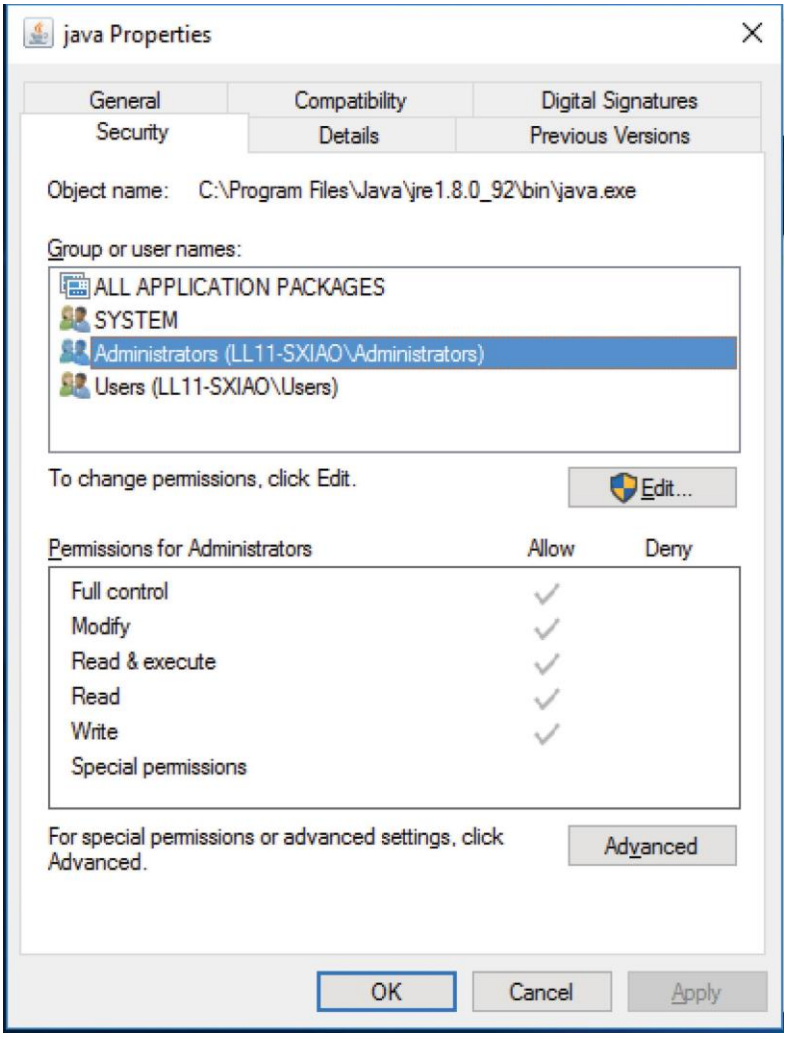
jendela pengguna khas mengelola kontrol akses daftar melalui itu GUI . Angka 13.12 menunjukkan A izin file jendela pada jendela 7 NTFS mengajukan sistem. Di dalam inicontoh, pengguna " tamu " adalah secara khusus ditolak mengakses ke itu mengajukan DaftarPanel.java .Lain kesulitan adalah menugaskan hak lebih tinggi Kapan izin Dan ACL s konflik. Untuk contoh, jika Walter adalah di dalam A file kelompok, yang memiliki membaca izin,Tetapi itu mengajukan memiliki sebuah ACL pemberian Walter membaca Dan menulis izin, sebaiknya Amenulis oleh Walter menjadi diberikan atau ditolak? Solaris Dan lainnya Pengoperasian sistem memberikan prioritas pada ACL (seperti mereka adalah lagi berbutir halus Dan adalah tidak ditugaskan oleh bawaan). Ini berikut itu umum aturan itu kekhususan sebaiknya memiliki prioritas.

13.4.3 Lainnya Perlindungan Pendekatan

Pendekatan lain terhadap masalah proteksi adalah dengan mengasosiasikan kata sandi dengan setiap file. Sama seperti akses ke sistem komputer yang sering dikendalikan oleh sebuah pass- Word, akses ke setiap file dapat dikontrol dengan cara yang sama. Jika kata sandi dipilih secara acak dan sering diubah, skema ini mungkin efektif dalam membatasi iting mengakses ke A mengajukan. Itu menggunakan dari kata sandi memiliki A sedikit kekurangan, Namun. Pertama, mungkin ada jumlah kata sandi yang perlu diingat pengguna besar, membuat itu skema tidak praktis. Kedua, jika hanya satu kata sandi adalah digunakan untuk semua file, kemudian setelah ditemukan, semua file dapat diakses; perlindungan aktif dasar semua atau tidak sama sekali. Beberapa sistem mengizinkan pengguna untuk mengasosiasikan kata sandi dengan subdirektori, bukan dengan file individual, untuk mengatasi masalah ini. Lebih umum lagi enkripsi partisi atau file individual memberikan kekuatan perlindungan, Tetapi kata sandi pengelolaan adalah kunci.

Dalam struktur direktori bertingkat, kita perlu melindungi tidak hanya individu file Tetapi Juga koleksi dari file di dalam subdirektori; itu adalah, Kami membutuhkan ke menyediakan mekanisme untuk perlindungan direktori. Operasi direktori yang harus dilakukan terlindung adalah agak berbeda dari itu mengajukan operasi. Kami ingin ke kontrol itu penciptaan Dan penghapusan dari file di dalam A direktori. Di dalam tambahan, Kami

mungkin ingin untuk mengontrol apakah pengguna dapat menentukan keberadaan file dalam direktori. Terkadang, pengetahuan tentang keberadaan dan nama file sangat penting diri. Oleh karena itu, membuat daftar isi direktori harus merupakan operasi yang dilindungi. Demikian pula, jika A jalur nama merujuk ke A mengajukan di dalam A direktori, itu pengguna harus menjadi diizinkan akses ke direktori dan file. Dalam sistem di mana file mungkin ada berbagai nama jalur (seperti grafik asiklik dan umum), mungkin dapat dilakukan oleh pengguna tertentu mempunyai hak akses yang berbeda terhadap suatu file tertentu, tergantung pada nama pathnya digunakan.



Angka 13.12 Windows 10 kontrol akses daftar pengelolaan.

13.5 Dipetakan Memori File

Di sana adalah satu lainnya metode dari mengakses file, Dan dia adalah sangat umumnya digunakan. Mempertimbangkan A sekuensial membaca dari A mengajukan pada disk menggunakan itu standar sistem panggilan buka() , baca() , dan tulis() . Setiap akses file memerlukan panggilan sistem dan disk mengakses. Alternatifnya, kita dapat menggunakan teknik memori virtual yang dibahas Bab 10 ke merawat mengajukan masukan/keluaran sebagai rutin Penyimpanan mengakses. Ini mendekati, diketahui sebagai **Penyimpanan pemetaan** A mengajukan, memungkinkan A bagian dari itu maya alamat ruang angkasa ke menjadi secara logis terkait dengan file tersebut. Seperti yang akan kita lihat, hal ini dapat membawa dampak yang signifikan pertunjukan meningkat.

13.5.1 Dasar Mekanisme

Penyimpanan pemetaan A mengajukan adalah ahli oleh pemetaan A disk memblokir ke A halaman (atau halaman) di dalam Penyimpanan. Awal mengakses ke itu mengajukan hasil melalui biasa tuntutan

paging, mengakibatkan kesalahan halaman. Namun, sebagian file berukuran halaman adalah membaca dari sistem file ke halaman fisik (beberapa sistem mungkin memilih untuk membaca di dalam lebih dari A berukuran halaman bingkah dari Penyimpanan pada A waktu). Setelah membaca Dan penulisan ke file ditangani sebagai akses memori rutin. Memanipulasi file melalui memori daripada menimbulkan overhead penggunaan read() dan menulis() sistem panggilan menyederhanakan Dan kecepatan ke atas mengajukan mengakses Dan penggunaan.

Perhatikan bahwa penulisan ke file yang dipetakan dalam memori belum tentu segera terjadi. diate (sinkronis) menulis ke itu mengajukan pada sekunder penyimpanan. Umumnya, sistem perbarui file berdasarkan perubahan pada gambar memori hanya ketika file tersebut tertutup. Di bawah tekanan memori, sistem akan mengalami perubahan perantara untuk menukar ruang agar tidak hilang saat mengosongkan memori untuk penggunaan lain. Kapan Jika file ditutup, semua data yang dipetakan memori ditulis kembali ke file tersebut sekunder penyimpanan Dan DIHAPUS dari itu maya Penyimpanan dari itu proses.

Beberapa sistem operasi menyediakan pemetaan memori hanya melalui tertentu panggilan sistem dan gunakan panggilan sistem standar untuk melakukan semua I/O file lainnya . Namun, beberapa sistem memilih untuk memetakan file dalam memori terlepas dari apakah file tersebut mengajukan dulu ditentukan sebagai dipetakan memori. Ayo mengambil Solaris sebagai sebuah contoh. Jika A mengajukan ditentukan sebagai dipetakan memori (menggunakan panggilan sistem mmap()), peta Solaris file ke dalam ruang alamat proses. Jika suatu file dibuka dan diakses menggunakan biasa sistem panggilan, seperti sebagai membuka() , membaca() , Dan menulis() , Solaris tetap memetakan memori file; namun, file tersebut dipetakan ke ruang alamat kernel. Tanpa memedulikan dari Bagaimana itu mengajukan adalah dibuka, Kemudian, Solaris suguhan semua mengajukan masukan/keluaran sebagai Penyimpanan- dipetakan, memungkinkan mengajukan mengakses ke mengambil tempat melalui itu efisien Penyimpanan subsistem dan menghindari overhead panggilan sistem yang disebabkan oleh masing-masing read() dan tradisional menulis() .

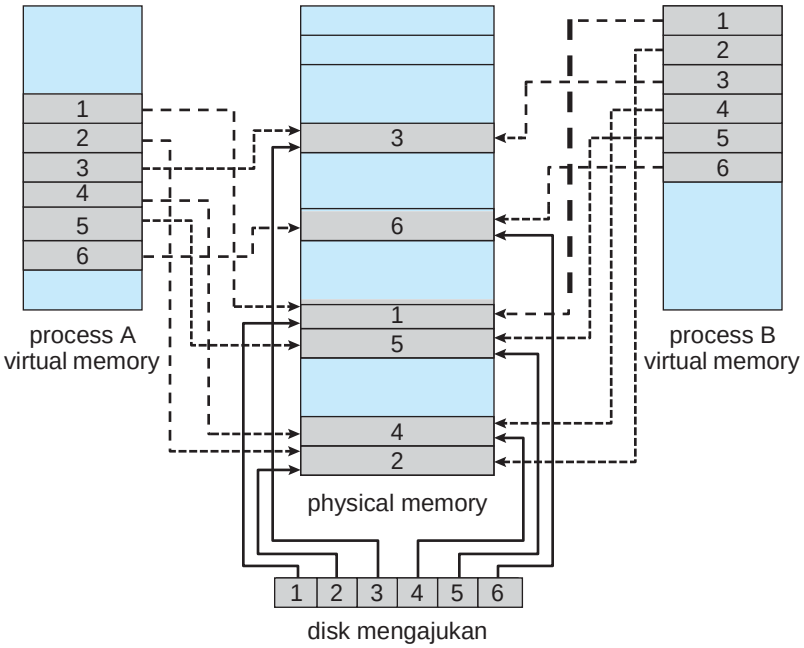
Beberapa proses mungkin diizinkan untuk memetakan file yang sama secara bersamaan mengizinkan membagikan dari data. Menulis oleh setiap dari itu proses memodifikasi itu data di dalam maya memori dan dapat dilihat oleh semua orang lain yang memetakan bagian file yang sama. Mengingat diskusi kita sebelumnya tentang memori virtual, seharusnya sudah jelas bagaimana cara kerjanya berbagi bagian memori yang dipetakan memori diimplementasikan: virtual peta memori dari setiap proses berbagi menunjuk ke halaman fisik yang sama memori — halaman yang menyimpan salinan blok disk. Berbagi memori ini adalah diilustrasikan pada Gambar 13.13. Panggilan sistem pemetaan memori juga dapat mendukung fungsionalitas copy-on-write, memungkinkan proses untuk berbagi file dalam mode read-only mode Tetapi ke memiliki milik mereka memiliki salinan dari setiap data mereka memodifikasi. Jadi itu mengakses ke itu bersama data adalah terkoordinasi, itu proses terlibat mungkin menggunakan satu dari itu mekanisme untuk mencapai saling pengecualian dijelaskan di dalam Bab 6.

Seringkali, memori bersama sebenarnya diimplementasikan dengan pemetaan memori file. Di bawah ini skenario, proses Bisa menyampaikan menggunakan bersama mem- ory dengan meminta proses komunikasi memetakan memori file yang sama ke dalamnya milik mereka maya alamat spasi. Itu dipetakan memori mengajukan melayani sebagai itu wilayah memori bersama antara proses komunikasi (Gambar 13.14). Kami memiliki

sudah terlihat ini di dalam Bagian 3.5, Di mana A POSIX Berbagi memori obyek dibuat dan setiap proses komunikasi memetakan memori objek ke dalam memorinya ruang alamat. Di bagian berikut, kita membahas dukungan di Windows API untuk bersama Penyimpanan menggunakan pemetaan memori file.

13.5.2 Bersama Penyimpanan di dalam itu jendela API

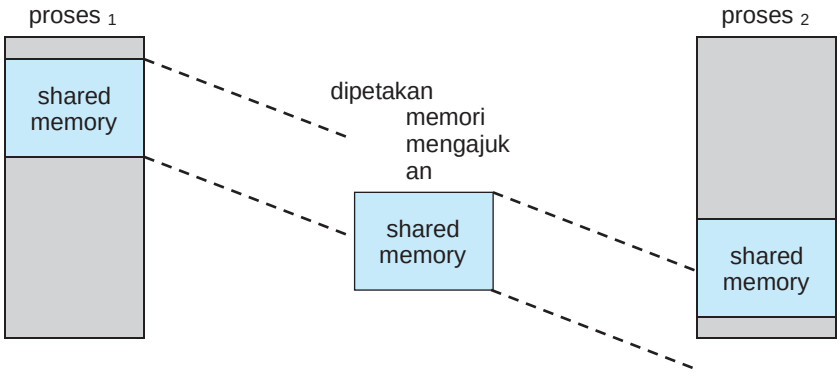
Garis besar umum untuk membuat wilayah memori bersama menggunakan memori- dipetakan file di dalam itu jendela API melibatkan Pertama menciptakan A **file pemetaan** untuk itu



Gambar 13.13 Dipetakan memori file.

file yang akan dipetakan dan kemudian membuat **tampilan** file yang dipetakan dalam suatu proses ruang alamat virtual. Proses kedua kemudian dapat membuka dan membuat tampilan file yang dipetakan dalam ruang alamat virtualnya. File yang dipetakan mewakili objek memori bersama yang akan memungkinkan terjadinya komunikasi antar itu proses.

Kami selanjutnya mengilustrasikan langkah-langkah ini secara lebih rinci. Dalam contoh ini, seorang produser proses pertama-tama membuat objek memori bersama menggunakan fitur pemetaan memori tur tersedia di Windows API . Produser kemudian menulis pesan kepada bersama Penyimpanan. Setelah itu, A konsumen proses terbuka A pemetaan ke itu bersama- Penyimpanan obyek Dan membaca itu pesan tertulis oleh itu konsumen.



Angka 13.14 Bersama Penyimpanan menggunakan dipetakan memori

masukan/keluaran.

Untuk membuat file yang dipetakan memori, suatu proses terlebih dahulu membuka file yang akan dipetakan dipetakan dengan fungsi CreateFile() , yang mengembalikan HANDLE ke berkas yang dibuka. Proses kemudian membuat pemetaan file HANDLE ini menggunakan Fungsi CreateFileMapping() . Setelah pemetaan file selesai, prosesnya menetapkan tampilan file yang dipetakan di ruang alamat virtualnya dengan Tampilan PetaDariFile() fungsi. Itu melihat dari itu dipetakan mengajukan mewakili itu por- tion dari file yang dipetakan dalam ruang alamat virtual proses— the seluruh mengajukan atau hanya A bagian dari dia mungkin menjadi dipetakan. Ini urutan di dalam itu program

```
#termasuk <windows.h>
#termasuk <stdio.h>

ke dalam utama(int argumen, arang *argumen[])
{
   MENANGANI hFile, hFile
   Peta;LPVOID
   lpMapAddress;

   hFile = BuatFile("temp.txt", /* mengajukan nama */
      UMUM MEMBACA | UMUM MENULIS, /* Baca tulis
      mengakses */0, /* TIDAK membagikan dari itu
      mengajukan */
      BATAL, /* bawaan keamanan */
      MEMBUKA SELALU, /* membuka baru atau yang ada mengajukan
      */
      FILE NORMAL, /* rutin mengajukan atribut */
      BATAL); /* TIDAK mengajukan templat */

   hMapFile = Buat Pemetaan File (hFile, /* mengajukan
      menangani */BATAL, /* bawaan keamanan */
      HALAMAN BACA TULIS, /* Baca tulis mengakses ke dipetakan
      halaman */0, /* peta seluruh mengajukan */
      0,
      TEXT("Objek Bersama")); /* bernama bersama Penyimpanan obyek */

   lpMapAddress = MapViewOfFile(hMapFile, /* dipetakan obyek menangani
      */ MENGAJUKAN PETA SEMUA MENGAKSES, /* Baca tulis mengakses */
      0, /* dipetakan melihat dari
      seluruh mengajukan */0,
      0);

   /* menulis ke bersama Penyimpanan */
   sprintf(lpMapAddress,"Dibagikan Penyimpanan pesan");

   UnmapViewOfFile( lpMa p A d r
   e s s ) ; CloseHandle(hFile);
   CloseHandle(hMapFile);
}
```

Angka 13.15 Produsen menulis ke bersama Penyimpanan menggunakan itu jendela API.

ditunjukkan pada Gambar 13.15. (Kami menghilangkan banyak kesalahan saat memeriksa kode keringkasan.)

Panggilan ke CreateFileMapping() menciptakan **objek memori bersama bernama** ditelepon Objek Bersama . Itu konsumen proses akan menyampaikan menggunakan ini segmen memori bersama dengan membuat pemetaan ke objek bernama sama. Produser kemudian membuat tampilan file yang dipetakan memori di virtualnya ruang alamat. Dengan melewati tiga parameter terakhir nilai 0, ini menunjukkan bahwa tampilan yang dipetakan adalah keseluruhan file. Sebaliknya, itu bisa saja melewati nilai-nilai menentukan sebuah mengimbangi Dan ukuran, dengan demikian menciptakan A melihat mengandung hanya A ayat dari file tersebut. (Penting untuk dicatat bahwa keseluruhan pemetaan mungkin tidak dimuat ke dalam memori ketika pemetaan dibuat. Sebaliknya, file yang dipetakan mungkin halaman permintaan, sehingga membawa halaman ke dalam memori hanya saat diakses.) Fungsi MapViewOfFile() mengembalikan pointer ke objek memori bersama; setiap mengakses ke ini Penyimpanan lokasi adalah dengan demikian mengakses ke itu dipetakan memori mengajukan. Dalam hal ini, proses produser menulis pesan " Dibagikan Penyimpanan pesan " ke bersama Penyimpanan.

Sebuah program yang menggambarkan bagaimana proses konsumen menetapkan pandangan itu bernama Berbagi memori obyek adalah ditampilkan di dalam Angka 13.16. Ini program adalah

```
#termasuk <windows.h>
#termasuk <stdio.h>

ke dalam utama(int argumen, arang
{
   *argumen[])MENANGANI hFile Peta;
   LPVOID lpMapAddress;

   hMapFile = OpenFileMapping(FILE PETA SEMUA MENGAKSES, /* R/W
      mengakses */PALSU, /* TIDAK warisan */
      TEXT("Objek Bersama")); /* nama dari dipetakan mengajukan
      objek */

   lpMapAddress = MapViewOfFile(hMapFile, /* dipetakan obyek
      menangani */MENGAJUKAN PETA SEMUA MENGAKSES, /* Baca tulis
      mengakses */
      0, /* dipetakan melihat dari
      seluruh mengajukan */0,
      0);

   /* membaca dari Berbagi memori */
   printf("Baca pesan %S", lpMapAddress);

   UnmapViewOfFile(lpMapAddress);
   CloseHandle(hMapFile);
}
```

Angka 13.16 Konsumen membaca dari bersama Penyimpanan menggunakan itu jendela

API.

agak lebih sederhana daripada yang ditunjukkan pada Gambar 13.15, karena semua itu diperlukan adalah untuk proses membuat pemetaan ke memori bersama bernama yang sudah ada obyek. Proses konsumen juga harus membuat tampilan file yang dipetakan sebagai itu produsen proses telah melakukan di dalam itu program di dalam Angka 13.15. Itu konsumen Kemudian membaca dari memori bersama pesan " Dibagikan Penyimpanan pesan " itu taditertulis oleh produser proses.

Terakhir, kedua proses menghapus tampilan file yang dipetakan dengan panggilan ke Hapus petaViewOfFile() . Kami menyediakan latihan pemrograman di akhir ini bab menggunakan bersama Penyimpanan dengan Penyimpanan pemetaan di dalam itu jendela API .

13.6 Ringkasan

- File adalah tipe data abstrak yang ditentukan dan diimplementasikan oleh operasi sistem. Ini adalah urutan catatan logis. Catatan logis mungkin berupa byte, sebuah garis (dengan panjang tetap atau variabel), atau item data yang lebih kompleks. Itu sistem operasi mungkin secara khusus mendukung berbagai jenis rekaman atau mungkin meninggalkan dukungan itu ke aplikasi program.
- Tugas utama sistem operasi adalah memetakan konsep file logis ke perangkat penyimpanan fisik seperti hard disk atau perangkat NVM . Sejak itu fisik catatan ukuran dari itu perangkat mungkin bukan menjadi itu sama sebagai itu logis catatan ukuran, mungkin perlu untuk mengurutkan catatan logis ke dalam catatan fisik. Sekali lagi, tugas ini mungkin didukung oleh sistem operasi atau diserahkan kepada aplikasi program.
- Dalam sistem file, berguna untuk membuat direktori untuk memungkinkan file berada terorganisir. Direktori satu tingkat dalam sistem multipengguna menyebabkan penamaan masalah, karena setiap file harus memiliki unik nama. A dua tingkat direktori memecahkan masalah ini dengan membuat direktori terpisah untuk setiap file pengguna. Itu direktori daftar itu file oleh nama Dan termasuk itu file lokasi pada itu disk, panjang, jenis, pemilik, waktu dari penciptaan, waktu dari terakhir menggunakan, Dan Jadi pada.
- Generalisasi alami dari direktori dua tingkat adalah terstruktur pohon direktori. A terstruktur pohon direktori memungkinkan A pengguna ke membuat subdirektori untuk mengatur file. Struktur direktori grafik asiklik memungkinkan pengguna untuk berbagi subdirektori dan file tetapi mempersulit pencarian dan penghapusan. Seorang jenderal struktur grafik memungkinkan fleksibilitas penuh dalam berbagi file dan arahan. cerita tetapi terkadang memerlukan pengumpulan sampah untuk memulihkan disk yang tidak digunakan ruang angkasa.
- Sistem file jarak jauh menghadirkan tantangan dalam keandalan, kinerja, dan keamanan. Sistem informasi terdistribusi memelihara pengguna, host, dan akses informasi Jadi itu klien Dan server Bisa membagikan negara informasi ke mengelola menggunakan dan akses.
- Sejak file adalah itu utama penyimpanan informasi mekanisme di dalam paling komputer sistem, perlindungan file diperlukan pada sistem multipengguna. Akses ke file Bisa menjadi dikendalikan secara terpisah

untuk setiap jenis dari mengakses- membaca, menulis, menjalankan, menambahkan, menghapus, daftar direktori, dan sebagainya. Perlindungan file dapat disediakan oleh mengakses daftar, kata sandi, atau yang lainnya teknik.

Praktik Latihan

13.1 Beberapa sistem secara otomatis menghapus semua pengguna file Kapan A pengguna log mati atau suatu pekerjaan dihentikan, kecuali pengguna secara eksplisit memintanya untuk menyimpannya. Sistem lain menyimpan semua file kecuali pengguna menghapusnya secara eksplisit. Membahas itu relatif manfaat dari setiap mendekati.

13.2 Mengapa Mengerjakan beberapa sistem menyimpan melacak dari itu jenis dari A mengajukan, ketika tetap yang lain serahkan pada pengguna dan orang lain tidak mengimplementasikan banyak file jenis? Yang sistem adalah " lebih baik " ?

13.3 Demikian pula, beberapa sistem mendukung banyak jenis struktur untuk suatu file data, sementara yang lain hanya mendukung aliran byte. Apa itu keuntungan dan kekurangannya setiap mendekati?

13.4 Bisa Anda mensimulasikan A bertingkat direktori struktur dengan A tingkat tunggal struktur direktori di mana nama panjang dapat digunakan? Jika Anda jawabannya adalah ya, jelaskan bagaimana Anda dapat melakukannya, dan kontraskan skema ini dengan skema direktori bertingkat. Jika jawaban Anda tidak, jelaskan apa mencegah milikmu simulasi kesuksesan. Bagaimana akan milikmu menjawab mengubah jikamengajukan nama adalah terbatas ke tujuh karakter?

13.5 Menjelaskan itu tujuan dari itu `membuka()` Dan `menutup()` operasi.

13.6 Dalam beberapa sistem, subdirektori dapat dibaca dan ditulis oleh penulis.diubah pengguna, hanya sebagai biasa file Bisa menjadi.

a. Menggambarkan itu perlindungan masalah itu bisa timbul.

b. Menyarankan A skema untuk berurusan dengan setiap dari ini perlindungan masalah-lem.

13.7 Mempertimbangkan A sistem itu mendukung 5.000 pengguna. Memperkirakan itu Anda ingin kemengizinkan 4.990 ini pengguna ke bisa ke mengakses satu mengajukan.

a. Bagaimana akan Anda menentukan ini perlindungan skema di dalam UNIX ?

b. Bisa Anda menyarankan lain perlindungan skema itu Bisa menjadi digunakan lagisecara efektif untuk ini tujuan dibandingkan itu skema asalkan oleh UNIX ?

13.8 Para peneliti telah menyarankan hal itu, daripada memiliki kontrol akses daftar terkait dengan setiap mengajukan (menentukan yang pengguna Bisa mengakses itu mengajukan, Dan Bagaimana), Kami sebaiknya memiliki A **pengguna kontrol daftar** terkait dengan setiap pengguna (menentukan yang file A pengguna Bisa mengakses, Dan Bagaimana). Membahas itu relatifmanfaat ini dua skema.

Lebih jauh Membaca

Struktur direktori bertingkat pertama kali diimplementasikan pada sistem MULTICS ([Organik (1972)]). Paling Pengoperasian sistem Sekarang melaksanakan bertingkat arah- struktur sejarah. Ini termasuk Linux ([Love

(2010)]), mac OS ([Singh (2007)]), Solaris ([McDougall Dan Mauro (2007)]), Dan semua versi dari jendela ([Rusia- novich et Al. (2017)]).

Diskusi umum tentang sistem file Solaris dapat ditemukan di Sun *Sys- Panduan Administrasi tem: Perangkat dan Sistem File* (http://docs.sun.com/app/ dokumen/doc/817-5093).

Sistem file jaringan (NFS), yang dirancang oleh Sun Microsystems, memungkinkan struktur direktori untuk disebarkan ke seluruh sistem komputer jaringan. NFS Versi: kapan 4 adalah dijelaskan di dalam RFC 3505 (http://www.ietf.org/rfc/rfc3530.txt).

A Besar sumber dari itu makna dari komputer jargon adalah http://www.catb.org/esr/jargon/ .

Bibliografi

[Cinta (2010)] R. Cinta, *Linux Inti Perkembangan,* Ketiga Edisi, Pengembang Perpustakaan (2010).

[McDougall Dan Mauro (2007)] R. McDougall Dan J. mauro, *Solaris Internal,* Kedua Edisi, Pembantu tukang Aula (2007).

[Organik (1972)] E. SAYA. organik, *Itu Multik Sistem: Sebuah Penyelidikan dari Dia Struktur-masa depan* , MIT Tekan (1972).

[Russinovich et Al. (2017)] M. Russinovich, D. A. Salomo, Dan A. Ionescu, *Menang-turun Internal–Bagian 1,* Ketujuh Edisi, Microsoft Tekan (2017).

[Singh (2007)] A. Singh, *Mac sistem operasi X Internal: A Sistem Mendekati* , Addison-Wesley (2007).

Bab 13 Latihan

13.9 Mempertimbangkan A mengajukan sistem di dalam yang a mengajukan dapat dihapus Dan -nya disk ruang angkasa direklamasi ketika tautan ke itu mengajukan tetap ada. Apa masalah mungkin terjadi jika A baru mengajukan adalah dibuat di dalam itu sama penyimpanan daerah atau dengan itu sama mutlak jalur nama? Bagaimana Bisa ini masalah menjadi dihindari?

13.10 Itu membuka file meja adalah digunakan ke menjaga informasi tentang file itu adalah saat ini membuka. Sebaiknya itu Pengoperasian sistem menjaga A memisahkan meja untuk setiap pengguna atau menjaga hanya satu meja itu mengandung referensi ke file yang saat ini sedang diakses oleh semua pengguna? Jika file yang sama sedang diakses oleh dua program atau pengguna yang berbeda, sebaiknya ada yang terpisahentri di dalam itu membuka file meja? Menjelaskan.

13.11 Apa kelebihan dan kekurangan pemberian wajib kunci alih-alih dari penasehat kunci yang menggunakan adalah kiri ke pengguna kebijaksanaan?

13.12 Berikan contoh aplikasi yang biasanya mengakses file menurut ke metode berikut:

- Sekuensial
- Acak

13.13 Beberapa sistem secara otomatis membuka file ketika direferensikan untuk yang pertama waktu Dan menutup itu mengajukan Kapan itu pekerjaan berakhir. Membahas itu keuntungan Dan kekurangan dari ini skema dibandingkan dengan itu lagi tradisional satu, Di mana itu pengguna memiliki ke membuka Dan menutup itu mengajukan secara eksplisit.

13.14 Jika itu Pengoperasian sistem tahu itu A yakin aplikasi dulu pergi untuk mengakses data file secara berurutan, bagaimana bisa memanfaatkan ini informasi ke memperbaiki pertunjukan?

13.15 Berikan contoh aplikasi yang dapat memperoleh manfaat dari pengoperasian- sistem mendukung untuk acak akses ke diindeks file.

13.16 Beberapa sistem menyediakan berbagi file dengan memelihara satu salinan a mengajukan. Sistem lain menyimpan beberapa salinan, satu untuk setiap pengguna membagikan itu mengajukan. Membahas itu relatif manfaat dari setiap mendekati.

CHAPTER 14

Berkas sistem Penerapan

Sebagai Kami gergaji di dalam Bab 13, itu mengajukan sistem menyediakan itu mekanisme untuk pada-garis penyimpanan Dan mengakses ke mengajukan isi, termasuk data Dan program. Mengajukan sistem biasanya tinggal secara permanen pada sekunder penyimpanan, yang adalah dirancangke memegang A besar jumlah dari data. Ini bab adalah terutama khawatir dengan masalahsekitarnya mengajukan penyimpanan Dan mengakses pada itu paling umum penyimpanan sekundermedia, keras disk drive Dan tidak mudah menguap Penyimpanan perangkat. Kami mengeksplorasi carake struktur mengajukan menggunakan, ke mengalokasikan penyimpanan ruang angkasa, ke pulih dibebaskan ruang angkasa, ke melacakitu lokasi dari data, Dan ke antarmuka lainnya bagian dari itu Pengoperasian sistem kesekunder penyimpanan. Pertunjukan masalah adalah dipertimbangkan selama itu bab.A diberikan tujuan umum Pengoperasian sistem menyediakan beberapa mengajukan sistem. Selain itu, banyak Pengoperasian sistem mengizinkan administrator atau pengguna ke menambahkan mengajukan sistem. Mengapa begitu banyak? Sistem file bervariasi dalam banyak hal, termasuk fitur, kinerja, keandalan, dan tujuan desain, serta sistem file yang berbeda dapat berfungsi tujuan yang berbeda. Misalnya, sistem file sementara digunakan untuk penyimpanan cepat dan pengambilan file nonpersisten, sedangkan file penyimpanan sekunder default sistem (seperti sebagai Linux ext4) pengorbanan pertunjukan untuk keandalan Dan fitur. Sebagai kita sudah terlihat selama ini belajar dari Pengoperasian sistem, di sana adalah banyak dari pilihan Dan variasi, membuat menyeluruh cakupan A tantangan. Di dalam ini bab, Kami konsentrat pada itu umum penyebut.

CHAPTER OBJECTIVES

- Describe the details of implementing local file systems and directory structures.
- Discuss block allocation and free-block algorithms and trade-offs.
- Explore file system efficiency and performance issues.
- Look at recovery from file system failures.
- Describe the WAFL file system as a concrete example.

14.1 Berkas sistem Struktur

Disk menyediakan paling dari itu sekunder penyimpanan pada yang mengajukan sistem adalah utama-tertahan. Dua karakteristik membuat mereka nyaman untuk ini tujuan:

1. Disk dapat ditulis ulang pada tempatnya; adalah mungkin untuk membaca satu blok dari disk, memodifikasi itu memblokir, Dan menulis dia kembali ke dalam itu sama memblokir.
2. Sebuah disk dapat mengakses secara langsung blok informasi apa pun yang dikandungnya. Jadi, memang demikian mudah untuk mengakses file apa pun baik secara berurutan atau acak, dan berpindah dari satu file ke file lainnya memerlukan drive yang menggerakkan kepala baca-tulis Dan menunggu untuk media ke memutar.

memori nonvolatile (NVM) semakin banyak digunakan untuk penyimpanan file Dan dengan demikian sebagai A lokasi untuk mengajukan sistem. Mereka berbeda dari keras disk di dalam itu mereka tidak dapat ditulis ulang pada tempatnya dan memiliki karakteristik kinerja yang berbeda- tics. Kami membahas disk Dan NVM -perangkat struktur di dalam detail di dalam Bab 11.

Ke memperbaiki masukan/keluaran efisiensi, masukan/keluaran transfer di antara Penyimpanan Dan massa penyimpanan dilakukan dalam satuan **blok** . Setiap blok pada hard disk drive memiliki satu atau lebih banyak sektor. Tergantung pada disk drive, ukuran sektor biasanya 512 byte atau 4.096 byte. Perangkat NVM biasanya memiliki blok 4.096 byte, dan transfer metode digunakan adalah serupa ke yang digunakan berdasarkan disk drive.

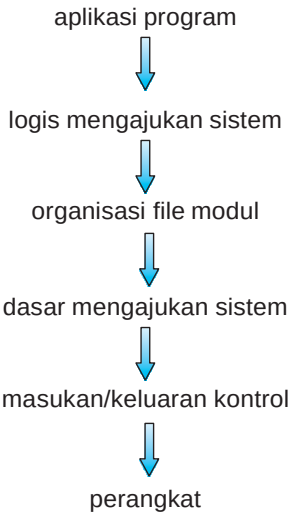
Mengajukan sistem menyediakan efisien Dan nyaman mengakses ke itu penyimpanan perangkat dengan memungkinkan data disimpan, ditemukan, dan diambil dengan mudah. Sebuah sistem file berpose dua masalah desain yang sangat berbeda. Masalah pertama adalah mendefinisikan bagaimana sistem file harus terlihat oleh pengguna. Tugas ini melibatkan pendefinisian file dan file-nya atribut, operasi yang diperbolehkan pada file, dan struktur direktori pengorganisasian file. Itu Kedua masalah adalah menciptakan algoritma Dan data struktur ke peta itu logis mengajukan sistem ke itu fisik penyimpanan sekunder perangkat.

Sistem file itu sendiri umumnya terdiri dari banyak level berbeda. Itu struktur ditampilkan di dalam Angka 14.1 adalah sebuah contoh dari A berlapis desain. Setiap tingkat di dalam desainnya menggunakan fitur-fitur tingkat yang lebih rendah untuk membuat fitur-fitur baru untuk digunakan oleh lebih tinggi tingkat.

Itu **masukan/keluaran kontrol** tingkat terdiri dari driver perangkat dan penangan interupsi untuk mentransfer informasi antara memori utama dan sistem disk. A perangkat pengemudi Bisa menjadi pikiran dari sebagai A Penerjemah. Dia memasukkan terdiri dari tinggi- tingkat perintah, seperti sebagai " mengambil memblokir 123. ” Dia keluaran terdiri dari level rendah, khusus perangkat keras instruksi itu adalah digunakan oleh itu perangkat keras pengontrol, yang menghubungkan perangkat I/O ke seluruh sistem. Driver perangkat biasanya menulis pola bit tertentu ke lokasi khusus di memori pengontrol I/O untuk memberi tahu pengontrol lokasi perangkat mana yang harus ditindaklanjuti dan tindakan apa yang harus diambil. Itu detail dari perangkat pengemudi Dan itu masukan/keluaran infrastruktur adalah tertutupi di dalam Bab12.

Sistem **file dasar** (disebut “ subsistem I/O blok ” di Linux) memerlukan

hanya untuk mengeluarkan perintah umum ke driver perangkat yang sesuai untuk membaca dan menulis blok pada itu penyimpanan perangkat. Dia masalah perintah ke itu menyetir berdasarkan pada alamat blok logis. Hal ini juga berkaitan dengan penjadwalan permintaan I/O . Ini lapisan Juga mengelola itu Penyimpanan buffer Dan cache itu memegang bermacam-macam mengajukan- sistem, direktori, dan blok data. Sebuah blok di buffer dialokasikan sebelum transfer blok penyimpanan massal dapat terjadi. Ketika buffer penuh, buffer Pengelola harus menemukan lagi penyangga Penyimpanan atau bebas ke atas penyangga ruang angkasa ke mengizinkan A



Angka 14.1 Berlapis mengajukan sistem.

diminta masukan/keluaran ke menyelesaikan. Tembolok adalah digunakan ke memegang sering digunakan berkas sistem metadata untuk meningkatkan kinerja, jadi mengelola kontennya sangat penting optimal sistem pertunjukan.

Organisasi file **modul** mengetahui tentang file dan blok logisnya. Blok logis setiap file diberi nomor dari 0 (atau 1) hingga *N*. File- organisasi modul Juga termasuk itu ruang bebas Pengelola, yang trek unal- terletak blok Dan menyediakan ini blok ke itu organisasi file modul Kapandiminta.

Terakhir, **sistem file logis** mengelola informasi metadata. Metadata mencakup semua struktur sistem file kecuali data aktual (atau konten file). Sistem file logis mengelola struktur direktori yang akan disediakan modul pengorganisasian file dengan informasi yang dibutuhkan, diberikan a nama file simbolis. Itu mempertahankan struktur file melalui blok kontrol file. File **kontrol memblokir** (**FCB**) (sebuah **inode** di dalam UNIX mengajukan sistem) mengandung informasi tentang file, termasuk kepemilikan, izin, dan lokasi konten file. Itu sistem file logis juga bertanggung jawab atas perlindungan, seperti yang dibahas dalam Bab 13 Dan 17.

Kapan A berlapis struktur adalah digunakan untuk berkas sistem penerapan, duplikat- tion dari kode adalah diminimalkan. Itu masukan/keluaran kontrol Dan Kadang-kadang itu dasar berkas sistem kode dapat digunakan oleh banyak sistem file. Setiap sistem file kemudian dapat memilikinya sendiri memiliki sistem file logis dan modul organisasi file. Sayangnya, berlapis-lapis dapat menimbulkan lebih banyak overhead sistem operasi, yang dapat mengakibatkan penurunan pertunjukan. Itu menggunakan dari pelapisan, termasuk itu keputusan tentang Bagaimana banyak berbaring-apa yang harus digunakan dan apa yang harus dilakukan setiap lapisan, merupakan tantangan besar dalam merancang yang baru sistem.

Banyak sistem file yang digunakan saat ini, dan sebagian besar sistem operasi mendukungnya lebih dari satu. Misalnya, sebagian besar CD-ROM ditulis dalam standar ISO 9660. mat, format standar yang disetujui oleh produsen CD-ROM . Sebagai tambahan sistem file media yang dapat dilepas, setiap sistem operasi memiliki satu atau lebih disk- berdasarkan mengajukan sistem. UNIX kegunaan itu **UNIX file sistem** (**UFS**), yang adalah berdasarkan pada itu

Berkeley Cepat Mengajukan Sistem (FFS). jendela mendukung disk berkas sistem format dari FAT , FAT 32, dan NTFS (atau Sistem File Windows NT), serta CD-ROM dan DVD format sistem file. Meskipun Linux mendukung lebih dari 130 sistem file yang berbeda, itu standar Linux mengajukan sistem adalah diketahui sebagai itu **diperpanjang mengajukan sistem** , dengan itu paling umum versi makhluk ext3 Dan ext4. Di sana adalah Juga didistribusikan mengajukan sistem di mana sistem file di server dipasang oleh satu atau lebih klien komputer melintasi suatu jaringan.

Penelitian sistem file terus menjadi area aktif sistem operasi desain Dan penerapan. Google dibuat -nya memiliki mengajukan sistem ke bertemu itu milik perusahaan spesifik penyimpanan Dan pengambilan kebutuhan, yang termasuk tinggi- akses kinerja dari banyak klien di sejumlah besar disk. Proyek menarik lainnya adalah sistem file FUSE , yang memberikan fleksibilitas dalam berkas sistem perkembangan Dan menggunakan oleh menerapkan Dan mengeksekusi mengajukan sistem sebagai kode tingkat pengguna dan bukan tingkat kernel. Dengan menggunakan FUSE , pengguna dapat menambahkan yang baru sistem file ke berbagai sistem operasi dan dapat menggunakan sistem file tersebut untuk mengelola dia file.

14.2 Berkas sistem Operasi

Seperti yang dijelaskan di Bagian 13.1.2, sistem operasi mengimplementasikan open() dan close() sistem memanggil proses untuk meminta akses ke konten file. Di dalam bagian, kami mempelajari struktur dan operasi yang digunakan untuk mengimplementasikan file- sistem operasi.

14.2.1 Ringkasan

Beberapa struktur di penyimpanan dan di memori digunakan untuk mengimplementasikan file sistem. Ini struktur bervariasi tergantung pada itu Pengoperasian sistem Dan itu mengajukan sistem, Tetapi beberapa umum prinsip menerapkan.

Pada penyimpanan, sistem file mungkin berisi informasi tentang caranya untuk boot sistem operasi yang disimpan di sana, jumlah total blok, jumlah dan lokasi blok bebas, struktur direktori, dan file individual. Kebanyakan struktur ini dirinci sepanjang sisa bab ini. Di Sini, Kami menggambarkan mereka secara singkat:

- Blok **kontrol boot** (per volume) dapat berisi informasi yang dibutuhkan oleh sistem ke boot sebuah Pengoperasian sistem dari itu volume. Jika itu disk melakukan bukan berisi sistem operasi, blok ini boleh kosong. Ini biasanya adalah Pertama memblokir dari A volume. Di dalam UFS , dia adalah ditelepon itu **boot memblokir** . Di dalam NTFS , dia adalah itu **partisi boot sektor** .
- Blok **kontrol volume** (per volume) berisi detail volume, seperti jumlah blok dalam volume, ukuran blok, jumlah blok bebas Dan blok bebas petunjuk, Dan A gratis- FCB menghitung Dan FCB petunjuk. Di dalam UFS , ini adalah ditelepon A **blok super** . Di dalam NTFS , dia adalah disimpan di dalam itu **menguasai file meja** .
- Struktur direktori (per sistem file) digunakan untuk mengatur file. Di UFS , ini termasuk mengajukan nama Dan terkait inode angka. Di dalam

NTFS , dia adalah disimpandi dalam itu menguasai mengajukan meja.

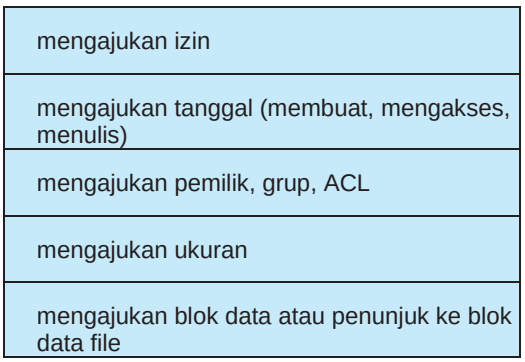
- FCB per file berisi banyak detail tentang file tersebut. Ini memiliki pengenal unik nomor ke mengizinkan asosiasi dengan A direktori pintu masuk. Di dalam NTFS , ini informasi- tion sebenarnya disimpan dalam tabel file master, yang menggunakan relasional basis data struktur, dengan berturut-turut per mengajukan.

Itu dalam kenangan informasi adalah digunakan untuk keduanya berkas sistem pengelolaan Dan peningkatan kinerja melalui caching. Data dimuat pada waktu pemasangan, diperbarui selama operasi sistem file, dan dibuang saat turun. Beberapa jenis struktur Mungkin termasuk.

- **Tabel pemasangan** di memori berisi informasi tentang setiap pemasangan volume.
- Cache struktur direktori dalam memori menyimpan informasi direktori dari direktori yang baru diakses. (Untuk direktori di mana volume berada dipasang, dia Bisa mengandung a penunjuk ke itu volume meja.)
- **File terbuka seluruh** sistem **tabel** berisi salinan FCB setiap terbuka mengajukan, demikian juga seperti lainnya informasi.
- Itu **per proses file terbuka meja** mengandung petunjuk ke itu sesuai entri dalam tabel file terbuka seluruh sistem, serta informasi lainnya, untuk semua file itu proses telah terbuka.
- Buffer menyimpan blok sistem file ketika sedang dibaca atau ditulis ke file sistem.

Untuk membuat file baru, suatu proses memanggil sistem file logis. File logis sistem mengetahui format struktur direktori. Untuk membuat file baru, itu mengalokasikan FCB baru . (Atau, jika implementasi sistem file dibuat semua FCB pada waktu pembuatan sistem file, FCB dialokasikan dari kumpulan gratis FCB s.) Itu sistem Kemudian membaca itu sesuai direktori ke dalam Penyimpanan, pembaruan dengan nama file baru dan FCB , dan menulisnya kembali ke sistem file. Sebuah tipikal FCB adalah ditampilkan di dalam Angka 14.2.

Beberapa Pengoperasian sistem, termasuk UNIX , merawat A direktori tepat itu sama sebagai A mengajukan — satu dengan A " jenis " bidang menunjukkan itu dia adalah A direktori. Lainnya operasi-



Angka 14.2 A khas kontrol file memblokir.

sistem makan, termasuk Windows, menerapkan panggilan sistem terpisah untuk file dan direktori dan memperlakukan direktori sebagai entitas yang terpisah dari file. Apa pun itu lebih besar struktural masalah, itu logis mengajukan sistem Bisa panggilan itu organisasi file modul ke peta itu direktori masukan/keluaran ke dalam penyimpanan memblokir lokasi, yang adalah lulus pada ke dasar berkas sistem Dan masukan/keluaran sistem pengaturan.

14.2.2 Penggunaan

Sekarang itu A mengajukan memiliki pernah dibuat, dia Bisa menjadi digunakan untuk masukan/keluaran . Pertama, meskipun, dia harusmenjadi dibuka. Itu membuka() panggilan berlalu A mengajukan nama ke itu logis mengajukan sistem. Itumembuka() sistem panggilan Pertama pencarian itu seluruh sistem membuka file meja ke melihat jika itu mengajukanadalah sudah di dalam menggunakan oleh lain proses. Jika dia adalah, A per proses membuka file meja pintu masuk adalahdibuat menunjuk ke itu yang ada seluruh sistem membuka file meja. Ini algoritma Bisamenyimpan besar atas. Jika itu mengajukan adalah bukan sudah membuka, itu direktori struktur adalah dicari untuk itu diberikan mengajukan nama. Bagian dari itu direktori struktur adalah biasanyadi-cache di dalam Penyimpanan ke kecepatan direktori operasi. Sekali itu mengajukan adalah ditemukan, ituFCB adalah disalin ke dalam A seluruh sistem membuka file meja di dalam Penyimpanan. Ini meja bukan hanyatoko itu FCB Tetapi Juga trek itu nomor dari proses itu memiliki itu mengajukan membuka.

Berikutnya, sebuah pintu masuk adalah dibuat di dalam itu per proses membuka file meja, dengan A penunjuk ke entri dalam tabel file terbuka seluruh sistem dan beberapa bidang lainnya. Ini lainnya bidang mungkin termasuk A penunjuk ke itu saat ini lokasi di dalam itu mengajukan (untuk itu Berikutnya operasi read() atau write()) dan mode akses di mana file dibuka. Panggilan open() mengembalikan pointer ke entri yang sesuai dalam setiap proses tabel sistem file. Semua operasi file kemudian dilakukan melalui penunjuk ini. Itu mengajukan nama mungkin bukan menjadi bagian dari itu membuka file meja, sebagai itu sistem memiliki TIDAK menggunakan untuk itu setelah FCB yang sesuai terletak pada disk. Namun, itu bisa di-cache menghemat waktu pada pembukaan berikutnya dari file yang sama. Nama yang diberikan pada entri tersebut bervariasi. Sistem UNIX menyebutnya sebagai **file deskripsi** ; Windows menyebutnya sebagai a **file menangani** .

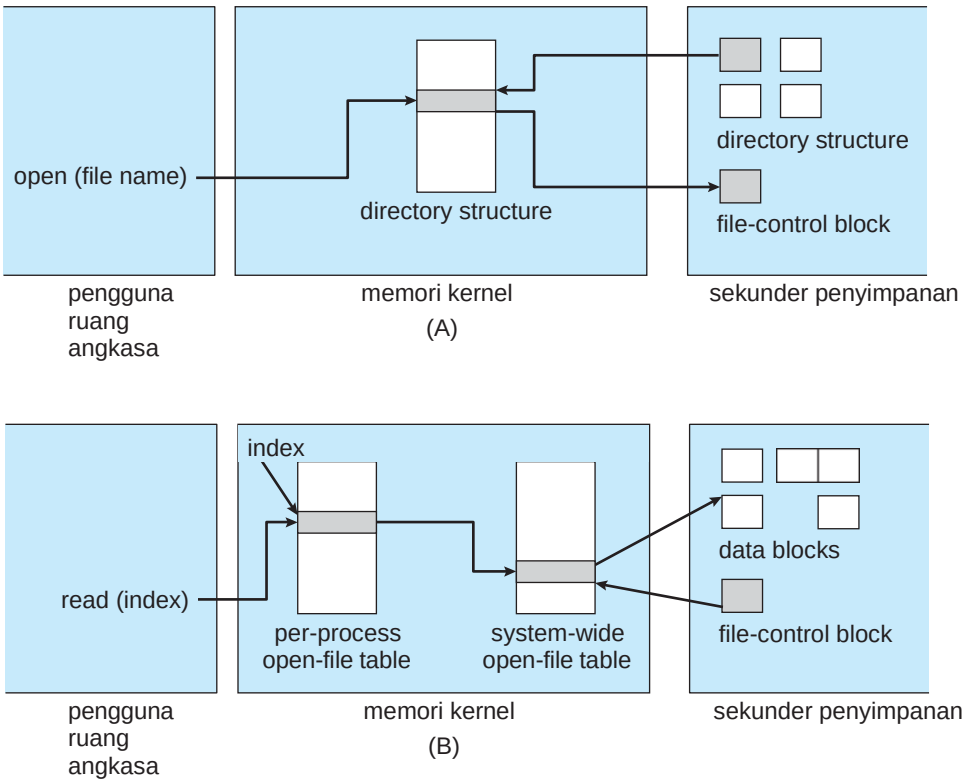
Kapan A proses menutup itu mengajukan, itu per proses meja pintu masuk adalah DIHAPUS, Dan jumlah entri yang terbuka di seluruh sistem dikurangi. Ketika semua pengguna yang memiliki membuka file, menutupnya, semua metadata yang diperbarui akan disalin kembali ke berbasis disk direktori struktur, Dan itu seluruh sistem membuka file meja pintu masuk adalah DIHAPUS.

Aspek caching pada struktur sistem file tidak boleh diabaikan. Paling sistem menyimpan semua informasi tentang sebuah membuka mengajukan, kecuali untuk -nya sebenarnya data blok, dalam memori. Sistem BSD UNIX tipikal dalam penggunaan cache dimanapun disk I/O dapat disimpan. Tingkat cache hit rata-rata sebesar 85 persen menunjukkan hal ini teknik ini layak untuk diterapkan. Sistem BSD UNIX dijelaskan sepenuhnya di dalam Lampiran C.

Itu Pengoperasian struktur dari A berkas sistem penerapan adalah diringkasdi dalam Angka 14.3.

14.3 Direktori Penerapan

Pemilihan algoritma alokasi direktori dan manajemen direktori secara signifikan mempengaruhi efisiensi, kinerja, dan keandalan sistem file item. Di dalam ini bagian, Kami membahas itu trade-off terlibat di dalam memilih satu dari inialgoritma.



Angka 14.3 Dalam kenangan berkas sistem struktur. (A) Mengajukan membuka. (B) Mengajukan membaca.

14.3.1 Linier Daftar

Metode paling sederhana dalam mengimplementasikan direktori adalah dengan menggunakan daftar file linier nama dengan petunjuk ke itu data blok. Ini metode adalah sederhana ke program tetapi memakan waktu untuk mengeksekusinya. Untuk membuat file baru, kita harus mencari terlebih dahulu direktori untuk memastikan tidak ada file yang memiliki nama yang sama. Lalu, kami menambahkan yang baru pintu masuk pada itu akhir dari itu direktori. Ke menghapus A mengajukan, Kami mencari itu direktori untuk itu bernama file dan kemudian lepaskan ruang yang dialokasikan untuk itu. Untuk menggunakan kembali direktori masuk, kita dapat melakukan salah satu dari beberapa hal. Kita dapat menandai entri tersebut sebagai tidak terpakai (oleh memberinya nama khusus, seperti nama yang kosong, memberinya nama yang tidak valid nomor inode (seperti 0), atau dengan menyertakan bit terpakai – tidak terpakai di setiap entri), atau kita dapat melampirkannya ke daftar entri direktori gratis. Alternatif ketiga adalah menyalin itu terakhir pintu masuk di dalam itu direktori ke dalam itu dibebaskan lokasi Dan ke mengurangi itu panjang dari direktori. Daftar tertaut juga dapat digunakan untuk mengurangi waktu yang diperlukan menghapus file.

Kerugian nyata dari daftar entri direktori linier adalah menemukan a file memerlukan pencarian linier. Informasi direktori sering digunakan, dan pengguna akan melihat jika mengakses ke dia adalah lambat. Di dalam fakta, banyak Pengoperasian sistem melaksanakan cache perangkat lunak untuk menyimpan informasi direktori yang terakhir digunakan. A cache

memukul menghindari itu membutuhkan ke selalu membaca kembali itu informasi dari sekunder penyimpanan. Daftar yang diurutkan memungkinkan pencarian biner dan mengurangi pencarian rata-rata waktu. Namun, persyaratan agar daftar tetap terurut dapat menyulitkan menciptakan Dan menghapus file, sejak Kami mungkin memiliki ke bergerak besar jumlah dari

informasi direktori untuk memelihara direktori yang diurutkan. Pohon yang lebih canggih struktur data, seperti pohon seimbang, mungkin membantu di sini. Sebuah keuntungan dari diurutkan daftar adalah itu A diurutkan direktori daftar Bisa menjadi diproduksi tanpa A memisahkan menyortir melangkah.

14.3.2 hash Meja

Struktur data lain yang digunakan untuk direktori file adalah tabel hash. Di sini, daftar linier menyimpan entri direktori, tetapi struktur data hash juga digunakan. Tabel hash mengambil nilai yang dihitung dari nama file dan mengembalikan pointer ke nama file dalam daftar linier. Oleh karena itu, ini dapat sangat mengurangi waktu pencarian direktori. Insersi Dan penghapusan adalah Juga secara adil mudah, meskipun beberapa persediaan harus dibuat untuk tabrakan — situasi di mana dua nama file di-hash ke sama lokasi.

Itu besar kesulitan dengan A hash meja adalah -nya umumnya tetap ukuran Dan itu ketergantungan fungsi hash pada ukuran itu. Misalnya, asumsikan bahwa kita membuat A penyelidikan linier hash meja itu memegang 64 entri. Itu hash fungsi menipu- ayat mengajukan nama ke dalam bilangan bulat dari 0 ke 63 (untuk contoh, oleh menggunakan itu sisa pembagian dengan 64). Jika nanti kita mencoba membuat file ke-65, kita harus memperbesar direktorinya. tabel hash tory— katakanlah, hingga 128 entri. Oleh karena itu, kita memerlukan fungsi hash baru yang harus memetakan nama file ke rentang 0 hingga 127, dan kita harus mengatur ulang yang ada direktori entri ke mencerminkan milik mereka baru nilai fungsi hash.

Sebagai alternatif, kita dapat menggunakan tabel hash chained-overflow. Setiap entri hash bisa berupa daftar tertaut, bukan nilai individual, dan kita bisa menyelesaikan konflik oleh menambahkan itu baru pintu masuk ke itu terhubung daftar. Pencarian mungkin menjadi agak melambat, karena mencari nama mungkin memerlukan penelusuran daftar tertaut entri tabel bertabrakan. Namun, metode ini mungkin jauh lebih cepat dibandingkan metode linier mencari melalui itu seluruh direktori.

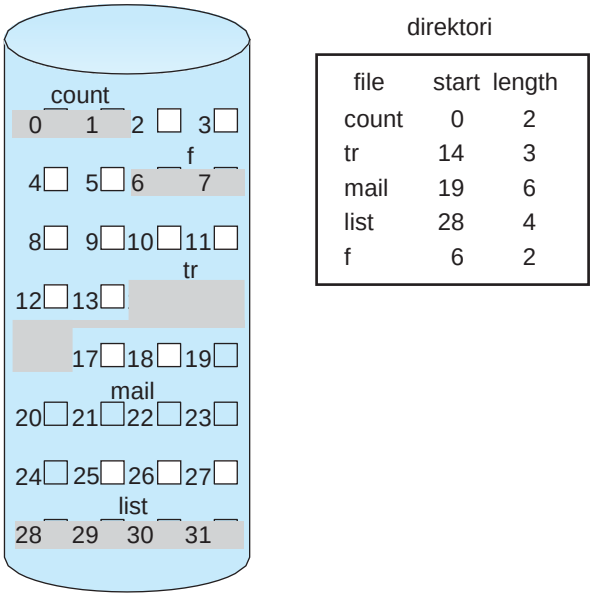
14.4 Alokasi Metode

Itu akses langsung alam dari sekunder penyimpanan memberi kita fleksibilitas di dalam itu menerapkan- penyebutan file. Di hampir setiap kasus, banyak file disimpan di tempat yang sama perangkat. Itu utama masalah adalah Bagaimana ke mengalokasikan ruang angkasa ke ini file Jadi itu penyimpanan ruang digunakan secara efektif dan file dapat diakses dengan cepat. Tiga metode utama- peluang dari mengalokasikan sekunder ruang penyimpanan adalah di dalam lebar menggunakan: bersebelahan, terhubung, dan diindeks. Masing-masing metode mempunyai kelebihan dan kekurangan. Meskipun beberapa sistem mendukung ketiganya, lebih umum bagi suatu sistem untuk menggunakan satu metode untuk semua file di dalam sebuah sistem file jenis.

14.4.1 Berdekatan Alokasi

Alokasi yang berdekatan mengharuskan setiap file menempati sekumpulan blok yang berdekatan pada itu perangkat. Perangkat alamat mendefinisikan A linier Memerintah pada itu perangkat. Dengan ini Memerintah, asumsi itu

hanya satu pekerjaan adalah mengakses itu perangkat, mengakses memblokir *B*
+1 setelah memblokir *B* biasanya memerlukan TIDAK kepala pergerakan. Kapan kepala pergerakan diperlukan (dari sektor terakhir dari satu silinder ke sektor pertama berikutnya silinder), itu kepala membutuhkan hanya bergerak dari satu melacak ke itu Berikutnya. Dengan demikian, untuk HDD , itu nomor dari disk mencari diperlukan untuk mengakses secara berdekatan dialokasikan file adalah



Angka 14.4 Berdekatan alokasi dari disk ruang angkasa.

minimal (asumsi blok dengan menutup logis alamat adalah menutup secara fisik), sebagai adalah mencari waktu Kapan sebuah pencarian adalah akhirnya dibutuhkan.

Alokasi file yang berdekatan ditentukan oleh alamat blok pertama dan panjang (dalam satuan blok) file. Jika file tersebut panjangnya n blok dan dimulai pada lokasi B, Kemudian dia menempati blok B, $B + 1$, $B + 2$, ..., $B + N - 1$. Itu direktori pintu masuk untuk setiap mengajukan menunjukkan itu alamat dari itu memulai memblokir Dan itu panjang dari itu daerah dialokasikan untuk ini mengajukan (Angka 14.4). Berdekatan alokasi adalah mudah ke melaksanakan Tetapi memiliki keterbatasan, Dan adalah Karena itu bukan digunakan di dalam modern mengajukan sistem.

Mengakses A mengajukan itu memiliki pernah dialokasikan secara berdekatan adalah mudah. Untuk sekuensial mengakses, itu mengajukan sistem ingat itu alamat dari itu terakhir memblokir dirujuk Dan, bila perlu, baca blok berikutnya. Untuk akses langsung ke blok i dari file itu dimulai dari blok b, kita bisa langsung mengakses blok $b + i$. Jadi, keduanya berurutan Dan langsung mengakses Bisa menjadi didukung oleh berdekatan alokasi.

Berdekatan alokasi memiliki beberapa masalah, Namun. Satu kesulitan adalah menemukan- mencari ruang untuk file baru. Sistem yang dipilih untuk mengelola ruang kosong menentukan bagaimana tugas ini diselesaikan; sistem manajemen ini dibahas dalam Bagian 14.5. Sistem manajemen apa pun dapat digunakan, namun ada pula yang lebih lambat yang lain.

Itu masalah alokasi bersebelahan Bisa menjadi terlihat sebagai A tertentu aplikasi dari masalah **alokasi penyimpanan dinamis umum** yang dibahas di Bagian 9.2, yang melibatkan bagaimana caranya memuaskan A meminta dari ukuran N dari A daftar gratis lubang. Pertama fit dan best fit adalah strategi paling umum yang digunakan untuk memilih hole bebas itu mengatur dari tersedia lubang. Simulasi memiliki ditampilkan itu keduanya Pertama bugar Dan terbaik bugar lebih efisien daripada yang terburuk dalam hal waktu dan pemanfaatan penyimpanan. Baik first fit maupun best fit jelas bukan yang

terbaik dalam hal pemanfaatan penyimpanan, namun yang pertama bugar umumnya lebih cepat.

Semua algoritma ini mengalami masalah **fragmentasi eksternal** . Sebagai file adalah dialokasikan Dan dihapus, itu bebas penyimpanan ruang angkasa adalah rusak ke dalam kecil bagian-bagian.

Fragmentasi eksternal terjadi setiap kali ruang kosong dipecah menjadi beberapa bagian. Dia menjadi masalah ketika potongan terbesar yang berdekatan tidak cukup untuk a meminta; penyimpanan terfragmentasi menjadi beberapa lubang, tidak ada satupun yang berukuran besar cukup untuk menyimpan data. Tergantung pada jumlah total penyimpanan disk dan rata-rata mengajukan ukuran, luar fragmentasi mungkin menjadi A minor atau A besar masalah.

Satu strategi untuk mencegah kehilangan dari penting jumlah dari penyimpanan ruang angkasa ke fragmentasi eksternal adalah menyalin seluruh sistem file ke perangkat lain. Perangkat asli kemudian dibebaskan sepenuhnya, menciptakan satu perangkat besar yang bersebelahan ruang bebas. Kami kemudian menyalin file kembali ke perangkat asli dengan mengalokasikan ruang yang berdekatan dari satu lubang besar ini. Skema ini secara efektif **memadatkan** semua ruang bebas menjadi satu ruang yang berdekatan, memecahkan masalah fragmentasi. Namun, biaya pemadatan ini adalah waktu, dan biayanya bisa sangat besar tinggi untuk perangkat penyimpanan besar. Memadatkan perangkat ini mungkin memerlukan waktu berjam-jam dan mungkin diperlukan setiap minggu. Beberapa sistem memerlukan fungsi ini dilakukan **secara offline** , dengan sistem file dilepas. Selama **waktu senggang ini** , pengoperasian sistem normal umumnya tidak diperbolehkan, sehingga pemadatan seperti itu diperbolehkan dihindari dengan segala cara pada mesin produksi. Kebanyakan sistem modern yang membutuhkan defragmentasi Bisa melakukan dia **on line** selama normal sistem operasi, Tetapiitu pertunjukan penalti Bisa menjadi besar.

Masalah lain dengan alokasi berdekatan adalah menentukan jumlahnya ruang angkasa adalah diperlukan untuk A mengajukan. Kapan itu mengajukan adalah dibuat, itu total jumlah dari ruang angkasa kebutuhannya harus ditemukan dan dialokasikan. Bagaimana cara pembuatnya (memprogram atau orang) mengetahui ukuran file yang akan dibuat? Dalam beberapa kasus, penentuan ini Tindakannya mungkin cukup sederhana (menyalin file yang sudah ada, misalnya). Secara umum, Namun, itu ukuran dari sebuah keluaran mengajukan mungkin menjadi sulit ke memperkirakan.

Jika Kami mengalokasikan juga kecil ruang angkasa ke A mengajukan, Kami mungkin menemukan itu itu mengajukan tidak bisa diperpanjang. Terutama dengan strategi alokasi yang paling sesuai, ruang pada keduanya sisi file mungkin sedang digunakan. Oleh karena itu, kami tidak dapat memperbesar file pada tempatnya. Ada dua kemungkinan. Pertama, program pengguna dapat dihentikan, dengan pesan kesalahan yang sesuai. Pengguna kemudian harus mengalokasikan lebih banyak ruang dan jalankan programnya lagi. Pengoperasian yang berulang-ulang ini mungkin memakan biaya. Untuk mencegahnya, pengguna biasanya akan melebih-lebihkan jumlah ruang yang dibutuhkan, sehingga mengakibatkan banyak ruang terbuang. Kemungkinan lainnya adalah menemukan lubang yang lebih besar, salin isi file ke ruang baru, dan lepaskan ruang sebelumnya. Ini serangkaian tindakan dapat diulangi selama masih ada ruang, meskipun bisa saja terjadi memakan waktu. Pengguna tidak perlu diberitahu secara eksplisit tentang apa yang ada namun terjadi; sistem terus berlanjut meskipun ada masalah, meskipun lebih banyak Dan lagi perlahan-lahan.

Meskipun jumlah total ruang yang dibutuhkan untuk suatu file diketahui sebelumnya, praalokasi mungkin tidak efisien. File yang akan tumbuh perlahan dalam jangka waktu lama (bulan atau tahun) harus diberi ruang yang cukup untuk ukuran akhirnya banyak dari itu ruang angkasa akan menjadi tidak terpakai untuk A panjang waktu. Itu mengajukan Karena itu

memiliki A besarjumlah dari internal fragmentasi.

Untuk meminimalisir kekurangan tersebut, suatu sistem operasi dapat menggunakan modifikasi skema alokasi bersebelahan. Di sini, sejumlah ruang yang berdekatan dialokasikan mulanya. Kemudian, jika itu jumlah membuktikan bukan ke menjadi besar cukup, lain bingkah dari berdekatan ruang angkasa, diketahui sebagai sebuah **cakupan** , adalah ditambahkan. Itu lokasi dari A file blok kemudian dicatat sebagai lokasi dan jumlah blok, ditambah link ke blok pertama tingkat selanjutnya. Pada beberapa sistem, pemilik file dapat mengatur luasnya ukuran, Tetapi ini pengaturan hasil di dalam inefisiensi jika itu pemilik adalah salah. Intern

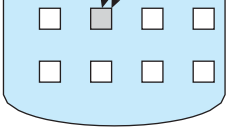
fragmentasi masih bisa menjadi masalah jika cakupannya terlalu besar dan bersifat eksternal fragmentasi dapat menjadi masalah karena luasan dengan ukuran yang berbeda-beda dialokasikan Dan tidak dialokasikan. Itu komersial Symantec kebenaran mengajukan sistem kegunaan luasnya untuk mengoptimalkan kinerja. Veritas adalah pengganti berkinerja tinggi untuk standar UNIX UFS .

14.4.2 Tertaut Alokasi

Alokasi tertaut menyelesaikan semua masalah alokasi yang berdekatan. Dengan tertaut alokasi, setiap file adalah daftar tertaut dari blok penyimpanan; bloknya mungkin tersebar di mana pun di perangkat. Direktori berisi penunjuk ke yang pertama dan terakhir blok file. Misalnya, file dengan lima blok mungkin dimulai pada blok 9 dan Lanjutkan di blok 16, lalu blok 1, lalu blok 10, dan terakhir blok 25 (Gambar 2). 14.5). Setiap blok berisi pointer ke blok berikutnya. Petunjuk ini tidak dibuat tersedia ke itu pengguna. Dengan demikian, jika setiap memblokir adalah 512 byte di dalam ukuran, Dan A memblokir alamat (itu penunjuk) memerlukan 4 byte, Kemudian itu pengguna melihat blok dari 508 byte.

Untuk membuat file baru, kita cukup membuat entri baru di direktori. Dengan alokasi tertaut, setiap entri direktori memiliki penunjuk ke blok pertama mengajukan. Penunjuk ini diinisialisasi ke null (nilai penunjuk akhir daftar) untuk menandakan sebuah berkas kosong. Bidang ukuran juga diatur ke 0. Penulisan ke file menyebabkan kebebasan sistem manajemen ruang untuk menemukan blok gratis, dan blok baru ini ditulis ke dan ditautkan ke akhir file. Untuk membaca file, kita cukup membaca blok demi blok mengikuti itu petunjuk dari memblokir ke memblokir. Di sana adalah TIDAK luar fragmentasi dengan terhubung alokasi, Dan setiap bebas memblokir pada itu ruang bebas daftar Bisa menjadi digunakan ke memuaskan A meminta. Itu ukuran dari A mengajukan membutuhkan bukan menjadi dideklarasikan Kapan itu mengajukan adalah dibuat. A berkas bisa melanjutkan ke tumbuh selama bebas blok tersedia. Akibatnya, dia adalah tidak pernah diperlukan ke kompak disk ruang angkasa.

Namun, alokasi tertaut mempunyai kelemahan. Masalah utama adalah itu dia Bisa menjadi digunakan secara efektif hanya untuk akses berurutan file. Ke menemukan itu *saya* th



Angka 14.5 Tertaut alokasi dari disk ruang angkasa.

blok suatu file, kita harus mulai dari awal file itu dan mengikuti petunjuknya sampai Kami mendapatkan ke itu *saya* th memblokir. Setiap mengakses ke A penunjuk memerlukan A penyimpanan perangkat membaca, Dan beberapa memerlukan sebuah HDD mencari. Akibatnya, dia adalah tidak efisien ke mendukungA akses langsung kemampuan untuk alokasi tertaut file.

Kerugian lainnya adalah ruang yang dibutuhkan untuk pointer. Jika sebuah penunjuk memerlukan 4 byte dari blok 512 byte, maka 0,78 persen disk sedang digunakan digunakan untuk petunjuk, lebih tepatnya dibandingkan untuk informasi. Setiap mengajukan memerlukan agak lagi ruang angkasa daripada itu akan sebaliknya.

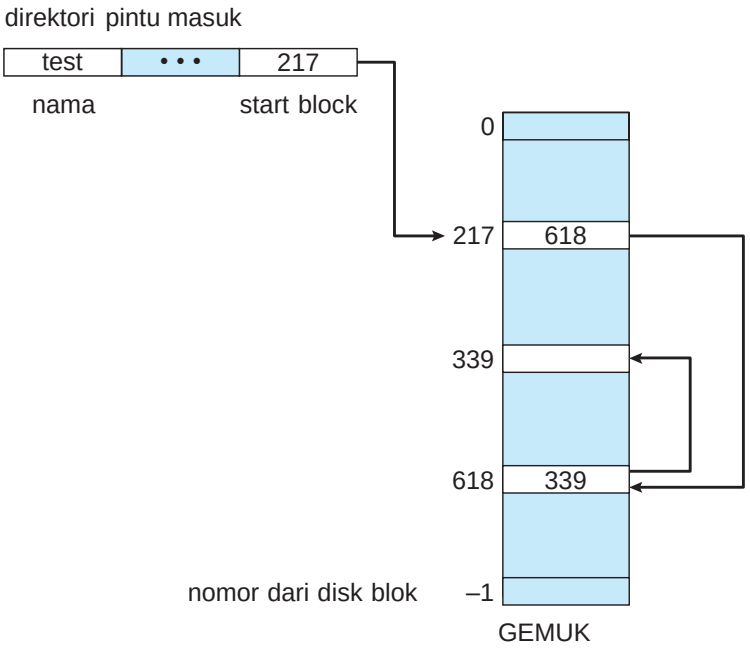
Itu biasa larutan ke ini masalah adalah ke mengumpulkan blok ke dalam kelipatan, ditelepon **cluster** , dan untuk mengalokasikan cluster daripada blok. Misalnya, sistem file dapat mendefinisikan cluster sebagai empat blok dan beroperasi pada penyimpanan sekunder perangkat hanya di unit cluster. Pointer kemudian menggunakan persentase yang jauh lebih kecil ruang file. Metode ini memungkinkan pemetaan blok logis-ke-fisik tetap sederhana Tetapi membaik HDD keluaran (Karena lebih sedikit kepala disk mencari adalah diperlukan) Dan berkurang itu ruang angkasa diperlukan untuk memblokir alokasi Dan daftar gratis pengelolaan. Kerugian dari pendekatan ini adalah peningkatan fragmentasi internal, Karena lagi ruang angkasa adalah sia-sia Kapan A gugus adalah sebagian penuh dibandingkan Kapan A memblokir sebagian penuh. Kinerja I/O acak juga terganggu karena permintaan untuk a kecil jumlah dari data transfer A besar jumlah dari data. Cluster Bisa menjadi digunakan ke memperbaiki itu akses disk waktu untuk banyak lainnya algoritma sebagai Sehat, Jadi mereka adalah digunakan di dalam paling mengajukan sistem.

Masalah lain dari alokasi terkait adalah keandalan. Ingatlah bahwa file adalah terhubung bersama oleh petunjuk berserakan semua lebih itu perangkat, Dan mempertimbangkan Apa akan terjadi jika pointer hilang atau rusak. Bug di sistem operasi kegagalan perangkat lunak atau perangkat keras dapat mengakibatkan pengambilan penunjuk yang salah. Ini kesalahan bisa di dalam berbelok hasil di dalam menghubungkan ke dalam itu ruang bebas daftar atau ke dalam lain mengajukan. Salah satu solusi parsial adalah dengan menggunakan daftar tertaut ganda, dan solusi lainnya adalah dengan menyimpan itu mengajukan nama Dan relatif memblokir nomor di dalam setiap memblokir. Namun, ini skema memerlukan bahkan lagi atas untuk setiap mengajukan.

Sebuah penting variasi pada terhubung alokasi adalah itu menggunakan dari A **alokasi file meja** (**GEMUK**). Ini sederhana Tetapi efisien metode dari ruang disk alokasi dulu digunakanoleh MS-DOS sistem operasi. A bagian dari penyimpanan pada awalnya setiap volume adalah mengatur ke samping ke berisi itu meja. Itu meja memiliki satu pintu masuk untuk setiap memblokir Dan adalah diindeks oleh memblokir nomor. Itu GEMUK adalah digunakan di dalam banyak itu sama jalan sebagaiA terhubung daftar. Itu direktori pintu masuk mengandung itu memblokir nomor dari itu Pertama memblokirdari itu mengajukan. Itu meja pintu masuk diindeks oleh itu memblokir nomor mengandung itu memblokirnomor dari itu Berikutnya memblokir di dalam itu mengajukan. Ini rantai berlanjut sampai dia mencapai itu terakhir memblokir, yang memiliki A spesial akhir file nilai sebagai itu meja pintu masuk. Sebuah tidak terpakai memblokir adalah ditunjukkan oleh A meja nilai dari 0. Mengalokasikan A baru memblokir ke A mengajukan adalah A sederhanaurusan dari temuan itu Pertama bernilai 0 meja pintu masuk Dan menggantikan itu sebelumnya akhir-dari file nilai dengan itu alamat dari itu baru memblokir. Itu 0 adalah Kemudian diganti dengan itu

akhir file nilai. Sebuah ilustratif contoh adalah itu GEMUK struktur ditampilkan di dalam Angka 14.6 untuk A mengajukan terdiri dari disk blok 217, 618, Dan 339.

Skema alokasi FAT dapat menghasilkan sejumlah besar head disk mencari, kecuali FAT di-cache. Kepala disk harus berpindah ke awal volume untuk membaca FAT dan menemukan lokasi blok yang dimaksud bergerak ke itu lokasi dari itu memblokir diri. Di dalam itu terburuk kasus, keduanya bergerak terjadi untuk masing-masing blok. Manfaatnya adalah waktu akses acak ditingkatkan, karena kepala disk dapat menemukan lokasi blok mana pun dengan membaca informasi di dalamnyaitu GEMUK .



Angka 14.6 Alokasi file meja.

14.4.3 Terindeks Alokasi

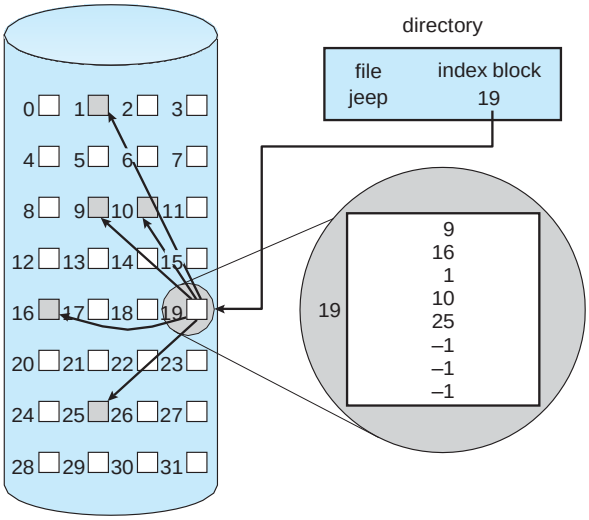
Tertaut alokasi memecahkan itu fragmentasi eksternal Dan deklarasi ukuran masalah- lem dari berdekatan alokasi. Namun, di dalam itu ketiadaan dari A GEMUK , terhubung mengalokasikan- tion tidak dapat mendukung akses langsung yang efisien, karena penunjuk ke blok tersebut adalah tersebar bersama blok-blok itu sendiri di seluruh disk dan harus diambil dalam urutan. **Alokasi terindeks** menyelesaikan masalah ini dengan memberikan semua petunjuk bersama ke dalam satu lokasi: itu **indeks memblokir** .

Setiap file memiliki blok indeksnya sendiri, yang merupakan array blok penyimpanan alamat. Entri *ke-i* di blok indeks menunjuk ke blok *ke-i* dari file tersebut. Direktori berisi alamat blok indeks (Gambar 14.7). Untuk menemukan dan membaca itu *saya* th memblokir, Kami menggunakan itu penunjuk di dalam itu *saya* th blok indeks pintu masuk. Ini skema adalah serupa ke pagingnya skema dijelaskan di dalam Bagian 9.3.

Saat file dibuat, semua pointer di blok indeks disetel ke null. Ketika blok *ke i* pertama kali ditulis, sebuah blok diperoleh dari ruang bebas Pengelola, Dan -nya alamat adalah meletakkan di dalam itu *saya* th blok indeks pintu masuk.

Terindeks alokasi mendukung langsung mengakses, tanpa menderita dari luar fragmentasi, karena setiap blok bebas pada perangkat penyimpanan dapat memenuhi a meminta lebih banyak ruang. Alokasi yang diindeks memang mengalami pemborosan ruang, Namun. Itu penunjuk atas dari itu indeks memblokir adalah umumnya lebih besar dibandingkan itu overhead penunjuk dari alokasi tertaut. Perhatikan kasus umum yang kita alami memiliki A mengajukan dari hanya satu atau dua blok. Dengan terhubung alokasi, Kami kehilangan itu ruang angkasa hanya satu pointer per blok. Dengan alokasi yang diindeks, seluruh blok indeks harus menjadi dialokasikan, bahkan jika hanya satu atau dua petunjuk akan menjadi bukan- batal.

Ini titik meningkatkan itu pertanyaan dari Bagaimana besar itu indeks memblokir sebaiknya menjadi. Setiap mengajukan harus memiliki sebuah indeks memblokir, Jadi Kami ingin itu indeks memblokir ke menjadi sebagai kecil sebagai



Angka 14.7 Terindeks alokasi disk ruang angkasa.

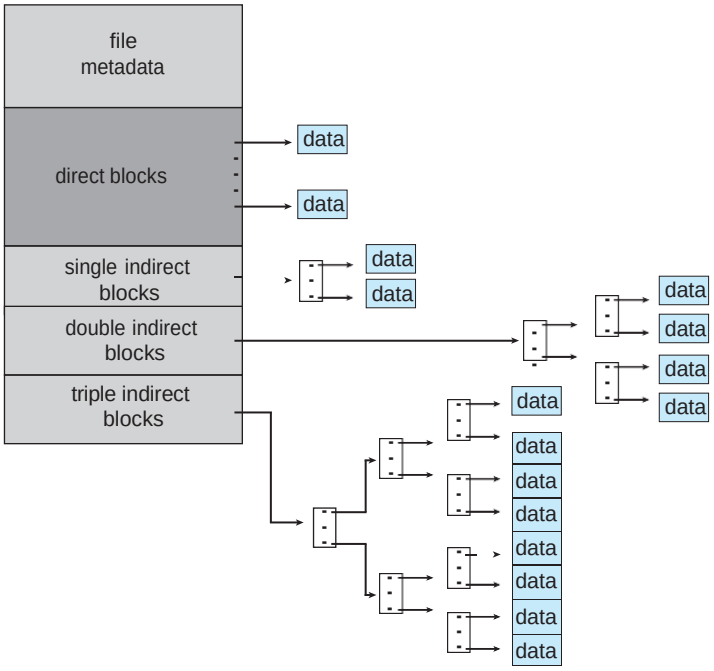
mungkin. Namun, jika blok indeks terlalu kecil, maka blok tersebut tidak akan mampu bertahan petunjuk yang cukup untuk file besar, dan mekanisme harus tersedia untuk itu kesepakatan dengan ini masalah. Mekanisme untuk ini tujuan termasuk itu mengikuti:

- **Skema terkait** . Blok indeks biasanya merupakan satu blok penyimpanan. Jadi, itu bisa dibaca dan ditulis langsung dengan sendirinya. Untuk memungkinkan file berukuran besar, kita dapat menautkan bersama beberapa indeks blok. Untuk contoh, sebuah indeks memblokir mungkin berisi A kecil tajuk memberi itu nama dari itu mengajukan Dan A mengatur dari itu Pertama 100 alamat blok disk . Alamat berikutnya (kata terakhir di blok indeks) adalah null (untuk A kecil mengajukan) atau adalah A penunjuk ke lain indeks memblokir (untuk A besar mengajukan).

- **Bertingkat indeks** . menyimpang dari terhubung perwakilan kegunaan A tingkat pertama indeks memblokir ke titik ke A mengatur dari tingkat kedua indeks blok, yang di dalam berbelok titik ke file tersebut diblokir. Untuk mengakses suatu blok, sistem operasi menggunakan level pertama indeks ke menemukan A tingkat kedua indeks memblokir Dan Kemudian kegunaan itu memblokir ke menemukan itu blok data yang diinginkan. Pendekatan ini bisa dilanjutkan ke yang ketiga atau keempat level, tergantung pada ukuran file maksimum yang diinginkan. Dengan blok 4.096 byte, kita dapat menyimpan 1.024 pointer empat byte dalam satu blok indeks. Dua tingkat indeks mengizinkan 1.048.576 data blok Dan A mengajukan ukuran dari ke atas ke 4 GB .

- **Gabungan skema** . Lain alternatif, digunakan di dalam berbasis UNIX mengajukan sistem, adalah untuk menyimpan yang pertama, katakanlah, 15 pointer dari blok indeks di inode file. 12 petunjuk pertama menunjuk ke **blok langsung** ; artinya, mereka mengandung alamat blok yang berisi data file. Jadi, datanya kecil file (dari TIDAK lagi dibandingkan 12 blok) Mengerjakan bukan membutuhkan A memisahkan indeks memblokir. Jika itu memblokir ukuran adalah 4 KB , Kemudian ke atas ke 48 KB dari data Bisa menjadi diakses secara langsung. Itu Berikutnya tiga petunjuk titik ke **tidak langsung blok** . Itu Pertama poin ke A **lajang tidak**

langsung blok , yang merupakan blok indeks yang tidak berisi data tetapi alamatnya blok itu Mengerjakan berisi data. Itu Kedua poin ke A **dobel tidak langsung memblokir** , yang mengandung itu alamat dari A memblokir itu mengandung itu alamat dari blok yang berisi pointer ke blok data aktual. Pointer terakhir berisi alamat dari A **tiga kali lipat tidak langsung memblokir** . (A UNIX inode adalah ditampilkan di dalam Angka 14.8.)



Angka 14.8 Itu UNIX inode.

Di bawah ini metode, itu nomor dari blok itu Bisa menjadi dialokasikan ke A mengajukan melebihi jumlah ruang yang dapat dialamatkan oleh penunjuk file 4-byte yang digunakan oleh banyak sistem operasi. Penunjuk file 32-bit hanya mencapai 2^{32} byte, atau 4GB . Banyak implementasi UNIX dan Linux sekarang mendukung file 64-bit pointer, yang memungkinkan file dan sistem file berukuran beberapa exbibytes. Itu ZFS mengajukan sistem mendukung 128-bit mengajukan petunjuk.

Skema alokasi terindeks mempunyai kinerja yang sama masalah sebagai melakukan terhubung alokasi. Secara khusus, itu indeks blok Bisa menjadi di-cache di dalam Penyimpanan, Tetapi itu data blok mungkin menjadi menyebar semua lebih A volume.

14.4.4 Pertunjukan

Itu alokasi metode itu Kami memiliki dibahas bervariasi di dalam milik mereka penyimpanan efisiensi Dan blok data mengakses waktu. Keduanya adalah penting kriteria di dalam memilih itu sesuai metode atau metode untuk sebuah Pengoperasian sistem untuk melaksanakan.

Sebelum memilih metode alokasi, kita perlu menentukan bagaimana caranya sistem akan digunakan. Sistem dengan sebagian besar akses berurutan sebaiknya tidak digunakan itu sama metode sebagai sistem dengan sebagian besar acak mengakses.

Untuk semua jenis akses, alokasi berdekatan hanya memerlukan satu akses mendapatkan satu blok. Karena kita dapat dengan mudah menyimpan alamat awal file di memori, kita dapat segera menghitung alamat blok ke *i* (atau blok berikutnya) Dan membaca dia secara langsung.

Untuk alokasi tertaut, kita juga dapat menyimpan alamat blok berikutnya memori dan membacanya secara langsung. Metode ini bagus untuk

akses berurutan; untuk langsung mengakses, Namun, sebuah mengakses ke itu *saya* th memblokir mungkin memerlukan *Saya* memblokir membaca. Ini

masalah menunjukkan Mengapa terhubung alokasi sebaiknya bukan menjadi digunakan untuk sebuah aplikasimembutuhkan langsung mengakses.

Akibatnya, beberapa sistem mendukung file akses langsung dengan menggunakan file yang berdekatan alokasi dan file akses berurutan dengan menggunakan alokasi tertaut. Untuk ini sistem, jenis akses yang akan dibuat harus dideklarasikan ketika file dibuat. File yang dibuat untuk akses berurutan akan ditautkan dan tidak dapat digunakan secara langsung mengakses. File yang dibuat untuk akses langsung akan bersebelahan dan dapat mendukung keduanya akses langsung dan akses sekuensial, tetapi panjang maksimumnya harus dinyatakan ketika itu dibuat. Dalam hal ini, sistem operasi harus sesuai struktur data dan algoritma untuk mendukung kedua metode alokasi. File bisa dikonversi dari satu jenis ke lain oleh itu penciptaan dari A baru mengajukan dari itu diinginkan ketik, ke mana isi file lama disalin. File lama mungkin menjadi dihapus dan itu baru mengajukan diganti namanya.

Terindeks alokasi lebih kompleks. Jika blok indeks sudah di dalam memmaaf, Kemudian itu mengakses Bisa menjadi dibuat secara langsung. Namun, penyimpanan itu indeks memblokirdi dalam Penyimpanan memerlukan besar ruang angkasa. Jika ini Penyimpanan ruang angkasa adalah bukan faedah-mampu, Kemudian Kami mungkin memiliki ke membaca Pertama itu indeks memblokir Dan Kemudian itu diinginkan data memblokir. Untuk A dua tingkat indeks, dua blok indeks membaca mungkin menjadi diperlukan.Untuk sebuah sangat besar mengajukan, mengakses A memblokir di dekat itu akhir dari itu mengajukan akan memerlukan membaca di dalam semua itu indeks blok sebelum itu diperlukan data memblokir Akhirnyabisa menjadi membaca. Dengan demikian, itu pertunjukan dari diindeks alokasi bergantung pada ituindeks struktur, pada itu ukuran dari itu mengajukan, Dan pada itu posisi dari itu memblokir diinginkan.Beberapa sistem menggabungkan berdekatan alokasi dengan diindeks alokasi oleh menggunakan berdekatan alokasi untuk kecil file (ke atas ke tiga atau empat blok) Dan mobil- secara matematis beralih ke sebuah diindeks alokasi jika itu mengajukan tumbuh besar. Sejak palingfile adalah kecil, Dan berdekatan alokasi adalah efisien untuk kecil file, rata-rata pertunjukan Bisa menjadi lumayan Bagus.

Banyak pengoptimalan lain yang sedang digunakan. Mengingat perbedaan antar CPU kecepatan Dan disk kecepatan, dia adalah bukan keterlaluan ke menambahkan ribuan dari tambahan instruksi- tions ke sistem operasi untuk menyimpan hanya beberapa pergerakan disk-head. Buluterlebih lagi, kesenjangan ini semakin meningkat dari waktu ke waktu, hingga mencapai ratusan dari ribuan dari instruksi bisa secara wajar menjadi digunakan ke mengoptimalkan kepala bergerak- catatan.

Untuk perangkat NVM , tidak ada pencarian head disk, jadi algoritmanya berbeda dan optimasi diperlukan. Menggunakan algoritma lama yang menghabiskan banyak CPU siklus mencoba ke menghindari A tidak ada kepala pergerakan akan menjadi sangat tidak efisien. Sistem file yang ada sedang dimodifikasi dan sistem file baru sedang dibuat untuk mencapainya kinerja maksimum dari perangkat penyimpanan NVM . Tujuan perkembangan ini untuk mengurangi jumlah instruksi dan jalur keseluruhan antar perangkat penyimpanan Dan aplikasi mengakses ke data.

14.5 Ruang bebas Pengelolaan

Karena ruang penyimpanan terbatas, kita perlu menggunakan kembali ruang dari file yang terhapus untuk baru file, jika mungkin. (Tulis sekali optik disk mengizinkan hanya satu menulis ke setiap sektor tertentu, dan dengan demikian penggunaan kembali secara fisik tidak mungkin dilakukan.) Untuk melacak secara gratis ruang disk, sistem mempertahankan **daftar ruang kosong** . Catatan daftar ruang bebas semua blok perangkat gratis — yang tidak dialokasikan ke beberapa file atau direktori. Untuk membuat mengajukan, Kami mencari itu ruang bebas daftar untuk itu diperlukan jumlah dari ruang angkasa Dan mengalokasikan

ruang itu ke file baru. Ruang ini kemudian dihapus dari daftar ruang kosong. Saat file dihapus, ruangnya ditambahkan ke daftar ruang kosong. Ruang bebas list, terlepas dari namanya, belum tentu diimplementasikan sebagai sebuah daftar, seperti yang kita bahas Berikutnya.

14.5.1 Sedikit Vektor

Seringkali, daftar ruang kosong diimplementasikan sebagai **bitmap** atau **bit vektor** . Setiap blok diwakili oleh 1 bit. Jika bloknya bebas, bitnya adalah 1; jika bloknya dialokasikan, itu sedikit adalah 0.

Misalnya, bayangkan sebuah disk dengan blok 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, Dan 27 adalah bebas Dan itu istirahat dari itu blok adalah dialokasikan. Itu bitmap ruang bebas akan menjadi

0011110011111100011000000011100000 ...

Keuntungan utama dari pendekatan ini adalah kesederhanaannya yang relatif dan efisiensinya. efisiensi dalam menemukan blok kosong pertama atau *n* blok kosong berturut-turut pada disk. Memang benar, banyak komputer menyediakan instruksi manipulasi bit yang dapat digunakan secara efektif untuk itu tujuan. Satu teknik untuk temuan itu Pertama bebas memblokir pada sebuah sistem yang menggunakan vektor bit untuk mengalokasikan ruang adalah dengan memeriksa masing-masing secara berurutan kata dalam bitmap untuk melihat apakah nilainya bukan 0, karena kata bernilai 0 hanya berisi 0 bit dan mewakili sekumpulan blok yang dialokasikan. Yang pertama bukan-0 kata adalah dipindai untuk itu Pertama 1 sedikit, yang adalah itu lokasi dari itu Pertama bebas memblokir.
Itu perhitungan dari itu memblokir nomor adalah

(jumlah bit per kata) × (jumlah kata bernilai 0) + offset 1 bit pertama. Lagi,

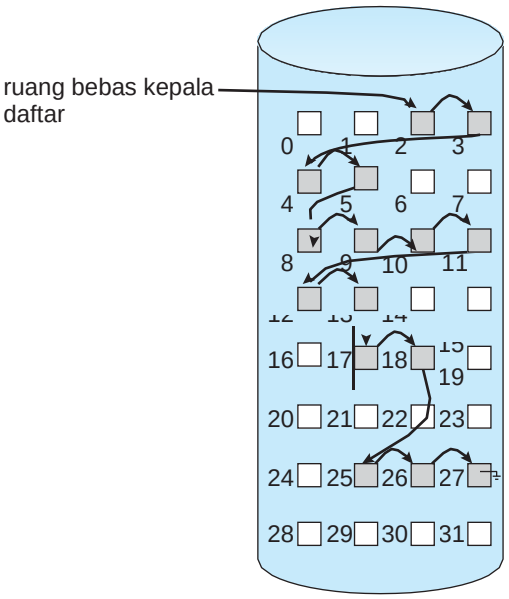
Kami melihat perangkat keras fitur menyetir perangkat lunak Kegunaan.

Sayangnya-
Akhirnya, vektor bit tidak efisien kecuali seluruh vektor disimpan di memori utama (dan kadang-kadang ditulis ke perangkat yang berisi sistem file untuk dipulihkan kebutuhan). Menyimpannya di memori utama dimungkinkan untuk perangkat yang lebih kecil tetapi tidak tentu saja untuk yang lebih besar. Disk berukuran 1,3 GB dengan blok 512 byte memerlukan a bitmap dari lebih 332 KB ke melacak -nya bebas blok, meskipun kekelompokan itu blok di dalam grup beranggotakan empat orang mengurangi jumlah ini menjadi sekitar 83 KB per disk. Disk 1 TB dengan 4- Blok KB memerlukan 32 MB ($2^{40} / 2^{12} = 2^{28}$ bit = 2^{25} byte = 2^{5} MB) sampai toko -nya bitmap. Diberikan itu disk ukuran selalu meningkat, itu masalah dengan sedikit vektor akan terus meningkat sebagai Sehat.

14.5.2 Tertaut Daftar

Pendekatan lain terhadap pengelolaan ruang bebas adalah dengan menghubungkan semua ruang bebas blok, menyimpan penunjuk ke blok bebas pertama di lokasi khusus dalam file sistem dan menyimpannya di memori. Blok pertama ini berisi pointer ke Berikutnya bebas memblokir, Dan Jadi pada. Mengingat kita lebih awal contoh (Bagian 14.5.1), di dalam yang blok 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, Dan 27 adalah bebas Dan itu istirahat dari itu blok itu dialokasikan. Di dalam ini situasi, kita akan menyimpan A penunjuk ke memblokir 2 sebagai itu Pertama bebas

memblokir. Memblokir 2 akan berisi A penunjuk ke memblokir 3, yang akan menunjuk ke blok 4, yang akan menunjuk ke blok 5, yang akan menunjuk ke blok 8, dan seterusnya (Gambar 14.9). Skema ini tidak efisien; untuk melintasi list, kita harus membaca setiap blok, yang memerlukan waktu I/O yang besar pada HDD . Untung, Namun, melintasi itu bebas daftar adalah bukan A sering tindakan. Biasanya,



Angka 14.9 Tertaut daftar ruang bebas pada disk.

itu Pengoperasian sistem secara sederhana kebutuhan A bebas memblokir Jadi itu dia Bisa mengalokasikan itu memblokir ke suatu file, jadi blok pertama dalam daftar gratis digunakan. Metode FAT menggabungkan blok bebas akuntansi ke dalam itu alokasi data struktur. TIDAK memisahkan metode adalahdiperlukan.

14.5.3 Pengelompokan

Modifikasi pendekatan daftar bebas menyimpan alamat dari n blok bebas di blok bebas pertama. $N - 1$ blok pertama sebenarnya gratis. Yang terakhir memblokir mengandung itu alamat dari lain N bebas blok, Dan Jadi pada. Itu alamat dari A besar nomor dari bebas blok Bisa Sekarang menjadi ditemukan dengan cepat, tidak seperti itu situasi Kapan itu standar daftar tertaut mendekati adalah digunakan.

14.5.4 Perhitungan

Lain mendekati dibutuhkan keuntungan dari itu fakta itu, umumnya, beberapa bersebelahan- ous blok mungkin menjadi dialokasikan atau dibebaskan serentak, khususnya Kapan ruang angkasa dialokasikan dengan algoritma alokasi bersebelahan atau melalui pengelompokan. Jadi, daripada menyimpan daftar n alamat blok gratis, kita dapat menyimpan alamat dari itu Pertama bebas memblokir Dan itu nomor (N) dari bebas berdekatan blok itu mengikuti itu Pertama memblokir. Setiap pintu masuk di dalam itu ruang bebas daftar Kemudian terdiri dari A perangkat alamat dan hitungan. Meskipun setiap entri memerlukan lebih banyak ruang daripada a sederhana disk alamat, itu keseluruhan daftar adalah singkat, sebagai panjang sebagai itu menghitung adalah umumnya lebih besar dari 1. Perhatikan bahwa metode pelacakan ruang kosong ini mirip dengan sejauh mana metode mengalokasikan blok. Entri-entri ini dapat disimpan secara

seimbang pohon, lebih tepatnya dibandingkan A terhubung daftar, untuk efisien Lihatlah, insersi, Dan penghapusan.

14.5.5 Ruang angkasa Peta

ZFS Oracle (ditemukan di Solaris dan beberapa sistem operasi lainnya) dirancang untuk mencakup sejumlah besar file, direktori, dan bahkan file sistem (di ZFS , kita dapat membuat hierarki sistem file). Pada skala ini, meta-data I/O dapat memiliki dampak kinerja yang besar. Misalnya, jika itu ruang bebas daftar adalah dilaksanakan sebagai A bitmap, bitmap harus menjadi diubah keduanya kapan blok dialokasikan dan kapan dibebaskan. Membebaskan 1 GB data pada a Disk 1 TB dapat menyebabkan ribuan blok bitmap diperbarui, karena blok data tersebut dapat tersebar di seluruh disk. Yang jelas datanya struktur untuk seperti A sistem bisa menjadi besar dan tidak efisien.

Dalam pengelolaan ruang kosongnya, ZFS menggunakan kombinasi teknik untuk mengontrol ukuran struktur data dan meminimalkan I/O yang diperlukan untuk mengelola itu struktur. Pertama, ZFS menciptakan **metaslab** ke membagi itu ruang angkasa pada itu perangkat menjadi potongan-potongan dengan ukuran yang bisa diatur. Volume tertentu mungkin berisi ratusan metaslab. Setiap metaslab memiliki peta ruang terkait. ZFS menggunakan penghitungan algoritma untuk menyimpan informasi tentang blok bebas. Daripada menulis penghitungan struktur ke disk, dia kegunaan terstruktur log berkas sistem teknik ke catatan mereka. Peta ruang angkasa adalah log dari semua aktivitas blok (mengalokasikan dan membebaskan), dalam waktu memesan, dalam format penghitungan. Ketika ZFS memutuskan untuk mengalokasikan atau mengosongkan ruang dari a metaslab, itu memuat peta ruang terkait ke dalam memori dalam pohon seimbang struktur (untuk operasi yang sangat efisien), diindeks dengan offset, dan memutar ulang log ke dalam struktur itu. Peta ruang dalam memori kemudian merupakan representasi akurat tion dari ruang yang dialokasikan dan bebas di metaslab. ZFS juga memadatkan peta sebanyak mungkin dengan menggabungkan blok bebas yang berdekatan menjadi satu entri. Terakhir, daftar ruang kosong diperbarui pada disk sebagai bagian dari berorientasi transaksi pengoperasian ZFS . Selama fase pengumpulan dan penyortiran, permintaan blok bisa tetap terjadi, Dan ZFS memuaskan ini permintaan dari itu catatan. Di dalam esensi, itu catatan plus itu pohon yang seimbang ***adalah*** bebas daftar.

14.5.6 MEMANGKAS Tidak digunakan Blok

HDD s Dan lainnya penyimpanan media itu mengizinkan blok ke menjadi ditimpa untuk pembaruan membutuhkan hanya itu bebas daftar untuk mengelola bebas ruang angkasa. Blok Mengerjakan bukan membutuhkan ke menjadi diperlakukan khususnya saat dibebaskan. Blok yang dibebaskan biasanya menyimpan datanya (tetapi tanpa data apa pun mengajukan petunjuk ke itu memblokir) sampai itu data adalah ditimpa Kapan itu memblokir adalah Berikutnyadialokasikan.

Perangkat penyimpanan yang tidak mengizinkan penimpaan, seperti NVM berbasis flash perangkat penyimpanan, sangat menderita ketika algoritma yang sama diterapkan. Mengingat dari Bagian 11.1.2 itu perangkat tersebut harus terhapus sebelum mereka Bisa lagi ditulis, dan penghapusan tersebut harus dilakukan dalam jumlah besar (blok, terdiri dari halaman-halaman) dan memerlukan waktu yang relatif lama dibandingkan dengan membaca atau menulis.

Mekanisme baru diperlukan agar sistem file dapat menginformasikan penyimpanan perangkat bahwa halamannya gratis dan dapat dipertimbangkan untuk dihapus (setelah blok dikon- mempertahankan halaman ini sepenuhnya

gratis). Mekanisme itu bervariasi berdasarkan penyimpanannya pengontrol. Untuk ATA - terpasang mengemudi, dia adalah MEMANGKAS , ketika untuk Berbasis elektronik NVM penyimpanan, dia adalah itu batal mengalokasikan memerintah. Apa pun itu spesifik pengontrol memerintah, ini mekanisme terus penyimpanan ruang angkasa tersedia untuk menulis. Tanpa seperti A kemampuan- ity, perangkat penyimpanan menjadi penuh dan memerlukan pengumpulan sampah dan penghapusan blok, terkemuka ke berkurang di dalam penyimpanan masukan/keluaran menulis pertunjukan (diketahui sebagai " A menulis tebing ”).

Dengan itu MEMANGKAS mekanisme Dan serupa kemampuan, itu sampah koleksi Dan langkah-langkah penghapusan dapat dilakukan sebelum perangkat hampir penuh, sehingga memungkinkan perangkat untuk melakukannya menyediakan lagi konsisten pertunjukan.

14.6 Efisiensi Dan Pertunjukan

Sekarang itu Kami memiliki dibahas bermacam-macam alokasi blok Dan direktori- pilihan manajemen, kita dapat mempertimbangkan lebih lanjut pengaruhnya terhadap kinerja Dan efisien penyimpanan menggunakan. Disk cenderung ke mewakili A besar kemacetan di dalam sistem pertunjukan, sejak mereka adalah itu paling lambat utama komputer komponen. Bahkan NVM perangkat adalah lambat dibandingkan dengan CPU Dan utama Penyimpanan, Jadi milik mereka pertunjukan juga harus dioptimalkan. Pada bagian ini, kita membahas berbagai teknik digunakan ke memperbaiki itu efisiensi Dan pertunjukan dari sekunder penyimpanan.

14.6.1 Efisiensi

Penggunaan ruang perangkat penyimpanan yang efisien sangat bergantung pada alokasinya dan algoritma direktori yang digunakan. Misalnya UNIX inode telah dialokasikan sebelumnya pada suatu volume. Bahkan disk kosong pun memiliki persentase ruang yang hilang karena inode. Namun, oleh praalokasi itu inode Dan menyebar mereka lintas itu volume, Kami memperbaiki itu mengajukan sistem pertunjukan. Ini ditingkatkan pertunjukan hasil dari alokasi UNIX dan algoritma ruang kosong, yang mencoba menyimpan file data blok dekat itu file inode memblokir untuk mengurangi mencari waktu.

Sebagai contoh lain, mari kita pertimbangkan kembali skema pengelompokan yang dibahas Bagian 14.4, yang membaik pencarian file Dan transfer file pertunjukan pada itu biaya fragmentasi internal. Untuk mengurangi fragmentasi ini, BSD UNIX memvariasikan ukuran cluster seiring bertambahnya file. Cluster besar digunakan di mana mereka dapat diisi, dan cluster kecil digunakan untuk file kecil dan cluster terakhir dari sebuah file. Ini sistem adalah dijelaskan di dalam Lampiran C.

Jenis data yang biasanya disimpan dalam entri direktori file (atau inode) juga memerlukan pertimbangan. Umumnya, " tanggal penulisan terakhir " dicatat pada persediaan informasi kepada pengguna dan untuk menentukan apakah file perlu dicadangkan ke atas. Beberapa sistem juga menyimpan " tanggal akses terakhir " , sehingga pengguna dapat menentukannya kapan file terakhir dibaca. Hasil dari menyimpan informasi ini adalah, setiap kali file dibaca, bidang dalam struktur direktori harus ditulisi. Itu berarti blok tersebut harus dibaca ke dalam memori, suatu bagian diubah, dan memblokir tertulis kembali keluar ke itu perangkat, Karena operasi pada sekunder penyimpanan terjadi hanya di dalam memblokir (atau gugus) potongan. Jadi setiap waktu A mengajukan adalah dibuka untuk membaca, FCB-nya harus dibaca dan ditulis juga. Persyaratan ini bisa jadi tidak efisien sering diakses file, Jadi Kami harus Menimbang -nya keuntungan melawan -nya pertunjukan biaya saat merancang sistem file. Umumnya, setiap item data terkait dengan A mengajukan kebutuhan ke menjadi dipertimbangkan untuk -nya memengaruhi pada efisiensi Dan pertunjukan.

Mempertimbangkan, untuk contoh, Bagaimana efisiensi adalah terpengaruh oleh itu ukuran dari itu petunjuk digunakan untuk mengakses

data. Kebanyakan sistem menggunakan pointer 32-bit atau 64-bit melalui-keluar dari sistem operasi. Menggunakan pointer 32-bit membatasi ukuran file menjadi 2^{32}, atau4 GB . Menggunakan 64-bit petunjuk memungkinkan sangat besar mengajukan ukuran, Tetapi 64-bit petunjuk memerlukan lebih banyak ruang untuk menyimpan. Akibatnya terjadi alokasi dan pengelolaan ruang bebas metode (terhubung daftar, indeks, Dan Jadi pada) menggunakan lagi penyimpanan ruang angkasa.

Salah satu kesulitan dalam memilih ukuran penunjuk— atau, tentu saja, alokasi tetap apa pun ukuran kation dalam sistem operasi — merencanakan dampak perubahan teknologi. Pertimbangkan bahwa IBM PC XT memiliki hard drive 10 MB dan MS-DOS Sistem file FAT yang mampu mendukung hanya 32 MB . (Setiap entri FAT adalah 12 bit, menunjuk ke cluster 8- KB .) Ketika kapasitas disk meningkat, disk yang lebih besar harus meningkat dipecah menjadi partisi 32 MB , karena sistem file tidak dapat melacak blok di luar 32 MB . Sebagai keras disk dengan kapasitas dari lebih 100 MB menjadi umum, itu disk data struktur Dan algoritma di dalam MS-DOS punya ke menjadi diubah ke mengizinkan sistem file yang lebih besar. (Setiap entri FAT diperluas menjadi 16 bit dan kemudian menjadi 32 bits.) Keputusan sistem file awal dibuat untuk alasan efisiensi; Bagaimana- pernah, dengan munculnya MS-DOS Versi 4, jutaan pengguna komputer merasa tidak nyaman ketika mereka harus beralih ke sistem file baru yang lebih besar. milik Solaris ZFS mengajukan sistem kegunaan 128-bit petunjuk, yang secara teoretis sebaiknya tidak pernah membutuhkan ke menjadi diperpanjang. (Itu minimum massa dari A perangkat mampu dari menyimpan 2^{128} byte menggunakan tingkat atom penyimpanan akan menjadi sekitar 272 triliun kilogram.)

Sebagai contoh lain, perhatikan evolusi sistem operasi Solaris. item. Semula, banyak data struktur adalah dari tetap panjang, dialokasikan pada sistem rintisan. Struktur ini mencakup tabel proses dan tabel file terbuka. Kapan itu proses meja menjadi penuh, TIDAK lagi proses bisa menjadi dibuat. Kapan itu mengajukan meja menjadi penuh, TIDAK lagi file bisa menjadi dibuka. Itu sistem akan gagal ke menyediakan jasa ke pengguna. Meja ukuran bisa menjadi ditingkatkan hanya oleh kompilasi ulang kernel dan me-reboot sistem. Dengan rilis Solaris selanjutnya, (seperti kernel Linux modern) hampir semua struktur kernel dialokasikan secara dinamis, menghilangkan batasan buatan pada kinerja sistem. Tentu saja, algo- ritme yang memanipulasi tabel ini lebih rumit, dan pengoperasiannya sistem sedikit lebih lambat karena harus mengalokasikan dan membatalkan alokasi secara dinamis meja entri; Tetapi itu harga adalah itu biasa satu untuk lagi umum Kegunaan.

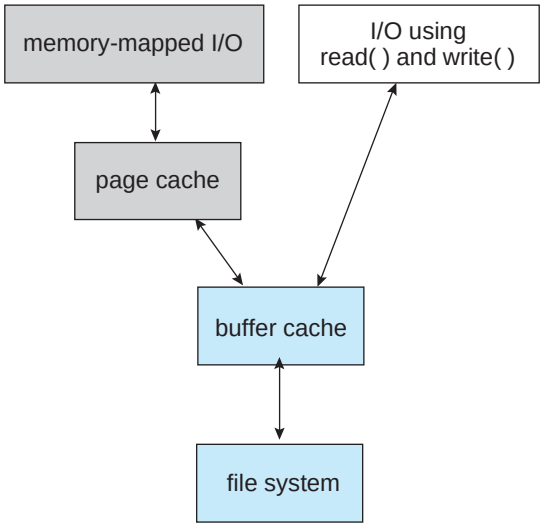
14.6.2 Pertunjukan

Bahkan setelah algoritma sistem file dasar dipilih, kita masih bisa memperbaiki pertunjukan di dalam beberapa cara. Sebagai dulu dibahas di dalam Bab 12, penyimpanan perangkat pengontrol termasuk lokal Penyimpanan ke membentuk sebuah di atas kapal cache itu adalah cukup besar untuk menyimpan seluruh trek atau blok sekaligus. Pada HDD , sekali pencarian dilakukan dilakukan, itu melacak adalah membaca ke dalam itu disk cache memulai pada itu sektor di bawah itu disk kepala (mengurangi latensi waktu). Itu disk pengontrol Kemudian transfer setiap sektor permintaan ke itu Pengoperasian sistem. Sekali blok membuat dia dari itu disk pengontrol ke dalam utama Penyimpanan, itu Pengoperasian sistem mungkin cache itu blok di sana.

Beberapa sistem memelihara bagian terpisah dari memori utama untuk **buffer cache** , di mana blok disimpan dengan asumsi bahwa blok tersebut akan digunakan lagi sebentar lagi. Sistem lain menyimpan data file cache menggunakan **cache halaman** . Halaman cache menggunakan teknik memori virtual untuk menyimpan data file sebagai halaman, bukan sebagai blok berorientasi sistem file. Menyimpan data file menggunakan alamat virtual masih jauh lagi efisien dibandingkan cache melalui fisik disk blok, sebagai mengakses antarmuka dengan memori virtual daripada sistem file. Beberapa sistem— termasuk Solaris, Linux, dan Windows— menggunakan

cache halaman untuk menyimpan cache kedua halaman proses Dan mengajukan data. Ini adalah dikenal sebagai **menyatukan maya Penyimpanan** .

Beberapa versi UNIX dan Linux menyediakan **pemersatu cache penyangga** . Ke menjelaskan itu manfaat dari itu bersatu penyangga cache, mempertimbangkan itu dua alternatif

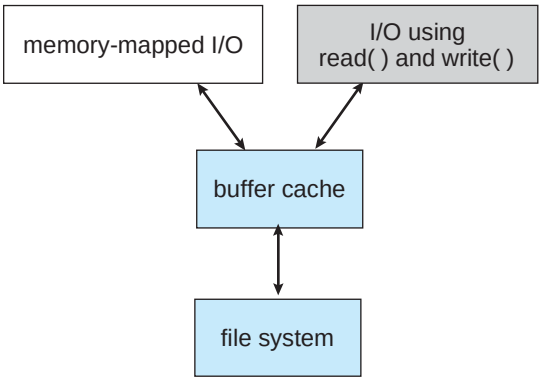


Angka 14.10 masukan/keluaran tanpa sebuah penyangga terpadu cache.

untuk membuka dan mengakses file. Salah satu pendekatannya adalah dengan menggunakan pemetaan memori (Pasal 13.5); yang kedua adalah menggunakan panggilan sistem standar read() dan menulis() . Tanpa A bersatu penyangga cache, Kami memiliki A situasi serupa ke Angka 14.10. Di Sini, itu membaca() Dan menulis() sistem panggilan pergi melalui itu penyangga cache. Namun, panggilan pemetaan memori memerlukan penggunaan dua cache— halaman cache dan cache buffer. Pemetaan memori dilanjutkan dengan membaca di disk blok dari sistem file dan menyimpannya dalam cache buffer. Karena sistem memori virtual tidak berinteraksi dengan cache buffer, isinya file dalam cache buffer harus disalin ke cache halaman. Situasi ini, dikenal sebagai **caching ganda** , memerlukan caching data sistem file dua kali. Tidak hanya apakah ini membuang-buang memori tetapi juga membuang-buang CPU dan siklus I/O secara signifikan ke pergerakan data tambahan dalam memori sistem. Selain itu, inkonsistensi antara dua cache dapat mengakibatkan file rusak. Berbeda dengan bila bersatu penyangga cache adalah asalkan, keduanya Penyimpanan pemetaan Dan itu membaca() Dan menulis() panggilan sistem menggunakan cache halaman yang sama. Ini mempunyai manfaat untuk menghindari hal ganda caching, dan memungkinkan sistem memori virtual untuk mengelola data sistem file. Itu bersatu penyangga cache adalah ditampilkan di dalam Angka 14.11.

Terlepas dari apakah kita melakukan cache blok penyimpanan atau halaman (atau keduanya), digunakan terakhir kali (LRU) (Bagian 10.4.4) tampaknya merupakan tujuan umum yang masuk akal algoritma untuk memblokir atau halaman penggantian. Namun, itu evolusi dari itu Solaris cache halaman algoritma mengungkapkan itu kesulitan di dalam memilih sebuah algoritma. Solaris memungkinkan proses Dan itu halaman cache ke membagikan tidak terpakai Penyimpanan. Versi lebih awal dibandingkan Solaris 2.5.1 tidak membuat perbedaan antara mengalokasikan halaman ke suatu proses Dan mengalokasikan mereka ke itu halaman cache. Sebagai A hasil, A sistem tampil banyak Operasi I/O menggunakan sebagian besar memori yang tersedia untuk menyimpan halaman. Karena dari tingginya tingkat I/O , pemindai halaman (Bagian 10.10.3) mengambil kembali halaman dari proses — lebih tepatnya dibandingkan dari itu halaman cache — Kapan bebas Penyimpanan berlari rendah. Solaris 2.6 dan Solaris 7 secara

opsional menerapkan paging prioritas, di mana pemindai halaman memberikan prioritas untuk memproses halaman daripada cache halaman. Solaris 8 terapan A tetap membatasi ke proses halaman Dan itu berkas sistem halaman cache, mencegah-



Angka 14.11 masukan/keluaran menggunakan A bersatu penyangga cache.

ing baik dari memaksa yang lain keluar dari memori. Solaris 9 dan 10 kembali mengalami perubahan itu algoritma ke maksimalkan Penyimpanan menggunakan Dan memperkecil labrakan.

Masalah lain yang dapat mempengaruhi kinerja I/O adalah apakah menulis ke atau tidak sistem file terjadi secara sinkron atau asinkron. **Penulisan sinkron** terjadi sesuai urutan subsistem penyimpanan menerimanya, dan penulisan tidak di-buffer. Oleh karena itu, pemanggilan rutin harus menunggu sampai datanya tiba mencapai itu menyetir sebelum dia Bisa melanjutkan. Di dalam sebuah **asinkron menulis** , itu data disimpan dalam cache, dan kontrol kembali ke pemanggil. Kebanyakan tulisan seperti itu asinkron. Namun, metadata menulis, di antara yang lain, Bisa menjadi sinkronis. Pengoperasian sistem sering termasuk A bendera di dalam itu membuka sistem panggilan ke mengizinkan sebuah proses untuk meminta agar penulisan dilakukan secara sinkron. Misalnya, database menggunakan fitur ini untuk transaksi atom, untuk memastikan jangkauan data stabil penyimpanan di dalam itu diperlukan memesan.

Beberapa sistem mengoptimalkan cache halamannya dengan menggunakan penggantian yang berbeda algoritma, tergantung pada jenis akses file. File sedang dibaca atau ditulis secara berurutan tidak boleh ada halaman-halamannya yang diganti secara LRU , karena halaman yang terakhir digunakan akan digunakan terakhir kali, atau mungkin tidak akan digunakan lagi. Alih-alih, akses sekuensial dapat dioptimalkan dengan teknik yang dikenal sebagai free-behind dan baca-depan. **Free-behind** menghapus halaman dari buffer segera setelah halaman berikutnya halaman diminta. Halaman sebelumnya kemungkinan besar tidak akan digunakan lagi dan ruang penyangga limbah. Dengan **read-ahead** , halaman yang diminta dan beberapa halaman berikutnya halaman dibaca dan di-cache. Halaman-halaman ini kemungkinan besar diminta setelah halaman saat ini diproses. Mengambil data ini dari disk dalam satu transfer dan menyimpannya dalam cache akan menghemat banyak waktu. Orang mungkin berpikir demikian cache track pada pengontrol akan menghilangkan kebutuhan untuk membaca terlebih dahulu pada a multiprogram sistem. Namun, Karena dari itu tinggi latensi Dan atas terlibat di dalam membuat banyak kecil transfer dari itu melacak cache ke utama Penyimpanan, tampil A baca-depan tetap bermanfaat.

Itu halaman cache, itu mengajukan sistem, Dan itu perangkat pengemudi memiliki beberapa minat- interaksi. Ketika sejumlah kecil data ditulis ke file,

halaman di-buffer dalam cache, dan driver perangkat penyimpanan mengurutkan antrian keluarannya menurut ke perangkat alamat. Ini dua tindakan mengizinkan A disk pengemudi ke memperkecil kepala disk mencari. Kecuali sinkronis menulis adalah diperlukan, A proses menulis ke disk secara sederhana menulis ke dalam itu cache, Dan itu sistem secara asinkron menulis itu

data ke disk bila nyaman. Proses pengguna melihat penulisan yang sangat cepat. Kapan data dibaca dari file disk, sistem blok I/O melakukan beberapa pembacaan terlebih dahulu; namun, penulisan hampir tidak sinkron dibandingkan dengan pembacaan. Dengan demikian, keluaran ke itu disk melalui itu mengajukan sistem adalah sering lebih cepat dibandingkan adalah memasukkan untuk kecil transfer, berlawanan dengan intuisi. Tidak peduli berapa banyak buffering dan caching I/O yang tersedia, besar, dan berkelanjutan dapat melampaui kapasitas dan berakhir dengan pembotolan. tergantung pada kinerja perangkat. Pertimbangkan untuk menulis file film besar ke a HDD . Jika file lebih besar dari cache halaman (atau bagian dari cache halaman tersedia untuk proses) maka cache halaman akan terisi dan semua I/O akan terjadi pada kecepatan berkendara. HDD saat ini membaca lebih cepat daripada menulis, jadi dalam hal ini pertunjukan aspeknya terbalik dari lebih kecil masukan/keluaran pertunjukan.

14.7 Pemulihan

File dan direktori disimpan di memori utama dan volume penyimpanan, Dan peduli harus menjadi diambil ke memastikan itu A sistem kegagalan melakukan bukan hasil di dalam kehilangan dari data atau ketidakkonsistenan data. Kegagalan sistem dapat menyebabkan ketidakkonsistenan antar sistem di penyimpanan berkas sistem data struktur, seperti sebagai direktori struktur, blok bebas pointer, dan pointer FCB gratis . Banyak sistem file menerapkan perubahan pada hal ini struktur yang ada. Operasi umum, seperti membuat file, dapat melibatkan banyak perubahan struktural dalam sistem file pada disk. Struktur direktori adalah diubah, FCB s adalah dialokasikan, data blok adalah dialokasikan, Dan itu bebas penting untuk semua blok ini berkurang. Perubahan ini dapat diinterupsi oleh a kecelakaan, dan ketidakkonsistenan antar struktur dapat terjadi. Misalnya, bebas FCB menghitung mungkin menunjukkan itu sebuah FCB telah pernah dialokasikan, Tetapi itu direktori struktur mungkin tidak menunjuk ke FCB . Yang memperparah masalah ini adalah caching itu Pengoperasian sistem Mengerjakan ke mengoptimalkan masukan/keluaran pertunjukan. Beberapa perubahan mungkin pergi langsung ke penyimpanan, sementara yang lain mungkin di-cache. Jika perubahan yang di-cache tidak mencapai itu penyimpanan perangkat sebelum A menabrak terjadi, lagi korupsi adalah mungkin.

Selain kerusakan, bug dalam implementasi sistem file, konfigurasi perangkat, troll, Dan bahkan pengguna aplikasi Bisa korup A mengajukan sistem. Mengajukan sistem memiliki berbagai metode untuk menangani korupsi, tergantung pada data sistem file struktur Dan algoritma. Kami kesepakatan dengan ini masalah Berikutnya.

14.7.1 Konsistensi Memeriksa

Apa pun itu menyebabkan dari korupsi, A mengajukan sistem harus Pertama mendeteksi itu masalah dan kemudian memperbaikinya. Untuk deteksi, scan semua metadata pada setiap file sistem Bisa mengonfirmasi atau membantah itu konsistensi dari itu sistem. Sayangnya, ini memindai Bisa mengambil menit atau jam Dan sebaiknya terjadi setiap waktu itu sistem sepatu bot. Alternatifnya, sistem file dapat mencatat statusnya dalam metadata sistem file. Pada awal setiap perubahan metadata, bit status disetel untuk menunjukkan bahwa metadata berubah-ubah. Jika semua pembaruan pada metadata berhasil diselesaikan, file sistem dapat menghapus bagian itu.

Namun, jika bit status tetap disetel, berarti konsistensi pemeriksa adalah berlari.

Pemeriksa konsistensi — program sistem seperti fsck di UNIX — membandingkan data dalam struktur direktori dan metadata lainnya dengan menyatakan pada penyimpanan dan mencoba memperbaiki ketidakkonsistenan yang ditemukan. Alokasi Dan manajemen ruang bebas algoritma mendikte Apa jenis dari masalah itu

pemeriksa Bisa menemukan Dan Bagaimana berhasil dia akan menjadi di dalam pemasangan mereka. Untuk contoh, jika terhubung alokasi adalah digunakan Dan di sana adalah A tautan dari setiap memblokir ke -nya Berikutnya blok, maka seluruh file dapat direkonstruksi dari blok data, dan struktur direktori dapat dibuat ulang. Sebaliknya, hilangnya entri direktori pada sistem alokasi yang diindeks dapat menjadi bencana, karena datanya diblokir tidak mempunyai pengetahuan satu sama lain. Karena alasan ini, beberapa sistem file UNIX cache direktori entri untuk membaca, Tetapi setiap menulis itu hasil di dalam ruang angkasa alokasi, atau perubahan metadata lainnya, dilakukan secara sinkron, sebelum yang bersangkutan blok data ditulis. Tentu saja masalah masih bisa terjadi jika sinkron menulis adalah terputus oleh A menabrak. Beberapa penyimpanan NVM perangkat berisi A baterai atau superkapasitor untuk memberikan daya yang cukup, bahkan saat listrik mati, untuk menulis data dari perangkat buffer ke itu penyimpanan media Jadi itu data adalah bukan hilang. Tetapi bahkan itu tindakan pencegahan Mengerjakan bukan melindungi melawan korupsi jatuh tempo ke A menabrak.

14.7.2 Terstruktur Log Mengajukan Sistem

Ilmuwan komputer sering kali menemukan bahwa algoritma dan teknologi aslinya digunakan di satu bidang sama bermanfaatnya di bidang lain. Demikian halnya dengan algoritma pemulihan berbasis log database. Algoritme pencatatan ini telah dilakukan berhasil diterapkan pada masalah pengecekan konsistensi. Hasilnya implementasinya dikenal sebagai **berorientasi transaksi berbasis log** (atau **penjurnalan**) mengajukan sistem.

Perhatikan bahwa dengan pendekatan pemeriksaan konsistensi yang dibahas di pra- bagian yang diserahkan, pada dasarnya kami mengizinkan struktur untuk dirusak dan diperbaiki pemulihan. Namun, di sana adalah beberapa masalah dengan ini mendekati. Satu adalah itu ketidakkonsistenan mungkin tidak dapat diperbaiki. Cek konsistensinya mungkin tidak bisa untuk memulihkan struktur, mengakibatkan hilangnya file dan bahkan seluruh direktori. Konsistensi memeriksa Bisa memerlukan manusia intervensi ke menyelesaikan konflik, Dan itu merepotkan jika tidak ada manusia yang tersedia. Sistem mungkin tetap tidak tersedia- mampu sampai manusia memberitahukannya bagaimana melanjutkannya. Pemeriksaan konsistensi juga memerlukan waktu sistem dan jam waktu. Ke memeriksa terabyte dari data, jam dari jam waktu mungkin menjadidiperlukan.

Solusi untuk masalah ini adalah dengan menerapkan teknik pemulihan berbasis log pembaruan metadata sistem file. Baik NTFS dan sistem file Veritas menggunakan ini metode, Dan dia adalah termasuk di dalam terkini versi dari UFS pada Solaris. Di dalam fakta, dia adalah Sekarang umum pada banyak orang mengajukan sistem termasuk ext3, ext4, Dan ZFS .

Pada dasarnya, semua perubahan metadata ditulis secara berurutan ke dalam log. Setiap mengatur dari operasi untuk tampil A spesifik tugas adalah A **transaksi** . Sekali itu perubahan ditulis untuk ini catatan, mereka adalah dianggap menjadi berkomitmen, dan panggilan sistem dapat kembali ke proses pengguna, memungkinkannya untuk melanjutkan eksekusi. Sementara itu, entri log ini diputar ulang di seluruh file- struktur sistem. Saat perubahan dilakukan, sebuah penunjuk diperbarui untuk menunjukkan yang tindakan memiliki lengkap Dan yang adalah tetap tidak lengkap. Kapan sebuah seluruh transaksi yang dilakukan selesai, dan entri dibuat di log yang menunjukkan itu. File log sebenarnya adalah buffer

melingkar. Buffer melingkar menulis ke akhir dari -nya ruang angkasa Dan Kemudian berlanjut pada itu awal, menimpa lebih tua nilai-nilai seiring berjalannya waktu. Kami tidak ingin buffer menulis data yang belum ditulis pernah diselamatkan, Jadi itu skenario adalah dihindari. Itu catatan mungkin menjadi di dalam A memisahkan bagian dari itu mengajukan sistem atau bahkan pada Yang terpisah penyimpanan perangkat.

Jika sistem crash, file log akan berisi nol atau lebih transaksi. Setiap transaksi dia mengandung adalah bukan lengkap ke itu mengajukan sistem, bahkan meskipun hal tersebut dilakukan oleh sistem operasi, sehingga sekarang harus diselesaikan. Transaksi dapat dieksekusi dari pointer hingga pekerjaan selesai sehingga struktur sistem file tetap konsisten. Satu-satunya masalah terjadi ketika suatu transaksi dibatalkan — yaitu, tidak dilakukan sebelum sistem jatuh. Setiap perubahan dari transaksi tersebut yang diterapkan pada file sistem harus dibatalkan, sekali lagi menjaga konsistensi sistem file. Ini pemulihan adalah semua itu adalah diperlukan setelah A menabrak, menghilangkan setiap masalah dengankonsistensi memeriksa.

Manfaat tambahan dari menggunakan pembaruan metadata disk adalah hal itu pembaruan berlangsung jauh lebih cepat dibandingkan ketika diterapkan langsung ke on- struktur data disk. Alasannya terletak pada keunggulan kinerja I/O berurutan melalui I/O acak . Metadata acak sinkron yang mahal penulisan diubah menjadi penulisan sekuensial sinkron yang jauh lebih murah terstruktur log mengajukan sistem pencatatan daerah. Itu perubahan, di dalam berbelok, adalah diputar ulang secara asinkron melalui penulisan acak ke struktur yang sesuai. Secara keseluruhan Hasilnya adalah peningkatan yang signifikan dalam kinerja operasi berorientasi metadata, seperti mengajukan penciptaan Dan penghapusan, pada HDD penyimpanan.

14.7.3 Lainnya Solusi

Alternatif lain untuk pemeriksaan konsistensi digunakan oleh Network Appli- sistem file WAFL ance dan sistem file Solaris ZFS . Sistem ini tidak pernah menimpa blok dengan baru data. Lebih tepatnya, A transaksi menulis semua data Dan meta- perubahan data ke blok baru. Ketika transaksi selesai, metadata struktur itu lancip ke itu tua versi dari ini blok adalah diperbarui ke titik ke blok baru. Sistem file kemudian dapat menghapus pointer lama dan blok lama dan membuatnya tersedia untuk digunakan kembali. Jika pointer dan blok lama disimpan, **snapshot** dibuat; snapshot adalah tampilan sistem file di a titik waktu tertentu (sebelum pembaruan apa pun setelah waktu tersebut diterapkan). Ini solusi seharusnya tidak memerlukan pemeriksaan konsistensi apakah pembaruan penunjuk telah selesai secara atomik. WAFL memang memiliki pemeriksa konsistensi, namun ada beberapa kegagalan skenario masih dapat menyebabkan kerusakan metadata. (Lihat Bagian 14.8 untuk rincian itu WAFL mengajukan sistem.)

ZFS mengambil pendekatan yang lebih inovatif terhadap konsistensi disk. Seperti WAFL , dia tidak pernah menimpa blok. Namun, ZFS pergi lebih jauh Dan menyediakan checksum- ming semua metadata dan blok data. Solusi ini (bila dikombinasikan dengan RAID) memastikan bahwa data selalu benar. Oleh karena itu ZFS tidak memiliki konsistensi pemeriksa. (Lagi detail pada ZFS adalah ditemukan di dalam Bagian 11.8.6.)

14.7.4 Cadangan Dan Memulihkan

Penyimpanan perangkat Kadang-kadang gagal, Dan peduli harus menjadi diambil ke memastikan itu itu data hilang dalam kegagalan seperti itu tidak hilang selamanya. Untuk tujuan ini, program sistem dapat digunakan digunakan untuk **mencadangkan** data dari satu perangkat penyimpanan ke perangkat penyimpanan lainnya, seperti magnet tape atau lainnya sekunder penyimpanan perangkat. Pemulihan dari itu kehilangan dari sebuah individu file, atau seluruh perangkat, mungkin hanya masalah **memulihkan** data darinya cadangan.

Untuk meminimalkan kebutuhan penyalinan, kita dapat menggunakan

informasi dari setiap file direktori pintu masuk. Untuk contoh, jika itu cadangan program tahu Kapan itu terakhir

pencadangan file telah selesai, dan tanggal penulisan terakhir file di direktori menunjukkan bahwa file tersebut tidak berubah sejak tanggal tersebut, maka file tersebut tidak perlu ada disalin lagi. A khas cadangan jadwal mungkin Kemudian menjadi sebagai berikut:

- **Hari 1** . Menyalin ke A cadangan sedang semua file dari itu mengajukan sistem. Ini adalah ditelepon**penuh cadangan** .
- **Hari 2** . Menyalin ke lain sedang semua file berubah sejak hari 1. Ini adalah sebuah
  tambahan cadangan .
- **Hari 3** . Menyalin ke lain sedang semua file berubah sejak hari 2.

. . .

- **Hari** *N* . Menyalin ke lain sedang semua file berubah sejak hari *Tidak* − 1. Kemudian pergi kembali ke hari 1.

Itu baru siklus Bisa memiliki -nya cadangan tertulis lebih itu sebelumnya mengatur atau ke A barumengatur cadangan media.

Menggunakan ini metode, Kami Bisa memulihkan sebuah seluruh mengajukan sistem oleh memulai memulihkan dengan itu penuh cadangan Dan melanjutkan melalui setiap dari itu tambahan cadangan. Dari kursus, itu lebih besar itu nilai dari *N*, itu lebih besar itu nomor dari media itu harus dibaca untuk pemulihan lengkap. Keuntungan tambahan dari siklus pencadangan ini adalah kita dapat memulihkan file apa pun yang terhapus secara tidak sengaja selama siklus dengan mengambil kembali file tersebut dihapus mengajukan dari itu cadangan dari itu sebelumnya hari.

Panjang siklus merupakan kompromi antara jumlah cadangan diperlukan Dan itu nomor dari hari tertutupi oleh A memulihkan. Ke mengurangi itu nomor dari kaset itu harus menjadi membaca ke Mengerjakan A memulihkan, sebuah pilihan adalah ke melakukan A penuh cadangan Dan Kemudian setiap hari kembali ke atas semua file itu memiliki berubah sejak itu penuh cadangan. Di dalam dengan cara ini, pemulihan dapat dilakukan melalui cadangan tambahan terbaru dan pencadangan penuh, tanpa memerlukan pencadangan tambahan lainnya. Imbalannya adalah itu lebih banyak file akan diubah setiap hari, sehingga setiap cadangan tambahan berturut-turut melibatkan lagi file dan banyak lagi media cadangan.

Pengguna mungkin menyadari bahwa file tertentu hilang atau rusak lama setelahnya kerusakan telah terjadi. Oleh karena itu, kami biasanya berencana untuk mengambil cadangan penuh dari waktu ke waktu yang akan disimpan “ selamanya. “ Merupakan ide bagus untuk menyimpannya cadangan permanen jauh dari cadangan reguler yang harus dilindungi bahaya, seperti kebakaran yang menghancurkan komputer dan semua cadangannya juga. Di dalam acara TV “ Mr. Robot, ” peretas tidak hanya menyerang sumber utama data bank tetapi juga situs cadangannya. Memiliki beberapa situs cadangan mungkin bukan menjadi buruk ide jika milikmu datanya adalah penting.

14.8 Contoh: Itu WAFL Mengajukan Sistem

Karena penyimpanan sekunder masukan/keluaran memiliki seperti A sangat besar dampak pada sistem pertunjukan, berkas sistem desain Dan penerapan

memerintah lumayan A banyak dari Perhatian dari perancang sistem. Beberapa sistem file memiliki tujuan umum, sehingga dapat digunakan memberikan kinerja dan fungsionalitas yang wajar untuk berbagai macam file ukuran, jenis file, dan beban I/O . Lainnya dioptimalkan untuk tugas tertentu dalam suatu percobaan ke menyediakan lebih baik pertunjukan di dalam itu daerah dibandingkan tujuan umum

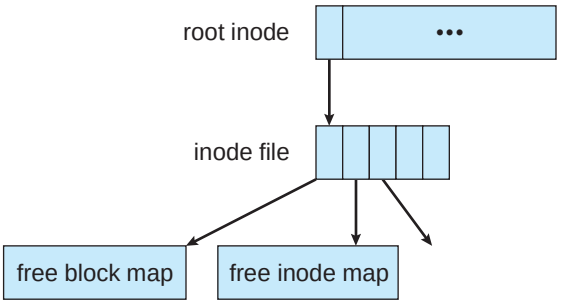
sistem file. Tata **letak file tulis di mana saja** (**WAFL**) dari NetApp, Inc contoh optimasi semacam ini. WAFL adalah sistem file yang kuat dan elegan dioptimalkan untuk penulisan acak.

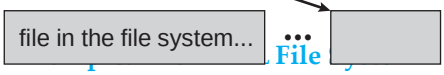
WAFL adalah digunakan khusus pada jaringan mengajukan server diproduksi oleh Aplikasi Net Dan dimaksudkan untuk digunakan sebagai sistem file terdistribusi. Itu dapat memberikan file ke klien melalui protokol NFS , CIFS , iSCSI , ftp , dan http , meskipun dirancang hanya untuk NFS Dan CIFS . Kapan banyak klien menggunakan ini protokol ke bicara ke A mengajukan pelayan, server mungkin melihat permintaan yang sangat besar untuk pembacaan acak dan bahkan lebih besar lagi permintaan untuk penulisan acak. Protokol NFS dan CIFS menyimpan data dalam cache dari pembacaan operasi, Jadi menulis adalah dari itu terbesar kekhawatiran ke server file pencipta.

WAFL digunakan pada server file itu termasuk NVRAM cache untuk menulis. Para desainer WAFL mengambil keuntungan dari menjalankan arsitektur tertentu untuk mengoptimalkan sistem file untuk I/O acak , dengan cache penyimpanan stabil di depan. Kemudahan penggunaan adalah salah satu prinsip panduan WAFL . Penciptanya juga merancangnya ke termasuk A baru foto Kegunaan itu menciptakan banyak hanya baca salinan dari itu mengajukan sistem pada berbeda poin di dalam waktu, sebagai Kami sebaiknya melihat.

Sistem filenya mirip dengan Berkeley Fast File System, dengan banyak file modifikasi. Ini berbasis blok dan menggunakan inode untuk mendeskripsikan file. Setiap inode berisi 16 pointer ke blok (atau blok tidak langsung) milik file yang dijelaskan oleh inode. Setiap sistem file memiliki inode root. Semua metadata ada dalam file. Semua inode ada dalam satu file, peta blok bebas di file lain, dan inode gratis peta di sepertiga, seperti yang ditunjukkan pada Gambar 14.12. Karena ini adalah file standar, file blok data tidak terbatas lokasinya dan dapat ditempatkan dimana saja. Jika sebuah file sistem diperluas dengan penambahan disk, panjang file metadata adalah secara otomatis diperluas oleh mengajukan sistem.

Jadi, WAFL sistem file adalah pohon blok dengan inode root sebagai miliknya basis. Untuk mengambil snapshot, WAFL membuat salinan inode root. File apa pun atau pembaruan metadata setelah itu masuk ke blok baru daripada menimpanya yang ada blok. Itu baru akar inode poin ke metadata Dan data berubah sebagai A hasil dari ini menulis. Sementara itu, itu foto (itu tua akar inode) tetap poin ke itu tua blok, yang memiliki bukan pernah diperbarui. Dia Karena itu menyediakan mengakses ke sistem file seperti saat snapshot dibuat — dan diambil sangat sedikit ruang penyimpanan untuk melakukannya. Intinya, ruang ekstra yang ditempati oleh a foto terdiri dari hanya itu blok itu memiliki pernah diubah sejak itu fotodulu diambil.



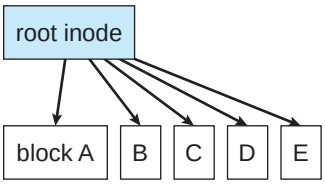


Angka 14.12 Itu WAFL mengajukan tata letak.

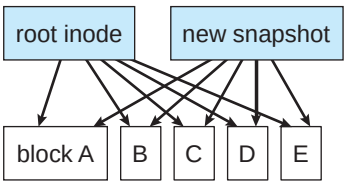
Perubahan penting dari sistem file yang lebih standar adalah free-block peta memiliki lebih dari satu bit per blok. Ini adalah bitmap dengan set bit untuk masing-masingnya snapshot yang menggunakan blok. Ketika semua snapshot yang telah menggunakan memblokir adalah dihapus, itu bitmap untuk itu memblokir adalah semua nol, Dan itu memblokir adalah bebas ke digunakan kembali. Blok bekas tidak pernah ditimpa, jadi penulisannya sangat cepat, karena A menulis Bisa terjadi pada itu bebas memblokir terdekat itu saat ini kepala lokasi. Di sana adalah banyak lainnya pertunjukan optimasi di WAFL sebagai Sehat.

Banyak snapshot dapat diambil secara bersamaan, sehingga satu dapat diambil setiap jam hari dan setiap hari dalam sebulan, misalnya. Seorang pengguna dengan akses ke ini snapshot Bisa mengakses file sebagai mereka adalah pada setiap dari itu waktu itu snapshot telah diambil. Fasilitas snapshot juga berguna untuk backup, pengujian, pembuatan versi, dan seterusnya. Fasilitas snapshot WAFL sangat efisien karena tidak merata memerlukan itu salin-saat-tulis salinan dari setiap data memblokir menjadi diambil sebelum itu memblokir dimodifikasi. Sistem file lain menyediakan snapshot, namun sering kali lebih sedikit efisiensi. WAFL snapshot adalah digambarkan di dalam Angka 14.13.

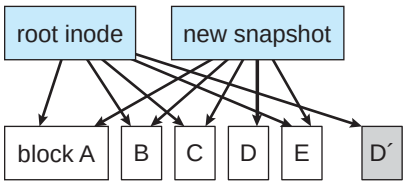
Versi WAFL yang lebih baru sebenarnya mengizinkan snapshot baca-tulis, yang dikenal sebagai **klon** . Klon juga efisien, menggunakan teknik yang sama seperti shapshot. Di dalam ini kasus, A hanya baca foto menangkap itu negara dari itu mengajukan sistem, Dan A klon merujuk kembali ke snapshot hanya-baca itu. Setiap penulisan ke klon disimpan di blok baru, dan penunjuk klon diperbarui untuk merujuk ke blok baru. Itu asli cuplikan adalah tidak dimodifikasi, tetap memberi A melihat ke dalam itu mengajukan sistem sebagai



(a) Sebelum A foto.



(b) Setelah A foto, sebelum setiap blok mengubah.



(c) Setelah memblokir D memiliki berubah ke D.

Angka 14.13 Jepretan di dalam WAFL.

ITU APEL MENGAJUKAN SISTEM

Di dalam 2017, Apel, Inc., dilepaskan A baru mengajukan sistem ke mengganti -nya berusia 30 tahun HFS + mengajukan sistem. HFS + telah pernah membentang ke menambahkan banyak baru fitur, Tetapi sebagai biasa, proses ini menambah kompleksitas, bersama dengan baris kode, dan membuat penambahan lebih banyak fitur lebih sulit. Memulai dari awal pada halaman kosong memungkinkan a desain ke awal dengan saat ini teknologi Dan metodologi Dan menyediakan itu akurat mengatur dari fitur diperlukan.

apel Mengajukan Sistem (**APFS**) adalah A Bagus contoh dari seperti A desain. Dia sasaran adalah ke berlari pada semua saat ini apel perangkat, dari itu apel Jam tangan melalui itu iPhone ke komputer Mac. Membuat sistem file yang berfungsi di OS jam tangan , saya/O , sistem operasi TV , Dan MacOS adalah tentu A tantangan. APFS adalah kaya fitur, termasuk 64-bit petunjuk, klon untuk file Dan direktori, cuplikan, ruang angkasa membagikan, cepat ukuran direktori, primitif penyimpanan aman atom, desain copy-on-write, enkripsi tion (kunci tunggal dan multi-kunci), dan penggabungan I/O . Ia memahami NVM juga sebagai HDD penyimpanan.

Sebagian besar fitur ini telah kita bahas, namun ada beberapa konsep baru bernilai menjelajah. **Ruang angkasa membagikan** adalah A ZFS -seperti fitur di dalam yang penyimpanan adalah faedah- mampu sebagai satu atau lebih ruang bebas besar (**kontainer**) dari sistem file mana Bisa menggambar alokasi (mengizinkan APFS -diformat volume ke tumbuh Dan menyusut). **Ukuran direktori yang cepat** menyediakan penghitungan dan pembaruan ruang yang terpakai dengan cepat. **Penyimpanan aman atom** bersifat primitif (tersedia melalui API , bukan melalui sistem file commands) yang melakukan penggantian nama file, kumpulan file, dan direktori sebagai operasi atom tunggal. Penggabungan I/O adalah pengoptimalan untuk perangkat NVM di mana beberapa tulisan kecil dikumpulkan menjadi satu tulisan besar mengoptimalkan menulis kinerja.

Apple memilih untuk tidak mengimplementasikan RAID sebagai bagian dari APFS baru tergantung pada mekanisme volume RAID Apple yang ada untuk perangkat lunak RAID . APFS juga kompatibel dengan HFS +, memungkinkan konversi yang mudah untuk yang sudah ada penerapan.

itu sebelum klon diperbarui. Klon juga dapat dipromosikan untuk menggantikan sistem file asli; ini melibatkan membuang semua petunjuk lama dan blok lama apa pun yang terkait. Klon berguna untuk pengujian dan peningkatan, seperti halnya versi asli tidak tersentuh dan klonnya dihapus saat pengujian selesaiatau jika itu meningkatkan gagal.

Fitur lain yang secara alami dihasilkan dari penerapan sistem file WAFL- tasi adalah **replikasi,** duplikasi dan sinkronisasi sekumpulan data melalui a jaringan ke sistem lain. Pertama, snapshot dari sistem file WAFL diduplikasi ke lain sistem. Kapan lain foto adalah diambil pada itu sumber sistem, dia relatif mudah untuk memperbarui sistem jarak jauh hanya dengan mengirimkan semua blok terkandung dalam snapshot baru. Blok-blok inilah yang telah berubah antara waktu kedua snapshot diambil. Sistem jarak jauh menambahkan ini blok ke itu mengajukan sistem Dan pembaruan -nya petunjuk, Dan itu baru sistem Kemudian adalah A duplikat dari itu sumber sistem sebagai dari itu waktu dari itu Kedua foto. Mengulangi proses ini mempertahankan sistem

jarak jauh sebagai salinan sistem jarak jauh yang paling mutakhir sistem. Replikasi tersebut digunakan untuk pemulihan bencana. Seharusnya sistem pertama menjadi hancur, paling dari -nya data adalah tersedia untuk menggunakan pada itu terpencil sistem.

Akhirnya, catatan itu itu ZFS mengajukan sistem mendukung demikian pula efisien cuplikan, klon, dan replikasi, dan fitur-fitur tersebut menjadi lebih umum di bermacam-macam mengajukan sistem sebagai waktu pergi oleh.

14.9 Ringkasan

- Paling mengajukan sistem tinggal pada sekunder penyimpanan, yang adalah dirancang ke memegang A sejumlah besar data secara permanen. Penyimpanan sekunder yang paling umum sedang adalah itu disk, Tetapi itu menggunakan dari NVM perangkat adalah meningkat.
- Penyimpanan perangkat adalah tersegmentasi ke dalam partisi ke kontrol media menggunakan Dan ke memungkinkan beberapa, mungkin berbeda-beda, sistem file pada satu perangkat. Berkas ini sistem adalah dipasang ke A logis mengajukan sistem Arsitektur ke membuat mereka tersedia untuk digunakan.
- Sistem file sering kali diimplementasikan dalam struktur berlapis atau modular. Tingkat yang lebih rendah berhubungan dengan sifat fisik perangkat penyimpanan dan berkomunikasi dengan mereka. Atas tingkat kesepakatan dengan simbolis mengajukan nama Danlogis properti file.
- Itu bermacam-macam file di dalam A mengajukan sistem Bisa menjadi dialokasikan ruang angkasa pada itu penyimpanan perangkat dalam tiga cara: melalui alokasi yang berdekatan, tertaut, atau terindeks. Berdekatan alokasi Bisa menderita dari luar fragmentasi. Langsung akses sangat tidak efisien dengan alokasi tertaut. Alokasi yang diindeks mungkin memerlukan besar atas untuk -nya indeks memblokir. Ini algoritma Bisa dioptimalkan dalam banyak hal. Ruang yang berdekatan dapat diperbesar melalui luasnya ke meningkatkan fleksibilitas Dan ke mengurangi luar fragmentasi. Alokasi yang diindeks dapat dilakukan dalam kelompok beberapa blok untuk ditingkatkan throughput dan untuk mengurangi jumlah entri indeks yang diperlukan. Pengindeksan di dalam besar cluster adalah serupa ke berdekatan alokasi dengan luasnya.
- Metode alokasi ruang kosong juga mempengaruhi efisiensi ruang disk penggunaan, kinerja sistem file, dan keandalan sekunder penyimpanan. Metode yang digunakan antara lain bit vektor dan daftar tertaut. Optimasi- tions termasuk pengelompokan, perhitungan, Dan itu GEMUK , yang tempat itu terhubung daftar di dalam satu wilayah yang berdekatan.
- Rutinitas manajemen direktori harus mempertimbangkan efisiensi, kinerja, dan keandalan. Tabel hash adalah metode yang umum digunakan karena cepat dan cepat efisien. Sayangnya, kerusakan ke itu meja atau A sistem menabrak Bisa hasil di dalam ketidakkonsistenan di antara itu direktori informasi Dan itu disk isi.
- Pemeriksa konsistensi dapat digunakan untuk memperbaiki struktur sistem file yang rusak. tur. Sistem operasi cadangan peralatan mengizinkan data ke menjadi disalin ke bersifat magnetis tape atau perangkat penyimpanan lainnya, memungkinkan pengguna untuk memulihkan dari kehilangan data atau bahkan seluruh perangkat hilang karena kegagalan perangkat keras, bug sistem operasi, atau pengguna

kesalahan.

- Karena peran mendasar yang dimainkan sistem file dalam operasi sistem, milik mereka pertunjukan Dan keandalan adalah penting. Teknik seperti sebagai catatan struktur- tur dan caching membantu meningkatkan kinerja, sementara struktur log dan SERANGAN memperbaiki keandalan. Itu WAFL mengajukan sistem adalah sebuah contoh dari mengoptimalkan- tion kinerja untuk mencocokkan yang spesifik masukan/keluaran memuat.

Praktik Latihan

14.1 Misalkan sebuah file yang saat ini terdiri dari 100 blok. Asumsikan bahwa blok kontrol file (dan blok indeks, dalam kasus alokasi yang diindeks tion) sudah ada dalam memori. Hitung berapa banyak operasi I/O disk diperlukan untuk alokasi yang berdekatan, tertaut, dan terindeks (tingkat tunggal). strategi tion, jika, untuk satu blok, kondisi berikut terpenuhi. Dalam alokasi bersebelahan kasus, menganggap itu di sana adalah TIDAK ruang ke tumbuh pada itu awal tetapi masih ada ruang untuk berkembang pada akhirnya. Asumsikan juga bahwa memblokir informasi ke menjadi ditambahkan adalah disimpan di dalam Penyimpanan.

a. Itu memblokir adalah ditambahkan pada itu awal.

b. Itu memblokir adalah ditambahkan di dalam itu tengah.

c. Itu memblokir adalah ditambahkan pada itu akhir.

d. Itu memblokir adalah DIHAPUS dari itu awal.

e. Itu memblokir adalah DIHAPUS dari itu tengah.

f. Itu memblokir adalah DIHAPUS dari itu akhir.

14.2 Mengapa harus itu sedikit peta untuk mengajukan alokasi menjadi disimpan pada massa penyimpanan, lebih tepatnya dibandingkan di dalam utama Penyimpanan?

14.3 Pertimbangkan sebuah sistem yang mendukung strategi yang berdekatan, terkait, dan alokasi yang diindeks. Kriteria apa yang harus digunakan dalam memutuskan yang mana strategi adalah terbaik dimanfaatkan untuk tertentu mengajukan?

14.4 Satu masalah dengan alokasi bersebelahan adalah pengguna harus melakukan pra-allo- sediakan cukup ruang untuk setiap file. Jika file menjadi lebih besar dari ruang yang dialokasikan untuk itu, tindakan khusus harus diambil. Salah satu solusi untuk masalah ini adalah mendefinisikan struktur file yang terdiri dari konfigurasi awal area bersebelahan dengan ukuran tertentu. Jika area ini terisi, sistem operasi tem secara otomatis mendefinisikan area luapan yang ditautkan ke awal berdekatan daerah. Jika itu meluap daerah adalah dipenuhi, lain meluap daerah dialokasikan. Bandingkan implementasi file ini dengan standar bersebelahan dan terhubung implementasi.

14.5 Bagaimana Mengerjakan cache membantu memperbaiki pertunjukan? Mengapa Mengerjakan sistem bukan menggunakan lagi atau lebih besar cache jika mereka adalah sangat berguna?

14.6 Mengapa adalah dia menguntungkan ke itu pengguna untuk sebuah Pengoperasian sistem ke dinamis- dengan tidak sopan mengalokasikan -nya intern tabel? Apa adalah itu hukuman ke itu Pengoperasian sistem untuk melakukan hal itu?

Lebih jauh Membaca

Itu internal dari itu BSD sistem UNIX adalah tertutupi di dalam penuh di

dalam [McKusick et Al. (2015)]. Detail mengenai sistem file untuk Linux dapat ditemukan di [Love (2010)]. Sistem file Google dijelaskan dalam [Ghemawat dkk. (2003)]. SEKERING bisa saja ditemukan pada http://fuse.sourceforge.net .

Organisasi file berstruktur log untuk meningkatkan kinerja dan konfigurasi konsistensi dibahas dalam [Rosenblum dan Ousterhout (1991)], [Seltzer et al. (1993)], dan [Seltzer dkk. (1995)]. Desain terstruktur log untuk file jaringan sistem diusulkan dalam [Hartman dan Ousterhout (1995)] dan [Thekkath et al. (1997)].

Itu ZFS sumber kode untuk ruang angkasa peta Bisa menjadi ditemukan pada http://src.opensolaris.o rg/sumber/xref/onnv/onnv-gate/usr/src/uts/common/ fs/zfs/spasi peta.c.

ZFS dokumentasi Bisa menjadi ditemukan pada http://www.opensolaris.org/os/community/ZFS/dokumen .

Itu NTFS mengajukan sistem adalah menjelaskan di dalam [Salomo (1998)], itu Ekst3 mengajukan sistem digunakan di Linux dijelaskan dalam [Mauerer (2008)], dan sistem file WAFL dijelaskan tertutupi di dalam [Hitz dkk Al. (1995)].

Bibliografi

[Ghemawat dkk. (2003)] S. Ghemawat, H. Gobioff, Dan S.-T. Leung, " Itu Google File System " , *Prosiding Simposium ACM tentang Sistem Operasi Prinsip* (2003).

[Hartman dan Ousterhout (1995)] JH Hartman dan JK Ousterhout, " Itu Zebra Striped Network File System " , *Transaksi ACM pada Sistem Komputer* , Volume 13, Nomor 3 (1995), halaman 274–310.

[Hitz dkk. (1995)] D. Hitz, J. Lau, dan M. Malcolm, " Desain Sistem File untuk sebuah NFS Mengajukan pelayan Peralatan " , Teknis laporan, Aplikasi Net (1995).

[Cinta (2010)] R. Love, *Pengembangan Kernel Linux,* Edisi Ketiga, Pengembang Perpustakaan (2010).

[Mauerer (2008)] W. Mauerer, *Arsitektur Kernel Linux Profesional* , John Wiley Dan Anak laki-laki (2008).

[McKusick dkk. (2015)] MK McKusick, GV Neville-Neil, dan RNM Wat- putra, *Itu Desain Dan Penerapan dari itu FreeBSD UNIX Pengoperasian Sistem- Kedua Edisi* , Pearson (2015).

[Rosenblum Dan Pemecatan (1991)] M. Rosenblum Dan J. K. pemecatan, " Itu Desain Dan Penerapan dari A Terstruktur Log Mengajukan Sistem " , *Proses dari ituACM Simposium pada Pengoperasian Sistem Prinsip* (1991), halaman 1–15.

[Seltzer dkk. (1993)] MI Seltzer, K. Bostic, MK McKusick, dan C. Staelin, " Implementasi Sistem File Terstruktur Log untuk UNIX " , *USENIX Winter* (1993), halaman 307–326.

[Seltzer dkk Al. (1995)] M. I.Seltzer, K. A. Smith, H. Balakrishnan, J. Chang, S.McMains, dan VN Padmanabhan, " File Sistem Pencatatan Versus Pengelompokan: A Pertunjukan Perbandingan " , *USENIX Musim dingin* (1995), halaman 249–264.

[Salomo (1998)] D. A. Salomo, *Di dalam jendela tidak,* Kedua Edisi, Microsoft Tekan (1998).

[Thekkath et Al. (1997)] C. A. Thekkath, T. Man, Dan E. K. Lee, " Frangipani: Sistem File Terdistribusi yang Dapat Diskalakan " , *Simposium Prinsip Sistem Operasi*(1997), halaman 224–237.

Bab 14 Latihan

14.7 Mempertimbangkan A mengajukan sistem itu kegunaan A diubah alokasi bersebelahan skema dengan dukungan untuk luasan. File adalah kumpulan luasan, dengan setiap cakupan sesuai ke A berdekatan mengatur dari blok. A kunci masalah dalam sistem seperti itu adalah tingkat variabilitas dalam ukuran luasan. Apa adalah itu keuntungan Dan kekurangan dari itu mengikuti skema?

a. Semua luasnya adalah dari itu sama ukuran, Dan itu ukuran adalah telah ditentukan sebelumnya.

b. Sejauh mana Bisa menjadi dari setiap ukuran Dan adalah dialokasikan secara dinamis.

c. Sejauh mana Bisa menjadi dari A sedikit tetap ukuran, Dan ini ukuran adalah ditentukan sebelumnya-beranjau.

14.8 Bandingkan kinerja ketiga teknik pengalokasian disk blok (bersebelahan, tertaut, dan terindeks) untuk sekuensial dan acak dom akses berkas.

14.9 Apa adalah itu keuntungan dari itu varian dari terhubung alokasi itu kegunaan A
GEMUK ke rantai bersama itu blok dari A mengajukan?

14.10 Mempertimbangkan A sistem Di mana bebas ruang angkasa adalah disimpan di dalam A ruang bebas daftar.

a. Memperkirakan itu itu penunjuk ke itu ruang bebas daftar adalah hilang. Bisa itusistem merekonstruksi itu ruang bebas daftar? Menjelaskan milikmu menjawab.

b. Pertimbangkan sistem file yang mirip dengan yang digunakan oleh UNIX alokasi yang diindeks. Berapa banyak operasi I/O disk yang mungkin dilakukan diperlukan untuk membaca konten file lokal kecil di `/a/b/c` ? Menganggap itu tidak ada dari itu disk blok adalah saat ini makhluk di-cache.

c. Sarankan skema untuk memastikan bahwa penunjuk tidak pernah hilang sebagai hasilnya dari Penyimpanan kegagalan.

14.11 Beberapa sistem file memungkinkan penyimpanan disk dialokasikan pada tingkat yang berbeda granularitas. Misalnya, sistem file dapat mengalokasikan 4 KB disk ruang sebagai satu 4- KB blok atau sebagai delapan blok 512-byte. Bagaimana bisa kita memanfaatkan fleksibilitas ini untuk meningkatkan kinerja? Apa modifikasi harus dilakukan pada pengelolaan ruang bebas skema di dalam memesan ke mendukung ini fitur?

14.12 Diskusikan bagaimana hasil optimasi kinerja untuk sistem file kesulitan dalam menjaga konsistensi sistem dalam acara tersebut dari komputer mogok.

14.13 Diskusikan kelebihan dan kekurangan tautan pendukung ke file itu menyeberang gunung poin (itu adalah, itu mengajukan tautan merujuk ke A mengajukan itu adalah disimpan di dalam berbeda

volume).

14.14 Pertimbangkan sistem file pada disk yang memiliki logika dan fisik memblokir ukuran dari 512 byte. Menganggap itu itu informasi tentang setiap mengajukan sudah ada dalam ingatan. Untuk masing-masing dari tiga strategi alokasi (kon- bersebelahan, terhubung, dan diindeks), jawab ini pertanyaan:

a. Bagaimana adalah itu logis-ke-fisik alamat pemetaan ahli dalam sistem ini? (Untuk alokasi yang diindeks, asumsikan bahwa sebuah file adalah selalu lebih sedikit dari 512 blok panjang.)

b. Jika Kami adalah saat ini pada logis memblokir 10 (itu terakhir memblokir diakses dulu blok 10) dan ingin mengakses blok logis 4, berapa banyak fisiknya blok harus menjadi membaca dari itu disk?

14.15 Pertimbangkan sistem file yang menggunakan inode untuk mewakili file. Blok disk adalah 8 KB di dalam ukuran, dan sebuah penunjuk ke A disk memblokir memerlukan 4 byte. Ini mengajukan sistem memiliki 12 blok disk langsung, serta blok tunggal, ganda, dan tiga kali lipat blok disk tidak langsung. Berapa ukuran maksimum file yang bisa dibuat disimpan di dalam ini mengajukan sistem?

14.16 Fragmentasi pada perangkat penyimpanan dapat dihilangkan melalui com- bagian. Perangkat disk pada umumnya tidak memiliki relokasi atau register dasar (seperti sebagai itu digunakan Kapan Penyimpanan adalah ke menjadi kompak), Jadi Bagaimana Bisa Kami memindahkan file? Berikan tiga alasan mengapa melakukan pemadatan dan relokasi fileadalah sering dihindari.

14.17 Menjelaskan Mengapa pencatatan pembaruan metadata memastikan pemulihan dari A mengajukan sistem-item setelah sebuah sistem file menabrak.

14.18 Mempertimbangkan itu mengikuti cadangan skema:

- **Hari 1** . Menyalin ke A cadangan sedang semua file dari itu disk.
- **Hari 2** . Menyalin ke lain sedang semua file berubah sejak hari 1.
- **Hari 3** . Menyalin ke lain sedang semua file berubah sejak hari 1.

Ini berbeda dari jadwal yang diberikan di Bagian 14.7.4 karena memiliki semuanya pencadangan berikutnya menyalin semua file yang diubah sejak pencadangan penuh pertama. Apa kelebihan sistem ini dibandingkan sistem di Bagian 14.7.4? Apa kekurangannya? Apakah operasi pemulihan menjadi lebih mudah atau lebih banyak sulit? Menjelaskan milikmu menjawab.

14.19 Diskusikan keuntungan dan kerugian berhubungan dengan remote sistem file (disimpan di server file) satu set semantik kegagalan yang berbeda dari mereka yang terkait dengan lokal sistem file.

14.20 Apa adalah itu implikasi dari mendukung UNIX konsistensi semantikuntuk bersama mengakses ke file disimpan pada terpencil mengajukan sistem?

CHAPTER 15

Mengajukan-SistemInternal

Seperti yang kita lihat di Bab 13, sistem file menyediakan mekanisme online penyimpanan dan akses ke konten file, termasuk data dan program. Bab ini adalah terutama berkaitan dengan struktur internal dan operasi sistem file. Kami mengeksplorasi secara rinci cara menyusun penggunaan file, mengalokasikan ruang penyimpanan, hingga memulihkan ruang kosong, untuk melacak lokasi data, dan untuk menghubungkan bagian lain dari operasi sistem ke penyimpanan sekunder.

CHAPTER OBJECTIVES

- Delve into the details of file systems and their implementation.
- Explore booting and file sharing.
- Describe remote file systems, using NFS as an example.

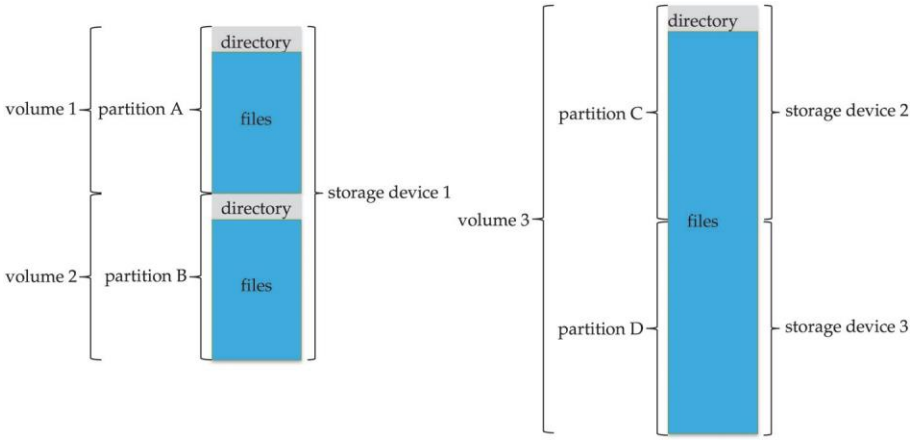
15.1 Mengajukan Sistem

Tentu, TIDAK tujuan umum komputer toko hanya satu mengajukan. Di sana adalah khas ribuan, jutaan, bahkan milyaran file dalam satu komputer. File disimpan di perangkat penyimpanan akses acak, termasuk hard disk drive, disk optik, dantidak mudah menguap Penyimpanan perangkat.

Seperti yang telah Anda lihat di bab sebelumnya, komputer untuk keperluan umum sistem dapat memiliki beberapa perangkat penyimpanan, dan perangkat tersebut dapat dipecah-pecah ke dalam partisi, yang memegang volume, yang di dalam berbelok memegang mengajukan sistem. Bergantung- Di manajer volume, sebuah volume juga dapat mencakup beberapa partisi. Angka 15.1 menunjukkan A khas berkas sistem organisasi.

Sistem komputer mungkin juga memiliki jumlah sistem file yang berbeda-beda, dan itu mengajukan sistem mungkin menjadi dari bervariasi jenis. Untuk contoh, A khas Solaris sistem mungkin memiliki lusinan sistem file dari selusin tipe berbeda, seperti yang ditunjukkan pada gambar berkas sistem daftar di dalam Angka 15.2.

Dalam buku ini, kami hanya membahas sistem file tujuan umum. Ini berharga mencatat, meskipun, itu di sana adalah banyak tujuan khusus mengajukan sistem. Mempertimbangkan itu jenis dari mengajukan sistem di dalam itu Solaris contoh tersebut di atas:



Angka 15.1 A khas penyimpanan perangkat organisasi.

- **tmpfs** —sistem file " sementara " yang dibuat di memori utama yang mudah menguap Dan memiliki -nya isinya terhapus jika itu sistem reboot atau mogok
- **objfs** —sistem file " virtual " (pada dasarnya adalah antarmuka ke kernel yang terlihat menyukai A mengajukan sistem) itu memberi debugger mengakses ke inti simbol
- **ctfs** - sistem file virtual yang menyimpan informasi " kontrak " untuk dikelola usia proses mana yang dimulai ketika sistem melakukan booting dan harus terus berjalan selama operasi
- **lofs** —sebuah " lingkaran kembali " sistem file itu memungkinkan satu sistem file yang akan diakses di dalam tempat dari yang lain
- **procfs** — A maya mengajukan sistem itu hadiah informasi pada semua proses sebagaiA mengajukan sistem
- **ufs, zfs** — tujuan umum mengajukan sistem

Maka, sistem file komputer bisa sangat luas. Bahkan di dalam file sistem, dia adalah berguna ke memisahkan file ke dalam kelompok Dan mengelola Dan bertindak pada itu kelompok. Ini organisasi melibatkan itu menggunakan dari direktori (melihat Bagian 14.3).

15.2 Berkas sistem Pemasangan

Sama seperti file yang harus dibuka sebelum dapat digunakan, sistem file juga harus dibuka dipasang sebelum dapat tersedia untuk proses pada sistem. Lebih spesifik, struktur direktori dapat dibangun dari beberapa sistem file yang berisi volume, yang harus dipasang agar tersedia dalam file- sistem nama ruang angkasa.

Prosedur pemasangannya sangat mudah. Sistem operasi diberikan nama perangkat dan **titik pemasangan** — lokasi dalam struktur file Di mana itu berkas sistem adalah ke dilampirkan. Beberapa Pengoperasian sistem memerlukan itu A berkas sistem jenis menjadi asalkan, ketika yang lain memeriksa itu struktur dari itu perangkat

/	ufs
/perangkat	devfs
/dev	dev
/system/kontrak	ctfs
/proc	proc
/etc/mnttab	mntfs
/etc/svc/volatile	tmpfs
/system/object	objfs
/lib/libc.so.1	lofs
/dev/fd	fd
/var	ufs
/tmp	tmpfs
/var/jalankan	tmpfs
/ pilih	ufs
/zpbge	zfs
/zpbge/cadangan	zfs
/ekspor/beranda	zfs
/var/mail	zfs
/var/spool/mqueue	zfs
/zpbg	zfs
/zpbg/zona	zfs

Angka 15.2 Solaris mengajukan sistem.

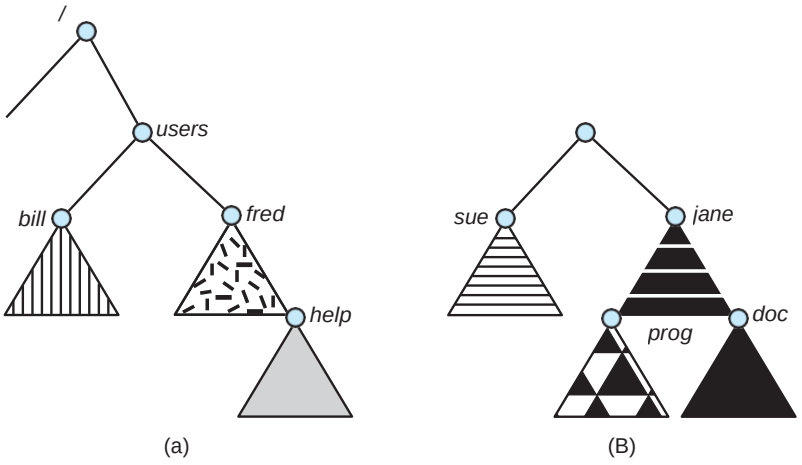
dan menentukan jenis sistem file. Biasanya, titik pemasangannya kosong direktori. Misalnya, pada sistem UNIX , sistem file berisi rumah pengguna direktori mungkin menjadi dipasang sebagai /rumah ; Kemudian, ke mengakses itu direktori struktur dalam sistem file itu, kita dapat mengawali nama direktori dengan /home , sebagai di dalam /rumah/jane . Pemasangan itu mengajukan sistem di bawah /pengguna akan melakukannya hasil di dalam itu jalur nama /pengguna/jane , yang Kami bisa menggunakan ke mencapai itu sama direktori.

Selanjutnya, sistem operasi memverifikasi bahwa perangkat berisi file yang valid sistem. Hal ini terjadi oleh meminta itu driver perangkat untuk membaca itu direktori perangkat Dan memverifikasi itu itu direktori memiliki itu mengharapkan format. Akhirnya, itu Pengoperasian catatan sistem dalam struktur direktorinya bahwa sistem file dipasang di titik pemasangan yang ditentukan. Skema ini memungkinkan sistem operasi untuk melintasi -nya direktori struktur, beralih di antara mengajukan sistem, Dan bahkan mengajukan sistem daribervariasi jenis, sewajarnya.

Untuk mengilustrasikan pemasangan file, perhatikan sistem file yang digambarkan pada Gambar 15.3, Di mana itu segitiga mewakili subpohon dari direktori itu adalah dari minat. Angka 15.3(a) menunjukkan sistem file yang ada, sedangkan Gambar 15.3(b) menunjukkan sistem file yang tidak di-mount volume yang berada di /device/dsk . Pada titik ini, hanya file yang ada sistem file dapat diakses. Gambar 15.4 menunjukkan efek pemasangan volume yang berada di /perangkat/dsk atas /pengguna . Jika volume sudah dilepas, itu mengajukan sistem adalah pulih ke itu situasi digambarkan di dalam Angka 15.3.

Sistem menerapkan semantik untuk memperjelas fungsionalitas.

Misalnya saja sebuah sistem mungkin melarang A gunung lebih A direktori yang berisi file; atau itu mungkin membuat itu

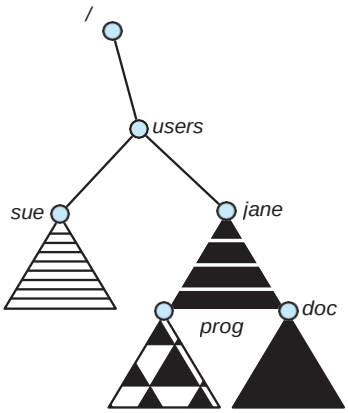


Angka 15.3 Mengajukan sistem. (A) Yang ada sistem. (B) Lepas volume.

sistem file yang dipasang tersedia di direktori itu dan mengaburkan direktori tersebut yang ada file sampai itu mengajukan sistem adalah lepas, mengakhiri itu menggunakan dari itu mengajukan sistem dan mengizinkan akses ke file asli di direktori itu. Sebagai yang lain Misalnya, suatu sistem mungkin mengizinkan sistem file yang sama untuk dipasang berulang kali, pada berbeda gunung poin; atau dia mungkin hanya mengizinkan satu gunung per mengajukan sistem.

Mempertimbangkan itu tindakan dari itu MacOS Pengoperasian sistem. Kapan pun itu sistem menemukan disk untuk pertama kalinya (baik pada saat boot atau saat sistem sedang berjalan berlari), itu MacOS Pengoperasian sistem pencarian untuk A mengajukan sistem pada itu perangkat. Jika ia menemukannya, ia secara otomatis memasang sistem file di bawah /Volumes direktori, menambahkan ikon folder yang diberi label dengan nama sistem file (seperti disimpan di direktori perangkat). Pengguna kemudian dapat mengklik ikon dan dengan demikian menampilkan yang baru file terpasang sistem.

Itu Microsoft jendela keluarga dari Pengoperasian sistem mempertahankan sebuah struktur direktori dua tingkat yang diperluas, dengan perangkat dan volume yang ditetapkan menyetir surat. Setiap volume memiliki A umum grafik direktori struktur terkait



Angka 15.4 Volume dipasang pada /pengguna.

dengan huruf drive-nya. Jalur ke file tertentu berbentuk drive- surat: \ jalur \ ke \ file . Itu lagi terkini versi dari jendela mengizinkan A mengajukan sistem untuk dipasang di mana saja di pohon direktori, seperti halnya UNIX . jendela Pengoperasian sistem secara otomatis menemukan semua perangkat Dan gunung semua terletak mengajukan sistem pada boot waktu. Di dalam beberapa sistem, menyukai UNIX , itu gunung perintahnya eksplisit. File konfigurasi sistem berisi daftar perangkat Dan gunung poin untuk otomatis pemasangan pada boot waktu, Tetapi lainnya gunung mungkinmenjadi dieksekusi secara manual.

Masalah tentang mengajukan sistem pemasangan adalah lebih jauh dibahas di dalam Bagian
15.3 Dan di dalam Bagian C.7.5.

15.3 Partisi Dan Pemasangan

Tata letak disk dapat memiliki banyak variasi, tergantung pada pengoperasiannya perangkat lunak manajemen sistem dan volume. Sebuah disk dapat diiris menjadi beberapa partisi, atau volume dapat menjangkau beberapa partisi pada banyak disk. Itu tata letak yang pertama dibahas di sini, sedangkan tata letak yang kedua, mana yang lebih tepat dipertimbangkan A membentuk dari serangan , adalah tertutupi di dalam Bagian 11.8.

Setiap partisi dapat berupa " mentah " , tidak mengandung sistem file, atau " matang " , mengandung A mengajukan sistem. **Mentah disk** adalah digunakan Di mana TIDAK mengajukan sistem adalah sesuai. Ruang swap UNIX dapat menggunakan partisi mentah, misalnya, karena menggunakan partisi miliknya sendiri format pada disk dan tidak menggunakan sistem file. Demikian pula, beberapa database menggunakan mentah disk Dan format itu data ke setelan milik mereka kebutuhan. Mentah disk Bisa Juga memegang informasi- perkawinan diperlukan oleh disk SERANGAN sistem, seperti sebagai bitmap menunjukkan yang blok adalah dicerminkan Dan yang memiliki berubah Dan membutuhkan ke menjadi dicerminkan. Demikian pula, mentah disk dapat berisi database mini yang menyimpan informasi konfigurasi RAID , seperti disk mana yang menjadi anggota setiap kumpulan RAID . Penggunaan disk mentah dibahas di dalam Bagian 11.5.1.

Jika suatu partisi berisi sistem file yang dapat di-boot— berarti partisi tersebut memiliki sistem file yang benar sistem operasi yang diinstal dan dikonfigurasi— maka partisi juga perlu boot informasi, seperti yang dijelaskan dalam Bagian 11.5.2. Informasi ini memiliki formatnya sendiri, Karena pada boot waktu itu sistem melakukan bukan memiliki itu berkas sistem kode sarat Dan oleh karena itu tidak dapat menafsirkan format sistem file. Sebaliknya, informasi booting adalah biasanya serangkaian blok berurutan yang dimuat sebagai gambar ke dalam memori. Eksekusi gambar dimulai pada lokasi yang telah ditentukan, seperti byte pertama. Gambar ini, itu **bootstrap pemuat** , di dalam berbelok tahu cukup tentang itu berkas sistem struktur ke menjadi mampu ke menemukan Dan memuat itu inti Dan awal dia mengeksekusi.

Boot loader dapat berisi lebih dari sekadar instruksi untuk mem-boot suatu spesifikasi. spesifik Pengoperasian sistem. Untuk contoh, banyak sistem Bisa menjadi **boot ganda** , mengizinkan- ing kita ke Install banyak Pengoperasian sistem pada A lajang sistem. Bagaimana melakukan sistem tahu yang mana yang harus di-boot? Boot loader yang memahami multi-beberapa sistem file dan beberapa sistem operasi dapat menempati ruang boot. Setelah dimuat, itu dapat mem-boot salah satu sistem operasi yang

tersedia di drive. Itu menyetir Bisa memiliki banyak partisi, setiap mengandung A berbeda jenis dari mengajukan sistem Dan A berbeda Pengoperasian sistem. Catatan itu jika sepatu bot pemuat melakukan bukan memahami A tertentu berkas sistem format, sebuah Pengoperasian sistem disimpan pada itu mengajukan sistem adalah bukan dapat di-boot. Ini adalah satu dari itu alasan hanya beberapa mengajukan sistem adalah didukung sebagai akar mengajukan sistem untuk setiap diberikan sistem operasi.

Itu **akar partisi** terpilih oleh itu boot pemuat, yang mengandung itu kernel sistem operasi dan terkadang file sistem lainnya, dipasang di waktu booting. Volume lain dapat dipasang secara otomatis saat boot atau secara manual dipasang nanti, tergantung pada sistem operasinya. Sebagai bagian dari kesuksesan operasi pemasangan, sistem operasi memverifikasi bahwa perangkat berisi a sistem file yang valid. Ia melakukannya dengan meminta driver perangkat untuk membaca perangkat direktori dan memverifikasi bahwa direktori memiliki format yang diharapkan. Jika format adalah tidak sah, itu partisi harus memiliki -nya konsistensi diperiksa Dan mungkin diperbaiki, baik dengan atau tanpa campur tangan pengguna. Terakhir, operasi catatan sistem dalam tabel pemasangan di dalam memori bahwa sistem file dipasang, beserta jenis sistem filenya. Detail fungsi ini bergantung pada itu sistem operasi.

Microsoft berbasis Windows sistem gunung setiap volume di dalam A memisahkan ruang nama, dilambangkan dengan huruf dan titik dua, seperti yang disebutkan sebelumnya. Untuk merekam itu A mengajukan sistem adalah dipasang pada `F:` , untuk contoh, itu Pengoperasian sistem tempat penunjuk ke sistem file di bidang struktur perangkat yang sesuai `F:` . Ketika suatu proses menentukan huruf driver, sistem operasi menemukan penunjuk sistem file yang sesuai dan melintasi struktur direktori di dalamnya perangkat untuk menemukan file atau direktori yang ditentukan. Versi Windows yang lebih baru bisa gunung A mengajukan sistem pada setiap titik di dalam itu yang ada direktori struktur.

Pada UNIX , mengajukan sistem Bisa menjadi dipasang pada setiap direktori. Pemasangan adalah menerapkan- disebutkan oleh pengaturan A bendera di dalam itu dalam kenangan menyalin dari itu inode untuk itu direktori. Bendera tersebut menunjukkan bahwa direktori tersebut adalah titik pemasangan. Sebuah bidang kemudian menunjuk ke entri di tabel pemasangan, yang menunjukkan perangkat mana yang dipasang di sana. Itu entri tabel mount berisi penunjuk ke superblock sistem file aktif itu perangkat. Ini skema memungkinkan itu Pengoperasian sistem ke melintasi - nya direktori struktur, beralih dengan mulus di antara mengajukan sistem dari bervariasi jenis.

15.4 Mengajukan Membagikan

Kemampuan berbagi file sangat diinginkan oleh pengguna yang ingin berkolaborasi Dan ke mengurangi itu upaya diperlukan ke meraih A komputasi sasaran. Karena itu, pengguna- berorientasi Pengoperasian sistem harus menampung itu membutuhkan ke membagikan file di dalam meskipundari itu sifat yang permanen kesulitan.

Di bagian ini, kita memeriksa lebih banyak aspek berbagi file. Kita mulai dengan membahas masalah umum yang muncul ketika banyak pengguna berbagi file. Sekali banyak pengguna diizinkan untuk berbagi file, tantangannya adalah memperluas pembagian file banyak mengajukan sistem, termasuk terpencil mengajukan sistem; Kami membahas itu tantangan sebagai Sehat. Akhirnya, Kami mempertimbangkan Apa ke Mengerjakan tentang bertentangan tindakan terjadi pada file bersama. Misalnya, jika banyak pengguna menulis ke sebuah file, seharusnya semua file penulisan diperbolehkan terjadi, atau haruskah sistem operasi melindungi pengguna tindakan dari satu sama lain?

15.4.1 Banyak Pengguna

Ketika suatu sistem operasi mengakomodasi banyak pengguna, masalah file membagikan, mengajukan penamaan, Dan mengajukan perlindungan menjadi unggul. Diberikan A direktori struktur yang memungkinkan file untuk dibagikan oleh pengguna, sistem harus memediasi mengajukan membagikan. Itu sistem Bisa salah satu mengizinkan A pengguna ke mengakses itu file dari lainnya pengguna

oleh bawaan atau memerlukan itu A pengguna secara khusus menganugerahkan mengakses ke itu file. Ini adalah itu masalah dari mengakses kontrol Dan perlindungan, yang adalah tertutupi di dalam Bagian 13.4.

Ke melaksanakan membagikan Dan perlindungan, itu sistem harus menjaga lagi mengajukan dan atribut direktori daripada yang dibutuhkan pada sistem pengguna tunggal. Meskipun banyak pendekatan telah diambil untuk memenuhi persyaratan ini, sebagian besar sistem telah melakukannya berevolusi untuk menggunakan konsep **pemilik file (atau direktori)** (atau **pengguna**) dan **grup** . Itu pemilik adalah itu pengguna WHO Bisa mengubah atribut Dan menganugerahkan mengakses Dan WHO memiliki itu kontrol terbesar atas file tersebut. Atribut grup mendefinisikan subkumpulan pengguna yang dapat berbagi akses ke file. Misalnya pemilik file pada sistem UNIX Bisa masalah semua operasi pada A mengajukan, ketika anggota dari itu file kelompok Bisa menjalankan satu subset dari operasi tersebut, dan semua pengguna lain dapat menjalankan subset lainnya operasi. Operasi mana yang dapat dijalankan oleh anggota grup Dan lainnya pengguna adalah dapat didefinisikan oleh file itu pemilik.

Pemilik dan ID grup dari file (atau direktori) tertentu disimpan dengan lainnya mengajukan atribut. Kapan A pengguna permintaan sebuah operasi pada A mengajukan, itu pengguna PENGENAL Bisa menjadi dibandingkan dengan itu pemilik atribut ke menentukan jika itu meminta pengguna adalah itu pemilik dari itu mengajukan. Juga, itu kelompok ID s Bisa menjadi dibandingkan. Itu hasil menunjukkan izin mana yang berlaku. Sistem kemudian menerapkan izin tersebut ke yang diminta operasi Dan memungkinkan atau menyangkal dia.

Banyak sistem memiliki beberapa sistem file lokal, termasuk volume a lajang disk atau banyak volume pada banyak terlampir disk. Di dalam ini kasus, pemeriksaan ID dan pencocokan izin sangatlah mudah, begitu filenya sistem dipasang. Namun pertimbangkan disk eksternal yang dapat dipindahkan sistem. Apa jika itu ID s pada itu sistem adalah berbeda? peduli harus menjadi diambil ke menjadi yakin bahwa ID cocok antar sistem ketika perangkat berpindah di antara sistem tersebut atau itu kepemilikan file diatur ulang ketika perpindahan tersebut terjadi. (Misalnya, kita dapat membuat A baru pengguna PENGENAL Dan mengatur semua file pada itu portabel disk ke itu PENGENAL , ke menjadi Tentu TIDAK file adalah secara tidak sengaja dapat diakses ke yang ada pengguna.)

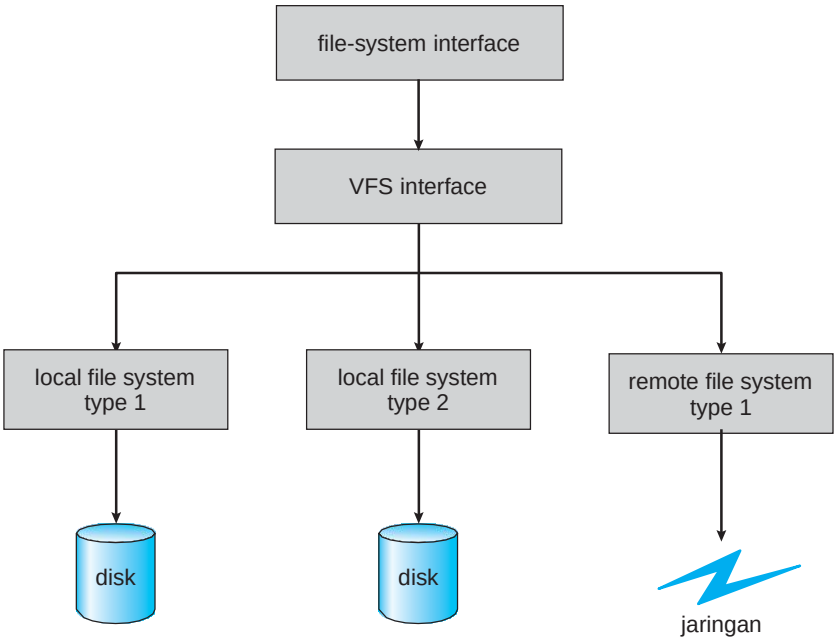
15.5 Maya Mengajukan Sistem

Seperti yang telah kita lihat, sistem operasi modern harus mendukung banyak hal secara bersamaan jenis sistem file. Tapi bagaimana sistem operasi mengizinkan banyak jenis sistem file untuk diintegrasikan ke dalam struktur direktori? Dan bagaimana pengguna bisa berpindah dengan mulus antar tipe sistem file saat mereka menavigasi sistem file ruang angkasa? Kami Sekarang membahas beberapa dari ini penerapan detail.

Sebuah jelas Tetapi kurang optimal metode dari menerapkan banyak jenis dari mengajukan sistem adalah ke menulis direktori Dan mengajukan rutinitas untuk setiap jenis. Alih-alih, Namun, sebagian besar sistem operasi, termasuk UNIX , menggunakan teknik berorientasi objek untuk menyederhanakan tolong, mengatur, Dan memodulasi itu penerapan. Itu menggunakan dari ini metode memungkinkan tipe sistem file yang sangat berbeda untuk diimplementasikan dalam sistem yang sama struktur,

termasuk sistem file jaringan, seperti NFS . Pengguna dapat mengakses file terkandung dalam beberapa sistem file di drive lokal atau bahkan di sistem file tersedia di seberang jaringan.

Struktur dan prosedur data digunakan untuk mengisolasi panggilan sistem dasar fungsionalitas dari detail implementasi. Dengan demikian, implementasi sistem file- pemikiran terdiri dari tiga besar lapisan, sebagai digambarkan secara skematis di dalam Angka
15.5. Itu Pertama lapisan adalah itu berkas sistem antarmuka, berdasarkan pada itu membuka() , membaca() , menulis() , Dan menutup() panggilan Dan pada mengajukan deskriptor.



Angka 15.5 Skema melihat dari A maya berkas sistem.

Itu Kedua lapisan adalah ditelepon itu **maya mengajukan sistem** (**VFS**) lapisan. Itu VFS lapisanmelayani dua penting fungsi:

1. Dia memisahkan file-sistem-generik operasi dari milik mereka penerapan oleh mendefinisikan A membersihkan VFS antarmuka. Beberapa implementasi untuk itu VFS antar wajah dapat hidup berdampingan di mesin yang sama, memungkinkan akses transparan berbeda jenis dari mengajukan sistem dipasang secara lokal.
2. Dia menyediakan A mekanisme untuk secara unik mewakili A mengajukan selama A bersih- bekerja. Itu VFS adalah berdasarkan pada A representasi file struktur, ditelepon A **simpul** , yang berisi penanda numerik untuk file unik di seluruh jaringan. (Inode UNIX unik hanya dalam satu sistem file.) Jaringan- keunikan yang luas diperlukan untuk mendukung sistem file jaringan. Itu kernel memelihara satu struktur vnode untuk setiap node aktif (file atau direktoricerita).

Dengan demikian, VFS membedakan file lokal dari file jarak jauh, dan file lokal membedakannya lebih jauh terpandang menurut ke milik mereka berkas sistem jenis.

VFS mengaktifkan operasi spesifik sistem file untuk menangani permintaan lokal sesuai dengan tipe sistem filenya dan memanggil prosedur protokol NFS (atau lainnya protokol Prosedur untuk lainnya jaringan mengajukan sistem) untuk terpencil permintaan. Mengajukan menangani adalah dibangun dari itu relevan vnodes Dan adalah lulus sebagai berdebat-catatan ke ini Prosedur. Itu lapisan menerapkan itu berkas sistem jenis atau itusistem file jarak jauh protokol adalah itu ketiga lapisan dari itu Arsitektur.

Mari kita periksa secara singkat arsitektur VFS di Linux. Empat objek utama jenis didefinisikan oleh Linux VFS adalah:

- Itu **inode objek** , yang mewakili sebuah individu mengajukan
- Itu **file objek** , yang mewakili sebuah membuka mengajukan
- Itu **superblok objek** , yang mewakili sebuah seluruh mengajukan sistem
- Itu **kedokteran gigi objek** , yang mewakili sebuah individu direktori pintu masuk

Untuk masing-masing dari empat tipe objek ini, VFS mendefinisikan serangkaian operasi yang mungkin dilaksanakan. Setiap objek dari salah satu tipe ini berisi penunjuk ke sebuah tabel fungsi. Tabel fungsi mencantumkan alamat fungsi sebenarnya itu melaksanakan yang ditentukan operasi untuk itu tertentu obyek. Untuk contoh, sebuah disingkat API untuk beberapa dari itu operasi untuk itu mengajukan obyek termasuk:

- ke dalam membuka(. . .) — Membuka A mengajukan.
- ke dalam menutup(. . .) — Menutup sebuah sudah terbuka mengajukan.
- ukuran T membaca(. . .) — Membaca dari A mengajukan.
- ukuran T menulis(. . .) — Menulis ke A mengajukan.
- ke dalam mmap(. . .) — Peta memori A mengajukan.

Sebuah penerapan dari itu mengajukan obyek untuk A spesifik mengajukan jenis adalah diperlukan ke menerapkan- ment setiap fungsi yang ditentukan dalam definisi objek file. (Yang lengkap definisi objek file ditentukan dalam operasi file struct file , yang adalah terletak di dalam itu mengajukan /usr/include/linux/fs.h .)

Dengan demikian, lapisan perangkat lunak VFS dapat melakukan operasi pada salah satunya objek dengan memanggil fungsi yang sesuai dari tabel fungsi objek, tanpa memiliki ke tahu di dalam maju tepat Apa baik dari obyek dia adalah berurusan dengan. VFS tidak mengetahui, atau peduli, apakah sebuah inode mewakili file disk , file direktori, atau file jarak jauh. Fungsi yang sesuai untuk read() file itu operasi akan selalu menjadi pada itu sama tempat di dalam -nya fungsi meja, Dan itu VFS lapisan perangkat lunak akan memanggil fungsi itu tanpa mempedulikan bagaimana data sebenarnyamembaca.

15.6 Terpencil Mengajukan Sistem

Dengan itu kedatangan dari jaringan (Bab 19), komunikasi di antara terpencil com- puter menjadi mungkin. Jaringan memungkinkan pembagian sumber daya tersebar di seluruh kampus atau bahkan di seluruh dunia. Salah satu sumber yang jelas untuk dibagikan adalah data di dalam itu bentuk file.

Melalui itu evolusi dari jaringan Dan mengajukan teknologi, terpencil file sharing metode telah berubah. Metode yang diterapkan pertama kali melibatkan manual mentransfer file antar mesin melalui program seperti ftp . Jurusan kedua metode menggunakan **file terdistribusi sistem** (**DFS**), di mana direktori jarak jauh berada bisa dilihat dari A lokal mesin. Di dalam beberapa cara, itu ketiga metode, itu **Dunia LebarWeb** , adalah pengembalian ke yang pertama. Browser diperlukan untuk mendapatkan akses ke file jarak jauh, dan operasi terpisah (pada dasarnya pembungkus untuk ftp) digunakan untuk mentransfer file. Semakin banyak komputasi awan (Bagian

1.10.5) yang digunakan mengajukan berbagi juga.

ftp adalah digunakan untuk keduanya anonim Dan diautentikasi mengakses. **Anonim akses** memungkinkan pengguna untuk mentransfer file tanpa memiliki akun di remote sistem. World Wide Web menggunakan pertukaran file anonim hampir secara eksklusif dengan penuh semangat. DFS melibatkan A banyak lebih ketat integrasi di antara itu mesin itu adalah mengakses itu terpencil file Dan itu mesin menyediakan itu file. Ini integrasi menambahkan kompleksitas, sebagai Kami menggambarkan di dalam ini bagian.

15.6.1 Itu Klien-pelayan Model

Terpencil mengajukan sistem mengizinkan A komputer ke gunung satu atau lagi mengajukan sistem dari satu atau lagi terpencil mesin. Di dalam ini kasus, itu mesin mengandung itu file adalah **servernya** , dan mesin yang mencari akses ke file tersebut adalah **kliennya** . Itu hubungan klien-server biasa terjadi pada mesin jaringan. Umumnya, server menyatakan bahwa sumber daya tersedia untuk klien dan menentukan dengan tepat sumber daya mana (dalam hal ini, file mana) dan klien mana. Sebuah server Bisa melayani banyak klien, Dan A klien Bisa menggunakan banyak server, tergantung pada itu penerapan detail dari yang diberikan klien- server fasilitas.

Server biasanya menentukan file yang tersedia pada volume atau direktori tingkat. Identifikasi klien lebih sulit. Seorang klien dapat ditentukan oleh jaringan bekerja nama atau lainnya pengenal, seperti sebagai sebuah AKU P alamat, Tetapi ini Bisa menjadi **dipalsukan** , atau ditiru. Sebagai akibat dari spoofing, klien yang tidak sah dapat diizinkan mengakses ke itu server. Lagi aman solusi termasuk aman autentikasi dari itu klien melalui terenkripsi kunci. Sayangnya, dengan keamanan datang banyak tantangan, termasuk memastikan kompatibilitas klien dan server (mereka harus menggunakan algoritma enkripsi yang sama) dan keamanan pertukaran kunci (kunci yang disadap dapat lagi mengizinkan akses yang tidak sah). Karena sulitnya penyelesaiannya ini masalah, tidak aman autentikasi metode adalah paling umumnya digunakan.

Dalam kasus UNIX dan sistem file jaringannya (NFS), otentikasi diperlukan tempatkan melalui informasi jaringan klien, secara default. Dalam skema ini, ID pengguna di klien dan server harus cocok. Jika tidak, server akan melakukannya tidak dapat menentukan hak akses ke file. Perhatikan contoh seorang pengguna yang punya ID 1000 di klien dan 2000 di server. Permintaan dari klien ke server untuk file tertentu tidak akan ditangani dengan tepat, seperti server akan menentukan apakah pengguna 1000 memiliki akses ke file daripada mendasarkannya itu tekad pada itu nyata pengguna PENGENAL dari 2000. Mengakses adalah dengan demikian diberikan atau ditolak berdasarkan pada salah autentikasi informasi. Itu server harus memercayai itu klien ke hadiah itu benar pengguna PENGENAL . Catatan itu itu NFS protokol mengizinkan banyak ke banyak hubungan. Artinya, banyak server dapat menyediakan file ke banyak klien. Nyatanya, mesin tertentu dapat menjadi server untuk beberapa klien NFS dan klien lainnyaNFS server.

Setelah sistem file jarak jauh dipasang, permintaan operasi file dikirim atas nama pengguna melalui jaringan ke server melalui protokol DFS . Biasanya, permintaan buka file dikirim bersama dengan ID pengguna yang meminta. Server kemudian menerapkan pemeriksaan akses standar untuk menentukan apakah pengguna memiliki kredensial untuk mengakses file dalam mode yang diminta. Permintaannya adalah keduanya diperbolehkan

atau ditolak. Jika diizinkan, pegangan file dikembalikan ke aplikasi klien. kation, Dan itu aplikasi Kemudian Bisa melakukan membaca, menulis, Dan lainnya operasi pada file. Klien menutup file ketika akses selesai. Pengoperasian sistem mungkin menerapkan semantik yang serupa dengan yang digunakan untuk pemasangan sistem file lokal atau mungkin menggunakan berbeda semantik.

15.6.2 Informasi yang Didistribusikan Sistem

Untuk membuat sistem klien-server lebih mudah dikelola, **sistem informasi terdistribusi tems** , juga dikenal sebagai **layanan penamaan terdistribusi** , menyediakan akses terpadu ke informasi yang diperlukan untuk komputasi jarak jauh. **Sistem nama** domain (**DNS**) menyediakan terjemahan nama host-ke-alamat-jaringan untuk seluruh Internet. Sebelum DNS tersebar luas, ada file yang berisi informasi yang sama dikirim melalui email atau ftp antara semua host jaringan. Jelas, metode ini- ogy dulu bukan terukur! DNS adalah lebih jauh dibahas di dalam Bagian 19.3.1.

Sistem informasi terdistribusi lainnya menyediakan *nama pengguna/kata sandi/pengguna Ruang ID / ID grup* untuk fasilitas terdistribusi. Sistem UNIX telah menggunakan a berbagai macam metode informasi yang didistribusikan. Sun Microsystems (sekarang bagian dari Peramal Perusahaan) diperkenalkan **kuning halaman** (sejak diganti namanya **jaringan layanan informasi** , atau **NIS**), dan sebagian besar industri mengadopsi penggunaannya. Dia memusatkan penyimpanan nama pengguna, nama host, informasi printer, dan sejenisnya. Sayangnya, ia menggunakan metode autentikasi yang tidak aman, termasuk pengiriman kata sandi pengguna tidak terenkripsi (dalam teks yang jelas) dan mengidentifikasi host berdasarkan alamat IP . NIS + dari Sun adalah pengganti NIS yang jauh lebih aman , namun lebih dari itu rumit dan tidak luas diadopsi.

Dalam kasus **sistem file Internet umum Microsoft** (**CIFS**), jaringan informasi digunakan bersama dengan otentikasi pengguna (nama pengguna dan kata sandi) ke membuat A jaringan Gabung itu itu server kegunaan ke memutuskan apakah untuk mengizinkan atau menolak akses ke sistem file yang diminta. Untuk otentikasi ini ke menjadi sah, itu pengguna nama harus cocok dari mesin ke mesin (sebagai dengan NFS). Microsoft menggunakan **direktori aktif** sebagai struktur penamaan terdistribusi menyediakan satu ruang nama untuk pengguna. Setelah didirikan, didistribusikan penamaan fasilitas digunakan oleh semua klien dan server untuk mengautentikasi pengguna melalui protokol otentikasi jaringan **Kerberos** versi Microsoft (https://web.mit.edu/kerberos/).

Industri ini bergerak menuju penggunaan **akses direktori ringan protokol** (**LDAP**) sebagai mekanisme penamaan terdistribusi yang aman. Faktanya, aktif direktori didasarkan pada LDAP . Oracle Solaris dan sebagian besar operasi besar lainnya sistem menyertakan LDAP dan memungkinkannya digunakan untuk otentikasi pengguna sebagai serta pengambilan informasi di seluruh sistem, seperti ketersediaan printer. Menurut pikiran, satu didistribusikan LDAP direktori bisa menjadi digunakan oleh sebuah organisasi ke toko semua pengguna Dan sumber informasi untuk semua itu organisasi komputer. Hasilnya adalah sistem masuk tunggal yang aman bagi pengguna, yang akan memasuki sistem mereka autentikasi informasi sekali untuk mengakses ke semua komputer di dalam itu organisasi- isasi. Hal ini juga akan memudahkan upaya sistem-administrasi dengan menggabungkannya menjadi satu lokasi, informasi itu adalah saat ini berserakan di dalam bermacam-macam file pada setiap sistem atau di dalam berbeda didistribusikan informasi jasa.

15.6.3 Kegagalan Mode

Sistem file lokal bisa gagal karena berbagai alasan, termasuk kegagalan file menyetir mengandung itu mengajukan sistem, korupsi dari itu direktori struktur atau lainnya informasi manajemen disk (secara kolektif disebut **metadata**), pengontrol disk kegagalan, kegagalan kabel, dan kegagalan

adaptor host. Pengguna atau administrator sistem kegagalan juga dapat menyebabkan file hilang atau seluruh direktori atau volume hilang dihapus. Banyak dari ini kegagalan akan menyebabkan A tuan rumah ke menabrak Dan sebuah kesalahan kondisi ke menjadi ditampilkan, Dan manusia intervensi mungkin menjadi diperlukan ke memperbaiki itu kerusakan.

Terpencil mengajukan sistem memiliki bahkan lagi kegagalan mode. Karena dari itu kompleksitas dari sistem jaringan Dan itu diperlukan interaksi di antara terpencil mesin, masih banyak lagi masalah yang dapat mengganggu pengoperasian yang benar sistem file jarak jauh. Dalam kasus jaringan, jaringan dapat terganggu di antara dua tuan rumah. Seperti interupsi Bisa hasil dari perangkat keras kegagalan, miskin perangkat keras konfigurasi, atau jaringan penerapan masalah. Meskipun beberapa jaringan memiliki ketahanan bawaan, termasuk beberapa jalur di antaranya tuan rumah, banyak yang tidak. Kegagalan apa pun dapat mengganggu aliran DFS perintah.

Pertimbangkan klien yang sedang menggunakan sistem file jarak jauh. Ada file yang terbuka dari host jarak jauh; di antara aktivitas lainnya, mungkin menjalankan direktori pencarian untuk membuka file, membaca atau menulis data ke file, dan menutup file. Sekarang mempertimbangkan A partisi dari itu jaringan, A menabrak dari itu pelayan, atau bahkan A dijadwalkan penutupan server. Tiba-tiba, sistem file jarak jauh tidak lagi dapat dijangkau. Skenario ini agak umum, sehingga tidak sesuai untuk klien sistem ke bertindak sebagai dia akan jika A lokal mengajukan sistem adalah hilang. Lebih tepatnya, itu sistem Bisa menghentikan semua operasi pada server yang hilang atau menunda operasi hingga server kembali dapat dijangkau. Semantik kegagalan ini didefinisikan dan diimplementasikan sebagai bagian dari protokol sistem file jarak jauh. Penghentian semua operasi bisa hasil di dalam pengguna kekalahan data— dan kesabaran. Dengan demikian, paling DFS protokol salah satu menegakkan atau mengizinkan penundaan operasi sistem file ke host jarak jauh, dengan harapan itu itu terpencil tuan rumah akan menjadi tersedia lagi.

Ke melaksanakan ini baik dari pemulihan dari kegagalan, beberapa baik dari **negara informasi- mation** dapat dipertahankan pada klien dan server. Jika kedua server dan klien mempertahankan pengetahuan tentang aktivitas mereka saat ini dan membuka file, lalu mereka Bisa mulus pulih dari A kegagalan. Di dalam itu situasi Di mana itu server mogok tetapi harus menyadari bahwa ia telah memasang sistem file yang diekspor dari jarak jauh dan dibuka file, NFS Versi: kapan 3 dibutuhkan A sederhana mendekati, menerapkan A **tanpa kewarganegaraan** DFS . Intinya, ini mengasumsikan bahwa permintaan klien untuk membaca atau menulis file tidak akan terjadi telah terjadi kecuali sistem file telah dipasang dari jarak jauh dan file tersebut telah pernah sebelumnya membuka. Itu NFS protokol membawa semua itu informasi diperlukan untuk menemukan file yang sesuai dan melakukan operasi yang diminta. Demikian pula, itu tidak melacak klien mana yang telah memasang volume yang diekspor, lagi dengan asumsi bahwa jika permintaan masuk, itu harus sah. Meskipun ini tanpa kewarganegaraan Pendekatan ini membuat NFS tangguh dan mudah diimplementasikan, dan juga membuatnya tidak aman. Misalnya, permintaan baca atau tulis palsu dapat diizinkan oleh server NFS . Masalah-masalah ini diatasi dalam Versi NFS standar industri 4, di mana NFS dibuat stateful untuk meningkatkan keamanan, kinerja, dan Kegunaan.

15.7 Konsistensi Semantik

Konsistensi semantik mewakili sebuah penting kriteria untuk mengevaluasi setiap mengajukan sistem itu mendukung mengajukan membagikan. Ini semantik menentukan Bagaimana banyak pengguna suatu sistem adalah mengakses file bersama secara bersamaan. Secara khusus, mereka menentukan

ketika modifikasi data oleh satu pengguna akan dapat diamati oleh pengguna lain. Ini semantik adalah khas dilaksanakan sebagai kode dengan itu mengajukan sistem.

Semantik konsistensi berhubungan langsung dengan proses sinkronisasi algoritma dari Bab 6. Namun, itu kompleks algoritma dari itu bab cenderung

tidak untuk diimplementasikan dalam kasus file I/O karena latensi yang besar dan lambat transfer tarif dari disk Dan jaringan. Untuk contoh, tampil sebuah atom transaksi ke disk jarak jauh dapat melibatkan beberapa komunikasi jaringan, beberapa disk membaca dan menulis, atau keduanya. Sistem yang mencoba set lengkap fungsi cenderung berkinerja buruk. Implementasi kompleks yang berhasil membagikan semantik Bisa menjadi ditemukan di dalam itu Andrew mengajukan sistem.

Untuk pembahasan berikut, kita asumsikan bahwa serangkaian akses file (itu adalah, membaca dan menulis) yang dicoba oleh pengguna ke file yang sama selalu terlampir antara terbuka() dan operasi tutup() . Serangkaian mengakses antara operasi open () dan close() membentuk sebuah **file sesi** . Untuk menggambarkan konsep, Kami sketsa beberapa menonjol contoh dari konsistensi semantik.

15.7.1 UNIX Semantik

Itu UNIX mengajukan sistem (Bab 19) kegunaan itu mengikuti konsistensi semantik:

- Menulis ke sebuah membuka mengajukan oleh A pengguna adalah bisa dilihat langsung ke lainnya pengguna WHO memiliki ini mengajukan membuka.
- Salah satu mode berbagi memungkinkan pengguna untuk berbagi penunjuk lokasi saat ini ke dalam file. Jadi, memajukan penunjuk oleh satu pengguna akan mempengaruhi semua orang berbagi pengguna. Di sini, sebuah file memiliki satu gambar yang menyisipkan semua akses, tanpa memedulikan dari asal mereka.

Di dalam itu UNIX semantik, A mengajukan adalah terkait dengan A lajang fisik gambar itu diakses sebagai sumber daya eksklusif. Pertentangan atas gambar tunggal ini menyebabkan penundaan di dalam pengguna proses.

15.7.2 Sidang Semantik

Itu Andrew mengajukan sistem (Buka AFS) kegunaan itu mengikuti konsistensi semantik:

- Penulisan ke file yang terbuka oleh pengguna tidak langsung terlihat oleh pengguna lainitu punya sama mengajukan membuka.
- Setelah file ditutup, perubahan yang dilakukan hanya terlihat di sesi awal-ing Nanti. Sudah membuka contoh dari itu mengajukan Mengerjakan bukan mencerminkan ini perubahan.

Menurut semantik ini, sebuah file dapat diasosiasikan sementara dengan beberapa file (mungkin berbeda) gambar-gambar pada itu sama waktu. Akibatnya, banyak pengguna adalah diizinkan ke melakukan keduanya membaca Dan menulis mengakses secara bersamaan pada milik mereka gambar-gambar file, tanpa penundaan. Hampir tidak ada batasan yang diterapkan pada penjadwalan mengakses.

15.7.3 File Bersama yang Tidak Dapat Diubah Semantik

Pendekatan yang unik adalah pendekatan **file bersama yang tidak dapat**

diubah. Setelah file dideklarasikan sebagai bersama oleh -nya pencipta, dia tidak bisa menjadi diubah. Sebuah kekal mengajukan memiliki dua kunci properti: namanya tidak boleh digunakan kembali, dan isinya tidak boleh diubah. Jadi, nama file yang tidak dapat diubah menandakan bahwa konten file tersebut adalah tetap. Itu penerapan dari ini semantik di dalam A didistribusikan sistem (Bab 19) adalah sederhana, Karena pembagian itu adalah berdisiplin (hanya baca).

15.8 NFS

Sistem file jaringan adalah hal yang lumrah. Mereka biasanya terintegrasi dengan struktur direktori keseluruhan dan antarmuka sistem klien. NFS adalah contoh yang baik dari file jaringan klien-server yang banyak digunakan dan diimplementasikan dengan baik sistem. Di Sini, Kami menggunakan dia sebagai sebuah contoh ke mengeksplorasi itu penerapan detail darijaringan mengajukan sistem.

NFS adalah keduanya sebuah penerapan Dan A spesifikasi dari A perangkat lunak sistem untuk mengakses terpencil file lintas LAN s (atau bahkan WAN). NFS adalah bagian dari ONC+ , yang paling UNIX vendor Dan beberapa komputer Pengoperasian sistem mendukung. Itu implementasi- tion dijelaskan Di Sini adalah bagian dari itu Solaris Pengoperasian sistem, yang adalah A diubah versi UNIX SVR4 . Ia menggunakan TCP atau UDP/IP protokol (tergantung pada jaringan interkoneksi). Spesifikasi dan implementasinya adalah terjalin dalam deskripsi kami tentang NFS . Kapan pun detail diperlukan, kami mengacu pada implementasi Solaris; kapan pun deskripsinya bersifat umum, itu berlaku untuk itu spesifikasi Juga.

Ada beberapa versi NFS , yang terbaru adalah Versi 4. Di sini, Kami menggambarkan Versi: kapan 3, yang adalah itu Versi: kapan paling umumnya dikerahkan.

15.8.1 Ringkasan

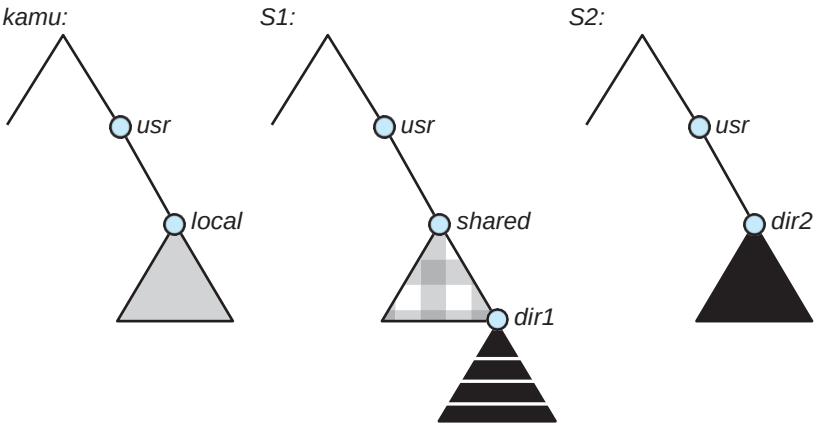
NFS dilihat A mengatur dari saling berhubungan stasiun kerja sebagai A mengatur dari mandiri mesin dengan sistem file independen. Tujuannya adalah untuk memungkinkan beberapa derajat membagikan di antara ini mengajukan sistem (pada eksplisit meminta) di dalam A transparan tata krama. Berbagi didasarkan pada hubungan klien-server. Sebuah mesin mungkin, dan sering adalah, keduanya A klien Dan A server. Membagikan adalah diizinkan di antara setiap pasangan dari mesin. Ke memastikan mesin kemerdekaan, membagikan dari A terpencil mengajukan sistem mempengaruhi hanyaitu klien mesin dan tidak ada yang lain mesin.

Jadi itu A terpencil direktori akan menjadi dapat diakses di dalam A transparan tata krama dari mesin tertentu - katakanlah, dari ***M1*** - klien dari mesin tersebut harus membawanya terlebih dahulu keluar operasi pemasangan. Semantik operasi melibatkan pemasangan a direktori jarak jauh melalui direktori sistem file lokal. Setelah operasi pemasangan asi adalah lengkap, itu dipasang direktori terlihat menyukai sebuah integral subpohon dari itu sistem file lokal, menggantikan subpohon yang turun dari direktori lokal. Itu direktori lokal menjadi nama root dari direktori yang baru dipasang. Spesifikasi direktori jarak jauh sebagai argumen untuk operasi mount tidak dilakukan secara transparan; lokasi (atau nama host) dari direktori jarak jauh harus disediakan. Namun, sejak saat itu, pengguna di mesin ***M1*** dapat mengakses file di dalam itu terpencil direktori di dalam A sama sekali transparan tata krama.

Untuk mengilustrasikan pemasangan file, perhatikan sistem file yang digambarkan pada Gambar 15.6, di mana segitiga mewakili subpohon direktori yang diminati. Itu gambar menunjukkan tiga sistem file independen mesin bernama ***U*** , ***S1*** , dan ***S2*** . Pada titik ini, di setiap mesin, hanya file lokal yang dapat diakses. Angka 15.7(a) menunjukkan efek pemasangan S1:/usr/shared pada U:/usr/local . Gambar ini menggambarkan pandangan pengguna di ***U*** terhadap sistem file mereka. Setelah gunung adalah menyelesaikan, mereka Bisa mengakses setiap

mengajukan di dalam itu dir1 direktori menggunakan itu awalan / usr/lokal/dir1 . Itu asli direktori /usr/lokal pada itu mesin adalah TIDAK lebih lama bisa dilihat.

Tunduk pada akreditasi hak akses, sistem file apa pun, atau direktori apa pun di dalam A mengajukan sistem, Bisa menjadi dipasang dari jarak jauh pada atas dari setiap lokal direktori.

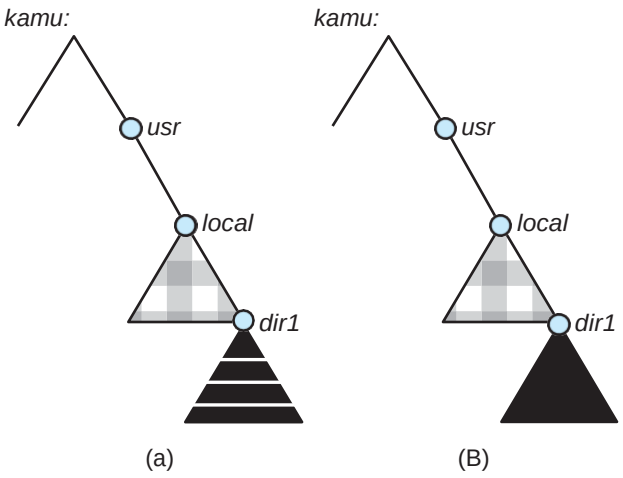


Angka 15.6 Tiga mandiri mengajukan sistem.

Tanpa disk stasiun kerja Bisa bahkan gunung milik mereka memiliki akar dari server. Bertingkat gunung adalah Juga diizinkan di dalam beberapa NFS implementasi. Itu adalah, A mengajukan sistem dapat dipasang melalui sistem file lain yang dipasang dari jarak jauh, bukan lokal. A mesin adalah terpengaruh oleh hanya itu gunung itu dia memiliki diri dipanggil. Pemasangan A sistem file jarak jauh tidak memberikan klien akses ke sistem file lain yang tadinya, secara kebetulan, dipasang pada sistem file sebelumnya. Jadi, mekanisme pemasangannyamelakukan bukan pameran a transitivitas Properti.

Pada Gambar 15.7(b), kami mengilustrasikan cascading mount. Gambar tersebut menunjukkan hasil pemasangan S2:/usr/dir2 melalui U:/usr/local/dir1 , yang sudah dipasang dari jarak jauh dari ***S1.*** Pengguna dapat mengakses file dalam dir2 di ***U*** menggunakan awalan /usr/local/dir1 . Jika A bersama mengajukan sistem adalah dipasang lebih A milik pengguna rumah direktori pada semua mesin di dalam A jaringan, itu pengguna Bisa catatan ke dalam setiap stasiun kerja Dan mendapatkan miliknya atau dia rumah lingkungan. Ini Properti izin pengguna mobilitas.

Satu dari itu desain sasaran dari NFS dulu ke beroperasi di dalam A heterogen mengepung- ment dari berbeda mesin, Pengoperasian sistem, Dan jaringan ilmu bangunan. Itu



Angka 15.7 Pemasangan di dalam NFS. (A) Gunung. (B) Bertingkat gunung.

NFS spesifikasi adalah mandiri dari ini media. Ini kemerdekaan adalah dicapai melalui penggunaan primitif RPC yang dibangun di atas representasi data eksternal tion (XDR) protokol digunakan di antara dua implementasi-independen antarmuka. Oleh karena itu, jika mesin heterogen dan sistem file sistem berfungsi dengan baik dihubungkan ke NFS , sistem file dari tipe yang berbeda dapat dipasang baik secara lokal Dan dari jarak jauh.

Spesifikasi NFS membedakan layanan yang disediakan oleh a mekanisme mount dan layanan akses file jarak jauh yang sebenarnya. Oleh karena itu, dua memisahkan protokol adalah ditentukan untuk ini jasa: A gunung protokol Dan A pro- tocol untuk akses file jarak jauh, **protokol NFS** . Protokol ditentukan sebagai set dari RPC s. Ini RPC s adalah itu bangunan blok digunakan ke melaksanakan transparanterpencil akses berkas.

15.8.2 Itu Gunung Protokol

Itu **gunung protokol** menetapkan itu awal logis koneksi di antara A server dan seorang klien. Di Solaris, setiap mesin memiliki proses server, di luar kernel,tampil itu fungsi protokol.

Jumlah operasi termasuk itu nama dari itu terpencil direktori ke menjadi dipasang Dan itu nama dari itu server mesin menyimpan dia. Itu gunung meminta dipetakan ke RPC yang sesuai dan diteruskan ke server mount berlari pada itu spesifik server mesin. Itu server mempertahankan sebuah **ekspor daftar** yang menentukan sistem file lokal yang diekspor untuk pemasangan, bersama dengan nama dari mesin itu adalah diizinkan ke gunung mereka. (Di dalam Solaris, ini daftar adalah itu
`/etc/dfs/dfstab` , yang Bisa menjadi diedit hanya oleh A pengguna super.) Itu spesifikasi Bisa Juga termasuk mengakses hak, seperti sebagai membaca hanya. Ke menyederhanakan itu pemeliharaan dari daftar ekspor dan tabel pemasangan, skema penamaan terdistribusi dapat digunakan memegang ini informasi Dan membuat dia tersedia ke sesuai klien.

Ingatlah bahwa direktori mana pun dalam sistem file yang diekspor dapat dipasang jarak jauh dengan mesin terakreditasi. Unit komponen adalah direktori tersebut. Ketika server menerima permintaan pemasangan yang sesuai dengan daftar ekspornya, server tersebut mengembalikan ke klien pegangan file yang berfungsi sebagai kunci untuk akses lebih lanjut file dalam sistem file yang terpasang. Pegangan file berisi semua informasi- tion bahwa server perlu membedakan file individual yang disimpannya. Di UNIX istilahnya, pegangan file terdiri dari pengidentifikasi sistem file dan nomor inode ke mengenali itu akurat dipasang direktori di dalam itu diekspor mengajukan sistem.

Itu server Juga mempertahankan A daftar dari itu klien mesin Dan itu sesuai- mencari direktori yang sedang dipasang. Daftar ini digunakan terutama untuk administrasi tujuan— misalnya, untuk memberi tahu semua klien bahwa server sedang down. Hanya melalui tambahan Dan penghapusan dari entri di dalam ini daftar Bisa itu server negara menjadi terpengaruh oleh memasang protokol.

Biasanya, suatu sistem memiliki prakonfigurasi pemasangan statis yang telah ditetapkan saat boot (`/etc/vfstab` di Solaris); namun, tata letak ini dapat diubah. Di dalam Selain prosedur pemasangan sebenarnya, protokol pemasangan mencakup beberapa lainnya Prosedur, seperti sebagai lepaskan Dan kembali ekspor daftar.

15.8.3 NFS Protokol

Protokol NFS menyediakan satu set RPC untuk operasi file jarak jauh. Proses-karena dukung operasi berikut:

- Mencari untuk A mengajukan di dalam A direktori
- Membaca A mengatur dari direktori entri
- Memanipulasi tautan Dan direktori
- Mengakses mengajukan atribut
- Membaca Dan menulis file

Prosedur ini hanya dapat dijalankan setelah file ditangani secara jarak jauh dipasang direktori memiliki pernah didirikan.

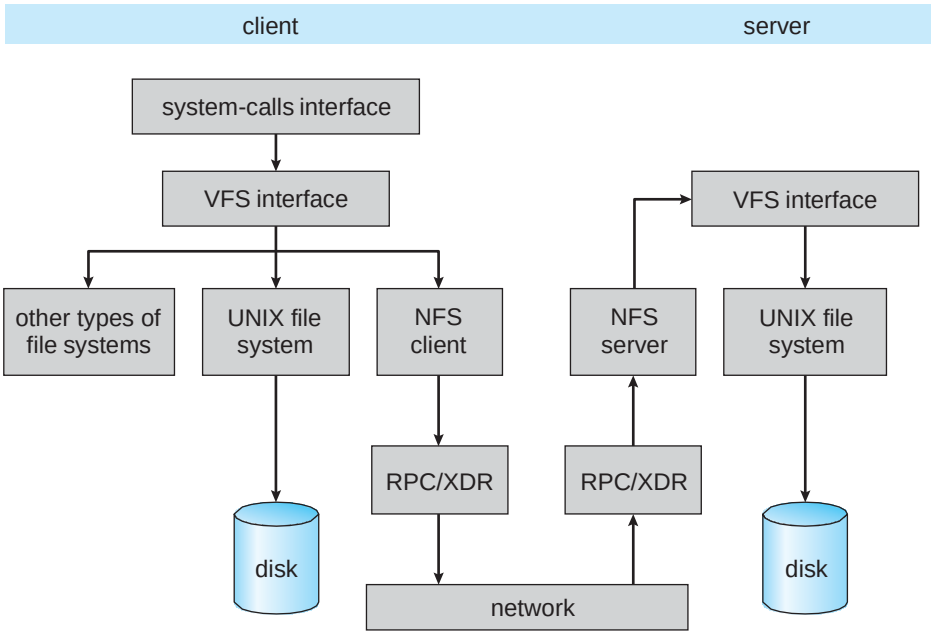
Penghilangan operasi buka dan tutup memang disengaja. Seorang yang menonjol Fitur server NFS adalah server tersebut tidak memiliki kewarganegaraan. Server tidak menyimpan informasi informasi tentang klien mereka dari satu akses ke akses lainnya. Tidak ada persamaannya dengan UNIX tabel file terbuka atau struktur file ada di sisi server. Oleh karena itu, masing-masing meminta memiliki ke menyediakan A penuh mengatur dari argumen, termasuk A unik mengajukan pengidentifikasi Dan sebuah mutlak mengimbangi di dalam itu mengajukan untuk itu sesuai operasi. Itu hasil- desainnya kuat; tidak ada tindakan khusus yang perlu diambil untuk memulihkan server setelah kecelakaan. Operasi file harus menjadi idempoten untuk ini tujuan - itu adalah, operasi yang sama yang dilakukan beberapa kali harus memiliki efek yang sama seperti jika itu hanya dilakukan sekali. Untuk mencapai idempotensi, setiap permintaan NFS memiliki A urutan nomor, memungkinkan itu server ke menentukan jika A meminta memiliki pernahdigandakan atau jika ada hilang.

Mempertahankan daftar klien yang kami sebutkan sepertinya melanggar keadaan tanpa kewarganegaraan pada server. Namun, daftar ini tidak penting untuk kebenaran pengoperasian klien atau server, dan karenanya tidak perlu dipulihkan setelah server crash. Akibatnya, ini mungkin berisi data yang tidak konsisten dan memang demikian diperlakukan hanya sebagai petunjuk.

Implikasi lebih lanjut dari filosofi server tanpa kewarganegaraan dan akibat dari sinkronisasi RPC adalah data yang dimodifikasi (termasuk tipuan dan status blok) harus dikomit ke disk server sebelum hasilnya dikembalikan itu klien. Itu adalah, A klien Bisa cache menulis blok, Tetapi Kapan dia memerah mereka ke server, diasumsikan bahwa mereka telah mencapai disk server. Server harus menulis semua NFS data serentak. Dengan demikian, A server menabrak Dan pemulihan akan tidak terlihat oleh klien; semua blok yang dikelola server untuk klien akan berada utuh. Itu dihasilkan pertunjukan penalti Bisa menjadi besar, Karena itu keuntungan cache hilang. Performanya bisa ditingkatkan dengan menggunakan penyimpanan sendiri cache nonvolatile (biasanya memori cadangan baterai). Pengontrol disk mengakui itu disk menulis Kapan itu menulis adalah disimpan di dalam itu tidak mudah menguap cache. Intinya, host melihat penulisan sinkron yang sangat cepat. Blok-blok ini tetap ada utuh bahkan setelah A sistem menabrak Dan adalah tertulis dari ini stabil penyimpanan ke disksecara berkala.

A lajang NFS menulis prosedur panggilan adalah terjamin ke menjadi atom Dan adalah bukan dicampur dengan panggilan tulis lainnya ke file yang sama. Namun, protokol NFS tidak menyediakan mekanisme kontrol konkurensi. Panggilan sistem write () mungkin dipecah menjadi beberapa penulisan RPC , karena setiap panggilan tulis atau baca NFS Bisa berisi ke atas ke 8 KB dari data Dan UDP paket adalah terbatas ke 1.500 byte. Sebagai A hasil, dua pengguna menulis ke itu sama terpencil mengajukan mungkin mendapatkan milik mereka data bercampur. Klaimnya adalah, karena

manajemen kunci pada dasarnya bersifat stateful, maka sebuah layanan di luar NFS harus menyediakan penguncian (dan Solaris menyediakannya). Pengguna disarankan ke koordinat mengakses ke bersama file menggunakan mekanisme di luar itu cakupan dari NFS .



Angka 15.8 Skema melihat dari itu NFS Arsitektur.

NFS diintegrasikan ke dalam sistem operasi melalui VFS . Sebagai ilustrasi itu Arsitektur, ayo jejak Bagaimana sebuah operasi pada sebuah sudah terbuka terpencil mengajukan adalah ditangani (ikuti contoh pada Gambar 15.8). Klien memulai operasi dengan panggilan sistem biasa. Lapisan sistem operasi memetakan panggilan ini ke VFS operasi pada vnode yang sesuai. Lapisan VFS mengidentifikasi file sebagai remote satu dan memanggil prosedur NFS yang sesuai . Panggilan RPC dilakukan ke layanan NFS lapisan pada itu terpencil server. Ini panggilan adalah disuntikkan kembali ke itu lapisan VFS pada itu terpencil sistem, yang menemukan itu dia adalah lokal Dan memanggil itu sesuai mengajukan- sistem operasi. Jalur ini ditelusuri kembali untuk mengembalikan hasilnya. Sebuah keuntungan dari ini Arsitektur adalah itu itu klien Dan itu server adalah identik; dengan demikian, A mesin mungkin menjadi A klien, atau A pelayan, atau keduanya. Itu sebenarnya melayani pada setiap server adalah dilakukanoleh inti benang.

15.8.4 Nama Jalur Terjemahan

Nama jalur terjemahan di dalam NFS melibatkan itu penguraian dari A jalur nama seperti sebagai
/usr/local/dir1/file.txt ke dalam memisahkan direktori entri, atau komponen:
(1) kami , (2) lokal , Dan (3) dir1 . Nama jalur terjemahan adalah Selesai oleh pemecahan itu jalur ke dalam komponen nama Dan tampil A memisahkan NFS Lihatlah panggilan untuk setiap pasangan nama komponen dan direktori vnode. Sekali menjadi titik pemasangan disilangkan, setiap pencarian komponen menyebabkan RPC terpisah ke server. Ini diperlukan skema path-name-traversal yang mahal, karena tata letak masing-masing ruang nama logis klien bersifat unik, ditentukan oleh tunggangan yang dimiliki klien dilakukan. Akan jauh lebih efisien untuk memberikan nama jalur kepada server dan menerima vnode target setelah titik pemasangan ditemukan. Kapan saja, Namun, di sana mungkin menjadi

lain gunung titik untuk itu tertentu klien dari yang itu tanpa kewarganegaraan server adalah tidak sadar.

Agar pencarian cepat, cache pencarian nama direktori di sisi klien memegang itu vnodes untuk terpencil direktori nama. Ini cache kecepatan ke atas referensi ke file dengan itu sama awal jalur nama. Itu direktori cache adalah dibuang Kapan atribut yang dikembalikan dari server tidak cocok dengan atribut yang di-cache vnode.

Ingatlah bahwa beberapa implementasi NFS memungkinkan pemasangan file jarak jauh sistem di atas sistem file jarak jauh lain yang sudah terpasang (cascading gunung). Ketika klien memiliki cascading mount, bisa lebih dari satu server terlibat dalam traversal nama jalur. Namun, ketika klien melakukan pencarian direktori tempat server memasang sistem file, klien melihatnya mendasari direktori alih-alih dari itu dipasang direktori.

15.8.5 Jarak Jauh Operasi

Dengan pengecualian membuka dan menutup file, hampir terjadi satu-ke-satu korespondensi antara panggilan sistem UNIX biasa untuk operasi file dan protokol NFS RPC s . Dengan demikian, operasi file jarak jauh dapat diterjemahkan secara langsung ke RPC yang sesuai . Secara konseptual, NFS menganut layanan jarak jauh paradigma; Tetapi di dalam praktik, buffering Dan cache teknik adalah dipekerjakan untuk demi kinerja. Tidak ada korespondensi langsung antara remote operasi dan sebuah RPC . Sebaliknya, file diblokir dan mengajukan atribut diambil oleh itu RPC s Dan adalah di-cache secara lokal. Masa depan terpencil operasi menggunakan itu di-cache data, subjek untuk konsistensi kendala.

Ada dua cache: cache atribut file (informasi inode) dan cache cache blok file. Ketika sebuah file dibuka, kernel memeriksa dengan remote server untuk menentukan apakah akan mengambil atau memvalidasi ulang atribut yang di-cache. Itu blok file cache hanya digunakan jika atribut cache yang sesuai sudah habis hingga saat ini. Cache atribut diperbarui setiap kali atribut baru datang server. Atribut yang di-cache, secara default, dibuang setelah 60 detik. Keduanya baca-depan Dan tertunda-menulis teknik adalah digunakan di antara itu server Dan itu klien. Klien tidak membebaskan blok penulisan tertunda sampai server mengonfirmasi hal itu itu data memiliki pernah tertulis ke disk. Tertunda-tulis adalah dipertahankan bahkan Kapan A mengajukan dibuka secara bersamaan, dalam mode yang saling bertentangan. Oleh karena itu, semantik UNIX (Bagian 15.7.1) adalah tidak dilestarikan.

Menyesuaikan sistem untuk kinerja membuat sulit untuk mengkarakterisasi semantik konsistensi NFS . File baru yang dibuat di mesin mungkin tidak terlihat di tempat lain selama 30 detik. Selain itu, menulis ke file di satu situs mungkin atau mungkin tidak terlihat di situs lain yang membuka file ini untuk dibaca. Baru membuka file, amati hanya perubahan yang telah dihapus ke server. Dengan demikian, NFS tidak menyediakan emulasi ketat terhadap semantik UNIX maupun semantik sesi Andrew (Bagian 15.7.2). Terlepas dari kelemahan ini, kegunaan Dan Bagus pertunjukan dari itu mekanisme membuat dia itu paling secara luas digunakan terdistribusi multi-vendor sistem di dalam operasi.

15.9 Ringkasan

- Tujuan umum Pengoperasian sistem menyediakan banyak berkas sistem

jenis, dari tujuan khusus melalui umum.

- Volume mengandung mengajukan sistem Bisa menjadi dipasang ke dalam itu komputer mengajukan-sistem ruang angkasa.
- Tergantung pada sistem operasinya, ruang sistem file mulus (sistem file terpasang terintegrasi ke dalam struktur direktori) atau berbeda (setiap dipasang berkas sistem memilikinya memiliki penamaan).
- Pada paling sedikit satu berkas sistem harus menjadi dapat di-boot untuk itu sistem ke bisa ke awal
  — itu adalah, dia harus berisi sebuah Pengoperasian sistem. Itu boot pemuat adalah berlari Pertama; dia adalah A sederhana program itu adalah mampu ke menemukan itu inti di dalam itu mengajukan sistem, memuat itu, dan memulai eksekusinya. Sistem dapat berisi beberapa partisi yang dapat di-boot, membiarkan itu administrator memilih yang ke berlari pada boot waktu.
- Paling sistem adalah multi-pengguna Dan dengan demikian harus menyediakan A metode untuk mengajukan berbagi-ing Dan mengajukan perlindungan. Sering, file Dan direktori termasuk metadata, seperti sebagai pemilik, pengguna, Dan kelompok mengakses izin.
- Massa penyimpanan partisi adalah digunakan salah satu untuk mentah memblokir masukan/keluaran atau untuk mengajukan sistem. Setiap sistem file berada dalam suatu volume, yang dapat terdiri dari satu volume partisi atau banyak partisi bekerja bersama melalui A volume Pengelola.
- Ke menyederhanakan penerapan dari banyak mengajukan sistem, sebuah Pengoperasian sistem dapat menggunakan pendekatan berlapis, dengan pembuatan antarmuka sistem file virtual mengakses ke mungkin berbeda mengajukan sistem mulus.
- Terpencil mengajukan sistem Bisa menjadi dilaksanakan secara sederhana oleh menggunakan A program seperti sebagai `ftp` atau itu web server Dan klien di dalam itu Dunia Lebar jaring, atau dengan lagi fungsionalitas melalui model klien-server. Permintaan pemasangan dan ID pengguna harus menjadi diautentikasi ke mencegah tidak disetujui mengakses.
- Fasilitas client-server tidak secara native membagi informasi, namun terdistribusi sistem informasi seperti DNS dapat digunakan untuk memungkinkan berbagi, pro- menyediakan ruang nama pengguna, manajemen kata sandi, dan sistem terpadu identifikasi. Misalnya, Microsoft CIFS menggunakan direktori aktif, yang mana mempekerjakan A Versi: kapan dari itu Kerbero jaringan autentikasi protokol ke pro- vide satu set lengkap penamaan dan otentikasi layanan antar komputer di dalam A jaringan.
- Setelah berbagi file dimungkinkan, model semantik konsistensi harus dipilih. sen Dan dilaksanakan ke sedang banyak bersamaan mengakses ke itu sama mengajukan. Model semantik mencakup UNIX , sesi, dan file bersama yang tidak dapat diubahsemantik.
- NFS adalah contoh sistem file jarak jauh, yang menyediakan jahitan bagi klien. lebih sedikit akses ke direktori, file, dan bahkan seluruh sistem file. Sebuah fitur lengkap terpencil mengajukan sistem termasuk A komunikasi protokol dengan terpencil opera- tions Dan nama jalur terjemahan.

Praktik Latihan

15.1 Menjelaskan Bagaimana itu VFS lapisan memungkinkan sebuah Pengoperasian sistem ke mendukung banyak-tip jenis dari mengajukan sistem dengan mudah.

15.2 Mengapa memiliki lagi dibandingkan satu mengajukan sistem jenis pada A diberikan sistem?

15.3 Pada sistem Unix atau Linux yang mengimplementasikan sistem file procfs, menentukan Bagaimana ke menggunakan itu procfs antarmuka ke mengeksplorasi itu proses nama ruang angkasa. Apa aspek dari proses Bisa menjadi dilihat melalui ini antarmuka? Bagaimana akankah informasi yang sama dikumpulkan pada sistem yang tidak memiliki proses mengajukan sistem?

15.4 Mengapa Mengerjakan beberapa sistem mengintegrasikan dipasang mengajukan sistem ke dalam itu akar mengajukan struktur penamaan sistem, sementara yang lain menggunakan metode penamaan tersendiriuntuk dipasang sistem file?

15.5 Diberikan A terpencil mengajukan mengakses fasilitas seperti sebagai ftp , Mengapa adalah terpencil mengajukan sistem menyukai NFS diciptakan?

Lebih jauh Membaca

Itu internal dari itu Sistem UNIX BSD adalah tertutupi di dalam penuh di dalam [McKusick et Al.(2015)]. Detail tentang mengajukan sistem untuk Linux Bisa menjadi ditemukan di dalam [Cinta (2010)]. Sistem file jaringan (NFS) dibahas dalam [Callaghan (2000)]. Versi NFS- sion 4 adalah A standar dijelaskan pada http://www.ietf.org/rfc/rfc3530.txt . [Pengusiran-hout (1991)] berdiskusi itu peran dari didistribusikan negara di dalam berjaringan mengajukan sistem. NFS Dan itu UNIX mengajukan sistem (UFS) adalah dijelaskan di dalam [Mauro Dan McDougall(2007)].

Protokol otentikasi jaringan Kerberos dieksplorasi di https://web.mit.edu/kerberos/ .

Bibliografi

[Callaghan (2000)] B. Callaghan, *NFS diilustrasikan* , Addison-Wesley (2000).

[Cinta (2010)] R. Love, *Pengembangan Kernel Linux,* Edisi Ketiga, Pengembang Perpustakaan (2010).

[Mauro dan McDougall (2007)] J. Mauro dan R. McDougall, *Solaris Internal: Inti Inti Arsitektur* , Pembantu tukang Aula (2007).

[McKusick dkk. (2015)] MK McKusick, GV Neville-Neil, dan RNM Wat- putra, *Itu Desain Dan Penerapan dari itu FreeBSD UNIX Pengoperasian Sistem- Kedua Edisi* , Pearson (2015).

[Penggulingan (1991)] J.Osterhout. " Peran Negara Terdistribusi " . *Di CMU Ilmu Komputer: Peringatan HUT ke-25* , RF Rashid, Ed., Addison- Wesley (1991).

Bab 15 Latihan

15.6 Asumsikan bahwa dalam augmentasi tertentu dari program akses file jarak jauh toko, setiap klien mempertahankan A nama cache itu cache terjemahan dari nama file ke pegangan file yang sesuai. Masalah apa yang harus kita ambil ke dalam akun di menerapkan itu cache nama?

15.7 Mengingat sistem file terpasang dengan operasi penulisan sedang berlangsung, dan a sistem crash atau listrik mati, apa yang harus dilakukan sebelum sistem file dipasang ulang jika: (a) Sistem file tidak terstruktur log? (b) Berkas sistem adalah terstruktur log?

15.8 Mengapa Mengerjakan Pengoperasian sistem gunung itu akar mengajukan sistem secara otomatis padaboot waktu?

15.9 Mengapa sistem operasi memerlukan sistem file selain root dipasang?