Nicholas Tanaka

203400024 / Operating system

Ch 5

5.1 Di Bagian 5.4, kami menyebutkan bahwa menonaktifkan interupsi sering dapat mempengaruhi jam sistem. Jelaskan mengapa hal ini dapat terjadi dan bagaimana caranya efek dapat diminimalkan. Menjawab: Jam sistem diperbarui pada setiap interupsi jam. Jika interupsi adalah dinonaktifkan—terutama untuk jangka waktu yang lama—mungkin saja jam sistem dapat dengan mudah kehilangan waktu yang tepat. Jam sistem adalah juga digunakan untuk tujuan penjadwalan. Misalnya, kuantum waktu untuk proses dinyatakan sebagai sejumlah clock ticks. Pada setiap interupsi jam, penjadwal menentukan apakah kuantum waktu untuk yang sedang berjalan proses telah kedaluwarsa. Jika interupsi jam dinonaktifkan, penjadwal tidak dapat secara akurat menetapkan kuantum waktu. Efek ini dapat diminimalkan dengan menonaktifkan interupsi jam hanya untuk periode yang sangat singkat.

5.2 Jelaskan mengapa Windows, Linux, dan Solaris menerapkan penguncian ganda mekanisme. Jelaskan keadaan di mana mereka menggunakan spin- kunci, kunci mutex, semaphore, kunci mutex adaptif, dan kondisi variabel. Dalam setiap kasus, jelaskan mengapa mekanisme itu diperlukan. Menjawab: Sistem operasi ini menyediakan mekanisme penguncian yang berbeda tergantung- sesuai dengan kebutuhan pengembang aplikasi. Spinlock berguna untuk sistem multiprosesor di mana utas dapat berjalan dalam loop sibuk (untuk waktu singkat) daripada menimbulkan biaya overhead yang dimasukkan ke dalam antrian tidur. Mutex berguna untuk mengunci sumber daya. Solaris 2 menggunakan mutex adaptif, artinya mutex diimplementasikan dengan spin mengunci mesin multiprosesor. Semaphore dan variabel kondisi adalah alat yang lebih tepat untuk sinkronisasi ketika sumber daya harus ditahan untuk jangka waktu yang lama, karena pemintalan tidak efisien untuk waktu yang lama durasi.

5.3 Apa arti dari istilah sibuk menunggu? Apa jenis lainnya? menunggu apakah ada di sistem operasi? Bisakah menunggu sibuk dihindari? sama sekali? Jelaskan jawabanmu. Menjawab: Sibuk menunggu berarti bahwa suatu proses sedang menunggu suatu kondisi dipenuhi condition dalam lingkaran ketat tanpa melepaskan prosesor. Atau, proses bisa menunggu dengan melepaskan prosesor, dan memblokir a kondisi dan menunggu untuk dibangunkan pada waktu yang tepat di masa depan. Penantian yang sibuk dapat dihindari tetapi menimbulkan biaya tambahan yang terkait dengan menempatkan proses untuk tidur dan harus membangunkannya ketika status program yang sesuai tercapai.

5.4 Jelaskan mengapa spinlock tidak sesuai untuk sistem prosesor tunggal namun sering digunakan dalam sistem multiprosesor. Menjawab: Spinlock tidak sesuai untuk sistem prosesor tunggal karena because kondisi yang akan mematahkan proses keluar dari spinlock dapat diperoleh hanya dengan menjalankan proses yang berbeda. Jika prosesnya tidak menyerah prosesor, proses lain tidak mendapatkan kesempatan untuk mengatur kondisi program yang diperlukan untuk proses pertama untuk membuat kemajuan. Di sebuah sistem multiprosesor, proses lain dijalankan pada prosesor lain dan dengan demikian memodifikasi status program untuk melepaskan proses pertama dari spinlock.

5.5 Tunjukkan bahwa, jika operasi semaphore wait() dan signal() tidak dieksekusi secara atom, maka pengecualian bersama dapat dilanggar. Menjawab: Operasi menunggu secara atom mengurangi nilai yang terkait dengan a tiang sinyal. Jika dua operasi wait dieksekusi pada semaphore ketika nilainya adalah 1, jika dua operasi tidak dilakukan secara atom, maka itu adalah mungkin kedua operasi dapat melanjutkan untuk mengurangi semaphore nilai, sehingga melanggar saling pengecualian.

5.6 Ilustrasikan bagaimana semaphore biner dapat digunakan untuk mengimplementasikan mutual pengecualian di antara n proses. Menjawab: Proses n berbagi semaphore, mutex, diinisialisasi ke 1. Setiap proses Pi diatur sebagai berikut:

do {

wait(mutex);

/\* critical section \*/

signal(mutex);

/\* remainder section \*/

} while (true);

Ch 13

13.1 Apa perbedaan utama antara daftar kemampuan dan daftar akses? Menjawab: Daftar akses adalah daftar untuk setiap objek yang terdiri dari domain dengan kumpulan hak akses yang tidak kosong untuk objek tersebut. Daftar kemampuan adalah daftar dari objek dan operasi yang diizinkan pada objek tersebut untuk setiap domain.

13.2 File MCP Burroughs B7000/B6000 dapat ditandai sebagai data sensitif. Ketika file seperti itu dihapus, area penyimpanannya akan ditimpa oleh beberapa bit acak. Untuk tujuan apa skema seperti itu berguna? Menjawab: Ini akan berguna sebagai tindakan pengamanan ekstra agar konten lama memori tidak dapat diakses, baik sengaja atau tidak sengaja, dengan program lain. Ini sangat berguna untuk semua yang sangat rahasia informasi.

13.3 Dalam sistem perlindungan cincin, level 0 memiliki akses terbesar ke objek, dan level n (di mana n > 0) memiliki hak akses yang lebih sedikit. Hak akses dari sebuah program pada tingkat tertentu dalam struktur cincin dianggap sebagai seperangkat kemampuan. Apa hubungan antara kemampuan domain di level j dan domain di level i ke objek (untuk j > i)? Menjawab: Dj adalah subset dari Di .

13.4 Sistem RC 4000, antara lain, telah mendefinisikan pohon proses (disebut pohon proses) sedemikian rupa sehingga semua turunan dari suatu proses dapat diberikan sumber daya (benda) dan hak akses hanya oleh nenek moyang mereka. Jadi, keturunan tidak akan pernah memiliki kemampuan untuk melakukan apa pun yang nenek moyangnya tidak bisa melakukan. Akar pohon adalah sistem operasi, yang memiliki has kemampuan untuk melakukan apa saja. Asumsikan himpunan hak akses terwakili oleh matriks akses, A. A(x,y) mendefinisikan hak akses dari proses x ke objek y. Jika x adalah turunan dari z, bagaimana hubungan antara A(x,y) dan A(z,y) untuk objek sembarang y? Menjawab: A(x,y) adalah himpunan bagian dari A(z,y).

13.5Masalah perlindungan apa yang mungkin muncul jika tumpukan bersama digunakan untuk melewati parameter? Menjawab: Isi tumpukan dapat dikompromikan oleh proses lain berbagi tumpukan.

13.6 Pertimbangkan lingkungan komputasi di mana nomor unik diasosiasikan dengan setiap proses dan setiap objek dalam sistem. Misalkan kita mengizinkan proses dengan nomor n untuk mengakses objek dengan nomor m saja jika n > m. Apa jenis struktur perlindungan yang kita miliki? Menjawab: Struktur hierarki.

13.7 Pertimbangkan lingkungan komputasi di mana suatu proses diberikan: hak istimewa untuk mengakses objek hanya n kali. Sarankan skema untuk menerapkan kebijakan ini. Menjawab: Tambahkan penghitung bilangan bulat dengan kemampuan.

13.8 Jika semua hak akses ke suatu objek dihapus, objek tersebut tidak dapat lagi diakses. Pada titik ini, objek juga harus dihapus, dan and ruang yang ditempatinya harus dikembalikan ke sistem. Sarankan yang efisien implementasi skema ini. Menjawab: Jumlah referensi.

13.9 Mengapa sulit untuk melindungi sistem di mana pengguna diizinkan untuk melakukannya? I/O mereka sendiri? Menjawab: Dalam bab-bab sebelumnya kami mengidentifikasi perbedaan antara kernel dan pengguna mode di mana mode kernel digunakan untuk melakukan operasi istimewa seperti I/O. Salah satu alasan mengapa I/O harus dilakukan dalam mode kernel adalah bahwa I/O memerlukan pengaksesan perangkat keras dan akses yang tepat ke perangkat keras diperlukan untuk integritas sistem. Jika kami mengizinkan pengguna untuk melakukan I/O mereka sendiri, kami tidak dapat menjamin integritas sistem.

13.10 Daftar kemampuan biasanya disimpan dalam ruang alamat pengguna. Bagaimana sistem memastikan bahwa pengguna tidak dapat mengubah konten? dari daftar? Menjawab: Daftar kemampuan dianggap sebagai "objek yang dilindungi" dan hanya dapat diakses secara tidak langsung oleh pengguna. Sistem operasi memastikan pengguna tidak dapat mengakses daftar kemampuan secara langsung.

Ch 15

15.1 Modul kernel yang dapat dimuat secara dinamis memberikan fleksibilitas saat driver ditambahkan ke sistem, tetapi apakah mereka memiliki kelemahan juga? Dibawah apa keadaan akan kernel dikompilasi menjadi satu file biner, dan kapan akan lebih baik untuk tetap membaginya menjadi modul? Jelaskan Anda menjawab. Menjawab: Ada dua kelemahan utama dengan penggunaan modul. Yang pertama adalah size: manajemen modul menggunakan memori kernel yang tidak dapat di-page, dan kernel dasar dengan sejumlah modul yang dimuat akan mengkonsumsi lebih banyak memori dari kernel yang setara dengan driver yang dikompilasi ke dalam gambar kernel itu sendiri. Ini bisa menjadi masalah yang sangat signifikan pada mesin dengan memori fisik yang terbatas. Kelemahan kedua adalah modul dapat meningkatkan kompleksitas dari proses bootstrap kernel. Sulit untuk memuat satu set modul dari disk jika driver perlu mengakses disk itu sendiri sebuah modul yang perlu dimuat. Akibatnya, mengelola bootstrap kernel dengan modul dapat memerlukan kerja ekstra dari pihak administrator: the modul yang diperlukan untuk bootstrap perlu ditempatkan ke dalam gambar ramdisk yang dimuat di samping gambar kernel awal saat sistem diinisialisasi. Dalam kasus tertentu lebih baik menggunakan kernel modular, dan dalam kasus lain kasus lebih baik menggunakan kernel dengan driver perangkatnya telah ditautkan sebelumnya. Di mana meminimalkan ukuran kernel itu penting, pilihannya akan tergantung pada seberapa sering berbagai driver perangkat digunakan. Jika mereka adalah dalam penggunaan konstan, maka modul tidak cocok. Ini terutama benar dimana driver dibutuhkan untuk proses booting itu sendiri. Di samping itu, jika beberapa driver tidak selalu diperlukan, maka mekanisme modul memungkinkan driver tersebut untuk dimuat dan dibongkar sesuai permintaan, berpotensi menawarkan penghematan bersih dalam memori fisik. Di mana kernel akan dibangun yang harus dapat digunakan pada berbagai macam mesin yang sangat berbeda, maka membangunnya dengan modul jelas lebih baik menggunakan satu kernel dengan lusinan driver yang tidak perlu memakan memori. Hal ini terutama berlaku untuk komersial kernel terdistribusi, yang mendukung berbagai perangkat keras dengan cara yang paling sederhana mungkin adalah prioritas. Namun, jika kernel sedang dibangun untuk satu mesin yang: konfigurasi diketahui terlebih dahulu, kemudian kompilasi dan menggunakan modul mungkin hanya menjadi kompleksitas yang tidak perlu. Dalam kasus seperti ini, penggunaan modul mungkin masalah selera.

15.2 Multithreading adalah teknik pemrograman yang umum digunakan. Menggambarkan tiga cara berbeda untuk mengimplementasikan utas, dan bandingkan ketiganya metode dengan mekanisme Linux clone(). Kapan mungkin menggunakan masing-masing mekanisme alternatif lebih baik atau lebih buruk daripada menggunakan klon? Menjawab: Implementasi thread dapat secara luas diklasifikasikan menjadi dua kelompok: utas berbasis kernel dan utas mode pengguna. Paket utas mode pengguna- usia bergantung pada beberapa dukungan kernel— mereka mungkin memerlukan interupsi timer fasilitas, misalnya — tetapi penjadwalan antar utas tidak dilakukan dibentuk oleh kernel tetapi oleh beberapa perpustakaan kode mode pengguna. Beberapa utas dalam implementasi seperti itu muncul di sistem operasi sebagai konteks eksekusi tunggal. Ketika proses multithreaded sedang berjalan, ia memutuskan sendiri utas mana yang akan dieksekusi, menggunakan non-lokal melompat untuk beralih di antara utas sesuai dengan preemptive atau aturan penjadwalan non-preemptive. Atau, kernel sistem operasi dapat menyediakan dukungan untuk benang itu sendiri. Dalam hal ini, utas dapat diimplementasikan sebagai terpisah proses yang terjadi untuk berbagi alamat umum yang lengkap atau sebagian ruang, atau mereka dapat diimplementasikan sebagai konteks eksekusi terpisah dalam satu proses. Bagaimanapun cara utasnya diatur, mereka muncul sebagai konteks eksekusi yang sepenuhnya independen untuk aplikasi. Implementasi hybrid juga dimungkinkan, di mana sejumlah besar a utas tersedia untuk aplikasi menggunakan nomor yang lebih kecil dari utas kernel. Utas pengguna yang dapat dijalankan dijalankan oleh yang pertama tersedia benang inti. Di Linux, utas diimplementasikan di dalam kernel oleh klon mekanisme yang menciptakan proses baru dalam alamat virtual yang sama ruang sebagai proses induk. Tidak seperti beberapa paket thread berbasis kernel, kernel Linux tidak membedakan antara utas dan proses: utas hanyalah proses yang tidak membuat virtual baru ruang alamat ketika diinisialisasi. Keuntungan utama mengimplementasikan utas di kernel bukan daripada di perpustakaan mode pengguna adalah bahwa: • sistem kernel-threaded dapat memanfaatkan banyak prosesor multiple jika tersedia; dan • jika satu utas memblokir dalam rutinitas layanan kernel (misalnya, a panggilan sistem atau kesalahan halaman), utas lainnya masih dapat berjalan. Keuntungan yang lebih rendah adalah kemampuan untuk menetapkan atribut keamanan yang berbeda ke setiap utas. Implementasi mode pengguna tidak memiliki keunggulan ini. Karena implementasi seperti itu berjalan sepenuhnya dalam satu eksekusi kernel konteks, hanya satu utas yang dapat berjalan sekaligus, bahkan jika banyak CPU tersedia. Untuk alasan yang sama, jika satu utas memasuki sistem panggilan, tidak ada utas lain yang dapat berjalan hingga panggilan sistem itu selesai. Sebagai hasilnya, satu utas yang melakukan pembacaan disk pemblokiran akan menahan setiap benang dalam aplikasi. Namun, implementasi mode pengguna melakukannya memiliki keunggulan masing-masing. Yang paling jelas adalah kinerja: memohon penjadwal kernel sendiri untuk beralih di antara utas melibatkan memasukkan domain perlindungan baru saat CPU beralih ke mode kernel, sedangkan beralih antar utas dalam mode pengguna dapat dicapai hanya dengan menyimpan dan memulihkan register CPU utama. Utas mode pengguna mungkin juga mengkonsumsi lebih sedikit memori sistem: sebagian besar sistem UNIX akan memesan di setidaknya satu halaman penuh untuk tumpukan kernel untuk setiap utas kernel, dan tumpukan ini mungkin tidak dapat di-page. Pendekatan hybrid, mengimplementasikan beberapa utas pengguna di atas sejumlah kecil utas kernel, memungkinkan keseimbangan di antara ini tradeoff yang ingin dicapai. Utas kernel akan memungkinkan banyak utas untuk memblokir panggilan kernel sekaligus dan akan mengizinkannya berjalan beberapa CPU, dan pergantian utas mode pengguna dapat terjadi di masing-masing utas kernel untuk melakukan utas ringan tanpa biaya tambahan memiliki terlalu banyak utas kernel. Kelemahan dari pendekatan ini adalah kompleksitas: memberikan kontrol atas tradeoff memperumit utas antarmuka pengguna perpustakaan.

15.3 Kernel Linux tidak mengizinkan paging keluar dari memori kernel. Apa apakah pembatasan ini berpengaruh pada desain kernel? Apa dua? keuntungan dan dua kerugian dari keputusan desain ini? Menjawab: Dampak utama dari pelarangan paging memori kernel di Linux adalah bahwa non-preemptability kernel dipertahankan. Apa saja proses mengambil kesalahan halaman, baik dalam kernel atau dalam mode pengguna, berisiko dijadwal ulang saat data yang diperlukan masuk dari disk. Karena kernel dapat diandalkan untuk tidak dijadwal ulang selama akses ke struktur data utamanya, persyaratan penguncian untuk melindungi integritas struktur data tersebut sangat disederhanakan. Meskipun kesederhanaan desain adalah manfaat itu sendiri, itu juga memberikan yang penting keunggulan kinerja pada mesin uniprosesor karena fakta bahwa tidak perlu melakukan penguncian tambahan pada sebagian besar data internal struktur. Ada sejumlah kerugian dari kurangnya kernel yang dapat di-page memori, namun. Pertama-tama, itu membebankan batasan pada jumlah memori yang dapat digunakan kernel. Tidak masuk akal untuk menyimpannya sangat besar struktur data dalam memori yang tidak dapat di-page, karena itu mewakili fisik memori yang sama sekali tidak dapat digunakan untuk hal lain. Ini memiliki dua dampak: pertama-tama, kernel harus memangkas banyak bagian internalnya struktur data secara manual, alih-alih dapat mengandalkan satu mekanisme memori virtual untuk menjaga penggunaan memori fisik di bawah kontrol. Kedua, membuatnya tidak mungkin untuk mengimplementasikan fitur-fitur tertentu certain yang membutuhkan sejumlah besar memori virtual di kernel, seperti: /tmp-filesystem (sistem file berbasis memori virtual yang cepat ditemukan di beberapa sistem UNIX Perhatikan bahwa kerumitan mengelola kesalahan halaman saat berjalan kode kernel tidak menjadi masalah di sini. Kode kernel Linux sudah bisa untuk menangani kesalahan halaman: itu harus dapat menangani panggilan sistem yang argumennya merujuk ke memori pengguna yang mungkin ditampilkan ke disk.

15.4 Diskusikan tiga keuntungan dari hubungan dinamis (bersama) perpustakaan dibandingkan dengan hubungan statis. Jelaskan dua kasus di mana hubungan statis lebih disukai. Menjawab: Keuntungan utama dari shared library adalah bahwa mereka mengurangi memori dan ruang disk yang digunakan oleh suatu sistem, dan mereka meningkatkan pemeliharaan. Ketika shared library sedang digunakan oleh semua program yang sedang berjalan, hanya ada satu contoh dari setiap rutin perpustakaan sistem pada disk, dan paling banyak satu contoh dalam memori fisik. Ketika perpustakaan yang bersangkutan adalah salah satu yang digunakan oleh banyak aplikasi dan program, kemudian disk dan penghematan memori bisa sangat besar. Selain itu, waktu mulai untuk menjalankan program baru dapat dikurangi, karena banyak fungsi yang dibutuhkan oleh program itu kemungkinan besar sudah dimuat ke dalam ingatan fisik. Pemeliharaan juga merupakan keuntungan utama dari hubungan dinamis atas statis. Jika semua program yang berjalan menggunakan perpustakaan bersama untuk mengakses access rutinitas perpustakaan sistem, kemudian memutakhirkan rutinitas tersebut, baik untuk menambah fungsionalitas baru atau untuk memperbaiki bug, dapat dilakukan hanya dengan menggantinya perpustakaan bersama. Tidak perlu mengkompilasi ulang atau menautkan ulang aplikasi apa pun; program apa pun yang dimuat setelah peningkatan selesai akan secara otomatis mengambil versi baru dari perpustakaan. Ada keuntungan lain juga. Sebuah program yang menggunakan perpustakaan bersama sering dapat disesuaikan untuk tujuan tertentu hanya dengan mengganti satu atau lebih banyak perpustakaannya, atau bahkan (jika sistem mengizinkannya, dan sebagian besar UNIX termasuk Linux do) menambahkan yang baru saat run time. Misalnya, perpustakaan debugging dapat diganti dengan yang normal untuk melacak masalah dalam sebuah aplikasi. Pustaka bersama juga memungkinkan binari program menjadi terkait dengan komersial, kode perpustakaan berpemilik tanpa benar-benar termasuk salah satu kode itu dalam file eksekusi akhir program. Ini adalah penting karena pada sebagian besar sistem UNIX, banyak standar yang dibagikan perpustakaan adalah hak milik, dan masalah lisensi dapat mencegah termasuk kode itu dalam file yang dapat dieksekusi untuk didistribusikan ke pihak ketiga. Di beberapa tempat, bagaimanapun, hubungan statis adalah tepat. Satu contoh ada di lingkungan penyelamatan untuk administrator sistem. Jika sebuah sistem administrator membuat kesalahan saat menginstal perpustakaan baru, atau jika perangkat keras mengembangkan masalah, sangat mungkin untuk berbagi yang ada perpustakaan menjadi korup. Akibatnya, seringkali satu set dasar penyelamatan utilitas terhubung secara statis, sehingga ada peluang untuk memperbaiki kesalahan tanpa harus bergantung pada fungsi perpustakaan bersama benar. Ada juga keunggulan kinerja yang terkadang membuat statis hubungan lebih disukai dalam kasus-kasus khusus. Sebagai permulaan, hubungan dinamis tidak tingkatkan waktu startup untuk suatu program, karena tautannya sekarang harus dilakukan pada saat run time daripada pada waktu kompilasi. Tautan dinamis juga dapat terkadang meningkatkan ukuran set kerja maksimum suatu program ( jumlah total halaman fisik memori yang diperlukan untuk menjalankan program). Di perpustakaan bersama, pengguna tidak memiliki kendali atas tempat di perpustakaan file biner berbagai fungsi berada. Karena sebagian besar fungsi tidak tepat mengisi satu halaman penuh atau halaman perpustakaan, memuat fungsi akan biasanya mengakibatkan pemuatan di bagian fungsi sekitarnya juga.Dengan hubungan statis, sama sekali tidak ada fungsi yang tidak direferensikan (langsung atau tidak langsung) oleh aplikasi perlu dimuat ke dalam memori. Masalah lain seputar hubungan statis termasuk kemudahan distribusi: lebih mudah untuk mendistribusikan file yang dapat dieksekusi dengan tautan statis daripada dengan hubungan dinamis jika distributor tidak yakin apakah penerima akan menginstal perpustakaan yang benar terlebih dahulu. Disana mungkin juga menjadi pembatasan komersial terhadap pendistribusian ulang beberapa binari sebagai perpustakaan bersama. Misalnya, lisensi untuk grafis “Motif” UNIX lingkungan memungkinkan binari menggunakan Motif untuk didistribusikan secara bebas selama karena mereka terhubung secara statis, tetapi perpustakaan bersama mungkin tidak digunakan tanpa lisensi.

15.5 Bandingkan penggunaan soket jaringan dengan penggunaan memori bersama sebagai mekanisme untuk mengkomunikasikan data antar proses pada satu komputer. Apa keuntungan dari masing-masing metode? Kapan masing-masing? disukai? Menjawab: Menggunakan soket jaringan daripada memori bersama untuk komunikasi lokal nication memiliki sejumlah keunggulan. Keuntungan utama adalah bahwa antarmuka pemrograman soket menampilkan serangkaian sinkronisasi yang kaya fitur. Suatu proses dapat dengan mudah menentukan kapan data baru telah tiba koneksi soket, berapa banyak data yang ada, dan siapa yang mengirimnya. Pro- cesses dapat memblokir hingga data baru tiba di soket, atau mereka dapat meminta bahwa sinyal dikirimkan ketika data tiba. Soket juga mengatur koneksi terpisah. Proses dengan soket terbuka untuk menerima can terima beberapa koneksi ke soket itu dan akan diberi tahu saat baru proses mencoba untuk terhubung atau ketika proses lama memutuskan koneksinya. Memori bersama tidak menawarkan fitur ini. Tidak ada cara untuk proses untuk menentukan apakah proses lain telah disampaikan atau diubah data dalam memori bersama selain dengan melihat isinya dari memori itu. Tidak mungkin suatu proses memblokir dan meminta bangun ketika memori bersama dikirimkan, dan tidak ada standar mekanisme untuk proses lain untuk membuat tautan memori bersama ke proses yang ada. Namun, memori bersama memiliki keuntungan yang sangat banyak lebih cepat daripada komunikasi soket dalam banyak kasus. Saat data dikirim melalui soket, biasanya disalin dari memori ke beberapa memori waktu. Pembaruan memori bersama tidak memerlukan salinan data: jika satu proses memperbarui struktur data di memori bersama, pembaruan itu segera terlihat oleh semua proses lain yang berbagi memori itu. Mengirim atau menerima data melalui soket mengharuskan panggilan layanan sistem kernel dilakukan untuk memulai transfer, tetapi komunikasi memori bersama dapat dilakukan sepenuhnya dalam mode pengguna tanpa memerlukan transfer kontrol. Komunikasi soket biasanya lebih disukai ketika koneksi man- manajemen penting atau ketika ada persyaratan untuk menyinkronkan pengirim dan penerima. Misalnya, proses server biasanya akan buat soket pendengar yang dapat dihubungkan klien saat mereka mau untuk menggunakan layanan itu. Setelah soket dibuat, permintaan individu juga dikirim menggunakan soket, sehingga server dapat dengan mudah menentukan kapan permintaan baru datang dan dari siapa permintaan itu datang. Namun, dalam beberapa kasus, memori bersama lebih disukai. Bersama memori seringkali merupakan solusi yang lebih baik ketika data dalam jumlah besar untuk ditransfer atau ketika dua proses membutuhkan akses acak ke kumpulan data umum. Dalam hal ini, bagaimanapun, proses komunikasi mungkin masih memerlukan mekanisme tambahan selain memori bersama untuk mencapai sinkronisasi di antara mereka sendiri. Sistem X Window, a lingkungan tampilan grafis untuk UNIX, adalah contoh yang baik untuk ini: kebanyakan permintaan grafis dikirim melalui soket, tetapi memori bersama ditawarkan sebagai transportasi tambahan dalam kasus khusus di mana bitmap besar harus akan ditampilkan di layar. Dalam hal ini, permintaan untuk menampilkan bitmap akan tetap dikirim melalui soket, tetapi sebagian besar data bitmap itu sendiri akan dikirim melalui memori bersama.

15.6 Pada suatu waktu, sistem UNIX menggunakan pengoptimalan tata letak disk berdasarkan pada posisi rotasi data disk, tetapi implementasi modern, termasuk Linux, cukup optimalkan untuk akses data sekuensial. Kenapa mereka melakukannya? Dari karakteristik perangkat keras apa akses berurutan? mengambil keuntungan? Mengapa optimasi rotasi tidak lagi berguna? Menjawab: Karakteristik kinerja perangkat keras disk telah berubah sub- secara konstan dalam beberapa tahun terakhir. Secara khusus, banyak peningkatan telah diperkenalkan untuk meningkatkan bandwidth maksimum yang dapat dicapai pada sebuah disk. Dalam sistem modern, mungkin ada saluran panjang antara sistem operasi dan kepala baca-tulis disk. Permintaan I/O disk memiliki untuk melewati pengontrol disk lokal komputer, melalui logika bus ke disk drive itu sendiri, dan kemudian secara internal ke disk, di mana ada kemungkinan menjadi pengontrol kompleks yang dapat menyimpan akses data dan berpotensi mengoptimalkan urutan permintaan I/O. Karena kerumitan ini, waktu yang dibutuhkan untuk satu permintaan I/O menjadi diakui dan untuk permintaan berikutnya dibuat dan diterima oleh disk dapat jauh melebihi jumlah waktu antara satu sektor disk lewat di bawah kepala baca-tulis dan header sektor berikutnya tiba. Agar dapat membaca beberapa sektor sekaligus secara efisien, disk akan menggunakan cache readahead. Sementara satu sektor sedang dilewatkan kembali ke komputer host, disk akan sibuk membaca sektor berikutnya di antisipasi permintaan untuk membacanya. Jika permintaan baca mulai berdatangan pesanan yang merusak pipa readahead ini, kinerja akan turun. Akibatnya, kinerja sangat menguntungkan jika sistem operasi operating mencoba untuk menjaga permintaan I/O dalam urutan yang ketat. Fitur kedua dari disk modern adalah geometrinya bisa sangat kompleks. Jumlah sektor per silinder dapat bervariasi sesuai dengan posisi silinder: lebih banyak data dapat diperas menjadi lebih lama trek lebih dekat tepi disk daripada di tengah disk. Untuk sebuah