#OPERASI ARSITEKTUR OS

1.MONOLITHIC

Struktur Monolitik Struktur paling sederhana untuk mengorganisasi sebuah sistem operasi adalah tanpa struktur sama sekali. Artinya, semua fungsionalitas kernel ditempatkan dalam satu berkas biner statis yang berjalan dalam satu ruang alamat. Pendekatan ini yang dikenal sebagai struktur monolitik merupakan teknik umum dalam merancang sistem operasi.

Contoh struktur terbatas seperti ini adalah sistem operasi UNIX versi awal, yang terdiri dari dua bagian yang dapat dipisahkan: kernel dan program-program sistem. Kernel kemudian dipisah lebih lanjut menjadi serangkaian interface dan device driver (pengendali perangkat), yang telah ditambahkan dan dikembangkan selama bertahun-tahun seiring dengan evolusi UNIX.

Kita dapat melihat sistem operasi UNIX tradisional sebagai sistem yang memiliki lapisan tertentu, seperti yang ditunjukkan dalam Gambar 2.12. Segala sesuatu yang berada di bawah interface pemanggilan sistem (system call interface) dan di atas perangkat keras fisik adalah bagian dari kernel. Kernel menyediakan sistem berkas (file system), penjadwalan CPU, manajemen memori, dan fungsi-fungsi sistem operasi lainnya melalui system call. Jika digabungkan, ini merupakan jumlah fungsionalitas yang sangat besar untuk dimasukkan ke dalam satu ruang alamat tunggal.

Sistem operasi Linux didasarkan pada UNIX dan disusun dengan cara yang serupa, seperti yang ditunjukkan dalam Gambar 2.13. Aplikasi biasanya menggunakan pustaka standar C glibc ketika berkomunikasi dengan system call interface menuju kernel. Kernel Linux bersifat monolitik karena berjalan sepenuhnya dalam mode kernel di satu ruang alamat, namun seperti akan kita lihat di Bagian 2.8.4, ia memiliki desain modular yang memungkinkan kernel dimodifikasi saat sistem sedang berjalan (run time).

Meskipun kernel monolitik tampak sederhana, mereka sebenarnya sulit untuk diimplementasikan dan dikembangkan lebih lanjut. Namun, kernel monolitik memiliki keuntungan kinerja yang jelas: terdapat overhead yang sangat kecil dalam system call interface, dan komunikasi dalam kernel berlangsung sangat cepat.Oleh karena itu, meskipun memiliki kekurangan pada kernel monolitik, kecepatan dan efisiensinya menjelaskan mengapa kita masih melihat bukti struktur ini pada sistem operasi UNIX, Linux, dan Windows.

2.LAYERED APPROACH

Pendekatan monolitik sering disebut sebagai sistem yang tightly coupled (terikat erat) karena perubahan pada satu bagian sistem dapat menimbulkan dampak yang luas pada bagian lainnya. Sebagai alternatif, kita dapat merancang sistem yang loosely coupled (longgar keterikatannya). Sistem seperti ini dibagi menjadi komponen-komponen kecil yang terpisah dengan fungsi yang spesifik dan terbatas. Semua komponen ini bersama-sama membentuk kernel. Keuntungan dari pendekatan modular ini adalah bahwa perubahan pada satu komponen hanya memengaruhi komponen tersebut saja dan tidak memengaruhi yang lain sehingga memberi kebebasan lebih bagi pengembang sistem dalam membuat dan mengubah mekanisme internal sistem.

Sebuah sistem dapat dibuat modular dengan berbagai cara. Salah satu metode adalah pendekatan berlapis, di mana sistem operasi dipecah menjadi sejumlah lapisan (layers atau levels). Lapisan terbawah (lapisan 0) adalah perangkat keras (hardware), sedangkan lapisan tertinggi (lapisan N) adalah antarmuka pengguna (user interface). Struktur berlapis ini ditunjukkan pada Gambar 2.14.

Sebuah lapisan sistem operasi merupakan implementasi dari sebuah objek abstrak yang terdiri dari data dan operasi yang dapat memanipulasi data tersebut. Lapisan sistem operasi yang tipikal misalnya lapisan M terdiri atas struktur data dan sekumpulan fungsi yang dapat dipanggil oleh lapisan tingkat lebih tinggi. Sebaliknya, lapisan M juga dapat memanggil operasi pada lapisan yang lebih rendah.

Keuntungan utama dari pendekatan berlapis adalah kesederhanaan dalam konstruksi dan proses debugging. Setiap lapisan dipilih sedemikian rupa sehingga hanya menggunakan fungsi (operasi) yang dan layanan hanya dari lapisan tingkat lebih rendah. Pendekatan ini menyederhanakan proses debugging dan verifikasi sistem. Lapisan pertama dapat diuji (debug) tanpa perlu khawatir terhadap bagian sistem lainnya, karena secara definisi, lapisan ini hanya menggunakan perangkat keras dasar (yang diasumsikan benar) untuk mengimplementasikan fungsinya. Setelah lapisan pertama selesai diuji, fungsinya dapat dianggap benar ketika lapisan kedua diuji, dan seterusnya. Jika sebuah kesalahan ditemukan saat debugging pada lapisan tertentu, maka kesalahan itu pasti ada di lapisan tersebut, karena lapisan di bawahnya sudah diuji sebelumnya. Dengan demikian, perancangan dan implementasi sistem menjadi lebih sederhana.

Setiap lapisan diimplementasikan hanya dengan operasi yang disediakan oleh lapisan tingkat lebih rendah. Sebuah lapisan tidak perlu mengetahui bagaimana operasi tersebut diimplementasikan lapisan itu hanya perlu mengetahui apa yang dilakukan operasi tersebut. Oleh karena itu, setiap lapisan menyembunyikan keberadaan struktur data tertentu, operasi, dan perangkat keras dari lapisan tingkat lebih tinggi.

Sistem berlapis telah berhasil digunakan dalam jaringan komputer (seperti TCP/IP) dan aplikasi web. Namun demikian, relatif sedikit sistem operasi yang menggunakan pendekatan berlapis murni. Salah satu alasannya adalah tantangan dalam mendefinisikan fungsi dari setiap lapisan secara tepat. Selain itu, kinerja keseluruhan dari sistem semacam ini cenderung buruk karena adanya overhead (beban tambahan) yang disebabkan oleh program pengguna yang harus melewati banyak lapisan untuk mendapatkan layanan sistem operasi.

Meski begitu, penggunaan sebagian pendekatan berlapis umum ditemukan pada sistem operasi modern. Umumnya, sistem ini memiliki lebih sedikit lapisan dengan fungsi yang lebih banyak, sehingga memberikan sebagian besar keuntungan dari kode modular sambil menghindari masalah terkait definisi dan interaksi antar-lapisan.

3.MICROKERNEIS

Kita telah melihat bahwa sistem UNIX asli memiliki struktur monolitik. Seiring berkembangnya UNIX, kernel menjadi besar dan sulit untuk dikelola. Pada pertengahan tahun 1980-an, para peneliti di Carnegie Mellon University mengembangkan sebuah sistem operasi bernama Mach yang memodularisasi kernel menggunakan pendekatan mikrokernel. Metode ini menyusun sistem operasi dengan cara menghilangkan address spaces.Semua komponen yang tidak penting dihapus dari kernel dan diimplementasikan sebagai program tingkat pengguna (user-level) yang berada di ruang alamat terpisah. Hasilnya adalah kernel yang lebih kecil. Masih belum ada konsensus mengenai layanan mana yang harus tetap berada di kernel dan mana yang harus diimplementasikan di ruang pengguna. Namun, secara umum, mikrokernel menyediakan manajemen proses dan memori minimal, selain fasilitas komunikasi. Gambar 2.15 menggambarkan arsitektur mikrokernel yang tipikal.

Fungsi utama mikrokernel adalah menyediakan komunikasi antara program klien dan berbagai layanan yang juga berjalan di ruang pengguna. Komunikasi ini dilakukan melalui message passing (pertukaran pesan), yang telah dijelaskan pada Bagian 2.3.3.5. Misalnya, jika program klien ingin mengakses sebuah berkas, program tersebut harus berinteraksi dengan file server. Program klien dan layanan tidak berinteraksi secara langsung. Mereka berkomunikasi secara tidak langsung dengan menukar pesan melalui mikrokernel.

Salah satu keuntungan pendekatan mikrokernel adalah mempermudah perluasan sistem operasi. Semua layanan baru ditambahkan ke ruang pengguna dan akibatnya tidak memerlukan modifikasi kernel. Ketika kernel harus dimodifikasi, perubahan yang diperlukan cenderung lebih sedikit, karena mikrokernel merupakan kernel yang lebih kecil. Sistem operasi yang dihasilkan lebih mudah dipindahkan (port) dari satu desain perangkat keras ke perangkat keras lainnya. Mikrokernel juga menyediakan keamanan dan keandalan yang lebih tinggi, karena sebagian besar layanan berjalan sebagai proses pengguna, bukan kernel. Jika sebuah layanan gagal, sisa sistem operasi tetap tidak terganggu.

Ilustrasi paling terkenal dari sistem operasi mikrokernel adalah Darwin, komponen kernel dari sistem operasi macOS dan iOS. Faktanya, Darwin terdiri dari dua kernel, salah satunya adalah mikrokernel Mach. Kita akan membahas sistem macOS dan iOS lebih detail pada Bagian 2.8.5.1.

Contoh lainnya adalah QNX, sistem operasi waktu nyata untuk sistem tertanam (embedded systems). Mikrokernel QNX Neutrino menyediakan layanan untuk pertukaran pesan dan penjadwalan proses. Mikrokernel ini juga menangani komunikasi jaringan tingkat rendah dan interupsi perangkat keras. Semua layanan lain di QNX disediakan oleh proses standar yang berjalan di luar kernel dalam mode pengguna.

Sayangnya, kinerja mikrokernel bisa menurun karena meningkatnya overhead fungsi sistem. Ketika dua layanan tingkat pengguna harus berkomunikasi, pesan harus disalin antara layanan yang berada di ruang alamat terpisah. Selain itu, sistem operasi mungkin harus berganti dari satu proses ke proses berikutnya untuk menukar pesan. Overhead yang terkait dengan penyalinan pesan dan pergantian proses menjadi hambatan terbesar bagi pertumbuhan sistem operasi berbasis mikrokernel.

Pertimbangkan sejarah Windows NT: Rilis pertama memiliki organisasi mikrokernel berlapis. Kinerja versi ini rendah dibandingkan dengan Windows 95. Windows NT 4.0 sebagian memperbaiki masalah kinerja dengan memindahkan lapisan dari ruang pengguna ke ruang kernel dan mengintegrasikannya lebih erat. Pada saat Windows XP dirancang, arsitektur Windows telah menjadi lebih monolitik dibandingkan mikrokernel. Bagian 2.8.5.1 akan menjelaskan bagaimana macOS menangani masalah kinerja pada mikrokernel Mach.

4.MODULES

Mungkin metodologi terkini yang terbaik untuk desain sistem operasi melibatkan penggunaan Loadable Kernel Modules (LKMs) atau modul kernel yang dapat dimuat. Di sini, kernel memiliki sekumpulan komponen inti dan dapat menautkan layanan tambahan melalui modul, baik saat booting maupun saat sistem sedang berjalan. Desain seperti ini umum digunakan pada implementasi modern UNIX, seperti Linux, macOS, dan Solaris, serta Windows.

Ide dari desain ini adalah agar kernel menyediakan layanan inti, sementara layanan lainnya diimplementasikan secara dinamis saat kernel berjalan. Menautkan layanan secara dinamis lebih disukai dibanding menambahkan fitur baru langsung ke kernel, karena menambahkan fitur langsung akan mengharuskan kernel dikompilasi ulang setiap kali ada perubahan. Misalnya, kita bisa membangun algoritma penjadwalan CPU dan manajemen memori langsung ke dalam kernel, kemudian menambahkan dukungan untuk berbagai sistem berkas melalui modul yang dapat dimuat.

Hasil keseluruhannya menyerupai sistem berlapis karena setiap bagian kernel memiliki antarmuka yang terdefinisi dan terlindungi; namun lebih fleksibel dibanding sistem berlapis, karena modul apa pun dapat memanggil modul lainnya. Pendekatan ini juga mirip dengan pendekatan mikrokernel, di mana modul utama hanya memiliki fungsi inti dan mengetahui cara memuat serta berkomunikasi dengan modul lain; tetapi lebih efisien, karena modul tidak perlu menggunakan message passing untuk berkomunikasi.

Linux menggunakan Loadable Kernel Modules, terutama untuk mendukung device driver dan sistem berkas. LKM dapat “dimasukkan” ke kernel saat sistem dimulai (boot) atau saat sistem sedang berjalan, misalnya ketika perangkat USB dicolokkan ke komputer yang sedang berjalan. Jika kernel Linux tidak memiliki driver yang diperlukan, driver tersebut dapat dimuat secara dinamis. LKM juga dapat dihapus dari kernel saat sistem berjalan.

Bagi Linux, LKM memungkinkan kernel yang dinamis dan modular, sekaligus mempertahankan keuntungan kinerja dari sistem monolitik. Pembuatan LKM di Linux akan dibahas melalui beberapa latihan pemrograman di akhir bab ini.

5.HYBRID SYSTEMS

Dalam praktiknya, sangat sedikit sistem operasi yang mengadopsi satu struktur tunggal yang benar-benar baku. Sebaliknya, mereka menggabungkan berbagai struktur, sehingga menghasilkan sistem hibrida yang menangani masalah kinerja, keamanan, dan kegunaan.

Misalnya, Linux bersifat monolitik, karena menempatkan sistem operasi dalam satu ruang alamat memberikan kinerja yang sangat efisien. Namun, Linux juga bersifat modular, sehingga fungsionalitas baru dapat ditambahkan secara dinamis ke kernel.

Windows sebagian besar juga bersifat monolitik (sekali lagi terutama untuk alasan kinerja), tetapi tetap mempertahankan beberapa perilaku khas dari sistem mikrokernel, termasuk menyediakan dukungan untuk subsistem terpisah (dikenal sebagai operating-system personalities) yang berjalan sebagai proses mode pengguna. Sistem Windows juga mendukung modul kernel yang dapat dimuat secara dinamis.

Studi kasus tentang Linux dan Windows 10 akan dibahas pada Bab 20 dan Bab 21, secara berurutan. Pada sisa bagian ini, kita akan mengeksplorasi struktur dari tiga sistem hibrida: sistem operasi Apple macOS dan dua sistem operasi mobile paling menonjol iOS dan Android.