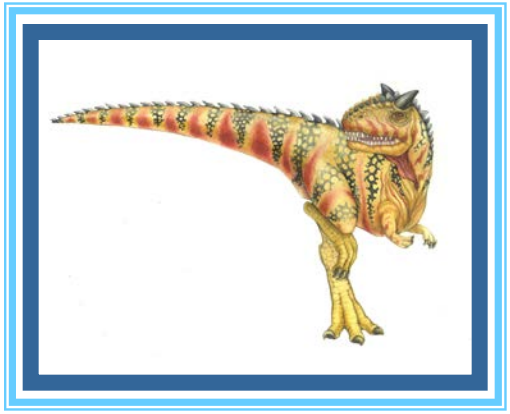
Bab 2: Sistem Operasi Struktur



Bab 2: Struktur Sistem Operasi

- Layanan Sistem Operasi Antarmuka
- Sistem Operasi Pengguna Panggilan
- Sistem
- Jenis Program Sistem
- Panggilan Sistem
- Perancangan dan Implementasi Sistem
- Operasi Struktur Sistem Operasi
- Debugging Sistem Operasi
- Pembuatan Sistem Operasi
- Boot Sistem

Tujuan

- Untuk mendeskripsikan layanan yang disediakan oleh sistem operasi kepada pengguna, proses, dan sistem lainnya
- Untuk membahas berbagai cara penataan sistem operasi
- Untuk menjelaskan bagaimana sistem operasi dipasang dan dikustomisasi dan bagaimana mereka melakukan booting

Layanan Sistem Operasi

- Sistem operasi menyediakan lingkungan untuk eksekusi program dan layanan untuk program dan pengguna
- Satu set layanan sistem operasi menyediakan fungsi yang membantu pengguna:
  - **Antarmuka pengguna**-Hampir semua sistem operasi memiliki antarmuka pengguna (**UI**).
    - Bervariasi antara**Garis komando(CLI),Antarmuka Pengguna Grafik(GUI),Kelompok**
  - **Eksekusi program**-Sistem harus dapat memuat program ke dalam memori dan menjalankan program itu, mengakhiri eksekusi, baik secara normal atau tidak normal (menunjukkan kesalahan)
  - **operasi I/O**-Program yang sedang berjalan mungkin memerlukan I/O, yang mungkin melibatkan file atau perangkat I/O

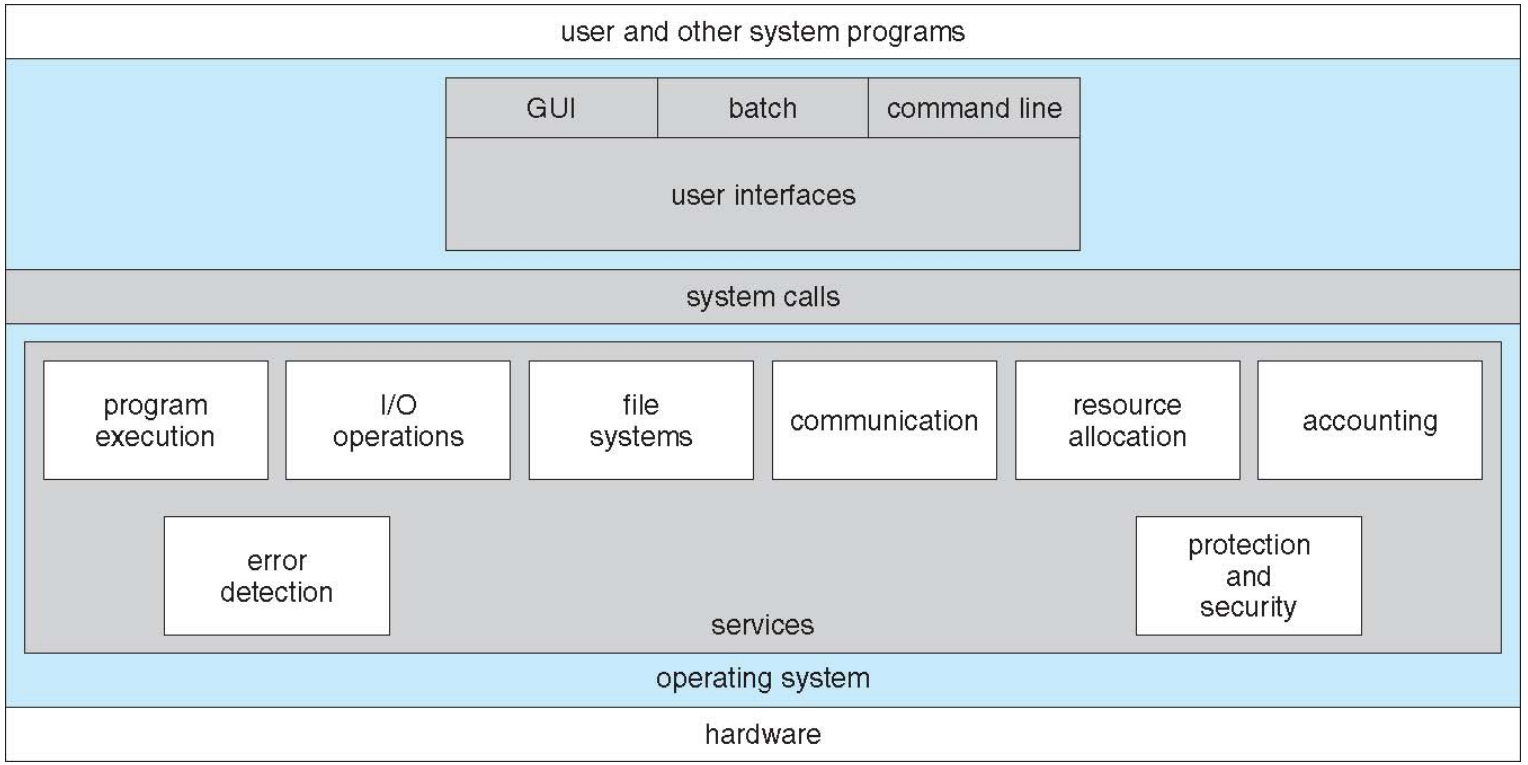
Layanan Sistem Operasi (Lanjutan)

- Satu set layanan sistem operasi menyediakan fungsi yang bermanfaat bagi pengguna (Lanjutan):
  - **Manipulasi sistem file**-Sistem file sangat menarik. Program perlu membaca dan menulis file dan direktori, membuat dan menghapusnya, mencarinya, mencantumkan Informasi file, manajemen izin.
  - **Komunikasi**–Proses dapat bertukar informasi, di komputer yang sama atau antar komputer melalui jaringan
    - Komunikasi mungkin melalui memori bersama atau melalui pengiriman pesan (paket dipindahkan oleh OS)
  - **Deteksi kesalahan**–OS harus selalu waspada terhadap kemungkinan kesalahan
    - Dapat terjadi di CPU dan perangkat keras memori, di perangkat I/O, di program pengguna
    - Untuk setiap jenis kesalahan, OS harus mengambil tindakan yang tepat untuk memastikan komputasi yang benar dan konsisten
    - Fasilitas debug dapat sangat meningkatkan kemampuan pengguna dan pemrogram untuk menggunakan sistem secara efisien

Layanan Sistem Operasi (Lanjutan)

- Satu set fungsi OS ada untuk memastikan operasi yang efisien dari sistem itu sendiri melalui pembagian sumber daya
  - **Alokasi sumber daya -**Ketika beberapa pengguna atau beberapa pekerjaan berjalan secara bersamaan, sumber daya harus dialokasikan untuk masing-masingnya
    - Banyak jenis sumber daya - siklus CPU, memori utama, penyimpanan file, perangkat I/O.
  - **Akuntansi -**Untuk melacak pengguna mana yang menggunakan berapa banyak dan jenis sumber daya komputer apa
  - **Perlindungan dan keamanan -**Pemilik informasi yang disimpan dalam sistem komputer multipengguna atau jaringan mungkin ingin mengontrol penggunaan informasi tersebut, proses bersamaan tidak boleh mengganggu satu sama lain
    - **Perlindungan**melibatkan memastikan bahwa semua akses ke sumber daya sistem dikendalikan
    - **Keamanan**sistem dari pihak luar memerlukan autentikasi pengguna, mencakup perlindungan perangkat I/O eksternal dari upaya akses yang tidak valid

Tampilan Layanan Sistem Operasi



Antarmuka Sistem Operasi Pengguna - CLI

CLI atau**juru perintah**memungkinkan masuknya perintah langsung

- Terkadang diimplementasikan dalam kernel, terkadang dengan program sistem
- Terkadang beberapa rasa diimplementasikan –**kerang** Terutama
- mengambil perintah dari pengguna dan menjalankannya
- Terkadang perintah bawaan, terkadang hanya nama program
  - Jika yang terakhir, menambahkan fitur baru tidak memerlukan modifikasi shell

Penerjemah Perintah Bourne Shell

Default

New Info Close Execute Bookmarks

Default Default

```
PBG-Mac-Pro:~ pbg$ w
15:24  up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER     TTY      FROM              LOGIN@  IDLE WHAT
pbg      console  -                 14:34      50 -
pbg      s000     -                 15:05       - w
PBG-Mac-Pro:~ pbg$ iostat 5
          disk0           disk1          disk10       cpu     load average
    KB/t tps  MB/s     KB/t tps  MB/s     KB/t tps  MB/s  us sy id   1m   5m   15m
   33.75 343 11.30    64.31  14  0.88    39.67   0  0.02  11  5 84  1.51 1.53 1.65
    5.27 320  1.65     0.00   0  0.00     0.00   0  0.00   4  2 94  1.39 1.51 1.65
    4.28 329  1.37     0.00   0  0.00     0.00   0  0.00   5  3 92  1.44 1.51 1.65
^C
PBG-Mac-Pro:~ pbg$ ls
Applications                    Music                           WebEx
Applications (Parallels)        Pando Packages                  config.log
Desktop                         Pictures                        getsmartdata.txt
Documents                       Public                          imp
Downloads                       Sites                           log
Dropbox                         Thumbs.db                       panda-dist
Library                         Virtual Machines                prob.txt
Movies                          Volumes                         scripts
PBG-Mac-Pro:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbg$
```

Antarmuka Sistem Operasi Pengguna - GUI

- Mudah digunakan**Desktop**antarmuka metafora
  - Biasanya mouse, keyboard, dan monitor **Ikon**
  - mewakili file, program, tindakan, dll
  - Berbagai tombol mouse di atas objek di antarmuka menyebabkan berbagai tindakan (menyediakan informasi, opsi, menjalankan fungsi, membuka direktori (dikenal sebagai a**map**)
  - Diciptakan di Xerox PARC
- Banyak sistem sekarang menyertakan antarmuka CLI dan GUI
  - Microsoft Windows adalah GUI dengan shell "perintah" CLI
  - Apple Mac OS X adalah antarmuka GUI "Aqua" dengan kernel UNIX di bawahnya dan cangkang tersedia
  - Unix dan Linux memiliki CLI dengan antarmuka GUI opsional (CDE, KDE, GNOME)

Antarmuka layar sentuh

- Perangkat layar sentuh memerlukan antarmuka baru
  - Mouse tidak mungkin atau tidak diinginkan
  - Tindakan dan pemilihan berdasarkan gerakan
  - Keyboard virtual untuk entri teks
- Perintah suara.

GUI Mac OS X


Grab File Edit Capture Window Help
15:06 EDT Mon 2 Jul
fig-dir
TIFF
Address Book
Apple Computer Inc.
Dictionary and Thesaurus
operating system
op·er·at·ing sys·tem
the software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.
PCalc

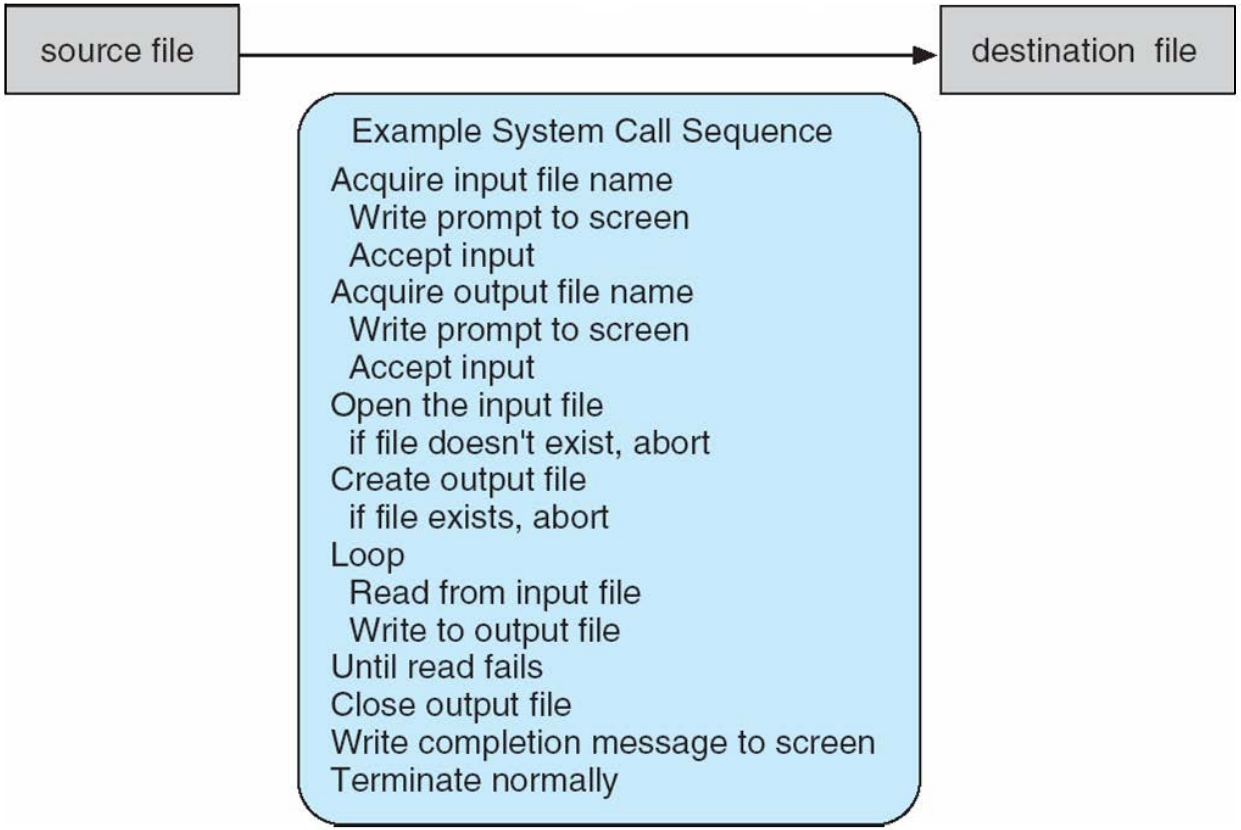
Panggilan Sistem

- Antarmuka pemrograman ke layanan yang disediakan oleh OS
- Biasanya ditulis dalam bahasa tingkat tinggi (C atau C++)
- Sebagian besar diakses oleh program melalui tingkat tinggi **Antarmuka Pemrograman Aplikasi(API)**daripada menggunakan panggilan sistem langsung
- Tiga API yang paling umum adalah Win32 API untuk Windows, POSIX API untuk sistem berbasis POSIX (termasuk hampir semua versi UNIX, Linux, dan Mac OS X), dan Java API untuk mesin virtual Java (JVM)

Perhatikan bahwa nama panggilan sistem yang digunakan di seluruh teks ini bersifat umum

Contoh Panggilan Sistem

- Urutan panggilan sistem untuk menyalin konten dari satu file ke file lain



Contoh API Standar

EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t    read(int fd, void *buf, size_t count)
```

return value — function name — parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

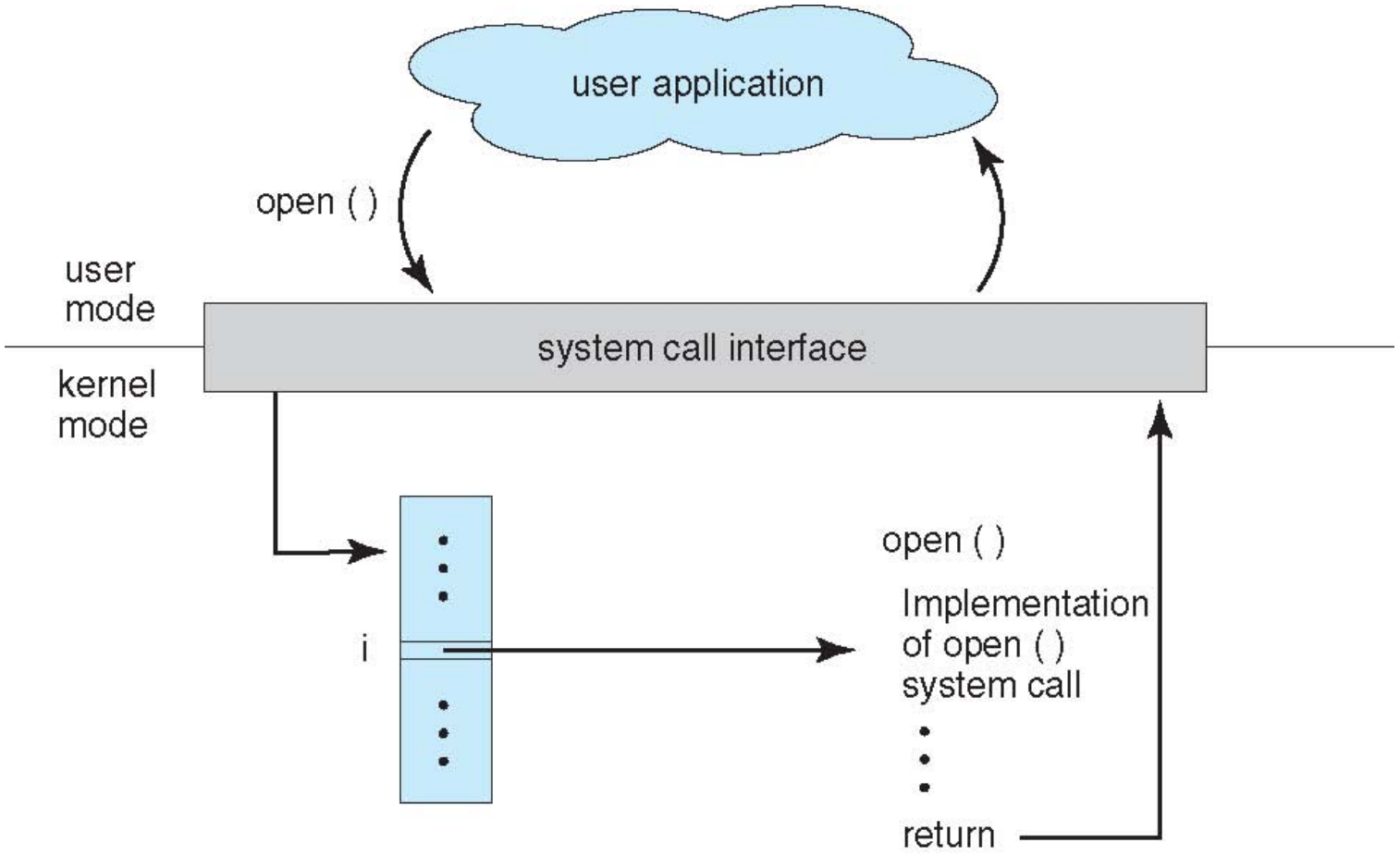
- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.

Implementasi Panggilan Sistem

- Biasanya, nomor yang terkait dengan setiap panggilan sistem
  - **Antarmuka panggilan sistem**memelihara tabel yang diindeks menurut angka-angka ini
- Antarmuka panggilan sistem memanggil panggilan sistem yang dimaksud dalam kernel OS dan mengembalikan status panggilan sistem dan nilai pengembalian apa pun
- Penelepon tidak perlu tahu apa-apa tentang bagaimana system call diimplementasikan
  - Hanya perlu mematuhi API dan memahami apa yang akan dilakukan OS sebagai panggilan hasil
  - Sebagian besar detail antarmuka OS disembunyikan dari programmer oleh API
    - Dikelola oleh pustaka dukungan run-time (kumpulan fungsi yang dibangun ke dalam pustaka yang disertakan dengan kompiler)

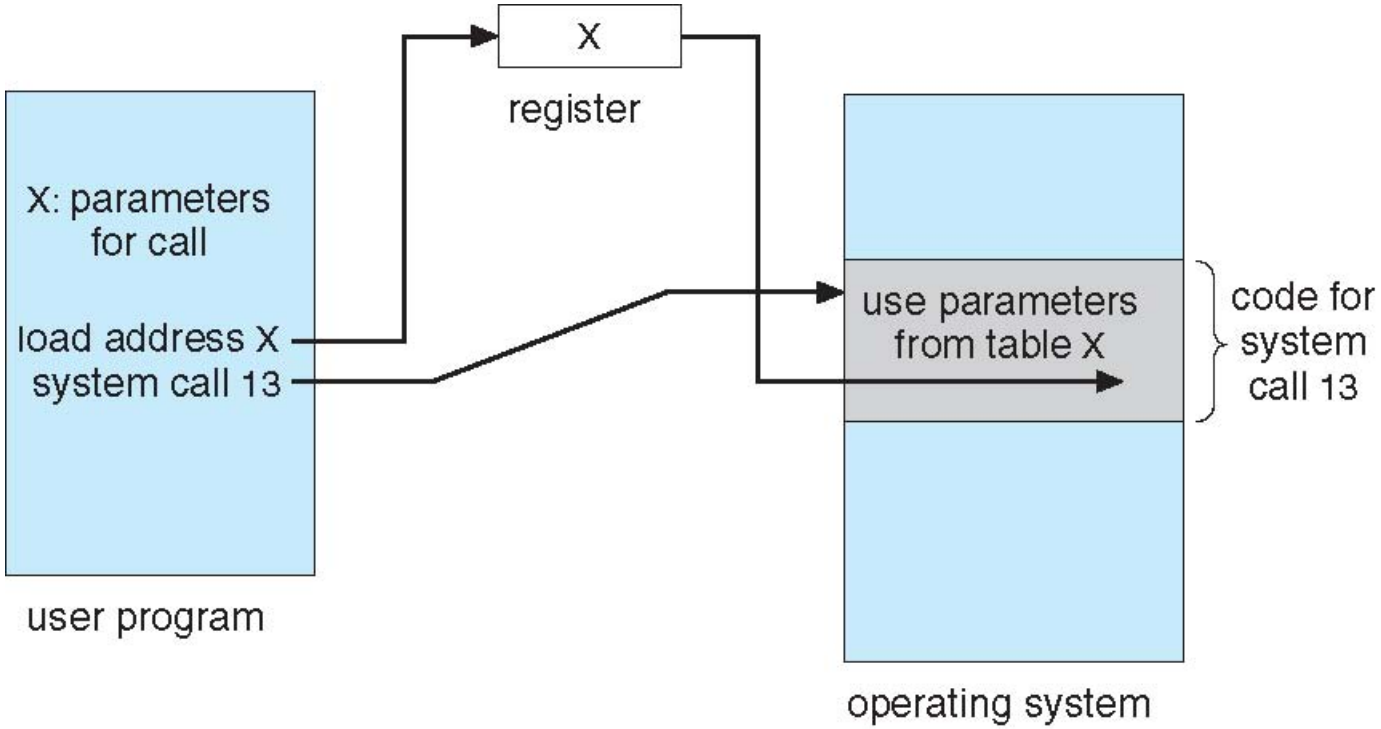
API – Panggilan Sistem – Hubungan OS



Melewati Parameter Panggilan Sistem

- Seringkali, lebih banyak informasi diperlukan daripada sekadar identitas panggilan sistem yang diinginkan
  - Jenis dan jumlah informasi yang tepat bervariasi menurut OS dan panggilan
- Tiga metode umum yang digunakan untuk meneruskan parameter ke OS
  - Paling sederhana: berikan parameter dalam register
    - Dalam beberapa kasus, mungkin lebih banyak parameter daripada register
  - Parameter disimpan dalam satu blok,atau tabel, dalam memori, dan alamat blok yang diteruskan sebagai parameter dalam register
    - Pendekatan ini diambil oleh Linux dan Solaris
  - Parameter ditempatkan, atau **didorong**,ke**tumpukan** oleh program dan **muncul** dari tumpukan oleh sistem operasi
  - Metode blok dan tumpukan tidak membatasi jumlah atau panjang parameter yang diteruskan

Melewati Parameter melalui Tabel



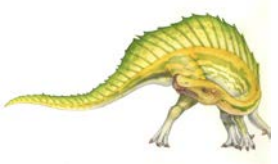
Jenis Panggilan Sistem

-Pengendalian proses

- buat proses, akhiri proses,
- batalkan
- memuat, mengeksekusi
- dapatkan atribut proses, atur atribut proses
- tunggu waktu
- tunggu acara, alokasikan acara
- sinyal dan kosongkan memori
- Buang memori jika kesalahan
- **Debugger**untuk menentukan**bug, satu langkah**eksekusi **Kunci**
- untuk mengelola akses ke data bersama antar proses

Jenis Panggilan Sistem

- Manajemen file
  - buat file, hapus file
  - buka, tutup file
  - baca, tulis, reposisi dapatkan
  - dan atur atribut file
- Manajemen perangkat
  - meminta perangkat, melepaskan perangkat
  - baca, tulis, posisikan ulang
  - dapatkan atribut perangkat, setel atribut perangkat
  - secara logis pasang atau lepas perangkat

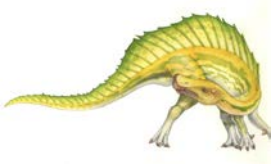


Jenis Panggilan Sistem (Lanjutan)

- Pemeliharaan informasi
  - dapatkan waktu atau tanggal, atur waktu atau
  - tanggal, dapatkan data sistem, atur data sistem
  - dapatkan dan atur atribut proses, file, atau perangkat
- Komunikasi
  - buat, hapus koneksi komunikasi
  - kirim, terima pesan jika**model penyampaian pesan**ke**nama tuan rumah**atau**nama proses**
    - Dari**klien**ke**server**
  - **Model memori bersama**membuat dan mendapatkan akses ke wilayah memori
  - informasi status transfer pasang dan
  - lepaskan perangkat jarak jauh

Jenis Panggilan Sistem (Lanjutan)

- Perlindungan
  - Kontrol akses ke sumber daya
  - Dapatkan dan setel izin Izinkan
  - dan tolak akses pengguna

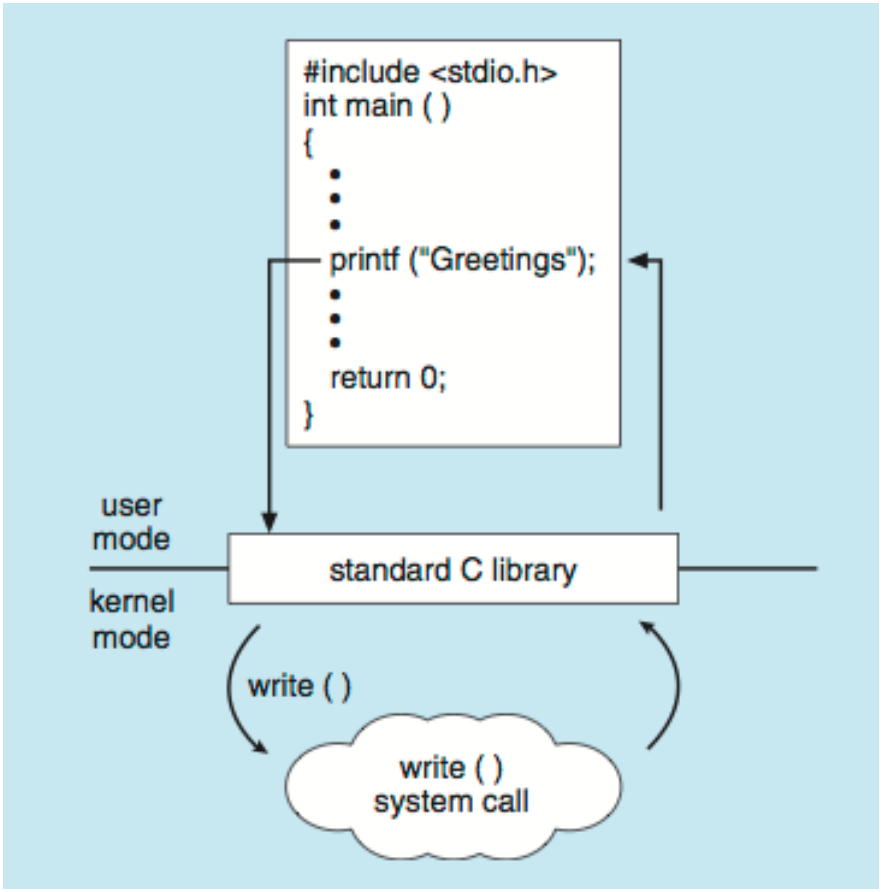


Contoh Panggilan Sistem Windows dan Unix

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

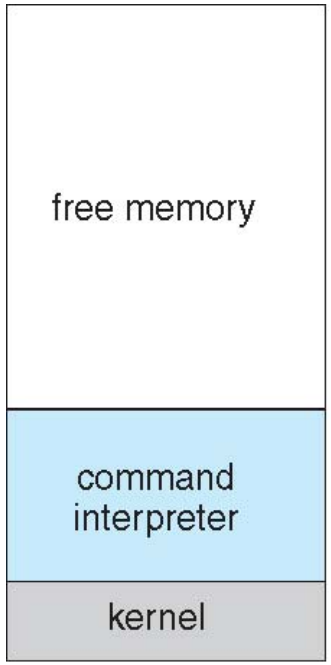
Contoh Perpustakaan C Standar

- Program C memanggil printf() library call, yang memanggil write() system call



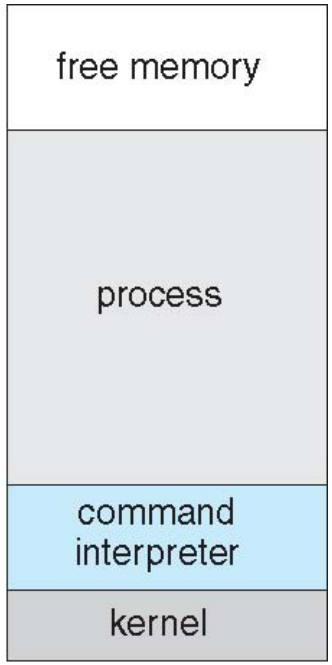
Contoh: MS-DOS

- Tugas tunggal
- Shell dipanggil saat sistem di-boot
- Metode sederhana untuk menjalankan program
  - Tidak ada proses yang
- dibuat Ruang memori tunggal
- Memuat program ke dalam memori, menimpa semua kecuali kernel
- Keluar dari program -> shell dimuat ulang



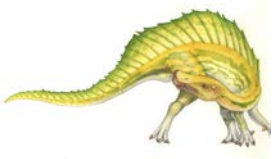
(a)

Saat memulai sistem



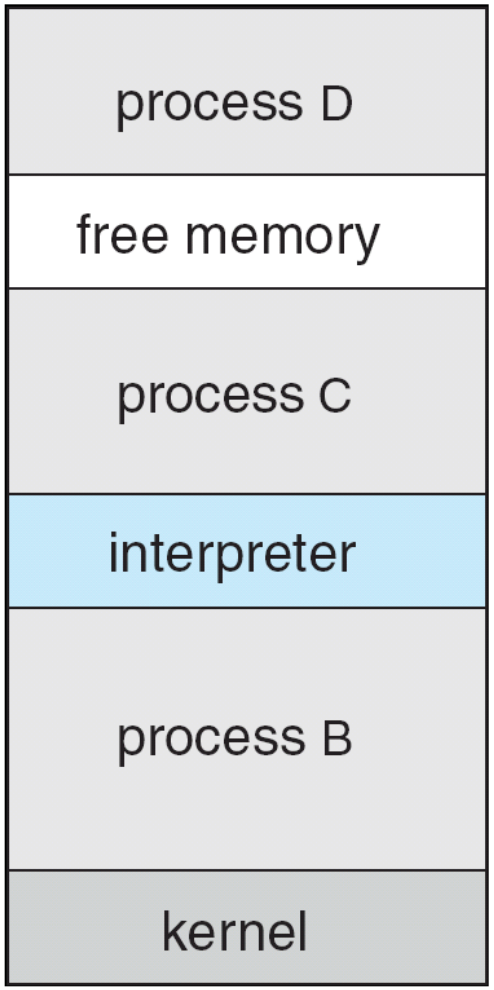
(b)

menjalankan program



Contoh: FreeBSD

- Varian Unix
- Multitasking
- Login pengguna -> aktifkan shell pilihan pengguna
- Shell mengeksekusi panggilan sistem fork() untuk membuat proses
  - Jalankan exec() untuk memuat program ke dalam proses
  - Shell menunggu proses berhenti atau melanjutkan dengan perintah pengguna
- Proses keluar dengan:
  - kode = 0 – tidak ada kode
  - kesalahan > 0 – kode kesalahan



Program Sistem

- Program sistem menyediakan lingkungan yang nyaman untuk pengembangan dan eksekusi program. Mereka dapat dibagi menjadi:
  - Manipulasi file
  - Informasi status terkadang disimpan dalam dukungan
  - bahasa Pemrograman modifikasi File
  - Memuat dan mengeksekusi
  - program Komunikasi
  - Layanan latar belakang
  - Program aplikasi
- Sebagian besar pandangan pengguna tentang sistem operasi ditentukan oleh program sistem, bukan panggilan sistem yang sebenarnya

Program Sistem

- Menyediakan lingkungan yang nyaman untuk pengembangan dan eksekusi program
  - Beberapa di antaranya hanyalah antarmuka pengguna untuk panggilan sistem; lainnya jauh lebih kompleks
- **Manajemen file**-Buat, hapus, salin, ganti nama, cetak, buang, daftar, dan umumnya manipulasi file dan direktori
- **Informasi status**
  - Beberapa meminta info dari sistem - tanggal, waktu, jumlah memori yang tersedia, ruang disk, jumlah pengguna
  - Lainnya memberikan informasi kinerja, logging, dan debugging yang mendetail
  - Biasanya, program ini memformat dan mencetak keluaran ke terminal atau perangkat keluaran lainnya
  - Beberapa sistem mengimplementasikan a**registry**-digunakan untuk menyimpan dan mengambil informasi konfigurasi

Program Sistem (Lanjutan)

- **Modifikasi berkas**
  - Editor teks untuk membuat dan memodifikasi file
  - Perintah khusus untuk mencari isi file atau melakukan transformasi teks
- **Dukungan bahasa pemrograman**-Compiler, assembler, debugger, dan interpreter terkadang disediakan
- **Pemuatan dan eksekusi program**- Mutlak loader, relocatable loader, linkage editor, dan overlay-loader, sistem debugging untuk tingkat yang lebih tinggi dan bahasa mesin
- **Komunikasi**-Menyediakan mekanisme untuk membuat koneksi virtual antara proses, pengguna, dan sistem komputer
  - Izinkan pengguna mengirim pesan ke layar satu sama lain, menjelajahi halaman web, mengirim pesan email, masuk dari jarak jauh, mentransfer file dari satu mesin ke mesin lainnya

Program Sistem (Lanjutan)

- **Layanan Latar Belakang**
  - Luncurkan saat boot
    - Beberapa untuk startup sistem, lalu hentikan
    - Beberapa dari boot sistem ke shutdown
  - Menyediakan fasilitas seperti pengecekan disk, penjadwalan proses, error logging, pencetakan
  - Jalankan dalam konteks pengguna bukan konteks kernel
  - Dikenal sebagai**jasa**,**subsistem**,**daemon**
- **Program aplikasi**
  - Jangan berhubungan dengan sistem
  - Dijalankan oleh pengguna
  - Biasanya tidak dianggap sebagai bagian dari OS
  - Diluncurkan oleh baris perintah, klik mouse, colek jari

Desain dan Implementasi Sistem Operasi

- Desain dan Implementasi OS tidak “dapat dipecahkan”, tetapi beberapa pendekatan telah terbukti berhasil
- Struktur internal Sistem Operasi yang berbeda dapat sangat bervariasi
- Mulai desain dengan menentukan tujuan dan spesifikasi
- Dipengaruhi oleh pilihan perangkat keras, jenis sistem
- **Pengguna** tujuan dan **Sistem** sasaran
  - Tujuan pengguna – sistem operasi harus nyaman digunakan, mudah dipelajari, andal, aman, dan cepat
  - Sasaran sistem – sistem operasi harus mudah dirancang, diimplementasikan, dan dipelihara, serta fleksibel, andal, bebas kesalahan, dan efisien

Desain dan Implementasi Sistem Operasi (Lanjutan)

- Prinsip penting untuk memisahkan

  Kebijakan:***Apa***akan selesai? **Mekanisme**:

  Bagaimanauntuk melakukannya?

- Mekanisme menentukan bagaimana melakukan sesuatu, kebijakan menentukan apa yang akan dilakukan
- Pemisahan kebijakan dari mekanisme adalah prinsip yang sangat penting, memungkinkan fleksibilitas maksimum jika keputusan kebijakan akan diubah kemudian (contoh – pengatur waktu)
- Menentukan dan mendesain OS adalah tugas yang sangat kreatif **rekayasa Perangkat Lunak**

Penerapan

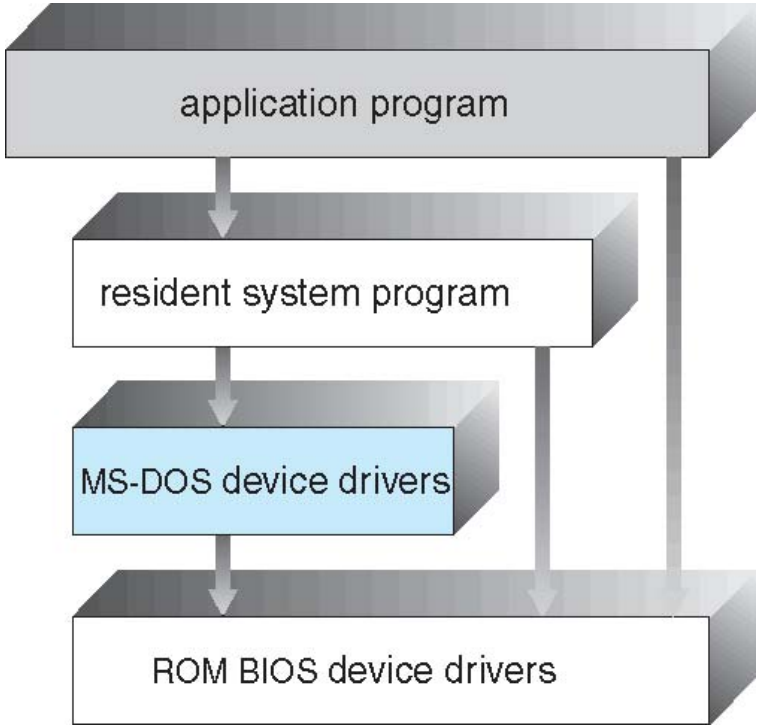
- Banyak variasi
  - OS awal dalam bahasa assembly
  - Kemudian bahasa pemrograman sistem seperti Algol, PL/1
  - Now C, C++
- Sebenarnya biasanya campuran bahasa
  - Level terendah dalam perakitan
  - Tubuh utama di C
  - Program sistem dalam C, C++, bahasa skrip seperti PERL, Python, skrip shell
- Lebih banyak bahasa tingkat tinggi lebih mudah**pelabuhan**ke perangkat keras lainnya
  - Tapi lebih lambat
- **Emulasi**dapat memungkinkan OS untuk berjalan pada perangkat keras non-asli

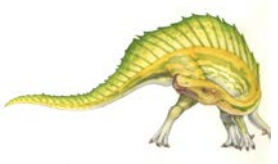
Struktur Sistem Operasi

- OS tujuan umum adalah program yang sangat besar
- Berbagai cara untuk menyusunnya
  - Struktur sederhana – MS-DOS
  - Lebih kompleks -- UNIX Layered
  - – sebuah abstraksi Microkernel
  - -Mach

Struktur Sederhana -- MS-DOS

- MS-DOS – ditulis untuk menyediakan fungsionalitas terbanyak dalam ruang terkecil
  - Tidak dibagi menjadi modul
  - Meskipun MS-DOS memiliki beberapa struktur, antarmuka dan tingkat fungsionalitasnya tidak dipisahkan dengan baik





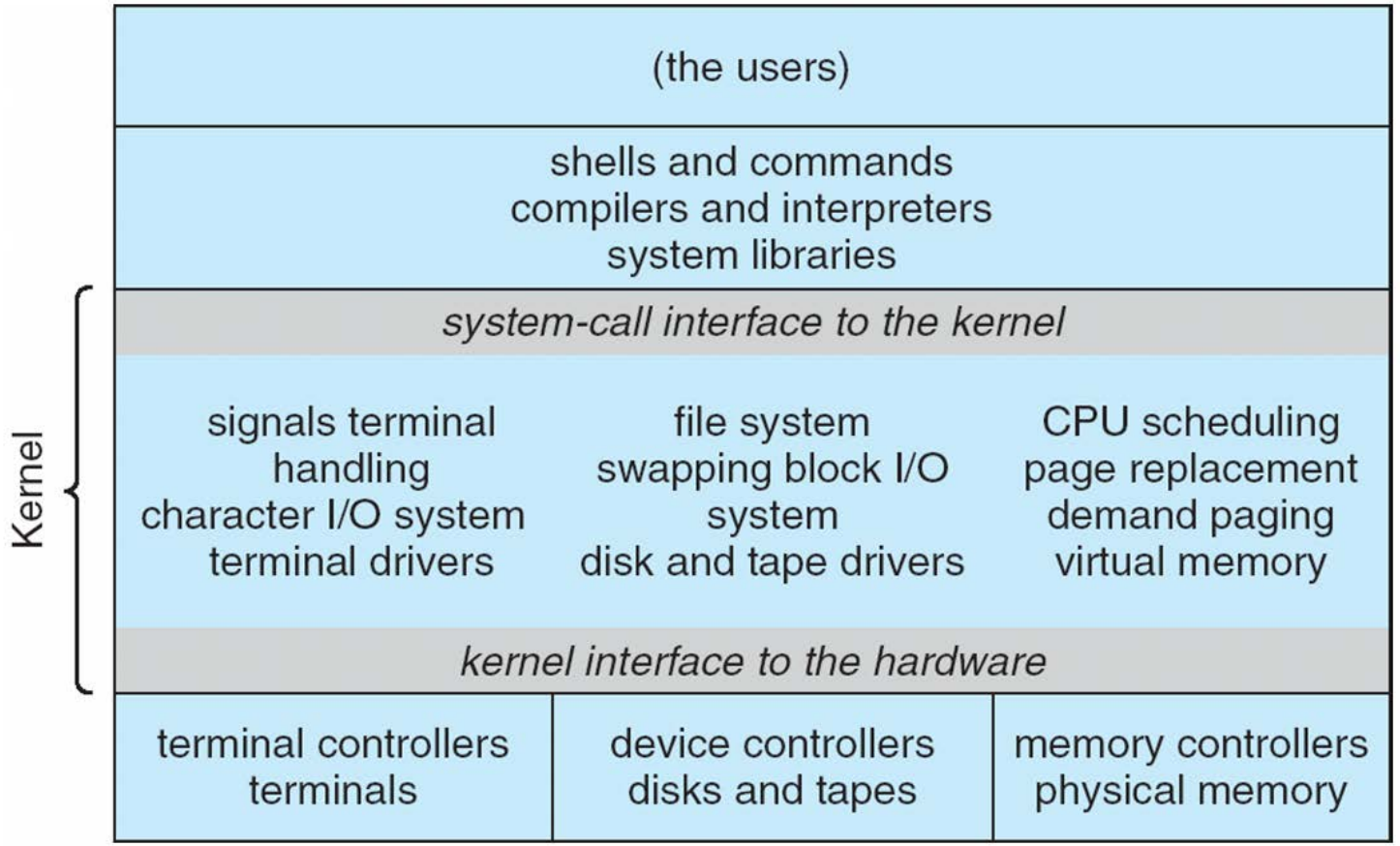
Struktur Tidak Sederhana -- UNIX

UNIX – dibatasi oleh fungsionalitas perangkat keras, sistem operasi UNIX asli memiliki penataan yang terbatas. OS UNIX terdiri dari dua bagian yang dapat dipisahkan

- Program sistem
- Kernel
  - Terdiri dari segala sesuatu di bawah antarmuka panggilan sistem dan di atas perangkat keras fisik
  - Menyediakan sistem file, penjadwalan CPU, manajemen memori, dan fungsi sistem operasi lainnya; sejumlah besar fungsi untuk satu tingkat

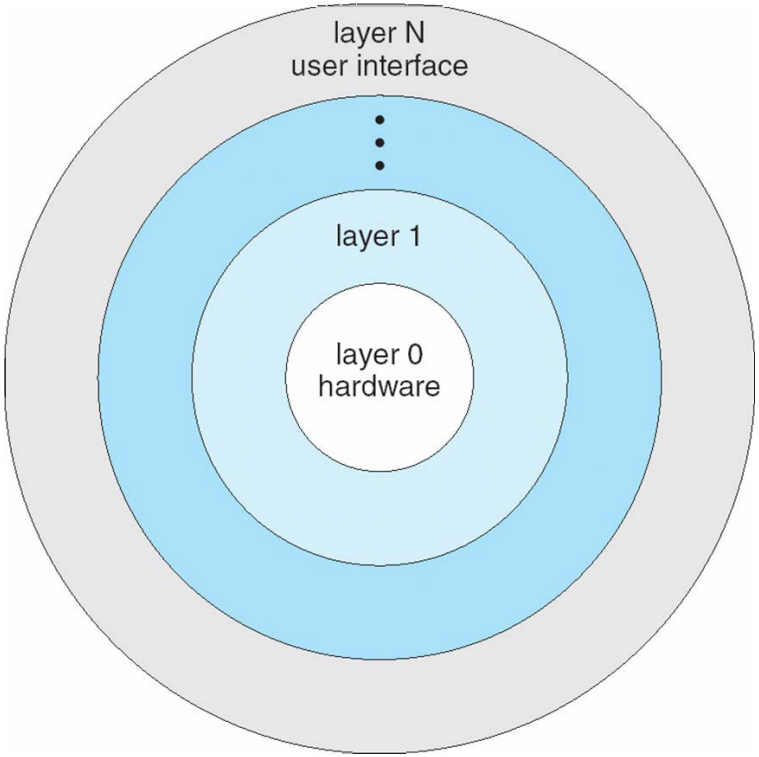
Struktur Sistem UNIX Tradisional

Di luar sederhana tetapi tidak sepenuhnya berlapis



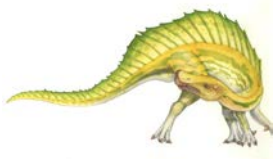
Pendekatan Berlapis

- Sistem operasi dibagi menjadi beberapa lapisan (level), masing-masing dibangun di atas dan di bawah lapisan. Lapisan paling bawah (lapisan 0), adalah perangkat keras; yang tertinggi (lapisan N) adalah antarmuka pengguna.
- Dengan modularitas, lapisan dipilih sedemikian rupa sehingga masing-masing menggunakan fungsi (operasi) dan layanan hanya dari lapisan tingkat rendah

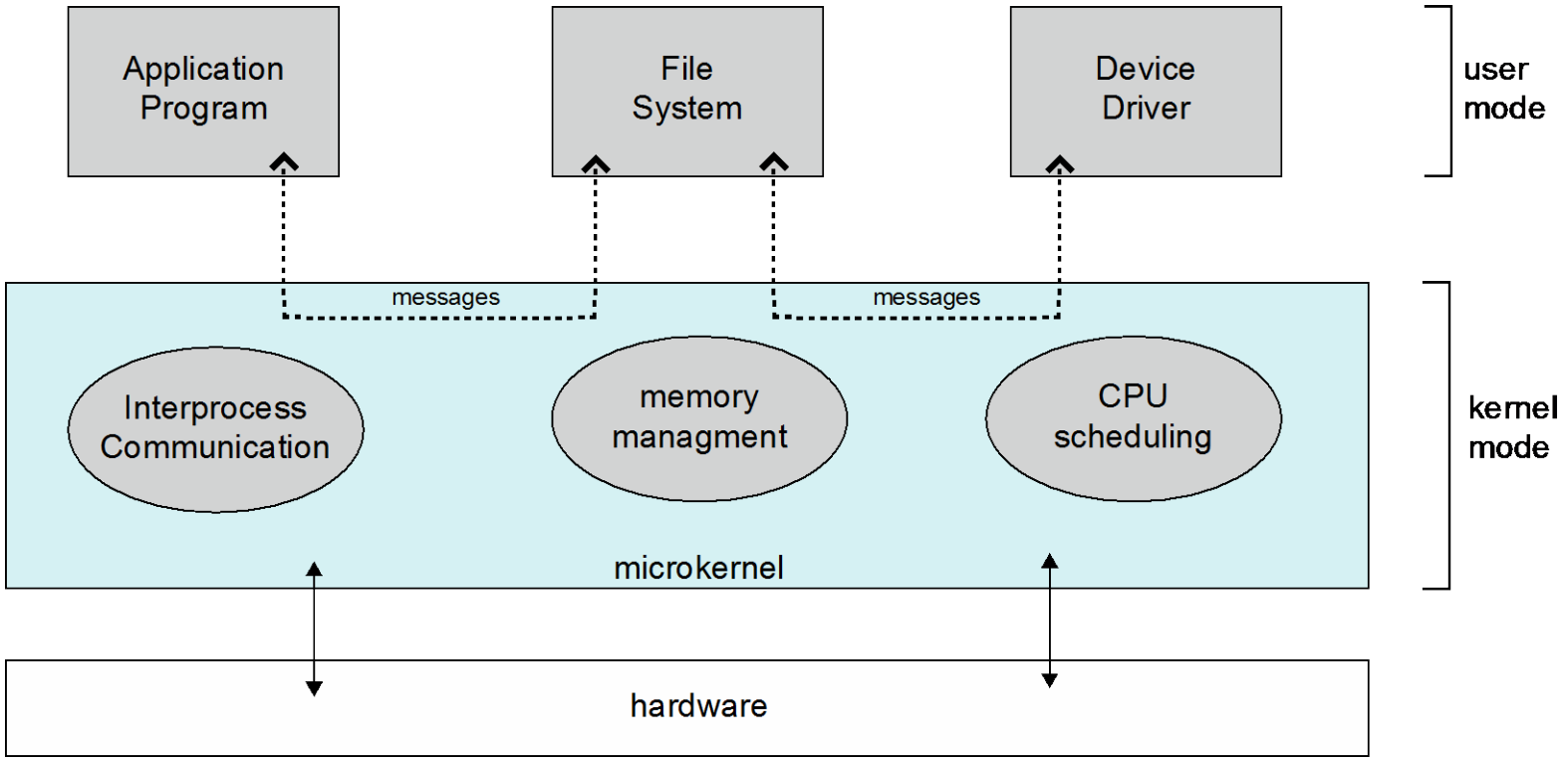


Struktur Sistem Mikrokernel

- Bergerak sebanyak mungkin dari kernel ke ruang
- pengguna **Mesin**contoh**mikrokernel**
  - Kernel Mac OS X (**Darwin**) sebagian berdasarkan Mach
- Komunikasi terjadi antara modul pengguna menggunakan **penyampaian pesan**
- Manfaat:
  - Lebih mudah untuk memperluas mikrokernel
  - Lebih mudah mem-porting sistem operasi ke arsitektur baru Lebih
  - andal (lebih sedikit kode yang dijalankan dalam mode kernel) Lebih
  - aman
- Kerugian:
  - Overhead kinerja ruang pengguna ke komunikasi ruang kernel



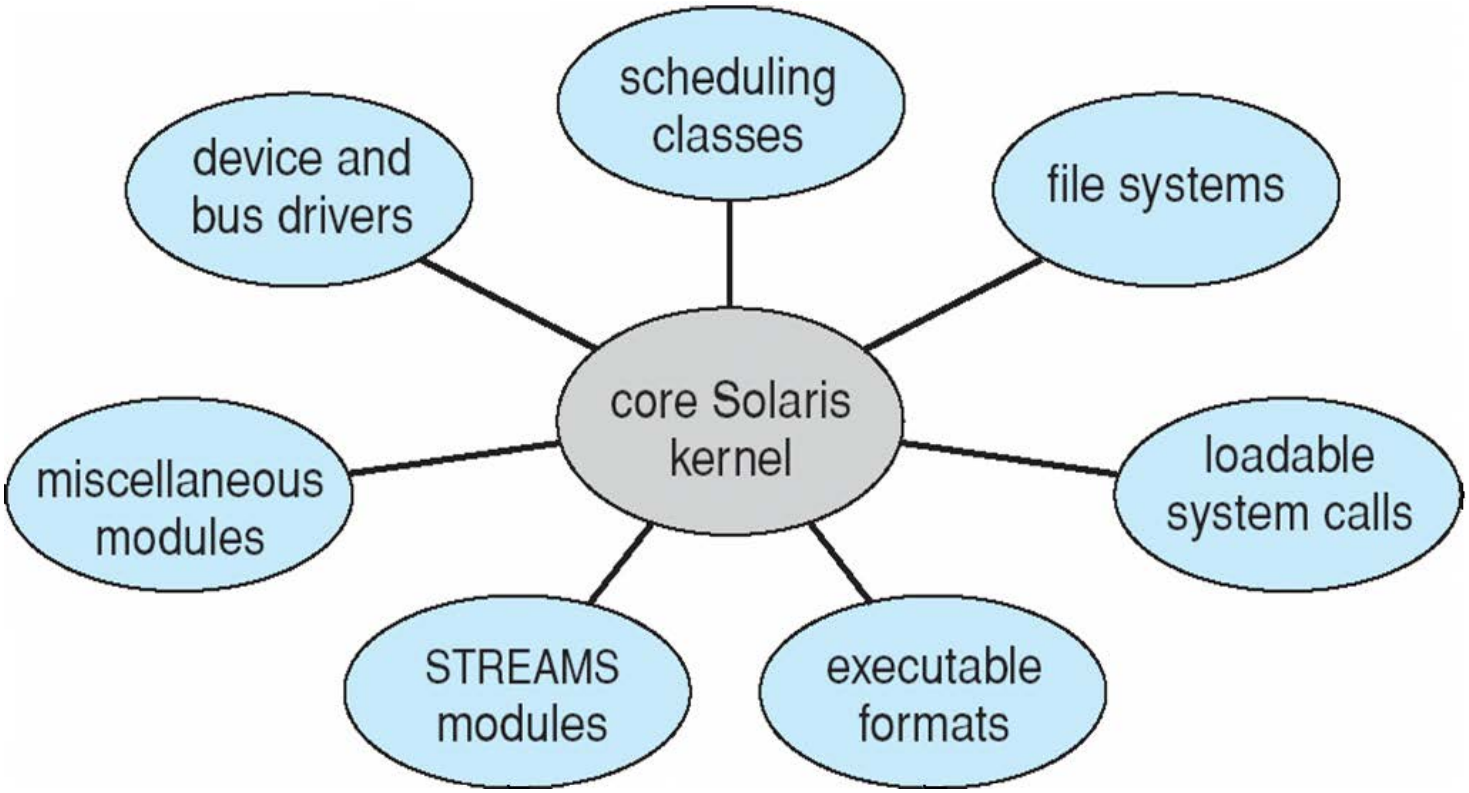
Struktur Sistem Mikrokernel



Modul

- Banyak sistem operasi modern menerapkan**modul kernel yang dapat dimuat**
  - Menggunakan pendekatan berorientasi
  - objek Setiap komponen inti terpisah
  - Masing-masing berbicara satu sama lain melalui antarmuka yang
  - diketahui. Masing-masing dapat dimuat sesuai kebutuhan di dalam kernel
- Secara keseluruhan, mirip dengan lapisan tetapi dengan lebih fleksibel
  - Linux, Solaris, dll

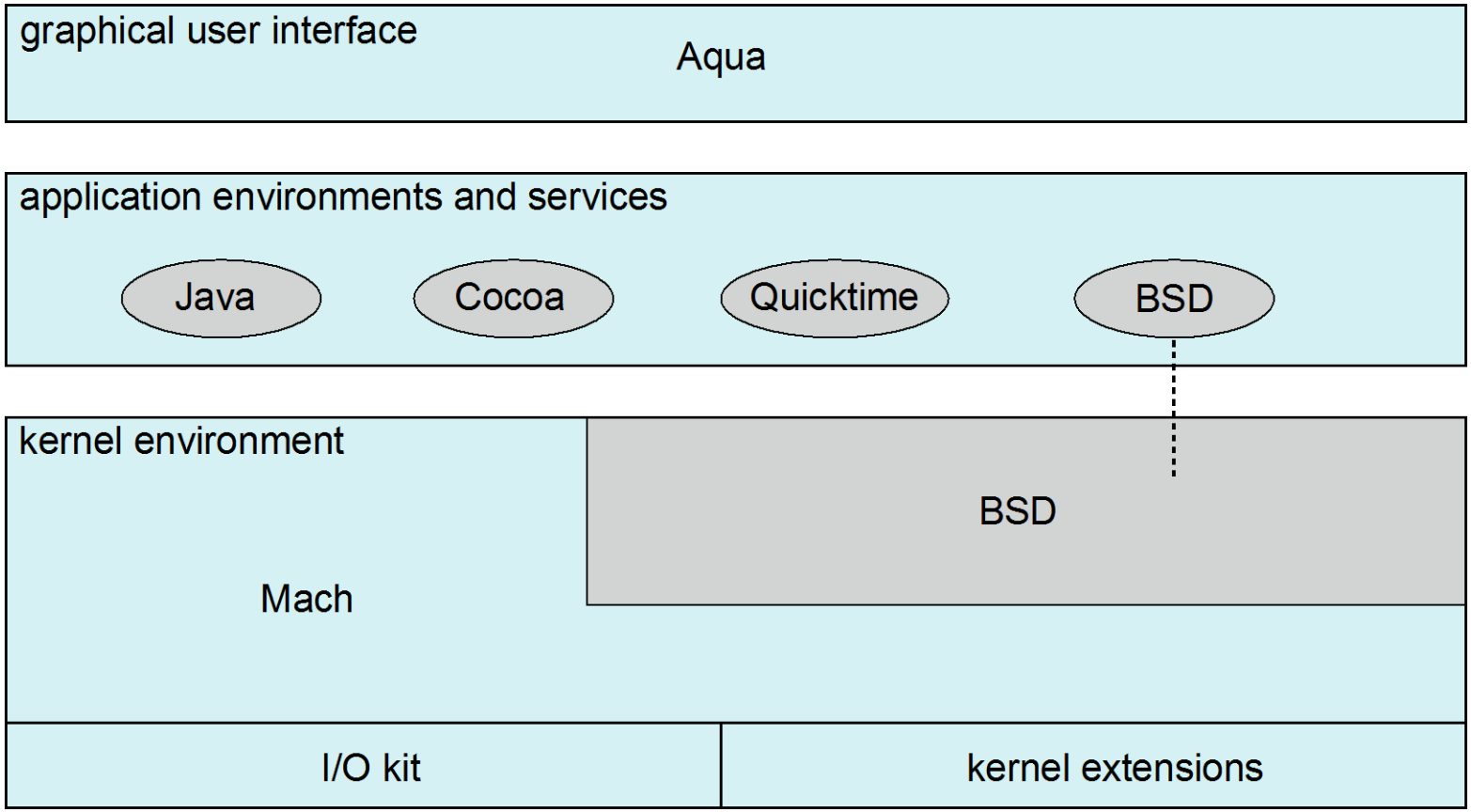
Pendekatan Modular Solaris



Sistem Hibrid

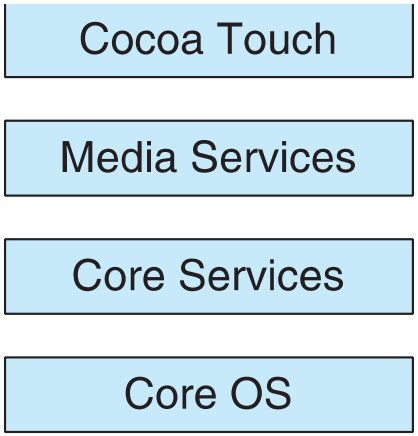
- Sebagian besar sistem operasi modern sebenarnya bukan satu model murni
  - Hybrid menggabungkan beberapa pendekatan untuk mengatasi kebutuhan kinerja, keamanan, kegunaan
  - Kernel Linux dan Solaris di ruang alamat kernel, jadi monolitik, plus modular untuk memuat fungsionalitas secara dinamis
  - Windows sebagian besar monolitik, ditambah mikrokernel untuk subsistem yang berbeda ***kepribadian***
- Apple Mac OS X hybrid, berlapis, **Aqua** UI plus **Biji cokelat** lingkungan pemrograman
  - Di bawah ini adalah kernel yang terdiri dari mikrokernel Mach dan bagian BSD Unix, ditambah kit I/O dan modul yang dapat dimuat secara dinamis (disebut **ekstensi kernel**)

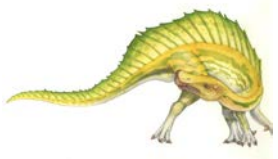
Struktur Mac OS X



iOS

- OS seluler Apple untuk ***iPhone***, ***iPad***
  - Terstruktur di Mac OS X, fungsionalitas tambahan
  - Tidak menjalankan aplikasi OS X secara asli
    - Juga berjalan pada arsitektur CPU yang berbeda (ARM vs. Intel)
  - **Sentuhan Kakao** Objective-C API untuk mengembangkan aplikasi
  - **Layanan media** lapisan untuk grafis, audio, video
  - **Layanan utama** menyediakan komputasi awan, database
  - Sistem operasi inti, berdasarkan kernel Mac OS X

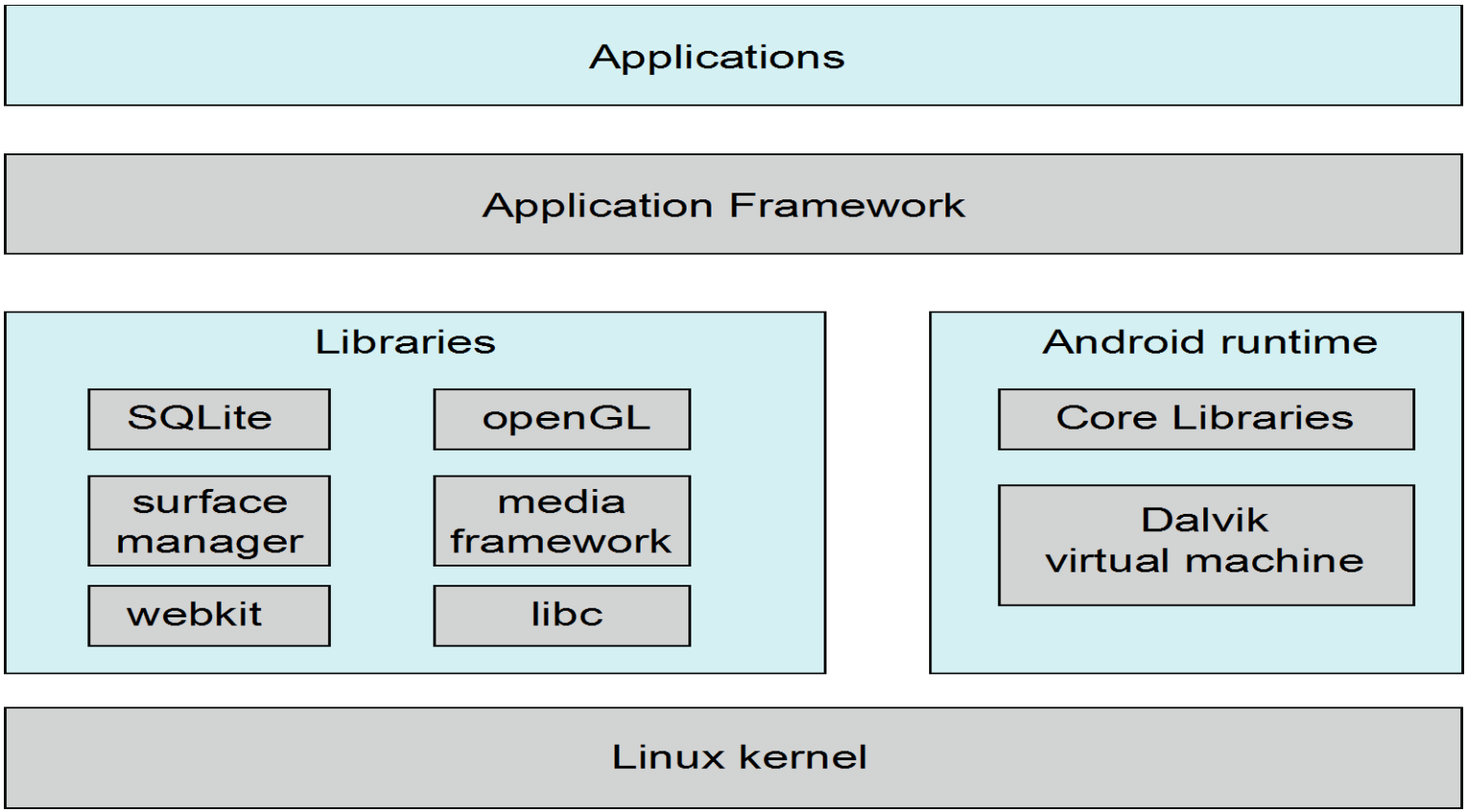




Android

- Dikembangkan oleh Open Handset Alliance (kebanyakan Google)
  - Sumber Terbuka
- Tumpukan yang mirip dengan iOS
- Berdasarkan kernel Linux tetapi dimodifikasi
  - Menyediakan proses, memori, manajemen perangkat-driver
  - Menambahkan manajemen daya
- Lingkungan runtime mencakup kumpulan inti perpustakaan dan mesin virtual Dalvik
  - Aplikasi yang dikembangkan di Java plus Android API
    - File kelas Java dikompilasi ke bytecode Java kemudian diterjemahkan menjadi executable kemudian dijalankan di Dalvik VM
- Pustaka mencakup kerangka kerja untuk peramban web (webkit), basis data (SQLite), multimedia, libc yang lebih kecil

Arsitektur Android



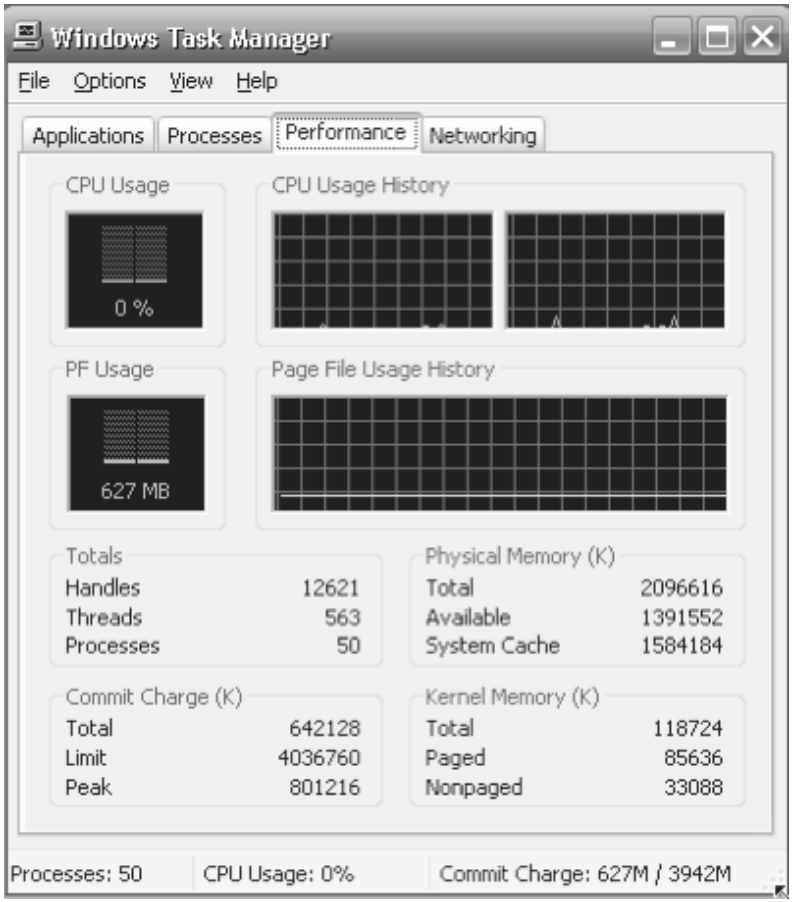
Debugging Sistem Operasi

- **Men-debug** adalah menemukan dan memperbaiki kesalahan,
- atau **bug** menghasilkan OS **file log** berisi informasi kesalahan
- Kegagalan aplikasi dapat menghasilkan **pembuangan inti** file menangkap memori dari proses
- Kegagalan sistem operasi dapat menghasilkan **tempat pembuangan sampah** file yang berisi memori kernel
- Di luar kerusakan, penyetelan kinerja dapat mengoptimalkan kinerja sistem
  - Terkadang menggunakan ***daftar jejak*** kegiatan, dicatat untuk analisis
  - **Pembuatan profil** adalah pengambilan sampel penunjuk instruksi secara berkala untuk mencari tren statistik

Hukum Kernighan: "Debugging dua kali lebih sulit daripada menulis kode di tempat pertama. Oleh karena itu, jika Anda menulis kode secerdas mungkin, menurut definisi Anda, Anda tidak cukup pintar untuk men-debug-nya."

Penyetelan Performa

- Meningkatkan kinerja dengan menghilangkan kemacetan
- OS harus menyediakan sarana komputasi dan menampilkan ukuran sistem perilaku
- Misalnya, program "teratas" atau Windows Task Manager

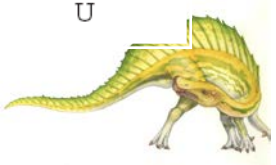


DTrace

- Alat DTrace di Solaris, FreeBSD, Mac OS X memungkinkan instrumentasi langsung pada sistem produksi
- **Probe**aktif ketika kode dijalankan dalam a**pemberi**, menangkap data status dan mengirimkannya ke**konsumen**dari probe tersebut
- Contoh mengikuti

  Panggilan sistem XEventsQueued berpindah dari perpustakaan libc ke kernel dan sebaliknya

```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
  0 -> XEventsQueued                          U
  0   -> _XEventsQueued                       U
  0     -> _X11TransBytesReadable             U
  0     <- _X11TransBytesReadable             U
  0     -> _X11TransSocketBytesReadable       U
  0     <- _X11TransSocketBytesreadable       U
  0     -> ioctl                              U
  0       -> ioctl                            K
  0         -> getf                           K
  0           -> set_active_fd                K
  0           <- set_active_fd                K
  0         <- getf                           K
  0         -> get_udatamodel                 K
  0         <- get_udatamodel                 K
...
  0         -> releasef                       K
  0           -> clear_active_fd              K
  0           <- clear_active_fd              K
  0           -> cv_broadcast                 K
  0           <- cv_broadcast                 K
  0         <- releasef                       K
  0       <- ioctl                            K
  0     <- ioctl                              U
  0   <- _XEventsQueued                       U
  0 <- XEventsQueued                          U
```



Dtrace (Lanj.)

- Kode DTrace untuk merekam jumlah waktu setiap proses dengan UserID 101 dalam mode berjalan (di CPU) dalam nanodetik

```
sched:::on-cpu
uid == 101
{
   self->ts = timestamp;
}

sched:::off-cpu
self->ts
{
   @time[execname] = sum(timestamp - self->ts);
   self->ts = 0;
}
```

```
# dtrace -s sched.d
dtrace: script 'sched.d' matched 6 probes
^C
    gnome-settings-d              142354
    gnome-vfs-daemon              158243
    dsdm                          189804
    wnck-applet                   200030
    gnome-panel                   277864
    clock-applet                  374916
    mapping-daemon                385475
    xscreensaver                  514177
    metacity                      539281
    Xorg                         2579646
    gnome-terminal               5007269
    mixer_applet2                7388447
    java                        10769137
```

Figure 2.21 Output of the D code.

Generasi Sistem Operasi

- Sistem operasi dirancang untuk berjalan di salah satu kelas mesin; sistem harus dikonfigurasi untuk setiap situs komputer tertentu
- **SYSGEN**program memperoleh informasi mengenai konfigurasi khusus dari sistem perangkat keras
  - Digunakan untuk membangun kernel terkompilasi khusus sistem atau systemtuned
  - Dapat menggeneralisasi kode yang lebih efisien daripada satu kernel umum

Boot Sistem

- Saat daya diinisialisasi pada sistem, eksekusi dimulai di lokasi memori tetap
  - ROM firmware digunakan untuk menyimpan kode boot awal
- Sistem operasi harus tersedia untuk perangkat keras sehingga perangkat keras dapat memulainya
  - Sepotong kecil kode –**pemuat bootstrap**, disimpan di**ROM**atau **EEPROM** menempatkan kernel, memuatnya ke dalam memori, dan memulainya
  - Terkadang proses dua langkah di mana**blok boot**di lokasi tetap dimuat oleh kode ROM, yang memuat pemuat bootstrap dari disk
- Pemuat bootstrap umum,**GRUB**, memungkinkan pemilihan kernel dari banyak disk, versi, opsi kernel
- Beban kernel dan sistem kemudian**berlari**

Akhir Bab 2