STRUKTUR SISTEM OPERASI

Akim Manaor Hara Pardede, ST., M.Kom
Email : akimmhp@live.com

Pembahasan

- Komponen-komponen Sistem
- Layanan OS
- System Calls
- System Programs
- System Structure
- Virtual Machines
- System Design & Implementasi
- System Generation

Struktur-struktur OS

- Tanpa adanya modularitas maka fungsi dan struktur OS secara keseluruhan rumit
- Dibagi dalam modul dengan fungsi tertentu, dengan akses (input, output) tertentu

I. Modul/Komponen dalam Sistem

- Modul Manajemen Proses
- Modul Manajemen Memori
- Modul Manajemen Storage/Data
- Modul Manajemen I/O dan Berkas (File)
- Modul Proteksi
- Modul Networking
- Modul Interface dengan user (command interpreter)

1. Manajemen Proses

- Proses adalah program yang dieksekusi, memerlukan resource, CPU time, memory, file, I/O device.
- OS bertanggung jawab dalam :
  - Create & delete ; baik proses user maupun sistem
  - Suspend & meneruskan proses
  - Mendukung mekanisme-mekanisme sinkronisasi proses
  - Mendukung mekanisme proses communication
  - Mendukung mekanisme penanganan deadlock

2. Manajemen Memori Utama

- Memory merupakan array words/byte dalam jumlah besar. Akses share data secara cepat oleh CPU dan I/O device
- Volatile storage device
- OS bertanggung jawab dalam :
  - Keep track bagian mana dari memori yang sedang digunakan & oleh siapa
  - Memutuskan proses-proses mana yang di-load ke ruang memori saat available
  - Alokasi & dealokasi ruang memori

3. Manajemen Secondary-Storage

- Back up main memory, non-volatile
- Data dan program disimpan dalam secondary storage (penyimpanan sekunder; disk)
- OS bertanggung jawab dalam :
  - Bagaimana mengelola ruang yang kosong dalam storage
  - Bagaimana mengalokasi storage
  - Bagaimana melakukan scheduling penggunaan disk

4. Manajemen I/O

- OS bertanggung jawab dalam :
  - “menyembunyikan” kekhususan perangkat keras tertentu dari user
  - Melakukan optimalisasi dalam akses
    - Buffer cache system : menampung sementara data dari/ke piranti I/O
    - Spooling : melakukan penjadwalan pemakaian I/O sistem supaya lebih efisien (antrian, dsb)
  - Interface device-driver : open, read, write, close
- Drivers untuk spesifik perangkat keras :
  - Menyediakan driver untuk melakukan operasi detail untuk perangkat keras tertentu

Manajemen File

- Berkas (File) adalah kumpulan informasi yang berhubungan (sesuai dengan tujuan pembuat berkas tersebut). Biasanya berkas merepresentasikan program dan data
- OS bertanggung jawab dalam :
  - Pembuatan dan penghapusan file
  - Pembuatan dan penghapusan direktori
  - Mendukung primitif-primitif manipulasi file dan direktori
  - Pemetaan file dalam secondary storage
  - Backup file dalam media yang stabil (non-volatile)

5. Networking (Distributed System)

- Distributed system : kumpulan prosesor yang terdistribusi, tidak berbagi (share) memory atau clock. Setiap prosesor memiliki memori lokal masing-masing
- Prosesor-prosesor dalam sistem terhubung dalam jaringan komunikasi
- Sebagai pengatur (protokol) dalam komunikasi data
- Menentukan strategi-strategi menangani masalah-masalah komunikasi
- Mengatur network file system
- Dengan adanya shared resource :
  - Peningkatan kecepatan komputasi
  - Peningkatan penyediaan data
  - Meningkatkan reliabilitas (kehandalan)

6. Sistem Proteksi

- Mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor atau user ke resource-resource dalam sistem komputer
- Mekanisme proteksi :
  - Dapat membedakan pemakaian yang sah (authorized) & yang tidak sah (unauthorized)
  - Spesifikasi kontrol yang dikenakan
  - Menyediakan alat untuk pemberlakuan sistem

7. Command Interpreter (1)

- Memungkinkan sistem berkomunikasi dengan user melalui perintah-perintah menjalankan proses yang telah didefinisikan dan parameternya serta melakukan respon
- OS menunggu instruksi dari user (*command driven*)
- Control statement berhubungan dengan :
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking

Command Interpreter (2)

- Program yang membaca instruksi dan mengartikan control statements (keinginan pengguna) umumnya disebut :
  - control-card interpreter
  - command-line interpreter
  - UNIX Shell
- Command-Interpreter System sangat bervariasi dari satu sistem operasi ke sistem operasi yang lain dan disesuaikan dengan tujuan dan teknologi I/O peranti yang ada.
  - Contohnya : CLI, Windows, Pen-based (touch), dll

II. Layanan Operating System (1)

- Eksekusi program : load program user ke memory dan menjalankannya (*run*)
- Operasi-operasi I/O : pengguna tidak bisa mengontrol I/O secara langsung (untuk efisiensi & keamanan), sistem harus bisa menyediakan mekanisme untuk melakukan operasi I/O
- Manipulasi file system : read, write, create & delete

Layanan Operating System (2)

- Komunikasi antar proses :
  - Baik yang run di komputer yang sama atau berlainan via jaringan. Implementasi melalui shared memory atau message passing
- Error detection
  - Menjamin komputasi yang benar dengan mendeteksi error : CPU, memori, I/O device, atau user program

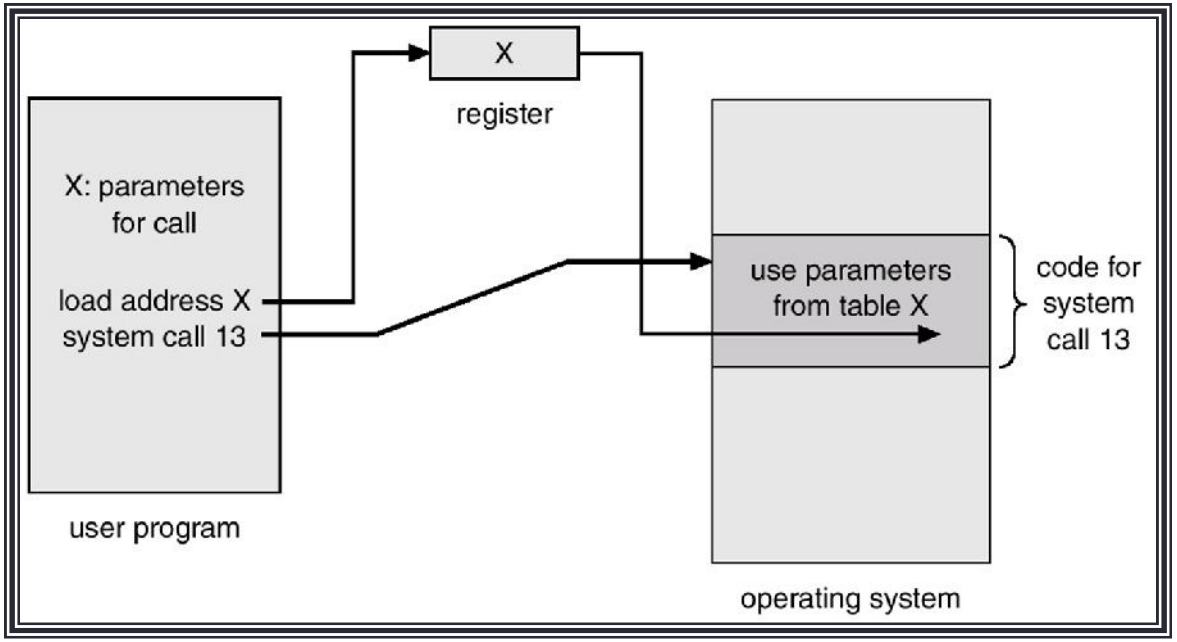
Layanan OS Internal

- Resource allocation
  - Alokasi resources bagi sejumlah user atau job yang running pada saat yang sama
- Accounting
  - Mencatat user mana, berapa banyak, dan resource komputer apa saja (untuk account billing atau penghitungan statistik)
- Protection
  - Menjamin agar semua akses ke resource-resource sistem terkendali

III. System Call

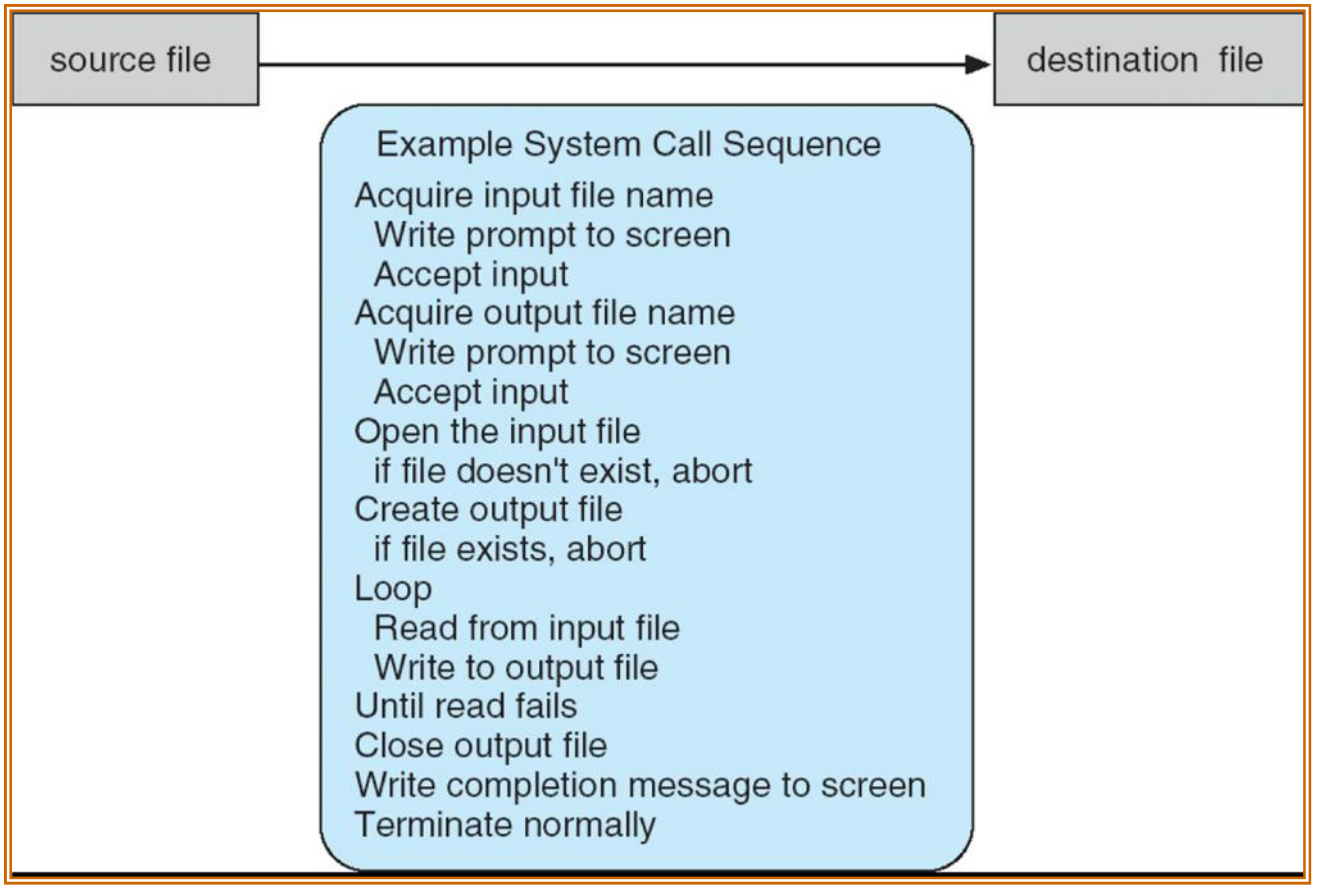
- Menyediakan antarmuka antara proses (program yang run) dengan OS
- Umumnya dalam bentuk instruksi bahasa assembly
- Bahasa untuk system programming tingkat tinggi biasanya memungkinkan system call dilakukan langsung
  - Misal C, C++, Bliss, PL/360
- Tiga metode untuk passing parameter antara running program dan OS :
  - Pass parameter melalui register
  - Meyimpan parameter dalam blok atau tabel pada memory, dan alamat tabel di-passing sebagai parameter dlm register
  - Menyimpan parameter (*push*) ke dalam stack (oleh program), dan *pop off* parameter pada stack (oleh OS)

Passing Parameter menggunakan Tabel

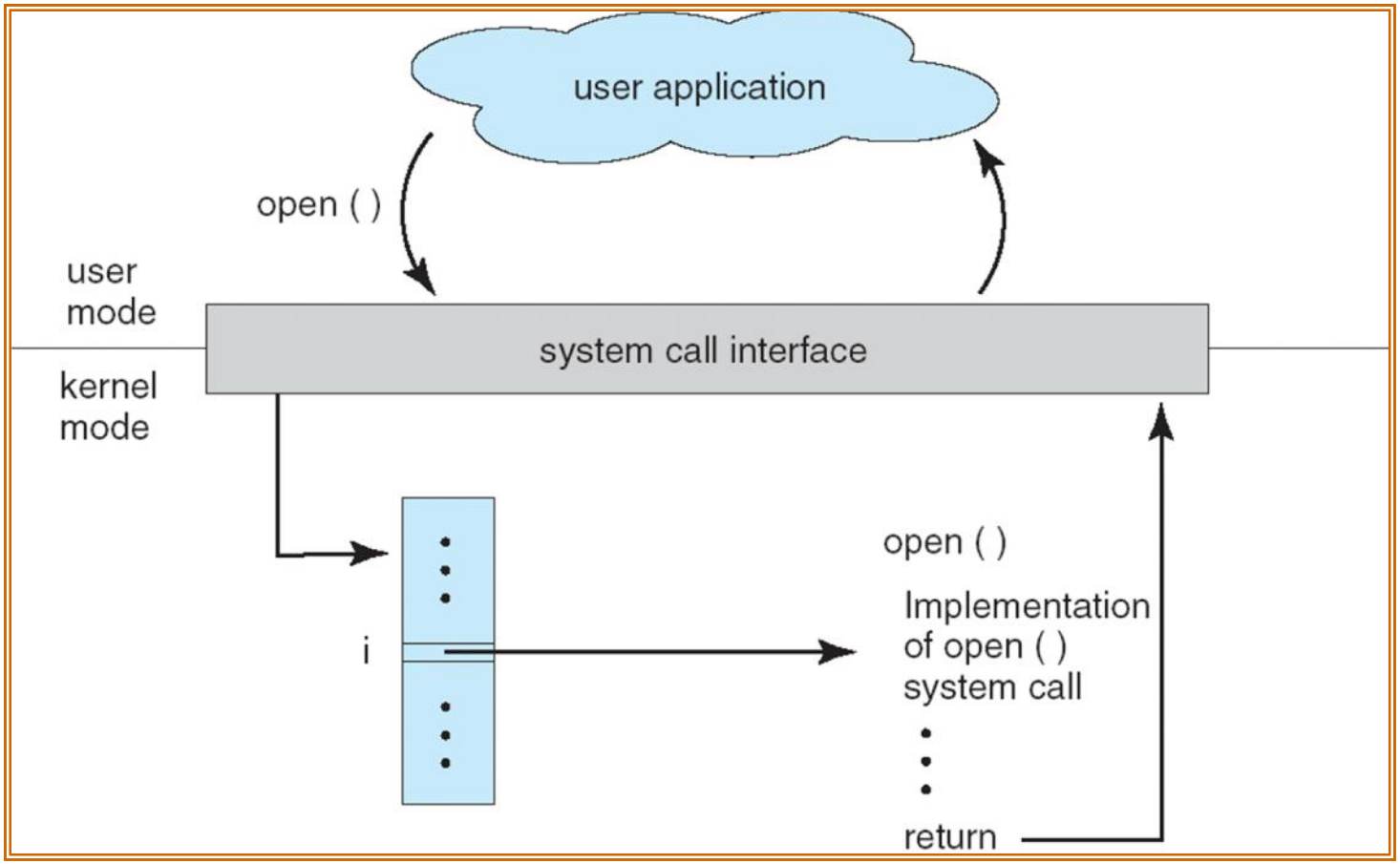


Contoh System Call

- Urutan System call untuk meng-copy isi file ke file yang lain

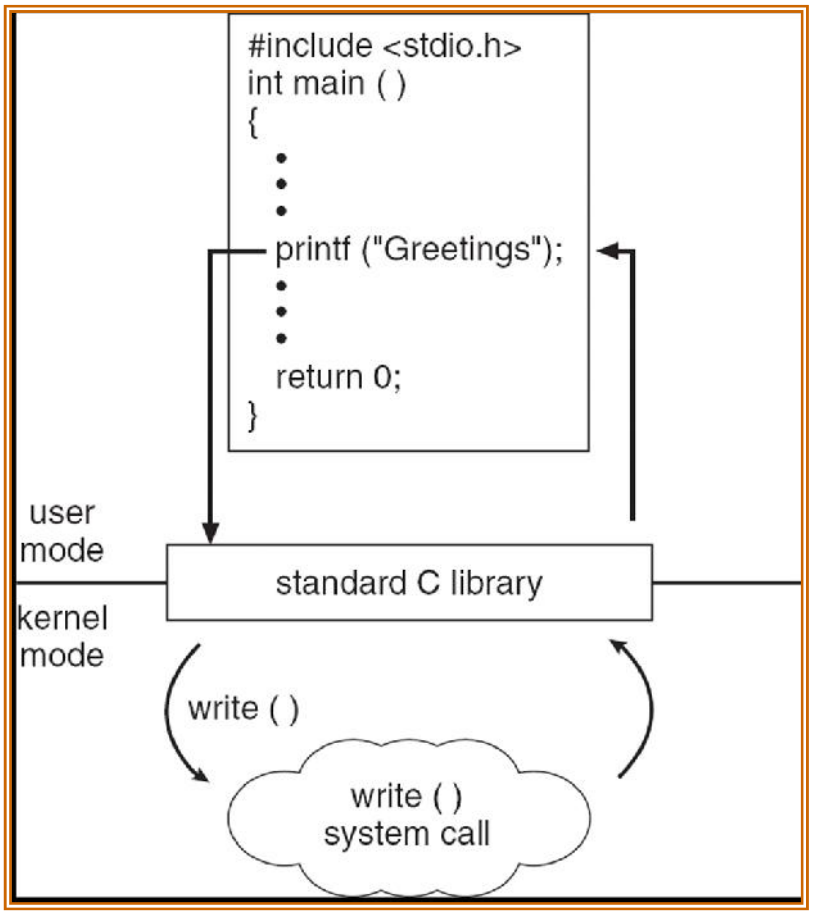


Hubungan API – System Call – OS



Standard C Library Example

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



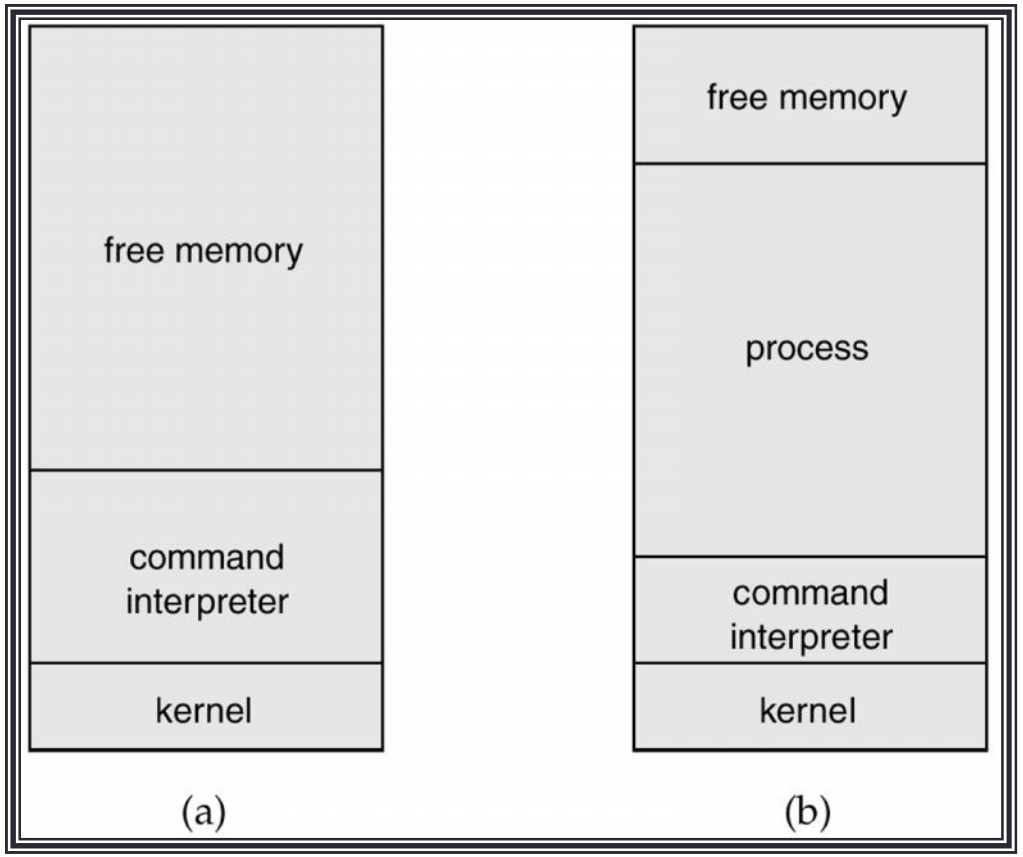
Jenis-jenis System Call

- Process Control
- File Manipulation
- Device Manipulation
- Information Maintenance
- Communication

Process Control

- Selesai, *abort*
- *Load*, eksekusi
- Membuat dan mengakhiri proses
- Mengambil dan mengeset atribut proses
- Menunggu waktu
- *Wait* event, *signal* event
- Alokasi dan pengosongan memori

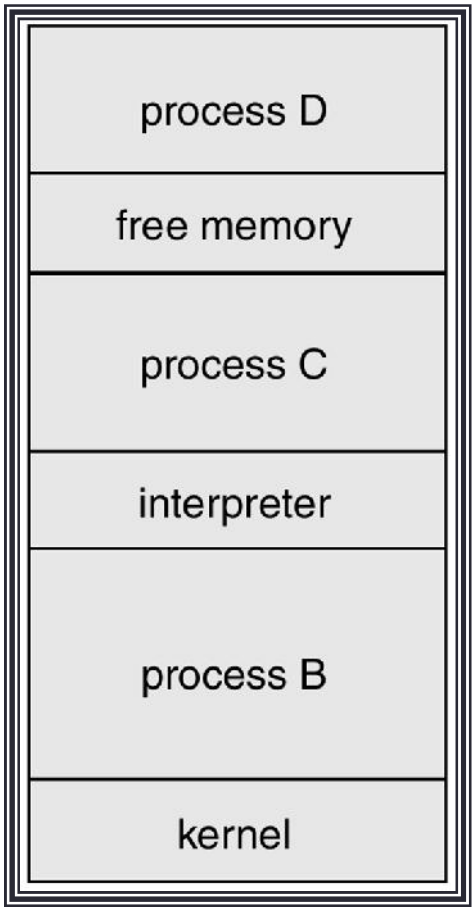
Eksekusi MS-DOS



At System Start-up

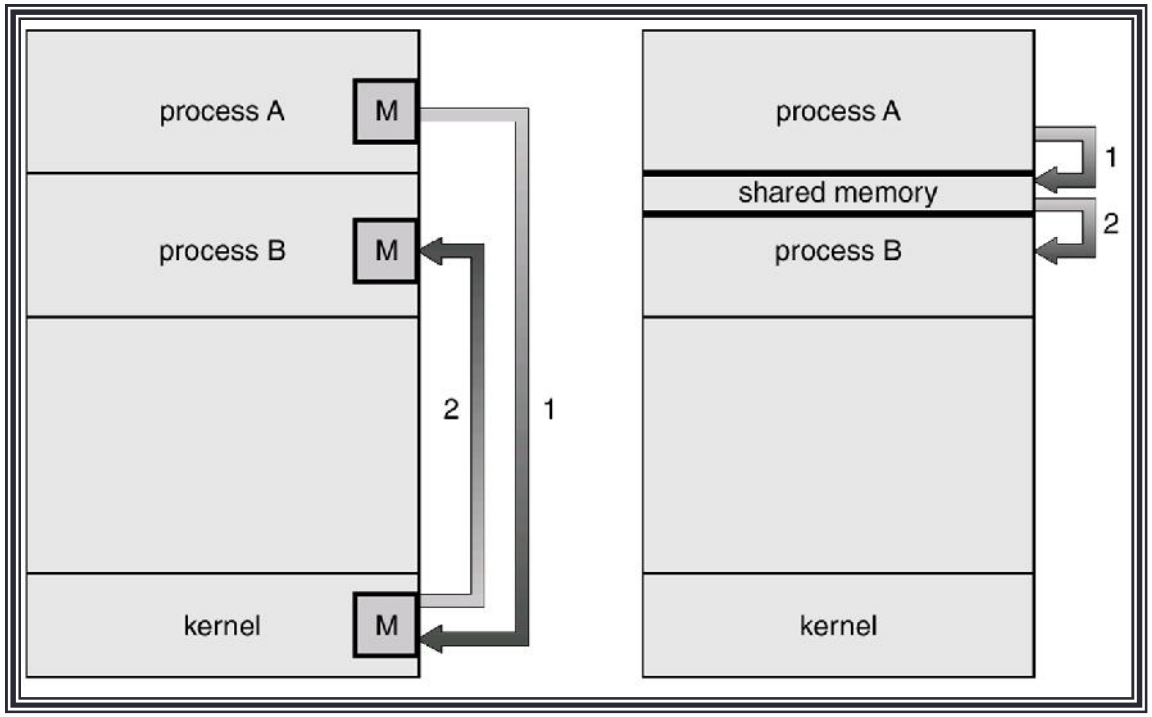
Running a Program

UNIX Menjalankan Multiple Program



Model Komunikasi

- Komunikasi dapat dilakukan dengan cara message passing atau shared memory



Message Passing

Shared Memory

IV. Pemrograman Sistem

- Pemrograman sistem menyediakan lingkungan yang memungkinkan pengembangan program dan eksekusi berjalan dengan baik
- Dapat dikategorikan :
  - Manipulasi Berkas (*File*)
  - Informasi Status : tanggal, jam, jumlah memori, disk, dll
  - Modifikasi Berkas
  - Mendukung bahasa pemrograman : kompilator, assembly, interpreter
  - Loading & eksekusi program
  - Komunikasi : menyediakan mekanisme komunikasi antara proses, user dan sistem komputer yang berbeda
- Dari sisi user, operasional sistem dilakukan dengan system program, bukan system call

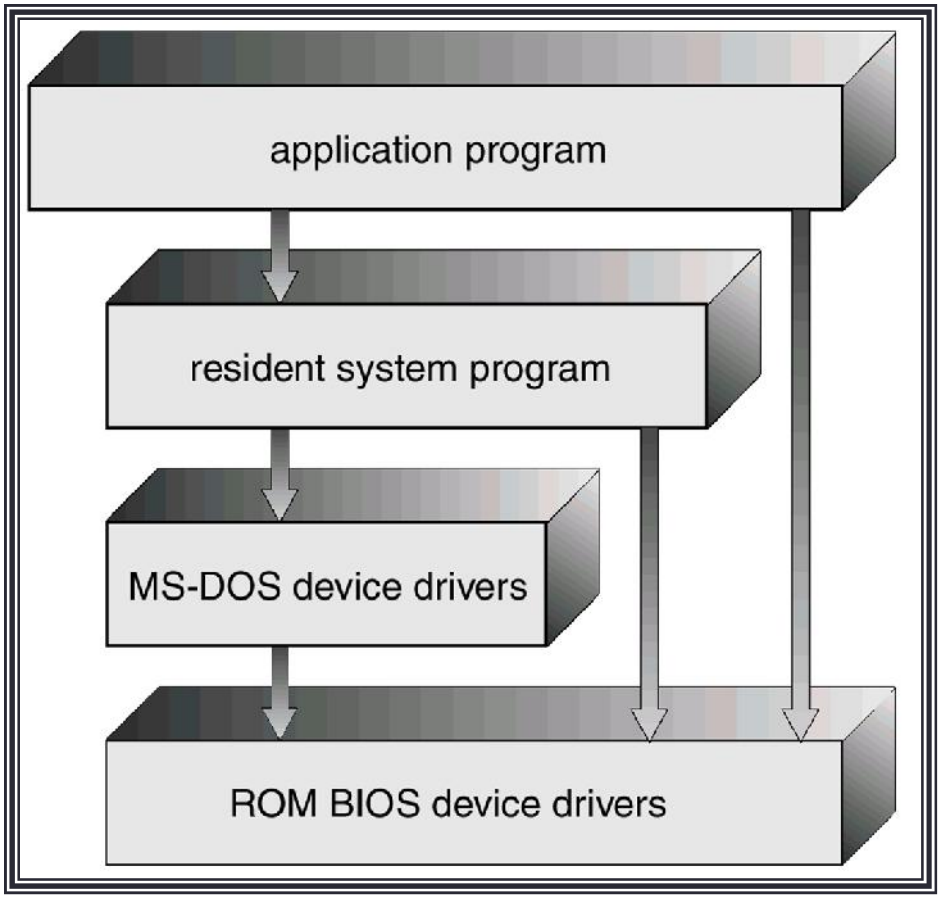
V. Struktur Sistem

- Struktur Sederhana
- Metode Pendekatan Terlapis (Layered Approach)
- Mikrokernel

Struktur Sederhana

- Dimulai dengan sistem yang kecil, sederhana dan terbatas kemudian berkembang dengan cakupan original
- Struktur sistem MS-DOS :
  - disusun untuk mendukung fungsi yang banyak pada ruang yang kecil

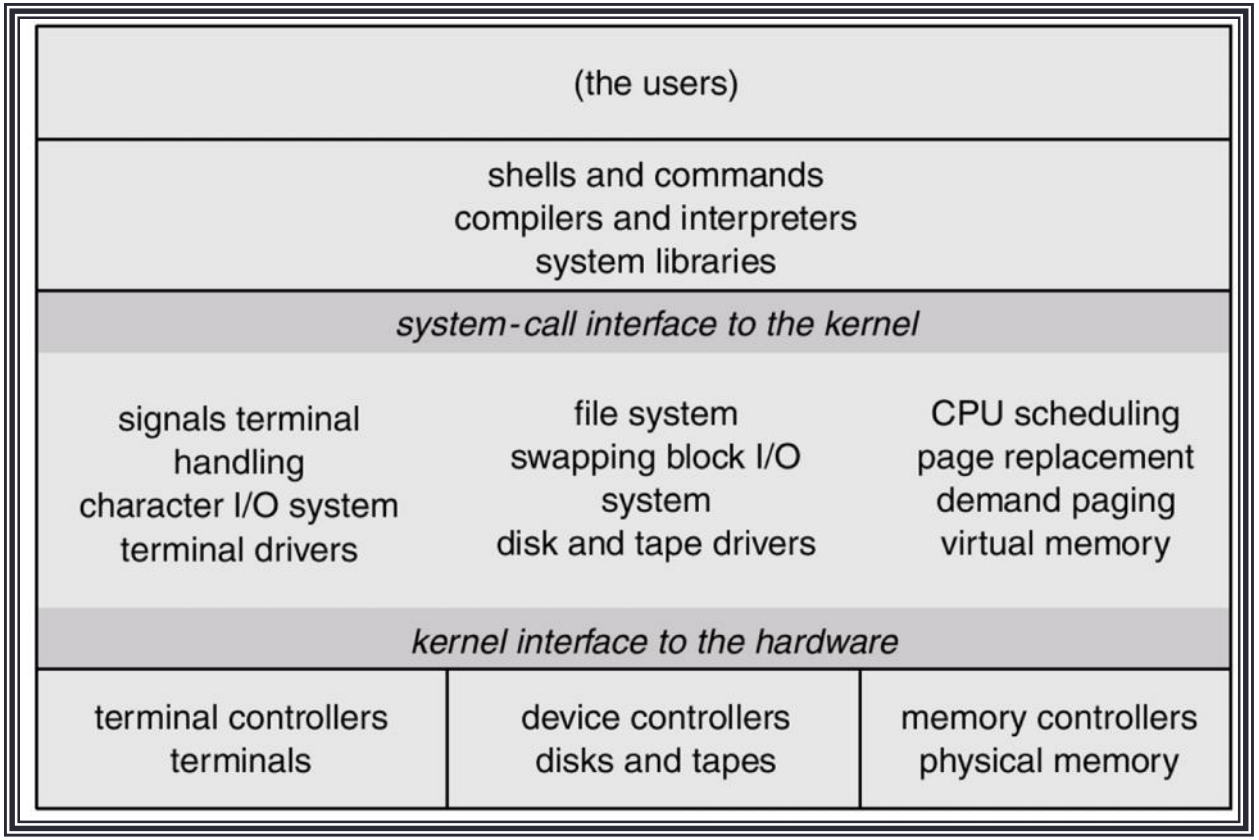
Struktur Lapisan MS-DOS



Struktur Sistem UNIX (1)

- Memiliki struktur yang terbatas
- Terdiri dari 2 bagian :
  - Kernel :
    - Berada dibawah antarmuka system call dan diatas hardware
    - Menyediakan sistem berkas, penjadualan CPU, manajemen memori, device driver, dan fungsi OS lainnya
  - Program Sistem

Struktur Sistem UNIX (2)



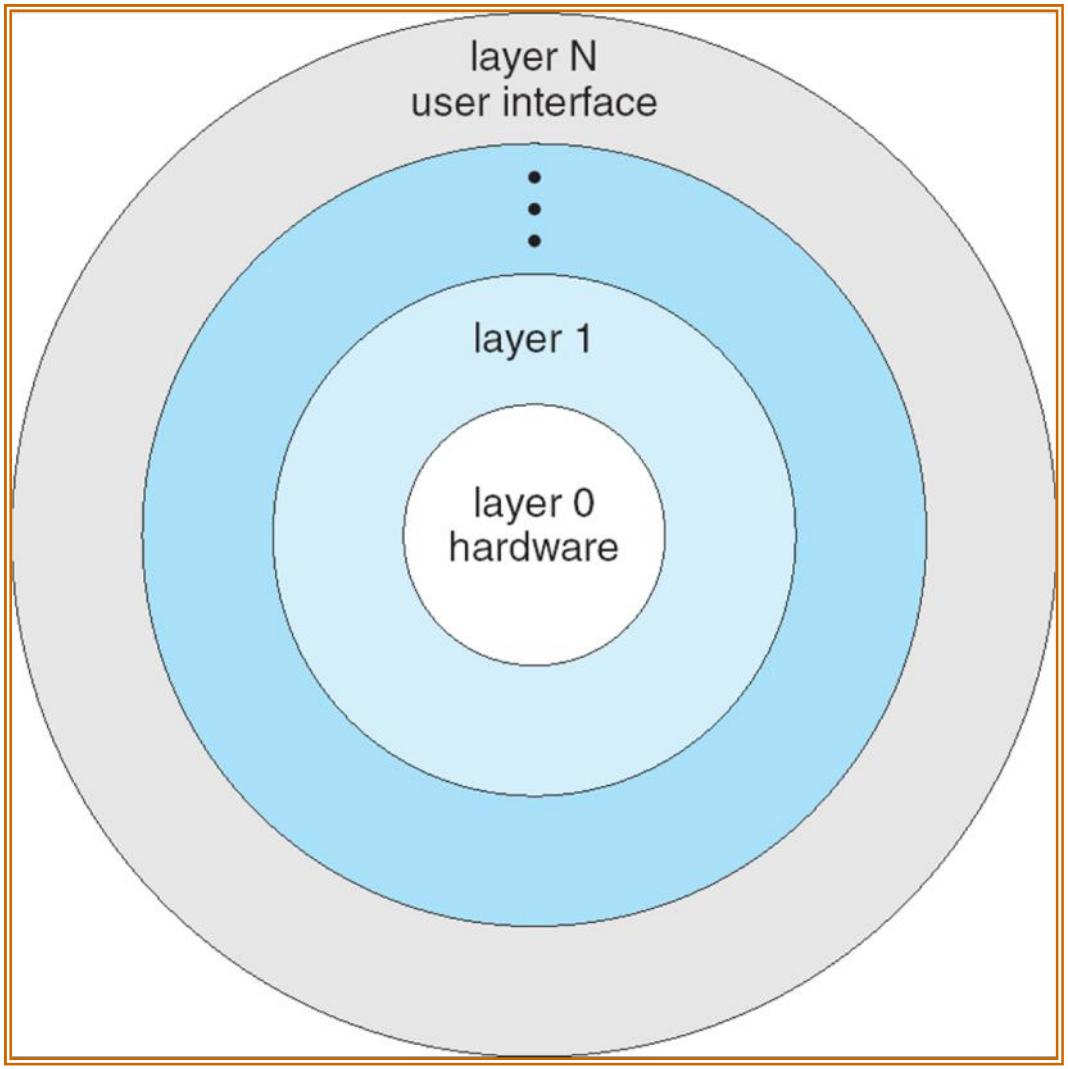
Pendekatan Terlapis (Layered Approach) (1)

- Lapisan adalah implementasi dari objek abstrak yang merupakan *enkapsulasi* dari data dan operasi yang bisa memanipulasi data tersebut
- Lapisan paling bawah : perangkat keras
- Lapisan paling atas : antarmuka pengguna

Pendekatan Terlapis (Layered Approach) (2)

- Keuntungan : modularitas
  - Mempermudah *debug* dan verifikasi sistem
  - Lapisan pertama bisa di-debug tanpa mengganggu sistem yang lain
- Kesulitan :
  - Hanya bisa menggunakan lapisan dibawahnya
  - Tidak efisien dibandingkan tipe yang lain

Lapisan Operating System

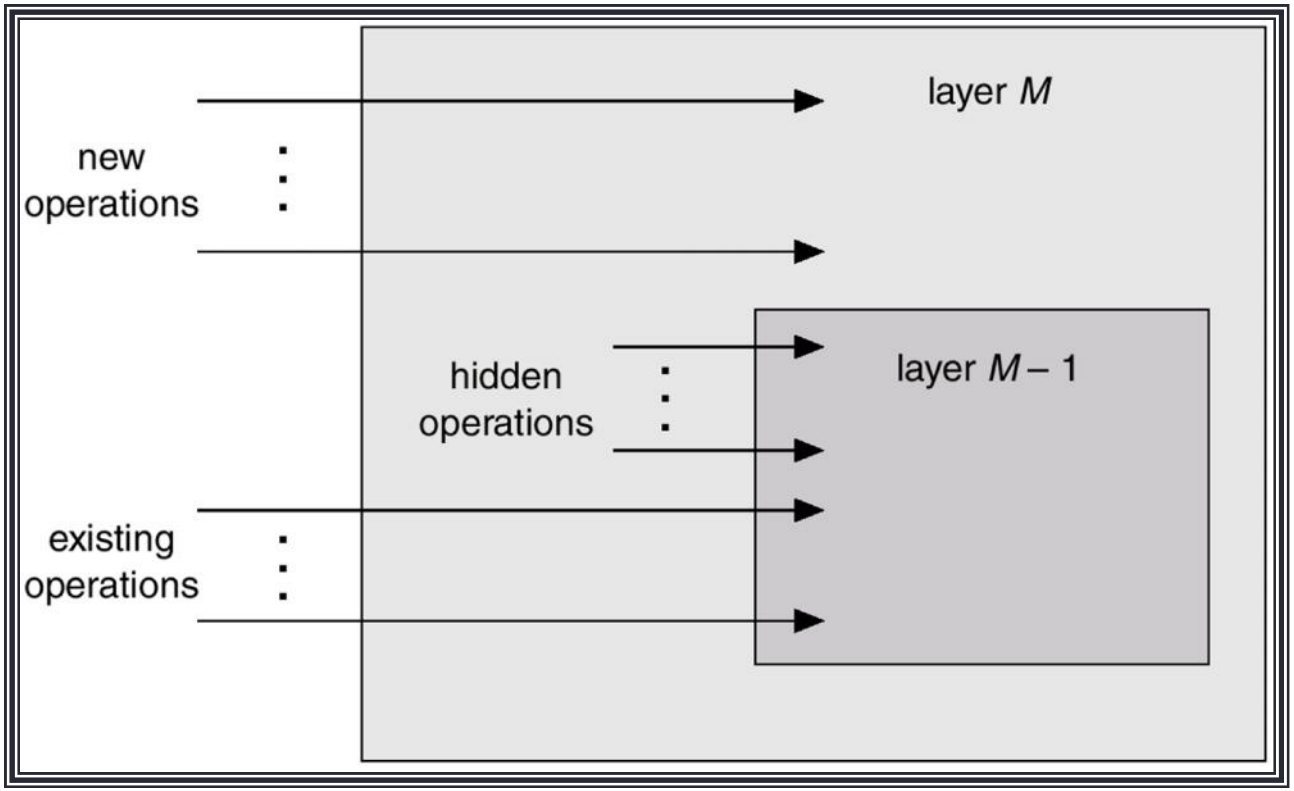


System Structure – Layered Approach

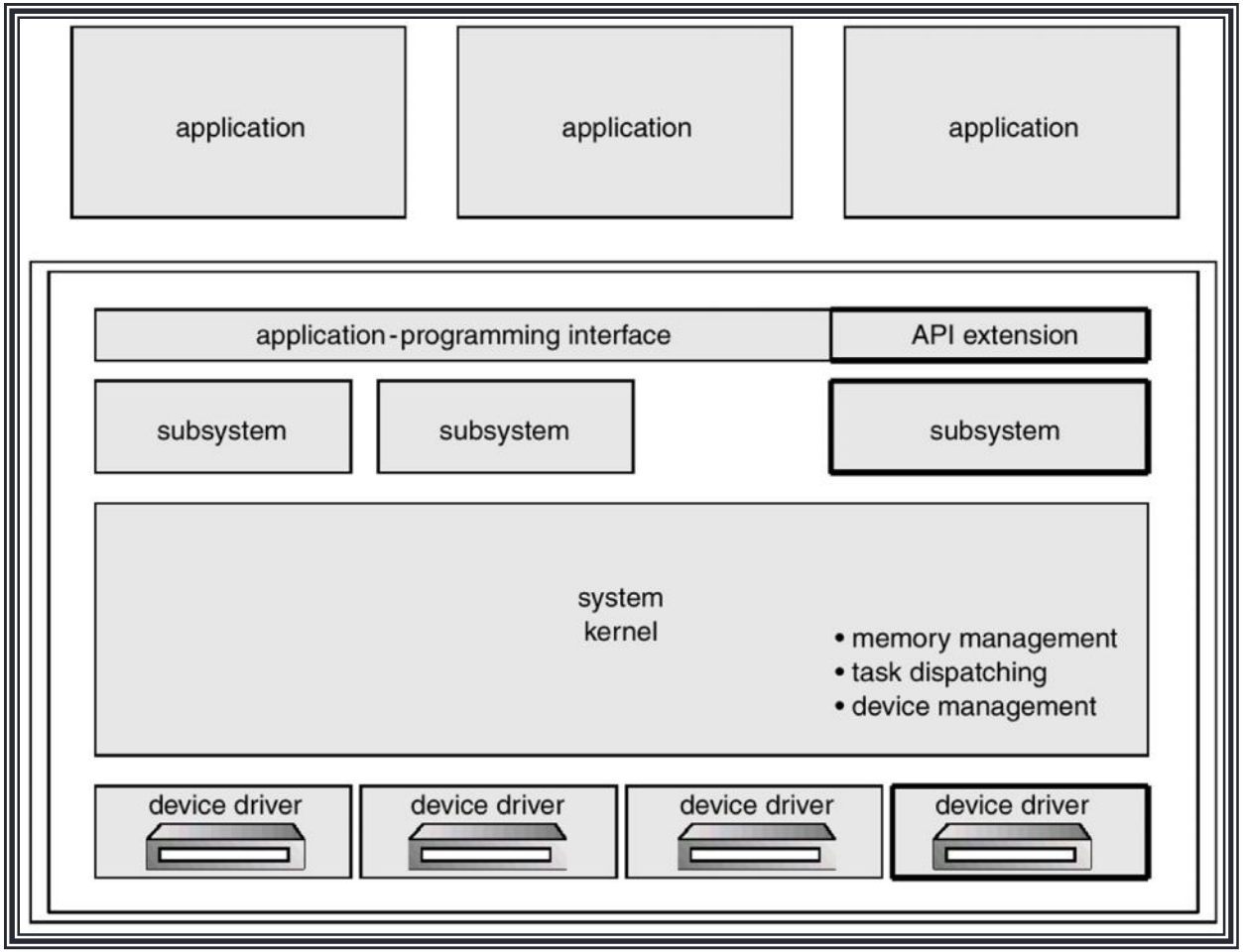
- Suatu rancangan yang pertama digunakan dalam OS, terdiri atas 6 layer : THE Operating System (Dijkstra, 1968)
  - Level 5 : user program
  - Level 4 : buffering untuk input & output device
  - Level 3 : operator-console device driver
  - Level 2 : memory management
  - Level 1 : CPU scheduling
  - Level 0 : hardware

  THE : Technische Hogeschool at Eindhoven

Abstraksi Lapisan Operasi OS



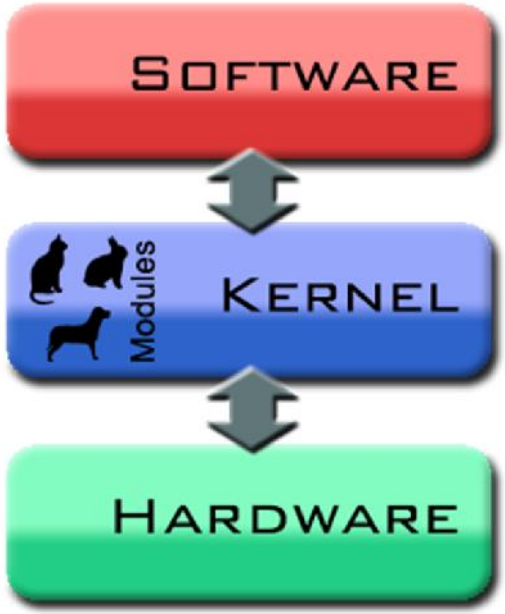
Struktur Lapisan OS/2



Kernel

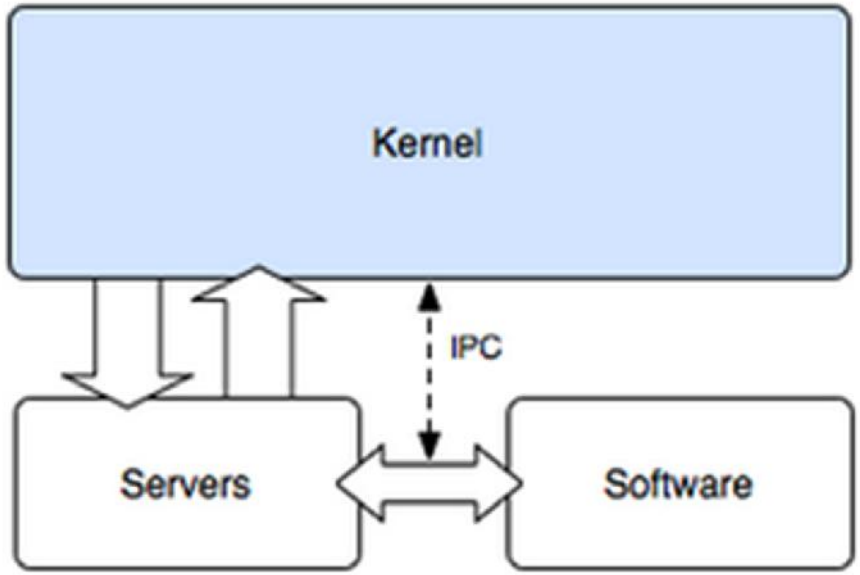
- Kategori kernel :
  - *Monolithic kernel*. Kernel yang menyediakan abstraksi akses ke perangkat keras yang kaya dan handal. Semua layanan OS dilakukan pada kernel .
  - *Microkernel*. Kernel yang menyediakan hanya sekumpulan kecil abstraksi perangkat keras sederhana, dan menggunakan aplikasi-aplikasi yang disebut sebagai server untuk menyediakan fungsi-fungsi lainnya.
  - *Hybrid* (modifikasi dari microkernel). Kernel yang mirip microkernel, tetapi ia juga memasukkan beberapa service tambahan di kernel (*network stack*, *file system*) agar menjadi lebih cepat.
  - *Exokernel*. Kernel yang tidak menyediakan sama sekali abstraksi hardware, tapi ia menyediakan sekumpulan library yang menyediakan fungsi-fungsi akses ke perangkat keras secara langsung.

Diagram Monolithic Kernel



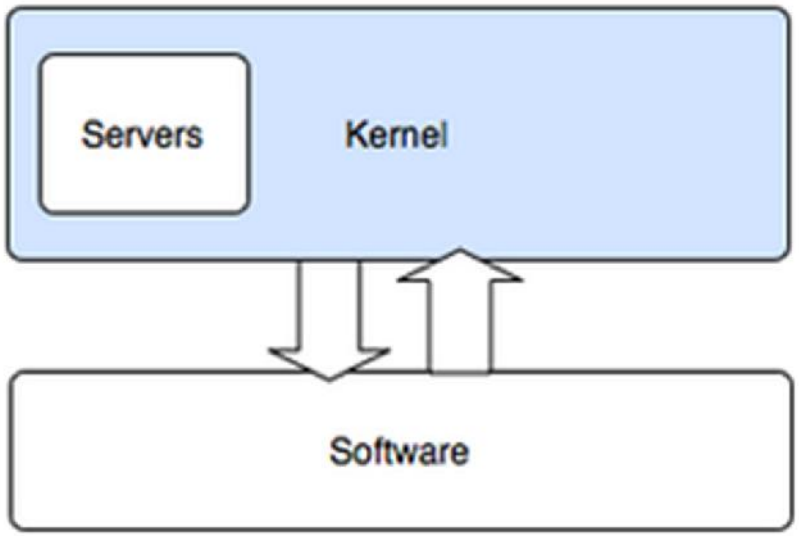
- Semua code pada address space yang sama (*kernel space*)
- Meningkatkan performance system
- Mudah dalam hal design dan implementasi
- Bugs dapat menyebabkan sistem *crash*

Diagram Mikrokernel



- Banyak layanan OS yang run pada *user space* untuk me-minimalisasi kernel (seperti networking)
- Modularity : mudah dalam me-memaintain code

Diagram Hybrid Kernel



- Running beberapa layanan OS (network stack, file system) dalam kernel space untuk mengurangi *performance overhead* dari metode microkernel, tetapi tetap menjalankan kernel code (seperti device driver) sebagai server di *user space*

Mikrokernel (1)

- Menyusun sistem operasi dengan menghapus semua komponen yang tidak esensial dari *kernel*, dan mengimplementasikannya sebagai sistem program dan level pengguna
- Fungsi utama : mendukung fasilitas komunikasi antara program klien dan bermacam-macam layanan yang juga berjalan di *user-space*

Mikrokernel (2)

- Keuntungan :
  - Ketika layanan baru akan ditambahkan ke *user-space*, *kernel* tidak perlu di-modif
  - OS lebih mudah ditempatkan (*porting*) pada suatu desain perangkat keras ke desain perangkat keras lainnya (asrsitektur sistem yang baru)
  - Mendukung keamanan & reliabilitas lebih
- Contoh sistem operasi :
  - Tru64 UNIX, MacOSX, QNX

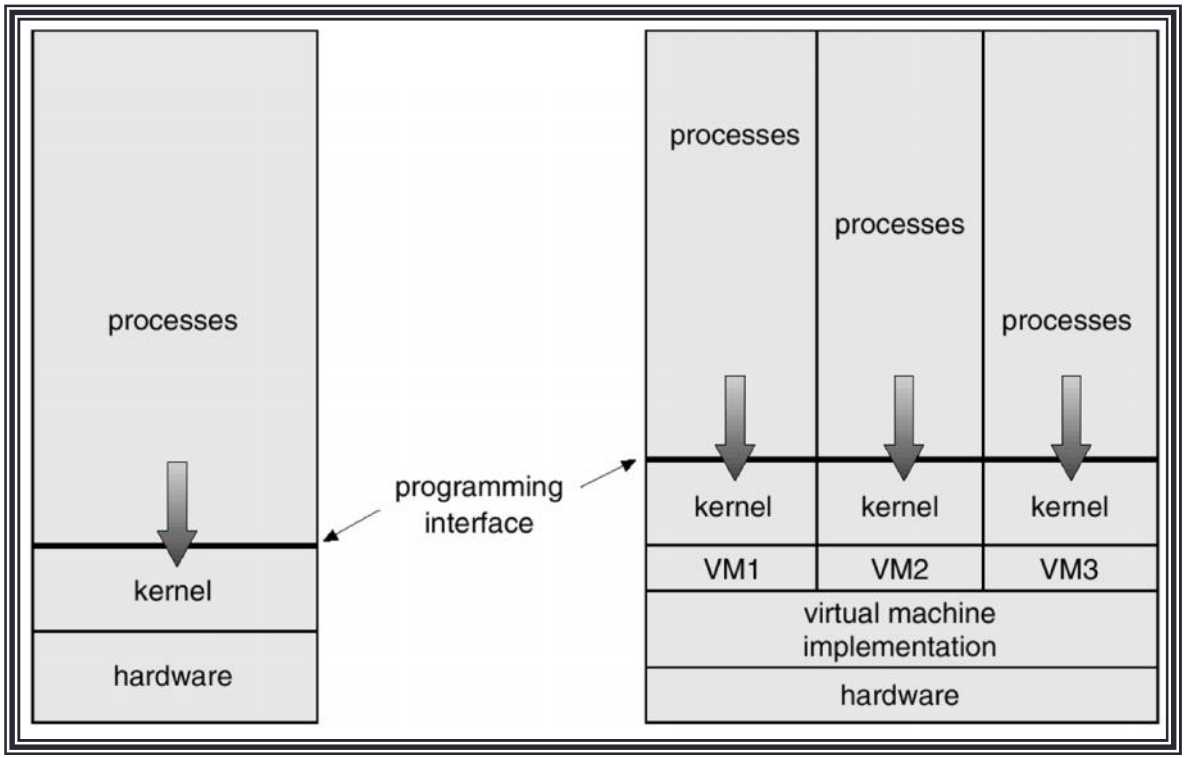
Virtual Machine (VM) (1)

- Menggunakan layered approach
- Melihat hardware dan kernel OS sebagai suatu “hardware”
- Menyediakan interface yang identik dengan *underlaying bare hardware*
- OS menyediakan illusion dari banyak proses yang masing-masing berjalan pada prosesornya serta memorinya (virtual) sendiri

Virtual Machine (VM) (2)

- Resource dari komputer fisiknya di-share menjadi sejumlah mesin-mesin virtual
  - CPU scheduling yang menciptakan penampilan seakan-akan user memiliki prosesor sendiri
  - Spooling & file system menyediakan *virtual card readers* dan *virtual line printers*
  - Sebuah *time-sharing terminal user* berlaku sebagai virtual console
- VM software membutuhkan ruang di dalam disk untuk menyediakan memori virtual dan *spooling*, yaitu sebuah disk virtual

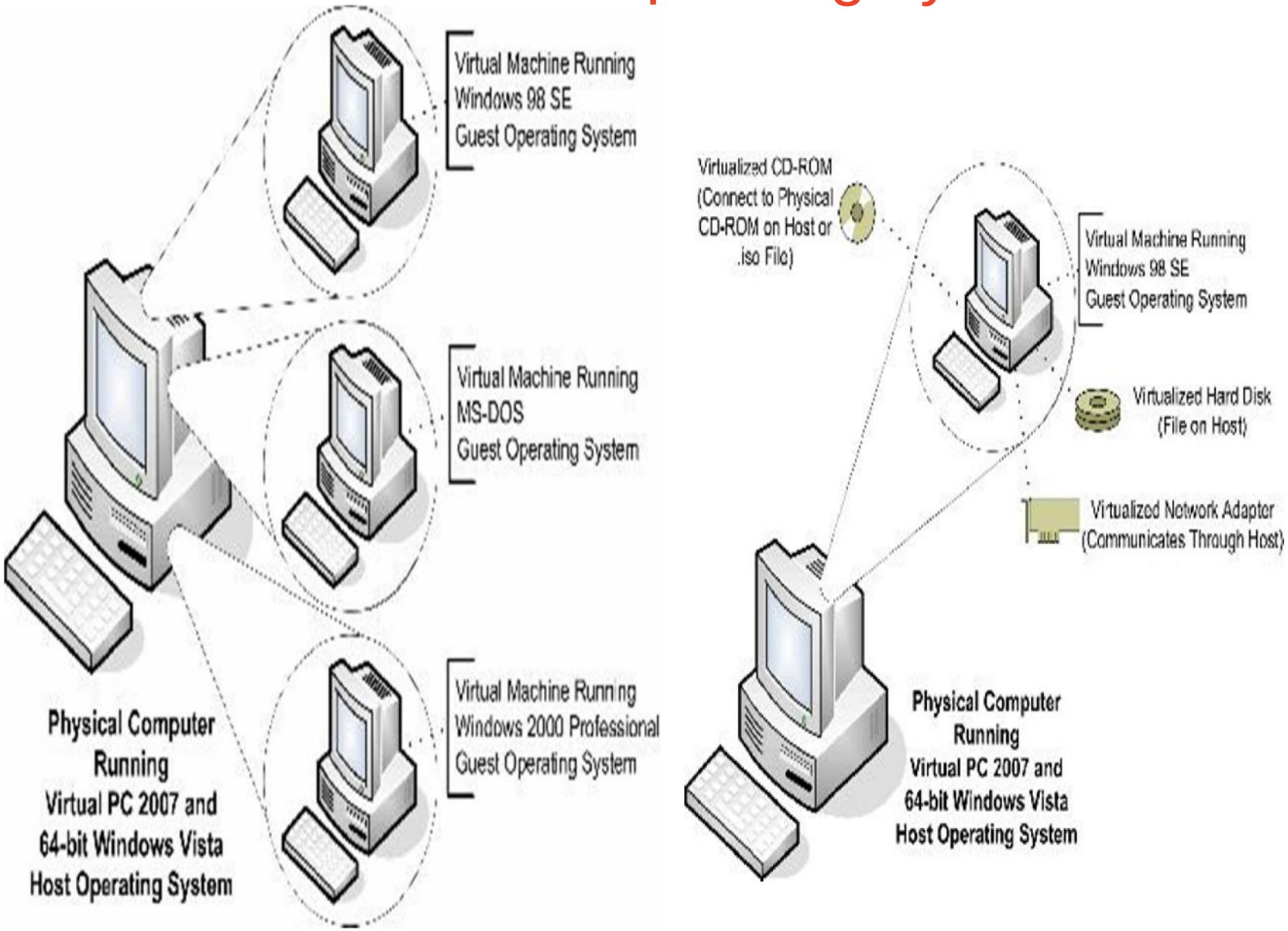
Perbandingan Model Sistem



Non-virtual Machine

Virtual Machine

Windows Vista Host Operating System



Keuntungan & Kerugian VM (1)

- Memberikan proteksi terhadap system resources karena setiap VM terisolasi dari yang lainnya
- Isolasi ini tidak memungkinkan *direct sharing* dari resources
- Merupakan alat *research & development* dalam OS

Keuntungan & Kerugian VM (2)

- Pengembangan sistem dilakukan pada VM sehingga tidak menganggu OS
- Konsep VM sulit diimplementasi akibat perlunya menyediakan duplikat yang persis dari mesin dibawahnya (*underlaying machine*)
  - Harus punya *virtual-user mode* dan *virtual-monitor mode* yang keduanya berjalan di-*physical mode*. Akibatnya, saat instruksi yang hanya membutuhkan *virtual monitor mode* dijalankan, register berubah dan bisa berefek pada *virtual user mode*, bahkan bisa me-restart VM
- Waktu yang dibutuhkan I/O bisa lebih cepat (karena ada *spooling*), tapi bisa lebih lambat (karena di-interpreted)

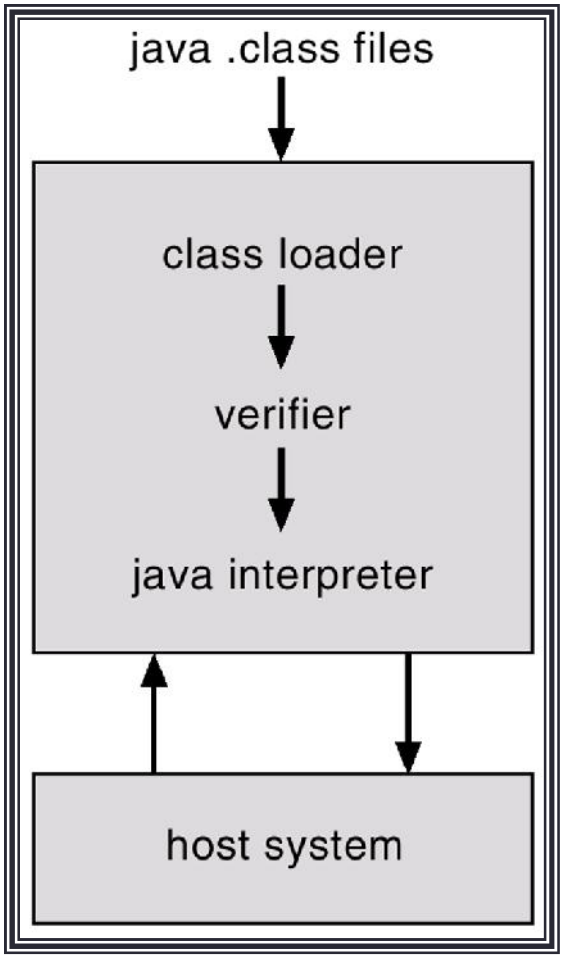
Java Virtual Machine (1)

- Program Java yang telah di-compile adalah platform-neutral bytecodes yang dieksekusi oleh Java Virtual Machine(JVM)
- JVM terdiridari :
  - class loader
  - class verifier
  - runtime interpreter
- Just In-Time(JIT) kompilator meningkatkan kinerja

Java Virtual Machine (2)

- Java Development Environment (JDE) terdiri dari sebuah *compile time environment* yang mengubah *java sources code* menjadi *bytecode*, dan sebuah *run time environment* yang menyediakan *Java platform system*

Java Virtual Machine (3)



VI. Perancangan Sistem

- Masalah : menentukan tujuan dan spesifikasi sistem.
  - Perancangan sistem dipengaruhi oleh perangkat keras dan jenis sistem sehingga kebutuhan-nya akan lebih sulit untuk dispesifikasikan.
- User goals
  - OS harus nyaman untuk digunakan, mudah dipelajari, reliable, aman dan cepat
- System goals
  - OS harus mudah dirancang, diimplementasikan dan di-maintain, serta fleksibel, reliable, error-free dan efisien

Mekanisme & Kebijakan

- *Mekanisme* menjelaskan bagaimana melakukan sesuatu, *kebijakan* menentukan apa yang akan dilakukan
- Pemisahan kebijakan dari mekanisme adalah hal yang sangat penting, untuk memungkinkan fleksibilitas yang tinggi jika kebijakan akan diubah suatu saat.
- Kebijakan penting untuk semua alokasi sumber daya dan menjadualkan masalah, menentukan perlu atau tidaknya mengalokasikan sumber daya.
- Mekanisme yang menentukan apa dan

Implementasi Sistem

- Secara tradisional OS ditulis dalam bahasa assembly, tapi sekarang OS dapat ditulis dalam bahasa pemrograman tingkat tinggi (HLL)
- Keuntungan penulisan dengan HLL :
  - Dapat ditulis lebih cepat
  - Lebih padat (*compact*)
  - Mudah dipahami & di-debug
  - Lebih portabel : mudah dipindahkan ke perangkat keras lain

System Generation (SYSGEN)

- OS dirancang untuk run pada berbagai kelas mesin, harus dikonfigurasikan untuk setiap spesifikasi komputer
- Program SYSGEN memperoleh informasi berkaitan dengan konfigurasi spesifik suatu sistem HW, antara lain :
  - CPU apa yang digunakan, pilihan yang diinstal
  - Berapa banyak memori yang tersedia
  - Peralatan yang tersedia
  - Sistem operasi pilihan apa yang diinginkan atau parameter apa yang digunakan

System Boot

- Booting – memulai komputer dengan me-load kernel
- Bootstrap program – code yang disimpan dalam ROM (Firmware) yang mencari kernel dan me-loadnya ke memori serta memulai eksekusinya