

# Konsep Dasar Sistem Operasi

# Konsep Dasar Sistem Operasi

- Komponen Sistem Operasi
- Layanan Sistem Operasi
- *System Calls*
- Pemrograman Sistem
- Struktur sistem
- Mesin Virtual
- *System Generation*
- Rancangan Sistem

# Komponen Sistem

- ❖ Manajemen Proses
- ❖ Manajemen Memori Utama
- ❖ Manajemen Berkas
- ❖ Manajemen I/O
- ❖ Manajemen Penyimpanan Sekunder
- ❖ Jaringan
- ❖ Sistem Proteksi
- ❖ *Command-Interpreter System*

# Managemen Proses (1)

❖ **Proses** adalah sebuah program yang sedang dijalankan (eksekusi).

Suatu proses memerlukan sumber daya pada saat ekesekusi:  
→ CPU time, memori, berkas dan peranti I/O

# Managemen Proses (2)

- ❖ Sistem operasi bertanggung jawab terhadap aktifitas yang berhubungan dengan manajemen proses:
  - Pembuatan dan penghapusan proses
  - Penundaan dan pelanjutan proses
  - Penyedia mekanisme untuk:
    - Sinkronisasi antar proses
    - Komunikasi antar proses
    - Penanganan *Deadlock*

# Managemen Memori Utama (1)

- ❖ Memori sebagai tempat penyimpanan instruksi/data dari program.
  - Penyimpanan yang cepat sehingga dapat mengimbangi kecepatan eksekusi instruksi CPU
  - Terdiri dari “array words/bytes” yang besar
  - Alamat digunakan untuk mengakses data (shared oleh CPU dan I/O devices)

# Managemen Memori Utama (2)

- ❖ Umumnya main memory bersifat “*volatile*” – tidak permanen
- ❖ Isinya akan hilang jika komputer di matikan.
- ❖ Sistem operasi bertanggung jawab untuk aktivitas berikut yang berhubungan dengan manajemen memori:
  - melacak pemakaian memori (siapa dan berapa besar?).
  - memilih program mana yang akan di load ke memori ketika bisa digunakan.
  - alokasi dan dealokasi memori sesuai yang dibutuhkan

# Managemen File (1)

- ❖ Berkas adalah kumpulan informasi yang berhubungan (sesuai dengan tujuan pembuat berkas tersebut). Biasanya berkas merepresentasikan program dan data.
  
- ❖ Sistem operasi bertanggung jawab untuk aktivitas berikut yang berhubungan dengan manajemen berkas:
  - pembuatan dan penghapusan berkas
  - pembuatan dan penghapusan direktori
  - Mendukung primitif untuk manipulasi berkas dan direktori
  - memetakan berkas pada sistem sekunder
  - *Backup* berkas pada media penyimpanan yang stabil (*nonvolatile*)

# Managemen sistem I/O

❖ Sistem I/O terdiri dari :

- Sistem *buffer* : menampung sementara data dari/ke peranti I/O
- *Spooling*: melakukan penjadwalan pemakaian I/O sistem supaya lebih efisien (antrian dsb)
- Antarmuka *devices-driver* yang umum :  
menyediakan *device driver* yang umum sehingga sistem operasi dapat seragam (buka, baca, tulis, tutup)
- *Drivers* untuk spesifik perangkat keras :  
menyediakan *driver* untuk melakukan operasi rinci/detail untuk perangkat keras tertentu.

# Manajemen Penyimpanan Sekunder

- ❖ Penyimpanan sekunder : Penyimpanan Permanen
  - Karena memori utama bersifat sementara dan kapasitasnya terlalu kecil, maka untuk menyimpan semua data dan program secara permanen, sistem komputer harus menyediakan penyimpanan sekunder untuk dijadikan *back-up* memori utama.
- ❖ Sistem Operasi bertanggung jawab dalam aktivitas yang berhubungan dengan manajemen penyimpanan sekunder :
  - manajemen ruang kosong
  - alokasi penyimpanan
  - penjadwalan disk

# Jaringan (Sistem Terdistribusi)

- ❖ Sistem Terdistribusi adalah kumpulan prosesor yang tidak berbagi memori atau *clock*. Setiap prosesor memiliki memori lokal masing-masing.
- ❖ Prosesor-prosesor dalam sistem terhubung dalam jaringan komunikasi.
- ❖ Sistem terdistribusi menyediakan akses pengguna ke bermacam-macam sumber daya. Akses tersebut menyebabkan :
  - Peningkatan kecepatan komputasi
  - peningkatan penyediaan data
  - peningkatan keandalan

# Sistem Proteksi

- ❖ Proteksi berkenaan dengan mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor, pengguna sistem maupun pengguna sumber daya.
- ❖ Mekanisme Proteksi harus :
  - membedakan antara penggunaan yang sah dan yang tidak sah.
  - spesifikasi kontrol untuk di terima
  - menyediakan alat untuk pemberlakuan sistem.

# Command-Interpreter System (1)

- ❖ Sistem Operasi menunggu instruksi dari pengguna (*command driven*).
- ❖ Program yang membaca instruksi dan mengartikan *control statements* (keinginan pengguna) umumnya disebut:
  - *control-card interpreter*
  - *command-line interpreter*
  - *UNIX shell*.

# Command-Interpreter System (2)

- ❖ *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)*, dan lain-lain.

# Pelayanan Sistem Operasi(1)

- ❖ **Eksekusi program** : *meload* program ke memory dan menjalankannya (*run*)
- ❖ **Operasi I/O** : pengguna tidak bisa mengontrol I/O secara langsung ( untuk efisiensi dan keamanan), sistem harus bisa menyediakan mekanisme utk melakukan operasi I/O
- ❖ **Manipulasi sistem berkas**: membaca, menulis, membuat, dan menghapus file

# Pelayanan Sistem Operasi (2)

- ❖ **Komunikasi** : pertukaran informasi, dapat dilaksanakan melalui *shared memory* atau *message passing*
- ❖ **Deteksi error** : mempertahankan kestabilan dengan mendeteksi error (pada CPU, perangkat keras memori, I/O, program pengguna) dan jika bisa, memperbaikinya

# Pelayanan Tambahan

- ❖ Lebih diarahkan kepada upaya untuk menjaga efisiensi sistem, bukan untuk membantu pengguna
- ❖ **Alokasi sumber daya** : mengalokasikan sumber daya kepada beberapa pengguna atau tugas yang dijalankan pada saat yang bersamaan
- ❖ *Accounting* : menentukan berapa banyak dan berapa lama users menggunakan sumber daya sistem
- ❖ **Proteksi** : menjaga semua akses ke sumber daya sistem terkontrol

# System calls (3)

- ❖ *System calls* menyediakan antarmuka antara proses (program yang sedang dijalankan) dan sistem operasi.
- ❖ Biasanya tersedia sebagai instruksi bahasa rakitan
- ❖ Beberapa sistem mengizinkan *system calls* dibuat langsung dari bahasa pemrograman tingkat tinggi

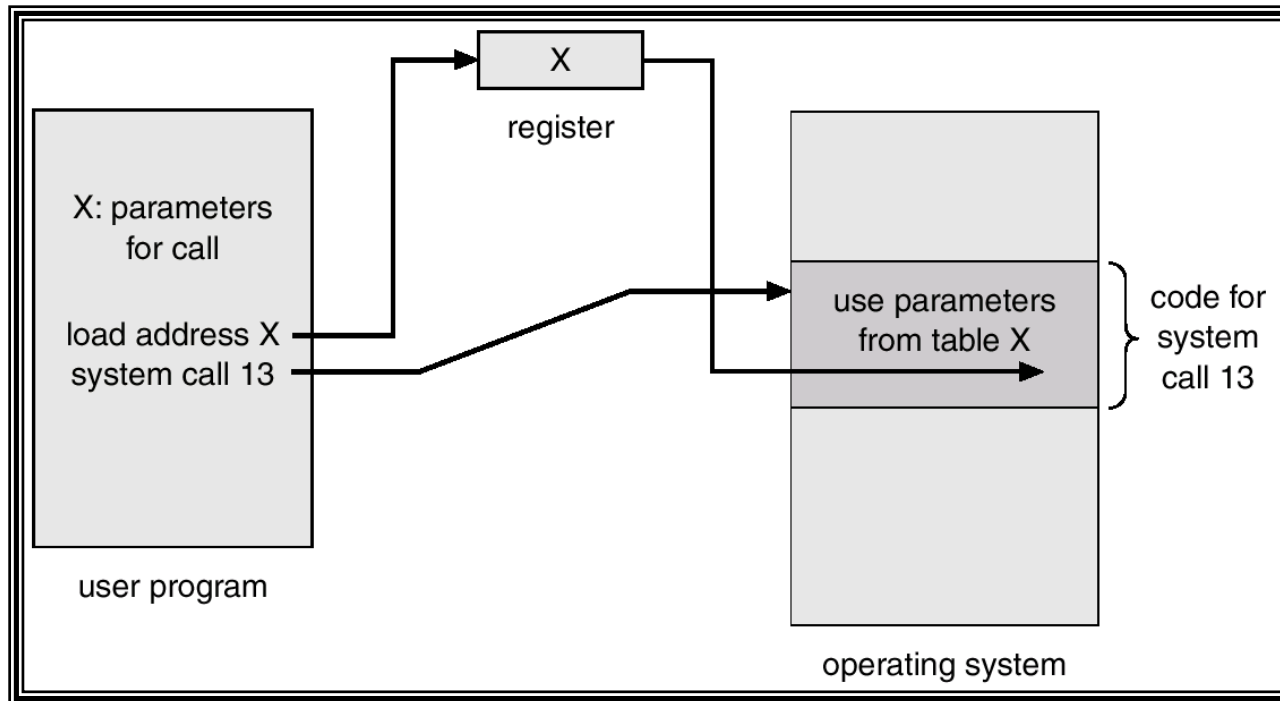
# System Calls (2)

- ❖ Beberapa bahasa pemrograman tingkat tinggi (contoh : C, C++) telah didefinisikan untuk menggantikan bahasa rakitan untuk sistem pemrograman

# System Calls (3)

- ❖ Tiga metode umum yang digunakan dalam memberikan parameter kepada sistem operasi
  - Melalui register
  - Menyimpan parameter dalam blok atau tabel pada memori dan alamat blok tersebut diberikan sebagai parameter dalam register
  - Menyimpan parameter (*push*) ke dalam *stack* (oleh program), dan melakukan *pop off* pada *stack* (oleh sistem operasi)

# Memberikan Parameter dalam Tabel



Sumber: Silberschatz, et.al, Operating System Concepts, 6<sup>th</sup> ed, .2003, New York: John Wiley & Son.Inc , page 65

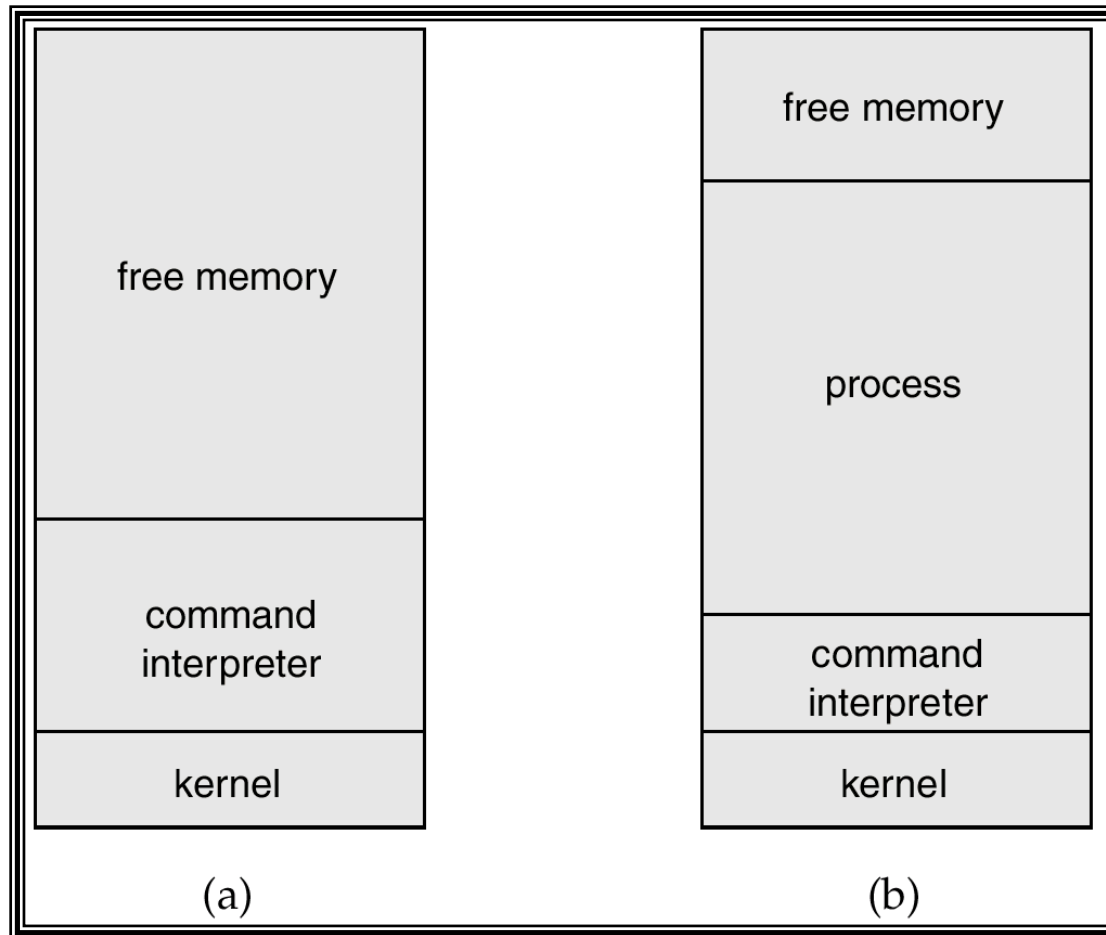
# Jenis *System Calls*

- ❖ **Pengendalian proses**
- ❖ **Manajemen berkas**
- ❖ **Manajemen Peranti**
- ❖ **Mempertahankan informasi**
- ❖ **Komunikasi**

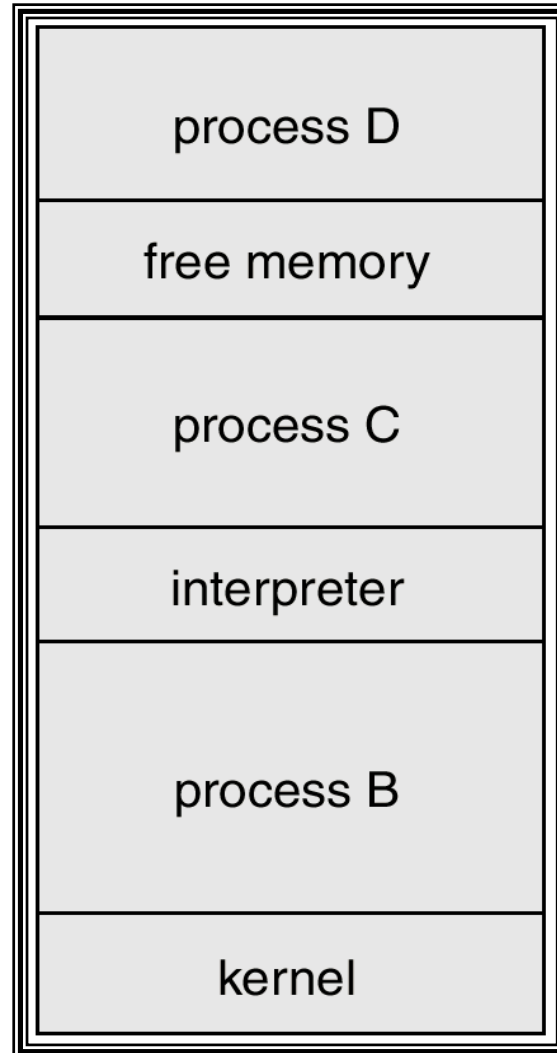
# Process Control

- ❖ 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



# UNIX Menjalankan *Multiple Program*



# Manajemen Berkas

- ❖ Membuat dan menghapus berkas
- ❖ Membuka dan menutup berkas
- ❖ *Read, write, reposition*
- ❖ Mengambil dan mengeset atribut berkas

# Manajemen Peranti

- ❖ Meminta peranti, melepaskan peranti
- ❖ *Read, write, reposition*
- ❖ Mengambil dan mengeset atribut peranti

# *Information Maintenance*

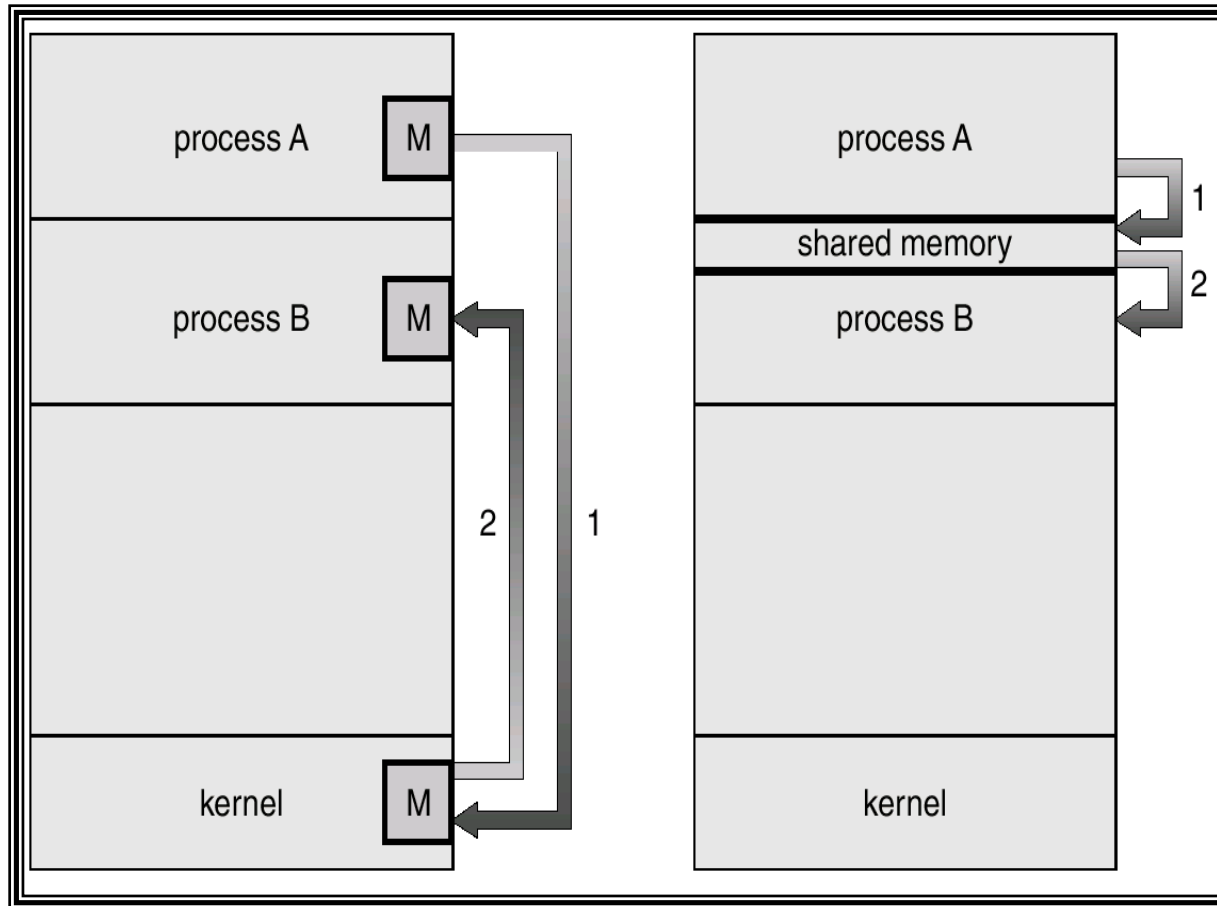
- ❖ Mengambil dan mengeset waktu dan tanggal
- ❖ Mengambil dan mengeset sistem data
- ❖ mengambil proses, berkas atau atribut peranti
- ❖ Mengeset proses, berkas atau atribut peranti

# Komunikasi

- ❖ Menciptakan, menghapus hubungan komunikasi
- ❖ Mengirim dan menerima pesan
- ❖ Mentransfer status informasi
- ❖ *Attach* atau *detach remote device*

Komunikasi dapat dilakukan melalui *message passing* atau *shared memory*

# Mekanisme Komunikasi



# Pemrograman sistem (1)

- ❖ Pemrograman sistem menyediakan lingkungan yang memungkinkan pengembangan program dan eksekusi berjalan dengan baik
- ❖ Dapat dikategorikan :
  - Manajemen/manipulasi berkas** : membuat, menghapus, *copy, rename, print*, memanipulasi berkas dan direktori

# Pemrograman Sistem (2)

- ❖ **Informasi Status** : tanggal, jam, jumlah memori dan *disk* yang tersedia, jumlah pengguna, dan informasi tentang status lainnya
- ❖ **Modifikasi Berkas** : modifikasi isi berkas
- ❖ **Mendukung bahasa pemrograman** : Kompilator, perakit, *interpreter*

# Pemrograman Sistem (3)

- ❖ **Loading dan eksekusi program** : *absolute loaders, relocatable loaders, linkage editors, overlay loaders*
- ❖ **Komunikasi** : menyediakan mekanisme komunikasi antara proses, pengguna, dan sistem komputer yang berbeda

System program yang paling penting adalah *command interpreter* (mengambil dan menerjemahkan *user-specified command* selanjutnya)

# Struktur Sistem

- ❖ Struktur Sederhana
- ❖ Metode pendekatan Terlapis
- ❖ 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 Sistem UNIX

- ❖ Terdiri dari 2 bagian:
  - Kernel :
    - antarmuka
    - *device drivers*
  
  - Program Sistem

# Pendekatan Terlapis

- ❖ 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

# Tingkatan Desain Sistem Operasi

| Level | nama                       | objek   |
|-------|----------------------------|---|
| 13    | shell                      | user programming environment                            |
| 12    | proses pengguna            | proses pengguna   |
| 11    | direktori                  | direktori   |
| 10    | peranti                    | peranti eksternal                                       |
| 9     | sistem berkas              | berkas  |
| 8     | komunikasi                 | pipa  |
| 7     | memori virtual             | segmen, halaman   |
| 6     | penyimpanan sekunder lokal | blok data, saluran peranti                              |
| 5     | proses primitif            | proses primitif, semafor, ready list                    |
| 4     | interupsi                  | program penanganan interupsi                            |
| 3     | prosedur                   | prosedur, call-stack, tampilan                          |
| 2     | set instruksi              | stack, microprogram interpreter, scalar, dan array data |
| 1     | sirkuit elektronik         | register, gerbang, bus, dll                             |

# Pendekatan Terlapis

## ❖ Keuntungan : **modularitas**

- mempermudah *debug* dan verifikasi sistem
- lapisan pertama bisa *didebug* tanpa mengganggu sistem yang lain

## ❖ Kesulitan :

- hanya bisa menggunakan lapisan dibawahnya
- tidak efisien dibandingkan tipe yang lain

# 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 pada suatu desain perangkat keras ke desain lainnya
- mendukung keamanan reliabilitas lebih

## ❖ Contoh sistem operasi :

Tru64 UNIX, MacOSX, QNX

# Mesin Virtual (1)

- ❖ Mesin virtual mengambil pendekatan terlapis sebagai kesimpulan logis. Mesin virtual memperlakukan hardware dan sistem operasi seolah-olah berada pada level yang sama sebagai perangkat keras.
- ❖ Pendekatan Mesin virtual menyediakan sebuah antarmuka yang identik dengan *underlying bare hardware*.
- ❖ Sistem Operasi membuat ilusi dari banyak proses, masing-masing dieksekusi pada prosesoranya sendiri dengan virtual memorinya sendiri.
- ❖ VM dibuat dengan pembagian sumber daya oleh komputer fisik

# Mesin virtual (2)

- ❖ Sumber daya dari komputer fisik dibagi untuk membuat VM
  - Penjadwalan CPU bisa menciptakan penampilan seakan – akan pengguna mempunyai prosesor sendiri
  - *Spooling* dan sistem data bisa menyediakan *virtual card readers* dan *virtual line printers*
  - Sebuah *time-sharing terminal user* yang normal melayani sebagaimana operator konsulat
- ❖ VM software membutuhkan ruang di dalam disk untuk menyediakan memori virtual dan *spooling*, yaitu sebuah disk virtual

# Keuntungan Penggunaan Mesin virtual

- ❖ Keamanan bukanlah masalah
  - VM mempunyai perlindungan lengkap pada berbagai sistem sumber daya
  - Tidak ada pembagian sumber daya secara langsung. Pembagian disk mini dan jaringan diimplementasikan pada perangkat lunak
- ❖ VM sistem adalah kendaraan yang “sempurna” untuk penelitian dan pengembangan sistem operasi
  - Dengan VM perubahan suatu bagian tidak akan mempengaruhi komponen yang lain

# Kerugian Penggunaan VM

- ❖ VM sulit diimplementasikan karena banyak syarat yang dibutuhkan untuk menyediakan duplikat yang tepat dari *underlying 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 diinterpreted)

# *Java Virtual Machine (1)*

- ❖ Program Java yang telah *dcompile* adalah *platform-neutral bytecodes* yang dieksekusi oleh Java Virtual Machine(JVM)
- ❖ JVM terdiri dari:
  - pengeloaad kelas
  - pemverifikasi kelas
  - *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 sourcecode* menjadi *bytecode*, dan sebuah *run time environment* yang menyediakan *Java platform system* untuk tuan rumah

# 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.
- ❖ Kebutuhan terdiri dari tujuan pengguna dan tujuan sistem.
- ❖ Pengguna ingin sistem yang enak digunakan, mudah dipelajari, terpercaya, aman, dan cepat. Tapi itu semua sebenarnya tidak dibutuhkan oleh sebuah sistem.
- ❖ Sistem ingin mudah dirancang dan diimplmentasikan, fleksibel, terpercaya, error yang minimal, dan efisien.

# Mekanisme dan Kebijakan

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

# Implementasi Sistem (1)

- ❖ Secara tradisional, sistem operasi ditulis dalam bahasa rakitan, tapi sekarang sering dibuat dalam bahasa tingkat tinggi.
- ❖ Keuntungan ditulis dalam bahasa tingkat tinggi adalah
  - kodenya bisa ditulis dengan lebih cepat
  - lebih padat
  - mudah dimengerti dan *didebug*

# Implementasi Sistem (2)

- ❖ Sistem operasi yang ditulis dengan bahasa tingkat tinggi akan mudah dipindahkan ke perangkat keras lain, tapi bisa mengurangi kecepatan dan membutuhkan penyimpanan yang lebih banyak.

# *System Generation*

- ❖ Sistem operasi dirancang untuk dapat dijalankan pada berbagai jenis mesin, sistemnya harus dikonfigurasi untuk setiap komputer.
  
- ❖ Program *Sysgen* mendapatkan informasi mengenai konfigurasi khusus tentang sistem perangkat keras dari sebuah data, antara lain sebagai berikut:
  - 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
  
- ❖ Satu kali info diperoleh, bisa digunakan dengan berbagai cara