Bab 1

# Pendahuluan

Bab ini berisi tiga pokok pembahasan. Pertama, membahas hal-hal umum seputar sistem operasi.Selanjutnya, menerangkan konsep perangkat keras sebuah komputer. Sebagai penutup akan diungkapkan, pokok konsep dari sebuah sistem operasi.

## Tujuan Pelajaran

Setelah mempelajari bab ini, Anda diharapkan :

- Memahami fungsi dasar Sistem Operasi.
- Mengetahui sejarah Sistem Operasi.
- Mengetahui dan memahami struktur suatu Sistem Komputer, meliputi Sistem Operasi Komputer, Struktur I/O, Struktur Penyimpanan, *Storage Hierarchy*, dan Proteksi Perangkat Keras.
- Mengetahui dan memahami struktur Sistem Operasi, meliputi Managemen Proses, Managemen Memori Utama, Managemen *Secondary Storage*, Managemen Sistem I/O, Managemen Berkas, Sistem Proteksi, Jaringan dan Command-Interpreter System.
- Memahami layanan apa saja yang disediakan Sistem Operasi.
- Memahami konsep *System Calls*,
- Memahami konsep Struktur Sistem Operasi
- Memahami Perancangan dan Implementasi Sistem
- Memahmi *System Generation*.

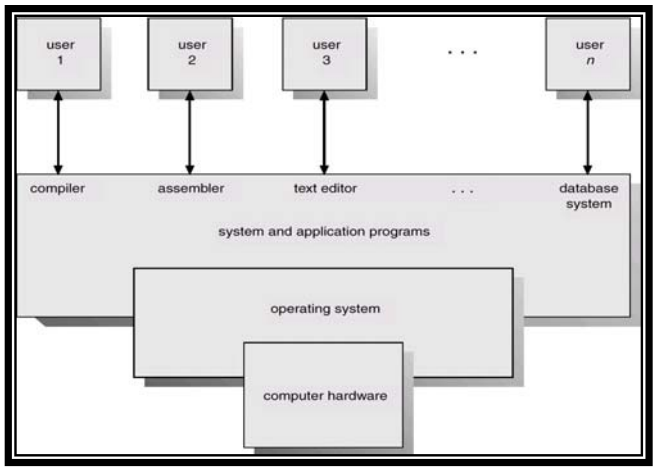
# 1.1. Sistem Operasi

Sistem operasi merupakan sebuah penghubung antara pengguna dari komputer dengan perangkat keras komputer. Sebelum ada sistem operasi, orang hanya mengunakan komputer dengan menggunakan sinyal analog dan sinyal digital. Seiring dengan berkembangnya pengetahuan dan teknologi, pada saat ini terdapat berbagai sistem operasi dengan keunggulan masing-masing. Untuk lebih memahami sistem operasi maka sebaiknya perlu diketahui terlebih dahulu beberapa konsep dasar mengenai sistem operasi itu sendiri.

Pengertian sistem operasi secara umum ialah pengelola seluruh sumber-daya yang terdapat pada sistem komputer dan menyediakan sekumpulan layanan (*system calls*) ke pemakai sehingga memudahkan dan menyamankan penggunaan serta pemanfaatan sumber-daya sistem komputer.

## 1.1.1. Fungsi Dasar

Sistem komputer pada dasarnya terdiri dari empat komponen utama, yaitu perangkat-keras, program aplikasi, sistem-operasi, dan para pengguna. Sistem operasi berfungsi untuk mengatur dan mengawasi penggunaan perangkat keras oleh berbagai program aplikasi serta para pengguna.




**Gambar 1-1 Komponen Sistem**

Sistem operasi berfungsi ibarat pemerintah dalam suatu negara, dalam arti membuat kondisi komputer agar dapat menjalankan program secara benar. Untuk menghindari konflik yang terjadi pada saat pengguna menggunakan sumber-daya yang sama, sistem operasi mengatur pengguna mana yang dapat mengakses suatu sumber-daya. Sistem operasi juga sering disebut *resource allocator*. Satu lagi fungsi penting sistem operasi ialah sebagai program pengendali yang bertujuan untuk menghindari kekeliruan (*error*) dan penggunaan komputer yang tidak perlu.

## 1.1.2. Tujuan Mempelajari Sistem Operasi

Tujuan mempelajari sistem operasi agar dapat merancang sendiri serta dapat memodifikasi sistem yang telah ada sesuai dengan kebutuhan kita, agar dapat memilih alternatif sistem operasi, memaksimalkan penggunaan sistem operasi dan agar konsep dan teknik sistem operasi dapat diterapkan pada aplikasi-aplikasi lain.

## 1.1.3. Sasaran Sistem Operasi

Sistem operasi mempunyai tiga sasaran utama yaitu *kenyamanan* -- membuat penggunaan komputer menjadi lebih nyaman, *efisien* -- penggunaan sumber-daya sistem komputer secara efisien, serta mampu *berevolusi* -- sistem operasi harus dibangun sehingga memungkinkan dan memudahkan pengembangan, pengujian serta pengajuan sistem-sistem yang baru.

## 1.1.4. Sejarah Sistem Operasi

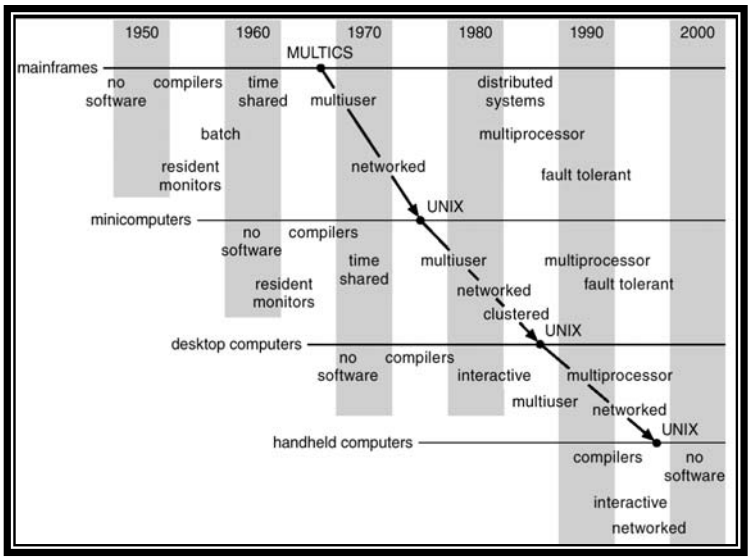
Menurut Tanenbaum, sistem operasi mengalami perkembangan yang sangat pesat, yang dapat dibagi kedalam empat generasi:

- Generasi Pertama (1945-1955)
  Generasi pertama merupakan awal perkembangan sistem komputasi elektronik sebagai pengganti sistem komputasi mekanik, hal itu disebabkan kecepatan manusia untuk menghitung terbatas dan manusia sangat mudah untuk membuat kecerobohan, kekeliruan bahkan kesalahan. Pada generasi ini belum ada sistem operasi, maka sistem komputer diberi instruksi yang harus dikerjakan secara langsung.

- Generasi Kedua (1955-1965)
  Generasi kedua memperkenalkan *Batch Processing System*, yaitu Job yang dikerjakan dalam satu rangkaian, lalu dieksekusi secara berurutan.Pada generasi ini sistem komputer belum dilengkapi sistem operasi, tetapi beberapa fungsi sistem operasi telah ada, contohnya fungsi sistem operasi ialah FMS dan IBSYS.

- Generasi Ketiga (1965-1980)
  Pada generasi ini perkembangan sistem operasi dikembangkan untuk melayani banyak pemakai sekaligus, dimana para pemakai interaktif berkomunikasi lewat terminal secara on-line ke komputer, maka sistem operasi menjadi *multi-user* (di gunakan banyak pengguna sekaligus) dan *multi-programming* (melayani banyak program sekali gus).

- Generasi Keempat (Pasca 1980an)
  Dewasa ini, sistem operasi dipergunakan untuk jaringan komputer dimana pemakai menyadari keberadaan komputer-komputer yang saling terhubung satu sama lainnya. Pada masa ini para pengguna juga telah dinyamankan dengan *Graphical User Interface* yaitu antar-muka komputer yang berbasis grafis yang sangat nyaman, pada masa ini juga dimulai era komputasi tersebar dimana komputasi-komputasi tidak lagi berpusat di satu titik, tetapi dipecah dibanyak komputer sehingga tercapai kinerja yang lebih baik.




**Gambar 1-2. Migrasi Sistem Operasi vs. Sistem Komputer**

## 1.1.5. Layanan Sistem Operasi

Sebuah sistem operasi yang baik menurut Tanenbaum harus memiliki layanan sebagai berikut: pembuatan program, eksekusi program, pengaksesan *I/O Device*, pengaksesan terkendali terhadap berkas pengaksesan sistem, deteksi dan pemberian tanggapan pada kesalahan, serta akunting.

Pembuatan program yaitu sistem operasi menyediakan fasilitas dan layanan untuk membantu para pemrogram untuk menulis program; Eksekusi Program yang berarti Instruksi-instruksi dan data-data harus dimuat ke memori utama, perangkat-parangkat masukan/ keluaran dan berkas harus di-inisialisasi, serta sumber-daya yang ada harus disiapkan, semua itu harus di tangani oleh sistem operasi; Pengaksesan *I/O Device*, artinya Sistem Operasi harus mengambil alih sejumlah instruksi yang rumit dan sinyal kendali menjengkelkan agar pemrogram dapat berfikir sederhana dan perangkat pun dapat beroperasi; Pengaksesan terkendali terhadap berkas yang artinya disediakannya mekanisme proteksi terhadap berkas untuk mengendalikan pengaksesan terhadap berkas; Pengaksesan sistem artinya pada pengaksesan digunakan bersama (*shared system*); Fungsi pengaksesan harus menyediakan proteksi terhadap sejumlah sumber-daya dan data dari pemakai tak terdistorsi serta menyelesaikan konflik-konflik dalam perebutan sumber-daya; Deteksi dan Pemberian tanggapan pada kesalahan, yaitu jika muncul permasalahan muncul pada sistem komputer maka sistem operasi harus memberikan tanggapan yang menjelaskan kesalahan yang terjadi serta dampaknya terhadap aplikasi yang sedang berjalan; dan Akunting yang artinya Sistem Operasi yang bagus mengumpulkan data statistik penggunaan beragam sumber-daya dan memonitor parameter kinerja.

# 1.2. Struktur Komputer

Struktur sebuah sistem komputer dapat dibagi menjadi:

- Operasi Sistem Komputer
- Struktur I/O.

- Struktur Penyimpanan.
- *Storage Hierarchy*.
- Proteksi Perangkat Keras.

## 1.2.1. Operasi Sistem Komputer

Dewasa ini sistem komputer multiguna terdiri dari CPU (*Central Processing Unit*); serta sejumlah *device controller* yang dihubungkan melalui *bus* yang menyediakan akses ke memori. Setiap *device controller* bertugas mengatur perangkat yang tertentu (contohnya *disk drive*, *audio device*, dan *video display*). CPU dan *device controller* dapat dijalankan secara bersamaan, namun demikian diperlukan mekanisme sinkronisasi untuk mengatur akses ke memori.

Pada saat pertama kali dijalankan atau pada saat *boot*, terdapat sebuah program awal yang mesti dijalankan. Program awal ini disebut program *bootstrap*. Program ini berisi semua aspek dari sistem komputer, mulai dari register CPU, *device controller*, sampai isi memori.

Interupsi merupakan bagian penting dari sistem arsitektur komputer. Setiap sistem komputer memiliki mekanisme yang berbeda. Interupsi bisa terjadi apabila perangkat keras (*hardware*) atau perangkat lunak (*software*) minta "dilayani" oleh prosesor. Apabila terjadi interupsi maka prosesor menghentikan proses yang sedang dikerjakannya, kemudian beralih mengerjakan *service routine* untuk melayani interupsi tersebut. Setelah selesai mengerjakan *service routine* maka prosesor kembali melanjutkan proses yang tertunda.

## 1.2.2. Struktur Input/Output

Bagian ini akan membahas struktur I/O, interupsi I/O, dan DMA, serta perbedaan dalam penanganan interupsi.

### 1.2.2.1. Interupsi I/O

Untuk memulai operasi I/O, CPU me-*load* register yang bersesuaian ke *device controller*. Sebaliknya *device controller* memeriksa isi register untuk kemudian menentukan operasi apa yang harus dilakukan.

Pada saat operasi I/O dijalankan ada dua kemungkinan, yaitu *synchronous I/O* dan *asynchronous I/O*. Pada *synchronous I/O*, kendali dikembalikan ke proses pengguna setelah proses I/O selesai dikerjakan. Sedangkan pada *asynchronous I/O*, kendali dikembalikan ke proses pengguna tanpa menunggu proses I/O selesai. Sehingga proses I/O dan proses pengguna dapat dijalankan secara bersamaan.

### 1.2.2.2. Struktur DMA

*Direct Memory Access* (DMA) suatu metoda penanganan I/O dimana *device controller* langsung berhubungan dengan memori tanpa campur tangan CPU. Setelah men-set *buffers*, *pointers*, dan *counters* untuk perangkat I/O, *device controller* mentransfer blok data langsung ke penyimpanan tanpa campur tangan CPU. DMA digunakan untuk perangkat I/O dengan kecepatan tinggi. Hanya terdapat satu interupsi setiap blok, berbeda dengan perangkat yang mempunyai kecepatan rendah dimana interupsi terjadi untuk setiap *byte* (*word*).

## 1.2.3. Struktur Penyimpanan

Program komputer harus berada di memori utama (biasanya RAM) untuk dapat dijalankan. Memori utama adalah satu-satunya tempat penyimpanan yang dapat diakses secara langsung oleh prosesor. Idealnya program dan data secara keseluruhan dapat disimpan dalam memori utama secara permanen. Namun demikian hal ini tidak mungkin karena:

- Ukuran memori utama relatif kecil untuk dapat menyimpan data dan program secara keseluruhan.
- Memori utama bersifat *volatile*, tidak bisa menyimpan secara permanen, apabila komputer dimatikan maka data yang tersimpan di memori utama akan hilang.

### 1.2.3.1. Memori Utama

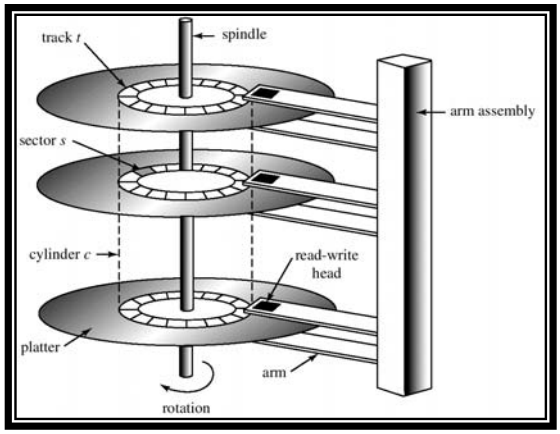
Hanya memori utama dan register merupakan tempat penyimpanan yang dapat diakses secara langsung oleh prosesor. Oleh karena itu instruksi dan data yang akan dieksekusi harus disimpan di memori utama atau register.

Untuk mempermudah akses perangkat I/O ke memori, pada arsitektur komputer menyediakan fasilitas pemetaan memori ke I/O. Dalam hal ini sejumlah alamat di memori dipetakan dengan *device register*. Membaca dan menulis pada alamat memori ini menyebabkan data ditransfer dari dan ke *device register*. Metode ini cocok untuk perangkat dengan waktu respon yang cepat seperti *video controller*.

Register yang terdapat dalam prosesor dapat diakses dalam waktu 1 *clock cycle*. Hal ini menyebabkan register merupakan media penyimpanan dengan akses paling cepat bandingkan dengan memori utama yang membutuhkan waktu relatif lama. Untuk mengatasi perbedaan kecepatan, dibuatlah suatu penyangga (*buffer*) penyimpanan yang disebut *cache*.

### 1.2.3.2. *Magnetic Disk*

*Magnetic Disk* berperan sebagai *secondary storage* pada sistem komputer modern. *Magnetic Disk* disusun dari piringan-piringan seperti CD. Kedua permukaan piringan diselimuti oleh bahan-bahan magnetik. Permukaan dari piringan dibagi-bagi menjadi *track* yang memutar, yang kemudian dibagi lagi menjadi beberapa sektor.

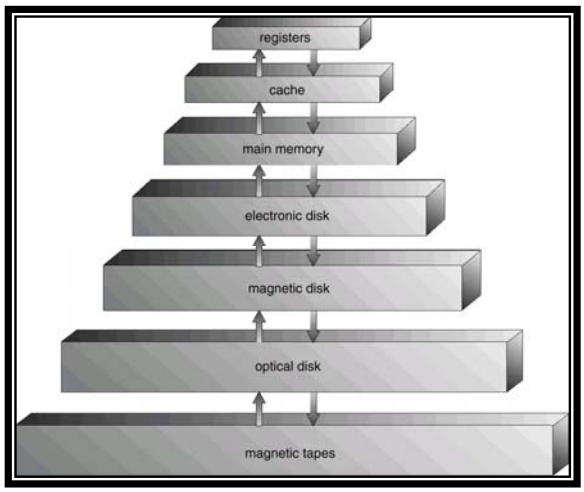



**Gambar 1-3. Mekanisme Magnetic Disk**

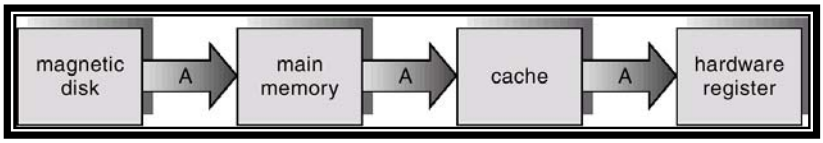
## 1.2.4. Hirarki *Storage*

Dalam *storage hierarchy structure*, data yang sama bisa tampil dalam level berbeda dari sistem penyimpanan. Sebagai contoh integer A berlokasi pada bekas B yang ditambahkan 1, dengan asumsi bekas B terletak pada *magnetic disk*. Operasi penambahan diproses dengan pertama kali mengeluarkan operasi I/O untuk menduplikat disk block pada A yang terletak pada memori utama Operasi ini diikuti dengan kemungkinan penduplikatan A ke dalam *cache* dan penduplikatan A ke dalam internal register. Sehingga penduplikatan A terjadi di beberapa tempat. Pertama terjadi di internal register dimana nilai A berbeda dengan yang di sistem penyimpanan. Dan nilai di A akan kembali sama ketika nilai baru ditulis ulang ke *magnetic disk*.

Pada kondisi multi prosesor, situasi akan menjadi lebih rumit. Hal ini disebabkan masing-masing prosesor mempunyai *local cache*. Dalam kondisi seperti ini hasil duplikat dari A mungkin hanya ada di beberapa *cache*. Karena CPU (register-register) dapat dijalankan secara bersamaan maka kita harus memastikan perubahan nilai A pada satu *cache* akan mengubah nilai A pada semua *cache* yang ada. Hal ini disebut sebagai *Cache Coherency*.




**Gambar 1-4. Hirarki *Storage***




**Gambar 1-5. Migrasi dari Disk ke Register**

## 1.2.5. Proteksi Perangkat Keras

Sistem komputer terdahulu berjenis *programmer-operated systems*. Ketika komputer dioperasikan dalam konsul mereka (pengguna) harus melengkapi sistem terlebih dahulu. Akan tetapi setelah sistem operasi lahir maka hal tersebut diambil alih oleh sistem operasi. Sebagai contoh pada monitor yang proses I/O sudah diambil alih oleh sistem operasi, padahal dahulu hal ini dilakukan oleh pengguna.

Untuk meningkatkan utilisasi sistem, sistem operasi akan membagi sistem sumber daya sepanjang program secara simultan. Pengertian *spooling* adalah suatu program dapat dikerjakan walau pun I/O masih mengerjakan proses lainnya dan disk secara

bersamaan menggunakan data untuk banyak proses. Pengertian *multi programming* adalah kegiatan menjalankan beberapa program pada memori pada satu waktu.

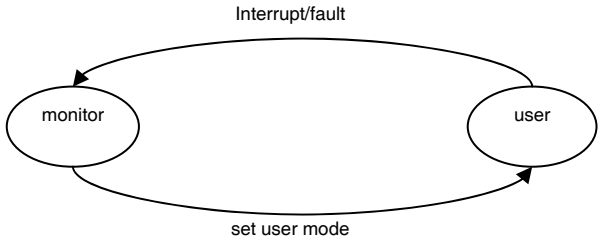
Pembagian ini memang menguntungkan sebab banyak proses dapat berjalan pada satu waktu akan tetapi mengakibatkan masalah-masalah baru. Ketika tidak di *sharing* maka jika terjadi kesalahan hanyalah akan membuat kesalahan program. Tapi jika di-*sharing* jika terjadi kesalahan pada satu proses/ program akan berpengaruh pada proses lainnya.

Sehingga diperlukan pelindung (proteksi). Tanpa proteksi jika terjadi kesalahan maka hanya satu saja program yang dapat dijalankan atau seluruh output pasti diragukan.
Banyak kesalahan pemprograman dideteksi oleh perangkat keras. Kesalahan ini biasanya ditangani oleh sistem operasi. Jika terjadi kesalahan program, perangkat keras akan meneruskan kepada sistem operasi dan sistem operasi akan menginterupsi dan mengakhirinya. Pesan kesalahan disampaikan, dan memori dari program akan dibuang. Tapi memori yang terbuang biasanya tersimpan pada disk agar *programmer*
bisa membetulkan kesalahan dan menjalankan program ulang.

### 1.2.5.1. Operasi Dual Mode

Untuk memastikan operasi berjalan baik kita harus melindungi sistem operasi, program, dan data dari program-program yang salah. Proteksi ini memerlukan *share resources*. Hal ini bisa dilakukan sistem operasi dengan cara menyediakan pendukung perangkat keras yang mengizinkan kita membedakan mode pengeksekusian program.
Mode yang kita butuhkan ada dua mode operasi yaitu:

- Mode Monitor.
- Mode Pengguna.




**Gambar 1-6. Operasi Dual Mode**

Pada perangkat keras akan ada bit atau Bit Mode yang berguna untuk membedakan mode apa yang sedang digunakan dan apa yang sedang dikerjakan. Jika Mode Monitor maka akan benilai 0, dan jika Mode Pengguna maka akan bernilai 1.

Pada saat *boot time*, perangkat keras bekerja pada mode monitor dan setelah sistem operasi di-*load* maka akan mulai masuk ke mode pengguna. Ketika terjadi *trap* atau interupsi, perangkat keras akan men-*switch* lagi keadaan dari mode pengguna menjadi mode monitor (terjadi perubahan *state* menjadi bit 0). Dan akan kembali menjadi mode pengguna jikalau sistem operasi mengambil alih proses dan kontrol komputer (*state* akan berubah menjadi bit 1).

### 1.2.5.2. Proteksi I/O

Pengguna bisa mengacaukan sistem operasi dengan melakukan instruksi I/O ilegal dengan mengakses lokasi memori untuk sistem operasi atau dengan cara hendak melepaskan diri dari prosesor. Untuk mencegahnya kita menganggap semua instruksi

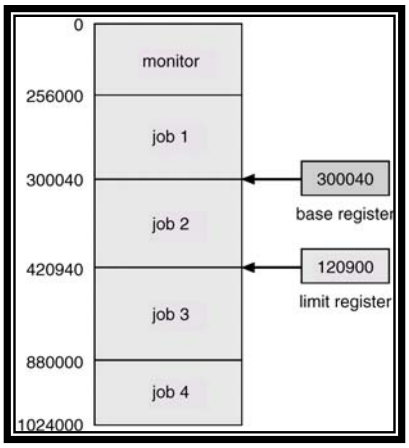
I/O sebagai *privilidge instruction* sehingga mereka tidak bisa mengerjakan instruksi I/O secara langsung ke memori tapi harus lewat sistem operasi terlebih dahulu. Proteksi I/O dikatakan selesai jika pengguna dapat dipastikan tidak akan menyentuh mode monitor. Jika hal ini terjadi proteksi I/O dapat dikompromikan.

### 1.2.5.3. Proteksi Memori

Salah satu proteksi perangkat keras ialah dengan proteksi memori yaitu dengan pembatasan penggunaan memori. Disini diperlukan beberapa istilah yaitu:

- Base Register yaitu alamat memori fisik awal yang dialokasikan/ boleh digunakan oleh pengguna.
- Limit Register yaitu nilai batas dari alamat memori fisik awal yang dialokasikan/boleh digunakan oleh pengguna
- Proteksi Perangkat Keras.

Sebagai contoh sebuah pengguna dibatasi mempunyai base register 300040 dan mempunyai limit register 120900 maka pengguna hanya diperbolehkan menggunakan alamat memori fisik antara 300040 hingga 420940 saja.




**Gambar 1-7. Penggunaan Base dan Limit Register**

# 1.3. Struktur Sistem Operasi

## 1.3.1. Komponen-komponen Sistem

Pada kenyataannya tidak semua sistem operasi mempunyai struktur yang sama. Namun menurut Avi Silberschatz, Peter Galvin, dan Greg Gagne, umumnya sebuah sistem operasi modern mempunyai komponen sebagai berikut:

- Managemen Proses.
- Managemen Memori Utama.
- Managemen *Secondary-Storage*.
- Managemen Sistem I/O.
- Managemen Berkas.
- Sistem Proteksi.
- Jaringan.
- *Command-Interpreter system*.

## 1.3.2. Managemen Proses

Proses adalah keadaan ketika sebuah program sedang di eksekusi. Sebuah proses membutuhkan beberapa sumber daya untuk menyelesaikan tugasnya. sumber daya tersebut dapat berupa *CPU time*, memori, berkas-berkas, dan perangkat-perangkat I/O.
Sistem operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan managemen proses seperti:

- Pembuatan dan penghapusan proses pengguna dan sistem proses.
- Menunda atau melanjutkan proses.
- Menyediakan mekanisme untuk proses sinkronisasi.
- Menyediakan mekanisme untuk proses komunikasi.
- Menyediakan mekanisme untuk penanganan *deadlock*.

## 1.3.3. Managemen Memori Utama

Memori utama atau lebih dikenal sebagai memori adalah sebuah *array* yang besar dari *word* atau *byte*, yang ukurannya mencapai ratusan, ribuan, atau bahkan jutaan. Setiap *word* atau *byte* mempunyai alamat tersendiri. Memori Utama berfungsi sebagai tempat penyimpanan yang akses datanya digunakan oleh CPU atau perangkat I/O. Memori utama termasuk tempat penyimpanan data yang sementara (*volatile*), artinya data dapat hilang begitu sistem dimatikan. Sistem operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan managemen memori seperti:

- Menjaga *track* dari memori yang sedang digunakan dan siapa yang menggunakannya.
- Memilih program yang akan di-*load* ke memori.
- Mengalokasikan dan meng-dealokasikan ruang memori sesuai kebutuhan.

## 1.3.4. Managemen *Secondary-Storage*

Data yang disimpan dalam memori utama bersifat sementara dan jumlahnya sangat kecil. Oleh karena itu, untuk meyimpan keseluruhan data dan program komputer dibutuhkan *secondary-storage* yang bersifat permanen dan mampu menampung banyak data. Contoh dari *secondary-storage* adalah *harddisk*, disket, dll.

Sistem operasi bertanggung-jawab atas aktivitas-aktivitas yang berkaitan dengan *disk-management*seperti: *free-space management*, alokasi penyimpanan, penjadualan disk.

## 1.3.5. Managemen Sistem I/O

Sering disebut *device manager*. Menyediakan "*device driver*" yang umum sehingga operasi I/O dapat seragam (membuka, membaca, menulis, menutup). Contoh: pengguna menggunakan operasi yang sama untuk membaca berkas pada *hard-disk*, CD-ROM dan *floppy disk*.

Komponen Sistem Operasi untuk sistem I/O:

- *Buffer*: menampung sementara data dari/ ke perangkat I/O.
- *Spooling*: melakukan penjadualan pemakaian I/O sistem supaya lebih efisien (antrian dsb.).
- Menyediakan *driver* untuk dapat melakukan operasi "rinci" untuk perangkat keras I/O tertentu.

## 1.3.6. Managemen Berkas

Berkas adalah kumpulan informasi yang berhubungan sesuai dengan tujuan pembuat berkas tersebut. Berkas dapat mempunyai struktur yang bersifat hirarkis (direktori, volume, dll.).

Sistem operasi bertanggung-jawab:

- Pembuatan dan penghapusan berkas.
- Pembuatan dan penghapusan direktori.
- Mendukung manipulasi berkas dan direktori.
- Memetakan berkas ke *secondary storage*.
- Mem-*backup* berkas ke media penyimpanan yang permanen (*non-volatile*).

## 1.3.7. Sistem Proteksi

Proteksi mengacu pada mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor, atau pengguna ke sistem sumber daya. Mekanisme proteksi harus:

- membedakan antara penggunaan yang sudah diberi izin dan yang belum.
- *specify the controls to be imposed*.
- *provide a means of enforcement*.

## 1.3.8. Jaringan

Sistem terdistribusi adalah sekumpulan prosesor yang tidak berbagi memori atau *clock*. Tiap prosesor mempunyai memori sendiri. Prosesor-prosesor tersebut terhubung melalui jaringan komunikasi Sistem terdistribusi menyediakan akses pengguna ke bermacam sumber-daya sistem. Akses tersebut menyebabkan:

- *Computation speed-up*.
- *Increased data availability*.
- *Enhanced reliability*.

## 1.3.9. *Command-Interpreter System*

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

## 1.3.10. Layanan Sistem Operasi

Eksekusi program adalah kemampuan sistem untuk "*load*" program ke memori dan menjalankan program. Operasi I/O: pengguna tidak dapat secara langsung mengakses sumber daya perangkat keras, sistem operasi harus menyediakan mekanisme untuk melakukan operasi I/O atas nama pengguna. Sistem manipulasi berkas dalah kemampuan program untuk operasi pada berkas (membaca, menulis, membuat, dan menghapus berkas). Komunikasi adalah pertukaran data/ informasi antar dua atau lebih proses yang berada pada satu komputer (atau lebih). Deteksi *error* adalah menjaga kestabilan sistem dengan mendeteksi "*error*", perangkat keras mau pun operasi.

Efesisensi penggunaan sistem:

- *Resource allocator* adalah mengalokasikan sumber-daya ke beberapa pengguna atau *job* yang jalan pada saat yang bersamaan.
- Proteksi menjamin akses ke sistem sumber daya dikendalikan (pengguna dikontrol aksesnya ke sistem).
- *Accounting* adalah merekam kegiatan pengguna, jatah pemakaian sumber daya (keadilan atau kebijaksanaan).

## 1.3.11. *System Call*s

*System call* menyediakan interface antara program (program pengguna yang berjalan) dan bagian OS.a*System call* menjadi jembatan antara proses dan sistem operasi. *System call* ditulis dalam bahasaa*assembly* atau bahasa tingkat tinggi yang dapat mengendalikan mesin (C). Contoh: UNIX menyediakan *system call*: *read, write* => operasi I/O untuk berkas.

Sering pengguna program harus memberikan data (parameter) ke OS yang akan dipanggil. Contoh pada UNIX: read(buffer, max_size, file_id);

Tiga cara memberikan parameter dari program ke sistem operasi:

- Melalui registers (sumber daya di CPU).
- Menyimpan parameter pada data struktur (table) di memori, dan alamat table tsb ditunjuk oleh *pointer* yang disimpan di register.
- *Push* (*store*) melalui "*stack*" pada memori dan OS mengambilnya melalui *pop* pada *stack* tsb.

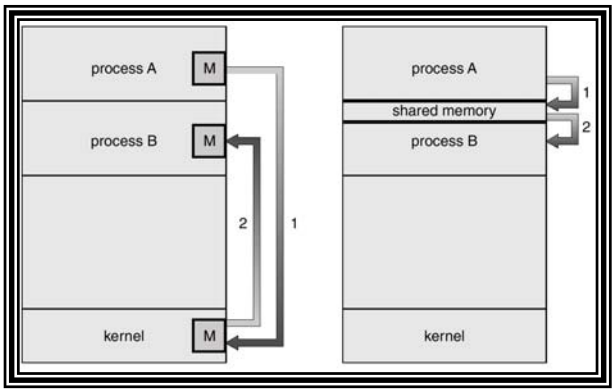
Pada dasarnya *system calls* dapat dikelompokkan dalam 5 kategori sebagai berikut :

- Kontrol Proses
  Hal-hal yang dilakukan :
  - Mengakhiri (end) dan membatalkan (abort)
  - Mengambil (load dan mengeksekusi (execute)
  - Membuat dan mengakhiri proses
  - Menentukan dan mengeset atribut proses
  - *Wait for time*
  - *Wait event, signal event*
  - Mengalokasikan dan membebaskan memori
- Manipulasi File
  Hal-hal yang dilakukan :
  - Membuat dan menghapus file
  - Membuka dan menutupfile
  - Membaca, menulis dan mereposisi file
  - Menentukan dan mengeset atribut file
- Manajemen Device
  Hal-hal yang dilakukan :
  - Meminta dan membebaskan device
  - Membaca, menulis dan mereposisi device
  - Menentukan dan mengeset atribut device
- Informasi Lingkungan
  Hal-hal yang dilakukan
  - Mengambil atau mengeset waktu atau tanggal
  - Mengambil atau mengeset sistem data
  - Mengambil atau mengeset proses, file atau atribut-atribut device
- Komunikasi
  Hal-hal yang dilakukan :
  - Membuat dan menghapus sambungan komunikasi

- Mengirim dan menerima pesan
- Mentransfer status informasi

Ada 2 model komunikasi

a. *Message-passing model*. Informasi saling ditukarkan melalui fasilitas yang telah ditentukan oleh sistem operasi
b. *Shared-memory model.* Proses-proses menggunakan map memory untuk mengakses daerah-daerah dimemori dengan proses-proses yang lain.




Message Passing Shared Memory

**Gambar 1-8. Komunikasi**

## 1.3.12. Program Sistem

Program Sistem adalah masalah yang realatif kompleks, namun dapat dibagi menjadi beberapa kategori, antara lain :

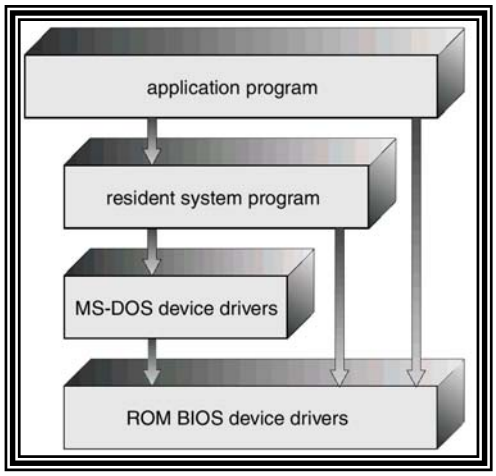
- Manipulasi File. Meliputi : membuat, menghapus, mencopy, rename, print, dump, list pada file dan direktory
- Status Informasi. Meliputi : tanggal, waktu (jam, menit, detik), penggunaan memori atau disk space, banyaknya user
- Modifikasi File. Ada beberapa editor yang sanggup digunakan sebagai sarana untuk menulis atau memodifikasi file yang tersimpan dalam disk atau tape
- Bahasa Pemrograman yang mendukung. Meliputi : Compiler, Assembler, dan interpreter untuk beberapa bahasa pemrograman (seperti : Fortran, Cobol, Pascal, Basic dan LISP)
- Pemanggilan dan Eksekusi Program. Pada saat program dicompile, maka harus dipanggil ke memori untuk dieksekusi. Suatu sistem biasanya memiliki *absolute loader*, mengalokasikan *loader, linkage editor* dan *overlay loader*. Juga dibutuhkan *debugging* sistem untuk bahasa tingkat tinggi.
- Komunikasi. Sebagai mekanisme untuk membuat hubungan virtual antar proses, user dan sistem komputer yang berbeda
- Program-program aplikasi. Sistem operasi harus menyokong program-program yang berguna untuk menyelesaikan permasalahan secara umum atau membentuk operasi-operasi secara umum, seperti kompiler, pemformat teks, paket plot, sistem basis data, spreadsheet, paket analisis statistik dan games.

## 1.3.13. Struktur Sistem Operasi

Sistem komputer modern yang semakin kompleks dan rumit memerlukan sistem operasi yang dirancang dengan sangat hati-hati agar dapat berfungsi secara optimum dan mudah untuk dimodifikasi.

- Struktur Sederhana

  Ada sejumlah sistem komersial yang tidak memiliki struktur yang cuup baik. Sistem operasi tersebut sangat kecil, sederhana dan memiliki banyak ketebatasan. Salah satu contoh sistem tesebut adalah MS DOS dirancang oleh orang-orang yang tidak memikirkan akan kepopuleran software tersebut. Sistem operasi tersebut terbatas pada hardware sehingga tidak terbagi terbagi menjadi modul-modul seperti terlihat pada di bawah. Karena Intel 8088 tidak menggunakan dual mode sehingga tidak ada proteksi harware.




**Gambar 1-9. Struktur Layer MS-DOS**

- Sistem Monolitic

  Pada dasarnya, sistem monolitic merupakan struktur sederhana yang dilengkapi dengan operasi dual mode. Pelayanan (*system calls*) yang diberikan oleh sistem operasi model ini dilakukan dengan cara mengambil sejumlah parameter pada tempat yang telah ditentukan sebelumnya, seperti register atau stack, dan kemudian mengeksekusi suatu instruksi trap tertentu pada monitor mode.

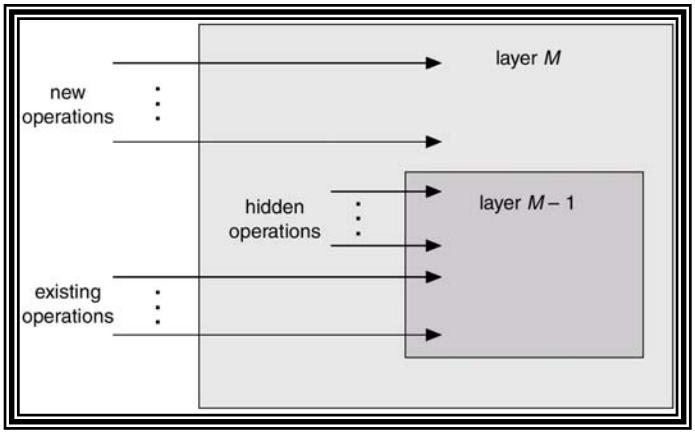
  Secara umum system calls dibuat dengan cara :

  - user program melakukan 'trap' pada kernel. Instruksi berpindah dari *user mode* ke *monitor mode* dan mentransfer kontrol ke sistem operasi
  - sistem operasi mengecek parameter-parameter dari pemanggilan tersebut untuk menentukan *system call* mana yang memanggil
  - sistem operasi menunjuk ke suatu tabel yang berisi slot ke –k yang menunjukkan *system call* k
  - setelah *system call* selesai mengerjakan tugasnya, konrol akan dikembalikan pada user program.

- Pendekatan Berlapis (*Layered Approach*)

  Teknik pendekatan berlapis pada dasarnya dibuat dengan cara membentuk sistem operasi menjadi bentuk modular. Dengan menggunakan pendekatan *top-down*, semua fungsi ditentukan dan dibagi menjadi komponen-komponen. Modularisasi sistem dilakukan denga cara memecah sistem operasi menajdi

beberapa lapis (tingkat). Lapisan terendah (lapis-0) adalah hardware dan lapisan teratas (lapisan N) adalah *user interface*.



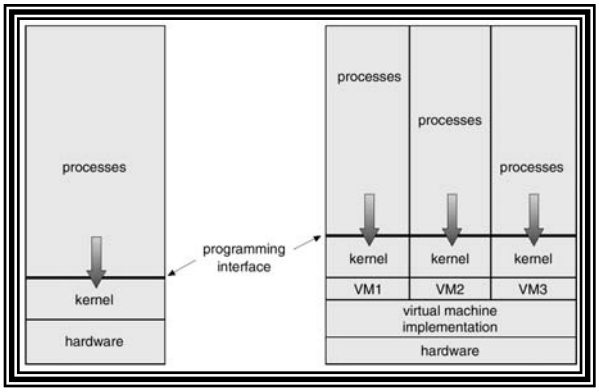

**Gambar 1-10. Lapisan Sistem Operasi**

- Mesin Virtual

Sebuah mesin virtual (*Virtual Machine*) menggunakan misalkan terdapat sistem program => control program yang mengatur pemakaian sumber daya perangkat keras. Control program = trap *System call* + akses ke perangkat keras. Control program memberikan fasilitas ke proses pengguna. Mendapatkan jatah CPU dan memori. Menyediakan *interface* "identik" dengan apa yang disediakan oleh perangkat keras => *sharing devices* untuk berbagai proses.

Mesin Virtual (MV) (MV) => control program yang minimal MV memberikan ilusi *multitasking*: seolah-olah terdapat prosesor dan memori ekslusif digunakan MV. MV memilah fungsi *multitasking* dan implementasi *extended machine* (tergantung proses pengguna) => flexible dan lebih mudah untuk pengaturan. Jika setiap pengguna diberikan satu MV => bebas untuk menjalankan OS (kernel) yang diinginkan pada MV tersebut. Potensi lebih dari satu OS dalam satu komputer. Contoh: IBM VM370: menyediakan MV untuk berbagai OS: CMS (interaktif), MVS, CICS, dll. Masalah: Sharing disk => OS mempunyai sistem berkas yang mungkin berbeda. IBM: virtual disk (minidisk) yang dialokasikan untuk pengguna melalui MV.

Konsep MV menyediakan proteksi yang lengkap untuk sumberdaya sistem, dikarenakan tiap MV terpisah dari MV yang lain. Namun, hal tersebut menyebabkan tidak adanya *sharing* sumberdaya secara langsung. MV merupakan alat yang tepat untuk penelitian dan pengembangan sistem operasi. Konsep MV susah untuk diimplementasi sehubungan dengan usaha yang diperlukan untuk menyediakan duplikasi dari mesin utama.




**Gambar 1-11. Model Mesin Virtual**

- Model *Client Server*
  Trend dari sistem operasi modern adalah memindahkan kode ke lapisan yang lebih tinggi dan menghapusnya sebanyak mungkin dari sistem operasi sehingga akan meninggalkan kernel yang minimal. Konsep ini biasanya diimplementasikan dengan cara menjadikannya fungsi-fungsi yang ada pada sistem operasi menjadi user proses. Jika suatu proses minta untuk dilayani, misalkan saja satu blok file, maka user proses (*client server*) mengirim permintaan tersebut ke server proses. Server proses akan melayani permintaan tersebut ke server proses. Server proses akan mealyani permintaan tersebut kemudian mengirimkan jawabannya kembali. Pada model ini, semua pekerjaan kernel ditekankan pada pengendalian komunikasi antara client dan server.

### 1.3.14. Perancangan Sistem dan Implementasi

Target untuk pengguna: sistem operasi harus nyaman digunakan, mudah dipelajari, dapat diandalkan, aman dan cepat. Target untuk sistem: sistem operasi harus gampang dirancang, diimplementasi, dan dipelihara, sebagaimana fleksibel, *error*, dan efisien.

Mekanisme dan Kebijaksanaan:

- Mekanisme menjelaskan bagaimana melakukan sesuatu kebijaksanaan memutuskan apa yang akan
- dilakukan. Pemisahan kebijaksanaan dari mekanisme merupakan hal yang sangat penting; ini mengizinkan fleksibilitas yang tinggi bila kebijaksanaan akan diubah nanti.
- Kebijaksanaan memutuskan apa yang akan dilakukan.

Pemisahan kebijaksanaan dari mekanisme merupakan hal yang sangat penting; ini mengizinkanfleksibilitas yang tinggi bila kebijaksanaan akan diubah nanti.

Implementasi Sistem biasanya menggunakan bahas *assembly*, sistem operasi sekarang dapat ditulis dengan menggunakan bahasa tingkat tinggi. Kode yang ditulis dalam bahasa tingkat tinggi: dapat dibuat dengan cepat, lebih ringkas, lebih mudah dimengerti dan didebug. Sistem operasi lebih mudah dipindahkan ke perangkat keras yang lain bila ditulis dengan bahasa tingkat tinggi.

### 1.3.14. *System Generation* (SYSGEN)

Sistem operasi dirancang untuk dapat dijalankan di berbagai jenis mesin; sistemnya harus di konfigurasi untuk tiap komputer. Program SYSGEN mendapatkan informasi mengenai konfigurasi khusus dari sistem perangkat keras.

- *Booting*: memulai komputer dengan me-*load* kernel.
- *Bootstrap program*: kode yang disimpan di code ROM yang dapat menempatkan kernel, memasukkannya kedalam memori, dan memulai eksekusinya.

## 1.4. Rangkuman

Sistem operasi telah berkembang selama lebih dari 40 tahun dengan dua tujuan utama. Pertama, sistem operasi mencoba mengatur aktivitas-aktivitas komputasi untuk memastikan pendayagunaan yang baik dari sistem komputasi tersebut. Kedua, menyediakan lingkungan yang nyaman untuk pengembangan dan jalankan dari program.

Pada awalnya, sistem komputer digunakan dari depan konsul. Perangkat lunak seperti *assembler*, *loader*, *linker*dan *compiler* meningkatkan kenyamanan dari sistem

pemrograman, tapi juga memerlukan waktu *set-up* yang banyak. Untuk mengurangi waktu *set-up* tersebut, digunakan jasa operator dan menggabungkan tugas-tugas yang sama (sistem *batch*). Sistem *batch* mengizinkan pengurutan tugas secara otomatis dengan menggunakan sistem operasi yang resident dan memberikan peningkatan yang cukup besar dalam utilisasi komputer. Komputer tidak perlu lagi menunggu operasi oleh pengguna. Tapi utilisasi CPU tetap saja rendah. Hal ini dikarenakan lambatnya kecepatan alat-alat untuk I/O relatif terhadap kecepatan CPU. Operasi *off-line* dari alat-alat yang lambat bertujuan untuk menggunakan beberapa sistem *reader-to-tape* dan *tape-to-printer* untuk satu CPU.

Untuk meningkatkan keseluruhan kemampuan dari sistem komputer, para developer memperkenalkan konsep ***multiprogramming***. Dengan *multiprogramming*, beberapa tugas disimpan dalam memori dalam satu waktu; CPU digunakan secara bergantian sehingga menambah utilisasi CPU dan mengurangi total waktu yang dibutuhkan untuk menyelesaikan tugas-tugas tersebut. *Multiprogramming*, yang dibuat untuk meningkatkan kemampuan, juga mengizinkan ***time sharing***. Sistem operasi yang bersifat *time-shared* memperbolehkan banyak pengguna untuk menggunakan komputer secara interaktif pada saat yang bersamaan. **Komputer Personal** adalah mikrokomputer yang dianggap lebih kecil dan lebih murah dibandingkan komputer *mainframe*. Sistem operasi untuk komputer-komputer seperti ini diuntungkan oleh pengembangan sistem operasi untuk komputer *mainframe* dalam beberapa hal. Namun, semenjak penggunaan komputer untuk keperluan pribadi, maka utilisasi CPU tidak lagi menjadi perhatian utama. Karena itu, beberapa desain untuk komputer *mainframe* tidak cocok untuk sistem yang lebih kecil.

**Sistem parallel** mempunyai lebih dari satu CPU yang mempunyai hubungan yang erat; CPU-CPU tersebut berbagi bus komputer, dan kadang-kadang berbagi memori dan perangkat yang lainnya. Sistem seperti itu dapat meningkatkan *throughput* dan reliabilititas. **Sistem *hard real-time*** sering kali digunakan sebagai alat pengontrol untuk applikasi yang dedicated. Sistem operasi yang ***hard real-time*** mempunyai batasan waktu yang tetap yang sudah didefinisikan dengan baik.Pemrosesan harus selesai dalam batasan-batasan yang sudah didefinisikan, atau sistem akan gagal. **Sistem *soft real-time*** mempunyai lebih sedikit batasan waktu yang keras, dan tidak mendukung penjadwalan denganmenggunakan batas akhir. Pengaruh dari internet dan *World Wide Web*baru-baru ini telah mendorong pengembangan sistem operasi modern yang menyertakan *web browser* serta perangkat lunak jaringan dan komunikasi sebagai satu kesatuan.

*Multiprogramming* dan *sistem time-sharing* meningkatkan kemampuan komputer dengan melampaui batas operasi (*overlap*) CPU dan I/O dalam satu mesin. Hal seperti itu memerlukan perpindahan dataantara CPU dan alat I/O, ditangani baik dengan polling atau *interrupt-driven* akses ke *I/O port*, atau dengan perpindahan DMA. Agar komputer dapat menjalankan suatu program, maka program tersebut harus berada di memori utama (memori utama). **Memori utama adalah** satu-satunya tempat penyimpanan yang besar yang dapat diakses secara langsung oleh prosessor, merupakan suatu *array* dari *word* atau *byte*, yang mempunyai ukuran ratusan sampai jutaan ribu. Setiap *word* memiliki alamatnya sendiri. Memori utama adalah tempat penyimpanan yang *volatile*, dimana isinya hilang bila sumber energinya (energi listrik) dimatikan. Kebanyakan sistem komputer menyediakan ***secondary storage*** sebagai perluasan dari memori utama. Syarat utama dari *secondary storage* adalah dapat menyimpan data dalam jumlah besar secara permanen. *Secondary storage* yang paling umum adalah disk magnetik, yang meyediakan penyimpanan untuk program mau pun data. **Disk magnetik** adalah alat penyimpanan data yang *nonvolatile* yang juga menyediakan akses secara random. **Tape magnetik** digunakan terutama untuk *backup*, penyimpanan informasi yang jarang digunakan, dan sebagai media pemindahan informasi dari satu sistem ke sistem yang lain.

Beragam sistem penyimpanan dalam sistem komputer dapat disusun dalam hirarki berdasarkan kecepatan dan biayanya. Tingkat yang paling atas adalah yang paling mahal, tapi cepat. Semakin kebawah, biaya perbit menurun, sedangkan waktu aksesnya semakin bertambah (semakin lambat).

Sistem operasi harus memastikan operasi yang benar dari sistem komputer. Untuk mencegah pengguna program mengganggu operasi yang berjalan dalam sistem, perangkat keras mempunyai dua mode: mode pengguna dan mode monitor. Beberapa perintah (seperti perintah I/O dan perintah halt) adalah perintah khusus, dan hanya dapat dijalankan dalam mode monitor. Memori juga harus dilindungi dari modifikasi oleh pengguna. *Timer* mencegah terjadinya pengulangan secara terus menerus (*infinite loop*). Hal-hal tersebut (dual mode, perintah khusus, pengaman memori, *timer interrupt*) adalah blok bangunan dasar yang digunakan oleh sistem operasi untuk mencapai operasi yang sesuai. Sistem operasi menyediakan banyak pelayanan. Di tingkat terrendah, ***sistem calls*** mengizinkan program yang sedang berjalan untuk membuat permintaan secara langsung dari sistem operasi. Di tingkat tertinggi, *command interpreter* atau *shell* menyediakan mekanisme agar pengguna dapat membuat permintaan tanpa menulis program. *Command* dapat muncul dari bekas sewaktu jalankan *batch-mode*, atau secara langsung dari terminal ketika dalam mode interaktive atau *time-shared*. Program sistem disediakan untuk memenuhi kebanyakan dari permintaan pengguna. Tipe dari permintaan beragam sesuai dengan levelnya. Level *sistem call* harus menyediakan fungsi dasar, seperti kontrol proses serta manipulasi alat dan bekas. Permintaan dengan level yang lebih tinggi (*command interpreter* atau program sistem) diterjemahkan kedalam urutan *sistem call*.

Pelayanan sistem dapat dikelompokkan kedalam beberapa kategori: kontrol program, status permintaan dan permintaan I/O. Program *error* dapat dipertimbangkan sebagai permintaan yang implisit untuk pelayanan. Bila sistem pelayanan sudah terdefinisi, maka struktur dari sistem operasi dapat dikembangkan. Berbagai macam tabel diperlukan untuk menyimpan informasi yang mendefinisikan status dari sistem komputer dan status dari sistem tugas. Perancangan dari suatu sistem operasi yang baru merupakan tugas yang utama. Sangat penting bahwa tujuan dari sistem sudah terdefinisi dengan baik sebelum memulai perancangan. Tipe dari sistem yang diinginkan adalah landasan dalam memilih beragam *algoritma* dan strategi yang akan digunakan. Karena besarnya sistem operasi, maka modularitas adalah hal yang penting. Merancang sistem sebagai suatu urutan dari *layer* atau dengan menggunakan *mikrokernel* merupakan salah satu teknik yang baik. Konsep *virtual machine* mengambil pendekatan *layer* dan memperlakukan baik itu *kernel* dari sistem operasi dan perangkat kerasnya sebagai suatu perangkat keras. Bahkan sistem operasi yang lain dapat dimasukkan diatas *virtual machine* tersebut. Setiap sistem operasi yang mengimplemen JVM dapat menjalankan semua program java, karena JVM mendasari dari sistem ke program java, menyediakan arsitektur tampilan yang netral.

Didalam daur perancangan sistem operasi, kita harus berhati-hati untuk memisahkan pembagian kebijakan (*policy decision*) dengan detail dari implementasi (*mechanism*). Pemisahan ini membuat fleksibilitas yang maksimal apabila *policy decision* akan diubah kemudian. Sistem operasi sekarang ini hampir selalu ditulis dengan menggunakan bahasa tingkat tinggi. Hal ini meningkatkan implementasi, perawatan portabilitas. Untuk membuat sistem operasi untuk suatu konfigurasi mesin tertentu, kita harus melakukan *system generation*.

# 1.5. Pertanyaan

1. Sebutkan tiga tujuan utama dari sistem operasi!
2. Sebutkan keuntungan dari *multiprogramming*!
3. Sebutkan perbedaan utama dari sistem operasi antara komputer *mainframe* dan PC?
4. Sebutkan kendala-kendala yang harus diatasi oleh *programmer* dalam menulis sistem operasi untuk lingkungan waktu nyata?
5. Jelaskan perbedaan antara *symmetric* dan *asymmetric multiprocessing*. Sebutkan keuntungan dan kerugian dari sistem *multiprosessor*!
6. Apakah perbedaan antara *trap* dan *interrupt*? Sebutkan penggunaan dari setiap fungsi tersebut!
7. Untuk jenis operasi apakah **DMA** itu berguna? Jelaskan jawabannya!
8. Sebutkan dua kegunaan dari *memory cache*! Problem apakah yang dapat dipecahkan dan juga muncul dengan adanya *cache* tersebut?
9. Beberapa **CPU** menyediakan lebih dari dua mode operasi. Sebutkan dua kemungkinan penggunaan dari mode tersebut?
10. Sebutkan lima kegiatan utama dari sistem operasi yang berhubungan dengan managemen proses!
11. Sebutkan tiga kegiatan utama dari sistem operasi yang berhubungan dengan managemen memori!
12. Sebutkan tiga kegiatan utama dari sistem operasi yang berhubungan dengan managemen *secondary-storage*!
13. Sebutkan lima kegiatan utama dari sistem operasi yang berhubungan dengan managemen berkas!
14. Apakah tujuan dari *command interpreter*? Mengapa biasanya hal tersebut terpisah dengan *kernel*?

# 1.6. Rujukan


1. Andrew S. Tanenbaum, *Operating Systems : Design and Implementation-2/E,* Prentice Hall, 1997
2. Harvey M.Deitel, Paul J.Deitel, David R.Choffness, *Operating Systems-3/E,* Prentice Hall, 2004
3. Lubomir F.Bic, Alan C.Shaw, *Operating Systems Principles,* Prentice Hall, 2003
4. Silberchatz, Galvin, Gagne, *Operating System Concepts-6//E,* John Wiley & Sons, 2001
5. William Shay, *Introduction to Operating Systems*, Prentice Hall, 1993
6. William Stallings, *Operating Systems : internals and Design Principles*, Prentice Hall, 2001
7. *http://www.csc.uvic.ca/~mcheng/360/notes/NOTES2.html* (http://www.csc.uvic.ca/~mcheng/360/notes/NOTES2.html)
8. *http://www.chipcenter.com/circuitcellar/march02/c0302dc4.htm* (http://www.chipcenter.com/circuitcellar/march02/c0302dc4.htm)
9. *http://www.osdata.com/kind/history.htm* (http://www.osdata.com/kind/history.htm)
10. *http://www.imm.dtu.dk/courses/02220/OS/OH/week7.pdf* (http://www.imm.dtu.dk/courses/02220/OS/OH/week7.pdf)
11. *http://www.mcsr.olemiss.edu/unixhelp/concepts/history.html* (http://www.mcsr.olemiss.edu/unixhelp/concepts/history.html)
12. *http://www.cs.panam.edu/fox/CSCI4334/ch3.ppt* (http://www.cs.panam.edu/fox/CSCI4334/ch3.ppt)
13. *http://www.cis.umassd.edu/~rbalasubrama/* (http://www.cis.umassd.edu/~rbalasubrama/)
14. *http://legion.virginia.edu/presentations/sc2000/sld001.htm* (http://legion.virginia.edu/presentations/sc2000/sld001.htm)
15. *http://www.cs.wpi.edu/~cs502/s99/* (http://www.cs.wpi.edu/~cs502/s99/)
16. *http://cs-www.cs.yale.edu/homes/avi/os-book/osc/slide-dir/* (http://cs-www.cs.yale.edu/homes/avi/os-book/osc/slide-dir/)
17. *http://www.hardware.fr/articles/338/page1.html* (http://www.hardware.fr/articles/338/page1.html)
18. *http://www.cs.technion.ac.il/~hagit/OSS98* (http://www.cs.technion.ac.il/~hagit/OSS98)
19. *http://www.ignou.ac.in/virtualcampus/adit/course/index-tr1.htm* (http://www.ignou.ac.in/virtualcampus/adit/course/index-tr1.htm)
20. *http://www.techrescue.net/guides/insthware.asp* (http://www.techrescue.net/guides/insthware.asp)
21. *http://agt.buka.org/concept.html* (http://agt.buka.org/concept.html)
22. *http://kos.enix.org/pub/greenwald96synergy.pdf* (http://kos.enix.org/pub/greenwald96synergy.pdf)