MODUL MATAKULIAH
SISTEM OPERASI

KASINI, S.KOM., M.KOM.

1.PENDAHULUAN

System operasi merupakan bagian penting dalam system computer, tanpa adanya system operasi semua aplikasi yang ada pada system computer tidak akan berjalan dengan baik. Pemilihan system operasi juga disesuaikan dengan kebutuhan pengguna.

1.1 Pengertian system operasi

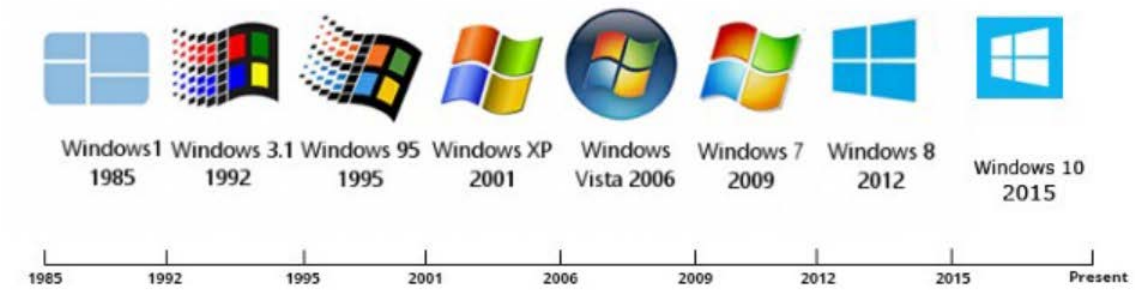
Pengertian sistem operasi secara umum ialah pengelola seluruh sumber-daya yang terdapat pada sistem komputer dan menyediakan sekumpulan layanan (system calls) ke pemakai sehingga memudahkan dan menyamankan penggunaan serta pemanfaatan sumber-daya sistem komputer.

Pengertian system operasi menurut tanenbaum dan silberchatz adalah : An operating system is software that manages the computer hardware ,as well as providing an environment for application programsto run. Perhaps the most visible aspect of on operating system is the interpace to the computer system it provides to the human user . (silberschatz, 2012,)

A layers of software whose job is to provide user programs with a better, simpler, cleaner, model of the computer and to handle managing all the resources just mentioned. (Tanenbaum , 2015)

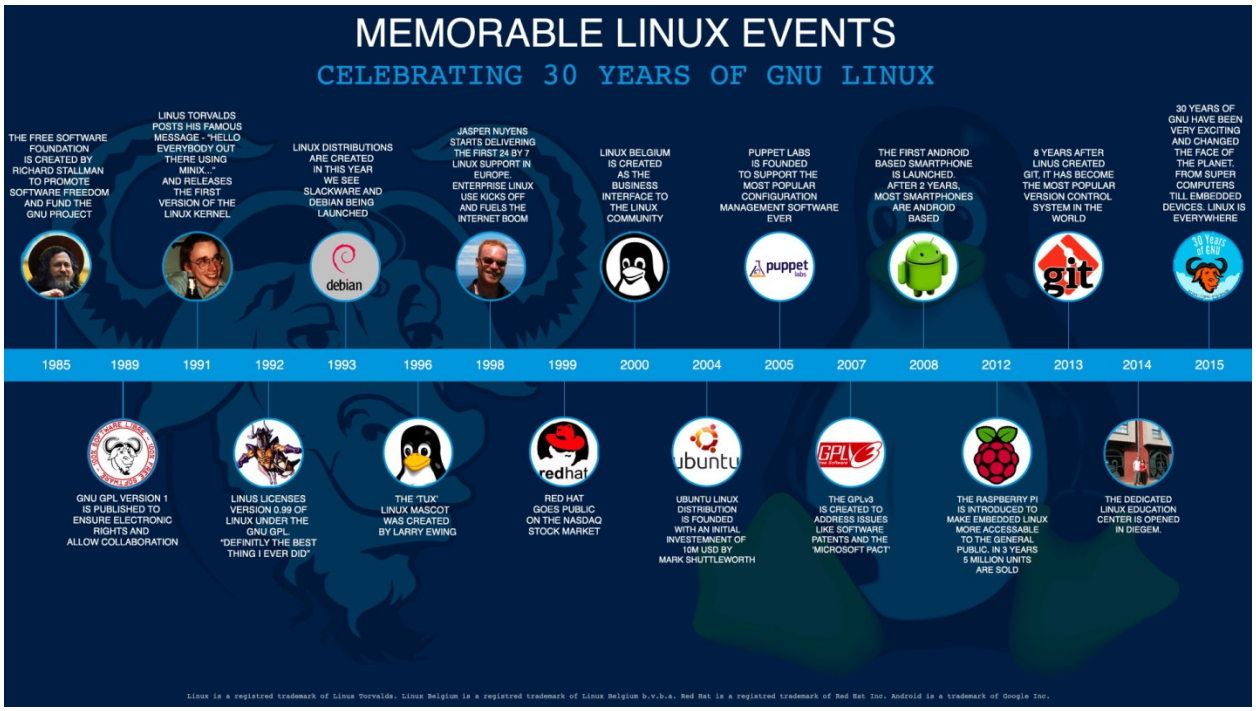
Sistem operasi sebuah perangkat lunak yang memanajemen perangkat keras komputer untuk menyediakan lingkungan bagi program aplikasi didalam komputer agar berjalan dengan baik dan menyediakan antarmuka yang simpel dan sederhana untuk memudahkan pengguna nya.

Pada masa sekarang ini perkembangan system operasi sangat berkembang dnegan pesat, baik system operasi mobile maupun system operasi computer. Berikut gabar dari perkembangan system operasi dari masa ke masa:



Gambar 1.1 perkembangan system operasi windows sampai saat ini

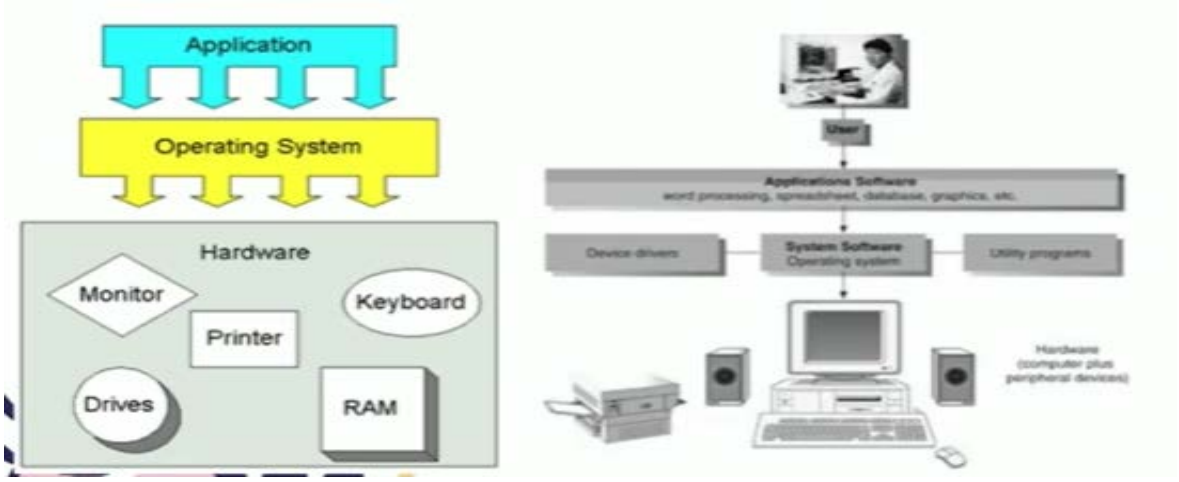
Selain windows system operasi yang berbasis open source juga berkembang pesat seperti pada gambar berikut ini:



Gambar 1.2 perkembangan system operasi linux

2. Tujuan

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 (Hartono, 2018).



Gambar 2.1 sistem operasi melakukan pengelolaan sumber daya

Sistem Operasi memiliki tugas mengelola seluruh komponen ada pada komputer agar dapat berjalan dengan baik dan efisien. Selain itu system operasi juga sebagai penyedia layanan yang

menyediakan sekumpulan layanan ke pengguna sehingga dapat memudahkan pengguna dalam menggunakan sistem komputer, sistem ini disebut dengan system call.

Didalam system operasi terdapat Sumber daya komputer yang memiliki pengertian seluruh komponen yang ada dikomputer yang memberi manfaat, memanfaatkan sumber daya secara efisien dan benar merupakan tujuan dari sistem operasi.

1. Sumber daya fisik seperti kabel, perangkat keras.
2. Sumber daya abstrak seperti data contohnya PCB, semaphore, tabel segmen, berkas/file, linklist, dan Program adalah kumpulan intruksi yang dijalankan oleh prosesor

Sistem operasi sebagai penyedia layanan yang memprogram di sistem komputer secara langsung melalui instruksi instruksi mesin bukanlah pekerjaan yang mudah dan tentu rumit karena memiliki rincian yang beragam dan detail. Sistem operasi menyediakan tatacara pemanfaatan sumber daya sistem komputer secara mudah dan seragam. Tata cara ini pemanfaatan ini diakukan dengan sekumpulan **system call** yang dapat dipanggil di program yang dibuat pemogram aplikasi. Pemanggilan system call jauh lebih mudah dibanding memprogram secara langgung. Sistem operasi ini menyediakan layanan pengaksesan sumberdaya sehingga pemogram tidak dirumitkan dengan rincian operasi perangkat keras yang sangat rumit, dengan cara ini pemogram dapat memandang sistem operasi sebagai penyedia layanan level lebih tnggi. Layanan layanan ini lebih mudah digunakan dibandingkan dengan membuat program dengan bahasa mesin secara langsung.

Layanan pada system operasi berupa:

1. Pembuatan program, berupa program utilitas (bukan bagian dari sistem operasi tetapi dapat diakses melalui sistem operasi)

2. Eksekusi program, sejumlah tugas perlu dilakukan untuk mengeksekusi program. Instruksi dan data harus dmuat dimeory lebih dahulu perngkat2 masukan keluaran dan berkas harus diinisialisasi lebih dahulu dan sumberdaya sumberdaya harus disiapkan terlebih dahulu. Sistem operasi harus dapat menangani semua itu untuk pengguna atau pemkaai program nya.

3. Pengaksesan perangkat I/O setiap perangkat I/O membutuhkan sejumlah instruksi atau sinyal kendali yang cukup rumit ,sistem operasi harus mengambil alih rincian rincian tersebut sehingga pemogram dapat berfikir lebih sederhana dalam memanfaatkan perangkat.

4. Kendali terhadap pengaksesan berkas pada sistem dengan banyak pemakai secarab simultan , sistem operasi menyediakan mekanisme sistem proteksi untuk mengendalikan pengaksesan kepada berkas.

5. Pengaksesan sistem pada syetem public atau sistem yang dipakai bersama, SO mengendalikan pengaksesan ke sumberdaya sumberdaya sistem secara keseluruhan, fungsi pengaksesan harus menyediakan proteksi sumberdaya dan data yang tidak diotorsasi, sehingga harus menyelesaikn konflik2 dalam perebutan sumber daya

6. Deteksi dan memberi tangapan tehadap kesalahan beragam kesalah dapat muncu dalam sistem komputer, maka dari itu sistem operasi harus memberikan tanggapan pada sistem seperti pengakhiran program yang menyebabkan kesalahan atau mencoba ulang atau sekedar melakukan kesalahan.

System call adalah prosedur atau rutin yang disediakan oleh pusaka api dari sistem operasi. Layanan system call pada sistem operasi berfungsi sebagai perantara antara program atau perangkat lunak dengan sistem operasi (OS).

Cara kerja system call:

a. Program pengguna mengeluarkan panggilan system.

b. CPU beralih dari mode pengguna ke mode kernel.

c. Fungsi terkait di kernel dijalankan.

d. Setelah operasi selesai, kontrol dikembalikan ke mode pengguna.

Didalam system operasi terdapat beberapa komponen utama:

1. Manajemen proses

2. Manajemen memori

3. Manajemen I/O

4. Manajemen berkas

3. Struktur Sistem Operasi

Ada 4 jenis struktur pada system operasi :

1. Monolitik

2. Berlapis [layered]

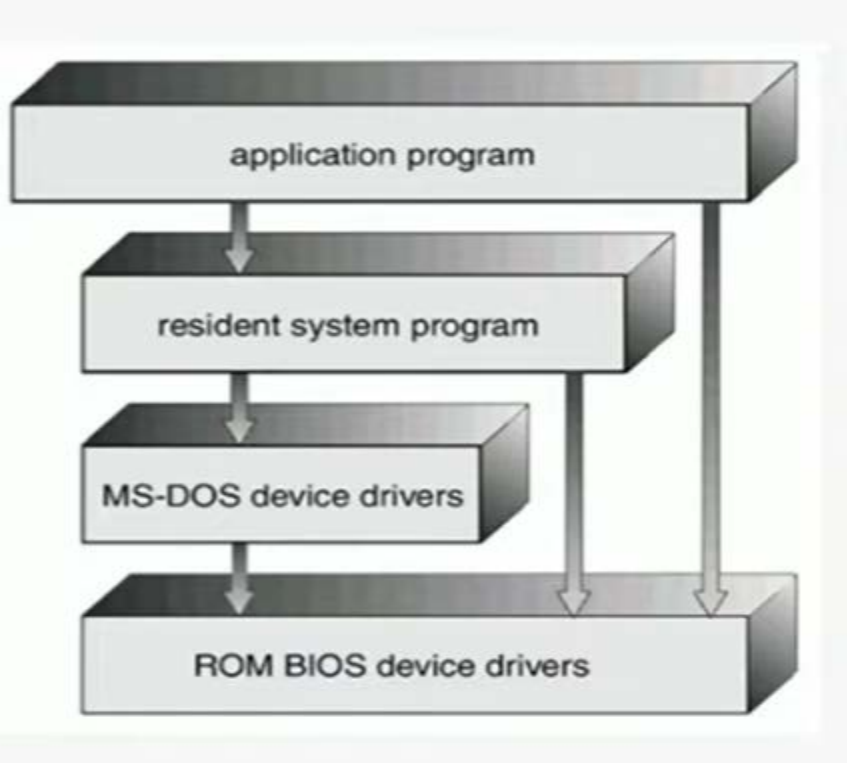
3. Client-server[mikrokernel]

4. Modular

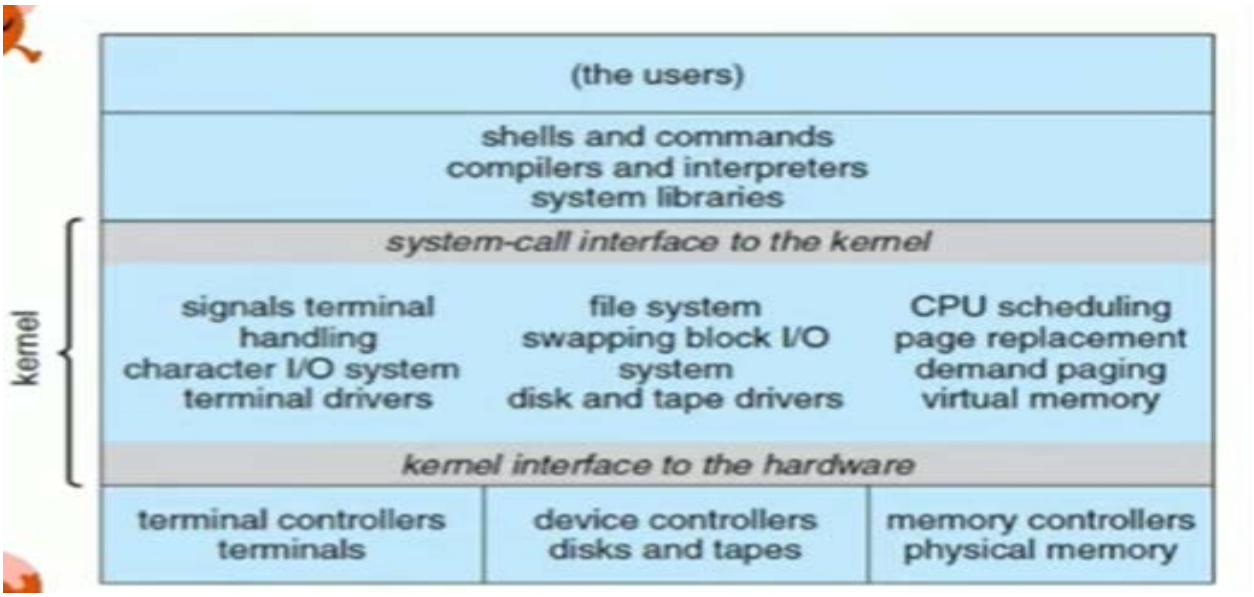
3.1 Struktur monolitik pada DOS

Struktur monolitik ini pada dasarnya konstruksi tanpa struktur. Jadi semua komponen sistem operasi mssih bercampur jdi satu. Semua rutin dapat mengakses rutin lainnya contohnya Ms.DOS.

Pada SO Ms Dos aplikasi dan sistem operasi belum ada pemisahan yang jelas , ini yg menyebabkan memudahkan program2 virus merusak dan memodifikasi sistem operasi Ms Dos. Pada sistem operasi Ms Dos program aplikasi memiliki akses untuk memodifikasi bagian sistem operasi.



Gambar 3.1 Sistem Monolitik



Gambar 3.2 system monolitik pada Unix

Pada sistem operasi unix sudah ada pemisahan antara proram aplikasi sistem operasi. Program aplikasi hanya dapat mengakses rutin rutin sistem operasi melalui system call, akan tetapi rutin Sistem operasi seperti manajemen sistem berkas, algoritma penjdwalan proses, driver disk dan tape masih bercampur menjadi satu, dan dsinilah masih ada sisi sistem monolitik. Pada sistem operasi unix ini sudah terlihat ada kernel, kernel adalah tempat dimana program2 sistem operasi dijalankan.

Kelebihan dan kekurangan struktur monolitik:

Kelebihannya:

a. Sederhana
b. Layananya cepat

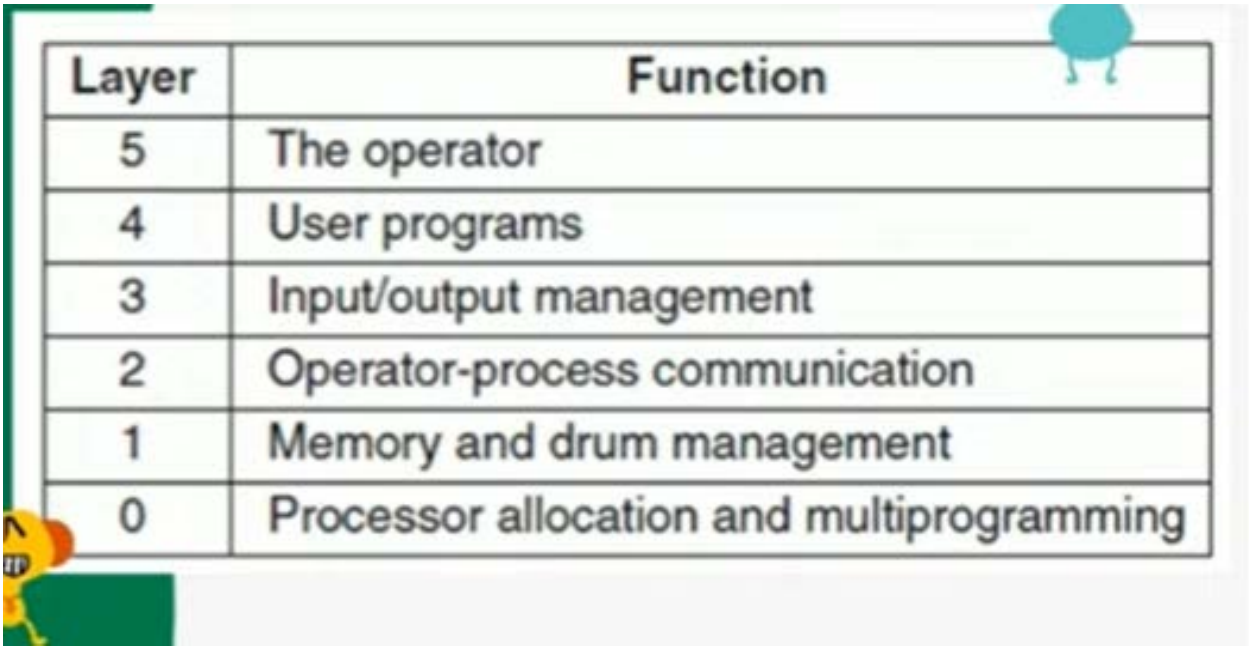
Kekurangannya:

a. Kurangnya keamanan
b. Kesalahan pada satu bagian dapat mengakibatkan seluruh sistem tergangu
c. Sulitnya pengembangan

3.2 Struktur berlapis (layered)

Pada struktur berlapis ini rutin rutin so dikelompokkan dalam lapisan, setiap lapisan bersifat modular. Lapisan paling bawah menangani detail operasi perangkat keras dan sedangkan lapisan bagian paling atas menangani antarmuka pengguna atau program aplikasi.

Salah satu ciri khas struktur berlapis ini rutin-utin lapisan hanya boleh menggunakan rutin2 lapisan yang ada dibawahnya. Sistem oeprasi yang menggunkan sistem ini adalah sistem operasi THE. THE OS adalah sistem operasi komputer yang dirancang oleh tim yang dipimpin oleh Edsger W. Dijkstra , dijelaskan dalam monograf pada 1965-1966 dan diterbitkan pada tahun 1968. Dijkstra tidak pernah menyebutkan sistemnya; “THE” hanyalah singkatan dari “Technische Hogeschool Eindhoven”. Sistem THE terutama merupakan sistem batch yang mendukung multitasking tu tidak dirancang sebagai sistem operasi multi-pengguna Itu jauh seperti SDS 940 , tetapi “rangkaian proses dalam sistem THE statis”.



Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Gambar 3.3 Struktur berlapis

Kelebihan dan kekurangan struktur berlapis (layered):

Kelebihan:

a. Pengembangan dan pemeliharaan lebih mudah

b. Kesalahan pada 1 lapisan tidak menggangu seluruh sistem

c. Pengembangan dapat di lakukan secara modular

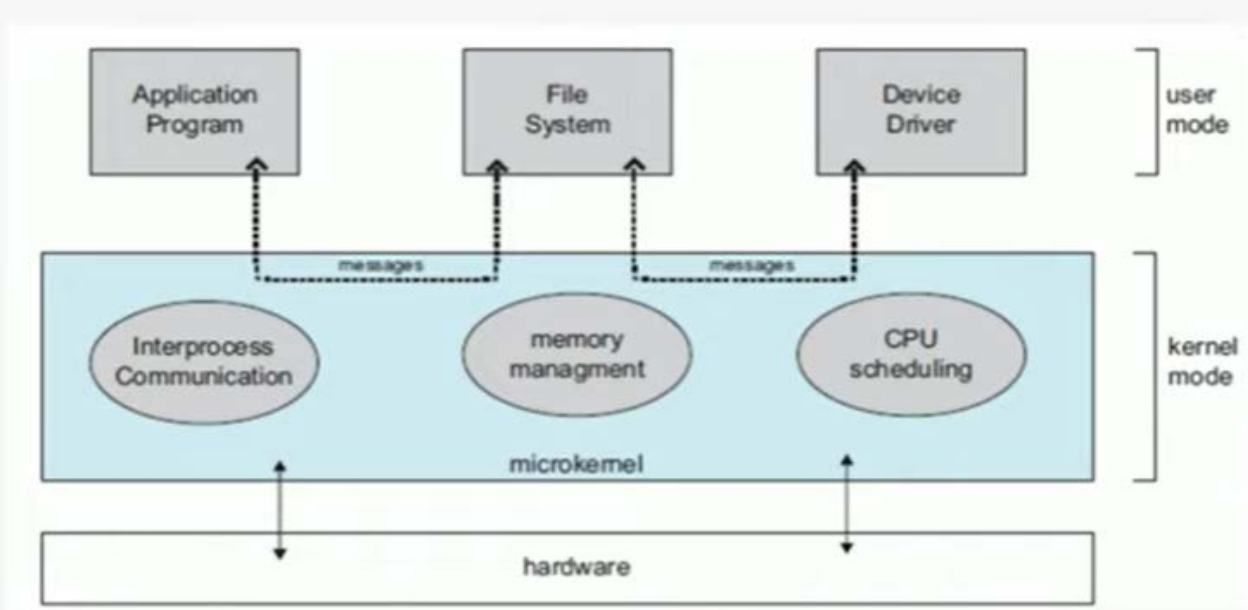
Kekurangan: Membagi rutin SO ke dalam lapiran –lapisan bukan hal yang mudah

3.3 Struktur client server (mikrokernel)

Sistem operasi merupakan sekumpulan proses dalam hal ini proses dikategorikan menjadi server dan client. Client dan server saling berinteraksi saling melayani yaitu server adalah proses yang menyediakan layanan sedangkan client adalah proses yg memerlukan atau

meminta layanan. Proses client yg memerlukan layanan mengirimkan pesan permintaan ke server dan menanti jawaban pesan. Proses server melakukan tugas yang diminta dan setelah itu proses server mengirim hasil dalam bentuk pesan jawaban ke proses client, proses server hnya memnaggapi permintaan client dan tidak memulai percakapan dengan client lebih dulu, jadi pada struktur cient server ini komunkasi dilakukan dengan pertukaran pesan.

Dengan struktur client server ini kode dapat diangkat ke level lebih tinggi sehingga kernel dapat dibuat sekecil mungkin. Hampir semua tugas diangkat menjadi level pemakai. Kernel hanya mengatur komunikasi antar client dan server. Client dengan ukuran kecil ini popular dengan sebutan mikrokernel



Gambar 3.4 Struktur Client Server

Kelebihan dan kekurangan struktur client –server(mikrokernel):

kelebihan

a. Pengembangan dan pemeliharaan lebih mudah
b. kesalahan pada 1 bagian tidak menggangu seluruh sistem
c. Mudah di adaptasi untuk membangun struktur yang baru

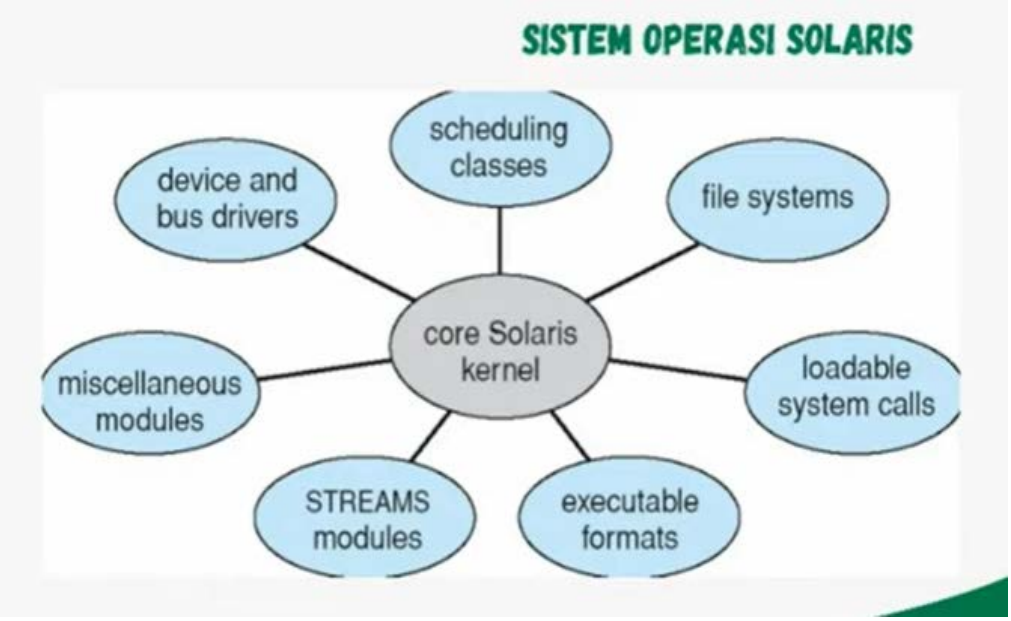
kekurangan

a. layanan lebih lambat --> message passing

b. Dapat terjadi fenomena bottle neck

3.4 Struktur modular

Sejauh ini metodologi desain sistem operasi menggunakan berorintasi objek teknik adalah teknik terbaik untuk membuat kernel modular. Kernel memiliki komponen dasar yang dapat terhubung secara dinamis selama proses boot atau selama proses running seprti pada sistem operasi linux, solaris dan Mac OS. Sistem Operasi tersebut mengimplemetasikan loadable modul yang dapat digunakan secara dinamis.



Gambar 3.4 Struktur Modular

Satu modul dapat memanggil modul lainnya, sebagai contoh suatu device and bus driver untuk perangkat keras yang spesifik tertentu ketika membutuhkan suatu modul lain maka dapat memangil modul lain tersebut sebagai loadable modul.

4. Manajemen Proses

Manajemen proses merupakan salah satu komponen utama dalam system operasi. Pada manajemen proses pembahasan materi sangat panjang dibanding dengan komponen yang lainnya. Mari kita simak materi pada manajemen proses.

4.1 Proses

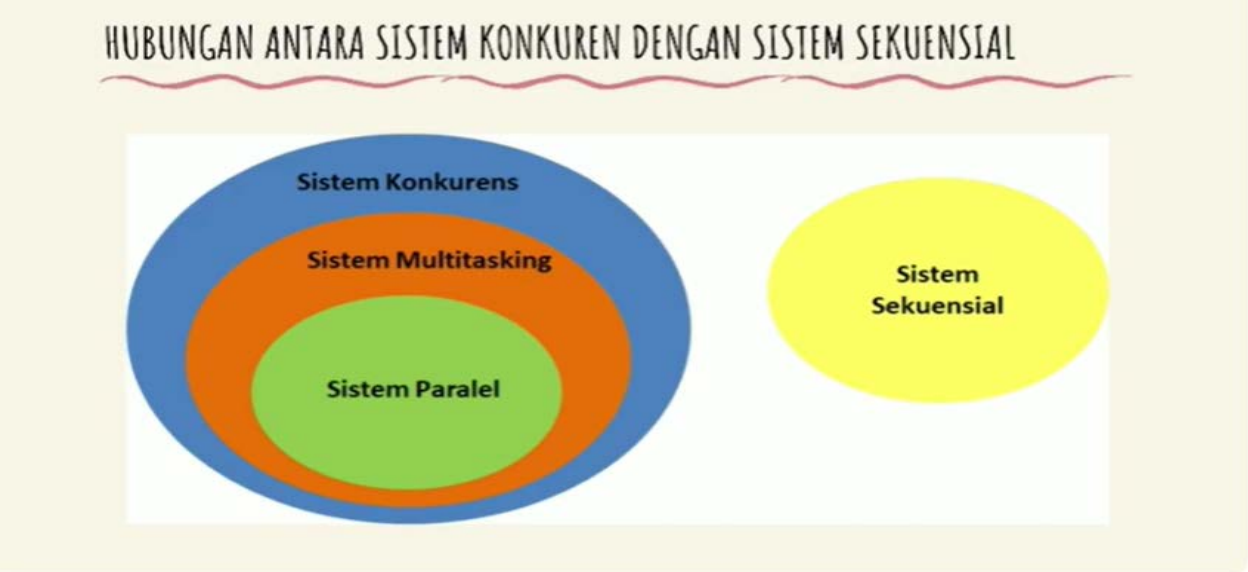
Proses adalah suatu program yang sedang dijalankan atau eksekusi dan berlangsung secara sequential. Proses merupakan unit kerja terkecil yang secara individu memiliki sumber daya sumber daya dan merupakan unit terkecil yang dijadwalkan oleh sistem operasi.

Pada saat proses terjadi ada beberapa yang akan dimuat yaitu :

1. Program counter yaitu suatu program yang dapat dilakukan berulang-ulang dan keberadaan nya di posisi paling depan yg akan diproses. Program counter digunakan untuk menyimpan alamat lokasi dari memori utama yang berisi instruksi yang sedadang diproses.

2. Register yaitu sebagai berkaitan dengan komponen tempat penyimpanan.

3. Variabel yaitu sama dengan nama file artinya jika kita akan memproses sebuah proses dan program tersebut dapat kita gunakan berulang ulang dan program tersebut disimpan dimemory penyimpanan memiliki nama sperti ms.office



Gambar 4.1 Hubungan antara system konkuren dan sekunsial

Konkurensi merupakan keadaan dimana 2 atau lebih proses berjalan pada saat yang “bersamaan”.Bersamaan disini berarti suatu proses bisa aktif berjalan tanpa harus menunggu proses lainnya selesai seluruhnya.

Sistem Sekuensial Suatu proses dieksekusi sampai selesai, baru kemudian proses berikutnya dieksekusiKetika proses yang sedang dieksekusi melakukan operasi I/O, prosesor harus menunggu sampai operasi tersebut selesai (prosesor tidak bisa dialihkan ke proses lain)Kelemahannya adalah tingkat utilitas prosesor rendah (idle time CPU besar).

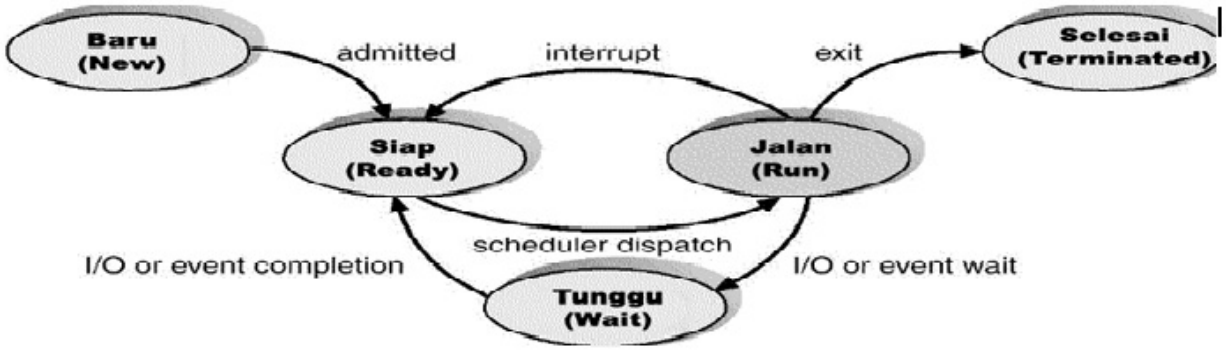
Pada sistem komputer dengan jumlah prosesor lebih dari satu atau satu prosesor multicore, proses-proses dieksekusi secara bersamaan pada saat yang bersamaan pula. Sistem ini dikenal dengan sebutan sistem paralel atau multiprocessing.Pada sistem single prosesor, konkurensi diimplementasikan dengan meng-interleave proses-proses, sedangkan pada sistem multiprosesing, proses-proses yg aktif di interleave dan di overlap.

Suatu proses meliputi kode program, data, stack dan atribut proses (informasi id proses, status proses, penggunaan sumber daya seperti memori, I/O, file,dll).

4.2 Siklus hidup (Status) proses system operasi

Pada saat program sedang diproses atau dieksekusi maka akan ada perubahan status. Status proses didefiniskan sebagai bagian dari aktivitas proses yang sedang berlangsung. Status proses dapat digambarkan dengan :

1. Diagram status proses
2. PCB (program/proses control block)
   Dalam status proses maka tiap tiap proses yang dilakukan memiliki status seperti: new, ready, running, waiting dan terminated. Berikut gambar diagram status proses:



Gambar 4.2 Diagram Status Proses

1. New
   a. Masih dalam tahap inisiasi oleh prosedur
   b. Meliputi alokasi memory utama untuk proses
   c. Pengisian tabel proses
   d. Pembuatan struktur data kendali untuk menyimpan informasi dan status proses
   e. Belum siap untuk di eksekusi
   f. Kondisi yang memicu proses new
      - Login ke sistem operasi
      - Permintaaan eksekusi program
      - Aplikasi yang menciptakan proses anak
      - Penciptaan proses baru dari eksekusi batch
2. Ready
   a. Proses yang telah berhasil di inisiasi
   b. Antrian penjadwalan prosesor dengan cara menyisipkan proses baru ke dalam antrian
   c. Berisi referensi atau pointer ke struktur data kendali proses
   d. Menandakan suatu proses siap berkompetisi untuk mendapatkan alokasi prosesor

e. Schduler adalah sistem operasi yang bertugas untuk memilih proses yang berada dalam proses ready

3. Running
    a. Proses menguasi prosesor sepenuhnya
    b. Memiliki tiga kemungkinan
    c. Teminated, proses yang telah selesai
    d. Ready, jika jatah waktu yang dialokasikan sudah habis
    e. Blocked/waiting
4. Blocked/Waiting
    a. Proses membutuhkan pembacaan data dari piranti I/O
    b. Proses ini akan disisipkan pada antrian penjadwalan peranti I/O atau event
    c. Jika I/O yang di tunggu sudah selesai maka proses akan kembali ke antrian ready dan menunggu pemillihan oleh schedule
5. EXIT/Terminated
    a. Proses tersebut sudah dihentikan eksekusinya
    b. Proses telah selesai secara normal
    c. Batas waktu total sudah terlewati
    d. Kekurang ruang memory
    e. Pelanggaran batas memory
    f. Pelanggaran proteksi berkas
    g. Kesalahan aritmatika
    h. Waktu tunggu melebihi batas
    i. Terjadi kegagalan I/O
    j. Instruksi tidak benar
    - Terjadi pemakaian instruksi yang tidak di izinkan

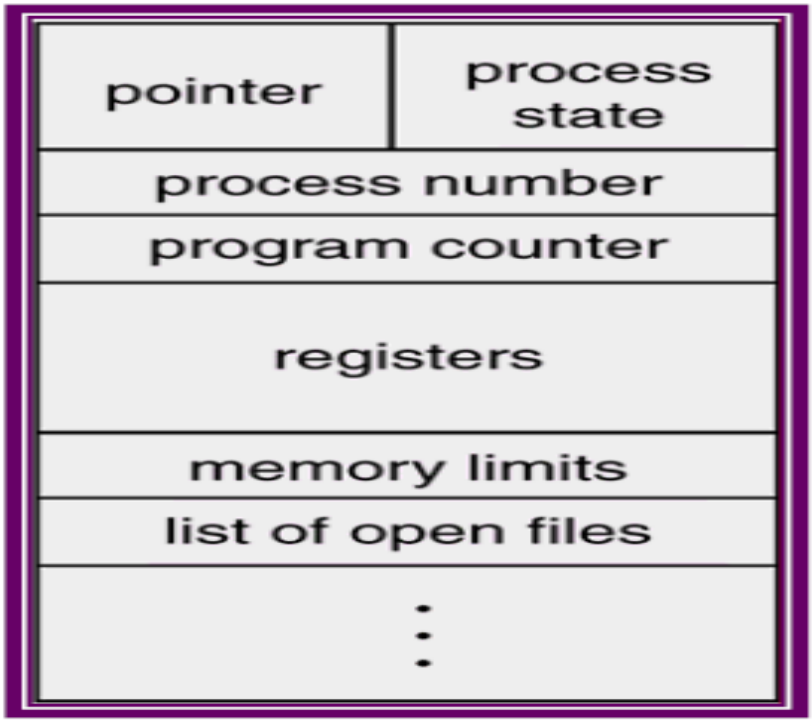
4.3 Proses control block (PCB)

Setiap proses di gambarkan dalam sistem operasi oleh sebuah PCB. PCB memuat informasi tentang proses, yaitu: sebuah tanda pengenal proses (Process ID) yang unik dan menjadi nomor identitas, status proses, prioritas eksekusi proses dan informasi lokasi proses dalam memori. Prioritas proses merupakan suatu nilai atau besaran yang menunjukkan seberapa sering proses harus dijalankan oleh prosesor. PCB hanya berfungsi sebagai tempat penyimpanan informasi yang dapat bervariasi, dari proses yang satu dengan yang lain.

PCB berisikan banyak bagian dari informasi yang berhubungan dengan sebuah proses yang spesifik :

1. Keadaan proses: Keadaan proses, new, ready, running, waiting, halted, dan juga banyak lagi.

2. Program counter: Counter mengindikasikan address dari perintah selanjutnya untuk dijalankan untuk proses ini.
3. CPU register: Register bervariasi dalam jumlah dan jenis, tergantung pada rancangan komputer. Register tersebut termasuk accumulator, index register, stack pointer, general-puposes register, ditambah code information pada kondisi apa pun. Besertaan dengan program counter, keadaan/ status informasi harus disimpan ketika gangguan terjadi, untuk memungkinkan proses tersebut berjalan/ bekerja dengan benar setelahnya.
4. Informasi managemen memori: Informasi ini dapat termasuk suatu informasi sebagai nilai dari dasar dan batas register, tabel page/ halaman, atau tabel segmen tergantung pada sistem memori yang digunakan oleh sistem operasi.
5. Informasi pencatatan: Informasi ini termasuk jumlah dari CPU dan waktu riil yang digunakan, batas waktu, jumlah akun, jumlah job atau proses, dan banyak lagi.
6. Informasi status I/O: Informasi termasuk daftar dari perangkat I/O yang di gunakan pada proses ini, suatu daftar open berkas dan banyak lagi.



Gambar 4.2 Proses Control Blok

Informasi di PCB dikelompokkan sebagai berikut:

1. Informasi Identifikasi Proses

    a. Berkaitan dengan identitas proses yang unik .
    b. Identifiernya adalah numerik yang meliputi

        – Identifier proses

– Identifier proses yang menciptakan

– Identifier pemakai

2. Informasi Status Proses, Informasi ini esensinya terdiri dari register- register pemroses. Saat proses berstatus running, informasi- informasi ini berada di register-register. Ketika proses diinterupsi semua informasi register harus disimpan agar dapat dikembalikan saat proses dieksekusi kembali. Jumlah dan ragam register bergantung pada arsitektur komputernya
3. Informasi Kendali Proses Adalah informasi-informasi lain yang diperlukan SO untuk mengendalikan dan koordinasi beragam proses aktif cth informasi penjadwalan.

4.3.1 Elemen-elemen dari Process Control Block (PCB) :

1. Identifier : menjelaskan proses yang sedang terjadi.
2. State : kondisi yang terjadi pada proses
3. Priority : urutan perintah yang jelas pad suatu proses
4. Program counter : instruksi pada proses
5. Memory pointers : media penyimpanan (penunjuk alamat) pada proses
6. Context data : data yang berkaitan dengan proses
7. I/O status information : terdapat masukan dan keluaran yang diinginkan
8. Accounting information : memberikan informasi yang dibutuhkan

4.4 Operasi-Operasi Pada Proses

SO dalam mengelola proses dapat melakukan operasi-operasi terhadap proses. Operasi-operasi terhadap proses

1. Penciptaan proses (create process)
2. Penghancuran/terminasi proses (destroy a process
3. Penundaan proses (suspend a process)
4. Pelanjutan kembali proses (resume process)
5. Pengubahan prioritas proses
6. Memblok proses
7. Membangunkan proses
8. Menjadwalkan proses
9. Memungkinkan proses berkomunikasi dengan proses lain

4.5 Thread

Thread adalah unit (urutan tugas)dari proses yang dieksekusi diantara sebuah proses. Sebuah proses memiliki sebuah tread yang dapat dikelola secara independen sesuai dengan jadwal. Thread adalah pengendali proses. Thread terdiri dari ID thread, himpunan register, program counter, dan stack. Thread dapat melakukan lebih dari satu pekerjaan pada waktu yang sama.

Thread merupakan bagian dari proses. Tiap proses mempunyai informasi statusdan sumber daya sendiri, thread berbagi infomasi status dan sumber daya dengan thread yang lain dalam satu proses. Tiap proses mempunyai alamat yang berbeda sedangkan thread berbagi alamat yang sama. Pergantian antar thread lebih cepat daripada antar proses.

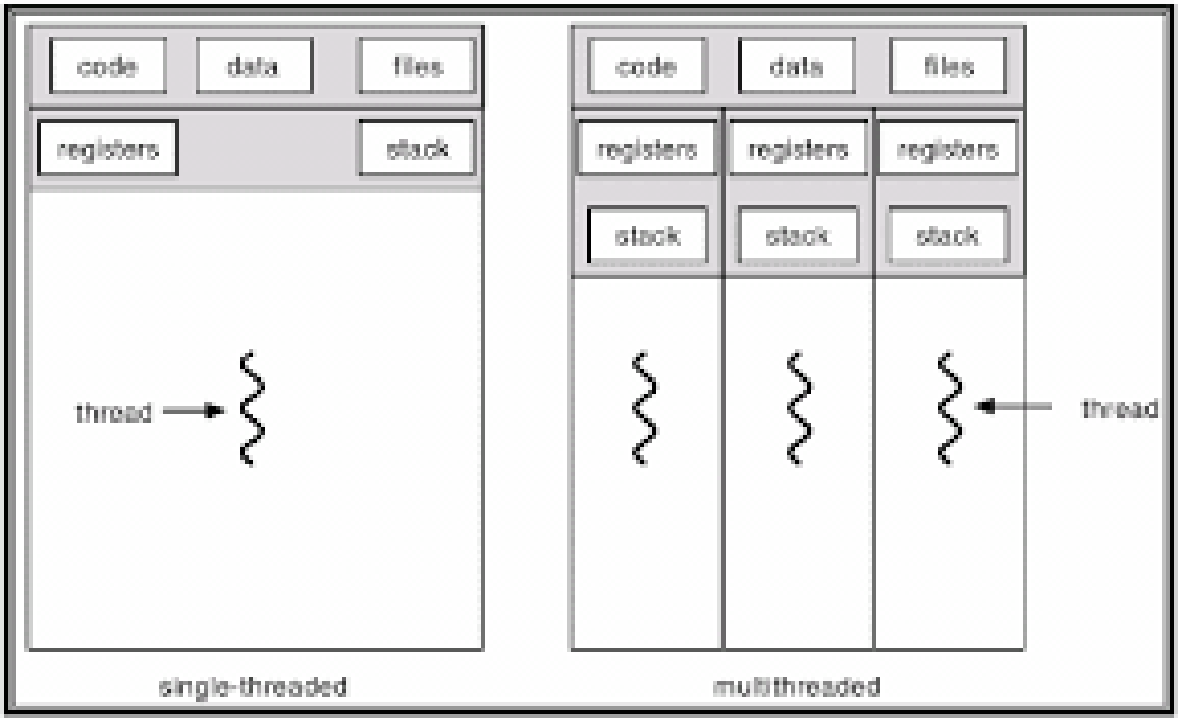
4.5.1 Komponen dari thread

1. Thread ID
2. Program counter
3. Register set
4. Stack space

Suatu thread dengan thread-thread lainnya bisa berbagi: Code section, Data section, Operating system. Resource

4.5.2 Jenis – Jenis Thread

1. Proses dengan singletrhread adalah proses menjalankan satu tugas dalam satu waktu.
2. Sistem operasi modern (multithread) adalah proses menjalankan banyak tugas dalam satu waktu. Multithread pada satu prosesor dijalankan bergantian dengan waktu yang sangat cepat sehingga tampak bersamaan. Multithread pada multiprosesor benar benar dijalankan secara bersamaan. Contohnya Web browser (memiliki satu thread untuk display image sedankan thread yang lain mengambil data dari jaringan). Thread untuk menampilkan halaman web, gambar, Thread untuk mengunduh data dari jaringan, Pengolah kata thread untuk menerima dan menampilkan hasil pengetikan Thread untuk mengecek grammar, Webserver thread untuk menangani request dari banyak user
   Keuntungan multithreading:
   a. Responsiveness : program akan tetap berjalan walaupun ada sebagian tugas yang memakan waktu lama Cth: web browser sedang ada aktivitas download
   b. Resource sharing : antar thread dalam satu proses sudah bisa berbagi data
   c. Economy: antar thread sudah berbagi resource yang dialokasikan untuk sebuah proses.
   d. scability: keuntungan dari multithreading dapat meningkat secara drastis dalam arsitektur multiprosesor, dimana setiap thread dapat berjalan secara pararel pada prosesor yang berbeda.



Gambar 5.1 Multithreading dan single thread

3. Hyperthread menjalankan banyak thread dalam waktu bersamaan. Os harus yang mendukung seperti yang multiprocessor.

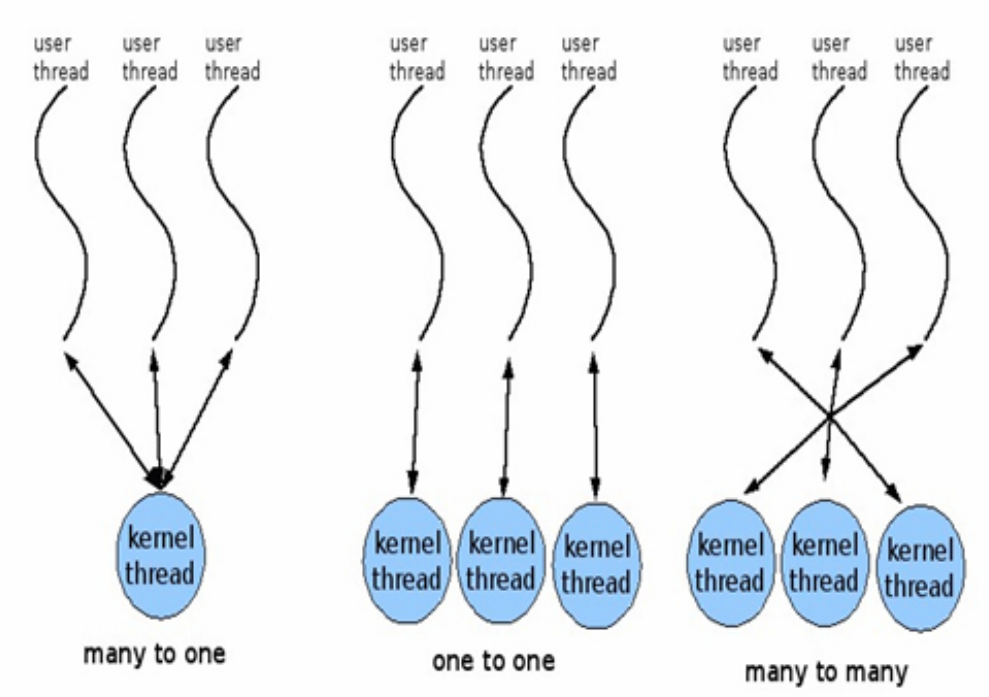
4.5.3 Kernel pada thread

Kernel threads didukung secara langsung dari sistem operasi. Pembuatan thread, penjadwalan, dan manajemen dilakukan oleh sistem operasi, secara umum kernel threads lebih lambat untuk dibuat dibanding user thread. Keuntungan:SO dapat menjadwalkan thread lain jika ada salah satu thread yang di block SO dapat menjadwalkan thread-thread pada multi processor Examples Windows XP/2000 ,Solaris, Linux Tru64, UNIX,Mac OS X.

4.5.4 Model model MultiThread

1. Many to One Model memetakan beberapa user-level threads ke satu kernel threads. Examples:Solaris Green ThreadsGNU Portable Threads. Kelemahan: Satu block semua block.
2. One to One model memetakan setiap user thread ke satu kernel thread. Example :Windows NT/XP/2000 Linux Solaris 9 and later. Keuntungan:Konkurensi Butuh space waktu,Tidak blocking, Terbatas
3. Many to many model memetakan banyak user-level thread ke kernel thread yang lebih sedikit atau sama.User dapat membuat banyak thread, masing2 kernel thread dpt

dijalankan di lingkungan multiprocessor. Examples Solaris prior to version 9Windows NT/2000 with the ,Thread Fiber package.



Gambar 4.2 Model model Multi Threading

4.6 Sinkronisasi Proses

Sinkronisasi adalah pengaturan jalannya beberapa proses pada system operasi yang berjalan secara bersamaan dan berbagi data. Tujuan sinkronisasi adalah untuk menghindari terjadi nya inkonsistensi data karena pengaksesan oleh beberapa proses yang berbeda atau mutual exclution. Mengatur urutan jalannya proses proses sehingga dapat berjalan dengan lancar dan proses terhindar dari deadlock atau starvation. Deadlock tidak dapat bergerak sedangkan starvation kehabisan atau kekurangan sumber daya. Manfaat sinkronisasi adalah sebagai penyimpan data sementara dan non sementara.

Konkurensi adalah sebuah kondisi dimana terdapat lebih dari satu proses berada pada saat yang sama. Proses proses yang konkuren memiliki beberapa masalah:

1) Mutual Exclusion yaitu Terdapat sumber daya yang tidak dapat dipakai bersama pada saat bersamaan, misalnya printer. Sumber daya yang tidak dapat dipakai bersama ini disebut sumber daya kritis. Bagian program yang menggunakan sumber kritis ini disebut memasuki critical section. Hanya satu program yang diijinkan masuk critical region. Sistem operasi menyediakan system call untuk hal ini.

2) Deadlock. Ilustrasi Deadlock Misalnya – Dua proses, P1 dan P2 – Dua sumber daya kritis, R1 dan R2 – Proses P1 dan P2 harus mengakses kedua sumber daya – Kondisi

yang terjadi R1 diberikan ke P1, sedang R2 diberikan ke P2. Karena untuk melanjutkan eksekusi memerlukan sumber daya sekaligus maka kedua proses akan saling menunggu sumber daya lain selamanya. Tak ada proses yang dapat melepaskan sumber daya yang telah dipegangnya karena menunggu sumber daya lain yang tak pernah diperolehnya. Kedua proses dalam keadaan deadlock, tidak dapat membuat kemajuan apapun

3) Starvation Misalnya Terdapat tiga proses P1, P2 dan P3 • P1, P2, P3 memerlukan pengaksesan sumber daya R secara periodik Skenario berikut terjadi :
    a. P1 sedang diberi sumber daya R, P2 dan P3 blocked menunggu sumber daya R
    b. Ketika P1 keluar dari critical section, P2 dan P3 diijinkan mengkakses R
    c. Asumsi P3 diberi hak akses. Kemudian setelah selesai, hak akses kembali diberikan ke P1 yang saat itu kembali membutuhkan sumber daya R Jika pemberian hak akses pergantian terus menerus antara P1 dan P3, maka P2 tidak pernah memperoleh pengaksesan sumber daya R, meski tidak ada deadlock. Pada situasi ini P2 disebut mengalami starvation.

Masalah yang timbul jika tidak ada sinkronisasi Proses yang berjalan secara bersamaan yang dapat membawa proses tersebut kedalam bahaya atau race condition. Race condition adalah suatu kondisi dimana dua atau lebih mengakses sumber daya secara bersamaan dan hasil akhir dari data tersebut tergantung dari proses mana yang terakhir dieksekusi.

Penyelesaian Masalah dalam sinkronisasi adalah sebagai berikut:

1) Critical section adalah suatu bagian yang berisi sejumlah variabel yang akan dipakai bersama (sharing)
2) Semaphore

Secara umum, penyelesaian critical section harus memenuhi 3 syarat:

1. Mutual exclusion. Jika suatu proses sedang mengerjakan critical section, maka tidak boleh ada proses lain yang masuk critical section tersebut

2. Progress. Jika tidak ada suatu proses yang mengerjakan critical section dan ada beberapa proses yang akan masuk ke critical section, maka hanya proses yang sedang berada pada entry-section saja yang boleh berkompetisi untuk mengerjakan critical section.

3. Bounded waiting. Besarnya waktu tunggu dari suatu proses yang akan memasuki critical section sejak proses itu meminta ijin untuk mengerjakan critical section, hingga permintaan itu dipenuhi

Dua solusi untuk critical section:

1) Solusi perangkat lunak

a. Algoritma decker adalah algoritma yang mencoba mengatasi masalah critical section untuk 2 proses yang menerapkan system bergilir pada proses yang ingin mengeksekusi critical section sehingga kedua proses bergantian menggunakan critical section.
b. Algoritma II adalah perbaikan dari algoritma dekker dengan flag. Flag ini mengatur proses mana yang akan masuk dalam critical section. Jika 0 = tidak memerlukan critical section jika 1= memerlukan critical section
c. Algoritma III (Peterson) adalah algoritma ini menggabungkan algoritma 1 dan II dengan menggabungkan flag and turn
d. Algoritma bakery menggunakan pada masing masing proses sehingga proses yang masuk ke critical section lebih berurutan

2) Solusi perangkat keras

a. Tes and set sebuah instruksi yang tidak bisa diiterupsi.

b. Semaphore adalah dua proses atau lebih dapat bekerja sama dengan menggunakan sinyal sederhana. Proses dapat dipaksa berhenti pada suatu tempat tertentu sampai menerima sinyal khusus ini. Variabel khusus untuk pensinyalan ini disebut semaphore.Untuk mengirim sinyal dengan semaphore, sebuah proses mengeksekusi primitive signal Untuk menerima sinyal dengan semaphore, sebuah proses mengeksekusi primitive wait Jika sinyal yang bersangkutan belum ditransmisikan, proses ter-suspend sampai transmisi terjadi.

4.7 Penjadwalan CPU

CPU pada dasarnya hanya dapat memproses satu proses dalam satu waktu. Proses lain akan dikerjakan jika keadaan CPU berstatus free. Penjadwalan adalah kebijaksaan dalam system operasi yang berhubungan dengan urutan kerja disistem operasi computer dengan konsep multiprogramming.

DIdalam system opersi terdapat CPU scheduler yang memiliki tugas scheduler yaitu memilih process di ready queue untuk dieksekusi di CPU. Keputusan untuk scheduling diambil ketika sebuah proses :

1. Pindah state dari running ke waiting
2. Pindah state dari running ke ready. e. x. interrupt
3. Pindah state dari waiting ke ready. e. x. I/O completion
4. Terminate
    a. Scheduling pada 1 dan 4 disebut nonpreemptive scheduling (tidak bisa interrupt) – Ekesekusi bisa berpindah tanpa harus menunggu proses yang sedang dieksekusi Terminate atau Waiting.
    b. Scheduling selainnya disebut preemptive scheduling

Kriteria penjadwalan nya sebagai berikut :

1. CPU utilization – buat CPU sesibuk mungkin. Jangan biarkan CPU idle
2. Throughput – Jumlah proses yang selesai dieksekusi per unit waktu
3. Turn around time – jumlah waktu untuk mengeksekusi sebuah proses
4. Waiting time – jumlah waktu sebuah proses harus menunggu di ready queue
5. Response time – jumlah waktu yang dibutuhkan dari pertama request eksekusi dikirim sampai respon pertama muncul (bukan output akhir)

Modul dispatcher memberikan kontrol CPU ke proses yang dipilih oleh short-time scheduler meliputi:

a. switching context
b. switching ke user mode
c. jumping to the proper location in the user program to restart that program
d. Dispatch latency – waktu yang dipakai dispatcher untuk menghentikan sebuah process dan menjalankan process lain

4.7.1 Tujuan utama penjadwalan system operasi

1. Adil / fairness artinya proses yang dijalankan dan yang ada dapat berjalan dan dapat diselesaikan
2. Efisiensi/procesor utilization artinya proses yang dijalankan harus sesuai dengan urutan kerja proses
3. Waktu tanggap/Respon time
4. Waktu berjalan/waiting time
5. Waktu selesai/ turn around time

4.7.2 Strategi Penjadwalan

1. Strategi preemptive adalah proses yang sudah diberikan waktu maka waktu tersebut dapat diambil alih oleh proses lain sebelum proses itu selesai
2. Strategi non preemptive adalah proses yang sudah diberikan waktu maka tidak dapat diambil oleh proses lainsampai proses tersebut selesai

Pada dasarnya, scheduling adalah kegiatan memilih sebuah process dalam ready queue untuk dieksekusi CPU . Butuh algoritma agar pilihan tadi memberi hasil optimal sesuai krteria Beberapa algoritma scheduling:

1. First-Come, First-Served (FCFS) Scheduling Merupakan penjadwalan strategi non preemptive, Penjadwalan tidak berprioritas dan sederhana. Semua job proses diberikan waktu proses berdasarkan watu yang disediakan/kedatangan (dating pertama dilayani pertama).Proses dijalankan hingga selesai.

   Istilah istilah dalam perhitungan algoritma penjadwalan :

   a. JOB = Proses (P)
   b. Burst time = waktu yang dibutuhkan CPU untuk melayani P dengan satuan ms
   c. Arrival Time = waktu tiba job (waktu P yang tersedia)
   d. Turn Around Time =waktu proses job
   e. Waiting Time = waktu tunggu job
   f. Finish time = waktu selesai job
   g. Finishtime =AT+WT
   h. Average Waiting time = total wt/banyak job
2. Shortest-Job-First (SJF) Scheduling Shortest-Job-First (SJF) Scheduling Process dengan CPU burst time lebih kecil akan dieksekusi lebih dulu . SJF memberi hasil scheduling yang optimal Berdasarkan kapan keputusan scheduling dilakukan, SJF dibagi menjadi :

- Non-preemptive SJF scheduling
- Preemptive SJF scheduling

3. Round Robin (RR) Setiap proses mendapat sebuah alokasi unit waktu CPU (quantum waktu q) biasanya sekitar 10 -100 ms. Pemrosesan dimulai dari process yang pertama masuk • Setelah quantum waktu q berlalu, eksekusi berpindah ke process berikutnya sampai waktu quantum q nya berakhir, dan seterusnya. Process sebelumnya ditempatkan di akhir ready queue . Jika waktu wuantum = q dengan n process di ready queue, maka setiap process mendapat menunggu paling laa (n-1)q unit waktu. • Butuh timer untuk interrupt setiap quantum agar proses berikutnya bisa dijadwalkan . Performance – q besar = FIFO – q kecil = q must be large with respect to context switch, otherwise overhead is too high.
4. Multilevel Queues
5. Priority Scheduling

4.8 Deadlock

Didalam eksekusi sebuah proses terdapat deadlock. Penyebab utama terjadinya deadlock adalah terbatasnya sumber daya yang akan digunakan oleh banyak proses. Tiap proses berkompetisi untuk memperebutkan sumber daya yang ada. Jadi deadlock sangat berhubungan dengan sumber daya yang ada pada computer. Deadlocks adalah kebuntuan, artinya bahwa didalam system operasi terjadi suatu kondisi dimana sekumpulan proses tidak dapat berjalan kembali dilayani tidak adanya komunikasi antar proses.

Keadaan dimana 2 proses atau lebih saling menunggu meminta resources untuk waktu yang tidak terbatas. Analoginya seperti dijalan raya yang terjadi kemacetan parah. Deadlock adalah

efek samping dari sinkronisasi dimana satu variable digunakan oleh 2 proses atau lebih. Definisi deadlocks adalah sekumpulan proses yang terblok dimana setiap proses tersebut memegang sumber daya dan menunggu untuk mendapatkan sumber daya yang sama yang juga dipegang oleh proses lain.

4.8.1 Ilustrasi deadlocks

Misalkan proses A memerlukan sumber daya X dan mendapatkannya. Proses B membutuhkan sumber daya Y dan mendapatkannya. Selang beberapa saat proses A memerlukan sumber daya Y, dan proses B memerlukan sumber daya X. Pada situasi tersebut kedua proses tersebut terblok, dan menunggu terus menerus, keadaan tersebut disebut deadlock. Berikut ilustrasi deadlock:

1. Hanya satu mobil yang dapat menempati satu tempat persimpangan pada suatu waktu (Mutual exclusion)
2. Mobil boleh diam dipersimpangan selama ia menunggu untuk sampai kepersimpangan berikutnya (Hold and wait)
3. Mobil tidak dapat dipindahkan dari tempatnya arus lalu lintasnya , hanya dapat maju kedepan (Non preemption)
4. Sebuah kumpulan mobil dalam situasi deadlock termasuk juga mobil yang ada ditengah persimpangan (Circular wait)

4.8.2 Terdapat empat syarat untuk terjadinya deadlock

Berikut ini 4 syarat terjadinya deadlock:

1. Mutual exclusion adalah hanya satu proses pada satu saat dapat mengunakan sebuah sumber daya, tidak ada proses lain yang boleh menggunakan resources tersebut.
2. Hold and wait yaitu sebuah proses memegang (menggunakan) paling sedikit sebuah sumber daya yang sedang menunggu untuk meminta penambahan sumber daya yang dilakukan oleh proses-proses lainnya.
3. No preemption yaitu jika suatu proses meminta izin untuk mengakses resources sementara resources tidak tersedia, maka permintaan tidak dapat dibatalkan.
4. Circular wait yaitu sebuah rantai sirkular dari proses proses yang mana masing masing proses menunggu satu resources atau lebih yang digunakan oleh proses berikutnya dalam rantai tersebut.
    - Contoh circular wait Misalkan system memiliki 2 disk drive
    - P1 dan P2 masing masing menggunakan satu diks dirive A dan diskdrive B

- Pada saat tertentu P1 memerlukan disk drive B, p2 membutuhkan A padahal A sedang digunakan oleh P1 dan B sedang digunakan oleh P2, maka P1 menunggu B dan P2 menunggu A

4.8.3 Metode penanganan deadlock

Dalam penanganan deadlock ada hal – hal yang perlu dilakukan dalam penanganan deadlock. Berikut ini adalah langkah langkah dan metode penanganan deadlock:

1. Deadlock prevention adalah pencegahan terjadinya deadlock, metode ini berkaitan dengan pengkondisian system sehingga menghilangkan kemungkinan terjadinya deadlock. Pencegahan merupakan solusi yang bersih dipandang dari sisi tercegahnya deadlock, namun metode ini sering menghasilkan penggunaaan sumber daya yang buruk. Pencegahan deadlock merupakan metode banyak dipakai.
2. Deadlock avoidance yaitu menghindari terjadinya deadlock,tujuan metode ini adalah menghindarkan kondisi2 yang paling mungkin menimbulkan deadlock agar memperoleh utilisasi sumber daya yang lebih baik, penghindaran bukan berarrti menghilangkan semua kemungkinan terjadinya deadlock. Secara teoritis deadlock dimungkinkan. Sistem operasi memeriksa semua permintaan sumber daya secara hati hati. jika system operasi mengetahui alokasi sumber daya mengakibatkan deadlock, system operasi menolak pengaksesan itu sehingga dapat menghindari terjadinya deadlock.
3. Deadlock Detection dan recovery ini adalah metode untuk mendeteksi dan memulihkan system dari deadlock. Metode ini digunakan untuk system yang mengizinkan terjadinya deadlock tujuannya untuk memeriksa apakah telah terjadi deadlock dan menentukan proses dan sumber daya yang terlibat deadlock secara presisi, seteleh dapat ditentukan system dipulihkan dari deadlock dengan metode pemulihan. Metode pemulihan dari deadlock berupaya untuk menghilangkan deadlock dari system sehingga system dari deadlock sehingga proses dapat beroperasi kembali dan bebas dari deadlock. Proses proses yang terlibat deadlock mungkin dapat menyelesaikan eksekusi dan membebaskan sumber daya-sumber daya nya.

Havender J.W menyarankan strategi strategi berikut untuk meniadakan syarat-syarat terjadinya deadlock, yaitu Tiap proses harus meminta sumber daya yang diperlukan sekaligus dan tidak berlanjut sampai semuanya diberikan. Jika proses telah sedang memegang sumber daya tertentu, untuk permintaan berikutnya proses harus melepas dulu sumber daya yang dipegangnya. Beri pengurutan linear terhadap tipe-tipe sumber daya pada semua proses, yaitu jika proses telah dialokasikan suatu tipe sumber daya, proses hanya boleh meminta sumber daya-sumber daya tipe pada urutan yang berikutnya.

4.8.3.1 Deadlock prevention

1. Mutual Exclusion

Bagaimana cara meniadakan mutual exclution. Deadlock disebab kan terdapat pengaksesan eksklusif terhadap sumber daya, jika tidak ada sumber daya ekslusif untuk satu proses tunggal maka tidak akan pernah dijumpai deadlock. Cara yang dapat ditempuh untuk mengakali pengaksesan eksklusif yaitu dengan cara melakukan spooling perangkat perangkat yang harus didedikasikan kesuatu proses. Pengaksesan sumberdaya seolah olah tidak eksklusif padahal eksklusif hanya dengan spooling berarti permintaan-permintaan itu diantrikan dihardisk. Job job di antrian spooler akan dilayani satu persatu. Tapi ada beberapa masalah pada metode ini:

a. Tidak setiap proses sumber daya eksklusif dapat dispooling misalnya table proses

b. Kompetisi pada ruang hardisk spooling dapat menuntun ke deadlock

Abstraksi ini berrati kembali terjadi kondisi yang mengharuskan exclusion namun mutual exclusion menjadi dilevel bawah yaitu level suatu lokasi memori bukan lagi suatu perangkat. jadi mutual exclusion tidak dapat dihindari akan tetapi hanya dapat diperkecil lama waktu berlangsungnya

2. Meniadakan Hold and Wait

Mengalokasikan semua sumber daya atau tidak sama sekali. Pada mekanisme ini semua proses hanya dilayani permintaannya jika sumber daya yang diperlukan tersedia, teknik ini berbasis pada kaidah memperoleh semua atau tidak sama sekali. Jika semua sumber daya tersedia proses dialokasikan. semua sumber daya dperlukan dan berjalan sampai selesai jika tidak tersedia sedikitnya satu sumber daya maka proses harus menunggu sampai sumber daya yang diperlukan tersedia untuk dialokasikan padanya. Namun permasalahannya :

- sulit untuk mengetahui lebih dulu sumber daya yang diperlukan suatu proses
- Cara ini dapat mengakibatkan penggunaan sumber daya yang tidak efisien

3. Hold and release ,setiap kali terjadi permintaan sumber daya maka proses harus melepas sumber daya lain yang digunakan , jadi pada satu saat hanya ada satu sumber daya yang dialokasikn untuk satu proses, tapi permasalahannya hal ini tidak mungkin terjadi sebab terdapat proses yang mensyaratkan harus memegang beberapa sumber daya sekaligus untuk melanjutkan ekeskusinyanya.
4. Meniadakan No Preemption
   Mencegah proses-proses lain menunggu, seluruh proses menjadi preemption agar tidak ada kejadian tunggu menunggu namun permasalahannya tidak mungkn meniadakan sama sekali non preemption ini, misalnya proses A menulis ke printer tiba tiba dihentikan proses B yang juga ingin menulis di printer yang sama, jika kondisi preemption ini dimungkinkan maka proses ini akan mencetak secara tidak benar. Preemption tetap diperlukan untuk beberapa kasus.
5. Meniadakan Circular wait

Proses ini dapat ditiadakan apabila:

1. Proses hanya dibolehkan menggengam satu sumber daya pada satu saat, ketika suatu proses meminta sumber daya proses lain maka ia harus melepas sumber daya lain yang sedang digunakan
2. Penomoran global semua sumber daya, jadi proses hanya dapat meminta proses kapanpun yang diinginkan tetapi syaratnya permintaan tersebut harus dilakukan secara terurut secara numerik untuk tidak menimbuklan terjadinya siklus akan tetapi masalahnya tidak ada sumber daya yang memuaskan semua pihak dan tergantung pada system operasinya.

4.8.3.2 Penanganan Deadlock avoidance

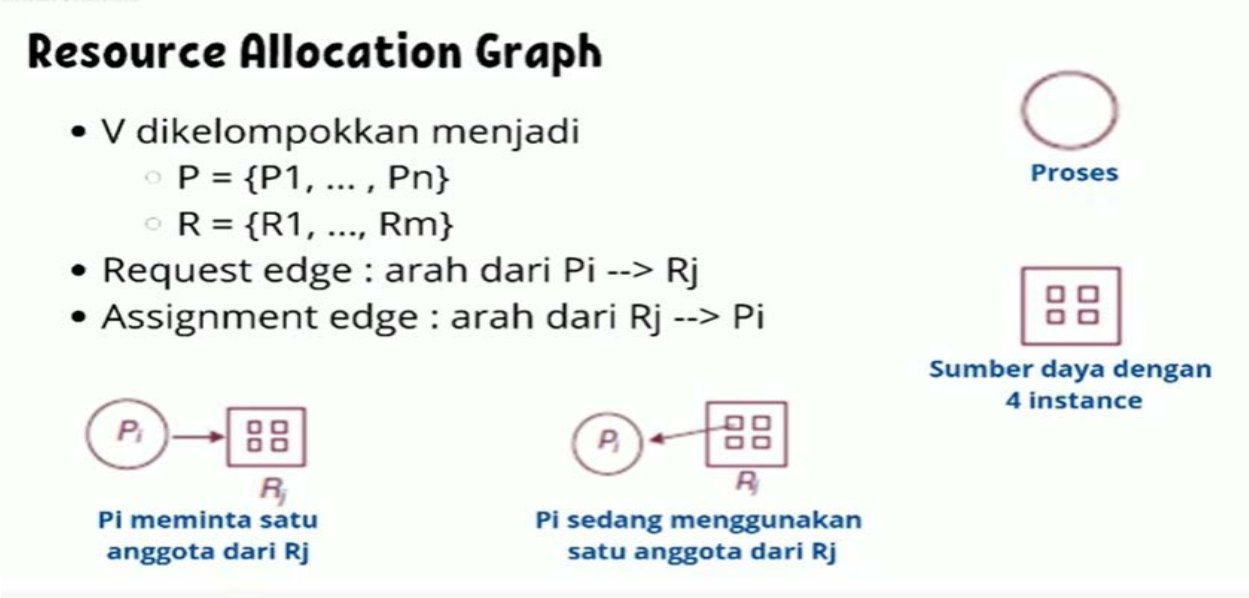
Secara prinsip deadlock dapat dihindari dengan memastikan bahwa system selalu dalam status safe. Cara ini memastikan agar setiap alokasi sumber daya tidak membuat sebuah system menjadi tidak aman atau unsafe. Mekanisnya adalah :

1. Setiap proses menyatakan jumlah maximum setiap sumber daya yang dibutuhkan
2. Algoritma deadlock avoidance yang secara dinamis memeriksa status alokasi sumber daya untuk memastikan bahwa tidak akan terjadinya circular wait
3. Status Alokasi sumber daya dinyatakan dengan jumlah sumber daya yang tersedia dan sedang dialokasikan dan jumlah maksimum permintaan dari setiap proses.

Terdapat beberapa algoritma pada deadlock avoidance

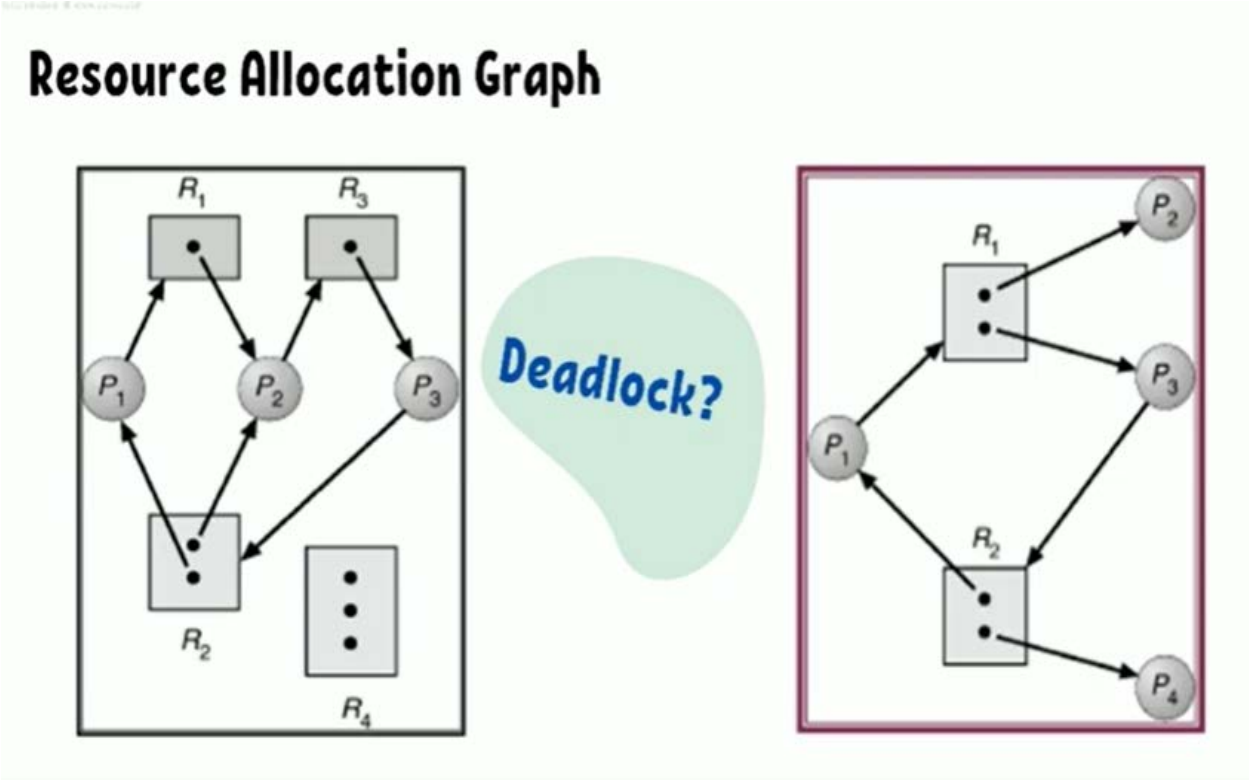
1. Resource Allocation Graph

   Kumpulan titik dan garis disebut Graph atau dikenal juga dengan vertex (titik) dan edge (garis).



Gambar 1.1 Allocation Graph

Contohn dari algoritma allocaton graph



Gambar 1. 2 Gambar perbandingan proses yang mengalami deadlock yang mana?

Keterangan :

- Gambar Kiri, terjadi deadlock karena P3 meminta resourse di R2 padahal R2 sudah dialokasikan ke P1 dan P2 maka dari itu siklus akan terus menerus memutar dan menimbulkan deadlock. Dan jika ada lebih dari satu instance akan terjadi deadlock karena tidak ada proses yang dapat melepaskan resource untuk dialokasikan ke proses lainnya atau jika hanya satu instance per resource maka pasti terjadi deadlock. Pada kasus ini, terdapat dua siklus pada sistem, yaitu :
- P1 -> R1 -> P2 -> R3 -> P3 -> R2 -> P1
- P2 -> R3 -> P3 -> R2 -> P2

  Graf tersebut memiliki dua perputaran , yaitu:

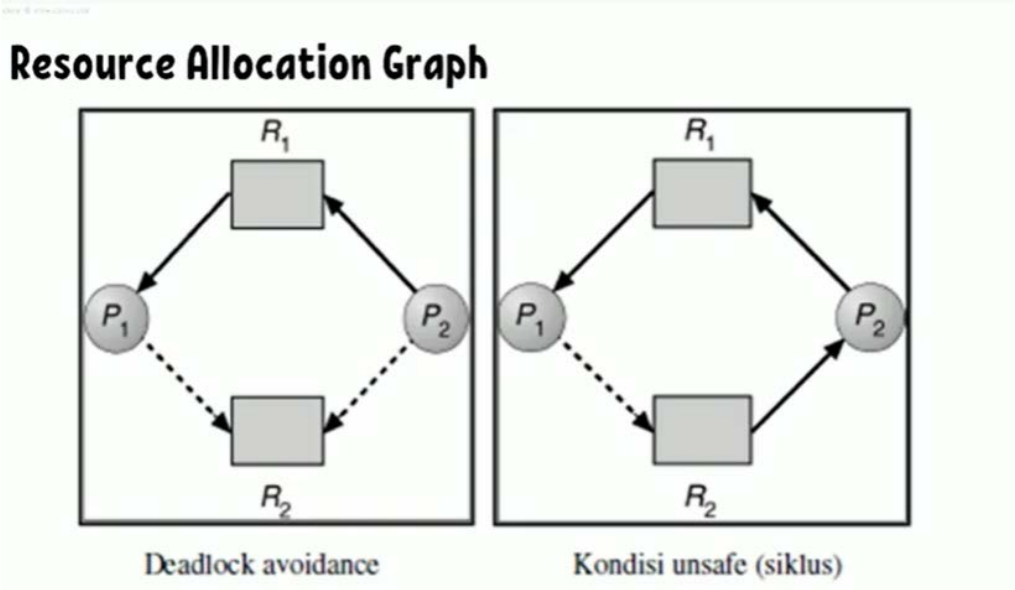
  R2 -> P1 -> R1 -> P2 -> R3 -> P3 -> R2

  R2 -> P2 -> R3 -> P3 -> R2

- Gambar Kanan, tidak terjadi deadlock karena proses P4 kemungkinan melepaskan sumber daya R2 kemudian sumber daya tersebut dialokasikan untuk P3 dan akan menghapus siklus. Adanya siklus merupakan syarat perlu namun bukan merupakan syarat cukup terjadinya deadlock. Pada kasus ini terdapat 1 siklus pada sistem, yaitu : P1 -> R1 -> P3 -> R2 -> P1

2. Algoritma Claim Resource Allocation Graph

   Pada gambar (1.3) dibawah ini tidak terdapat cycle karena request edge menggunakan claim edge yang artinya meskipun kedua proses tersebut membutuhkan sumber daya yang sama, namun tidak harus segera dipenuhi, sehingga pengalokasian sumber daya pada sistem ini dalam keadaan aman (safe state). Sebaliknya jika ditemukan cycle / pada gambar (b), maka akan membuat sistem ini dalam keadaan tidak aman (unsafe state), maka Pi harus menunggu permintaannya dipenuhi. Berikut gambar yang dapat dilihat:



Gambar 3.3 Resource Allocation Graph

3. Safe state-algoritma banker

   Status safe yaitu system dapat mengalokasikan resource untuk tiap proses (sampai #maks) dalam urutan yang tepat tanpa terjadinya deadlock. Safe artinya tidak ada deadlock. Unsafe artinya kemungkinan terjadinya deadlock.

   Avoidance meyakinkan bahwa system tidak pernah memasuki keadaan status unsafe. Misalkan ada n proses dalam system dan m tipe resources , terdapat data struktur sebagai berikut:

   a. Available (resource yang terjadi pada suatu saat) suatu vector dengan panjang m
   b. Max adalah matriks n x m yang mendefinisikan maksimum permintaan (request) untuk tiap tiap proses.
   c. Allocation yaitu matrik n x m ayng mendefiniskan jumlah resources untuk tiap tiap tipe yang sedang dialokasikan untuk tisp proses.
   d. Need yaitu matriks n x m yang menunjukan sisa resource yang dibutuhkan untuk tiap proses.
   e. Kelemahan algoritma banker:

   - Tidak semua proses mengetahui max resource
   - Jumlah proses tidak tetap
   - Beberapa resource terkadang bisa diambil dari system sewaktu waktu . Sehingga meskipun kelihatannya ada , namun kenyataan nya tidak ada
   - Algoritma menghendaki membrikan semua permintaan hingga waktu yang berhingga
   - Proses harus berjalan terpisah (indenpenden) sehingga urutan eksekusi dibatasi oleh kebutuhan sinkronisasi proses.
   - Menghendaki client mengembalikan resource setelah batas tertentu.

4.8.3.3 Deadlock detection and recovery

Detection adalah teknik untuk menentukan apakah deadlock terjadi serta mengidentifikasi proses proses atau sumber daya yang terlibat deadlock. Umumnya algoritma deteksi adalah menentukan keberadaan atau menunggu sircular atau sircular wait.penggunaan algoritma deteksi deadlock mengakibatkan overhead pada saat berjalan karena secara periodic untuk mendeteksi sircular wait. Pada resource pada single instance biasanya dapat dilihat apakah pada allocation grap pada grap terdapat sircular wait sedangkan pada multiple instance dapat dilihat apakah jumlah request lebih besar dari jumlah sumber daya yang tersedia

Recovery proses deadlock dapat dijabarkan sebagai berikut:

1. Recovery adalah pemulihan deadlock.
2. Pendekatan yang dilakukan untuk pemulihan deadlock

- Mengagalkan semua proses yang terlibat deadlock

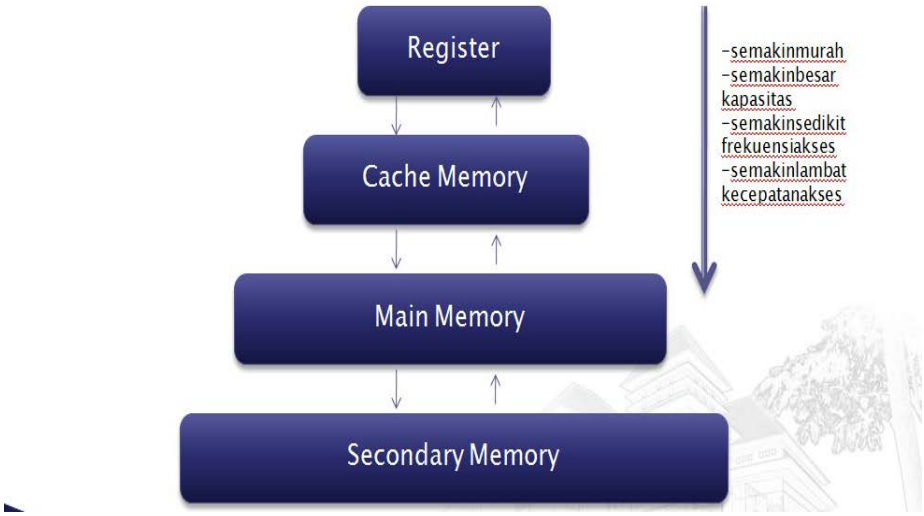
- Backup semua proses yang deadlock kemudian rollback dan restart semua proses tersebut
- Menggagalkan semua proses deadlock secara berturut-turut sehingga tidak ada deadlock
- Secara berurutan mengpreemptive kan sumber daya sehingga tidak ada deadlock lagi

Kriteria penyingkiran proses:

- Waktu proses yang telah dijalankan paling kecil
- Jumlah baris keluaran yang diproduksi paling kecil
- Mempunyai estimasi sisa waktu eksekusi terbesar
- Jumlah total sumber daya terkecil setelah dialokasikan
- Prioritas terendah

5. Manajemen Memory

Manajemen memori berkaitan dengan aktivitas pengelolaan penggunaan memori pada saat computer aktif dan menjalankan proses-proses. Berikut ini gambar jenis jenis memori dan dapat dilihat dari berbagai sisi.



Gambar 5.1 Jenis jenis memori dilihat dari berbagai sisi

Dari gambar diatas dapat dilihat dari berbagai sisi, dari sisi harga, sisi kapasitas serta dari sisi frekuensi kecepatan akses.

5.1 PENGALAMATAN MEMORI

Prosesor mengakses data atau kode instruksi menggunakan referensi alamat I/O dan alamat memori utama. Kompilator, mengubah source program yang ditulis programmer menjadi file yang berisi kode instruksi program yang dapat dijalankan oleh prosesor. Untuk melakukan fungsinya kompilator mengacu pada metode pengalamatan memori.

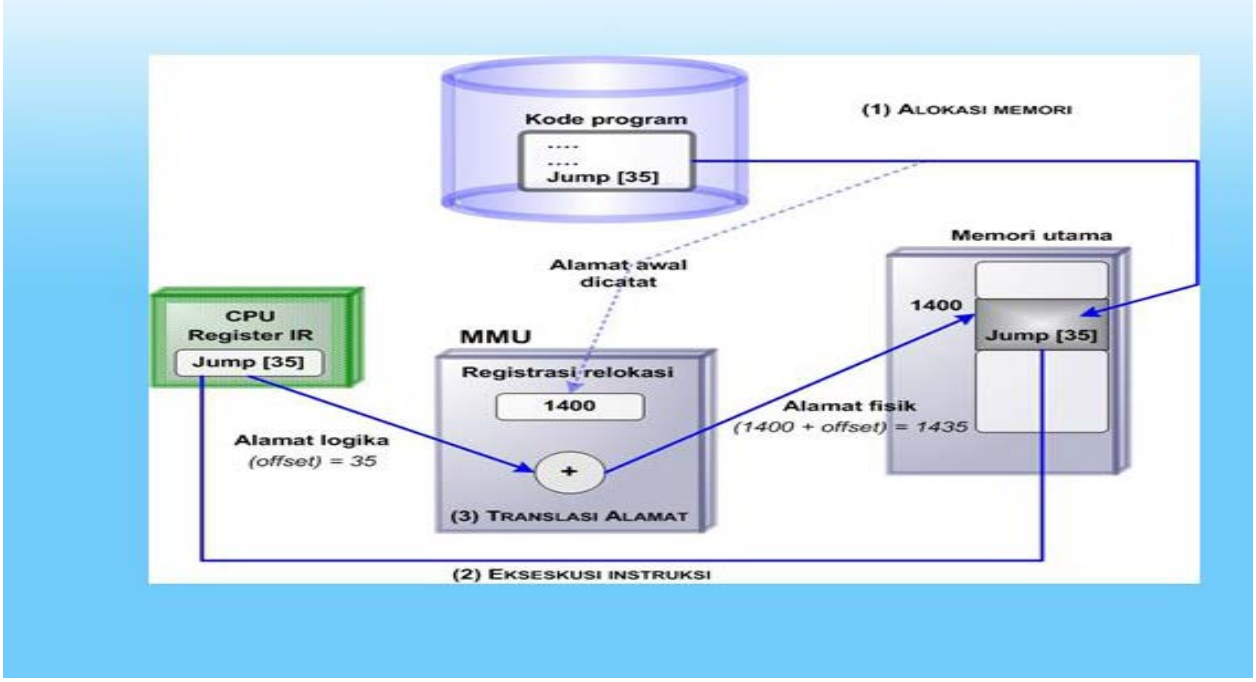
Ada 3 cara dalam pengalamatan memory

1. Pengalamatan secara fisik

Alamat yang ditulis pada kode instruksi program hasil kompilasi merupakan alamat fisik memori utama yang sesungguhnya. Namun konsekuensinya adalah Pada saat penyalinan image proses ke memory utama maka kode instruksi dan data program harus disalin pada posisi yang sesuai dengan referensi tersebut. Pada saat eksekusi prosesor akan memproses alamat pada kode instruksi program secara langsung tanpa melakukan translasi alamat memori.Tidak membutuhkan translasi alamat memori.

2. Pengalamatan secara relative

Digunakan pada system yang menggunakan alokasi memori berurut, dimana keseluruhan image proses harus terletak pada satu area memori utuh. Alamat pada kode instruksi program merupakan alamat relative atau offside terhadap posisi awal program. Pada saat image proses dari program tersebut disalin ke memory utama, alamat awal memorinya dicatat ke alamat register suatu alokasi. Pada saat eksekusi pengaksesan alamat akan ditranslasi dengan menjumlahkan alamat referensi pada instruksi dengan isi register alokasi untuk mendapatkan alamat fisik memori yang akan benar benar diakses. Membutuhkan translasi alamat memori pada saat eksekusi. Translasi alamat memori ->MMU (Management Memory Unit)



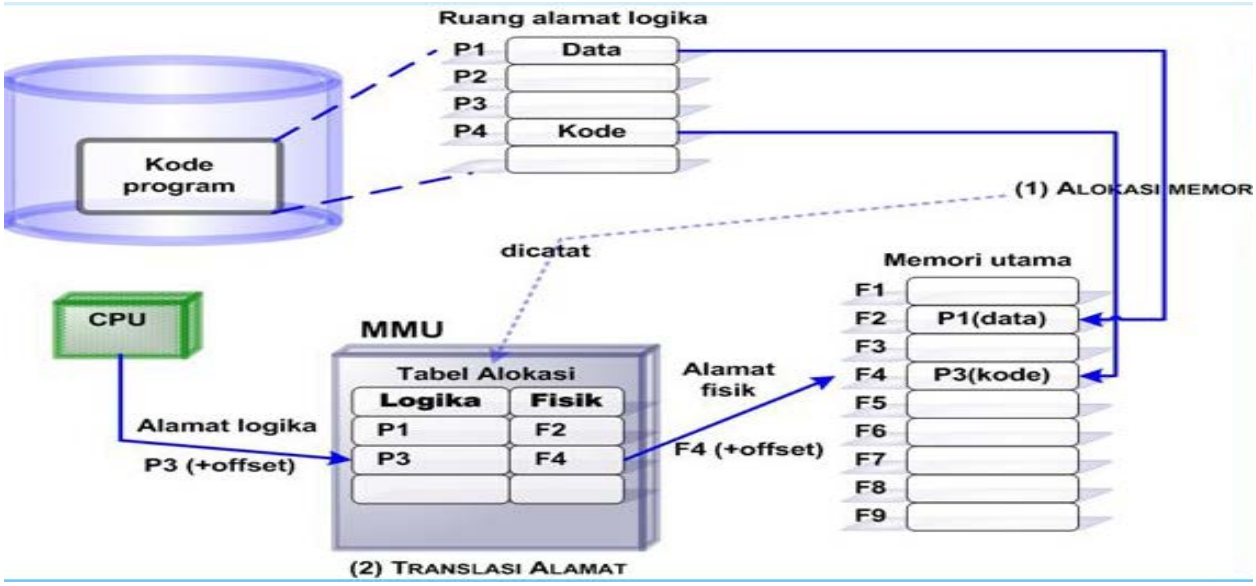
Gambar 5.2 Proses Alokasi Memori Dan Translasi Alamat Pada Pengalamatan Memori Relatif

3. Pengalamatan logika

Pengalamatn secara logika memerlukan translasi alamat yang dilakukan saat eksekusi proses . Alamat yang ada pada kode program merupakan suatu alamat logika yang masih perlu diterjemahkan atau ketranslasikan ke alamat fisik memori utama pada saat eksekusi. Umumnya translasi alamat utuk pengalamatan logika terjadi pada saat eksekusi.

- Kelebihan nya adalah:

1. Relokasi program dapat dilakukan secara fleksibel, bahkan ruang atau kapasitas ruang alamat logika program dapat lebih besar dari kapasitas fisik memori utama. Misalnya program dapat menggunakan ruang alamat logika sebesar 2 GB sedangkan memori utama fisik dimana program tersebut dijalankan sebenarnya hanya memilkik kapasitas 256MB
2. Image proses dapat dialokasikan secara parsial dan tersebar pada memori utama (alokasi memori utama dapat dilakukan dengan paging atau segmentasi). Pengalamatan logika prinsip kerjanya, ruang alamat pada kode program menggunakan suatu peta alamat logika tersendiri.



Gambar 3.4 Proses alokasi memori dan translasi alamat pada pengalamatn logika

Proses alokasi memori dan translasi alamat pada pengalamatn logika:

- Ruang alamat logika program dalam partisi statis yang berukuran sama(page)
- Segmentasi membagi ruang logika alamat logikaprogram dalam fragmen yang berukuran berbeda-beda dan pemartisian memori utama bersifat dinamis dengan ukuran yang bervariasi(segmen)
- Hal ini mempengaruhi:Bagaimana memori utama di partisi dan dialokasikan ke aplikasi, Informasi apa yang perlu dicatat di tabel alokasi proses saat terjadi alokasi memori,dan Proses translasi alamat.

5.2 Address Binding

Aktivitas translasi alamat disebut dengan address binding. Translasi alamat dapat terjadi pada saat :

a. Compile time (Jika lokasi instruksi di memori sudah diketahui sebelumnya, maka pada saat compile, alamat absolut kode program sudah bisa ditentukan. Pada saat loading, kode program harus diletakkan pada lokasi memori yang sudah ditentukan sebelumnya.)

b. Loading Time (Jika lokasi kode program di memori belum diketahui sebelumnya, maka pada saat compile time, alamat yang bisa direlokasi (mis alamat relatif) diberikan kepada kode program. Alamat fisik diberikan/diikatkan kepada kode saat program diload ke memori. Proses binding harus diulangi jika kode program dipindahkan ke bagian lain dari memori.)

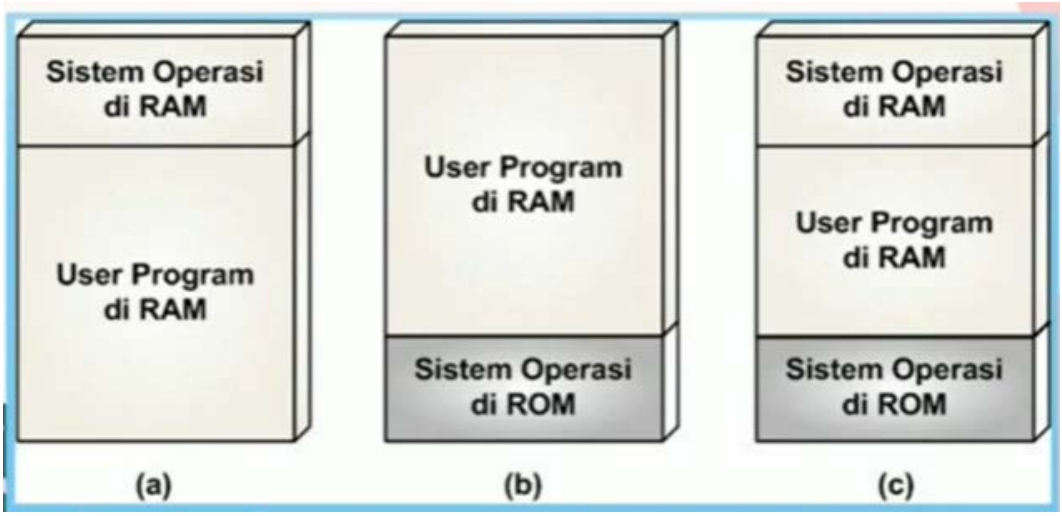
c. Excecution Time (Alamat logis dapat diterjemahkan menjadi alamat fisik saat program berjalan). Pada kasus swapping, kode program bisa dipindahkan ke lokasi memori yang

berlainan selama siklus hidupanya. Dalam kasus ini, alamat absolut diberikan kepada kode program pada saat kode dieksekusi. Hal ini bisa diwujudkan dengan bantuan hardware khusus seperti MMU**.**

5.3 Manajemen memori pada sistem monoprogramming

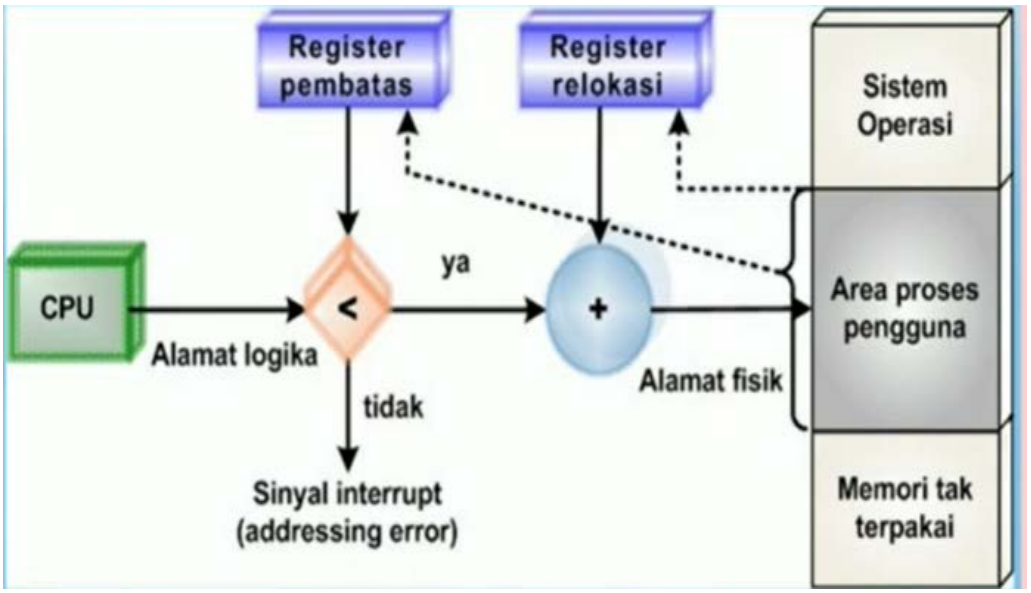
Hanya ada satu proses pada satu saat dan menggunakan seluruh area memori pengguna. Program dimuatkan seluruhnya ke memori disk/tape. Pada saat eksekusi program mengambil kendali seluruh sumber daya computer**.** Alokasi memori dilakukan secara berurutan , yang artinya image proses dari program harus menempati area memori utuh.

Pada sistem monoprogramming, bagian memori utama yang tidak dipakai untuk sistem operasi atau device driver dialokasikan semuanya kepada proses pengguna.



Gambar 5.4 Manajemen memori pada sistem monoprogramming

Pada sistem monoprogramming, kode instruksi dan data dari sistem operasi butuh dilindungi dari akses langsung ataupun modifikasi oleh proses aplikasi pengguna.Proteksi dapat diimplementasikan dengan bantuan register relokasi dan register pembatas (berisi jangkauan atau lebar ruang alamat logika dari proses aplikasi pengguna).

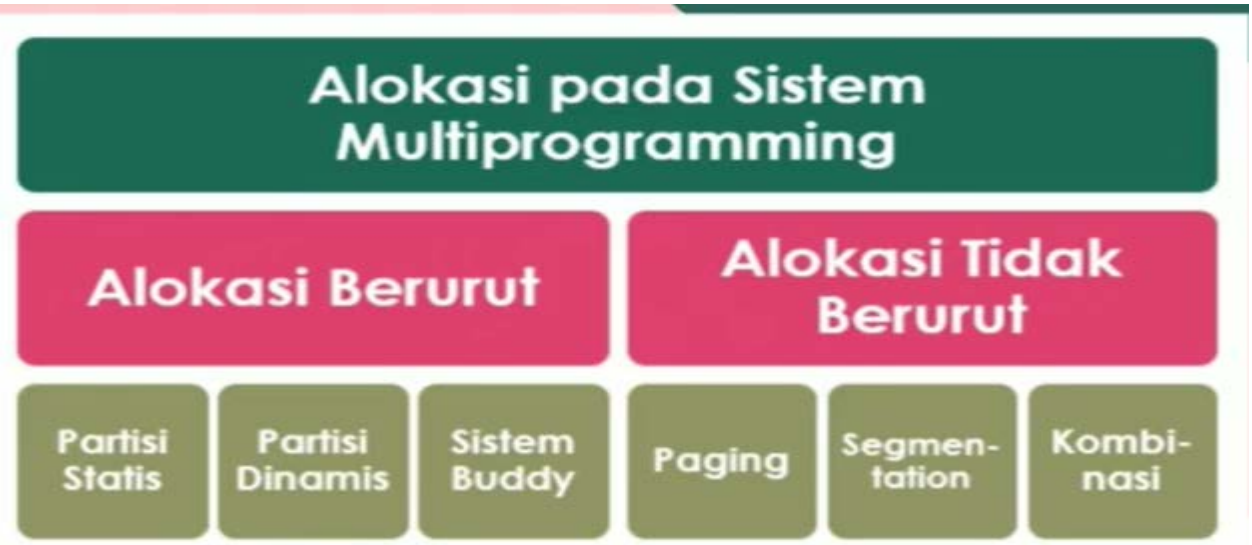


Gambar 5.4 Model Proteksi memori monoprogramming

5.4 Manajemen Memori Pada Sistem Multiprogramming

Ciri-ciri manajemen memori pada sistem multiprogramming antara lain:

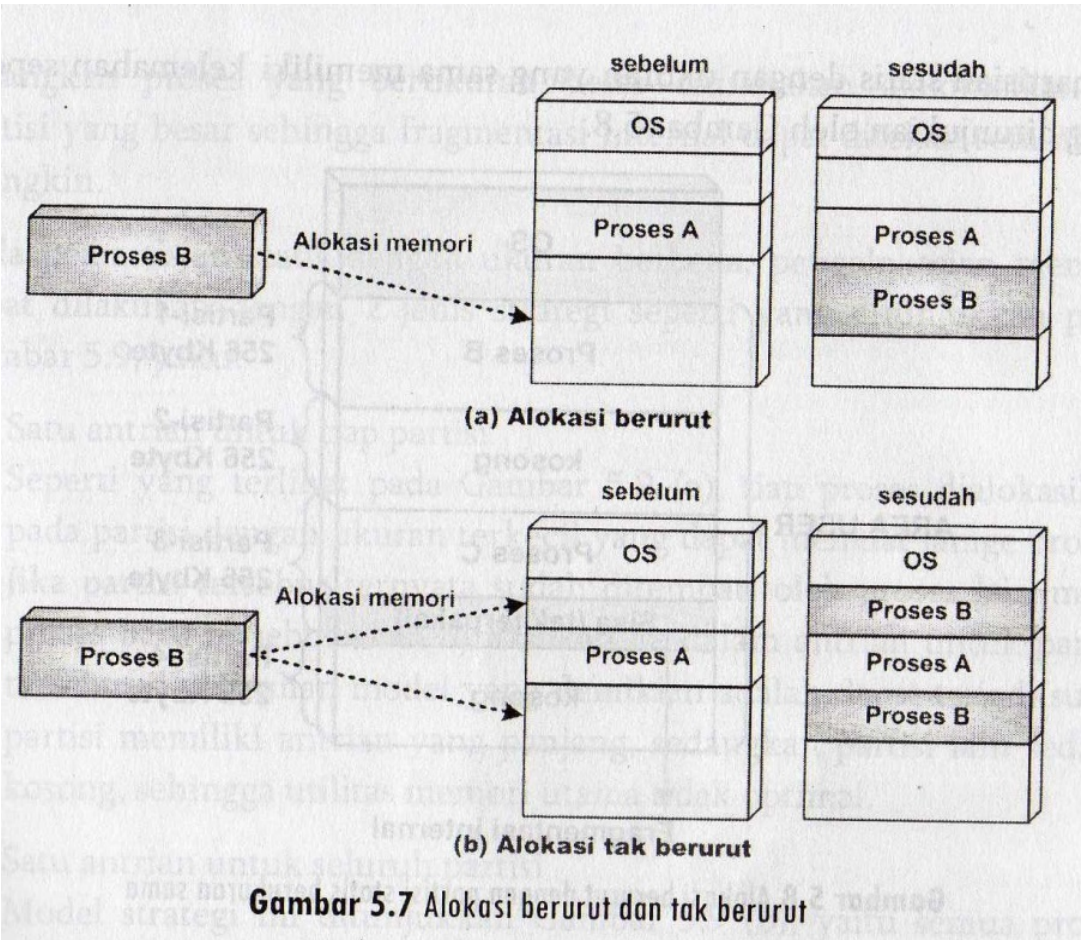
- Ada sejumlah proses yang menempati memori utama pada setiap saat. Image proses dari program dapat dimuat seluruhnya atau sebagian saja ke memori utama.
- Alokasi memori ke proses bisa secara berurutan ataupun tidak berurutan.
- Dimungkinkan seluruh atau sebagian image proses berpindah lokasi memori utama selama eksekusinya.
- Dimungkinkan suatu lokasi memori bisa diakses oleh beberapa proses (sharing)



Gambar 5. 5 Alokasi pada system Multiprogramming

Perbedaan alokasi berurut dan tidak berurut:

Pada sistem multiprogramming, metode alokasi memori ke proses-proses dapat dikategorikan sebagai berikut :Alokasi berurut mengalokasikan suatu proses secara utuh ke suatu bagian memori yang berurut. Alokasi tak berurut bisa membagi suatu proses menjadi beberapa bagian kecil dan mengalokasikan setiap bagian ke lokasi memori yang berbeda.



Gambar 5.7 Alokasi berurut dan tak berurut

Ciri-ciri alokasi berurut dengan partisi statis:

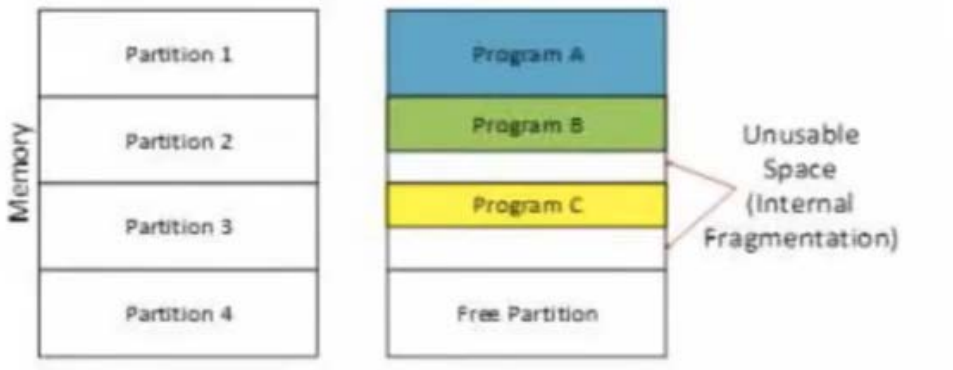
- Memori dibagi menjadi partisi-partisi dengan ukuran yang tetap.
- Satu proses hanya menggunakan satu partisi. Jika proses sudah selesai, partisi tersebut bisa digunakan oleh proses lain.
- Membutuhkan pengelolaan informasi mengenai partisi-partisi yang kosong (bisa dialokasikan).
- Alokasi berurut dengan partisi statis dapat dibedakan atas: Partisi statis berukuran sama.Besarnya tiap partisi adalah sama besarnya

- Partisi statis berukuran tidak sama. Ukuran tiap partisi berbeda-beda

Gambar 5.5 Contoh pemartisian berukuran sama

Kelemahan pemartisian statis dengan ukuran yg sama:

- Proses yang ukurannya lebih besar dari ukuran partisi tidak dapat dialokasikan, sehingga dibutuhkan overlay.
- Jika ukuran proses lebih kecil dibanding ukuran partisi yg dialokasikan, maka akan terjadi fragmentasi internal, yaitu terjadi sisa ruang di dalam partisi yang tidak terpakai.

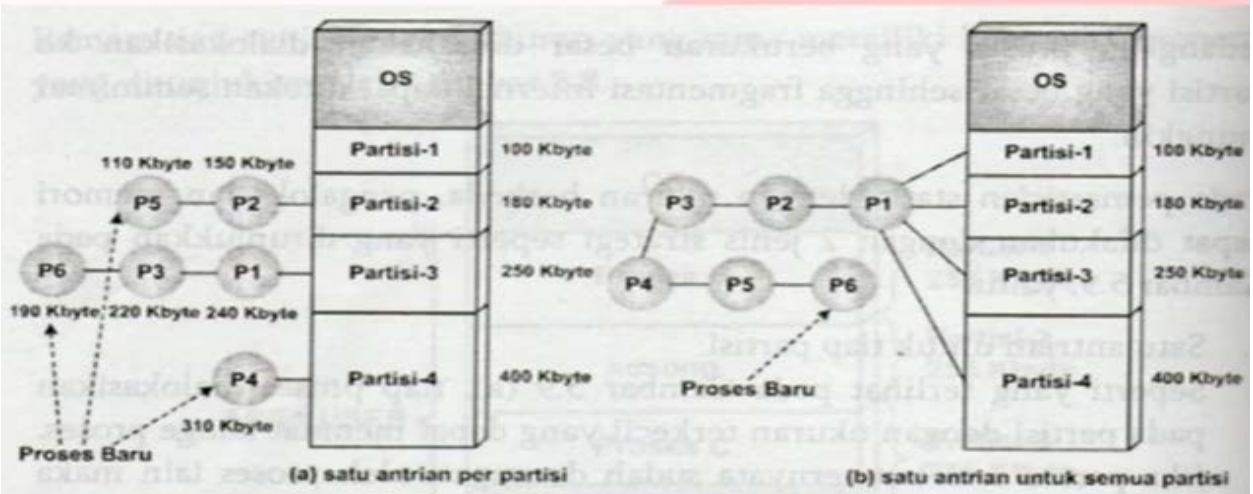


Gambar 5.6 Fragementasi Internal.

Pengalokasian berurut dengan partisi dinamis (ukuran tidak sama). Pemartisian statis dengan ukuran berbeda mengantisipasi beragamnya ukuran image proses, proses berukuran kecil diusahakan dialokasi ke partisi yang kecil, sedangkan proses besar diusahakan dialokasikan ke partisi yang besar, sehingga fragmentasi internal bisa ditekan seminim mungkin. Pada pemartisian statis dgn ukuran berbeda, pengalokasian memori bisa dilakukan dengan 2 jenis strategi:

1. Satu antrian untuk tiap partisi, Proses diantrikan pada partisi dengan ukuran terkecil yang bisa memuat proses (namun bisa terjadi antrian yg panjang)

2. Satu antrian untuk seluruh partisi, Semua proses berada dlm satu antrian yang sama. Proses dialokasikan ke partisi terkecil yg tersedia yg bisa memuat proses tersebut pada saat dijadwalkan. (namun bisa terjadi suatu proses kecil terpaksa dialokasikan ke partisi besar yang tersisa)



Gambar 5.9 Alokasi berurut dengan partisi statis berukuran berbeda dengan (a) satu antrian per partisi (b) satu antrian untuk semua partisi

- Pengalokasian berurut dengan partisi dinamis. Ciri-ciri:

a. Pada kondisi awal, memori tidak dibagi menjadi partisi-partisi.

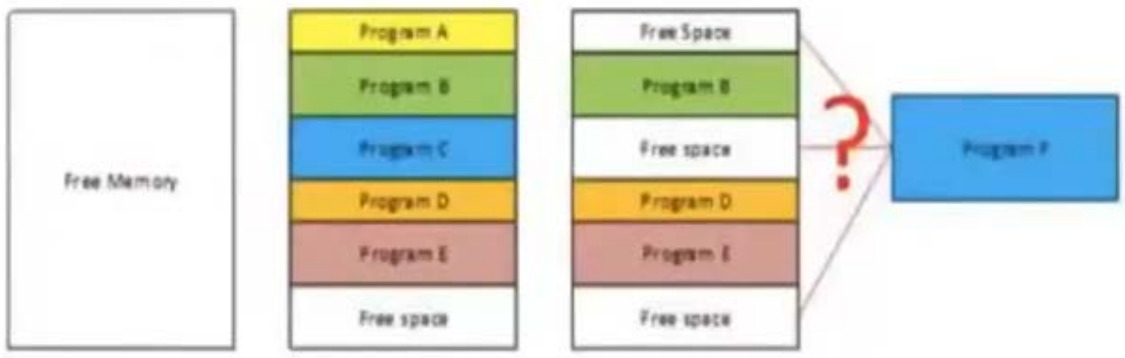
b. Pemartisian dilakukan pada saat image proses akan disalin ke memori utama.

c. Ukuran partisi yang dialokasikan akan disesuaikan dengan ukuran image proses.

d. Partisi akan dibebaskan jika program sudah selesai

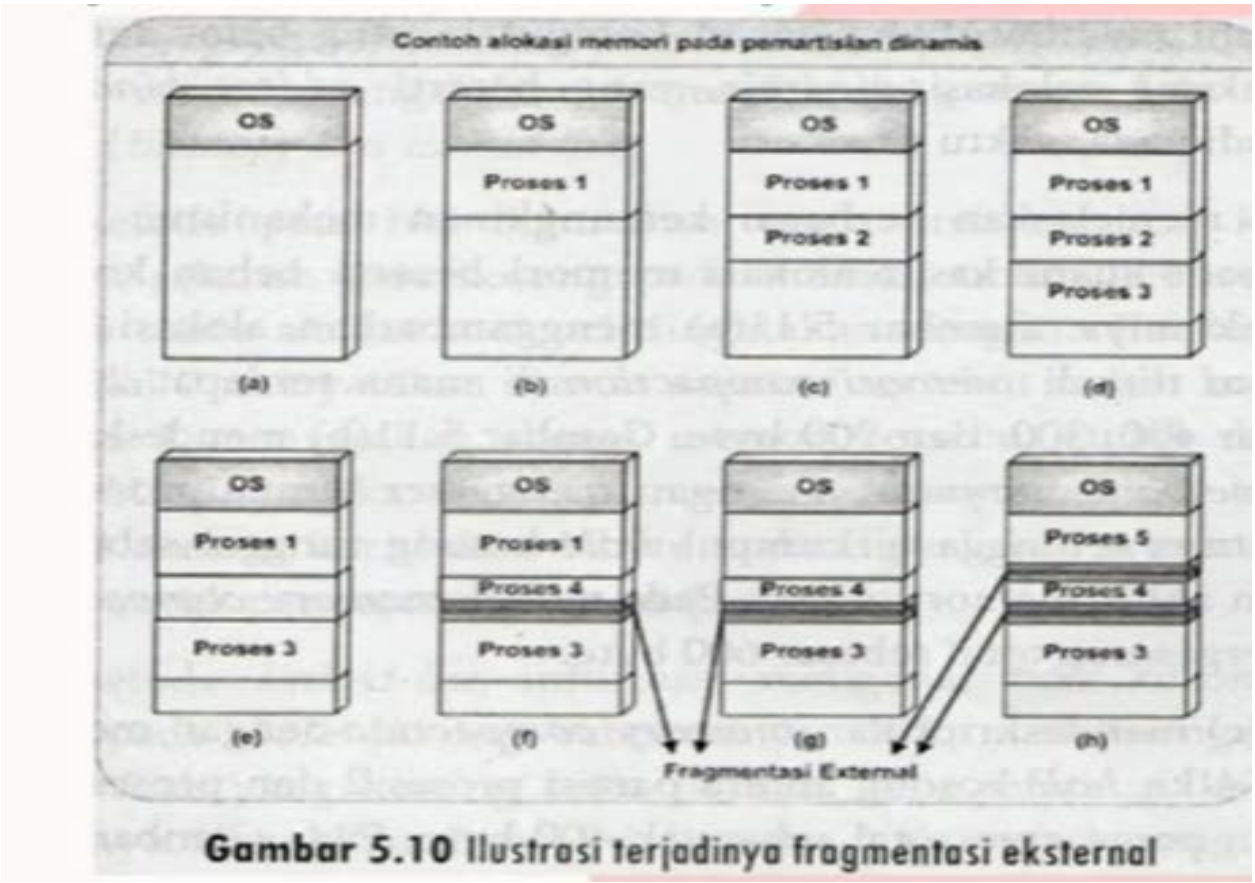
e. Keuntungan : tidak terjadi fragmentasi internal alokasi memori disesuaikan dengan besarnya image proses.

Pengalokasian berurut dengan partisi dinamis dilakukan dengan mencari hole yaitu suatu ruang memori utama yang kosong, yang cukup besar untuk menampung image proses. Hole sisa kadang kala terlalu kecil untuk dapat dialokasikan ke proses lainnya sehingga tidak bisa digunakan lagi (fragmentasi eksternal).



Gambar 5.7 Fragmentasi Eksternal

Berikut ini ilustrasi terjadinya fragmentasi Eksternal.



Gambar 5.10 Ilustrasi terjadinya fragmentasi eksternal

Pengalokasian berurut dengan partisi dinamis. Salah satu cara untuk mengatasi fragmentasi eksternal adalah melakukan memory compaction Yaitu: menggeser image proses-proses yang ada di memori sehingga hole terkumpul di satu tempat saja.

Kelemahan partisi dinamis:

- Proses alokasi dan dealokasi menjadi lebih rumit
- Perlu pengelolaan informasi area memori yang masih kosong.
- Ada 2 metode pengelolaan memori kosong:

1. Peta bit (bitmap) Menggunakan area memori khusus untuk

mencatat seluruh area kosong pada memori utama.

Memakai nilai 0 dan 1

Nilai 0 → alamat memori tersebut masih kosong

Nilai 1 → alamat memori tersebut sudah terisi

2. Linked list, Informasi mengenai hole kosong berikutnya dicatat pada hole kosong sebelumnya.Tidak diperlukan area memori khusus. Karena seluruh informasi tercatat di area memori kosong itu sendiri sehingga menghemat kapasitas memori utama.

Pengelolaan informasi kosong membutuhkan algoritma : Diperlukan algoritma untuk menentukan hole mana yang akan dialokasikan ke suatu proses. Algoritma Best-fiT, Algoritma First-fit, Algoritma Next-fit, Algoritma Worst-fit.

a. Algoritma Best-fit, Mencari memori blok yang paling kecil yang dapat menampung image proses. Memerlukan waktu lama karena harus searching seluruh blok memori utamaFragmentasi eksternal dapat ditekan sekecil mungkin.

b. Algoritma First-fit, Mencari memori kosong dari alamat awal sampai menemukan blok yang dapat menampung image proses. Sederhana dan cepat, Algoritma Next-fit hampir sama dengan First-fit. Bedanya: proses searching dimulai dari alamat alokasi terakhir

c. Algoritma worst-fit, Mencari hole yang paling besar di seluruh area memori utama. Tujuannya: hole sisa yang tercipta setelah alokasi masih cukup besar untuk dialokasikan ke proses lainnya.

5.4 Pengalokasian berurut dengan sistem buddy

Berupa pemartisian secara dinamis, Ciri khusus adalah partisi yang terbentuk senantiasa berukuran besar sebesar bilangan 2n (pangkat) 2,4,8,16…..256,512,1024(1Mb)

Alokasi memori pada sistem buddy:

- Menentukan ukuran partisi.
- Ditentukan ukuran partisi untuk menampung image proses yaitu ukuran bilangan pangkat 2 terkecil
- Misal : ukuran image proses = 12kb maka ukuran partisi yang bisa digunakan adalah 16kb.

5.5 Pengalokasian tak berurut dengan sistem paging

Pada model pengalokasian tak berurut, bagian-bagian dari image proses dapat diletakkan secara terpisah di memori utama.Pada sistem paging memerlukan pengalamatan logika khusus yang membagi menjadi blok-blok dengan ukuran sama yang disebut page, Pada sistem paging , perlu adanya translasi alamat ke memori fisik yang dipartisi secara statis yang disebut frame, yang ukurannya sama dengan page pada ruang alamat logika.

Konsep dasar alokasi memori :

- Memori utama dibagi menjadi frame-frame kecil berukuran sama dan diberi nomor frame sebagai referensi.
- Ruang alamat logika proses dibagi menjadi page-page seukuran frame
- Loading time: page-page image proses diletakkan pada frame-frame kosong dan dicatat pada page table.

Proteksi Memori pada sistem paging, Berfungsi menghindari pengaksesan memori secara ilegal.misal: pengaksesan bagian memori yang sudah ditempati proses lain. Proteksi frame dilakukan dengan cara menambahkan bit proteksi untuk tiap entry page table . misal: cek apakah frame tersebut bersifat read atau read-write.cek apakah alamat logika yang dituju suatu proses valid atau invalid.

Memory Sharing pada sistem paging. Untuk menghemat penggunaan kapasitas memori.Jika ada 2 atau lebih proses yang memiliki bagian kode instruksi, atau data yang sama maka dapat digunakan bersama dan cukup diletakkan sekali di frame memori.Masing-masing proses mengacu ke frame yang sama pada page table nya.

5.6 Pengalokasian tak berurut dengan sistem segmentation

Memakai sistem partisi dinamis.Pada pengalamatan logika, image proses dibagi menjadi bagian-bagian yang disebut segmen. Pembagian segmen biasanya mengikuti struktur program oleh kompiler, yang biasanya tiap segmen berupa main program, stack, routine, symbol table. Partisi memori utama terjadi pada saat alokasi yang besarnya sesuai dengan besar segmen program yang dialokasikan. Konsep alokasi memori:Image proses dibagi menjadi beberapa segmen yang ukurannya tidak harus sama. Segmen-

segmen image proses dialokasikan ke tempat-tempat kosong di memori utama, dan informasi alokasi dicatat pada segmen table. Segmen table berisi nilai limit (panjang segmen) dan nilai base (alamat awal bagian memori yang dialokasikan)

Proteksi memori pada sistem segmentation dengan membandingkan nilai segmen yang ada di pengalamatan logika dengan nilai limit yang ada di segmen table. Apabila nilai segmen yang ada di pengalamatan logika lebih besar (>) daripada nilai limit yang ada di segmen table, berarti terjadi usaha pengaksesan lokasi diluar area segmen program itu, sehingga memicu terjadinya trap (addresing error)

Memori sharing pada sistem segmentation. Sharing segmen antara 2 atau lebih proses dapat dilakukan dengan mencatat lokasi alokasi segmen tersebut ke tabel segmen masing-masing proses.

6. Manajemen Device Input/Output (I/O)

Adapun tugas dari manajemn device adalah:

1) Mengontrol operasi operasi peranti I/O atau device pada sistem komputer, Selama suatu aplikasi digunakan umumnya diperlukan piranti I/O seperti mendapatkan masukan pengguna dari keyboard atau memunculkan hasil pemrosesan ke monitor. Sistem operasi bertugas mengambil data masukan dari keyboard atau I/O lain untuk diinput proses lain .
2) Sistem operasi mengontrol mengirimkan instruksi atau memeriksa status peranti I/O yang bersangkutan misalnya mengecek kesiapan piranti keluaran printer.
3) Menyediakan antar muka yang mudah dan beragam untuk operasi operasi I/O serta menyembunyikan detail perbedaan anatar piranti I/O. Dengan kata lain sistem operasi harus mengelola perangkat keras dan lunak piranti I/O.

Peranti I/O (DEVICE)	**Device Controller (ADAPTER)**	**BUS I/O**
Memiliki karakteristik yang khas sesuai dengan fungsi teknologi yang di gunakan . Piranti I/O dapat berupa fisik maupun mekanik	Agar piranti I/O dapat berfungsi dan berkomunikasi dengan sistem komputer maka harus ada adapter yaitu sebagai penghubung Antar muka antara peranti I/O dengan	Terdiri dari bus data ,alamat ,dan kontrol yang berfungsi menghubungkan device controller dengan elemen internal komputer seperti prosesor dan memori . Terdapat bus ekspansi yang di gunakan untuk dengan peranti

Contoh: monitor, keybord, mouse, printer , sccaner, DLL	sistem internal komputer.	I/O dengan sifat movable dan umumnya terletak di luar kotak komputer . Seperti serial, bus paralel,

Gambar 6.1 Organisasi Sistem I/O Secara Fisik (Perangkat Keras)

Berikut ini akan di gambarkan organisasi I/O perangkat lunak:

Lapisan interrupt handler	**Lapisan Device Driver**	**(Lapisan subsistem I/O atau kernel I/O)**	**(Lapisan pustaka I/O aplikasi)**
Lapisan perangkat lunak ini Menangani terjadinya intrupsi dan pengalihan eksekusi intrupt handler yang bersesuain .	Mengimplentasi khusus Abstaksi peranti I/O yang sangat beragam yang menyediakan antar muka yang seragam untuk peranti I/O yang sejenis.	Menyediakan antar muka atau fungsi I/O yang generik bagi komponen lain sistem operasi maupun aplikasi .	Mengimplentasikan pustaka pengaksesan I/O atau API (Aplication Progrraming Interface) bagi aplikasi untuk melakukan operasi I/O.

Gambar 6.2 organisasi I/O perangkat lunak

Didalam sebuah perangkat I/O terdapat bermacam macam prangkat input/outputnya. Berikut ini piranti I/O Berdasarkan karakteristiknya:

Aspect	**Varation**	**Example**
Data transfer mode	Character Block	Terminal Keyboard Disk magnetik
Akses method	Squintal Random	Modem CD room
Transfer schedule	Synchhronus Arsychrouns	Tape keyboard

Sharing	Dedycated sharably	Tape Keyboard
Device speed	Latency (waktu tggu diantrian) seek time wktu mncari lokasi data) transfer late (waktu trasnfer data dari atau ke memori)	
I/O direction	Read only Write only Read Write	CD-ROOM Grapic controller Disk

Gambar 6.3 piranti I/O berdasarkan karakteristiknya

Perangkat Keras I/O Peranti I/O Berdasarkan Fungsionalitasnya digambarkan sebagai berikut:

Piranti Antar Muka Pengguna	**Peranti Tramisi**	**Peranti Penyimpanan Data**
Piranti yang menjembatani interaksi langsung antara pengguna sistem komputer peranti input: keyboard mouse Dan monito	Piranti yang berfungsi mentransmisikan data secara internal dan ekstrenal contoh : nic (network interface card) dan modem	Piranti yang berfungsi untuk penyimpanan data. Contoh : harddisk ,floopy disk. Cd room, dsb

Gambar 6.4

6.2 Device Controller

Merupakan bagian dari organisasi fisik sistem I/O yang berfungsi sebagai pengendali digital terhadap piranti I/O dan juga bertanggung jawab atas komunikasi data antara piranti I/O dengan sistem internal computer.Device controller disisi perangkat keras dan device driver disisi perangkat lunak. Device controller dapat berupa suatu kartu rangkaian digital ataupun chipset yang ditempatkan pada rangkaian induk sistem komputer, mainboard, ataupun di piranti I/O. Pengendalian digital terhadap peranti I/O dan juga bertanggung jawab atas komunikasi data antara peranti I/O dengan sistem internal komputer. Device controller disi perangkat keras dan

device driver disisi perangkat lunak. Port contrroler device controrller khusus yang mengatur pengiriman data antara bus I/O internal cth PCI bus dengan bus I/O dengan bus eksternal seperi bus paralel, serial dan USB.

6.3 Bus I/O

Terdiri atas buah data alamat dan control yang menghubungkan device controller dengan elemen internal komputer dan memory. Contoh : bus PCI yang menghubungkan device ke memori dan prosesor . Bus ekspansi untuk menghubungan sistem internal komputer contoh : bus IDE, bus serial, paralel , dan komunikasinya di atur oleh port controller .

6.4 Pengalamatan Peranti I/O

1. Untuk dapat mengakses peranti I/O register register pada device controller harus di beri alamat. Metode pengalamatan :

- direct mapped I/O adalah addresingg piranti I/O memiliki alamat yang terpisah . Ruang alamat piranti I/O berdiri sendiri.
- memory mapped I/O addressing memiliki alamat yang merupakan bagian dari ruang alamat memori secara global. Bagian tertentu dari ruang alamat memori yang dialokasikan khusus sebagai alamat dari alamat piranti piranti I/O.

Komunikasi data lewat bus khusus ini masing-masing diatur oleh sebuah port controller. Secara fisik bus ekspansi diatas umumnya berbentuk kabel, sedangkan port controller berupa kepingan chipset pada mainboard.

I/O Address Range (Hexa)	Device
000-00F	DMA controller
020-021	Interupt controller
040-043	Timer
200-20F	Game controller
2F8-2FF	Serial port (secondary)
320-32F	Hard-disk controller
378-37F	Parallel port
3D0-3DF	Graphics controller
3F0-3F7	Diskette-drive controller
3F8-3FF	Serial port (primary)

6.5 Metode Penanganan Transfer Data Programed I/O Atau Pooling

1) Prosesor bertangung jawab atas pemeriksaan selesainya operasi transfer data yang dilakukan oleh device controller serta bertanggung jawab atas pemindahan data dari atau ke memori utama.
2) Prosesor memberikan instruksi transfer data ke device controller, ntuk mengetahui kesiapan trnsfer data ke memori utama . Jika sudah siap maka prosesor akan memindahkan ke memori utama.
3) Pemindahan data perlu dikendalikan oleh prosesor karena device controller tidak punya kendali dan hak akses langsung terhadap jalur ke memory utama, tidak efisien untuk transfer ke blok data yang besar .

6.6 Metode Transfer Data Interrupt – Driven I/O

1. Prosesor hanya bertanggung jawab atas pemindahan data dari atau ke memori utama .
2. Prosesor memberikan intruksi transfer data ke device controller untuk melanjutkan intruksi lainya.
3. Prosesor tidak perlu memeriksa ketersediaan data ke device controlle. Device controller akan memberikan sinyal interupsi ketika data sudah tersedia untuk di salinkan ke memory utama . Begitu menerima instruksi interupsi maka Prosesor menunda eksekusi proses yang sedang berlangsung dan mengalikan eksekusi ke rutin penanganan interupsi yang selanjutnya akan memindahkan data ke memory.

6.7 Metode Transfer Data Dma(Direct Memory Acses)

1) Prosesor tidak bertugas untuk melakukan controller transfer data I/O
2) Membutuhkan perangkat keras DMA control yang memiliki kendali atas bus internal dan jalur ke memori utama
3) Jika data sudah di transfer ke memori utama DMA controller akan menginterupsi prosesor sebagai informasi bahwa data I/O yang di minta oleh proses sebelumnya telah tersedia di memory utama.

Langkah langkah dma (direct memory accses):

a. Proses yang sedang berjalan melakukan operasi I/O dengan memangil salah satu fungsi device driver untuk melakukan transfer data dari disk buffer X ke memory utama.

b. Device driver menginstruksikan disk controller untuk transfer data sebesar C bytes dari dari disk ke buffer X di memori utama

c. Disk controller akan menginisiasi transfer DMA – DMA contoller mengirimkan DMA request ke prosesor. Selanjutnya Prosesor mengisi DMA controller dengan informasi transfer data . DMA akan mengirimkan sinyal DMA acknowledge sebagai izin untuk memakai bus sistem (prosesor ke memory bus)

d. DMA controler akan mengatur pemindahan setiap word dari data disk controller ke buffer X dimemori utama. Dalam aktivitas ini terjadi kompetisi pemakian bus sistem dengan proses yang sedang menggunakan prosesor.

e. Jika sudah selesai DMA mnegeluarkan sinyal interupt kepada prosesor dan mengembalikan hak pemakaian bus sistem ke prosesor.

6.8 Tujuan perancangan perangkat lunak i/o

1. Device independen , membangun lapisan bawah perangkat lunak yaitu interupt handler dan device handler sedemikian rupa sehinggalapisan perangkat lunak I/O diatasnya tidak membutuhkanpengetahuan tentang rincian oeprasi piranti I/O yang sangat bergam. Contohnya ketika menulis program yang terletak pada laisan atas , tidak diperlukan membuat berbagai versi program atau fungsi untuk setiap pirantu penyimpanan yang berbeda .
2. Uniform naming adalah penamaan yang seragam untuk berkas yang di simpan di berbagai jenis media penyimpanna yang berbeda. Nama berkas tetap sama baik disimpan di fd, cd room dan sbagainya.
3. Error handling adalah bagaimana menangani kesalahan terutama kesalahan baca dalam sistem operasi I/O. Kesalahan di tangani pada semua lapisan perangkat lunak sistem I/O.
4. Transfer sinkron vs asinkron . suatu operasi di katakan sinkron apabila operasi tersebut dapat melanjutkan eksekusinya hanya setelah permintaanya terpenuhi . suatu operasi di katakan asinkron apabila operasi tersebut dapat terus berjalan sekalipun permintaanya belum terpenuhi atau masih sedang diproses. pada proses data di asinkron proses mulai transfer data dan menjalan kan proses lainnya sampai mendapatkan sinyal bahwa operasi transfer data sudah selesai , begitu sinyal diterima, prosesor baru akan memproses atau menyelesaikan penanganan transfer data tersebut. Pada transfer sinkron, prosesor akan berhenti sampai data yang diperlukan tersedia di buffer memori.
5. Sharable vs dedicate device . suatu piranti di katakan sharble jika dapat di gunakan oleh beberapa pengguna saat bersama contoh : pembacaan berkas berkas yang terdapat pada suatu disk oleh sejumlah pengguna komputer secara bersamaan. suatu peranti di katakan dedicated jika hanya satu pengguna yang dapat menggunakan pada suatu waktu sampai tugas nya selesai contoh : printer

6.9 Komponen Perangkat Lunak Sistem I/O

1) Lapisan interupt handler, lapisan ini menangani terjadinya interupsi dan pengalihan eksekusi ke rutin penanganan interupsi yaitu interupt handler yang bersesuaian. Fasilitas interupsi memungkinkan transfer data peranti I/O secara asinkron sehingga prosesor tidak harus idle selama operasi I/O terjadi. Proses yang melakukan transfer data I/O akan beralih ke status blocked selama transfer data I/O berlangsung dan prosesor dapat dialokasikan ke proses lain
2) Lapisan device driver, membantu perangkat lunak sistem I/O untuk mencapai ketidak tergantungan dengan keberagaman piranti I/OLapisan device driver mengimplementasi

secara khusus rincian operasi dari masing-masing jenis pengendali piranti I/O atau device controller setiap device controller akan ditangani oleh suatu device driver yang khusus. Lapisan ini merupakan abstaksi terhadap operasi peranti I/O yang sangat beragam dan menyediakan antar muka yang seragam untuk piranti I/O yang sejenis.

3) Lapisan subsistem I/O atau kernel l/O . Lapiasan ini mengimplemasikan fungsi fungsi manajemen sistem I/O umum dan tidak tergantung spesifikasi dan artisektur dari sistem komputernya . Lapisan ini menyediakan antar muka atau fungsi I/O yang generik bagi komponen lain ssitem operasi maupun aplikasi. Fungsi yan diimplementasi antara lain fungsi penamaan piranti I/O, proteksi dan pelaporan kesalahan.
4) Lapisan mengimplementasi pustaka pengaksesan I/O atau API (Application Programming Interface) bagi aplikasi untuk melakukan operasi I/O. Lapisan ini memudahkan pemrogram aplikasi, karena pengaksesan ke berbagai macam piranti yang berbeda menggunakan operasi (primitive) yang sama. contohnya pustaka WIN32 sub system yang menyediakan API untuk operasi I/O dan juga operasi grafis pada sistem operasi keluarga Windows.

6.10 Manajemen Device Scheduling

Melakukan penjadwalan penggunaan suatu peranti I/O. Jika suatu proses akan menggunakan suatu peranti I/O maka akan melakukan sistem call kepada sistem operasi . Sistem call yang berhubungan deangan layanan call I/O. Jika peranti yang di minta sedang sibuk melayani proses lainmaka request tersebut di masukan kedalam antrian peranti I/O yang bersangkutan . Kernel I/O bertugas untuk melakukan penjadwalan I/O request jika piranti I/O sudah menyelesaikan I/O request sebelumnya .

6.11 Manejemen Device Buffering

1. Menampung sementara data operasi I/O baik operasi baca atapun tulis di memori utama
2. Data yang hendak di baca atau di tulis ke piranti I/O di salin terlabih dahulu ke memori utama sebelum dipidahkan ketujuan akhir . Cth: data keluaran yang hendak disimpan ke tempat penyimpanan sperti hardisk atau fd.
3. Keuntungan mekanisme buffering:

✓ mengatasi perbedaan kecepatan antar piranti I/O.
✓ mengatasi perbedaan bandtwidth transfer antar peranti I/O cth:
✓ mempertahankan semantik penyalinan data

6.12 Manajemen Device Caching

Pengakseaan peranti I/O lebih lambat di bandingkan dengan penggaksesan memori utama. pengaksesan yang terlalu sering ke piranti I/O akan memperlambat eksekusi proses secara keseluruhan dapat diatasi dengan caching. Konsep caching memakai memori kecepatan tinggi untuk menampung salinan data dari suatu peranti I/O yang merupakan kunci meningkatkan kecepatan piranti I/O . Mekanisme caching data yang di akses dari peranti I/O akan disalin ke memori utama (cache memori) sebelum data tersebut digunakan oleh sebuah proses. Pada saat operasi baca terhadap piranti I/O maka kernel I/O akan memeriksa apakah data yang hendak dibaca sudah ada dicache

memori atau belum. jika sudah, maka data akan diambil dari cache memori jika belum barulah dilakukan akses kepiranti I/O.

6.13 Manajemen Device Spooling

Kebanyakan peranti I/O hanya dapat melyani satu tugas pada suatu waktu di atasi dengan mekanisme spooling . Proses akan tetap akan mengirimkan data ke peranti I/O yang bersangkutan sehingga status proses nya tidak blocked. Namun karena kernel I/O sedang sibuk , maka kernel I/O akan menampung dulu data yang hendak dikirimkan ke piranti I/O dan menempatkannya dalam suatu antrian. Spooling di lakukan untuk alat seperti printer ,plooter dan alat cetak lainya .

6.14 Manejemen Device Reservation

Kernel I/O harus memastikan selama pengaksesan peranti I/O oleh suatu proses tidak ada intervensi dari proses lainnya .Kernel I/O bertanggung jawab memelihara dan mengaudit suatu piranti I/O sehingga tidak terjadi pemakaian yang tumpang tindi. Karnel I/O memastikan pemakain dan reservasi suatu peranti I/O tidak membat kondisi deadlock terhadap sistem komputer .

6.15 Manejemen Device Error Handling

Pada operasi I/O, data dapat rusak di peranti I/O ataupun selama proses pengiriman. Kernel I/O bertugas menangani kerusakan kerusakan data yang masih dapat di perbaiki . Kernel I/O juga bertugas untuk memulihkan keadaan yang bermasalah dan mencatat atau melaporkan kesalahan kepada pengguna.

7. Manajemen Berkas

File atau berkas adalah kumpulan informasi yg saling berkaitan dan diberi nama serta disimpan di penyimpanan sekunder. Ada bermacam jenis file, masing-masing menyimpan jenis informasi yang berbeda dan diberi penamaan (ekstensi). Biasanya sebuah berkas .merepresentasikan data atau program. Beberapa jenis berkas diantaranya:

1. Text file. yaitu urutan dari karakter-karakter yang diatur menjadi barisan dan mungkin halaman.
2. Source file yaitu urutan dari berbagai subroutine dan fungsi yang masing-masing kemudian diatur sebagai deklarasi-deklarasi diikuti oleh pernyataan-pernyataan yang dapat diexecute.
3. Object file yaitu urutan dari byte-byte yang diatur menjadi blok-blok yang dapat dipahami oleh penghubung system.
4. Executable file adalah kumpulan dari bagian-bagian kode yang dapat dibawa ke memori dan dijalankan olehloader.

   Adapun karakteristik berkasnya :

- Persistence: kemampuan sebuah data untuk tetap bertahan atau disimpan secara permanen (tidak hilang) setelah sebuah aplikasi atau sistem yang membuatnya ditutup atau dimatikan.
- Size: File umumnya berukuran besar. Memungkinkan menyimpan informasi yang sangat besar disimpan.
- Sharability : File dapat digunakan banyak proses mengakses informasi secara kongkuren

7.1 Manajemen Berkas

1. Sistem akses, Berkaitan dengan bagaimana cara data yang disimpan pada file diakses.
2. Manajemen file, Berkaitan dengan penyediaan mekanisme operasi pada file seperti :Penyimpanan, Pengacuan, Pemakaian bersama, Pengamanan.
3. Manajemen ruang penyimpan, Berkaitan dengan alokasi ruang untuk file di perangkat penyimpan.
4. Mekanisme integritas file, Berkaitan dengan jaminan bahwa sebuah file tetap tidak berubah dan tidak rusak, baik oleh kesalahan sistem maupun oleh gangguan yang disengaja.

7.2 Atribut Berkas

Selain nama dan data, sebuah berkas dikaitkan dengan informasi-informasi tertentu yg disebut atribut, diantaranya adalah :

1. Nama yang merupakan nama berkas simbolik ini adalah informasi satu-satunya yang disimpan dalam format yang dapat dibaca oleh pengguna.
2. Identifier yaitu tanda pengenal yang digunakan untuk secara unik mengidentifikasi suatu file atau objek dalam sistem file.
3. Jenis yaitu untuk sistem-sistem yang mendukung jenis berkas yang berbeda.
4. Lokasi yaitu penunjuk pada sebuah device dan pada lokasi berkas pada device tersebut.
5. Ukuran yaitu Ukuran dari sebuah berkas (dalam bytes, words, atau blocks) dan mungkin ukuran maksimum dimasukkan dalam atribut ini juga.
6. Proteksi yaitu menentukan siapa yang dapat melakukan read, write, execute, dan lainnya.
7. Waktu dan identifikasi penggua yaitu untuk pembuatan berkas, modifikasi terakhir, dan penggunaan terakhir. Data-data ini dapat berguna untuk proteksi, keamanan, dan monitoring pengguna.

7.3 Operasi berkas

Operasi berkas : create, Kita perlu dua langkah untuk membuat suatu berkas. Pertama, kita harus temukan tempat didalam sistem berkas. Kedua, sebuah entri untuk berkas yang baru harus dibuat dalam direktori. Entri dalam direktori tersebut merekam nama dari berkas dan lokasinya dalam sistem berkas.

a. Operasi berkas : write, Untuk menulis sebuah berkas, kita membuat sebuah system call yang menyebutkan nama berkas dan informasi yang akan di-tulis kedalam berkas.

b. Operasi berkas : read, Untuk membaca sebuah berkas menggunakan sebuah system call yang menyebut nama berkas yang dimana dalam blok memori berikutnya dari sebuah berkas harus diposisikan.

c. Operasi berkas : memposisikan. Direktori dicari untuk entri yang sesuai dan current-file-position diberi sebuah nilai. Operasi ini di dalam berkas tidak perlu melibatkan M/K, selain itu juga diketahui sebagai file seek.

d. Operasi berkas : delete. Untuk menghapus sebuah berkas kita mencari dalam direktori untuk nama berkas tersebut.Setelah ditemukan, kita melepaskan semua spasi berkas sehingga dapat digunakan kembali oleh berkas-berkas lainnya dan menghapus entry direktori.

e. Operasi berkas : truncate. Tindakan untuk memangkas ukuran sebuah berkas hingga ukuran tertentu.

Jenis	Ekstensi	Fungsi
Executable	exe, com, bin	Berisi bahasa mesin yang dapat langsung dibaca oleh mesin
Object	obj, dcu, o	Berisi bahasa mesin yang belum dapat langsung dieksekusi
Source Code	c, cpp, pas, bas, php, java	Berisi kode-kode program dalam bahasa pemrograman tertentu
Batch	bat, sh, wsc	Berisi kode-kode untuk memerintah *commmand interpreter* (*shell*)
Text	txt, ini, inf	Berisi teks
Document	doc, wri, rtf	Berisi data dokumen (format *word processor*)
Library	lib, so, dll, ocx	Pustaka rutin untuk programmer
Printing	pdf, ps	Berkas yang dipersiapkan untuk pencetakan
Bitmap	bmp, jpg, tiff, png, pcx, gif	Berisi data citra atau gambar bitmap (gambar digital)

Gambar 7.1 Jenis jenis berkas

- Berkas di simpan pada media penyimpan sekunder (Contoh: floopy disk, hard dsik, cakram optis-CDROM/CDRW,DVDROM/DVDRW, mobile, googledrive,icloud). Unit alokasi terkecil pada berkas blok. Direktori pada berkas (atau sering disebut folder) adalah struktur penyimpanan yang digunakan untuk mengatur dan mengelompokkan berkas-berkas di sistem komputer

- Tujuan dari adanya direktori (folder) adalah:

✓ Efisiensi – yaitu untuk menemukan suatu file.

✓ Penamaan - lebih bersahabat dengan user.

✓ Pengelompokan – terutama file yang sejenis.

- Implementation

✓ linear list

✓ tabel hash

Operasi-operasi terhadap berkas dan direktori dapat dilakukan oleh program aplikasi ataupun oleh pengguna lewat suatu aplikasi shell. shell merupakan aplikasi interaktif yang memberikan antarmuka bagi pengguna untuk mengakses layanan yang diberikan oleh sistem operasi. Contoh aplikasi shell:explorer dan cmd(WindowsXP)bash, tsh(Linux), Menutup berkas.

Operasi terhadap berkas antara lain:

- Membuat dan menghapus berkas
- Membuka suatu berkasMembaca dan menulis berkas
- Melompat ke suatu posisi
- Mengubah Nama berkas
- Mereset berkas.
- Operasi terhadap direktori antara lain:
- Membuat dan menghapus direktori
- Mencari berkas
- Melihat isi direktori
- Mengubah nama direktori

7.4 Pengelolaan Ruang Kosong

- Teknik-teknik yang dapat digunakan untuk pencatatan ruang kosong media penyimpan antara lain:

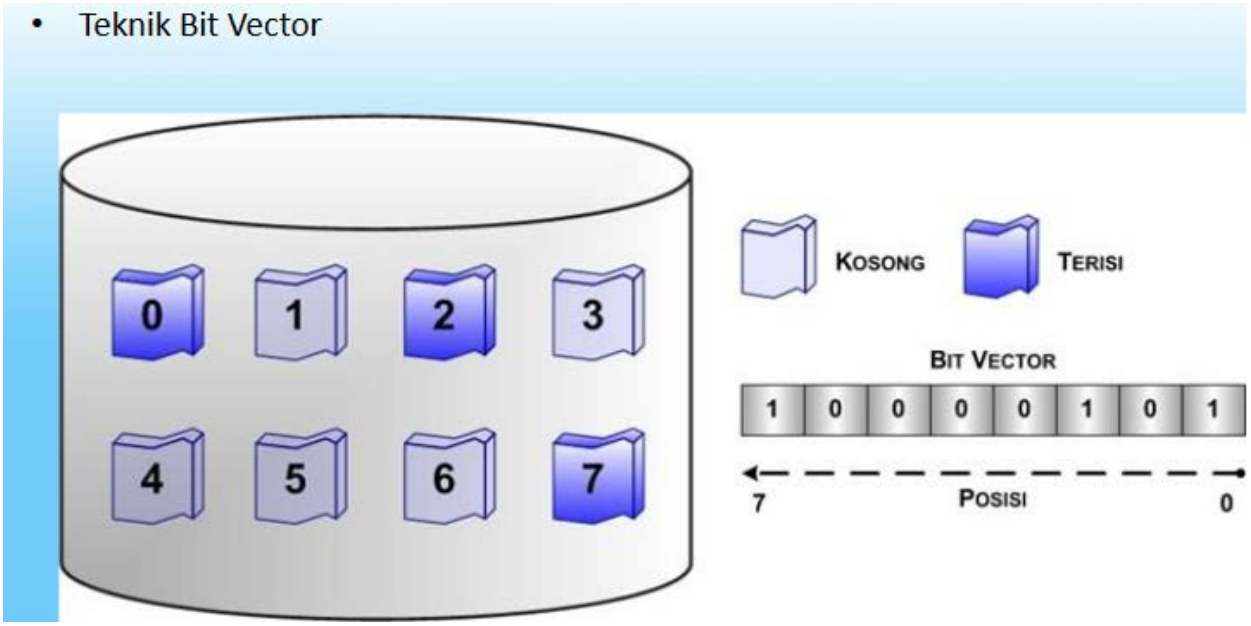
1. Menggunakan bit-vector (vektori bit)

✓ Tiap bit merepresentasi tersedia atau tidaknya (kosong) suatu blok di disk. Jumlah bit yang dibutuhkan sesuai jumlah blok disk secara logika.

✓ Vektor bit ini disimpan secara khusus di media penyimpan dan diakses pada saat terjadi alokasi berkas baru, perbesaran ukuran berkas maupun pada saat terjadi penghapusan berkas.

✓ Kelebihan: Teknik bit-vector memungkinkan pencarian ruang kosong secara cepat.

✓ Kelemahan: diperlukan ruang tambahan untuk pencatatan bit-vector yang besarnya cukup besar jika jumlah bloknya banyak.



Gambar 7.2 Teknik Bit Vektor

2. Menggunakan link list

✓ Tiap blok kosong memiliki pointer yang menunjuk ke blok kosong berikutnya, dst.

✓ Informasi pointer ini dapat disimpan di awal atau akhir setiap blok yang kosong.

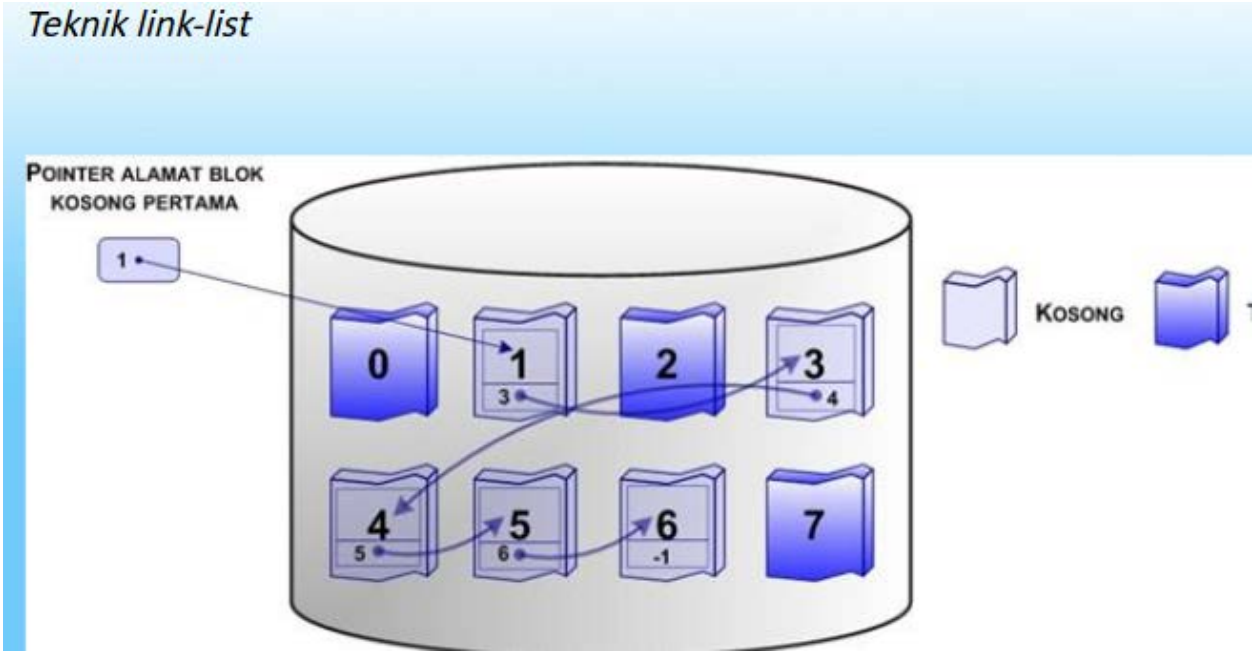
✓ Informasi yang butuh dicatat secara khusus hanyalah pointer alamat blok kosong pertaama.

✓ Kelebihan: Teknik ini menghemat penggunaan ruang khusus untuk mencatat informasi ruang kosong karena informasi blok kosong dicatat pada ruang yang memang belum dialokasikan

- ✓ Kelemahan:

- o jika terjadi kerusakan pada salah satu blok kosong, maka pengelolaan ruang kosong menjadi terganggu

- o pengalokasian sejumlah blok kosong ke berkas relatif lebih lambat karena harus mengikuti link-link pada blok kosong



Gambar 7.3 Teknik Link list

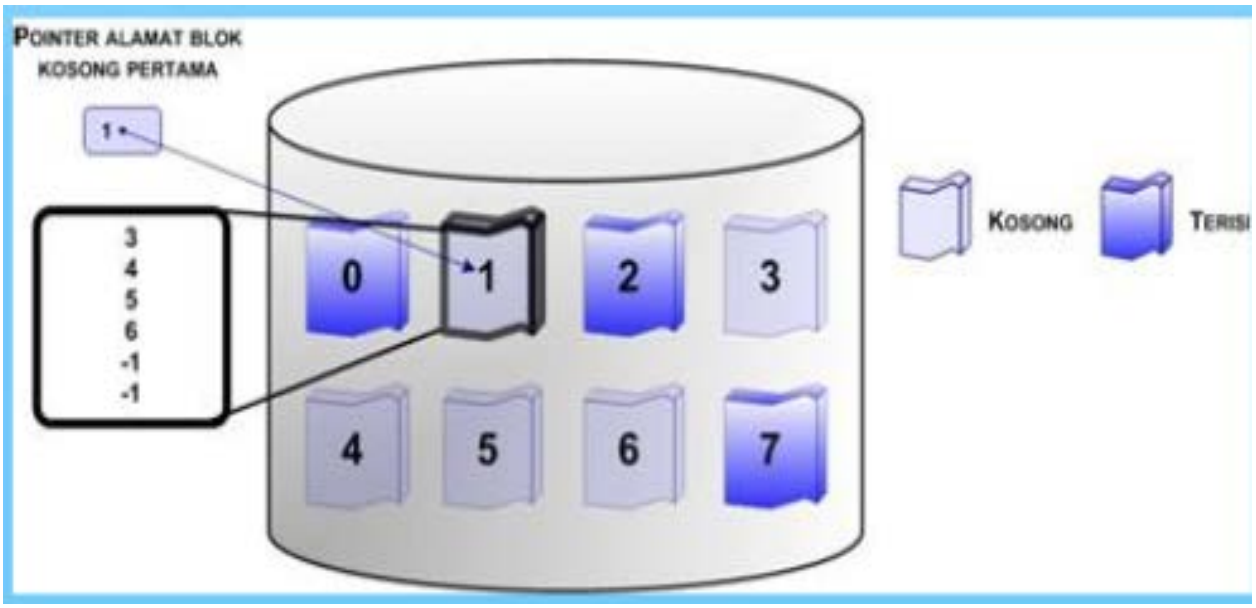
3. Menggunakan grouping

- ✓ Blok kosong pertama menyimpan seluruh pointer ke blok kosong di seluruh disk. Pencarian ruang kosong jadi lebih cepat (Bandingkan dengan menggunakan link list)

- ✓ Alamat blok kosong pertama perlu dicatat secara khusus pada struktur sistem operasi

4. Menggunakan counting

- ✓ Blok-blok kosong di disk di catat di suatu table yang berisi alamat logika disk blok kosong pertama serta jumlah blok kosong yang mengikutinya.



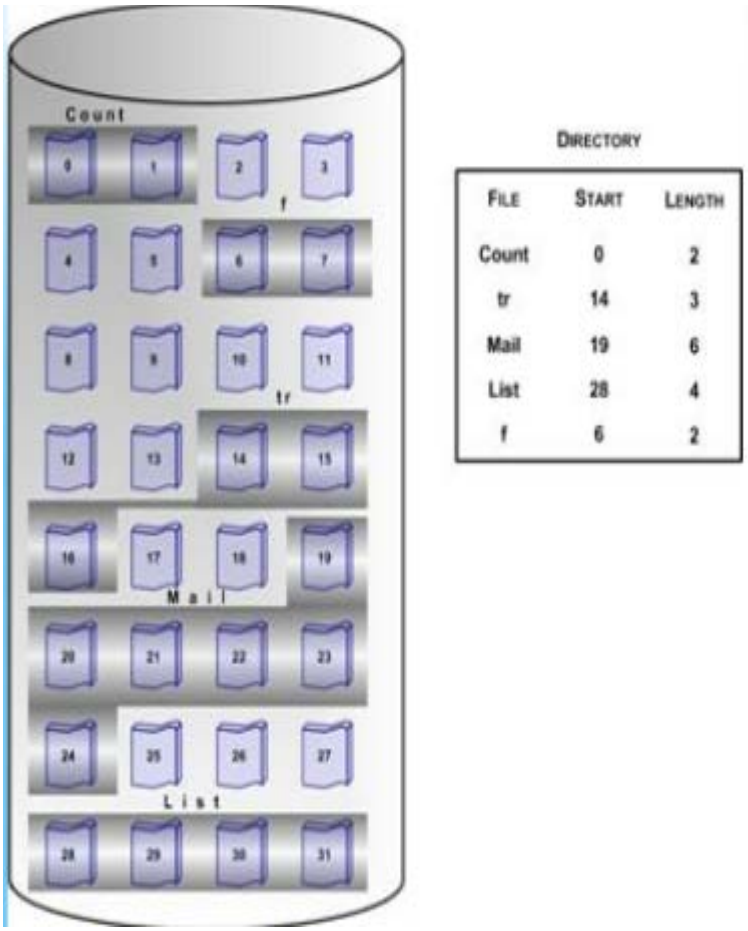
Gambar 7.4 Teknik Counting

7.5 Pengelolaan Alokasi Berkas

Berbagai cara alokasi berkas, antara lain:

1) Alokasi Berurut (Contiquous Allocation)

- ✓ File diletakkan pada lokasi disk yang berdampingan secara logika. Bisa diterapkan pada pengaksesan file secara berurut maupun langsung.
- ✓ Kelemahan: alokasi menjadi sangat tidak fleksibel karena harus mendapatkan segmen ruang kosong yang dapat menampung seluruh data berkas.



Gambar 7.5 Alokasi berurut

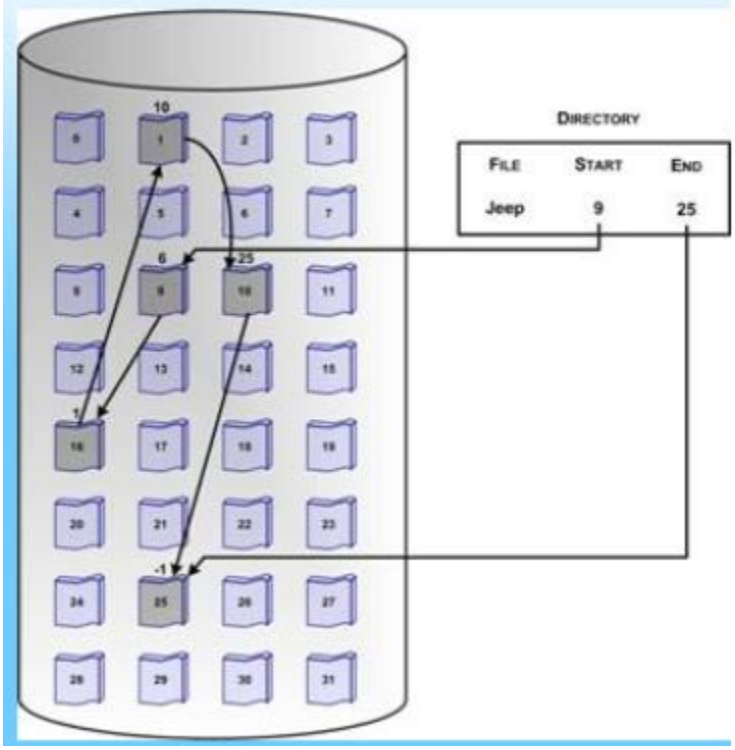
2. Alokasi Berantai (Linked Allocation)

✓ File diletakkan pada lokasi disk yang terpisah secara logika dan tersambung secara link list. Hanya bisa diterapkan pada pengaksesan file secara berurut.

✓ Kelemahan:

o kebutuhan akan ruang yang lebih untuk menyimpan pointer ke blok alokasi berikutnya.

o kesalahan data pada salah satu blok dapat membuat kerusakan pada informasi pointer sehingga menyebabkan pengaksesan ke blok lain dari berkas tidak dapat diperkirakan.

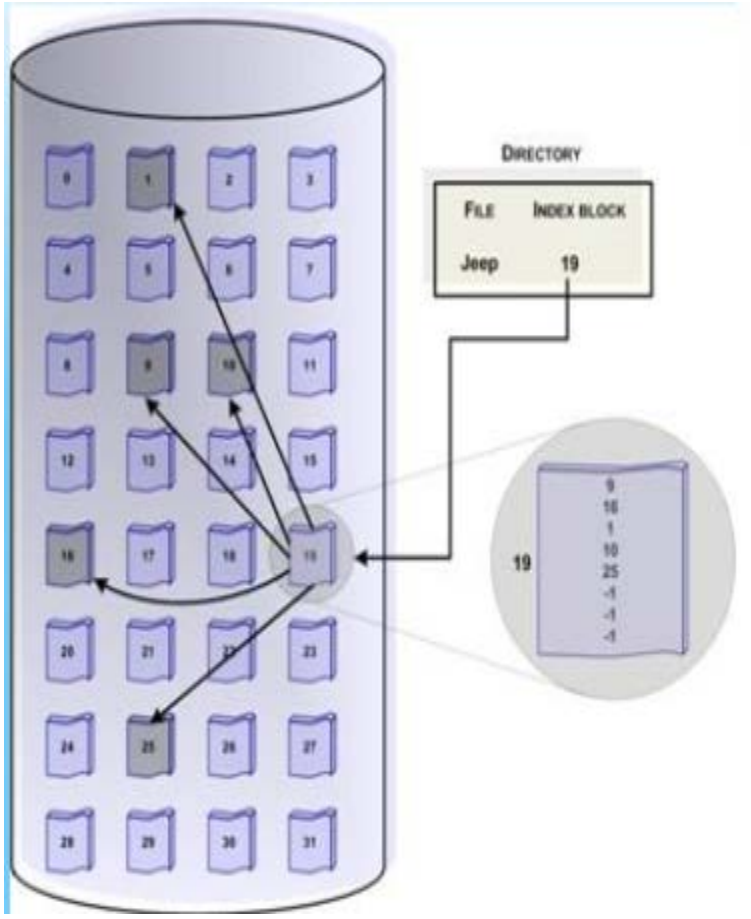


Gambar 7.6 Alokasi berantai

3. Alokasi Berindeks (Indexed Allocation)

✓ File dipecah atas blok yang sama besar dan dialokasikan di lokasi disk yang terpisah secara logika. Alokasi dicatat disuatu blok indeks yang berisi nomor-nomor blok disk yang dialokasi untuk blok-blok dari file. Pengaksesan file dilakukan lewat blok indeks tsb. Alokasi ini mirip alokasi dengan sistem paging

✓ Kelemahan: kebutuhan ruang tambahan untuk blok indeks. Jika berkasnya sangat besar, maka kemungkinan diperlukan lebih dari satu blok indeks, dan penanganannya menjadi lebih rumit.



Gambar 7.7 Alokasi Berindex

- Beberapa mekanisme proteksi yang sering digunakan adalah:

1. Proteksi berkas dengan password.

✓ Proteksi dilakukan dengan memberikan suatu password untuk setiap berkas yang disimpan

- 2. Proteksi berkas dengan list atau daftar kontrol akses (Access Control List)

✓ Proteksi dilakukan dengan membuatkan suatu daftar kontrol akses atau ACL (Access Control List) untuk setiap berkas dan disimpan pada rekaman di direktori

✓ Pada sistem operasi UNIX dan variannya, umumnya mengelompokan pengakses berkas ke dalam 3 kategori yaitu:

o Owner: pengguna yang membuat berkas tersebut.

o Group: pengguna yang berada dalam kelompok pengguna yang sama dengan pembuat berkas tersebut.

- Everyone:setiap pengguna yang tidak termasuk 2 kategori diatas.

Backup & Recovery berkas:

- Mekanisme untuk mendeteksi dan memulihkan berkas seperti:

✓ Pemeriksaan konsistensi data.Misalnya memeriksa konsistensi attribut berkas yang tersimpan pada rekaman di direktori dengan kondisi berkas yang sesungguhnya. Selain itu sistem operasi juga memeriksa apakah terjadi kerusakan pada berkas program sebelum dijalankan. Umumnya sistem operasi akan mencegah eksekusi program yang isinya sudah berubah secara tidak wajar.

✓ Backup dan pemulihan berkas Backup adalah penyalinan berkas ke media penyimpan lain sebagai cadangan dan digunakan untuk memulihkan berkas jika terjadi kerusakan pada berkas aslinya. Salah satu fungsi lain dari manajemen berkas adalah melakukan backup secara otomatis terhadap sejumlah berkas-berkas yang penting.

DAFTAR PUSTAKA

- Sri Kusumadewi, 2000, Sistem Operasi, Graha Ilmu.
- Tim SO, 2006, Sistem Operasi, Ilmu Komputer
- William Stallings, 2003, Sistem Operasi, Price Hall
- Kusnadi, Kusworo, dan Y. Sigit Purnomo, 2008, SISTEM OPERASI.