BAB II

TINJAUAN PUSTAKA

II.1. Pengembangan Perancangan Perangkat Lunak

Perancangan adalah langkah awal pada tahap pengembangan suatu produk atau sistem. Perancangan dapat didefinisikan sebagai proses untuk mengaplikasikan berbagai macam teknik dan prinsip untuk tujuan pendefinisian secara rinci suatu perangkat, proses atau sistem agar dapat direalisasikan dalam bentuk fisik. Tujuan perancangan adalah menghasilkan suatu model atau penggambaran dari suatu entity yang akan dibangun kemudian.

Perancangan perangkat lunak adalah disiplin manajerial dan teknis yang berkaitan dengan pembuatan dan pemeliharaan produk perangkat lunak secara sistematis, termasuk pengembangan dan modifikasinya, yang dilakukan pada waktu yang tepat dan dengan mempertimbangkan faktor biaya. (Laksono, 2005:136-160)

Tujuan dari perancangan perangkat lunak ini adalah untuk memperbaiki kualitas produk perangkat lunak, meningkatkan produktivitas, serta memuaskan teknisi perangkat lunak.

Menurut Laksono (2005:136-160), hal-hal yang perlu diperhatikan dalam pengembangan sebuah produk perangkat lunak adalah :

1. Kebutuhan dan batasan-batasan yang diinginkan pengguna harus ditentukan dan dinyatakan secara tegas.

2. Produk perangkat lunak harus dirancang sedemikian rupa sehingga mampu mengakomodasi paling tidak kepentingan 3 pihak berikut, yaitu pelaksana implementasi, pengguna, dan pemelihara produk.
3. Penulisan *source code* harus dilakukan dengan hati-hati dan senantiasa melalui tahap uji.
4. Dilengkapi dengan dokumen-dokumen pendukung, seperti : prinsip pengoperasian, *user's manual*, instruksi instalasi, dokumen pemeliharaan.
5. Menyiapkan bantuan pelatihan.

Beberapa atribut yang merupakan ukuran kualitas perangkat lunak adalah :

1. Kegunaan, yaitu pemenuhan terhadap kebutuhan pengguna.
2. Keandalan, yaitu kemampuan melaksanakan fungsi yang diinginkan.
3. Kejelasan, yaitu penulisan program dilakukan secara jelas dan mudah dimengerti.
4. Efisiensi, terutama dalam waktu eksekusi dan penggunaan memori.

II.2. Model Dan Simulasi

II.2.1. Model

Model adalah representasi dari suatu objek (benda) atau ide-ide dalam bentuk yang lain. Model juga dapat didefenisikan sebagai suatu gambaran, abstraksi atau imajinasi suatu sistem nyata. Ataupun berupa suatu abstraksi dunia nyata yang pada akhirnya dapat digunakan untuk mengambil keputusan. Model berisi informasi-informasi tentang sesuatu yang dibuat dengan tujuan untuk mempelajari sistem yang sebenarnya. Model dapat berupa tiruan dari suatu benda,

sistem atau peristiwa sesungguhnya yang hanya mengandung informasi-informasi yang dipandang penting untuk ditelaah. (Taha, A Hamdy, 1997:153)

Model yang dibuat dapat berguna untuk :

1. Membantu dalam berpikir, model menyajikan deskripsi yang sistematis tentang suatu sistem sehingga dapat mempermudah mempelajari sistem tersebut.
2. Membantu untuk berkomunikasi atau mempermudah menjelaskan tentang suatu sistem kepada orang lain.
3. Sebagai alat latihan, untuk melatih keterampilan orang-orang yang berhubungan dengan sistem sebenarnya yang dimodelkan. Contohnya, simulator dalam dunia penerbangan, ini digunakan untuk melatih seorang calon pilot yang dalam taraf belajar, belum boleh mengemudikan pesawat yang sebenarnya, tetapi belajar mengemudikan suatu model yang mewakili pesawat dan juga mengoperasikan model tersebut terhadap suatu model lapangan terbang, udara, lingkungan terbang dan sebagainya.
4. Sebagai alat prediksi terhadap kelakuan sistem untuk waktu yang akan datang, yaitu pengaruh-pengaruh yang ingin diketahui jika ada perubahan sistem atau operasi sistem.
5. Membantu dalam melakukan percobaan, dalam hal melakukan percobaan atau eksperimen tidak mungkin langsung dilaksanakan atau diadakan secara praktis, karena biaya yang mahal dan bahaya atau resiko yang tinggi.

Dalam pembuatan model, proses pembuatan model tidak dapat digambarkan secara pasti, namun ada petunjuk yang dapat digunakan, yaitu :

1. Pemecahan masalah melalui penyederhanaan.
2. Menyatakan objek dengan persyaratan yang jelas karena objek sangat menentukan model.
3. Mencari analog-analog dan sistem atau model yang sudah ada untuk mempermudah konstruksi.
4. Menentukan komponen-komponen yang akan dimasukkan ke dalam model.
5. Menentukan variabel, konstanta dan parameter, hubungan fungsional serta konstrain dari fungsi-fungsi kriterianya.
6. Untuk membuat model matematik, harus dipikirkan cara untuk menyatakan masalah secara numerik jika ingin disimulasikan dengan komputer.
7. Nyatakan dengan simbol-simbol.
8. Menuliskan persamaan matematikanya.
9. Bila model terlalu rumit, terdapat beberapa cara untuk menyederhanakan model, seperti:
    a. Buat harga variabel menjadi parameter.
    b. Eliminasi/ kombinasi variabel-variabel.
    c. Asumsikan linieritas.
    d. Tambahkan asumsi dan batasan yang ketat.
    e. Perjelas batasan sistem.

II.2.2. Simulasi

Simulasi adalah proses merancang model dari suatu sistem yang sebenarnya, mengadakan percobaan-percobaan terhadap model tersebut dan

mengevaluasi hasil percobaan tersebut. Simulasi dapat juga didefinisikan sebagai suatu sistem yang digunakan untuk memecahkan atau menguraikan persoalan-persoalan dalam kehidupan nyata yang penuh dengan ketidakpastian dengan tidak atau menggunakan model atau metode tertentu dan lebih ditekankan pada pemakaian komputer untuk mendapatkan solusinya. (Kakiay, 2003:2)

Adapun manfaat dari simulasi ini adalah sebagai berikut :

1. Menjelaskan kelakuan sistem.
2. Menirukan bekerjanya suatu sistem melalui suatu model.
3. Memecahkan suatu persoalan matematik dengan analisis numerik.
4. Mempelajari dinamika suatu sistem.
5. Memberikan suatu deskripsi perilaku sistem dalam perkembangan sejalan dengan bertambahnya waktu.
6. Membangun teori atau hipotesa yang mempertanggungjawabkan kelakuan dari sistem yang diamati.
7. Meramalkan kelakuan sistem yang akan dating, yaitu pengaruh yang dihasilkan oleh perubahan-perubahan sistem atau perubahan operasinya.

II.2.2.1. Teknik Simulasi

Simulasi dapat juga dianggap sebagai proses mendesain model dari suatu sistem nyata dan melakukan eksperimen dengan model tersebut untuk memahami perilaku sistem itu dan atau mengevaluasi berbagai strategi operasi dari sistem. Kata permodelan dan simulasi menunjukan kompleksitasnya aktivitas-aktivitas yang berhubungan dengan pembentukan model sistem nyata dan

mensimulasikannya pada komputer. Elemen utama yang menjadi perhatian dalam model simulasi adalah sistem nyata, model dan komputer. Model dalam simulasi tidak hanya memperhatikan elemen-elemen didalamnya, tetapi juga berhubungan antar elemennya.

Permodelan berkaitan dengan hubungan antara sistem nyata dan model, sedangkan simulasi berkaitan dengan hubungan antara komputer dengan model. Suatu sistem nyata dapat berarti suatu bagian dari dunia nyata yang memiliki suatu kepentingan tertentu. Sistem tersebut dapat berupa sistem alami atau buatan, pada kenyataan ini atau direncanakan untuk masa yang akan datang. Secara umum, sistem nyata adalah suatu sumber data perilaku kondisi dibandingkan terhadap waktu.

II.2.2.2. Kelebihan Dan Kekurangan Simulasi

Kelebihan simulasi :

1. Tidak semua sistem (terutama sistem yang kompleks) dapat direpresentasikan dalam model matematika sehingga simulasi merupakan alternatif yang tepat.
2. Model yang sudah dibuat dapat dipergunakan berulang kali dan untuk menganalisis tujuan.
3. Analisis dengan metode simulasi dapat dilakukan dengan input data yang bervariasi.
4. Simulasi dapat mengestimasi performansi suatu sistem pada kondisi tertentu dan dapat memberikan alternatif desain berdasarkan spesifikasi yang diinginkan.

5. Simulasi memungkinkan untuk melakukan percobaan terhadap sistem tanpa adanya resiko pada sistem nyata.
6. Simulasi memungkinkan untuk melakukan studi suatu sistem jangka panjang dalam waktu yang relatif singkat.

Kekurangan simulasi :

1. Simulasi hanya mengestimasi karakteristik sistem nyata berdasarkan masukan tertentu saja.
2. Harga model simulasi relatif mahal dan memerlukan waktu yang cukup banyak untuk mengembangkannya.
3. Kualitas dan analisis model tergantung kepada kualitas keahlian si pembuat model.
4. Tidak dapat menyelesaikan masalah, hanya dapat memberikan informasi darimana solusi dapat dicari.

II.3. Teori Antrian

II.3.1. Pengertian Antrian

Dalam kehidupan sehari-hari kita sering berhadapan dengan kondisi antrian. Pada sistem non-anufaktur, kita jumpai kondisi antrian ketika menunggu pelayanan di depan loket bioskop, bank, dan lain-lain. Pada sistem manufaktur, kita jumpai kondisi antrian ketika bahan baku atau barang setengah jadi menunggu untuk diproses oleh mesin-mesin yang terbatas. Dari kedua sistem diatas dapat dilihat, bukan orang saja yang mengalami antrian, tetapi bisa juga barang atau juga mesin yang menunggu untuk diperbaiki.

Antrian ialah suatu garis tunggu dari nasabah (satuan) yang memerlukan layanan dari satu atau lebih pelayan (fasilitas layanan). Suka atau tidak suka, manusia tetap harus melakukan aktivitas antrian tersebut. (Siagian, 1987:5-10)

Pada umumnya, sistem antrian dapat diklasifikasikan menjadi sistem yang berbeda – beda, dimana teori antrian dan simulasi sering diterapkan secara luas, seperti :

1. Sistem pelayanan komersial
2. Sistem pelayanan bisnis – industri
3. Sistem pelayanan transportasi
4. Sistem pelayanan sosial

II.3.2. Komponen Dasar Proses Antrian

Menurut Heizer dan Render (2006:659), terdapat 3 komponen karakteristik dalam sebuah sistem antrian, yaitu :

1. Kedatangan atau masukan sistem. Kedatangan memiliki karakteristik seperti ukuran populasi, perilaku, dan sebuah distribusi statistik.
2. Disiplin antrian. Karakteristik antrian mencakup apakah jumlah antrian terbatas atau tidak terbatas panjangnya dan materi atau orang-orang yang ada di dalamnya.
3. Fasilitas pelayanan. Karakteristiknya meliputi desain dan distribusi statistik untuk pelayanan.

Masing-masing komponen memiliki karakteristik sebagai berikut :

1. Kedatangan

Setiap masalah antrian melibatkan kedatangan, misalnya orang, mobil, atau panggilan telepon untuk dilayani. Kedatangan sering juga dinamakan proses input. Pada kedatangan memiliki tiga karakteristik utama, yaitu ukuran populasi kedatangan, perilaku kedatangan, pola kedatangan. Ukuran populasi kedatangan dilihat sebagai tidak terbatas atau terbatas. Jika jumlah kedatangan pada sebuah waktu tertentu tidak terbatas jumlahnya, maka disebut sebagai populasi tak terbatas. Dan sebaliknya, jika jumlah kedatangan pada waktu tertentu dibatasi, maka dikatakan populasi terbatas. Hampir semua model antrian berasumsi bahwa pelanggan yang datang adalah pelanggan yang sabar.

Pelanggan yang sabar adalah mesin atau orang yang menunggu dalam antrian hingga mereka dilayani dan tidak berpindah garis antrian. Pola kedatangan pada sistem antrian merupakan pola kedatangan yang acak. Kedatangan dianggap acak bila kedatangan tersebut tidak terikat satu sama lain dan kejadian kedatangan tersebut tidak dapat diramalkan secara tepat.

2. Disiplin Antrian

Garis antrian pada sebuah baris bisa terbatas atau tidak terbatas. Sebuah antrian disebut terbatas jika baris antrian tidak dapat menampung lagi antrian yang ada dikarenakan keterbatasan fisik. Model antrian dikatakan tidak terbatas ketika ukuran antrian tersebut tidak dibatasi, seperti pada kasus pintu tol yang melayani mobil yang datang. Menurut Taha (2007:548), pada baris antrian terdapat lima jenis disiplin antrian, yaitu :

a. *First Come First Served (FCFS)*

FCFS merupakan salah satu disiplin antrian dimana pelanggan yang dilayani terlebih dahulu adalah pelanggan yang datang lebih awal.

b. *Last Come First Served (LCFS)*

LCFS merupakan salah satu disiplin antrian dimana pelanggan yang datang paling akhirlah yang akan dilayani terlebih dahulu.

c. *Service in Random Order (SIRO)*

SIRO merupakan salah satu elemen sistem disiplin antrian dimana pelayanan dilakukan dalam urutan acak.

d. *Shortest Processing Time (SPT)*

SPT merupakan salah satu disiplin antrian dimana pelanggan yang memiliki waktu pelayanan atau pemrosesan yang paling singkatlah yang akan dilayani atau diproses terlebih dahulu.

e. *General Service Discipline (GD)*

GD digunakan jika disiplin antrian tidak ditentukan dan hasil yang diperoleh akan sama dengan disiplin antrian yang lain, misalnya *FCFS* dan *LCFS*.

3. Fasilitas Pelayanan

Komponen pelayanan memiliki 2 hal penting dalam karakteristik pelayanan, yaitu desain sistem pelayanan dan distribusi waktu pelayanan. Pada desain sistem pelayanan umumnya digolongkan menurut jumlah saluran yang ada

dan jumlah tahapan. Untuk distribusi pelayanan, pola pelayanan serupa dengan pola kedatangan dimana pola ini bisa konstan ataupun acak.

II.4. Penyelesaian Dari *Lamport's Bakery Algorithm*

Solusi Lamport disebut juga sebagai solusi tukang kue (*bakery algorithm*). Algoritma ini didasarkan pada penjadualan di toko kue, es krim, atau daging. Pada saat memasuki toko, tiap pembeli menerima nomor antrian. Jika terdapat pelayan yang sedang *idle* (tidak sedang/ akan melayani pembeli lainnya), maka pembeli langsung meminta pelayanan dari pelayan tersebut. Jika tidak, maka pembeli duduk di tempat tunggu pembeli dan menunggu hingga gilirannya. Pembeli yang telah selesai dilayani keluar dari toko dan pelayan yang sedang *idle* tersebut melayani pembeli bernomor terendah yang sedang menunggu di tempat tunggu pembeli.

Langkah kerja dari algoritma Lamport ini dapat dijabarkan sebagai berikut:

1. Pembeli memasuki toko dan menerima nomor urut.
2. Jika terdapat pelayan yang sedang *idle* (tidak sedang/ akan melayani pembeli lainnya), maka pembeli langsung meminta pelayanan dari pelayan tersebut.
3. Jika tidak, maka pembeli duduk di tempat tunggu pembeli dan menunggu hingga gilirannya.
4. Pembeli yang telah selesai dilayani keluar dari toko dan pelayan yang sedang *idle* tersebut melayani pembeli bernomor terendah yang sedang menunggu di tempat tunggu pembeli.

Asumsi yang digunakan oleh *Lamport* adalah :

1. Hanya satu orang pembeli yang masuk dalam waktu tertentu.
2. Jumlah item dari setiap pembeli dicatat, namun jenis item tidak diperhatikan.

II.5. Proses

II.5.1. Definisi Proses

Sistem operasi pada dasarnya adalah seperti perangkat lunak lain, yaitu program yang perlu dieksekusi pemroses. Sistem operasi adalah kumpulan proses (*process-based operating system*) yang memiliki fungsi-fungsi yang dieksekusi dalam proses pemakai.

Proses adalah entitas dinamis, yang berisi instruksi dan data, *program counter* dan semua *register* pemroses dan *stack* berisi data sementara, seperti parameter rutin, alamat pengiriman dan variabel-variabel lokal. Proses juga merupakan unit kerja terkecil yang secara individu memiliki sumber daya dan dijadualkan sistem operasi.

Proses dapat didefinisikan sebagai suatu bagian program yang sedang dalam keadaan eksekusi. Suatu proses juga memuat *program counter*, *register* dan variabel. Suatu proses membutuhkan *resource*, seperti CPU *time*, memori, *file* dan *I/O device* untuk menyelesaikan pekerjaannya. *Resource-resource* tersebut dapat dialokasikan pada saat dibuat atau pada saat dieksekusi.

II.5.2. Penjadualan Proses

Suatu komputer dapat memiliki lebih dari satu proses yang berjalan di dalamnya pada saat tertentu, oleh karena itu dibutuhkan suatu penjadualan. Penjadualan merupakan kumpulan kebijaksanaan dan mekanisme di sistem operasi yang berkaitan dengan urutan kerja yang dilakukan sistem komputer.

Penjadualan bertugas untuk memutuskan :

1. Proses yang harus berjalan.
2. Kapan dan berapa lama proses itu berjalan.

II.5.2.1. Sasaran Utama Penjadualan Proses

Sasaran utama dari penjadualan proses antara lain :

1. Keadilan (*fairness*)

   Keadilan artinya adalah proses-proses diperlakukan sama, yaitu mendapat jatah waktu pemroses yang sama dan tak ada proses yang tidak kebagian layanan pemroses sehingga mengalami *starvation*. Proses dikatakan mengalami *starvation* bila proses-proses itu menunggu alokasi sumber daya sampai tak berhingga, sementara proses-proses lain dapat memperoleh alokasi sumber daya. Sasaran penjadualan seharusnya menjamin tiap proses mendapat pelayanan dari pemroses yang adil.

2. Efisiensi

   Efisiensi atau utilisasi pemroses dihitung dengan perbandingan (rasio) waktu sibuk pemroses. Sasaran penjadualan adalah menjaga agar pemroses tetap dalam keadaan sibuk sehingga efisiensi mencapai maksimum. Sibuk adalah

pemroses tidak menganggur, termasuk waktu yang dihabiskan untuk mengeksekusi program pemakai dan sistem operasi.

3. Waktu Tanggap (*response time*)

   Waktu tanggap berbeda untuk :

   a. Sistem Interaktif

      Waktu tanggap dalam sistem interaktif didefinisikan sebagai waktu yang dihabiskan saat karakter terakhir dari perintah dimasukkan atau transaksi sampai hasil pertama muncul di layar (terminal).

   b. Sistem Waktu Nyata

      Pada sistem waktu nyata (*real-time*), waktu tanggap didefinisikan sebagai waktu dari saat kejadian (internal atau eksternal) sampai instruksi pertama rutin layanan yang disebut dengan *event response time*. Sasaran penjadualan adalah meminimalkan waktu tanggap.

   c. *Turn Around Time*

      Turn around time adalah waktu yang dihabiskan dari saat program atau *job* mulai masuk ke sistem sampai proses diselesaikan sistem. Waktu yang dimaksud adalah waktu yang dihabiskan di dalam sistem, diekspresikan sebagai penjumlah waktu eksekusi (waktu pelayanan *job*) dan waktu menunggu, yaitu :

Turn around time = waktu eksekusi + waktu menunggu

   d. *Throughput*

      Throughput adalah jumlah kerja yang dapat diselesaikan dalam satu unit waktu. Cara untuk mengekspresikan *throughput* adalah

dengan jumlah *job* pemakai yang dapat dieksekusi dalam satu unit/ interval waktu. Sasaran penjadualan adalah memaksimalkan jumlah *job* yang diproses per satu interval waktu. Lebih tinggi angka *throughput*, lebih banyak kerja yang dilakukan sistem.

II.5.2.2. Tipe-Tipe Penjadualan

Terdapat 3 tipe penjadual berada secara bersama-sama pada sistem operasi, yaitu:

1. Penjadual Jangka Pendek (*short-term scheduller*)

   Penjadual ini bertugas menjadualkan alokasi pemroses di antara proses-proses *ready* di memori utama.

2. Penjadual Jangka Menengah (*medium-term scheduller*)

   Penjadual jangka menengah berfungsi untuk menangani proses-proses *swapping*. Proses-proses mempunyai kepentingan kecil saat itu sebagai proses yang tertunda.

3. Penjadual Jangka Panjang (*long-term scheduller*)

   Penjadual jangka panjang bekerja terhadap antrian *batch* dan memilih *batch* berikutnya yang harus dieksekusi. *Batch* biasanya adalah proses-proses dengan penggunaan sumber daya yang intensif (yaitu waktu pemroses, memori, perangkat masukan/ keluaran), program-program berprioritas rendah.

II.5.2.3. Strategi Penjadualan

Terdapat 2 strategi penjadualan, yaitu :

1. Penjadualan *Non-Preemptive*

   Begitu proses diberi jatah waktu, maka pemroses tidak dapat diambil alih oleh proses lain sampai proses itu selesai. Contoh algoritma yang menerapkan strategi ini adalah algoritma FIFO (*First In First Out*), SJF (*Shortest Job First*), HRN (*Highest-Ratio Next*), MFQ (*Multiple Feedback Queues*), dan sebagainya.

2. Penjadualan *Preemptive*

   Saat proses diberi jatah waktu, maka pemroses dapat diambil alih proses lain sehingga proses disela sebelum selesai dan harus dilanjutkan menunggu jatah waktu pemroses tiba kembali pada proses itu. Contoh algoritma yang menerapkan strategi ini adalah algoritma *Round Robin* (RR), *Shortest-Remaining First* (SRF), *Priority Schedulling* (PS), *Guaranteed Schedulling* (GS), dan sebagainya.

II.5.3. Status Proses

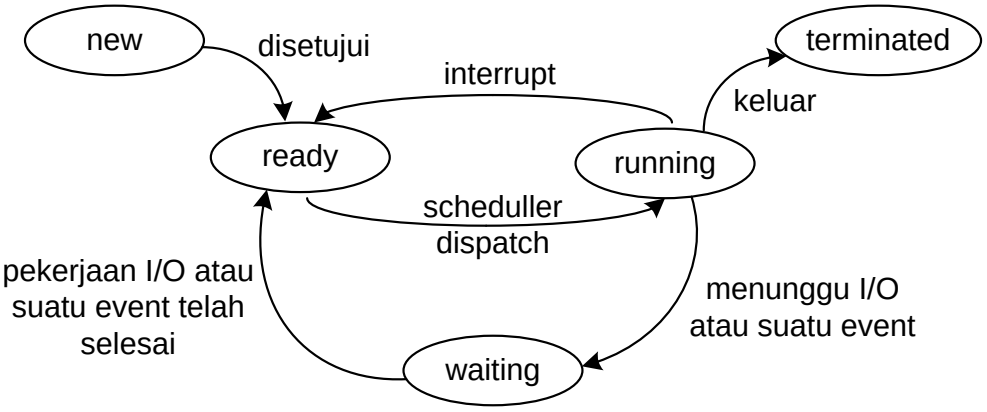
Meskipun tiap-tiap proses terdiri dari suatu kesatuan yang terpisah, namun adakalanya proses-proses tersebut perlu saling berinteraksi. Satu proses dapat dibangkitkan dari *output* proses lainnya sebagai *input*.

Pada saat dieksekusi, akan terjadi perubahan status. Status proses didefinisikan sebagai bagian dari aktivitas proses yang sedang berlangsung saat itu.

Tiap-tiap proses memiliki salah satu status di bawah ini, yaitu :

1. *New*, yaitu proses sedang dibuat.

2. *Running*, yaitu proses dieksekusi, karena CPU tidak sedang mengerjakan tugas yang lain.
3. *Waiting*, yaitu proses sedang menunggu beberapa *event* yang akan terjadi (seperti menunggu untuk menyelesaikan *I/O* atau menerima sinyal).
4. *Ready*, yaitu proses menunggu jatah waktu dari prosesor.
5. *Terminated*, yaitu proses telah selesai dieksekusi.



Gambar II.1. Diagram Status Proses

II.5.4. Konkurensi Pada Proses

Konkurensi merupakan landasan umum perancangan sistem operasi. Proses-proses disebut konkuren jika proses-proses (lebih dari satu proses) berada dalam sistem/ memori pada saat yang sama. Proses konkuren dapat sepenuhnya tak bergantung dengan lainnya tapi dapat juga saling berinteraksi.

Masalah pada konkurensi antara lain :

1. *Mutual Exclusion*, adalah jaminan hanya satu proses yang mengakses sumber daya pada suatu interval waktu.
2. Sinkronisasi

Proses-proses konkuren dapat sepenuhnya tak bergantung dengan lainnya tapi dapat juga saling berinteraksi. Proses-proses yang berinteraksi memerlukan sinkronisasi agar terkendali dengan baik.

3. *Deadlock*

   Proses disebut *deadlock* jika proses menunggu satu kejadian tertentu yang tak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. Proses menunggu kejadian yang tidak akan pernah terjadi.

4. Komunikasi Antara Proses

   Masalah lebih besar dibandingkan dengan sinkronisasi adalah komunikasi antar proses *IPC* (*Inter Process Communication*), memungkinkan pertukaran data atau komunikasi antar proses untuk bekerjasama mencapai sasaran aplikasi.

Pada *multiprocessor*, beberapa instruksi dapat dilakukan konkuren di pemroses berbeda. Kemampuan ini dapat mereduksi waktu eksekusi proses.

Misalkan terdapat program berikut :

1. a: = x + y
2. b: = z + 1
3. c: = a – b
4. w: = c + 1

Pernyataan 3 tidak dapat dieksekusi sebelum pernyataan 1 dan 2 dieksekusi. Pernyataan 4 tidak dapat dieksekusi sebelum pernyataan 3 dieksekusi. Pernyataan 1 dan 2 dapat dieksekusi konkuren karena tidak saling bergantung.

II.6. *Mutual Exclusion And Critical Section*

Mutual exclusion merupakan salah satu masalah yang muncul pada konkurensi. Dalam sistem operasi, terdapat sumber daya yang tidak dapat dipakai pada saat bersamaan, seperti *printer*. Sumber daya semacam ini disebut sumber daya kritis. Program yang menggunakan sumber daya kritis disebut memasuki *critical region/ section*. Pemrogram tidak dapat bergantung pada sistem operasi untuk memahami dan memaksakan batasan ini.

Sistem operasi hanya menyediakan layanan untuk mencegah proses masuk *critical region* jika ada proses lain sedang dalam *critical region*-nya. Pemrogram harus menspesifikasikan bagian-bagian *critical region*, sehingga sistem operasi akan menjaganya dengan suatu mekanisme untuk mencegah proses lain masuk *critical region* yang sedang dipakai oleh proses lain.

Secara umum, penyelesaian *critical section* harus memenuhi 3 syarat, yaitu :

1. Jika suatu proses sedang mengerjakan *critical section*, maka tidak boleh ada proses lain yang masuk (mengerjakan) *critical section* tersebut.
2. Jika tidak ada suatu proses yang mengerjakan *critical section* dan ada beberapa proses yang masuk ke *critical section*, maka hanya proses-proses

yang sedang berada pada *entry-section* saja yang boleh berkompetisi dan mengerjakan *critical section*.

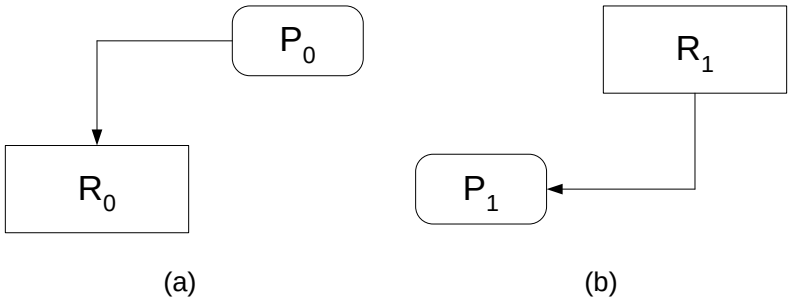
3. Besarnya waktu tunggu dari suatu proses yang akan memasuki *critical section*, sejak proses itu meminta ijin untuk mengerjakan *critical section*, hingga permintaan itu dipenuhi.

Penyelesaian *mutual exclusion* untuk N proses secara perangkat lunak dapat dijabarkan oleh beberapa ahli, seperti berikut :

1. Solusi pertama masalah *mutual-exclusion* untuk N proses diberikan Dijkstra. Solusi lebih sederhana diberikan Peterson. Kedua solusi ini tidak memiliki batas atas penungguan memasuki *critical section*.
2. Knuth memberikan solusi yang memiliki batas atas, yaitu sebesar 2^n giliran. Solusi kemudian diperbaiki oleh DeBrujin dengan batas atas n^2 giliran.
3. Solusi lainnya diberikan oleh Lamport dengan algoritma yang dikenal sebagai *Bakery Algorithm* yang akan dibahas pada bagian berikutnya.

II.7. *Deadlock*

Proses disebut *deadlock* jika proses menunggu satu kejadian tertentu yang tidak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian, yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. *Deadlock* terjadi ketika proses-proses mengakses secara eksklusif sumber daya. Semua *deadlock* yang terjadi melibatkan persaingan memperoleh sumber daya eksklusif oleh dua proses atau lebih.

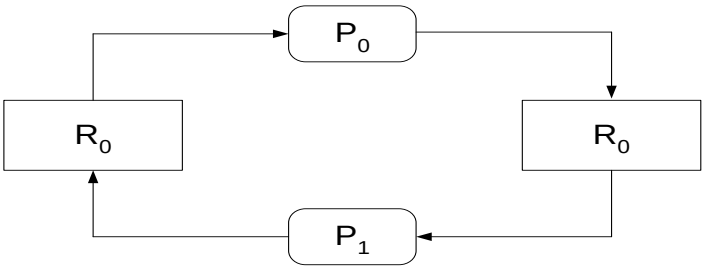


Gambar II.2. Graph Meminta Sumber Daya Dan Alokasi Sumber Daya

Gambar II.2 (a), P_0 meminta sumber daya R_0, ditandai busur (*edge*) berarah dari proses P_0 ke sumber daya R_0. Gambar II.2 (b), sumber daya R_1 dialokasikan ke P_1, ditandai busur berarah dari sumber daya R_1 ke proses P_1, kemudian terjadi skenario berikut :

- P_0 sambil masih menggenggam R_0, meminta R_1
- P_1 sambil masih menggenggam R_1, meminta R_0

Kejadian ini mengakibatkan *deadlock* karena sama-sama akan saling menunggu. *Deadlock* tidak hanya terjadi pada 2 proses dan 2 sumber daya, *deadlock* juga dapat terjadi dengan melibatkan lebih dari 2 proses dan 2 sumber daya.



Gambar II.3. Graph *Deadlock* 2 Proses Dan 2 Sumber Daya

II.7.1. Syarat-Syarat Terjadinya *Deadlock*

Menurut Coffman (2000), terdapat 4 syarat terjadinya *deadlock*, yaitu :

1. Kondisi *mutual exclusion* (*mutual exclution condition*)

   Tiap sumber daya saat itu diberikan pada tepat satu proses.

2. Kondisi genggam dan tunggu (*hold and wait condition*)

   Proses-proses yang sedang mengenggam sumber daya, menunggu sumber daya baru.

3. Kondisi *non-preemption* (*non-preemption condition*)

   Sumber daya yang sebelumnya diberikan tidak dapat diambil paksa dari proses itu. Sumber daya harus secara eksplisit dilepaskan dari proses yang menggenggamnya.

4. Kondisi menunggu secara sirkuler (*circular wait condition*)

   Harus terdapat rantai sirkuler dari 2 proses atau lebih, masing-masing menunggu sumber daya yang digenggam oleh anggota berikutnya pada rantai itu.

Ketiga syarat pertama merupakan syarat perlu (*necessary condition*) bagi terjadinya *deadlock*. Keberadaan *deadlock* selalu berarti terpenuhi kondisi-kondisi di atas, tidak mungkin terjadi *deadlock* bila tidak ada ketiga kondisi itu. *Deadlock* terjadi berarti terdapat ketiga kondisi itu, tetapi adanya ketiga kondisi itu belum berarti terjadi *deadlock*. *Deadlock* baru benar-benar terjadi bila syarat keempat terpenuhi. Kondisi keempat merupakan keharusan bagi terjadinya peristiwa *deadlock*. Bila salah satu kondisi saja tidak terpenuhi, maka *deadlock* tidak terjadi.

II.7.2. Pencegahan *Deadlock*

Havender, J.W (2001), dalam bukunya yang berjudul "*Avoiding Deadlock In Multitasking Systems*", mengemukakan jika sembarang syarat dari keempat syarat tidak terpenuhi, maka tidak akan terjadi *deadlock*.

Havender menyarankan starategi-strategi berikut untuk meniadakan syarat-syarat tersebut, yaitu :

1. Tiap proses harus meminta sumber daya yang diperlukan sekaligus dan tidak berlanjut sampai semuanya diberikan.
2. Jika proses telah sedang memegang sumber daya tertentu, untuk permintaan berikutnya proses harus melepas dulu sumber daya yang dipegangnya. Jika diperlukan, proses meminta kembali sekaligus dengan sumber daya yang baru.
3. Beri pengurutan linear terhadap tipe-tipe sumber daya pada semua proses, yaitu jika proses telah dialokasikan ke suatu tipe sumber daya, proses hanya boleh meminta sumber daya pada urutan yang berikutnya.

II.7.3. *Starvation*

Proses dikatakan mengalami *starvation*, bila proses-proses itu menunggu alokasi sumber daya sampai tak terhingga, sementara proses-proses lain dapat memperoleh alokasi sumber daya. *Starvation* bisa disebabkan pada strategi alokasi sumber daya. Kondisi ini harus dihindari karena tidak adil, tetapi dikehendaki penghindaran dilakukan seefisien mungkin.

II.8. Animasi

II.8.1. Sejarah Animasi

Persepsi gambar bergerak pertama kali ditemukan pada tahun 1834 oleh William Horner, yang disebut "*Daedalum*" atau "roda setan". Penemuan ini dinyatakan lebih efisien dan dapat digunakan lebih dari satu orang pada saat yang bersamaan.

Sampai pada tahun 1867, William F. Lincoln menamainya menjadi "*Zoetrope*". *Zoetrope* terbuat dari silinder yang gemuk dan pendek yang berputar dengan sumbu yang simetri dan di dalam silinder tersebut terdapat gambar yang sedikit berbeda dengan gambar yang ada disampingnya, sehingga ketika silinder tersebut berputar akan menghasilkan efek gambar yang bergerak.

Animasi lain yang populer pada saat tersebut adalah *Flipbook*, yang merupakan urutan gambar yang diletakkan pada penjepit kertas secara berurutan dan dibalikkan secara cepat sehingga menimbulkan efek pergerakan. Animasi berkembang pesat ketika trik yang lebih baik didapatkan. Dengan menggunakan teknologi film dan projektor, maka dapat dibuat animasi yang cukup panjang pada saat itu, walaupun masih tidak ada suara. Sampai pada tahun 1928, *Disney* membuat animasi dengan suara pada film kartun "*Steamboat Willie*" yang merupakan awal dari industri perfilman modern.

II.8.2. Definisi Animasi

Animasi merupakan salah satu bagian grafika komputer yang menyajikan tampilan-tampilan yang sangat atraktif dan juga merupakan sekumpulan gambar

yang ditampilkan secara berurutan dengan cepat untuk mensimulasi gerakan yang hidup. Pemanfaatan animasi dapat ditujukan untuk simulasi, menarik perhatian pemakai komputer pada bagian tertentu dari layar, memvisualisasikan cara kerja suatu alat, atau menampilkan keluaran program dengan gambar-gambar yang menarik dibanding dengan sederetan angka, serta tidak ketinggalan untuk program-program permainan.

Pada dasarnya, animasi adalah transformasi objek, dimana semua titik pada sembarang objek akan diubah sesuai dengan aturan tertentu, sementara sistem koordinaatnya tetap. Implementasi pada animasi dapat dikerjakan secara interaktif maupun non-interaktif. Dibandingkan animasi non-interaktif, animasi interaktif memberikan tampilan yang lebih menarik dan dinamis. Pada animasi interaktif, pergerakan objek mengikuti perintah yang diberikan oleh pemakai lewat perangkat interaktif. Sedangkan animasi non-interaktif, pergerakan objek hanya dikendalikan dari prosedur yang ada di dalam sebuah program. Untuk animasi interaktif kebanyakan digunakan untuk program-program permainan, sedangkan animasi non-interaktif kebanyakan untuk melakukan simulasi objek.

Pembuatan animasi masih dilakukan secara sederhana dan konvensional, dengan cara menggerakkan beberapa gambar secara bergantian dan cepat sebelum tahun 1970-an. Gambar tersebut masih menggunakan lukisan tangan atau menggunakan foto dari serangkaian kejadian. Komputer digital yang berkembang pesat sangat mempengaruhi proses pengerjaan animasi. Animasi kemudian membentuk suatu bidang baru dalam ilmu komputer, yaitu grafika komputer yang dapat digunakan untuk menggambarkan cara kerja suatu alat dan menampilkan

keluaran program berupa gambar yang lebih hidup dan interaktif. Animasi banyak digunakan pada berbagai bidang seperti bidang perekayasaan, arsitektur, ekonomi, kedokteran, dan lain-lain.

Animasi yang bagus dihasilkan dari gambar yang cukup banyak agar gambar yang dihasilkan akan tampak gerakan yang berkesan halus. Dalam hal ini, maka gambar-gambar tersebut haruslah berpindah posisi sekecil mungkin agar pada perubahan atau pergantian gambar terlihat lebih menarik dan bagus. Selain itu diperlukan juga kecepatan tertentu untuk tampilan gambar yang akan dibuat dalam animasi, hal ini tergantung pada jumlah gambar yang diberikan. Kecepatan yang dimaksud yaitu begitu satu gambar ditampilkan maka akan berganti gambar berikutnya dengan kecepatan tertentu. Makin cepat pergantian antara satu gambar dengan gambar berikutnya maka akan menghasilkan gerakan gambar yang semakin halus.

II.8.3. Teknik Animasi

Untuk menghasilkan animasi yang baik dan bagus dalam pembuatan animasi, maka diperlukan teknik-teknik atau metode-metode animasi yang sangat mendukung dan sesuai untuk pembuatan animasi yang bagus dan baik.

Berikut beberapa teknik animasi yang mendukung dalam pembuatan animasi :

1. Teknik Animasi Dengan Mengubah Warna

   Teknik animasi ini akan menghasilkan animasi berwarna. Hal ini mendukung pembuatan animasi yang sangat menarik dan menakjubkan. Teknik animasi

ini hampir mirip dengan cara membuat suatu lukisan berwarna, dimana sebelum para pelukis menggoreskan cat pada kanvas, terlebih dahulu mereka mencampur cat yang berwarna-warni untuk memperoleh warna cat yang diinginkan. Maka tentunya akan menghasilkan lukisan dengan tata warna yang indah dan menakjubkan. Melalui teknik dengan mengubah warna inilah bisa diperoleh efek animasi yang diinginkan dan lebih menarik.

2. Teknik Animasi *Tweening*

   Teknik animasi *tweening* merupakan teknik yang mudah untuk dipelajari. Dalam pembuatan program, seperti program untuk menganimasikan diagram garis dan diagram batang bisa digunakan teknik animasi *tweening*. Cara kerja teknik animasi ini pada dasarnya adalah dengan menentukan posisi awal dan akhir dari objek (gambar atau tulisan), kemudian dihitung posisi objek yang baru, menghapus objek pada posisi semula dan menggambar objek pada posisi yang baru sampai objek berada pada posisi akhir yang dituju.

   Teknik animasi *tweening* ini memperlihatkan objek yang sesudah dianimasikan tidak harus sama dengan objek sebelum dianimasikan. Hal ini diperoleh dengan cara mengubah koordinaat dari titik-titik yang akan membentuk gambar yang tidak sama dengan gambar aslinya.

3. Teknik Animasi Dengan Permainan Halaman

   Dalam teknik ini perlu diperhatikan bagaimana menampilkan halaman secara benar, dan juga cara menggambar objek yang akan dianimasikan. Pada prakteknya, yang terlihat pada layar tampilan adalah gambar yang terdapat pada halaman. Dengan demikian, dapat digambarkan beberapa gambar yang

hampir sama pada halaman yang ada untuk memperoleh efek animasi yang diinginkan. Melakukan teknik animasi ini adalah dengan memanfaatkan beberapa halaman (*page*) yang dimiliki oleh sejumlah adapter grafik, walaupun tidak semua adapter grafik mendukung lebih dari sebuah halaman. Pelaksanaan teknik animasi ini adalah dengan menggambar objek yang akan dianimasikan sedikit berbeda untuk setiap halaman. Jika halaman-halaman tersebut secara bergantian ditampilkan, akan diperoleh efek animasi yang diinginkan.

4. Teknik Animasi Dengan Pergerakan Citra

   Teknik ini digunakan untuk menghasilkan animasi dengan proses yang lebih cepat dengan menggerakkan gambar di sekeliling layar tampilan. Prosesnya adalah menggerakkan gambar secara keseluruhan untuk memperoleh efek animasi. Animasi dengan menggerakkan gambar di sekeliling layar ini mengikuti urutan operasi yang sudah tertentu. Urutan operasi ini mencakup penentuan ukuran citra, penentuan lokasi (tempat) peubah dinamis untuk penyimpanan citra, pemindahan posisi citra ke posisi lain, dan penghapusan lokasi peubah dinamis untuk penyimpanan citra.