## 1. Pentingnya Mutual Exclusion.

***Mutual Exclusion*** adalah jaminan hanya sau proses yang mengakses sumber daya pada suatu interval waktu tertentu, Bagian program yang sedang mengakses memori atau sumber daya yang dipakai bersama disebut ***Critical Section / Region***

Sekenario :
Proses A membaca variable ***in*** bernilai 10. belum sempat A menyelesaikan proses, penjadwal menjadwalkan proses B berjalan. Proses B yang telah dijadwalkan juga membaca variable ***in***, yang tentunya variable ***in*** masih bernilai 10. B dapat menyelesaikan prosesnya. Proses B menyimpan berkasnya di slot ke 10. A dijadwalkan kembali dan menyimpan berkas A di slot ke 10. berkas B ditimpa berkas A. B tidak akan pernah mendapatkan hasil

Diketahui terdapat suatu kondisi diatas dimana dua proses atau lebih sedang membaca atau menulis data yang di pakai bersama dengan hasil akhir bergantung bagaimana proses-proses itu berjalan. Hasil akhir tidak dapat di prediksi. Kondisi ini disebut kondisi pacu atau ***( race condition )***

Untuk mengatasi kondisi pacu harus dijamin tidak boleh dua proses atau lebih memasuki critical section yang sama secara bersamaan.

Kesuksesan proses-proses kongkuren memerlukan pendefinisian critical section, dan memaksakan mutual exclusion diantara proses-proses kongkuren yang sedang berjalan. Pemaksaan mutual exclusion merupakan

landasan pemrosesan kongkuren. Fasilitas atau kemampuan menyediakan dukukngan mutual exclusion harus memenuhi kriteria-kriteria berikut. [ TAN-92, STA-95]

1. mutual exclusion harus dijamin
   hanya satu proses pada saat diijinkan masuk critical selection. Proses-proses lain dilarang masuk critical section yang sama pada saat telah terdapat satu proses masuk critical section itu

2. proses yang berada di non critical section, dilarang mem-blocked proses-proses lain yang ingin masuk critical section

3. harus dijamin proses yang ingin masuk critical section tidak menunggu selama waktu yang tak berhingga. Atau tak boleh terdapat deadlock atau starvation.

4. ketika tidak ada proses pada critical section maka proses yang ingin masuk critical section harus diijinkan masuk tanpa waktu tunda.

5. tidak ada asumsi mengenai kecepatan relatif proses atau jumlah proses yang ada.

6. proses hanya tinggal pada critical section selama satu waktu yang berhingga.

## 2. Metode Naif.

### 2.1. Metode Variabel Lock.

Ketika proses hendak masuk critical section, lebih dahulu memeriksa variabel lock

- Jika variabel lock bernilai 0, proses men-set variabel lock menjadi satu dan kemudian masuk critical section
- jika variabel lock bernilai 1, proses maka proses menunggu sampai variabel lock bernilai 0 ( berarti terdapat proses lain pada critical section )

scenario :

proses A setelah membaca variable lock, dan sebelum men-set variable lock menjadi 1. penjadwal dapat menjadwalkan exekusi proses B. Proses B pun membaca variable lock bernilai 0 dan masuk critical section. Penjadwal menggilir A, maka karena telah membaca variable lock bernilai 0 maka proses A pun seegera masuk critical section yang telah dimasuki B. secara bersamaan A dan B masuk critical section yang sama.

metode ini masih gagal karena tidak menjamin proses tidak masuk critical section yang telah dimasuki proses lain

## 2.2. Metode Bergantian Secara Ketat.

Metode ini mengasumsikan dapat menggilir masuk critical section secara bergantian terus menerus

Variabel turn diinisialisasi dengan 0, variabel turn mencatat ( ***nomor*** ) proses yang sedang memasuki critical section, memeriksa dan memperbarui memori yang dipakai bersama.

Skenario yang terjadi.

- Proses 0 menginspeksi variabel turn, menemukan nilainya 0 dan segera memasuki critical section.
- Proses 1 menemukan variabel turn bernilai 0, maka melakukan loop yang secara terus menerus memeriksa variabel turn untuk memeriksa apakah turn menjadi 1.

Kondisi terus menerus memeriksa variabel, menunggu suatu nilai muncul disebut ***busy waiting***. Kalau busy waiting dapat terjadi dalam waktu yang lama maka harus dihindari cara ini karena menyiakan banyak waktu pemroses. Hanya kalau lama menunggu adalah singkat, maka metode dengan busy waiting dapat digunakan.

Skenario pelanggaran yang dapat terjadi:

- Proses 0 meninggalkan critical section, men set turn 1, mengijinkan proses 1 memasuki critical section.
- Proses 1 mengakhiri critical section dengan cepat, maka keduanya berada pada non critical section, dengan variabel turn bernilai 0.

- Proses 0 kembali memasuki critical section dengan cepat dan segera memasuki non-critical section, maka variabel turn bernilai 1.
- Proses 0 hendak kembali memasuki critical section tapi variabel turn bernilai 1.

Proses 1 yang berada di non critical section memblok proses 0 sehingga tak dapat memasuki critical section. Metode ini melanggar kriteria solusi mutual exclusion, yaitu proses di blocked oleh proses yang sedang tidak berada di critical section.

## 3. Metode dengan Busy Waiting

### 3.1. Metode Secara Perangkat Lunak

#### 3.1.1. Metode Penyelesaian Dekker

Algoritma dekker mempunyai properti-properti berikut:

- Tidak memerlukan instruksi-instruksi perangkat keras yang khusus
- Proses yang beroprasi diluar critical section tidak dapat mencegah proses lain masuk critical section itu
- Proses yang ingin masuk critical section akan segera masuk bila memungkinkan

#### 3.1.2. Metode Penyelesaian Peterson

Mekanisme kerja algoritma peterson adalah sebagai berikut:

Sebelum masuk critical section, proses memanggil enter_critical_ section. Sebelum memanggil enter_critical section, proses memeriksa sampai kondisi aman masuk critical_section. Terjadi bussy waiting. Setelah selesai dengan critical section, proses menandai pekerjaan telah selesai dilakukan dan mengijinkan proses lain masuk.

## 3.2. Metode dengan Dukungan Perangkat Keras

### 3.2.1. Metode Pematian Interupsi

Proses mematikan intrupsi ke pemroses dan segera masuk critical section. Proses mengaktifkan kembali interupsi begitu meninggalkan critical section.

Cara pematian interupsi mengakibatkan

- Pemroses tidak dapat beralih ke proses lain, karena interupsi clock dimatikan sehingga penjadwalpun tidak dieksekusi.
- Proses dapat memeriksa dam memperbarui memori bersama tanpa takut proses lain intervensi karena memang tidak ada proses lain yang dieksekusi pada saat itu.

Tekhnik dengan mematikan ini mempunyai kelemahan utama, yaitu:

- Bila proses yang mematikan interupsi mengalami gangguan yaitu crash, maka proses tak pernah menghidupkan interupsi kembali.

- Jika terdapat dua pemroses atau lebih, mematikan interupsi hanya berpengaruh pada proses yang mengeksekusi instruksi itu.

### 3.2.2. Metode dengan Instruksi *Test and Set Lock ( tsl )*

Instruksi ini membaca isi memori ke register dan kemudian menyimpan nilai non-zero ke alamat memori itu. Oprasi membaca isi memori dan menyimpan ke memori dijamin tak dapat diinterupsi. Tak ada pemroses lain yang dapat mengakses memori itu sampai instruksi berakhir. Pemroses yang menginstruksi tsl mengunci bus memori, mencegah pemroses-pemroses lain mengakses memori itu.

### 3.2.3. Metode dengan Instruksi *Exchange ( XCHG )*

Instruksi ini saling menukarkan dua isi memori. Instruksi xchg diseiakan mesin (intel memakai pendekatan ini) proses pembacaan dan penukaran tersebut dilakukan secara anatomik, tidak dapat disela. Instruksi dapat digunakan untuk implementasi mutual exclusion.

### 3.2.4. Karakteristik Pendekatan dengan Instruksi Mesin

Instruksi mesin:

- Test and set lock ( tsl ).
- Test and set ( ts atau tas )
- Compare and set ( cs )

- Exchange ( xchg )
- Dan sebagainya.

Penggunaan instruksi-instruksi mesin untuk memaksakan mutual exclusion mempunyai sejumlah keunggulan [ STA-95 ] diantaranya:

- Pendekatan ini dapat diterapkan ke sembarang jumlah proses baik pemroses tunggal maupun banyak pemroses yang memakai memori bersama.
- Pendekatan ini sederhana dan mudah diverifikasi.
- Pendekatan ini dapat digunakan untuk mendukung banyak critical region, masing-masing critical region didefinisasikan dengansuatu variabel.

Kelemahan serius yang dipunyai pendekatan ini adalah

- Merupakan metode dengan busy waiting sangat tidak efisien. Selagi proses menunggu memasuki critical region, proses berlanjut mengkonsumsi waktu pemroses.
- Adanya busy waiting memungkinkan terjadinya deadlock dan starvation.

### 3.3. Dampak Adanya Busy Waiting

Metode-metode dengan busy waiting mempunyai keterbatasan yaitu tidak dapat diterapkan pada sistem dengan penjadwalan berprioritas. Pada sistem-sistem dengan penjadwalan berprioritas, metode dengan busy waiting dapat menimbulkan

dead lock. Aturan penjadwalan berprioritas selalu menjadwalkan proses-proses berprioritas lebih tinggi yang terdapat di antrian proses ready.

## 4. Metode dengan Semaphore

### 4.1. Metode Semaphore

#### 4.1.1 Deskripsi Semaphore

Dua proses atau lebih dapat bekerja sama dengan menggunakan penanda-penanda sederhana seperti proses dapat dipaksa berhenti pada suatu saat sampai proses mendapatkan penanda tertentu itu. Sembarang kebutuhan koordinasi komplex dapat dipenuhi dengan struktur penanda yang cocok untuk kebutuhan itu. Variabel khusus untuk penandaan ini disebut semaphore.

Semaphore mempunyai du properti yaitu:

1. Semaphore dapat diinisialisasi dengan nilai nonnegatif.
2. Terdapat dua oprasi terhadap semaphore yaitu Down dan Up.

**Oprasi Down** : oprasi ini menurunkan nilai semaphore. Jika nilai sempahore menjadi nonpositive maka proses yang mengeksekusinya di- blocked

**Oprasi up** : oprasi menaikan nilai semaphore. Jika suatu proses atau lebih telah di blocked pada semaphore itu tak dapat menyelesaikan oprasi Down, maka salah satu

dipilih oleh sistem dan dibolehkan menyelesaikan oprasi Downnya.

### 4.1.2 Mutual Exclusion dengan Semaphore

Skema penyelesaian mutual exclusion dengan semaphore
Sebelum masuk critical section, proses melakukan Down. Bila berhasil maka proses masuk critical section. Bila tidak berhasil maka proses di-Blocked atas semaphore itu. Proses yang di-blocked akan dapat melanjutkan kembali bila proses yang berada di critical section keluar dan melakukan oprasi Up sehingga menjadikan proses yang di-Blocked ready dan melanjutkan sehingga oprasi Down nya berhasil

## 4.2. Implementasi Semaphore

Dengan pematian interupsi
Sistem oprasi mematikan semua interupsi selagi memeriksa semaphore, memperbaharui dan menjadikan proses di blocked. Karena semua aksi hanya memerlukan beberapa instruksi, tak rugi dilakukan dalam kondisi interupsi dimatikan.

Dengan instruksi tsl
Pada banyak pemroses, tiap semaphore dilindungi variabel lock dan instruksi tsl agar menjamin hanya satu pemroses yang saat itu memanipulasi semaphore.

# 5. Masalah Sinkronisasi

Yang dimaksud dengan Sinkronisasi adalah apabila didalam suatu system terjadi suatu proses maka proses tersebut harus bersama-sama menghindari kondisi malapetaka atau bertubrukan agar kedua proses mencapai tujuan secara benar. Sinkronisasi dapat terlihat jelas dalam masalah Poducer-Consumer

**Masalah Producer-Consumer**

Masalah Producer-Consumer disebut juga bounded Buffer (masalah buffer yang terbatas}. Dua proses mempunyai buffer bersama, buffer berukuran tetap. Satu proses adalah produsen yang meletakan informasi ke buffer. Prose lain adalah Consumer yang mengambil informasi dari buffer.

Karena buffer yang terbatas akan mengakibatkan petaka (bencana)

- Petaka produsen

  Petaka terjadi ketika buffer penuh, sementara prodesen ingin meletakan informasi ke buffer yang telah penuh itu.

- Petaka consumer

  Petaka terjadi ketika konsumen ingin mengambil informasi sementara buffer telah/sedang kosong.

Kedua proses memerlukan sinkronisasi agar sama-sama dapat menghindari petaka untuk masing-masig kondisi.

### 5.1. Penyelesaian dengan Metode Sleep and Wake-Up

Sleep dan Wake-up. Kedua rutin ini atomic, jadi ketika kedua rutin dieksekusi maka tak akan ada interupsi yang dapat menyela. **Sleep** adalah rutin yang menyebabkan pemanggil di-blocked, ditunda sampai ada proses lain yg membangunkan (Wake-up). **Wake-up** adalah rutin yang digunakan untuk

membangunkan proses yang sleep. Wake-up mempunyai satu parameter, yaitu proses yang dibangunkan.

- Petaka producer adalah tidur atau di-blocked yaitu producer memanggil system begitu mengetahui buffer telah penuh maka produsen menyimpan informasi ke buffer. Producer tak aktif lagi kecuali dibangunkan dibangunkan oleh proses lain yang memberitahu bahwa satu item atau lebih telah diambil sehingga terdapat ruang untuk menyimpan informasi.
- Petaka consumer yaitu consumer memanggil system call sleep begitu buffer telah kosong saat consumer mengambil item. Konsumen tak aktif kecuali dibangunkan oleh proses lain yang memberitahu bahwa buffer telah terisi satu item atau lebih.

Solusi dengan Sleep dan Wake-up mempunyai kelemahan disebabkan terjadinya akses variabel count yang tidak terjaga (terlindungi).

## 5.2. Penyelesaian dengan Metode Semaphore

Kelemahan metode semaphore adalah pengurutan operasi harus benar-benar diperhatikan pemrogram. Karena apabila terjadi kesalahan dalam buffer akan terjadi deadlock antara producer dan consumer.

## 5.3. Penyelesaian dengan Metode Event-Counters

Terdapat dua bagian pada metode ini, yaitu EventCount dan Sequencer. EvenCount berfungsi untuk mencatat jumlah kemunculan kejadian-kejadian dari kelas kejadian tertentu yang berhubungan. Sequencer berfungsi mengurutkan kejadian-kejadian. EvenCounter adalah penghitung (bilangan bulat) evencount diinisialisasikan dengan nol.

Terdapat tiga operasi primitive yang didefinisikan pada E (event counter)

1. Read (E)

   Mengirimkan nilai E saat itu
2. Advance (E)

   Merupakan operasi atomik yang menaikan E dan I
3. Await (E,v)

   Menunggu sampai E mempunyai nilai sebesar v atau lebih.

## 5.4. Penyelesaian dengan Metode Monitors

Semaphore merupakan alat bantu (tools) ampuh dan fleksibel dalam memaksakan mutual exclusion dan mengkoordinasi proses-proses. Monitor adalah kumpulan prosedur, variabel dan struktur data di satu modul atau paket khusus, proses dapat memanggil prosedur-prosedur di monitor kapan menginginkan.

Monitor merupakan bentukan bahasa pemrograman yang menyediakan fungsionalitas ekivalen dengan semaphore tapi dengan cara lebih mudah. Bentukan monitor telah diimplementasikan di sejumlah kompilator bahasa pemrograman.

Dalam Metode ini terdapat cara agar proses yang tidak berlangsung di-blocked, yaitu menambahkan variabel kondisi, dengan dua operasi **Wait** dan **Signal**

- Wait

  Ketika prosedur monitor tidak dapat berlanjut, misal producer menemui buffer penuh menyebabkan proses pemanggil di-blocked dan mengijinkan proses lain masuk monitor.

- Signal

  Proses membangunkan partner-nya yang sedang blocked dengan signal pada variabel kondisi yang sedang ditunggu partner-nya. Signal terdapat dua versi yaitu:

  1. Versi Hoare

     Setelah signal, membangunkan proses baru agar berjalan dan menunda proses lain

  2. Versi Brinch Hansen

     Setelah melakukan signal, proses segara keluar dari monitor.

Keunggulan monitor atas semaphore adalah semua fungsi sinkronisasi dilakukan terpusat di monitor. Dengan ini lebih

mudah memverifikasi kebenaran sinkronisasi dan mendeteksi kesalahan-kesalahan (bugs).

## 5.5. Kelemahan Metode Semaphore dan Metode Monitors

- Metode hanya menyelesaikan mutual-exclusion pada satu pemroses atau lebih yang mempunyai memori dipakai bersama (**Tighly Couple multiprocessor**)
- Metode tidak dapat diterapkan pada system terdistribusi berisi banyak pemroses dan memiliki memori sendiri (**loosely coupled multiprocessor**)

## 5.6. Penyelesaian dengan Metode Message-Passing

Message Passing dapat dilalukukan dengan beragam cara. Secara umum system message-passing menggunakan dua primitive, yaitu

- Send (destination, message), digunakan untuk mengirim pesan (message) ke tujuan tertentu (destination).
- Receive (source, message), digunakan untuk menerima pesan (message) dari sumber tertentu

Hal-hal yang perlu diperhatikan dalam merancang system message passing, yaitu

1. Sinkronisasi (synchronization)
2. Pengalamatan (addressing)
3. Format pesan (message format)
4. Disiplin antrian (queuing discipline)

Beberapa-beberapa masalah yang muncul pada system ini, yaitu

- Berurusan dengan hilangnya pesan di jaringan

- Berurusan dengan **authentication** yaitu menjamin komunikasi benar-benar dilakukan antar mesin-mesin yang diotorisasi
- Kinerja bila **message passing** diterapkan pada proses-proses pada mesin yang (uniprocessors)

Masalah-masalah pada perancangan dan implementasi system message passing

1. Masalah sinkronisasi proses
    - Pada proses pengiriman (send), dapat berupa
        - § Blocking, yaitu setelah mengirim (send) maka proses di-blocked menunggu jawaban dari proses yang dikirimi
        - § Non-blocking, yaitu setelah mengirim (send) maka proses tetap melanjutkan eksekusi intruksi-intruksi berikutnya
    - Pada proses penerima (receive), dapat berupa
        - § Blocking, yaitu setelah menyatakan menerima (receive) maka proses di blocked menunggu kiriman pesan dari proses yang diinginkan
        - § Non-blocking, yaitu setelah menyatakan menerima maka proses tetap melanjutkan eksekusi intruksi-intruksi berikutnya
    - Masalah yang muncul pada penerima adalah bagaimana mengimplementasikan pemeriksaan adanya kedatangan pesan

2 Masalah pengalamatan (addressing), terdapat beragam alternative, yaitu :

a. Pengalamatan langsung (direct), berbeda untuk

- § Prose pengirim
- § Proses penerima, mempunyai alternative
  - v Pengalamatan eksplisit
  - v Pengalamatan implicit

b. Pengalamatan tak langsung (direct)

- § Pengalamatan statis
- § Pengalamatan dinamis
- § Pengalamatan kepemilikan

3 Masalah format pesan (message format) berupa

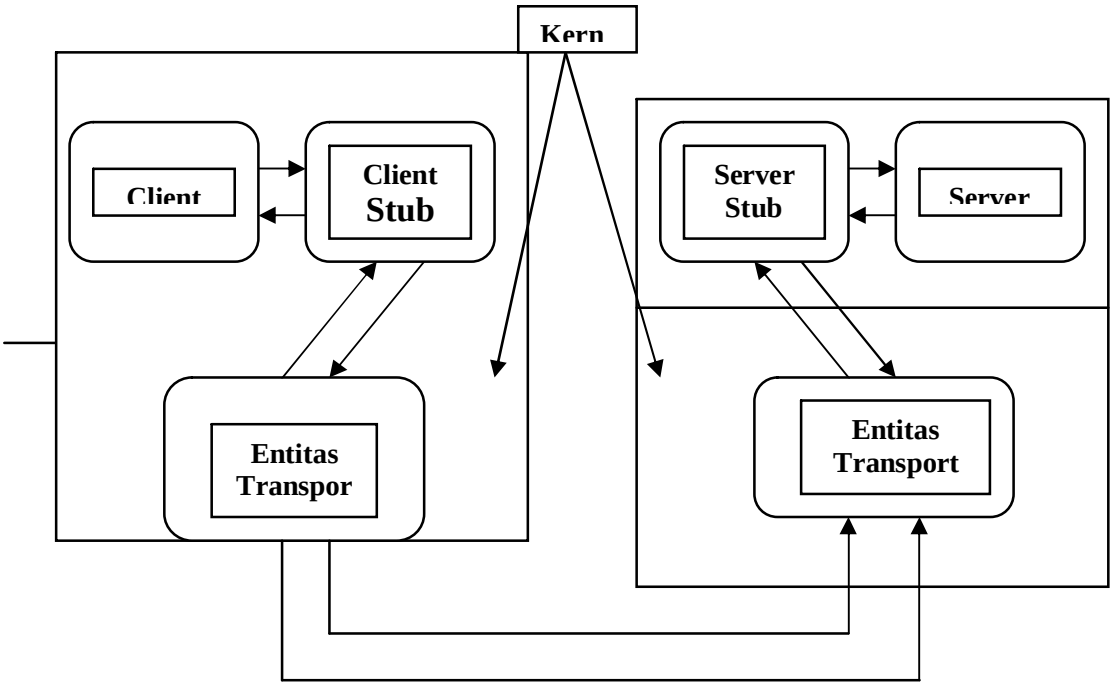
- Isi pesan (content)
- Panjang pesan mempunyai alternative
  - v Pesan dengan panjang tetap
  - v Pesan dengan panjang dapat beragam

4 Masalah disiplin antrian (queuing discipline), dalam mengolah pesan-pesan yang telah masuk dapat berupa

- FIFO
- Berprioritas

## 5.7. Penyelesaian dengan Metode Remote Procedure Call ( RPC )

Tujuan RPC adalah membuat prosedur jarak jauh (dikerjakan di mesin lain) seperti pemanggilan prosedur lokal. Dengan RPC, pemrogram akan menulis program seperti biasanya tanpa memperdulikan apakah prosedurnya dijalankan di pemroses setempat atau dijalankan si pemroses jauh.




Gambar skema eksekusi RPC

Panggilan remoter terdiri dari 10 langkah. ***Langkah 1*** berisi program client (atau prosedur) memanggil stub prosedur yang dilink di ruang alamatnya. Setelah pesan dibangun, pesan diberikanke lapisan transport untuk transmisi ***Langkah 2***, ***Langkah 3*** pada system Lan tak koneksi, entitas transport hanya mencatolkan header ke pesan dan meletakan ke jaringan tanpa banyak kerja lain, ***Langkah 4*** ketika pesan tiba di server, entitas transport melewatkan ke sub-serveryang me-unmarshal parameter-parameter ***Langkah 5*** Stub server kemudian memanggil prosedur server, ***Langkah 6*** Setelah menyelesaikan kerja, prosedur server mengembalikan, ***Langkah 7*** Stub server kemudian me-marshaall hasil ke pesan dan melepaskan ke interface transport, ***Langkah 8*** Setelah jawaban tiba di mesin client, ***Langkah 9*** Jawaban ditangani stub client, ***Langkah 10*** stub client mengembalikan ke pemanggil, prosedur client. Suatu nilai yang dikembalikan server pada langkah 6 diberikan ke client dalam langkah 10.

**Beberapa masalah untuk mengimplementasikan RPC antara lain**

1 Parameter passing terutama ***passing by reference***

2 Banyak waktu yang dihabiskan untuk konversi representasi internal pada mesin-mesin yang berbeda

3 Terdapat beberapa altenatif dalam mengartikan kegagalan pada RPC, semantics kegagalan dapat dikategorikan menjadi tiga, yaitu :

- **At least once,** menjamin server crash telah mengeksekusisekali atau lebih
- **At most once,** tak ada panggilan yang dieksekusi lebih dari satu kali
- **Maybe,** system tidak menjamin apapun, paling mudah diimplementasikan

## 5.8. Penyelesaian dengan Metode Rendezvous Bahasa ADA

Bahasa ADA merupakan bahasa pemrograman kongkuren pertama kali yang memakai rendezvouz sistem ini biasanya dipakai oleh militer. Rendezvouz merupakan teknik yang diajukan Hoare. Teknik ini secara efektif dapat menyelesaikan kombinasi masalah

- Mutual exclusion
- Sinkronisasi proses
- Komunikasi antar proses

- **Sinkronisasi**

Yang dimaksud dengan Sinkronisasi adalah apabila didalam suatu system terjadi suatu proses maka proses tersebut harus bersama-sama menghindari kondisi malapetaka atau bertubrukan agar kedua proses mencapai tujuan secara benar. Sinkronisasi dapat terlihat jelas dalam masalah Poducer-Consumer

- **Masalah Producer-Consumer**

Masalah Producer-Consumer disebut juga bounded Buffer (masalah buffer yang terbatas}. Dua proses mempunyai buffer bersama, buffer berukuran tetap. Satu proses adalah produsen yang meletakan informasi ke buffer. Prose lain adalah Consumer yang mengambil informasi dari buffer.

Karena buffer yang terbatas akan mengakibatkan petaka (bencana)

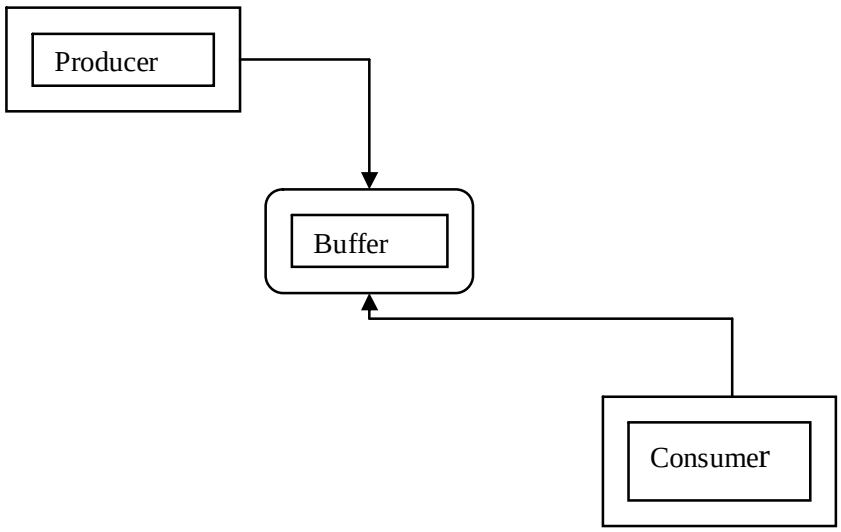
- Petaka produsen

  Petaka terjadi ketika buffer penuh, sementara prodesen ingin meletakan informasi ke buffer yang telah penuh itu.

- Petaka consumer

  Petaka terjadi ketika konsumen ingin mengambil informasi sementara buffer telah/sedang kosong.

Kedua proses memerlukan sinkronisasi agar sama-sama dapat menghindari petaka untuk masing-masig kondisi. Dapat tergambarkan sebagai berikut



- **Penyelesaian dengan Metode Sleep and Wake-up**

Sleep dan Wake-up. Kedua rutin ini atomic, jadi ketika kedua rutin dieksekusi maka tak akan ada interupsi yang dapat menyela. **Sleep** adalah rutin yang menyebabkan pemanggil di-blocked, ditunda sampai ada proses lain yg membangunkan (Wake-up). **Wake-up** adalah rutin yang digunakan untuk membangunkan proses yang sleep. Wake-up mempunyai satu parameter, yaitu proses yang dibangunkan.

- Petaka producer adalah tidur atau di-blocked yaitu producer memanggil system begitu mengetahui buffer telah penuh maka produsen menyimpan informasi ke buffer. Producer tak aktif lagi kecuali dibangunkan dibangunkan oleh proses lain yang memberitahu bahwa satu item atau lebih telah diambil sehingga terdapat ruang untuk menyimpan informasi.
- Petaka consumer yaitu consumer memanggil system call sleep begitu buffer telah kosong saat consumer mengambil item. Konsumen tak aktif kecuali dibangunkan oleh proses lain yang memberitahu bahwa buffer telah terisi satu item atau lebih.

Solusi dengan Sleep dan Wake-up mempunyai kelemahan disebabkan terjadinya akses variabel count yang tidak terjaga (terlindungi).

- **Penyelesaian dengan Metode Semapore**

Kelemahan metode semaphore adalah pengurutan operasi harus benar-benar diperhatikan pemrogram. Karena apabila terjadi kesalahan dalam buffer akan terjadi deadlock antara producer dan consumer.

- **Penyelesaian dengan Metode Even –Counters**

Terdapat dua bagian pada metode ini, yaitu EventCount dan Sequencer. EvenCount berfungsi untuk mencatat jumlah kemunculan kejadian-kejadian dari kelas kejadian tertentu yang berhubungan. Sequencer berfungsi mengurutkan kejadian-kejadian. EvenCounter adalah penghitung (bilangan bulat) evencount diinisialisasikan dengan nol.
Terdapat tiga operasi primitive yang didefinisikan pada E (event counter)

4. Read (E)

Mengirimkan nilai E saat itu

5. Advance (E)

   Merupakan operasi atomik yang menaikan E dan I

6. Await (E,v)

   Menunggu sampai E mempunyai nilai sebesar v atau lebih.

- **Penyelesaian dengan Metode Monitors**

Semaphore merupakan alat bantu (tools) ampuh dan fleksibel dalam memaksakan mutual exclusion dan mengkoordinasi proses-proses. Monitor adalah kumpulan prosedur, variabel dan struktur data di satu modul atau paket khusus, proses dapat memanggil prosedur-prosedur di monitor kapan menginginkan. Monitor merupakan bentukan bahasa pemrograman yang menyediakan fungsionalitas ekivalen dengan semaphore tapi dengan cara lebih mudah. Bentukan monitor telah diimplementasikan di sejumlah kompilator bahasa pemrograman.

Dalam Metode ini terdapat cara agar proses yang tidak berlangsung di-blocked, yaitu menambahkan variabel kondisi, dengan dua operasi **Wait** dan **Signal**

- Wait

  Ketika prosedur monitor tidak dapat berlanjut, misal producer menemui buffer penuh menyebabkan proses pemanggil di-blocked dan mengijinkan proses lain masuk monitor.

- Signal

  Proses membangunkan partner-nya yang sedang blocked dengan signal pada variabel kondisi yang sedang ditunggu partner-nya. Signal terdapat dua versi yaitu:

  3. Versi Hoare

     Setelah signal, membangunkan proses baru agar berjalan dan menunda proses lain

  4. Versi Brinch Hansen

     Setelah melakukan signal, proses segara keluar dari monitor.

Keunggulan monitor atas semaphore adalah semua fungsi sinkronisasi dilakukan terpusat di monitor. Dengan ini lebih mudah memverifikasi kebenaran sinkronisasi dan mendeteksi kesalahan-kesalahan (bugs).

**- Kelemahan Metode Semaphore dan Monitors**

- Metode hanya menyelesaikan mutual-exclusion pada satu pemroses atau lebih yang mempunyai memori dipakai bersama (**Tighly Couple multiprocessor**)
- Metode tidak dapat diterapkan pada system terdistribusi berisi banyak pemroses dan memiliki memori sendiri (**loosely coupled multiprocessor**)

**- Penyelesaian dengan Metode Message Passing**

Message Passing dapat dilalukukan dengan beragam cara. Secara umum system message-passing menggunakan dua primitive, yaitu

- Send (destination, message), digunakan untuk mengirim pesan (message) ke tujuan tertentu (destination).
- Receive (source, message), digunakan untuk menerima pesan (message) dari sumber tertentu

Hal-hal yang perlu diperhatikan dalam merancang system message passing, yaitu

1. Sinkronisasi (synchronization)
2. Pengalamatan (addressing)
3. Format pesan (message format)
4. Disiplin antrian (queuing discipline)

Beberapa-beberapa masalah yang muncul pada system ini, yaitu

- Berurusan dengan hilangnya pesan di jaringan
- Berurusan dengan **authentication** yaitu menjamin komunikasi benar-benar dilakukan antar mesin-mesin yang diotorisasi
- Kinerja bila **message passing** diterapkan pada proses-proses pada mesin yang (uniprocessors)

Masalah-masalah pada perancangan dan implementasi system message passing

2. Masalah sinkronisasi proses
    - Pada proses pengiriman (send), dapat berupa

- § Blocking, yaitu setelah mengirim (send) maka proses di-blocked menunggu jawaban dari proses yang dikirimi
- § Non-blocking, yaitu setelah mengirim (send) maka proses tetap melanjutkan eksekusi intruksi-intruksi berikutnya

- Pada proses penerima (receive), dapat berupa
  - § Blocking, yaitu setelah menyatakan menerima (receive) maka proses di blocked menunggu kiriman pesan dari proses yang diinginkan
  - § Non-blocking, yaitu setelah menyatakan menerima maka proses tetap melanjutkan eksekusi intruksi-intruksi berikutnya
- Masalah yang muncul pada penerima adalah bagaimana mengimplementasikan pemeriksaan adanya kedatangan pesan

2 Masalah pengalamatan (addressing), terdapat beragam alternative, yaitu :

- Pengalamatan langsung (direct), berbeda untuk
  - § Prose pengirim
  - § Proses penerima, mempunyai alternative
    - v Pengalamatan eksplisit
    - v Pengalamatan implicit
- Pengalamatan tak langsung (direct)
  - § Pengalamatan statis
  - § Pengalamatan dinamis
  - § Pengalamatan kepemilikan

5 Masalah format pesan (message format) berupa

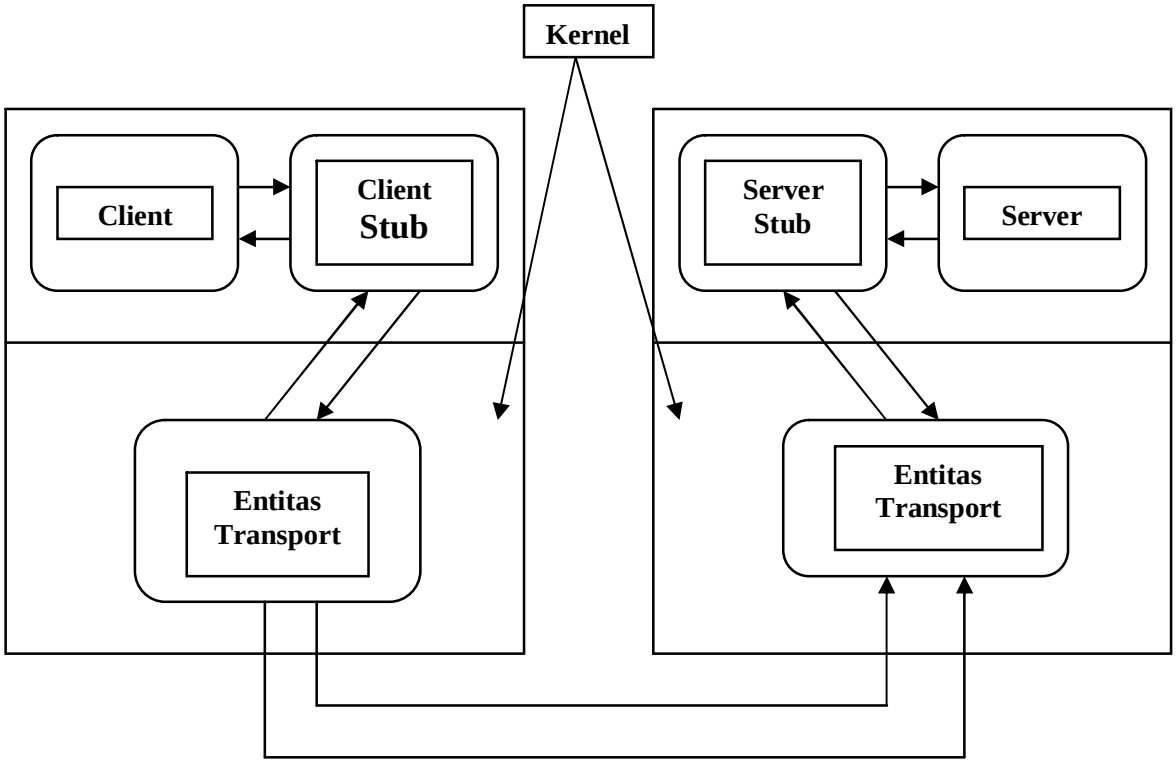
- Isi pesan (content)
- Panjang pesan mempunyai alternative
  - v Pesan dengan panjang tetap
  - v Pesan dengan panjang dapat beragam

6 Masalah disiplin antrian (queuing discipline), dalam mengolah pesan-pesan yang telah masuk dapat berupa

- FIFO
- Berprioritas

**- Penyelesaian dengan Metode Remote Procedure Call (RPC)**

Tujuan RPC adalah membuat prosedur jarak jauh (dikerjakan di mesin lain) seperti pemanggilan prosedur lokal. Dengan RPC, pemrogram akan menulis program seperti biasanya tanpa memperdulikan apakah prosedurnya dijalankan di pemroses setempat atau dijalankan si pemroses jauh.




Gambar skema eksekusi RPC

Panggilan remoter terdiri dari 10 langkah. ***Langkah 1*** berisi program client (atau prosedur) memanggil stub prosedur yang dilink di ruang alamatnya. Setelah pesan dibangun, pesan diberikanke lapisan transport untuk transmisi ***Langkah 2***, ***Langkah 3*** pada system Lan tak koneksi, entitas transport hanya mencatolkan header ke pesan dan meletakan ke jaringan tanpa banyak kerja lain, ***Langkah 4*** ketika pesan tiba di server, entitas transport melewatkan ke sub-serveryang me-unmarshal parameter-parameter ***Langkah 5*** Stub server kemudian memanggil prosedur server, ***Langkah 6*** Setelah

menyelesaikan kerja, prosedur server mengembalikan, ***Langkah 7*** Stub server kemudian me-marshaall hasil ke pesan dan melepaskan ke interface transport, ***Langkah 8*** Setelah jawaban tiba di mesin client, ***Langkah 9*** Jawaban ditangani stub client, ***Langkah 10*** stub client mengembalikan ke pemanggil, prosedur client. Suatu nilai yang dikembalikan server pada langkah 6 diberikan ke client dalam langkah 10.

**- Beberapa masalah untuk mengimplementasikan RPC antara lain**

4 Parameter passing terutama ***passing by reference***

5 Banyak waktu yang dihabiskan untuk konversi representasi internal pada mesin-mesin yang berbeda

6 Terdapat beberapa altenatif dalam mengartikan kegagalan pada RPC, semantics kegagalan dapat dikategorikan menjadi tiga, yaitu :

- **At least once,** menjamin server crash telah mengeksekusisekali atau lebih
- **At most once,** tak ada panggilan yang dieksekusi lebih dari satu kali
- **Maybe,** system tidak menjamin apapun, paling mudah diimplementasikan

**- Penyelesaian dengan Metode Rendezvouz Bahasa ADA**

Bahasa ADA merupakan bahasa pemrograman kongkuren pertama kali yang memakai rendezvouz sistem ini biasanya dipakai oleh militer. Rendezvouz merupakan teknik yang diajukan Hoare. Teknik ini secara efektif dapat menyelesaikan kombinasi masalah

- Mutual exclusion
- Sinkronisasi proses
- Komunikasi antar proses

# Tugas Sistem Oprasi

# Mutual Exclusion dan Sinkronisasi

Disusun oleh:

Haris H ( 1201027 )

Chandra P ( 1201009 )

SEKOLAH TINGGI MANAJEMEN IINFORMATIKA DAN KOPUTER

BANDUNG 2006/2007