MODUL SISTEM OPERASI

MUTUAL EXCLUSION

-Race Condition & Semaphore-

Pengantar

Beberapa proses terkadang membutuhkan sumber daya yang sama pada saat bersamaan. Sumber daya seperti ini disebut sumber daya kritis. Bagian program yang menggunakan sumber daya kritis disebut memasuki ***critical region/section***.

Pengantar

Hanya satu program pada saat yang diijinkan masuk *critical region*. Kondisi yang tidak dapat diprediksi hasilnya, bergantung pada proses-proses berjalan yang sedang bersaing disebut Kondisi Pacu ***(Race Condition)***.

Pengantar

Sistem operasi hanya menyediakan layanan (berupa *system call*) untuk mencegah proses masuk *critical section* yang sedang dimasuki proses lain.

Pengantar

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 proses lain.

Inilah yang dimaksud dengan mutual exclusion.

Definisi

Mutual Exclusion adalah suatu cara yang menjamin jika ada sebuah proses yang menggunakan variabel atau berkas yang sama (digunakan juga oleh proses lain), maka proses lain akan dikeluarkan dari pekerjaan yang sama.

Kriteria Penyelesaian

- *Mutual Exclusion* harus dijamin.
- Hanya satu proses pada satu saat yang diizinkan masuk *Critical Section/Region.*
- Proses yang berada di *noncritical section*, dilarang memblok proses-proses yang ingin masuk *critical section.*

Kriteria Penyelesaian

- Harus dijamin proses yang ingin masuk critical section tidak menunggu lama hingga waktu tak terhingga, agar tidak terjadi deadlock atau starvation.

Kriteria Penyelesaian

- Ketika ada proses di critical section maka proses yang ingin masuk critical section harus diijinkan segera masuk tanpa waktu tunda.
- Tidak ada asumsi mengenai kecepatan relative proses atau jumlah proses yang ada.

Metode Penyelesaian

1. **Metode Naif**

- Sebenarnya metode ini tidak menyelesaikan mutual exclusion, karena masih terdapat scenario proses yang membuat situasi kacau. Metode ini sering disebut **metode variable lock sederhana.**

Metode Penyelesaian

- Ketika proses hendak masuk critical section, proses lebih dulu memeriksa variable lock dengan ketentuan :
- Jika variable lock bernilai 0, proses mengeset variable lock menjadi 1 dan segera masuk critical section.
- Jika variable lock bernilai 1, proses menunggu sampai nilai variabel lock menjadi 0.

Metode Penyelesaian

2. Metode untuk situasi tertentu

- Metode ini sering disebut metode bergantian secara ketat yang mengasumsikan proses-proses yang hendak masuk critical section secara bergantian terus menerus.

Metode Penyelesaian

- Proses memeriksa terus menerus sehingga kondisi siap untuk diproses.
- Kondisi ini tidak dapat ditentukan lamanya waktu sehingga menyia-nyiakan waktu pemroses.
- Suatu saat kondisi akan crash ketika ada proses yang harus segera masuk sementara ada proses lain yang masih berjalan.

Metode Penyelesaian

3. Metode Busy Waiting

a. Algoritma Dekker mempunyai property-property berikut :

- Tidak memerlukan instruksi-instruksi perangkat keras khusus.
- Proses yang beroperasi di luar critical section tidak dapat mencegah proses lain memasuki critical section.
- Proses yang ingin masuk critical section akan segera masuk bila dimungkinkan

Metode Penyelesaian

3. Metode Busy Waiting

b. Metode Penyelesaian Peterson

Sebelum masuk critical section, proses memanggil *enter_critical_section*, namun sebelumnya proses memeriksa sampai kondisi aman. Terjadi busy waiting, setelah selesai proses menandai pekerjaan dan mengijinkan proses lain masuk. Keadaan awal tidak ada proses di critical section. Proses 0 akan masuk critical section. Proses menandai elemen arraynya dan mengeset turn ke 0. Proses memeriksa kondisi, dan prosedur *enter_critical_section* dilaksanakan. Jika kemudian, proses 1 akan masuk, proses akan menunggu sampai interest(0) menjadi FALSE. Kondisi ini hanya terjadi jika proses 0 mengeset elemen itu dan keluar dari critical section

Metode Penyelesaian

3. Metode Busy Waiting

c. Metode Pematian Interupsi

Proses mematikan interupsi ke pemroses dan segera masuk ke critical section. Proses kembali mengaktifkan interupsi segera setelah meninggalkan critical section. Metode ini mengakibatkan :

1. Pemroses tidak dapat beralih ke proses lain karena interupsi clock dimatikan sehingga penjadual pun tidak dieksekusi. Karena penjadual tidak beroperasi maka tidak terjadi alih proses.
2. Proses dapat memakai memori bersama tanpa takut terinvensi proses lain karena memang tidak ada proses lain yang dieksekusi saat itu.

Kelemahan utama :

1. Bila proses yang mematikan interupsi mengalami gangguan maka proses tidak akan pernah menghidupkan interupsi kembali. Kejadian ini mengakibatkan kematian seluruh system.
2. Jika terdapat dua pemroses atau lebih, mematikan interupsi hanya berpengaruh pada pemroses yang sedang mengeksekusi intruksi itu. Proses lain masih dapat memasuki critical section.

Metode Penyelesaian

3. Metode Busy Waiting

d.Metode Test and Set Lock (TSL)

Metode ini membaca isi memori ke register dan kemudian menyimpan nilai bukan 0 ke alamat memori. Pemroses yang mengeksekusi instruksi tsl mengunci bus memori, mencegah pemroses lain mengkases memori.

e.Metode Exchange (XCHG)

Metode ini menggunakan instruksi exchange (xchg). Instruksi xchg menukarkan dua isi memori.

f. Metode Instruksi Mesin

Keunggulan : Sederhana dan mudah diverifikasi, dapat diterapkan ke sembarang jumlah proses, dapat digunakan untuk mendukung banyak critical region.

Kelemahan :

Merupakan metode dengan busy waiting, sangat tidak efisien.

Adanya busy waiting memungkinkan terjadi deadlock dan starvation.

Metode Penyelesaian

4.Metode Penyelesaian Level Tinggi (Metode Semapore)

Dua proses atau lebih dapat bekerja sama dengan menggunakan penanda-penanda sederhana. Proses berhenti sampai proses memperoleh penanda tertentu. Variabel khusus untuk penandaan ini disebut semaphore. Semaphore mempunyai dua property :
Semaphore dapat diinisialisasi dengan nilai bukan negative.
Ada dua operasi terhadap semaphore yaitu **Operasi Up** dan **Operasi Down**.

a. **Operasi Down**

Operasi ini menurunkan nilai semaphore. Jika nilai semaphore menjadi bukan positif maka proses yang mengeksekusinya diblok. Operasi Down adalah atomic (atomic action), tidak dapat diinterupsi sebelum selesai. Menurunkan nilai, memeriksa nilai, menempatkan proses pada antrian dan memblok sebagai instruksi tunggal. Tidak ada proses lain yang dapat diakses sampai proses selesai.

Metode Penyelesaian

b. Operasi Up

Operasi ini menaikkan nilai semaphore. Jika satu proses atau lebih telah diblok pada suatu semaphore tidak dapat menyelesaikan operasi down maka salah satu dipilih oleh system dan dibolehkan menyelesaikan operasi downnya. Operasi Up menaikan nilai semaphore, memindahkan dari antrian dan menempatkan satu proses ke senarai ready tidak dapat diinterupsi.

Sebelum masuk critical section, proses melakukan down. Bila berhasil maka proses masuk critical section. Bila tidak berhasil maka proses diblok pada semaphore. Proses yang diblok dapat melanjutkan jika proses yang berada di critical section keluar dan melakukan operasi up dan menjadikan proses yang diblok menjadi ready dan berlanjut hingga operasi downnya berhasil

Implementasi Semaphore

1.Pematian Interupsi
Sistem operasi mematikan interupsi selagi memeriksa semaphore, memperbarui, dan menjadikan proses diblok. Karena semua aksi hanya memerlukan beberapa instruksi, pematian interupsi tidak merugikan.

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

KUIS

1. Jelaskan Keuntungan & Kelemahan Mutual Exclusion?
2. Jelaskan metode pembentukan Mutual Exclusion?
3. Jelaskan prinsip kerja Semaphore pada saat menjamin Mutual Exclusion?