SISTEM OPERASI

(Tugas 5)

Disusun Oleh:

Prames Ray Lapian - 140810210059

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2022

1. Apa Yang dimaksud Kongkurensi?

[JAWAB]

Kongkurensi merupakan sistem multiprocessing, multiprogramming, terdistribusi dan paralel yang mengharuskan adanya proses-proses yang berjalan bersama dalam waktu yang bersamaan

Kongkurensi juga merupakan ondisi dimana pada saat yang bersamaan terdapat lebih dari satu proses disebut dengan kongkurensi (proses-proses yang kongkuren).

Kongkuren dapat berdiri sendiri (independen) atau dapat saling berinteraksi, sehingga membutuhkan sinkronisasi atau koordinasi proses yang baik.

Kongkurensi merupakan kegiatan yang berhubungan dengan :

- Alokasi waktu pemroses untuk proses-proses yang aktif.
- Pemakaian bersama dan persaingan untuk mendapatkan sumber daya.
- Komunikasi antar proses.
- Sinkronisasi aktivitas banyak proses

2. Jelaskan tentang istilah berikut ini:

a. Critical Section

[JAWAB]

Critical Section merupakan bagian dari program dimana shared memori diakses, dimana setiap proses mempunyai “code” yang mengakses/ manipulasi shared data tersebut => “critical section”.

Atau bisa dikatakan Critical section adalah sebuah protokol di mana proses-proses dapat menggunakannya secara bersama-sama dengan setiap prosesnya harus ‘meminta izin’ untuk memasuki Critical Section-nya

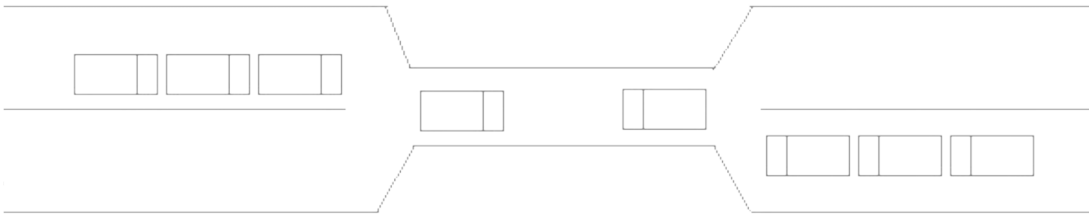
Manfaat dari Critical Section ini adalah Menjamin jika ada satu proses yang sedang “eksekusi” pada bagian “critical section” tidak ada proses lain yang diperbolehkan masuk ke “code” critical section dari proses tersebut.

b. Deadlock

[JAWAB]

Deadlock merupakan suatu kondisi dimana dua proses atau lebih saling menunggu proses yang lain untuk melepaskan resource yang sedang dipakai. Karena beberapa proses itu saling menunggu, maka tidak terjadi kemajuan dalam kerja proses-proses tersebut.

Deadlock juga merupakan masalah yang biasa terjadi ketika banyak proses yang membagi sebuah resource yang hanya boleh dirubah oleh satu proses saja dalam satu waktu.



Deadlock dianalogikan sebagai dua antrian mobil yang akan menyeberangi jembatan. Dalam kasus diatas, antrian di sebelah kiri menunggu antrian kanan untuk

mengosongkan jembatan (resource), begitu juga dengan antrian kanan. Akhirnya tidak terjadi kemajuan dalam kerja dua antrian tersebut.Dalam implementasinya misal ada proses A mempunyai resource X, proses B mempunyai resource Y. Kemudian kedua proses ini dijalankan bersama, proses A memerlukan resource Y dan proses B memerlukan resource X, tetapi kedua proses tidak akan memberikan resource yang dimiliki sebelum proses dirinya sendiri selesai dilakukan. Sehingga akan terjadi tunggu-menunggu.

c. Mutual Exclusion

[JAWAB]

Mutual Exclusion merupakan suatu kondisi jika satu proses sedang melakukan sesuatu, maka proses lain dicegah untuk tidak bisa melakukan proses yang sama terhadap data yang sama. Tujaun dari Mutual Exclusion ini adalah untnk mencegah terjadinya pengaksesan Critical Section oleh lebih dari satu proses.

d. Race Condition

[JAWAB]

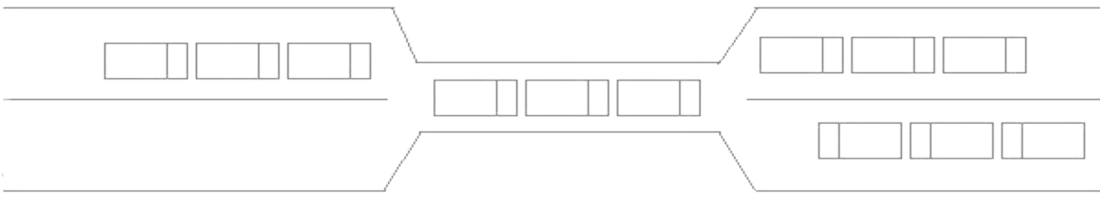
Race condition merupakan proses-proses yang bekerja bersama sering harus berbagi (storage) dimana masing-masing proses dapat membaca dan menulis shared data pada waktu yang sama.

Terjadi kondisi race (balapan) untuk mengakses satu lokasi yang sama dan hasil akhir tergantung pada proses yang berjalan tepat waktu.

e. Starvation

[JAWAB]

Starvation adalah kondisi yang biasanya terjadi setelah deadlock. Proses yang kekurangan resource (karena terjadi deadlock) tidak akan pernah mendapat resource yang dibutuhkan sehingga mengalami starvation (kelaparan). Namun, starvation juga bisa terjadi tanpa deadlock. Hal ini ketika terdapat kesalahan dalam sistem sehingga terjadi ketimpangan dalam pembagian resouce. Satu proses selalu mendapat resource, sedangkan proses yang lain tidak pernah mendapatkannya.



Ilustrasi starvation tanpa deadlock di dunia nyata dapat dilihat pada gambar diatas, pada antrian kanan terjadi starvation karena resource (jembatan) selalu dipakai oleh antrian kiri, dan antrian kanan tidak mendapatkan giliran.

3. Sebutkan beberapa cara untuk mencapai mutual exclusion dan jelaskan secara singkat!

[JAWAB]

I. Disabling Interrupt (Menon-aktifkan)

Interrupt di non-aktif-kan setelah memasuki critical section dan bisa lagi di interupsi sesaat sebelum meninggalkan Critical Section.

Kelemahan: Kurang bijaksana, karena ada kemungkinan lupa untuk mengaktifkan lagi. Disabling Interrupt berguna pada saat-saat tertentu untuk Kernel, tetapi untuk user kurang tepat.

II. Mengunci (lock) variable

Setiap proses yang akan mengakses ke critical section-nya harus memeriksa lock variable. Jika 0 berarti proses dapat memasuki critical section-nya dan jika 1 maka proses harus menunggu sampai lock variable = 0.

Kelemahannya adalah 2 proses masih dapat memasuki critical section-nya pada saat yang bersamaan. Sewaktu satu proses memeriksa lock variable = 0, pada saat akan men-set 1 ada interupsi untuk melaksanakan proses lain yang juga ingin memasuki critical sectionnya, maka akan terjadi race condition.

III. Strict Alternation

Dengan mengamati variable turn untuk menentukan siapa yang akan memasuki critical section-nya bukanlah ide yang baik jika proses lebih lambat dari yang lain.

IV. Solusi Peterson

Solusi Peterson merupakan kombinasi antara variabel turn dan lock variable Dimana proses tidak akan diteruskan sampai while terpenuhi, bila interested[other] = TRUE, maka proses akan menunggu sampai FALSE.

Kelemahannya: jika proses memanggil enter_region-nya secara hampir bersamaan, yang disimpan di turn adalah data yang ditulis terakhir.

V. Instruksi TSL (Test and Set Lock)

TSL membaca isi dari memory word ke dalam suatu register dan memberi nilai bukan nol pada alamat memori. Operasi pembacaan dan store ini dijamin tak terbagi, tidak ada prosesor lain dapat mengakses kata tersebut sampai instruksi selesai. CPU mengeksekusi instruksi TSL mengunci bus untuk melarang CPU lain mengakses memori sampai proses ini selesai.

Penyelesaian dengan solusi Peterson dan TSL keduanya benar tetapi memboroskan waktu CPU dengan harus memenuhi busy waiting.