Praktikum 11

Sinkronisasi Proses

**POKOK BAHASAN:**

Membuat program simuliasi tentang

* Bounded Buffer
* Reader-Writer
* Dining-Philosophers

**TUJUAN BELAJAR:**

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

* Memahami Sinkronisasi Proses.
* Membuat Aplikasi simulasi Bounded Buffer
* Membuat Aplikasi simulasi Reader-Writer
* Membuat Aplikasi simulasi Dining-Philosophers

**DASAR TEORI:**

Proses-proses yang konkuren adalah proses-proses (lebih dari satu) berada pada saat yang sama. Proses-proses ini dapat sepenuhnya tak bergantung dengan yang lainnya, tapi dapat juga saling berinteraksi. Proses-proses yang berinteraksi memerlukan sinkronisasi agar terkendali dengan baik.

Proses-proses yang melakukan akses secara konkuren pada data yang digunakan bersama-sama menyebabkan data tidak konsisten (inconsistence). Agar data konsisten dibutuhkan mekanisme untuk menjamin eksekusi yang berurutan pada proses-proses yang bekerja sama. Pada model shared memory untuk penyelesaian permasalahan bounded-buffer paling banyak menyimpan n – 1 item pada buffer pada saat yang bersamaan. Untuk mendapatkan solusi dimana semua N buffer digunakan bukan masalah yang sederhana. Misalnya dilakukan modifikasi kode producerconsumer dengan menambahkan variabel counter yang diinisialisasi 0 dan dinaikkan setiap satu item baru ditambahkan ke buffer.

1. **MASALAH-MASALAH KLASIK SINKRONISASI**

Untuk mengimplementasikan permasalahan sinkronisasi dapat menggunakan model yang digunakan untuk permasalahan Bounded Buffer, Reader Writer dan Dining Philosopher yang akan dijelaskan di bawah ini.

* 1. **Bounded-Buffer (Producer-Consumer) Problem**

Produsen menghasilkan barang dan konsumen yang akan menggunakannya. Ada beberapa batasan yang harus dipenuhi, antara lain :

- Barang yang dihasilkan oleh produsen terbatas

- Barang yang dipakai konsumen terbatas

- Konsumen hanya boleh menggunakan barang yang dimaksud setelah produsen menghasilkan barang dalam jumlah tertentu

- Produsen hanya boleh memproduksi barang jika konsumen sudah kehabisan barang

Untuk penyelesaian permasalahan bounded buffer menggunakan semaphore menggunakan variabel umum berikut :

*semaphore full, empty, mutex;*

Inisialisasi untuk variable diatas, full = 0, empty = n, mutex = 1. Struktur program untuk produsen adalah

*do {*

*…*

*menghasilkan item pada nextp*

*…*

*wait(empty);*

*wait(mutex);*

*…*

*menambah nextp ke buffer*

*…*

*signal(mutex);*

*signal(full);*

*} while (1);*

Sedangkan struktur program untuk konsumen adalah

*do {*

*wait(full)*

*wait(mutex);*

*…*

*mengambil item dari buffer ke nextc*

*…*

*signal(mutex);*

*signal(empty);*

*…*

*menggunakan item pada nextc*

*…*

*} while (1);*

* 1. **Reader and Writer Problem**

Terdapat dua variasi pada masalah ini, yaitu :

a. seorang reader tidak perlu menuggu reader lain untuk selesai hanya karena ada writer menunggu (reader memiliki prioritas lebih tinggi disbanding dengan writer)

b. Jika ada writer yang sedang menunggu, maka tidak boleh ada reader lain yang bekerja (writer memiliki prioritas yang lebih tinggi) Jika terdapat writer dalam critical section dan terdapat n reader yang menunggu, maka satu reader akan antri di wrt dan n-1 reader akan antri di mutex.

Jika writer mengeksekusi signal(wrt), maka dapat disimpulkan bahwa eksekusi adalah menunggu reader atau menunggu satu writer. Variabel umum yang digunakan adalah

*semaphore mutex, wrt;*

Inisialisasi variable diatas adalah *mutex = 1, wrt = 1, readcount = 0.*

Struktur proses writer adalah

*wait(wrt);*

*…*

*menulis*

*…*

*signal(wrt);*

Sedangkan struktur proses reader adalah

*wait(mutex);*

*readcount++;*

*if (readcount == 1)*

*wait(rt);*

*signal(mutex);*

*…*

*membaca*

*…*

*wait(mutex);*

*readcount--;*

*if (readcount == 0)*

*signal(wrt);*

*signal(mutex):*

* 1. **Dining-Philosophers Problem**

Permasalahan dining-philosophers digambarkan pada Gambar 5-2 dimana terdapat 5 filosof yang akan makan. Di sana disediakan 5 supit. Jika filosof lapar, ia akan mengambil 2 supit yaitu di tangan kanan dan kiri. Namun adakalanya hanya diambil supit satu saja. Jika ada filosof yang mengambil 2 supit, maka ada filosof yang harus menunggu sampai supit tersebut diletakkan. Hal ini dapat diimplementasikan dengan wait dan signal.



***Gambar 11-1 : Lima filosof dalam satu meja makan***

Struktur data yang digunakan untuk penyelesaian permasalahan ini dengan semaphore adalah

*semaphore chopstick[5];*

Dimana semua nilai array dinisialisasi 1. Struktur program untuk filosof ke i adalah

*do {*

*wait(chopstick[i])*

*wait(chopstick[(i+1) % 5])*

*…*

*makan*

*…*

*signal(chopstick[i]);*

*signal(chopstick[(i+1) % 5]);*

*…*

*berfikir*

*…*

*} while (1);*

Meskipun solusi ini menjamin bahwa tidak ada 2 tetangga yang makan bersamasama, namun masih mungkin terjadi deadlock, yaitu jika tiap-tiap filosof lapar dan mengambil supit kiri, maka semua nilai supit = 0, dan juka kemudian tiap-tiap filosof akan mengambil supit kanan, maka akan terjadi deadlock. Ada beberapa cara untuk menghindari deadlock, antara lain :

a. mengijinkan paling banyak 4 orang filosof yang duduk bersama-sama pada satu meja.

b. Mengijinkan seorang filosof mangambil supit hanya jika kedua supit itu ada (dengan catatan, bahwa ia harus mengambil pada critical section)

c. Menggunakan suatu solusi asimetrik, yaitu filosof pada nomor ganjil mengambil supit kanan dulu baru supit kiri. Sedangkan filosof yang duduk di kursi genap mengambil supit kanan dulu baru supit kiri.

TUGAS PENDAHULUAN:

1. Apa yang dimaksud dengan race condition?

2. Apakah yang dimaksud dengan critical section ? Untuk menyelesaikan masalah critical section , ada tiga hal yang harus dipenuhi, sebutkan dan jelaskan !

3. Bagaimana algoritma Bakery untuk sinkronisasi banyak proses (n proses) ?

4. Apa yang dimaksud semaphore dan sebutkan operasi pada semaphore

5. Bagaimana struktur semaphore yang digunakan untuk menyelesaikan permasalahan :

a. bounded buffer problem.

b. reader and writer problem.

c. dining philosopher problem.

PERCOBAAN:

1. Bukalah program (boleh menggunakan bahasa apa saja pemrograman yang sudah di pelajari)

**Percobaan 1 : bounded buffer**

1. Buatlah program untuk mengimplementasikan permasalahan bounded buffer pada permasalahan sebagai berikut:
   1. Buatlah sebuah program produser yang menghasilkan karakter acak dengan kecepatan yang dapat diatur, dan berhenti ketika buffer tersebut penuh
   2. Karakter tersebut disimpan didalam sebuah buffer yang kapasitasnya dapat diatur
   3. Buat pula progam yang berfungsi untuk menghapus karakter yang ada di buffer dengan kecepatan yang dapat atur

**Percobaan 2 : reader and writer**

1. Buatlah program untuk meNsimulasikan permasalahan reader dan writer

**Percobaan 3 dining philosopher**

1. Buatlah program untuk mesimulasikan permasalahan Dining Phisopher

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
2. Buatlah Program di atas.
3. Berikan kesimpulan dari praktikum ini.