KONGKURENSI (KEBERSAMAAN)

Haryono Setiadi, ST, M.Eng

OBJEK PEMBELAJARAN

- Overview Konkurensi
- Persaingan Dan Kerjasama Antar Proses
- Kesulitan-kesulitan dalam kongkurensi
- Masalah pada konkurensi
  - Mutual exclusion
  - Deadlock
  - Startvation
  - **Race condition**

Overview Konkurensi (1)

- Proses-proses disebut **kongkuren** jika proses-proses (lebih dari 1 proses) berada pada saat yang sama.
- Proses-proses kongkuren dapat sepenuhnya tak bergantung dengan lainnya tapi dapat juga saling berinteraksi/kerjasama.
- Proses-proses yang berinteraksi memerlukan sinkronisasi/koordinasi agar terkendali dengan baik.

Overview Konkurensi (2)

- Kongkurensi merupakan landasan umum perancangan sistem operasi.
- Perkembangan sistem komputer mendatang → multi-processing, multiprogramming, terdistribusi dan paralel → mengharuskan adanya proses-proses yang berjalan bersama dalam waktu yang bersamaan (konkuren)
- Untuk penanganan kongkuren, bahasa pemograman saat ini telah memiliki mekanisme kongkurensi dimana dalam penerapannya perlu dukungan sistem operasi.

- **Contoh Kasus**

  Sambil menunggu selesainya layanan (misalnya trasfer data oleh modem) pemakai dapat berinteraksi dengan aplikasi lain seperti aplikasi permainan game atau mengetikkan perintah pada text editor

  Proses tersebut harus berjalan konkuren dan tidak terjadi deadlock (hang)

Persaingan Dan Kerjasama Antar Proses (1)

- Persaingan antar proses terjadi ketika beberapa proses akan menggunakan sumber daya yang sama.
- Jika ada 2 proses yang akan mengakses ke suatu sumber daya tunggal, kemudian satu proses dialokasikan ke sumber daya tersebut oleh SO → proses yang lainnya akan menunggu.

Persaingan Dan Kerjasama Antar Proses (2)

- Pada kasus yang ekstrim, proses yang menunggu tersebut ada kemungkinan tidak akan pernah mendapatkan akses ke sumber daya sehingga tidak akan pernah selesai dengan sempurna.
- Hal ini juga terjadi akibat antar proses yang saling tidak peduli.
- Proses-proses yang mengalami kongkuren dapat berdiri sendiri (independen) atau dapat saling berinteraksi, sehingga membutuhkan sinkronisasi atau koordinasi proses yang baik.

- Meskipun proses-proses tidak bekerja bersama, SO perlu mengatur persaingan diantara proses-proses itu dalam memperoleh sumber daya yang terbatas

- Contoh :

  Dua buah aplikasi (word & corel) berusaha mengakses printer yang sama.

  Bila kedua aplikasi mengakses printer yang sama benar-benar secara bersamaan maka kedua proses akan memperoleh hasi yang tidak di hendaki.

Masalah pada konkurensi (1)

Beberapa masalah yang muncul pada konkurensi antara lain :

- Mutual exclusion
- Deadlock
- Startvation
- Race condition

Mutual exclusion

Masalah pada konkurensi (2)

Mutual exclusion adalah jaminan hanya satu proses yang mengakses sumber daya pada suatu interval waktu tertentu, sedangkan proses lain dilarang mengerjakan hal yang sama.

contoh : sumberdaya printer hanya bisa diakses 1 proses, tidak bisa bersamaan → sumber daya ini disebut sumber daya kritis

Deadlock

Masalah pada konkurensi (3)

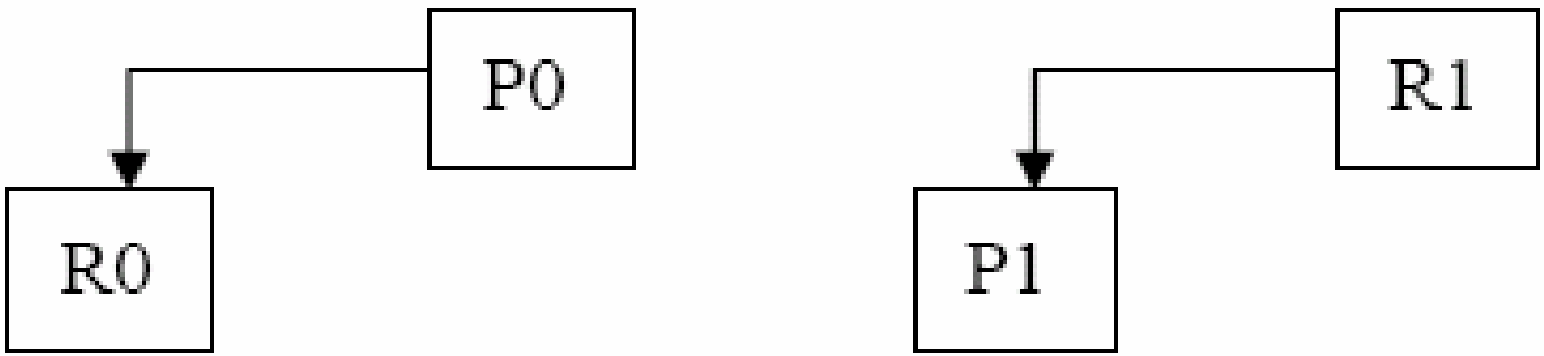
- **Deadlock**

  Adalah banyak proses yang saling menunggu hasil dari proses yang lain untuk dapat melanjutkan atau menyelesaikan tugasnya.

Masalah pada konkurensi (4)

Model deadlock 2 proses dan 2 sumber daya

- Misal : 2 proses P0 dan P1
- 2 sumber daya R0 dan R1
- P0 meminta sumberdaya R0.
- Sumber daya R1 dialokasikan ke P1.

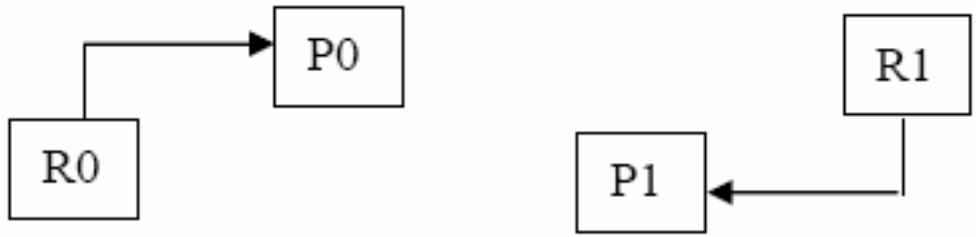


Masalah pada konkurensi (5)

Skenario yang menimbulkan deadlock :

P0 dialokasikan R0

P1 dialokasikan R1

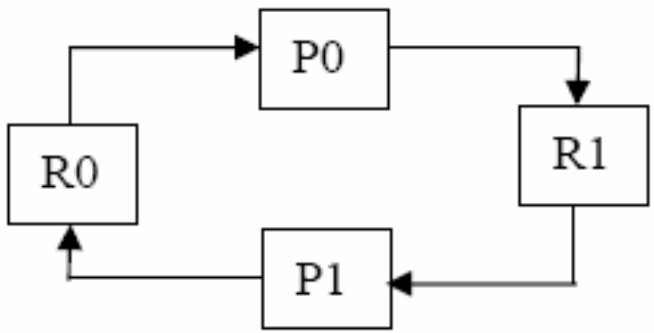


Kemudian

P0 sambil masih menggenggam R0, meminta R1

P1 sambil masih menggenggam R1, meminta R0

Terjadi deadlock karena sama-sama akan saling menunggu. Terjadinya deadlock ditandai munculnya graph melingkar.



Deadlock dapat terjadi dengan melibatkan lebih dari 2 proses dan 2 sumber daya.

Startvation

Masalah pada konkurensi (6)

- **Startvation**

  Adalah suatu proses akan menunggu suatu kejadian atau hasil suatu proses lain, supaya dapat menyelesaikan tugasnya, tetapi kejadian yang ditunggu tidak pernah terjadi karena selalu diambil lebih dulu oleh proses yang lain.

Contoh :Terdapat tiga proses, yaitu P1, P2 dan P3.

P1, P2 dan P3 memerlukan pengaksesan sumber daya R secara periodik

Skenario berikut terjadi :

- P1 sedang diberi sumber daya R sedangkan P2 dan P3 diblocked menunggu sumber daya R.
- Ketika P1 keluar dari critical section, maka P2 dan P3 diijinkan mengakses R.
- Asumsi P3 diberi hak akses, kemudian setelah selesai, hak akses kembali diberikan ke P1 yang saat itu kembali membutuhkan sumber daya R.

Jika pemberian hak akses bergantian terus-menerus antara P1 dan P3, maka P2 tidak pernah memperoleh pengaksesan sumber daya R.

Dalam kondisi ini memang tidak terjadi deadlock, hanya saja P2 mengalami starvation (tidak ada kesempatan untuk dilayani).

Race condition

Merupakan sebuah kondisi dimana 2 atau lebih proses membaca atau menulis data/variabel yang digunakan bersama,dan hasilnya tergantung dari proses mana yang terakhir menggunakan data tersebut.

Contoh : Aplikasi Bank

- Pada aplikasi tabungan, misal rekening A berisi Rp. 1 juta terdaftar di kantor cabang Solo.
- Pada suatu saat program aplikasi di kantor cabang Jakarta melayani penyetoran Rp.3 juta ke rekening tersebut.
- Program aplikasi membaca saldo akhir rekening A.
- Pada waktu yang hampir bersamaan di kantor cabang Solo juga terjadi transaksi yaitu penyetoran Rp. 5 juta ke rekening A.
- Program aplikasi di Solo membaca saldo Akhir rekening A

- Beberapa skenario yang terjadi bila mutual exclusion tidak terjamin, yaitu:

1. Program aplikasi Solo Menulis ke Rekening A secara cepat sehingga dihasilkan saldo 6.000.000, setelah itu kantor cabang jakarta menimpal hasil itu dengan saldo 4.000.000 bukan 10.000.000 (yang seharusnya)
2. Program aplikasi jakarta menulis kerekening secara cepat sehingga dihasilkan 4.000.000. Setelah itu program aplikasi di kantor cabang solo menimpa hasilnya itu dengan saldo 6.000.000,-

   Hasil yang lebih baik dibandingkan dengan skenario pertama ternyata masih di bawah hasil yang seharusnya.

END CHAPTER

NEXT CHAPTER 6 : DEADLOCK