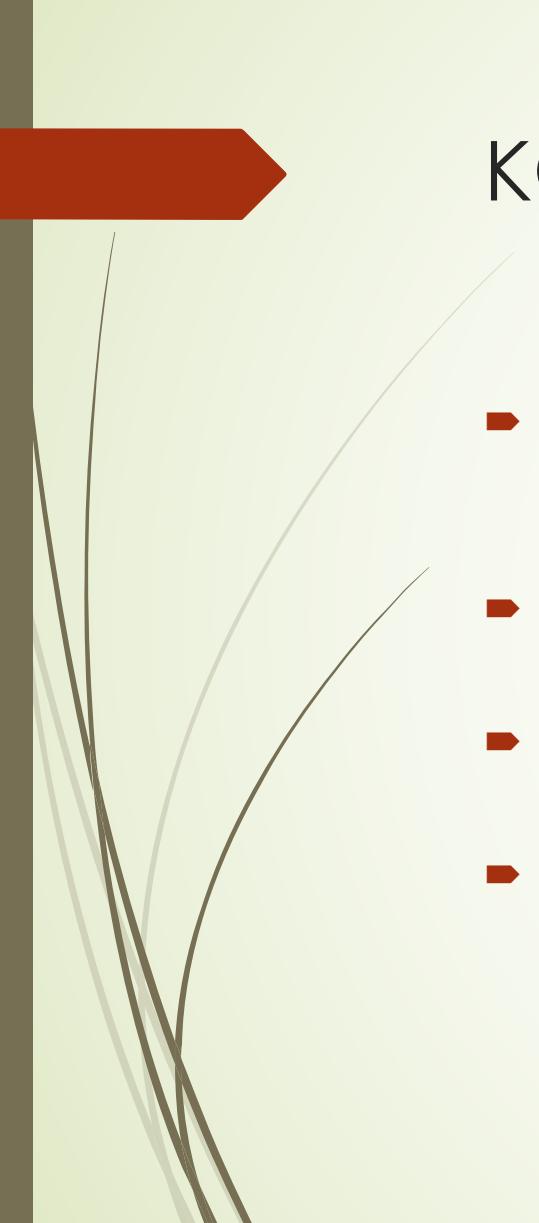




KONKURENSI

Willy Permana Putra



KONKURENSI ?

- ▶ Merupakan landasan umum perancangan sistem operasi. Proses-proses disebut konkuren jika proses-proses 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.



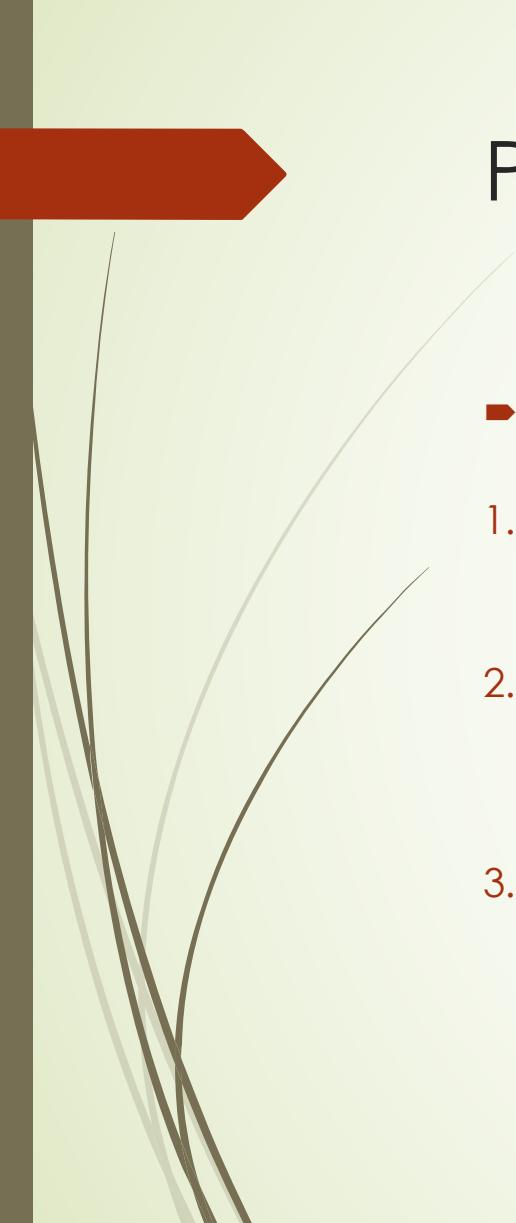
CRITICAL-SECTION

- ▶ Suatu sistem dimana semuanya berkompetisi menggunakan data yang digunakan bersama-sama
- ▶ Masing-masing proses mempunyai sebuah kode segmen yang disebut dengan **critical section**, dimana proses memungkinkan untuk mengubah variabel umum, mengubah sebuah tabel, menulis file dan lain sebagainya.
- ▶ Gambaran penting dari sistem adalah, ketika sebuah proses dijalankan di dalam **critical section**, tidak ada proses lain yang diijinkan untuk menjalankan critical section-nya.



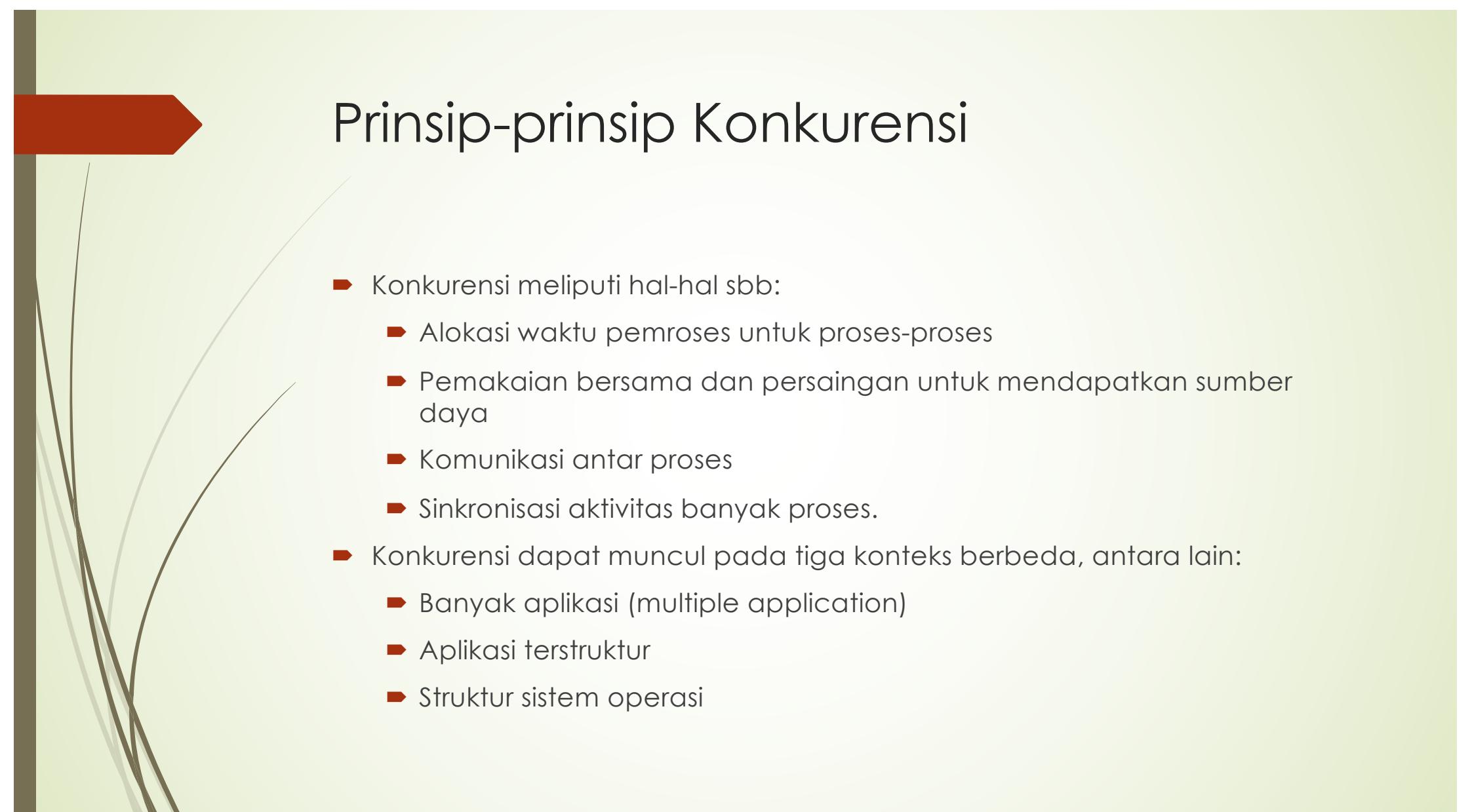
Lanjut...

- ▶ Eksekusi dari **critical section** oleh proses-proses tersebut berlaku eksklusif (*mutually exclusive*).
- ▶ Masing-masing proses harus meminta ijin untuk memasuki critical section-nya. Daerah kode yang mengimplementasikan perintah ini disebut daerah entry.



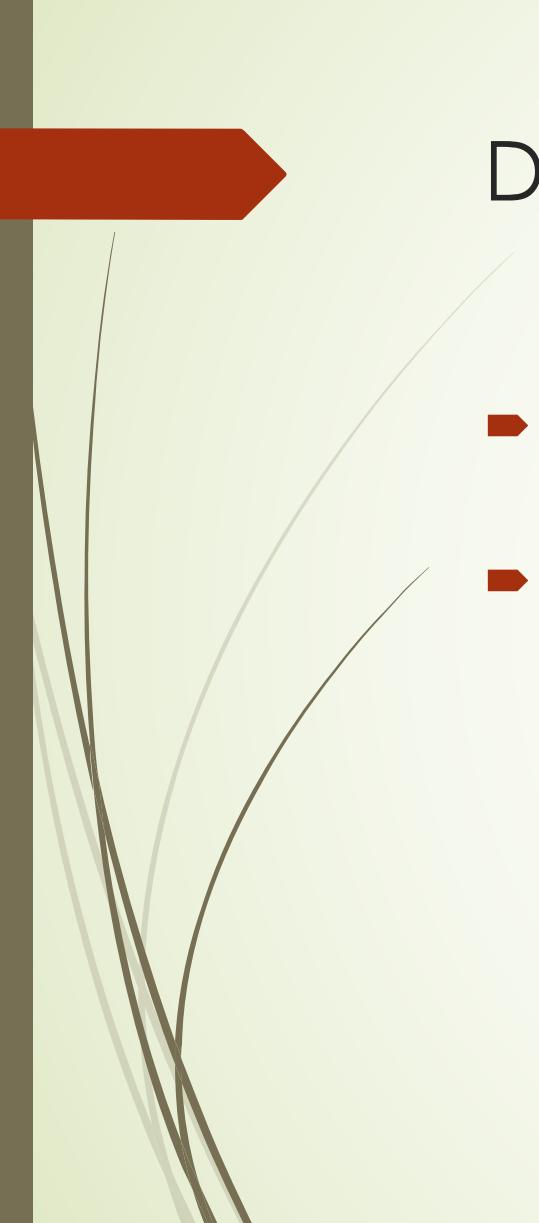
Proses Konkuren

- ▶ Pada proses-proses konkuren yang berinteraksi mempunyai beberapa masalah yang harus diselesaikan:
 1. Mutual Exclusion
 - ▶ Apabila proses menjalankan critical section-nya, maka tidak ada proses lain yang dapat menjalankan critical section.
 2. Proses
 - ▶ Apabila tidak ada proses yang menjalankan critical section-nya dan terdapat beberapa proses yang akan memasuki critical section-nya, maka hanya proses-proses itu yang tidak diproses
 3. Bounded Waiting
 - ▶ Terdapat batasan jumlah waktu yang diijinkan oleh proses lain untuk memasuki critical section setelah sebuah proses membuat permintaan untuk memasuki critical section-nya dan sebelum permintaan dikabulkan.



Prinsip-prinsip Konkurensi

- ▶ Konkurensi meliputi hal-hal sbb:
 - ▶ Alokasi waktu pemroses untuk proses-proses
 - ▶ Pemakaian bersama dan persaingan untuk mendapatkan sumber daya
 - ▶ Komunikasi antar proses
 - ▶ Sinkronisasi aktivitas banyak proses.
- ▶ Konkurensi dapat muncul pada tiga konteks berbeda, antara lain:
 - ▶ Banyak aplikasi (multiple application)
 - ▶ Aplikasi terstruktur
 - ▶ Struktur sistem operasi



Deadlock

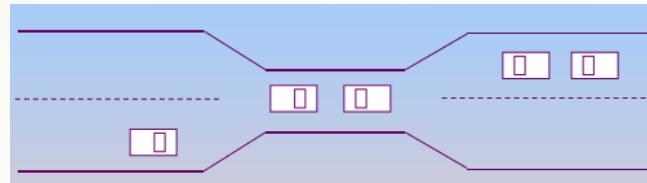
- ▶ Suatu kondisi dimana proses tidak berjalan lagi atau pun tidak ada komunikasi lagi antar proses.
- ▶ *Deadlock* disebabkan karena proses yang satu menunggu sumber daya yang sedang dipegang oleh proses lain yang sedang menunggu sumber daya yang dipegang oleh proses tersebut.

Permasalahan Deadlock

- ▶ Kondisi dimana masing-masing proses membawa resource dan saling menunggu mendapatkan resource yang dibawa proses lain
- ▶ Contoh
 - Sistem mempunyai 2 tape drive P1 dan P2 masing-masing membawa satu tape drive dan masing-masing memerlukan tape drive lainnya.

P_0	P_1
wait (A);	wait(B)
wait (B);	wait(A)

Contoh Jembatan Penyebrangan

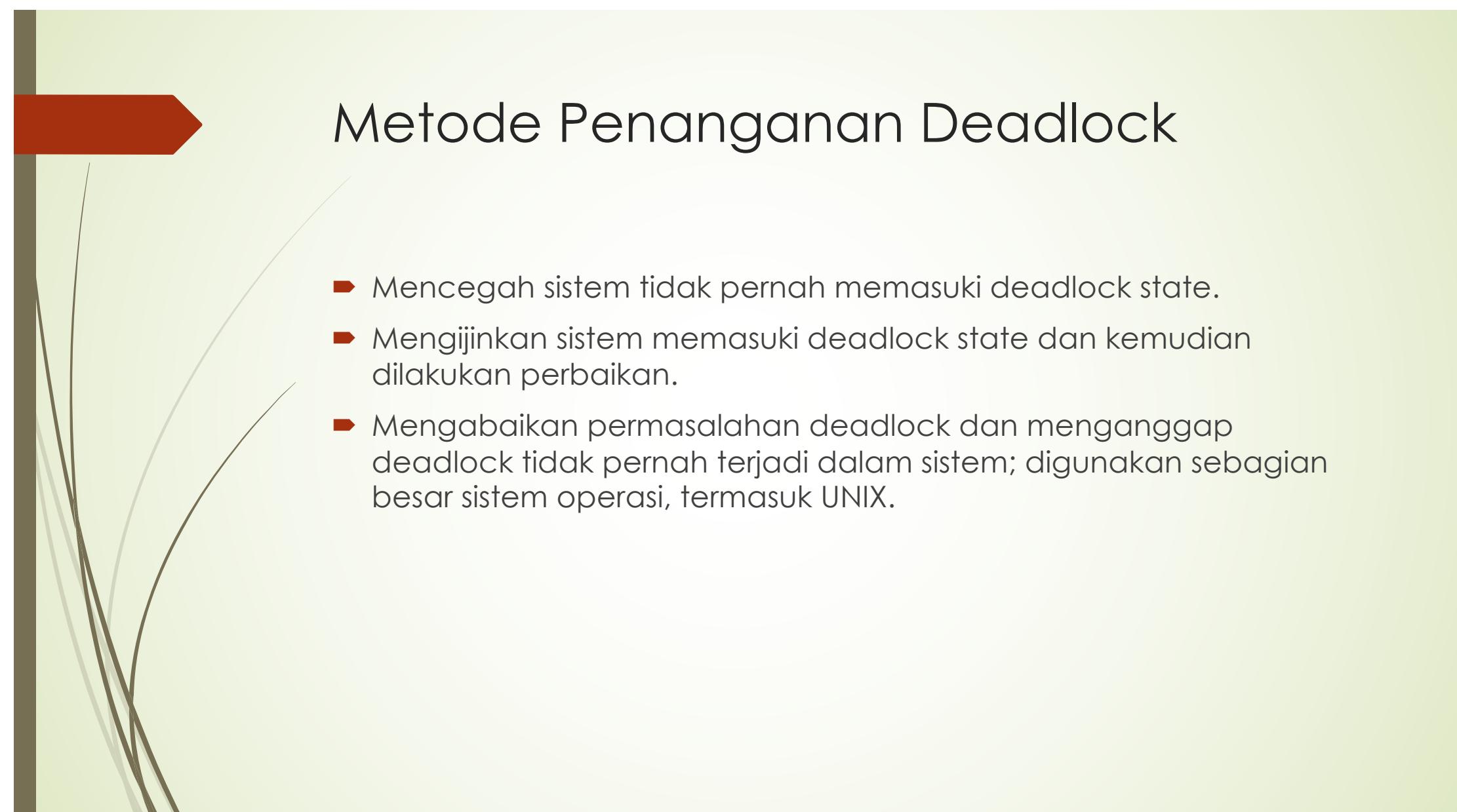


- ▶ Jalur hanya untuk satu arah
- ▶ Setiap bagian jembatan dianggap sebagai resource
- ▶ Jika terjadi deadlock, dapat dipecahkan jika satu mobil mundur (melepas resource dan rollback) mundur (melepas resource dan rollback)
- ▶ Beberapa mobil harus mundur jika terjadi deadlock
- ▶ Kemungkinan starvation

Karakteristik Deadlock

Deadlock dapat terjadi jika terdapat **4 kondisi** yang terjadi secara simultan:

- ▶ **Mutual exclusion:** hanya satu proses pada satu waktu yang dapat menggunakan satu resource.
- ▶ **Hold and wait:** sebuah proses membawa sedikitnya satu resource sedang menunggu mendapatkan resource tambahan yang dibawa oleh proses-proses lain
- ▶ **No preemption:** sebuah resource dapat dibebaskan hanya oleh proses yang membawanya, setelah proses menyelesaikan task/pekerjaan
- ▶ **Circular wait:** terdapat sekumpulan $\{P_0, P_1, \dots, P_n\}$ dari proses yang menunggu dimana P_0 menunggu resource yang dibawa oleh P_1 , P_1 menunggu resource yang dibawa oleh P_2 , ..., P_{n-1} menunggu resource yang dibawa oleh P_n , dan P_0 menunggu resource yang dibawa oleh P_0



Metode Penanganan Deadlock

- ▶ Mencegah sistem tidak pernah memasuki deadlock state.
- ▶ Mengijinkan sistem memasuki deadlock state dan kemudian dilakukan perbaikan.
- ▶ Mengabaikan permasalahan deadlock dan menganggap deadlock tidak pernah terjadi dalam sistem; digunakan sebagian besar sistem operasi, termasuk UNIX.



Pencegahan Deadlock

Mencegah dari kemungkinan 4 karakteristik deadlock.

- ◆ **Mutual Exclusion** – tidak tersedia resource yang dapat digunakan bersama-sama; semua proses membawa resource yang tidak dapat digunakan bersama-sama. → tidak dapat dicegah
- ◆ **Hold and Wait** – harus menjamin bahwa ketika sebuah proses meminta resource, proses tersebut tidak sedang membawa resource membawa resource
 - ◆ Sebelum eksekusi proses perlu meminta dan dialokasikan semua resource, atau memperbolehkan proses meminta resource hanya jika proses tidak membawa resource
 - ◆ Utilitas resource menjadi rendah; kemungkinan starvation



Lanjut...

► No Preemption–

- ◆ Jika sebuah proses membawa beberapa resource dan meminta resource lain yang tidak dapat segera dipenuhi meminta resource lain yang tidak dapat segera dipenuhi, maka semua resource yang sedang dibawa proses tersebut harus dibebaskan
- ◆ Resource yang dapat ditunda (preempt resource) ditambahkan ke daftar resource untuk proses yang menunggu
- ◆ Proses akan di-restart ketika proses hanya mendapatkan kembali resource lama setelah meminta resource baru.

► Circular Wait–memberlakukan pemesanan terlebih dahulu untuk total jenis resource yang dibutuhkan dan setiap proses meminta resource sesuai urutan nomor.



Contoh Deadlock

- ▶ State Selamat
- ▶ Contoh:
- ▶ Pada sistem dengan 10 sumber daya setipe, proses A memerlukan sumber daya maksimum sebanyak 10, sedang saat ini menggenggam 2. Proses B memerlukan sumber daya maksimum sebanyak 3, sedang saat ini menggenggam 1. Proses C memerlukan sumber daya maksimum sebanyak 7, sedang saat ini menggenggam 3. Maka, masih tersedia 4 sumber daya.

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	2	10
B	1	3
C	3	7
	Tersedia 4	

Dengan hati-hati, sistem penjadwalan ini dapat terhindarkan dari deadlock, dengan cara sbb:

Langkah 1. Alokasikan 4 sumber daya ke proses C sehingga sumber daya yang tersedia tinggal 1 dan nantikan sampai proses C berakhir.

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	2	10
B	1	3
C	7	7
	Tersedia 0	



Maka setelah proses C selesai dan menjadi :

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	2	10
B	1	3
C	0	0
Tersedia 7		



Langkah 2: Alokasikan 2 sumber daya ke proses B, nantikan proses B sampai berakhir.

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	2	10
B	3	3
C	0	0
Tersedia 5		

Maka, setelah proses B selesai dan menjadi:

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	2	10
B	0	0
C	0	0
Tersedia 8		

Langkah 3: Alokasikan 8 sumber daya ke proses A, nantikan proses A sampai berakhir.

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	10	10
B	0	0
C	0	0
	Tersedia 0	

Maka, setelah proses A selesai dan menjadi :

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	0	0
B	0	0
C	0	0
	Tersedia 0	

Maka, ketiga proses tersebut dengan penjadwalan hati-hati dapat menyelesaikan proses mereka dengan sempurna.

State Tak Selamat

Contoh

Soal di bawah ini adalah seperti soal pada state selamat, sate ini dapat berubah menjadi state tak selamat bila alokasi sumber daya tak terkendali.

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	2	10
B	1	3
C	3	7
Tersedia 4		

State tersebut dapat menjadi state tak selamat bila:

- dua permintaan sumber daya oleh proses A dilayani, kemudian
- Permintaan satu sumber daya oleh proses B dilayani



Maka State menjadi

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	4	10
B	2	3
C	3	7
Tersedia 1		

- ➡ Tabel di atas adalah state tak selamat. Pada state ini tidak terdapat barisan pengalokasian yang dapat membuat semua proses selesai

Langkah 1: Alokasikan satu sumber daya ke proses B sehingga sumber daya tersedia tinggal 1 dan nantikan sampai proses B berakhir.

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	4	10
B	3	3
C	3	7
Tersedia 0		



Maka, setelah proses B selesai dan menjadi:

Proses	Jumlah Resource digenggam	Maksimum Resource yang dibutuhkan
A	4	10
B	0	-
C	3	7
Tersedia 3		

- Saat hanya tersedia tiga sumber daya sementara dua proses yang sedang aktif masingmasing membutuhkan 6 dan 4 sumber daya.
- Perlu diingat!, bahwa state tak selamat bukan berarti deadlock, hanya menyatakan bahwa state tersebut memiliki kemungkinan deadlock.