Sistem Operasi

Komunikasi & Sinkronisasi Proses

2016

UNIVERSITAS Telkom

Concurrency

- Multiprogramming : Pengaturan banyak proses dengan menggunakan uniprocessor
- Multiprocessing : Pengaturan banyak proses dengan menggunakan multiprocessor
- Distributed Processing : Pengaturan banyak proses pada multiple, distributed computer.

- Concurrency menurut kamus : *existing, happening, or done at the same time*

UNIVERSITAS Telkom

Concurrency

- Menyebabkan permasalahan :
  - Bagaimana antara proses dapat saling berkomunikasi
  - Berbagi atau bersaing resource (memory, file, I/O, dsb) yang ada
  - Sinkronisasi

- Concurrency terjadi karena :
  - Multiple applications : mengizinkan multiprogramming yang melakukan dynamic shared untuk waktu prosesnya
  - Structured applications : program yang dibuat sudah terstruktur dan diatur untuk berjalan bersamaan
  - OS structure : OS sering kali diimplementasi sebagai sekumpulan thread

UNIVERSITAS Telkom

Concurrency

- Menimbulkan kesulitan :
  - Dikarenakan banyak proses yang dapat berjalan pada saat yang bersamaan, maka terdapat pula beberapa variable global yang digunakan bersamaan yang dapat di baca/tulis. Apalagi kalau proses baca/tulis nya dilakukan pada saat kritis
  - Sulit bagi OS untuk mengatur resource secara optimal. OS tidak bisa secara bebas mengunci resource yang akan digunakan proses lain.
  - Sangat sulit untuk mengalokasikan programming error

UNIVERSITAS Telkom

Contoh

```
void echo()
{
    chin = getchar();
    chout = chin;
    putchar(chout);
}
```

- Program disamping adalah program yang akan mendukung pemanggilan karakter dari keyboard dan akan ditampilkan ke layar komputer

UNIVERSITAS Telkom

Contoh – Single Processor

- Sangat sederhana apabila dijalankan.
- Akan mulai menjadi kendala apabila 2 buah proses menjalankan program yang sama pada saat yang bersamaan!

UNIVERSITAS Telkom

Contoh

- Proces P1 memanggil prosedur **echo** dan langsung diinterupsi setelah getchar mengembalikan nilai dan menyimpan di chin. Pada saat ini, karakter yang dimasukkan adalah X, dan tersimpan pada variable chin.
- Proces P2 mulai dijalankan dan memanggil prosedur echo, yang kemudian jalan dan menampilkan karakter Y pada layar.
- Proses P1 dilanjutkan lagi. Pada saat ini variable chin telah berubah dikarenakan P2 merubah nilainya, maka pada layar akan ditampilkan karakter Y

- Y muncul 2x dilayar! Bagaimana solusinya?

UNIVERSITAS Telkom

Contoh - Multiprocessor

Hasilnya???

Process P1

```
•
chin = getchar();
•
chout = chin;
putchar(chout);
•
•
```

Process P2

```
•
•
chin = getchar();
chout = chin;
•
putchar(chout);
•
```

Sama saja…

UNIVERSITAS Telkom

Solusi?

- Blocking!
- Saat P1 sedang menjalankan echo, salah satu proses harus mengalah, tidak mengerjakan prosesnya terlebih dahulu.

UNIVERSITAS Telkom

Race Condition

- Dua atau lebih proses mengakses shared data dan hasil akhir berubah sesuai dengan urutan mereka bekerja.
- Daerah Kritis: Bagian dari code yang mengakses shared data di kondisi race.

UNIVERSITAS Telkom

Race Condition

- Contoh : P3 dan P4 akan dikerjakan pada saat yang bersamaan, P3 memiliki rumus : b = b + c; dan P4 memiliki rumus : c = b + c. Inisialisasi b = 1 dan c =2. Berapakah hasil akhir yang didapatkan apabil P3 berjalan lebih dulu dari P4? Bagaimana sebaliknya?

UNIVERSITAS Telkom

Proses Interaction

Degree of Awareness	Relationship	Influence That One Process Has on the Other	Potential Control Problems
Processes unaware of each other	Competition	• Results of one process independent of the action of others • Timing of process may be affected	• Mutual exclusion • Deadlock (renewable resource) • Starvation
Processes indirectly aware of each other (e.g., shared object)	Cooperation by sharing	• Results of one process may depend on information obtained from others • Timing of process may be affected	• Mutual exclusion • Deadlock (renewable resource) • Starvation • Data coherence
Processes directly aware of each other (have communication primitives available to them)	Cooperation by communication	• Results of one process may depend on information obtained from others • Timing of process may be affected	• Deadlock (consumable resource) • Starvation

UNIVERSITAS Telkom

Kompetisi Resources - Mutual Exclusion

- Concurrent proses menyebabkan konflik dimana beberapa proses membutuhkan resource yang sama, hal ini bisa saja terjadi apabila antar proses tidak aware dengan proses yang lainnya.
- Dengan menggunakan mutual exclusion, kita membuat kondisi kritikal dimana tidak ada yang bisa menginterupsi. Sehingga tidak ada proses lain yang dapat mengambil/merubah resource yang sedang digunakan.

UNIVERSITAS Telkom

Kompetisi Resources - Mutual Exclusion

- Dapat menyebabkan deadlock.
- OS menugaskan R1 ke P2, dan R2 ke P1. Masing-masing proses saling menunggu satu resource lagi, dimana tidak ada proses yang mau melepaskan resource yang sudah dimiliki, dan melakukan operasi yang membutuhkan kedua resource tersebut. Dan dua proses tersebut dapat dikatakan deadlock.

UNIVERSITAS Telkom

Kompetisi Resources - Mutual Exclusion

- Dapat menyebabkan starvation.
- 3 proses, P1, P2, dan P3 dikerjakan. P1 memasuki area kritis, P2 dan P3 ditunda. Ketika P1 selesai dari area kritis, P2 dan P3 boleh menggunakan resource yang ada. Asumsikan OS memilih P3 untuk dikerjakan. Dan pada saat P3 belum selesai dari area kritisnya, P1 sudah antri lagi. Ketika P3 selesai dari masa kritisnya, P1 dan P2 dapat berjalan. Apabila OS memilih mengerjakan P1, maka P2 dapat dikatakan starvation

UNIVERSITAS Telkom

Kerjasama - Sharing

- Pada kerjasama, biasanya melakukan sharing variabel dan sebagainya.

Contoh :

Dua buah variable a dan b harus menjaga nilainya sehingga a = b. Terdapat 2 buah proses yang akan dijalankan P1 dan P2.

- P1 : (1) a = a + 1; (2) b = b + 1
- P2 : (1) b = 2 * b; (2) a = 2 * a

Apa yang terjadi apabila secara berurutan dikerjakan P1(1), P2(1), P1(2) dan P2(2)??

UNIVERSITAS Telkom

Kerjasama - Komunikasi

	Person A	Person B
3:00	Look in fridge. Pizza!	
3:05	Leave for store.	Look in fridge. Pizza!
3:10	Arrive at store.	Leave for store.
3:15	Buy pizza.	Arrive at store.
3:20	Arrive home.	Buy pizza.
3:25	Put away pizza.	Arrive home.
3:30		Put pizza away.
		Oh no!

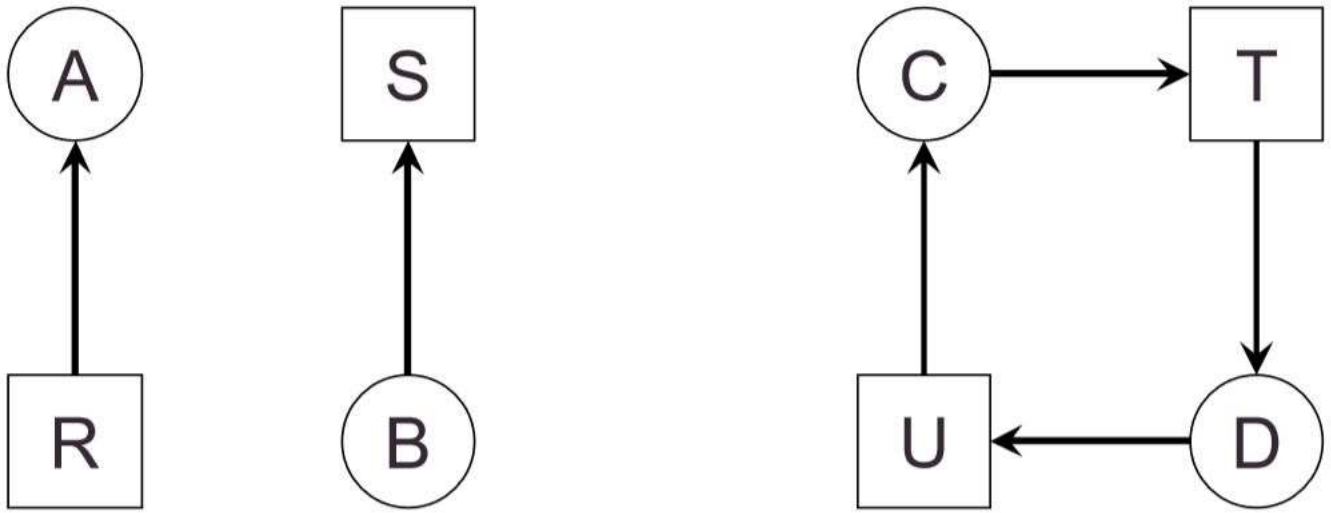
Deadlock

- **Deadlock –** dua atau lebih proses menunggu tak terhingga untuk sebuah even yang bisa disebabkan oleh salah satu proses yang menunggu.
- Misal *S* dan *Q* adalah dua semaphore diinitialisasi ke 1

P_0	P_1
wait(*S*);	*wait*(*Q*);
wait(*Q*);	*wait*(*S*);
⋮	⋮
signal(*S*);	*signal*(*Q*);
signal(*Q*)	*signal*(*S*);

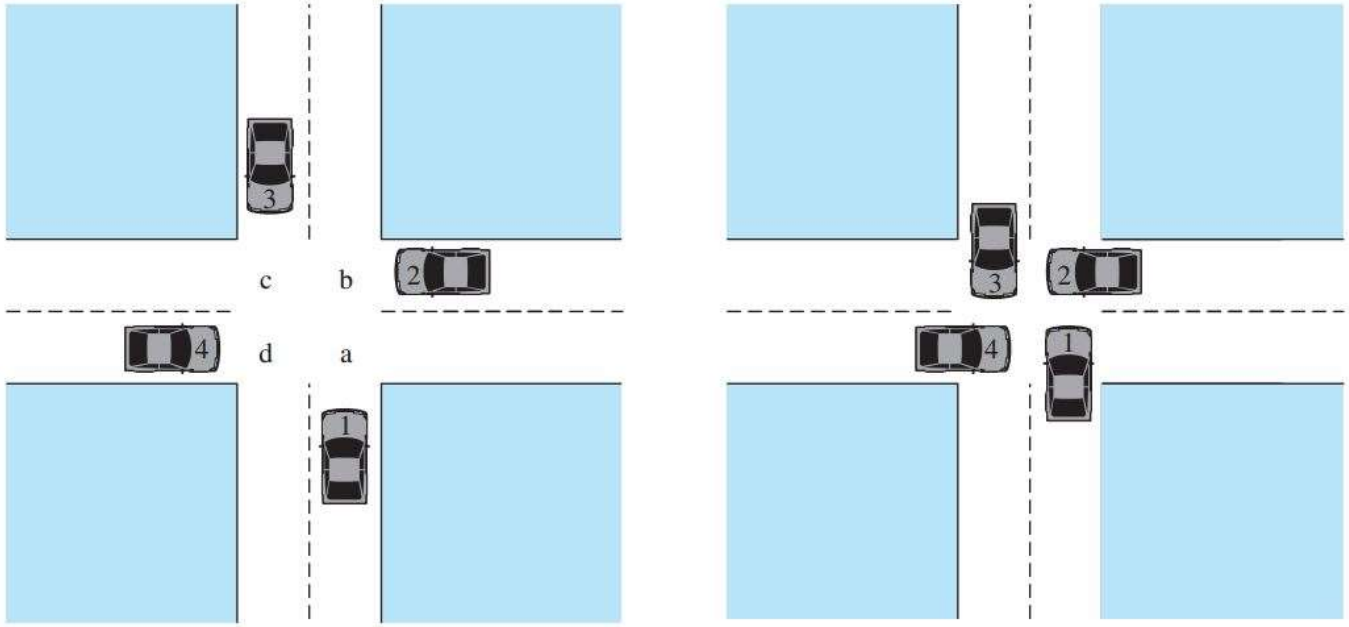
UNIVERSITAS Telkom

Deadlock



Resource R ditugaskan ke proses A
Proses B menunggu/req resource S
Proses C dan D deadlock dikarenakan resource T dan U

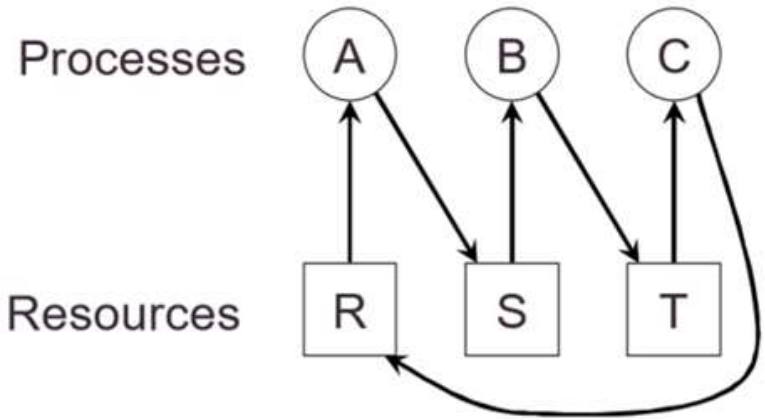
Deadlock



(a) Deadlock possible

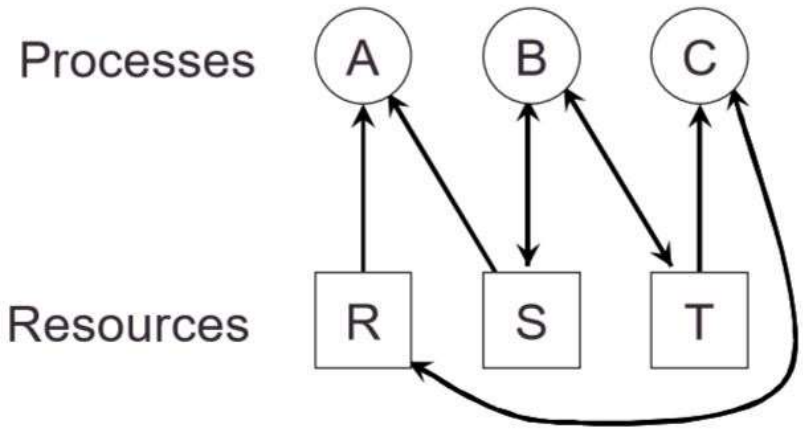
(b) Deadlock

Terjadinya Deadlock



- A requests R
- B requests S
- C requests T
- A requests S
- B requests T
- C requests R

Terjadinya Deadlock



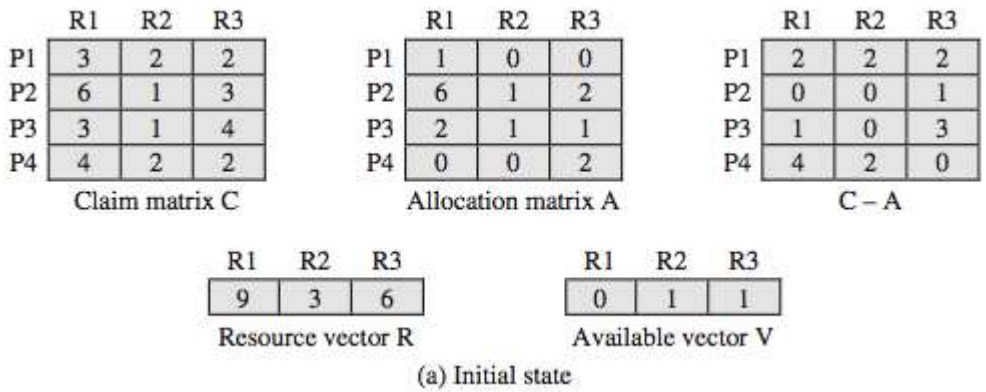
- A requests R
- C requests T
- A requests S
- B requests S
- B requests T
- C requests R
- A releases S
- A releases R
- C releases R
- C releases T

Deadlock

- Prevention : Mutual exclusion, Hold and Wait, No Preemption, Circular Wait
- Avoidance : Process Initiation Denial, Resource Allocation Denial (banker's algorithm)
- Detection : Deadlock Detection Algorithm, Recovery

UNIVERSITAS Telkom

Banker's Algorithm – Safe State



	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

C – A

R1	R2	R3
9	3	6

Resource vector R

R1	R2	R3
0	1	1

Available vector V

(a) Initial state

Banker's Algorithm – Safe State

	R1	R2	R3
P1	3	2	2
P2	0	0	0
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	1	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	2	2	2
P2	0	0	0
P3	1	0	3
P4	4	2	0

C – A

R1	R2	R3
9	3	6

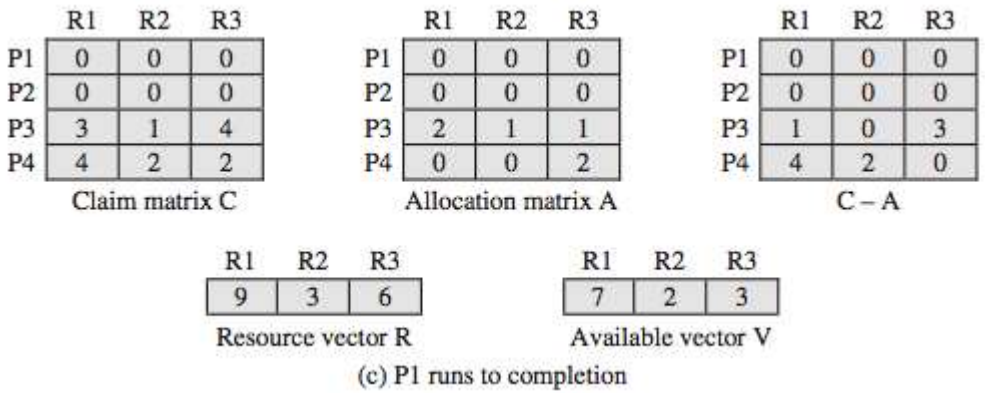
Resource vector R

R1	R2	R3
6	2	3

Available vector V

(b) P2 runs to completion

Banker's Algorithm – Safe State



	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	1	0	3
P4	4	2	0

C – A

R1	R2	R3
9	3	6

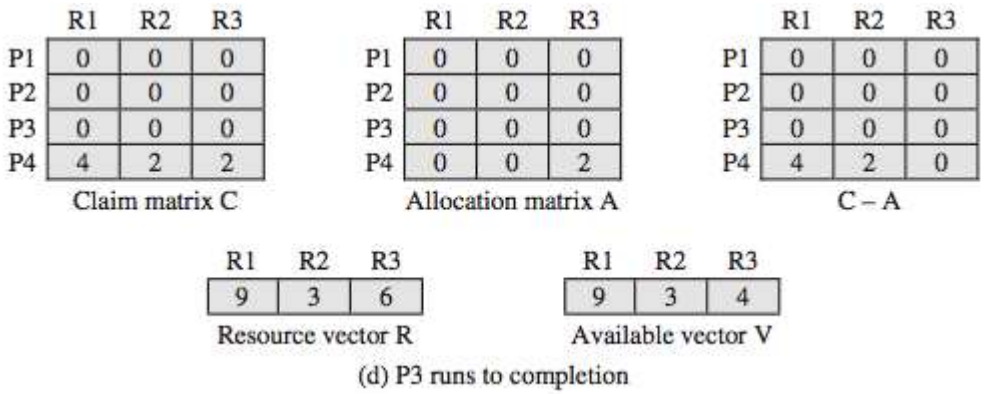
Resource vector R

R1	R2	R3
7	2	3

Available vector V

(c) P1 runs to completion

Banker's Algorithm – Safe State



	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	0

C – A

R1	R2	R3
9	3	6

Resource vector R

R1	R2	R3
9	3	4

Available vector V

(d) P3 runs to completion

Banker's Algorithm – Unsafe State

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	1	0	0
P2	5	1	1
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	2	2	2
P2	1	0	2
P3	1	0	3
P4	4	2	0

C – A

R1	R2	R3
9	3	6

Resource vector R

R1	R2	R3
1	1	2

Available vector V

(a) Initial state

Banker's Algorithm – Unsafe State

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	2	0	1
P2	5	1	1
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	1	2	1
P2	1	0	2
P3	1	0	3
P4	4	2	0

C – A

R1	R2	R3
9	3	6

Resource vector R

R1	R2	R3
0	1	1

Available vector V

(b) P1 requests one unit each of R1 and R3

Deadlock Detection Algorithm

1. Mark each process that has a row in the Allocation matrix of all zeros.
2. Initialize a temporary vector **W** to equal the Available vector.
3. Find an index i such that process i is currently unmarked and the ith row of **Q** is less than or equal to **W**. That is, $Q_{ik} \leq W_k$, for $1 \leq k \leq m$. If no such row is found, terminate the algorithm.
4. If such a row is found, mark process i and add the corresponding row of the allocation matrix to **W**. That is, set $W_k = W_k + A_{ik}$, for $1 \leq k \leq m$. Return to step 3.

UNIVERSITAS Telkom

Deadlock Detection Algorithm

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Request matrix Q

	R1	R2	R3	R4	R5
P1	1	0	1	1	0
P2	1	1	0	0	0
P3	0	0	0	1	0
P4	0	0	0	0	0

Allocation matrix A

R1	R2	R3	R4	R5
2	1	1	2	1

Resource vector

R1	R2	R3	R4	R5
0	0	0	0	1

Available vector

Deadlock Detection Algorithm

1. Mark P4, because P4 has no allocated resources.
2. Set $\mathbf{W} = (0\,0\,0\,0\,1)$.
3. The request of process P3 is less than or equal to $\mathbf{W}$, so mark P3 and set $\mathbf{W} = \mathbf{W} + (0\,0\,0\,1\,0) = (0\,0\,0\,1\,1)$.
4. No other unmarked process has a row in $\mathbf{Q}$ that is less than or equal to $\mathbf{W}$. Therefore, terminate the algorithm.

The algorithm concludes with P1 and P2 unmarked, indicating that these processes are deadlocked.

UNIVERSITAS Telkom

Recovery

1. Batalkan semua proses yang deadlock! Solusi umum yang terjadi di OS
2. Backup semua proses ke proses sebelum terjadi deadlock. Proses ini membutuhkan sistem melakukan mekanisme rollback dan restart. Resiko yang akan terjadi adalah deadlock akan kembali terjadi
3. Batalkan proses yang menyebabkan deadlock, ulangi sampai deadlock dapat dihindari. Algoritma deteksi harus dapat dijalankan dengan baik.
4. Jalankan kembali proses-proses yang ada tanpa terjadi deadlock

UNIVERSITAS Telkom

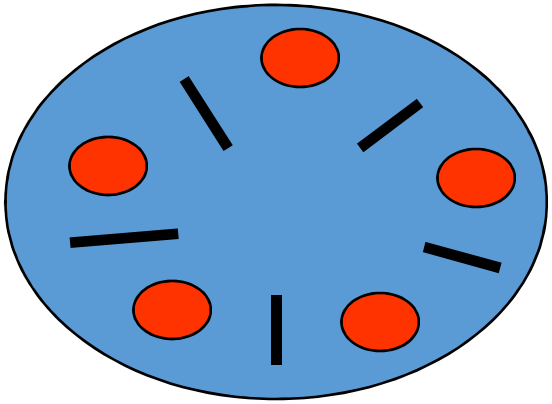
Recovery

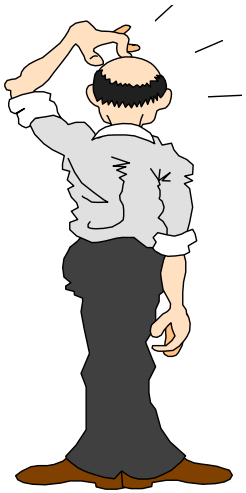
- Pilih salah satu untuk nomor 3 dan 4:
  - least amount of processor time consumed so far
  - least amount of output produced so far
  - most estimated time remaining
  - least total resources allocated so far
  - lowest priority

UNIVERSITAS Telkom

Filsuf Makan Malam

- Filsuf
  - Berpikir
  - Duduk
  - Makan
  - Berpikir
- Butuh 2 sumpit untuk makan





Solusi Lain

- Biarkan paling tidak N-1 untuk duduk pada satu saat
- Mengambil sumpit hanya diperbolehkan jika keduanya tersedia

UNIVERSITAS
Telkom