

MODUL PRAKTIKUM SISTEM OPERASI

PRAKTIKUM VIII

Deadlock

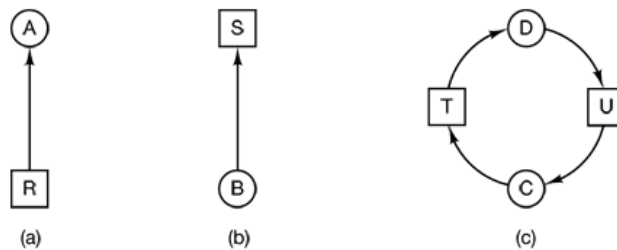
A. Tujuan

Pada akhir praktikum ini, peserta dapat:

1. Menggunakan graf pengalokasi sumber daya (*resource allocation graph*) untuk mendeteksi deadlock.
2. Menggunakan simulator untuk membuat situasi deadlock.
3. Menggunakan beberapa metode untuk mengatasi deadlock.

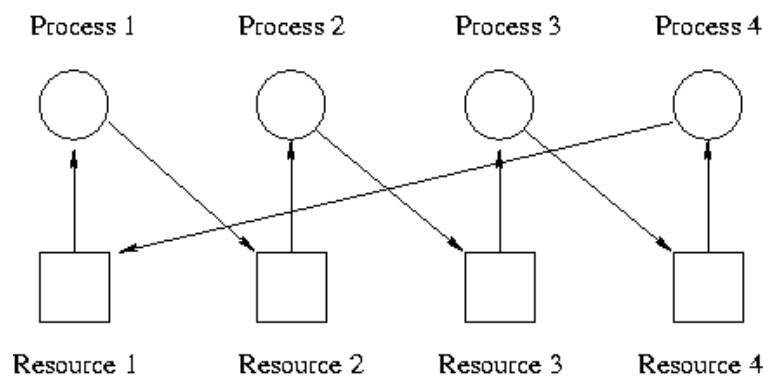
B. Dasar Teori

Deadlock yang terjadi ketika proses-proses membutuhkan akses ke lebih dari satu sumber daya (yang tidak dapat dipakai bersama) sering terjadi pada sistem operasi modern.



Keterangan:

- a. Proses A sedang memakai (allocate) sumber daya R
- b. Proses B meminta (request) sumber daya S
- c. Terjadi deadlock, karena proses C meminta sumber daya T yang sedang dipakai oleh D, namun D tidak (akan) melepas D karena sedang meminta sumber daya U yang dipakai oleh C.



Kasus Deadlock
dengan empat proses dan empat sumber daya

Syarat-syarat terjadinya deadlock :

1. *Mutual exclusion condition*
Tiap sumber daya saat itu diberikan pada tepat satu proses / proses mengklaim kontrol eksklusif terhadap sumber daya yang dibutuhkannya.
2. *Hold and wait condition*
Proses-proses yang sedang menggenggam sumber daya yang telah dialokasikan untuknya sementara menunggu sumber daya – sumber daya tambahan yang baru.
3. *Non-preemption condition*
Sumber daya – sumber daya yang sebelumnya diberikan tidak dapat diambil paksa dari proses sampai sumber daya tersebut digunakan sampai selesai. Sumber daya - sumber daya harus secara eksplisit dilepaskan dari proses yang menggenggamnya.
4. *Circular wait condition*
Harus terdapat rantai sirkuler/ satu lingkaran proses dari dua proses atau lebih dengan setiap proses memegang satu atau lebih sumber daya yang diminta oleh proses berikutnya pada lingkaran tersebut, masing-masing menunggu sumber daya yang digenggam oleh anggota berikutnya pada rantai itu.

Tiga syarat yang pertama merupakan syarat perlu bagi terjadinya *deadlock*. Keberadaan *deadlock* selalu berarti terpenuhi kondisi-kondisi diatas, tidak mungkin terjadi *deadlock* bila tidak ada ketiga kondisi tersebut. Jika terjadi *deadlock* berarti terdapat tiga kondisi itu, tetapi keberadaan ketiga kondisi itu belum tentu terjadi *deadlock*.

Deadlock baru benar-benar terjadi bila syarat keempat terpenuhi. Kondisi keempat merupakan keharusan bagi terjadinya peristiwa *deadlock*. Bila salah satu dari empat kondisi tidak terpenuhi maka *deadlock* tidak terjadi.

Metode-metode mengatasi deadlock :

1. *deadlock prevention*
pengkondisian sistem agar menghilangkan kemungkinan terjadinya *deadlock*. Pencegahan merupakan solusi yang bersih dipandang dari sudut tercegahnya *deadlock*. Jika mulainya satu atau lebih proses akan menyebabkan terjadinya *deadlock*, proses tersebut tidak akan dimulai sama sekali.
 - tiap proses harus meminta resource yang dibutuhkan sekaligus dan tidak bisa berjalan sampai semua di dapat (untuk “wait for”)
 - jika ada resource yang tidak terpenuhi, yang lainnya harus dilepas (untuk “no preemption”)
 - urutkan tipe-tipe resource secara linier / linier ordering (untuk “circular wait”)
2. *deadlock avoidance*
menghindarkan kondisi yang paling mungkin menimbulkan *deadlock* agar memperoleh sumber daya lebih baik. Penghindaran bukan berarti menghilangkan semua kemungkinan terjadinya *deadlock*. Jika sistem operasi mengetahui bahwa alokasi sumber daya menimbulkan resiko *deadlock*, sistem menolak / menghindari pengaksesan itu. Dengan demikian menghindari terjadinya *deadlock*. Contohnya dengan menggunakan algoritma Banker yang diciptakan oleh Dijkstra.
3. *deadlock detection and recovery*
deteksi digunakan pada sistem yang mengijinkan terjadinya *deadlock*, dengan memeriksa apakah terjadi *deadlock* dan menentukan proses dan sumber daya yang terlibat *deadlock* secara presisi. Begitu telah dapat ditentukan, sistem dipulihkan dari *deadlock* dengan metode pemulihan. Metode pemulihan dari *deadlock* sistem sehingga beroperasi kembali, bebas dari *deadlock*. Proses yang terlibat *deadlock* mungkin dapat menyelesaikan eksekusi dan membebaskan sumber dayanya.

Pencegahan deadlock :

1. **Disallow hold and wait.** Jika proses (telah) sedang memegang sumber daya tertentu, untuk permintaan berikutnya proses harus melepas dulu sumber daya yang dipegangnya.
2. **Disallow circular wait.** Tiap proses harus meminta semua sumber daya yang diperlukan sekaligus dan tidak berlanjut sampai semuanya diberikan.
3. **Total ordering.** Memberi nomor urut terhadap tipe-tipe sumber daya pada semua proses, sehingga jika proses telah memegang suatu tipe sumber daya, maka proses tersebut berikutnya hanya boleh meminta tipe sumber daya pada urutan yang lebih besar.

C. Langkah-langkah Praktikum

1. Pada kertas, buatlah suatu siklus *resource allocation graph* untuk 4 proses (P1, P2, P3, dan P4), dan 4 sumber daya yang tersedia (R0, R1, R2, dan R3). Tiap proses menggunakan hanya satu sumber daya, serta meminta sumber daya yang lainnya. Kondisi ini akan menyebabkan deadlock.

1# Gambarlah graf yang sesuai untuk skenario ini.

Tanyakan asisten, jangan melanjutkan langkah berikutnya sebelum gambar anda benar.

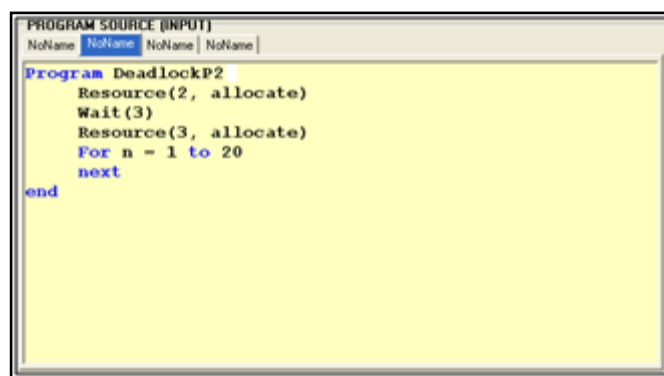
Jalankan CPU-OS Simulator.

2. Tuliskan kode berikut ini pada compiler.

```
Program DeadlocksPN
  Resource(X, allocate)
  Wait(3)
  Resource(Y, allocate)
  For n = 1 to 20
  next
end
```

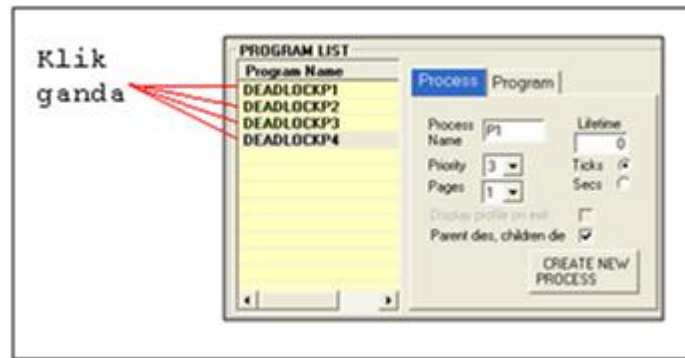
Kode diatas membuat suatu program yang berusaha mengalokasikan dua sumber daya untuk dirinya. Setelah pengalokasian yang pertama kemudian menunggu 3 detik dan mencoba mengalokasikan sumber daya yang lain. Diakhiri dengan menghitung 1 hingga 20, kemudian selesai

- a. Salin kode diatas dan tempelkan (*paste*) ke 3 jendela editor baru, jadi sekarang kita memiliki 4 program.
- b. Pada tiap program gantilah N dengan nama 1 hingga 4, misal: DeadlockP1, DeadlockP2, dst.
- c. Perhatikan gambar graf yang telah anda kerjakan pada langkah (1) diatas dan gunakan informasi tersebut untuk mengisi tiap X dan Y pada empat kode program tersebut.



Contoh Kode sumber

- d. Lakukan kompilasi pada tiap program.
- e. Muatkan keempat kode yang dihasilkan ke dalam memori.
- f. Beralihlah ke OS Simulator.
- g. Buatlah instansiasi untuk tiap program. Caranya dengan melakukan klik-ganda pada tiap nama program di PROGRAM LIST.

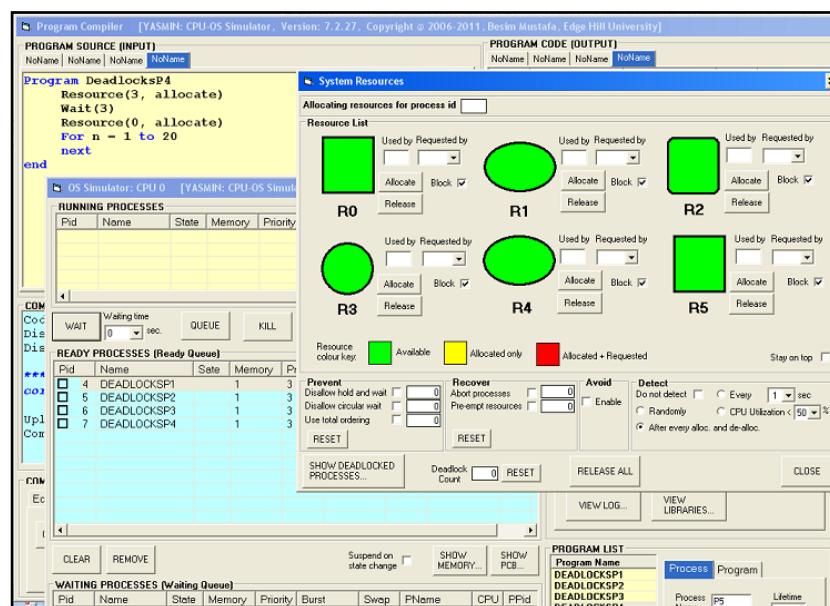


Klik ganda tiap program yang telah dimuatkan ke memori

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU
4	DEADLOCKP1		1	3	0	No	P1	
5	DEADLOCKP2		1	3	0	No	P2	
6	DEADLOCKP3		1	3	0	No	P3	
7	DEADLOCKP4		1	3	0	No	P4	

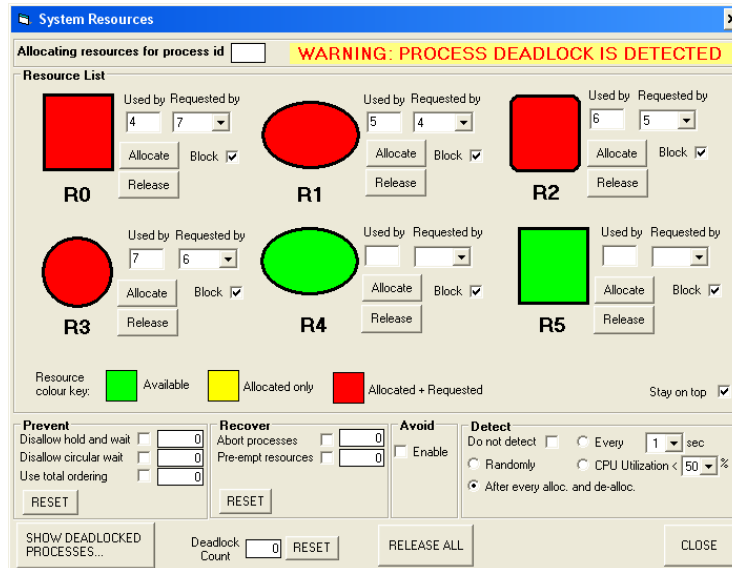
Proses dalam keadaan ready

- Pilih penjadwalan **Round Robin (RR)**.
- Maksimalkan kecepatan simulator. (**jangan klik START dulu**)
- Pilih tab **Views** dan klik tombol **VIEW RESOURCES**.
- Klik **Stay on top** pada jendela System Resource.



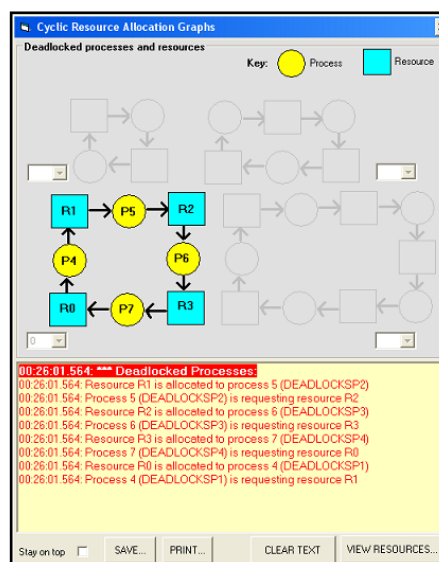
Kondisi awal

- l. Kembali ke tab OS Control, tekan **START**, dan amati simulasi pada jendela proses dan resources.
- m. Apakah anda mendapat kondisi deadlock seperti yang di rancang pada langkah (1) diatas? Jika belum, teliti kembali langkah-langkah diatas atau ulang dari awal jika diperlukan. **Jangan melanjutkan sebelum anda berhasil memperoleh peringatan Deadlock seperti gambar dibawah ini.**



Deadlock terdeteksi

- n. Jika telah memperoleh kondisi deadlock, kemudian klik **SHOW DEADLOCKED PROCESSES** pada jendela System Resources. Apakah gambar graf yang disajikan tampak seperti yang anda gambarkan pada langkah (1)?



Jendela pengamatan kejadian deadlock

3. Metode untuk **keluar** dari kondisi deadlock
 - a. Pada jendela System Resources, seharusnya tampak empat resources berwarna merah yang menandakan resources tersebut sedang digunakan (*allocated*) oleh satu proses dan dibutuhkan (*requested*) oleh proses lain.
 - b. Pilih satu diantara resources (merah) tersebut dan klik tombol **Release** didekatnya.
 - c. **Amati** yang terjadi pada proses-proses melalui jendela OS simulator.
 - d. Apakah situasi deadlock teratasi?
2# Jelaskan bagaimana langkah tersebut dapat mengatasi deadlock.
 - e. Buat ulang kondisi deadlock seperti sebelumnya (langkah 2).
 - f. Setelah kondisi deadlock diperoleh kembali, pada jendela OS simulator pilih satu proses pada waiting queue dalam frame WAITING PROCESS.
 - g. (Tekan tombol suspend) Lalu tekan tombol **REMOVE**, kemudian amati yang terjadi pada proses-proses.
 - h. Apakah situasi deadlock teratasi?
3# Jelaskan bagaimana langkah tersebut dapat mengatasi deadlock.
4. Metode untuk **mencegah** kondisi deadlock (sebelum terjadi).
 - a. Pada jendela System Resources beri tanda check pada **Disallow hold and wait**

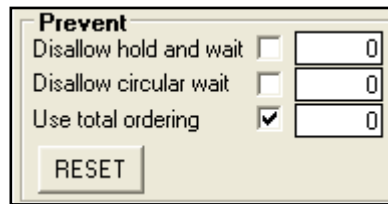
Prevent		
Disallow hold and wait	<input checked="" type="checkbox"/>	0
Disallow circular wait	<input type="checkbox"/>	0
Use total ordering	<input type="checkbox"/>	0
RESET		

- b. Buatlah kondisi deadlock seperti sebelumnya. Apakah berhasil? Apa yang terjadi? Klik tombol **SHOW DEADLOCKED PROCESSES** dan amati informasi yang ditampilkan pada jendela teks.
- c. Berikutnya singkirkan tanda check pada *Disallow hold and wait*, dan beri tanda check pada **Disallow circular wait**.

Prevent		
Disallow hold and wait	<input type="checkbox"/>	7
Disallow circular wait	<input checked="" type="checkbox"/>	0
Use total ordering	<input type="checkbox"/>	0
RESET		

- d. Ulangi langkah (4b).

5. Metode pencegahan deadlock yang ketiga menggunakan “**Total ordering**”.
- a. Pada jendela System Resources pilih Use Total Ordering, dan singkirkan tanda check pada pilihan yang lain.



Prevent	
Disallow hold and wait	<input type="checkbox"/> 0
Disallow circular wait	<input type="checkbox"/> 0
Use total ordering	<input checked="" type="checkbox"/> 0
RESET	

- b. Buatlah kondisi deadlock seperti sebelumnya. Apakah berhasil? Apa yang terjadi? Klik tombol **SHOW DEADLOCKED PROCESSES** dan amati informasi yang ditampilkan pada jendela teks.

4# Jelaskan prinsip pencegahan deadlock menggunakan tiap metode tersebut.