

Sistem Operasi - Kelas B

Kondisi untuk Mencapai Deadlock

1. Mutual Exclusion (mutual exclusion conditional)

- ↳ Jika suatu proses menggunakan suatu resource, tidak ada proses lain yg boleh menggunakan resource tersebut.

2. Kondisi genggam & tunggu (Hold and Wait)

- ↳ Pada saat suatu proses mengakses suatu resource, proses tersebut dapat meminta izin utk mengakses resource lain.

3. Kondisi non-preemption (no-preemption condition)

- ↳ Jika suatu proses meminta izin untuk mengakses resource, sementara resource tidak tersedia, maka permintaan tidak dapat dibatalkan.

4. Kondisi menunggu secara sirkuler (Circular wait condition)

- ↳ Jika proses P_i sedang mengakses resource R_i , dan meminta izin untuk mengakses resource R_j , & pada saat bersamaan proses P_j sedang mengakses R_j & minta izin utk mengakses resource R_i .

Penangan Deadlock

1. Mengabaikan Permasalahan (The Ostrich Algorithm)

- ↳ Dalam Algoritma ini dikatakan bahwa utk menghadapi deadlock ialah dgn berpura-pura bahwa tidak ada masalah apa pun. Artinya sistem operasi Windows & UNIX menanggulangi dgn cara tidak mendeteksi deadlock & membiarkannya secara otomatis restart program sekiranya tidak terjadi apapun.

2. Deteksi & pemulihan (Recovery).

- ↳ Hal2 yg terjadi dlm mendeteksi adanya deadlock, & deadlock dpt dipulihkan dgn hal2 berikut:
 - a. Permintaan sumber daya dikabulkan selama memungkinkan
 - b. Sistem operasi memeriksa adakah kondisi circular wait secara periodik.
 - c. Pemeriksaan adanya deadlock dpt dilakukan setiap ada sumber daya yg hendak digunakan oleh sebuah proses.

3. Pencegahan, dgn meniadakan salah satu dr empat kondisi deadlock

- ↳ Meniadakan Mutual Exclusion

Melakukan spooling perangkat2 yg harus diedikasikan ke suatu proses. Dengan

Sistem Operasi - Kelas B

spooling, permintaan di antrikan di harddisk.

4. Pengalokasian Sumber Daya yg efisien.

- ↳ jika semua sumber daya tersedia, proses dialokasikan yg diperlukan & berjalan sampai selesai. jika tidak tersedia harus menunggu sampai semua sumber daya yg diperlukan tersedia utk dialokasikan.

(non-preemptive condition)

↳ jika suatu proses meminta izin untuk mengakses resource, sementara resource tidak tersedia maka proses akan ditunda.

(holder wait condition)

↳ jika proses P sedang mengakses resource R, dan meminta izin untuk mengakses resource R, maka proses P akan menunggu sampai R selesai mengakses R.

Pengertian Deadlock

(The Circular Wait Algorithm)

↳ Dalam algoritma ini, kita akan memeriksa apakah ada proses yang memegang resource yang dibutuhkan oleh proses lain. Jika ada, maka proses tersebut akan menunggu sampai resource tersebut selesai digunakan.

(The Banker's Algorithm)

↳ Hal yang harus diperhatikan dalam algoritma ini adalah: apakah ada proses yang meminta resource yang sedang digunakan oleh proses lain?

a. Memeriksa apakah ada proses yang meminta resource yang sedang digunakan oleh proses lain?

b. Jika ada, maka proses tersebut akan menunggu sampai resource tersebut selesai digunakan.

c. Jika tidak ada, maka proses tersebut akan berjalan sampai selesai.

↳ Hal yang harus diperhatikan dalam algoritma ini adalah: apakah ada proses yang meminta resource yang sedang digunakan oleh proses lain?

3. Pengalokasian sumber daya secara efisien & adil.

(The Banker's Algorithm)

↳ Dalam algoritma ini, kita akan memeriksa apakah ada proses yang meminta resource yang sedang digunakan oleh proses lain.