Nama : Angela Lisanthoni
NPM : 2108301003 2
Kelas : Sistem Operasi A

No.: Date.:

a. Kondisi untuk mencapai Deadlock

1. Mutual exclusion

Jika suatu proses menggunakan suatu sumber daya, tidak ada proses lain yang boleh menggunakan sumber daya tersebut. Jadi satu proses di satu waktu Contoh: hanya ada satu proses pada satu waktu yang diperbolehkan mengirim perintah pada printer

2. Hold and wait

Pada saatu suatu proses mengakses suatu sumber daya, proses tersebut dapat meminta izin untk mengakses sumber daya lain

3. Kondisi non-preemption

Jika suatu proses meminta izin untk mengakses sumber daya sementara sumber daya tidak tersedia, maka permintaan tidak bisa dibatalkan.

4 Circular wait condition

Jika suatu proses Pi sedang mengkases resource Ri, dan meminta izin untuk mengakses resource Rj, dan pada saat bersamaan ada proses Pj sedang mengakses Rj dan minta izin untuk mengakses resource Ri

b. Penanganan Deadlock

1. Mengabaikan permasalahan

algoritma ostrich mengansumsikan bahwa permasalahan sangat jarang terjadi, sehingga permasalahan dapat diabaikan dan berpura-pura seakan tidak terjadi masalah. Terdapat 2 jenis yakni Trade offs (jika kondisi belum teridentifikasi, maka masalah yang sangat sarang terjadi dapat terjadi kembali) dan pendekatan hybrid (menentukan bahwa deadlock jarang atau tidak sama sekali terjadi)

No. : Date. :

2. Recovery
mengizinkan sistem yang mengalami deadlock, namun kemudian harus segera dapat diperbaiki. Tujuannya memeriksa apakah telah terjadi deadlock dan menentukan proses serta sumberdaya yang terlibat sehingga sistem dipulihkan agar sistem beroperasi kembali

3. Pencegahan dengan menjadakan salah satu dari empat kondisi deadlock
kondisi untuk mengatasi deadlock dengan cara menyakinkan bahwa paling sedikit satu dari kondisi tidak terjadi. Bisa dengan :
* mutual exclusion → buat resource shareable
* Hold and wait → melepas resource pada saat request
* No preemption → melepas resource pada saat waiting
* Circular wait → request berurutan dengan memberi nomor

4. Pengalokasian sumber daya yang efisien
sistem dapat mengalokasikan resource untuk tiap proses dalam urutan yang tepat tanpa terjadinya deadlock. Biasanya menggunakan algoritma graf alokasi. Algoritma ini bekerja dengan mendeteksi perputaran dalam sistem. Jika tidak ada perputaran dalam graf, sistem berada dalam status aman tetapi, jika perputaran ditemukan maka sistem dalam status tidak aman.