Dwinggrit Oktaviani
2108301010012

No.

Date . .

# SISTEM OPERASI / B

## KONDISI UNTUK MENCAPAI DEADLOCK

1. Mutual Exclusion (mutual exclusion conditional)
   ↳ Jika suatu proses menggunakan suatu resource, tidak ada proses lain yang boleh menggunakan resource tersebut.
2. Kondisi genggam dan tunggu (hold and wait)
   ↳ Jika terdapat proses mengakses suatu resource, proses tersebut dapat meminta ijin untuk mengakses resource lain.
3. Kondisi non-preemption (non-preemption condition)
   ↳ Jika suatu proses meminta ijin untuk mengakses resource, sementara resource tidak tersedia, maka permintaan tidak dapat dibatalkan
4. Kondisi menunggu secara sirkuler (circular wait condition)
   ↳ Jika proses Pi sedang mengakses resource Ri, dan meminta ijin untuk mengakses resource Rj, dan pada saat bersamaan proses Pj sedang mengakses Rj dan minta ijin untuk mengakses Resource Ri.

## PENANGANAN DEADLOCK

1. Mengabaikan permasalahan (The Ostrich Algorithm)
   ↳ digunakan dalam menangani deadlock pada pemrograman concurrent jika deadlock diyakini sangat jarang terjadi, dan jika biaya untuk mendeteksi/pencegahan lebih tinggi.
2. Deteksi dan Pemulihan (Recovery)
   ↳ Digunakan pada sistem yang mengijinkan terjadinya deadlock. Bertujuan untuk memeriksa apakah telah terjadi deadlock dan menentukan proses-proses dan sumberdaya** yang terlibat deadlock secara presisi (Deteksi)
   ↳ Digunakan untuk menghilangkan deadlock dari sistem, sehingga sistem beroprasi kembali, bebas dari deadlock dan proses yg terlibat deadlock dapat menyelesaikan eksekusi & membebaskan sumberdaya** nya (Pemulihan)
3. Pencegahan, dengan menjadakan salah satu dari 4 kondisi deadlock
   ↳ Pencegahan yang secara struktur bertentangan dengan empat kondisi terjadinya deadlock dengan deadlock prevention sistem untuk memastikan bahwa salah satu kondisi yang penting tidak dapat menunggu.

   ▷ Mencegah Mutual Exclusion
   → mutual exclusion benar** tidak dapat dihindari, karena tidak ada sumberdaya yang dapat digunakan bersama-sama

No.

Date . .

▷ Mencegah Hold and Wait
→ Sistem harus menjamin bila suatu proses meminta sumber daya, maka proses tersebut tidak sedang memegang sumberdaya yang lain.

▷ Mencegah Non Preemption
→ Peniadaan non preemption mencegah proses ** lain harus menunggu. Seluruh proses menjadi preemption, sehingga tidak ada tunggu menunggu.

▷ Mencegah Kondisi Menunggu Sirkular
→ Proses dapat meminta proses kapanpun menginginkan, tapi permintaan harus dibuat terurut secara numerik

4. Pengalokasian Sumber daya yang efisien
↳ Situasi ketika sumberdaya dialokasikan pada penggunaan nilai tertinggi. tidak ada alternatif untuk menggunakan sumber daya lebih lanjut tanpa membuat yang lain lebih buruk