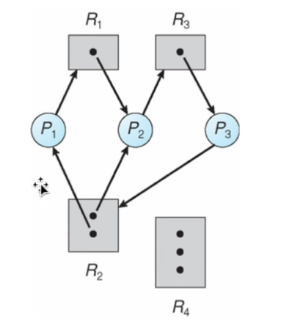
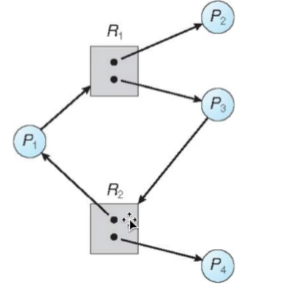


Hepsi ayni priority’ye sahipse burda bir deadlock durumu yoktur.



P3 bitemiyor R2’yi istiyor, P2 bitemiyor R3’u bekliyor ve P1 de bitemiyor o da R1’i bekliyor dolayisiyla burda bir deadlock durumu vardir.

Burda bir cycle vardir. Yani basladigimiz yere donup dolasip tekrar gelebiliyoruz. Iste bu durum graph teorisine gore kesinlikle bir deadlock vardir deriz. Cunku R1 de bir kaynak var R3 de bir kaynak var eger bu yerler tek degil de coklu kaynaga sahip olsaydi bu durumda kesin deadlock var diyemezdik. (Graph teorisini hatirla)



Burda bir cycle var fakat bu cylce’in sahip oldugu kaynaklar coklu oldugu icin burda kesinlikle bir deadlock vardir diyemeyiz. Oiabilir. Arastirdigimizda goruyoruz ki burda bir deadlock durumu soz konusu degildir. Bu arada eger graph ta bir cycle yoksa da kesinlikle deadlock yoktur denilir.

Deadlock prevention = Hic deadlock olmayacak bir sistem yapmaktir.

Deadlock avoidence = Deadlock olusacak bir sistemde olusabilecek deadlock’larin engellenmesi durumu.

**Deadlock onlenmesi**

**Mutual Exclusion** = Ayni anda bir resource’nin bir process tarafindan kullanilmasina izin vermek. Tabi bu durum shareable durumlar icin gecerli degildir cunku bir kaynagi 2 process paylasiyorsa bu durumda boyle bir sey mumkun olmayacak ama non-shareable sistemleri dusunursek processlerin birbirlerini dislamasi deadlock durumunu onleyecektir.

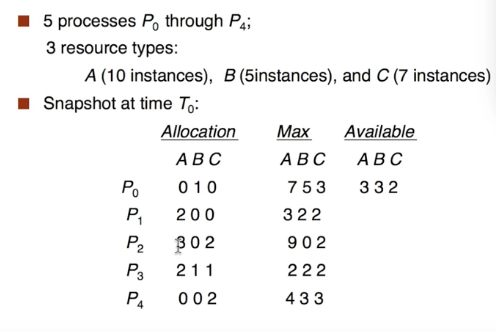
**Hold and Wait** = Process’in calismadan once isteyecegi kaynaklari bilmeyi on kosul olarak soyler. Baslangicta kaynaklarin ne oldugunu biliyorsunuz ve eger bir processin kullanacagi tum kaynaklar musaitse o processi isleme koyacak. Tabi iki process icin ayni anda kaynaklarin bos olmasi durumu olup da yine bir sorun cikmasin diye yontemimiz Atomic calismak zorundadir yani bir process icin kaynaklar kontrol edilirken baska hic bir process icin bu islem yapilmayacaktir.

**No Preemption** = Iste herhangi bir deadlock olma durumunda tum processlerin ellerindeki kaynaklari serbest birakmasini saglayan yontemdir.

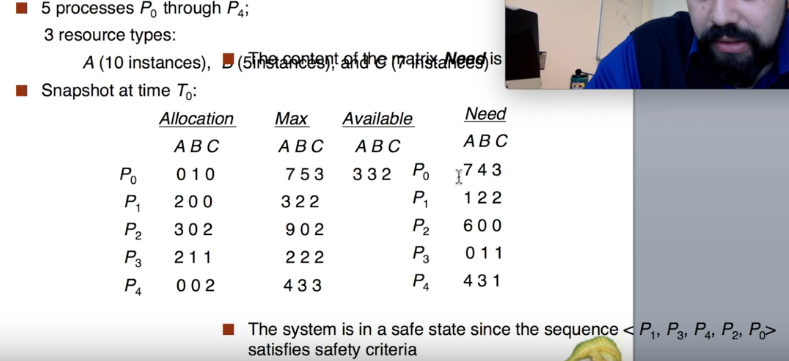
**Circular Wait** = Processlerin bir sirayla calismasi saglanip deadlock bu sekilde onlenir.

Transactionlar atomic olmak zorundadir. Mesela siz bir yerden para gonderiyorsunuz iste baska biri de size gonderiyor falan bunlarin hepsi tek process olarak yapilmali. (Bank Transactions)

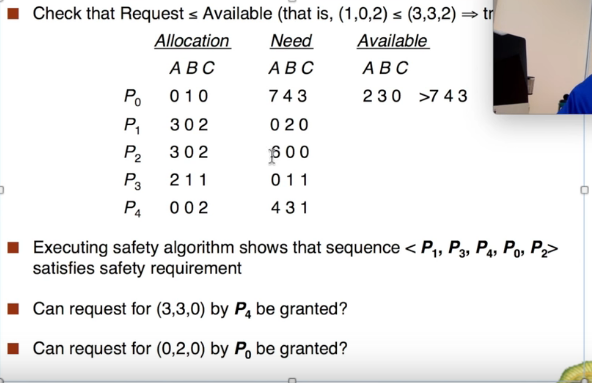
**Banker Algorithm**



Allocate edilen kaynaklar processler bittikten sonra serbest birakilir. Ve available kaynaklar guncellenir processler kaynak allocate edildikce de yine guncellenip dusurulur.



DIGER ORNEK

****

