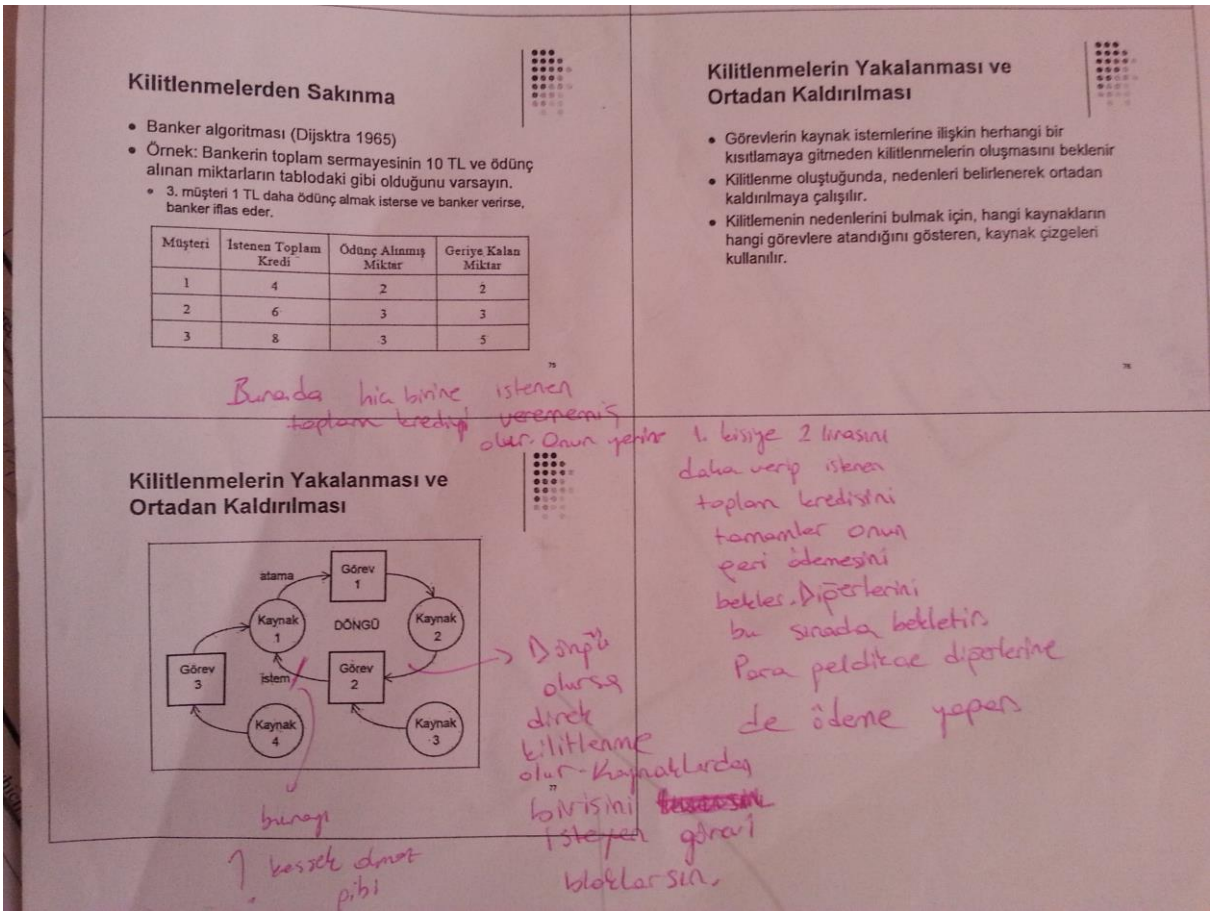
Finalde Çıkan Sorular



Kilitlenmelerden Sakınma

- Banker algoritması (Dijsktra 1965)
- Örnek: Bankerin toplam sermayesinin 10 TL ve ödünç alınan miktarların tablodaki gibi olduğunu varsayın.
  - 3. müşteri 1 TL daha ödünç almak isterse ve banker verirse, banker iflas eder.

Müşteri	İstenen Toplam Kredi	Ödünç Alınmış Miktar	Geriye Kalan Miktar
1	4	2	2
2	6	3	3
3	8	3	5

75

Burada hiç birine istenen toplam krediyi verememiş olur. Onun yerine 1. kişiye 2 liranın daha verip istenen toplam kredisini tamamlar onun geri ödemesini bekler. Diğerlerini bu sırada bekletir. Para peldikçe diğerlerine de ödeme yapar

Kilitlenmelerin Yakalanması ve Ortadan Kaldırılması

- Görevlerin kaynak istemlerine ilişkin herhangi bir kısıtlamaya gitmeden kilitlenmelerin oluşmasını beklenir
- Kilitlenme oluştuğunda, nedenleri belirlenerek ortadan kaldırılmaya çalışılır.
- Kilitlemenin nedenlerini bulmak için, hangi kaynakların hangi görevlere atandığını gösteren, kaynak çizgeleri kullanılır.

76

Kilitlenmelerin Yakalanması ve Ortadan Kaldırılması

77

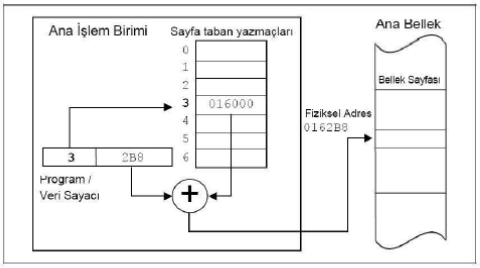
Döngü olursa direk kilitlenme olur. Kaynaklardan birisini isteyen görevi bloklarsın.

bunu ? kesek direkt gibi

Buradaki ilk slayta benzer bir soru vardı boşluk bırakmıştı hoca oradaki değerler ne olursa banker batmaz diye bir şey sormuştu. Bunun için en az birinin istediği toplam kredi karşılanacak kadar para elimizde kalmalı. Diğerlerine en çok ne kadar ödünç para daha verebilir dediğinde birisinin istediği toplam krediyi tamamlamak için en az ne kadar para gerekiyorsa o kadarını ayırmalıyız geri kalanı ödünç verebiliriz. Mesela yukarıdaki örnekte toplam 10 tl nin 8i ödünç verilmiş geri kalan 2 tl de ”geriye kalan miktar” ı en az olan müşteriye verilmeli burada artan para yok. Ama artsaydı artan miktar kadar diğerlerine borç verebilecektik.

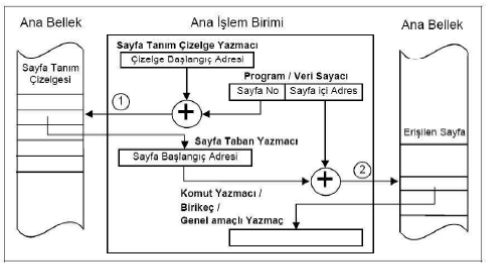
Kilitlenmelerle ilgili de testlerde soru vardı ya da doğru yanlış var mıydı emin olamadım orada gelmiş olabilir.

Sayfa Taban Yazmaçları ile Adresleme Mantığı



31

Sayfa Tanım Çizelgesi Kullanımı



32

Bu da ilk sorulardan biriydi. Program veri sayacı 32 bitten oluştuğuna göre ne kadarı sayfa no ne kadarı sayfa içi adres için ayrılmıştır diyordu. Ve bir tane de örnek verip fiziksel adresi nereye karşılık gelir diye soruyordu.

Ön Bellek Kullanımının Etkinliği

- $t_e = t_d + t_m$

  t_d: Sayfa tanım çizelgesine erişim süresi
  t_m: Fiziksel adrese erişim süresi

- $t_d = p\ t_c + (1-p)(t_c+t_m)$
  $= t_c + (1-p)\ t_m$

  t_c: Ön belleğe erişim süresi
  p: Erişilecek sayfanın başlangıç adresini bellekte olma olasılığı

- $t_e = t_d + t_m$
  $= t_c + (1-p)\ t_m + t_m$
  $= t_c + (2-p)\ t_m$

35

Adres Dönüştürme Düzeneklerinin Karşılaştırılması

Adres Dönüştürme Düzenek Türü	$(t_e-t_m)/t_m$
1. Sayfa Tanım Çizelgesinin AİB Taban Yazmaçlarında	% 0
2. Sayfa Tanım Çizelgesinin tümüyle Ön Bellekte tutulması	% 10
3. Sayfa Tanım Çizelgesinin bir kesiminin Ön Bellekte tutulması	% 25
4. Sayfa Tanım Çizelgesinin tümüyle Ana Bellekte tutulması	% 100

- Anabelleğe erişimin, önbelleğe erişimden 10 kat kadar yavaş olduğunu varsayılarak hesaplanmıştır.
- 3. durumda p = 0.85 olduğunu, yani 0,85 olasılıkla aradığımız sayfa başlangıç adresinin ön bellekte bulduğumuzu varsayıyoruz.

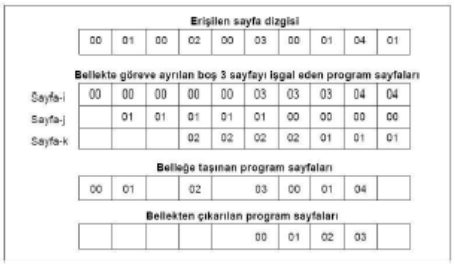
36

Bu da ilk sorulardan biriydi ana belleğe erişim süresi ön belleğe erişim süresinin 20 katıydı . % 95 oranında sanırım bellekte buluyorduk aradığımızı.

Yani çizelge

1. Taban yazmaçlarındaysa hep %0
2. Ön bellekte ise verilen orana göre değişiyor %5
3. Burada formülü kullanıyoruz.
4. Ana bellekteyse de %100

İlk Giren Sayfayı Çıkarma (FIFO) Algoritması için Örnek İşletim



67

En Erken Erişilmiş Sayfayı Çıkarma (LRU) Algoritması için Örnek İşletim

68

En Geç Erişilecek Sayfayı Çıkarma Algoritması için Örnek İşletim

69

Belady Anormalliği

- Görevlere ayrılan sayfa sayısının çok düşük tutulması, görevin çalışma sırasında ihtiyaç duyduğu sayfaları sık sık ana bellekte bulamamasına ve eksik sayfa uyarılarının artışına sebep olur.
  - Bu nedenle, görevlere ayrılan sayfa sayısının, bir alt sınırı olmalıdır.
- Bellekte bir göreve ayrılan sayfa sayısı ile görevin üreteceği eksik sayfa uyarısı sayısının ters orantılı olduğu, yani sayfa sayısı arttıkça eksik sayfa uyarısının azalacağı, düşünülebilir.

70

İki tane bu konudan soru vardı diye hatırlıyorum bunların nasıl işlediğine bakın.

Belady Anormalliği

- Bazen bir göreve ayrılan boş sayfa sayısı artırılsa bile eksik sayfa uyarısı artabilir. Buna Belady anormalliği denir.
  - Çünkü, görevlere ayrılan sayfa sayısı artırılırsa, sistemde görev sayısının çok olduğu durumlarda, bellekte boş sayfa bulmak zorlaşacaktır.
- Görevlere aynı anda atanabilecek sayfa sayısı, genelde eksik sayfa uyarısı sayısı taban alınarak işletim sisteminde belirlenir.

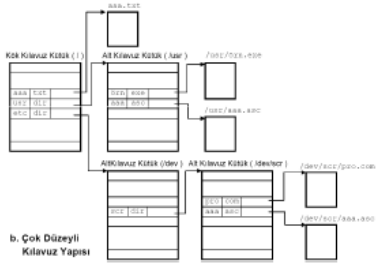
Bu da ya testte ya da doğru yanlışta vardı. Belady anormalliğinin tanımını bilin.

Kılavuz Kütük

- Bir kütük sistemindeki kütükler temel olarak iki çeşittir:
  - Kılavuz kütükler
  - Ve diğerleri (veri saklamak üzere kullanılan kütükler)
- Kılavuz kütükler, sistemde yer alan kütüklere erişimi sağlamak için şu tür bilgileri saklar:
  - Kütük adı
  - Kütük türü
  - Kütük yaratılma, erişim, ve son günlenme zamanları
  - Kütüğe erişim hakları
  - Fiziksel erişim bilgileri
- Kök kılavuz, :\ , / gibi damgalarla simgelenir ve işletim sistemi tarafından oluşturulur.

9

Çok Düzeyli Kılavuz Yapıları



b. Çok Düzeyli Kılavuz Yapısı

11

Kılavuz Kütük Satırı

- Bir kılavuz kütük satırında, ilgili kütük hakkında saklanan bilgiler:
  - Kütük adı
  - Kütük türü
  - Kütük yaratılma, erişim, ve son günlenme zamanları
  - Kütüğe erişim hakları
  - Fiziksel erişim bilgileri

Kütük Adı	Kütük Türü	Öznitelik Bilgisi	Yaratılma Tarihi	Günlenme Tarihi	Fiziksel Öbek Adresleri ya da İlk Öbek Göstergesi

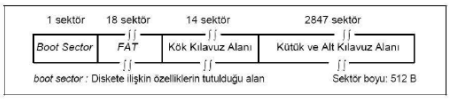
Disket Sürücü için FAT Boyu Hesaplanması

- 1.44 MB'lık bir disket, 512 byte uzunluğunda öbeklere sahipse, FAT tablosunun
- Satır sayısı: 1440 KB / 0.5KB = 2880
- Herbir satırın boyu: 12 bit, ($2^{11} < 2880 < 2^{12}$)
- Toplam boyu: 2880 * 12/8 = 4320 byte
- Diskte kapladığı öbek sayısı: 9 * 2 = 18 öbek
  - FAT iki kopya olarak saklanır.

33

MS-DOS'ta 1.44MB Disket Birimlerinin Düzenlenişi

- Kök kılavuzun yer aldığı disk öbeklerine nasıl erişileceğini gösteren bir yapı bulunmadığından, kök kılavuzların sürücü üzerindeki konumları değişmezdir.
  - *MS-DOS*, 1.44 MB'lık disketlerde, kök kılavuzu 19 uncu sektörden başlayarak, en çok 14 sektörlük yer kaplayacak biçimde yaratır.

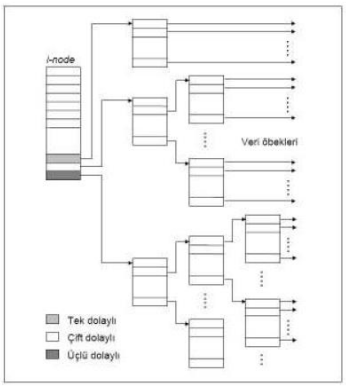


34

Bu tarz şeyler de test veya doğru yanlışta çıktı kılavuz kütüğün özelliklerini falan bilin. Neredeki pointer nereyi gösteriyor falan.

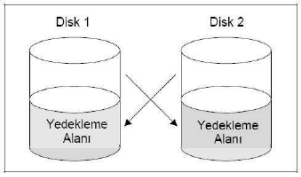
UNIX'te *i-node* ve Kılavuz Kütük Görünümü

UNIX'te Fiziksel Disk Öbeklerine Erişim



i-node larla ilgili de bir kaç soru vardı.

Diskte Yedekleme

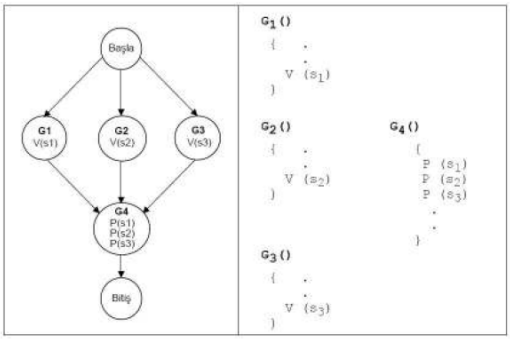


- Striping (RAID0)
- Mirroring (Aynalama) (RAID1)
  - Yazma/Okuma hızları, Hataya duyarlılık nasıl olur?

57

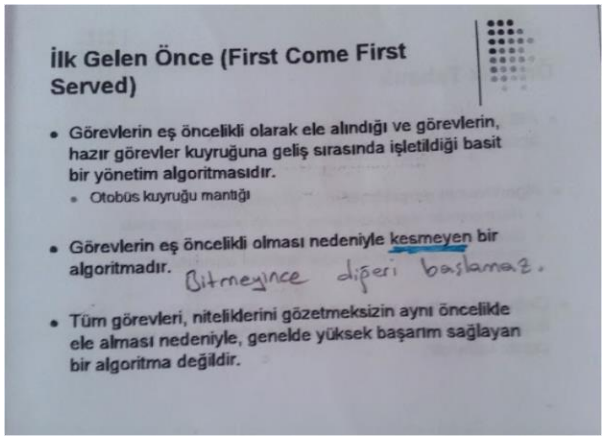
Bir soruda raid0 ve raid1 i kullanarak okumayı ve yazmayı bilmem kaç kat arttıracak yedekleme mimarisi tasarlayın diyordu.

Öncelik Çizgesi ve Birden Fazla Görev Arası Zamanuyumlama



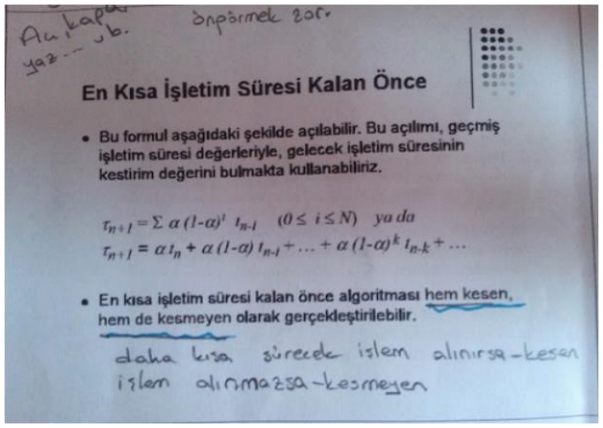
40

Böyle bir soru vardı hoca görevleri yazmıştı bizden içlerini doldurmamızı istiyordu. Bunlarda da bir görevden ne kadar ok çıkıyorsa o kadar V() yazmanız lazım ne kadar ok giriyorsa da o kadar P() yazmalısınız mesela G1 den V(s1) demiş G4 de ona karşılık gelen P() yi yazmanız gerekiyor yani P(s1) bu s1 s2 falan sizin verdiğiniz şeyler önemli olan bağlantıyı doğru yapmanız yani G1 in içine V(s4) de yazabilirdik bu durumda G4 ün içinde de P(s1) yerine P(s4) yazmamız gerek.



İlk Gelen Önce (First Come First Served)

- Görevlerin eş öncelikli olarak ele alındığı ve görevlerin, hazır görevler kuyruğuna geliş sırasında işletildiği basit bir yönetim algoritmasıdır.
  - Otobüs kuyruğu mantığı
- Görevlerin eş öncelikli olması nedeniyle kesmeyen bir algoritmadır. Bitmeyince diğeri başlamaz.
- Tüm görevleri, niteliklerini gözetmeksizin aynı öncelikle ele alması nedeniyle, genelde yüksek başarım sağlayan bir algoritma değildir.



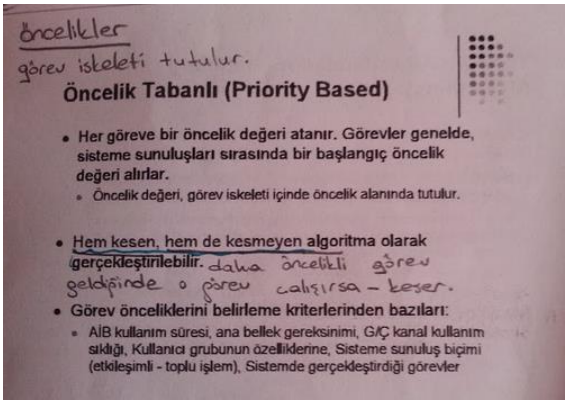
En Kısa İşletim Süresi Kalan Önce

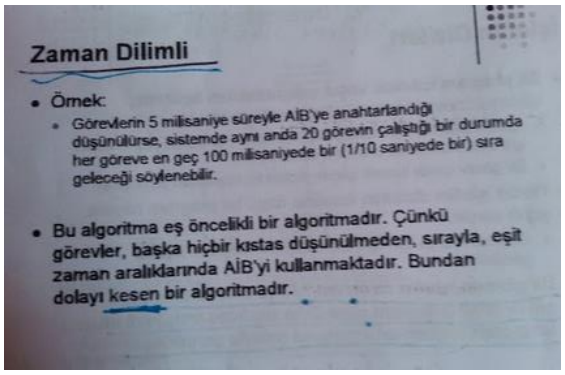
- Bu formul aşağıdaki şekilde açılabilir. Bu açılımı, geçmiş işletim süresi değerleriyle, gelecek işletim süresinin kestirim değerini bulmakta kullanabiliriz.

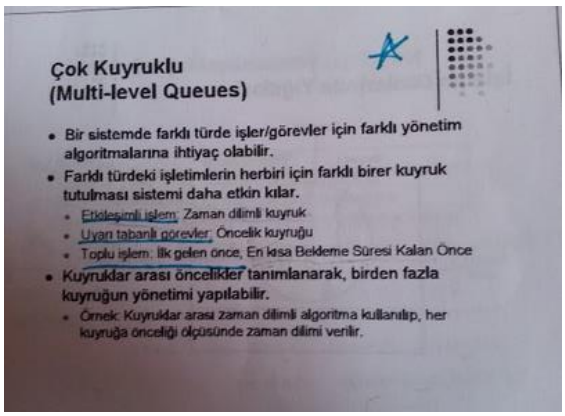
$$\tau_{n+1} = \Sigma\, \alpha(1-\alpha)^i\, t_{n-i} \quad (0 \le i \le N) \quad \text{ya da}$$
$$\tau_{n+1} = \alpha t_n + \alpha(1-\alpha) t_{n-1} + \ldots + \alpha(1-\alpha)^k t_{n-k} + \ldots$$

- En kısa işletim süresi kalan önce algoritması hem kesen, hem de kesmeyen olarak gerçekleştirilebilir.

daha kısa sürecek işlem alınırsa-kesen
işlem alınmazsa-kesmeyen







Bu kesen ve kesmeyen algoritmalarla ilgili de soru vardı birkaç tane hangisinin kesen hangisinin kesmeyen olduğunu bilin.

Kesilme düzeneklerine falan da bakın karşılıklı dışlama tıkanma falan bunlarla ilgili de soru vardı.

Test ve yorum sorularının çoğu 5. Ve 6. ünitedendi en çok onlara bakın derim.

Onun dışında vizelerde çıkan konulara da bir bakın bence send-receive, threadlerin özellikleri, fork, monitör falan.