

Finalde Çıkan Sorular

Kilitlenmelerden Sakınma

- Banker algoritması (Dijkstra 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

Burada hiç birine istenen toplam krediyi verememiş olur. Onun yerini 1. kişiye 2 lirasını daha verip istenen toplam kredisini tamamlar onun geri ödemesini bekler. Diğerlerini bu sırada bekletin. Para peşinlikle 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.
- Kilitlenmenin nedenlerini bulmak için, hangi kaynakların hangi görevlere atandığını gösteren, kaynak çizgeleri kullanılır.

Kilitlenmelerin Yakalanması ve Ortadan Kaldırılması

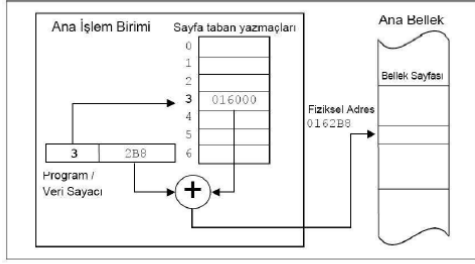
Burada bir döngü var. Görev 1 ve Görev 2 birbirini bekliyor. Kaynak 1 Görev 1'e atandı, Kaynak 2 Görev 1'e ve Görev 2'ye atandı. Kaynak 2 Görev 2'ye atandı, Kaynak 3 Görev 2'ye atandı. Kaynak 1 Görev 3'e atandı, Kaynak 4 Görev 3'e atandı. Kaynak 1 Görev 3'e atandı, Kaynak 4 Görev 3'e atandı. Kaynak 1 Görev 3'e atandı, Kaynak 4 Görev 3'e atandı.

Döngü oluşsa direkt kilitlenme olur. Kaynaklarda birisini istemek görevi bloklar sen.

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.

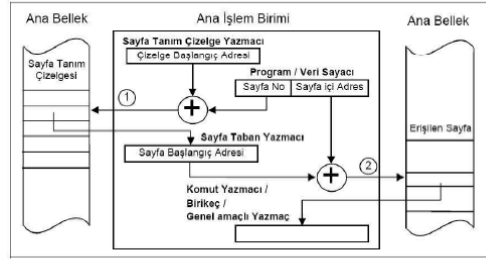
Kilitlenmelerle ilgili de testlerde soru vardı ya da doğru yanlış var mıydı emin olamadım orada gelmiş olabilir.

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_c : Sayfa tanım çizelgesine erişim süresi
 t_m : Fiziksel adrese erişim süresi
- $t_d = p \cdot t_c + (1-p) \cdot (t_c + t_m)$
 $= t_c + (1-p) \cdot t_m$
 t_c : Ön belleğe erişim süresi
 p : Erişilecek sayfanın başlangıç adresini bellekte olma olasılığı
- $t_e = t_d + t_m$
 $= t_c + (1-p) \cdot t_m + t_m$
 $= t_c + (2-p) \cdot t_m$

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, ön belleğe erişimden 10 kat kadar yavaş olduğunu varsayarak hesaplanmıştır.
- 3. durumda $p = 0.85$ olduğunu, yani 0,85 olasılıkla aradığımız sayfa başlangıç adresinin ön bellekte bulunduğumuzu 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

Erişilen sayfa dizisi										
00	01	00	02	00	03	00	01	04	01	
Bellekte görevi ayrılan boş 3 sayfayı işgal eden program sayfaları										
Sayfa-i	00	00	00	00	00	03	03	03	04	04
Sayfa-j		01	01	01	01	01	00	00	00	00
Sayfa-k				02	02	02	02	01	01	01
Belleğe taşınan program sayfaları										
	00	01		02		03	00	01	04	
Bellekten çıkarılan program sayfaları										
						00	01	02	03	

67

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

Erişilen sayfa dizisi										
00	01	00	02	00	03	00	01	04	01	
Bellekte görevi ayrılan boş 3 sayfayı işgal eden program sayfaları										
Sayfa-i	00	00	00	00	00	00	00	00	00	00
Sayfa-j		01	01	01	01	03	03	03	04	04
Sayfa-k				02	02	02	02	01	01	01
Belleğe taşınan program sayfaları										
	00	01		02		03		01	04	
Bellekten çıkarılan program sayfaları										
						01		02	03	

68

En Geç Erişilecek Sayfayı Çıkarma Algoritması için Örnek İşletim

Erişilen sayfa dizisi										
00	01	00	02	00	03	00	01	04	01	
Bellekte görevi ayrılan boş 3 sayfayı işgal eden program sayfaları										
Sayfa-i	00	00	00	00	00	00	00	00	00	00
Sayfa-j		01	01	01	01	01	01	01	01	01
Sayfa-k				02	02	03	03	03	04	04
Belleğe taşınan program sayfaları										
	00	01		02		03			04	
Bellekten çıkarılan program sayfaları										
						02			03	

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 artmasına sebep olur.
 - Bu nedenle, görevlere ayrılan sayfa sayısının, bir alt sınırı olmalıdır.
- Bellekte bir görevi 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 görevi ayrılan boş sayfa sayısı artırılabilir. 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 sistemince belirlenir.

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

Kılavuz Kütük

- Bir kütük sistemindeki kütükler temel olarak iki çeşittir:
 - Kılavuz kütükler
 - Ve diğerleri (veriyi 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 günclenme zamanları
 - Kütüğe erişim hakları
 - Fiziksel erişim bilgileri
- **Kök kılavuz**, \, / gibi damgalarla simgelerin ve **işletim sistemi tarafından** oluşturulur.

Çok Düzeyli Kılavuz Yapıları



Kılavuz Kütük Satırı

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

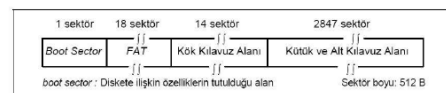
Kütük Adı	Kütük Türü	Öznitelik Bilgisi	Yaratılma Tarihi	Günlendirme Tarihi	Fiziksel Öbek Adresleri ya da İlk Öbek Göstergesi
-----------	------------	-------------------	------------------	--------------------	---

Disket Sürücü için FAT Boyu Hesaplanması

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

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 yarıtır.

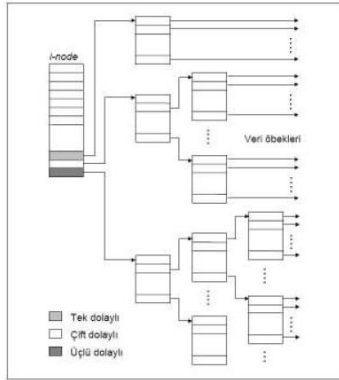


Bu tarz şeyler de test veya doğru yanılıştta çıktı kılavuz kütüğün özelliklerini falan bilin. Neredeki pointer nereyi gösteriyor falan.

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

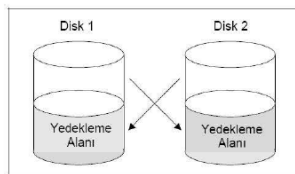


UNIX'te Fiziksel Disk Öbeklerine Erişim



i-node larla ilgili de bir kaç soru vardı.

Diskte Yedekleme

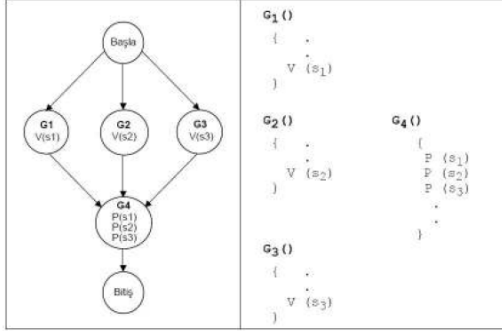


- Striping (RAID0)
- Mirroring (Aynalama) (RAID1)
 - Yazma/Okuma hızları, Hataya duyarlılık nasıl olur?

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



$G_1()$
{
·
V {s₁}
}

$G_2()$
{
·
V {s₂}
}

$G_3()$
{
·
V {s₃}
}

$G_4()$
{
·
P {s₁}
P {s₂}
P {s₃}
·
}

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 kescmeyen bir algoritmadır. *Bitmeyince diğeri başlanmaz.*
- Tüm görevleri, niteliklerini gözlemeksizin aynı öncelikte ele alması nedeniyle, genelde yüksek başarımlı sağlayan bir algoritma değildir.

En Kısa İşletim Süresi Kalan Önce

Açık kapı yaz -- vb. örnekleme 2017

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

$$t_{n+1} = \sum \alpha (1-\alpha)^i t_{n-i} \quad (0 \leq i \leq N) \text{ ya da}$$

$$t_{n+1} = \alpha t_n + \alpha (1-\alpha) t_{n-1} + \dots + \alpha (1-\alpha)^k t_{n-k} + \dots$$

- En kısa işletim süresi kalan önce algoritması hem kesen, hem de kescmeyen olarak gerçekleştirilebilir. *daha kısa sürecekle işlem alınırsa -kesen işlem alınmazsa -kescmeyen*

öncelikler
görev iskeleti tutulur.

Öncelik Tabanlı (Priority Based)

- Her göreve bir öncelik değeri atanır. Görevler genelde, sisteme sunulmaları sırasında bir başlangıç öncelik değeri alırlar.
- Öncelik değeri, görev iskeleti içinde öncelik alanında tutulur.
- Hem kesen, hem de kesmeyen algoritma olarak gerçekleştirilebilir. *daha öncelikli görev geldiğinde o görev çalışırsa - keser.*
- Görev önceliklerini belirleme kriterlerinden bazıları:
 - AİB kullanım süresi, ana bellek gereksinimi, GİÇ kanal kullanım sıklığı, Kullanıcı grubunun özelliklerine, Sisteme sunulmuş biçimi (etkileşimli - toplu işlem), Sistemde gerçekleştirdiği görevler

Zaman Dilimli

- Örnek:
 - Görevlerin 5 milisaniye süreyle AİB'ye anahtarıldığı düşünülürse, sistemde aynı anda 20 görevin çalıştığı bir durumda her göreve en geç 100 milisaniyede bir (1/10 saniyede bir) sıra geleceği söylenebilir.
- Bu algoritma eş öncelikli bir algoritmadır. Çünkü görevler, başka hiçbir kıstas düşünülmeden, sırayla, eşit zaman aralıklarında AİB'yi kullanmaktadır. Bundan dolayı kesen bir algoritmadır.

Çok Kuyruklu (Multi-level Queues)

- Bir sistemde farklı türde işler/görevler için farklı yönetim algoritmalarına ihtiyaç olabilir.
- Farklı türdeki işlemlerin herbiri için farklı birer kuyruk tutulması sistemi daha etkin kılar.
 - Etkileşimli işlem: Zaman dilimli kuyruk
 - Uyarı tabanlı görevler: Öncelik kuyruğu
 - Toplu işlem: İlk gelen önce, En kısa Bekleme Süresi Kalan Önce
- Kuyruklar arası öncelikler tanımlanarak, birden fazla kuyruğun yönetimi yapılabilir.
 - Örnek: Kuyruklar arası zaman dilimli algoritma kullanılıp, her kuyruğa önceliği ölçüsünde zaman dilimi verilir.

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. ünitelendi en çok onlara bakın derim.

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