07.12.2010

İ.K.Ü. FEN-EDEBİYAT FAKÜLTESİ
MATEMATİK-BİLGİSAYAR BÖLÜMÜ

2010-2011 GÜZ YARIYILI "MC 78K İŞLETİM SİSTEMLERİ"
I.VİZE ÇÖZÜMLERİ

1) Aşağıdaki kavramları kısaca açıklayınız.

- İzlek, iş parçacığı (*Thread*)
- Karşılıklı Dışlama (*Mutual Exclusion*)
- Semafor (*Semaphore*)
- İş Sıralama (*Job/CPU Scheduling*)

İzlek (*Thread*) : Bir prosesin sözde-paralel olarak çalışabilen iş parçacıklarından her biri. Bir prosesin bu şekilde "*mini proseslere*" parçalanmasının temel avantajı, I/O bağımlı bir iş parçacığının I/O işlem isteğinde bloke olup merkezi işlem birimini (CPU) aynı prosese ait başka bir izleğe bırakabilmesi ve bu sebeple sistem performansında oluşacak artıştır. Aynı prosesin izlekleri prosese ait olan ortak adres alanını paylaşır; bu şekilde aynı kaynaklara erişebilirler ve kolay haberleşebilirler.

Karşılıklı Dışlama (*Mutual Exclusion*) : Paylaşılan (ortak erişilen) bir kaynağı kullanan (veya kendi kritik bölgesinde olan) bir proses, diğer proseslerin o kaynağa erişimini (veya kritik bölgelerine girmesini) engellemeli; diğer bir ifadeyle yarış durumunun oluşmasını önlemek için diğer prosesleri "dışlamalıdır".

Semafor (*Semaphore*) : Kritik bölge kontrolü ve proses senkronizasyonu için kullanılan ve iki adet kesilemeyen işleme (*up (V), down (P)*) sahip olan tamsayı bir değişkendir. **P** işlemi ile semaforun değeri sıfırdan büyükse bu değer bir azaltılır ve işleme devam edilir; semaforun değeri 0 ise proses bu semafor üzerinde bloke olur (*sleep*). **V** işlemi ile semaforun değeri 0'dan farklı ise (veya 0 olup da bu semafor üstünde bloke olmuş hiç proses yoksa) bu değer bir arttırılır; semaforun değeri 0 ise ve üzerinde bloke olmuş en azından bir proses varsa, bu proseslerden biri uyandırılır (*wakeup*), semaforun değeri arttırılmaz. Bu iki işlem atomiktir; kesilemez.

İş Sıralama (*Job/CPU Scheduling*) : Çalışmaya hazır durumda olup hazır kuyruğunda bekleyen proseslerin belirli bir algoritmaya göre seçilip merkezi işlem birimine sırayla atanmaları. İşletim sisteminin iş sıralama yapan modülüne iş planlayıcısı veya iş çizelgeleyicisi (*scheduler*) denir.

2) Yarış durumu (*race condition*) nedir, hangi durumlarda oluşur ? Aşağıda sembolik kod ile verilen iki prosesin (A ve B) çalışmasında hangi senaryo gerçekleşirse ortaya bir yarış durumu çıkar ? Örnek veriniz.

```
global int a = 0
```

```
// PROSES A : Seri port denetçisinden
// kesme geldiğinde devreye girer
gorev A()
    a = a + 1
    print "RX"
gorev sonu
```

```
// PROSES B : Her saniyede bir
// devreye girer
gorev B()
    if (a mod 2 = 0)
        print a
    end if
gorev sonu
```

Yarış Durumu (*Race Condition*) : Ortak bir bellek alanına (veya ortak bir değişkene) okuma ve/veya yazma amaçlı erişen iki veya daha fazla sayıdaki prosesi göz önüne alalım. Paylaşılan ortak değişkenin son değeri, bu proseslerin merkezi işlem birimindeki çalışma sıralarının farklılıklarına göre değişiklik gösteriyorsa bu istenmeyen duruma *yarış durumu* denir.

Verilen örnekteki **a** değişkeni paylaşılan değişken olup, A prosesi bu değişkenin değerini 1 arttırmaktadır. B prosesi ise **a** değişkeninin değeri bir çift sayı olduğunda bu değeri ekrana çıkarmaktadır; dolayısıyla B prosesinin tek sayıları (ör:1,3,5...) ekrana çıkarma durumu söz konusu olmamalıdır.

Şimdi aşağıdaki senaryonun oluştuğunu varsayalım :

- İlk olarak **B** prosesi devreye girsin ve **a** değişkeninin değerini okusun (**a**=0). (0 mod 2)=0 olduğu için bu değeri ekrana yazmak üzereyken seri port denetçisinden gelen bir kesme sebebiyle *print* komutunu çalıştıramadan önce kesilsin.
- **A** prosesi devreye girer, ve **a=1** yapıp ekrana "RX" mesajını çıkarır.
- **B** prosesi tekrar devreye girdiğinde kesildiği noktadan işlemine devam eder. **a** değişkeninin güncel değerini okur ve "*print a*" komutunu işleterek ekrana **1** değerini çıkarır (!!).

Problemin kaynağı paylaşılan **a** değişkenine erişim esnasında proseslerin kesilebilmesidir.
(*Örnek wikipedia sitesinden alınmıştır*).

3) a) İçerik Anahtarlama (*context switching*) ne demektir, içerik anahtarlama sırasında işletim sistemi tarafından yapılan temel işlemler nelerdir ?

b) Round-robin iş sıralama algoritmasındaki **quantum** süresinin çok uzun veya çok kısa olmasını, içerik anahtarlama ve genel sistem performansı açısından karşılaştırınız.

a) İçerik Anahtarlama (*Context Switching*) : İşletim sisteminin merkezi işlem biriminde çalışmakta olan bir prosesin yerine başka bir prosesi alması sürecinde yapılan işlemlere denir. Çalışmakta olan proses normal koşullarla sonlanabilir veya çeşitli nedenlerden dolayı (örneğin; prosesin bir I/O isteğinde bulunması, daha yüksek öncelikli bir prosesin hazır kuyruğuna varması, prosese ayrılan sürenin dolması gibi nedenlerle) işletim sistemi tarafından kesilebilir. Kesilen prosesin kesilme anındaki durumu (CPU yazmaçları, program sayacı, ...) prosese ilişkin kontrol bloğuna yazılır (böylelikle bu proses tekrar devreye girdiğinde bu bilgiler geri yüklenerek prosesin kaldığı yerden devam etmesi sağlanır). Merkezi işlem biriminde çalıştırılmaya başlanacak olan yeni prosesin kontrol bloğundaki bilgiler ise sisteme yüklenir ve içerik anahtarlama işlemi tamamlanmış olur. Bu işlemler ek yüktür (*overhead*); zira işletim sistemi bu süre zarfında yararlı bir iş yapmaz (kullanıcı prosesleri çalışmaz).

b) Round-robin iş sıralama algoritmasındaki **quantum** süresi çok kısa olduğunda prosesler daha sık aralıklarla kesilir, dolayısıyla çok fazla sayıda içerik anahtarlama yapılır. Böyle bir durumda genelde ihmal edilen içerik anahtarlama sürelerinin toplamı ihmal edilemeyecek bir orana yükselebilir. Bu ise kullanıcı prosesleri açısından sistem performansında bir düşüş yaşanmasına sebep olur (CPU işlem zamanının belirli bir yüzdesi sistem ek yükü olarak harcanır). Diğer yandan **quantum** süresi çok uzun tutulduğunda proseslerin bir quanta sürede sonlanma olasılığı artar, içerik anahtarlama sayısı azalır, böylelikle sistemin ek yükü ihmal edilir, sistem performansında artış sağlanır. Fakat quantum süresinin çok uzun olması round-robin algoritmasının FIFO algoritması gibi çalışması anlamına gelir; bu algoritmanın zayıf tarafı ise ortalama proses bekleme süresinin yüksek çıkabilmesidir. Bu sebeplerden dolayı round-robin algoritmasını kullanan işletim sistemlerinde **quantum** makul bir değerde seçilir.

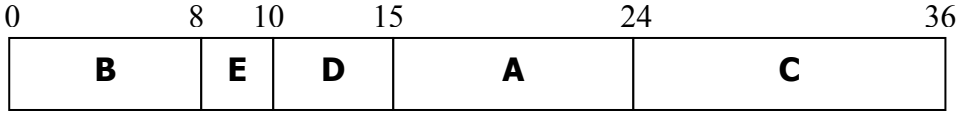
4) Bir işletim sisteminin hazır kuyruğunda aşağıdaki sırada gelen prosesler bulunmaktadır. Buna göre "**en kısa süreli iş ilk"** (*shortest job first*) ve quantum süresi **4 msn** olan "**çevrimsel sıralama"** (*round robin scheduling*) algoritmalarına göre oluşacak zaman diyagramlarını çizip her iki yöntem için proseslerin ortalama bekleme sürelerini (*average waiting time*) hesaplayınız.

Proses	**Kuyruğa Varış Zamanı**	**İşlem Süresi**
A	0	9
B	0	8
C	2	12
D	5	5
E	7	2

En Kısa Süreli İş İlk (SJF – Shortest Job First)

Bu algoritma *non-preemptive* yapıdadır; dolayısıyla kuyruğa ulaşan proseslerin çalışan prosesi kesme durumu söz konusu değildir.

Zaman diyagramı :



Bekleme Süreleri

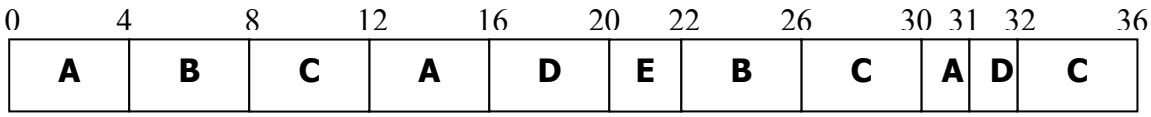
A → 15 – 0 = 15 mSn
B → 0 – 0 = 0 mSn
C → 24 – 2 = 22 mSn
D → 10 – 5 = 5 mSn
E → 8 – 7 = 1 mSn

Ortalama Proses Bekleme Süresi :
(15+0+22+5+1) / 5 = 43/5 = **8.6** mSn

Çevrimsel Sıralama (RR – Round Robin)

quantum = 4 mSn

Zaman diyagramı :



Bekleme Süreleri

A → (12-4)+(30-16) = 22 mSn
B → (4-0) + (22-8) = 18 mSn
C → (8-2)+(26-12)+(32-30) = 22 mSn
D → (16–5)+(31-20) = 22 mSn
E → (20–7) = 13 mSn

Ortalama Proses Bekleme Süresi :
(22+18+22+22+13) / 5 = 97/5 = **19.4** mSn

5) Bir sistemdeki prosesler, sistemde bulundukları süre boyunca CPU ve Giriş/Çıkış çevrimleri yaparlar. Aşağıda 4 proses için bu çevrimlerin süresi verilmiştir. Buna göre "**kalan iş süresi en az olan ilk**" (*Shortest-Job-Remaining First*) algoritmasına göre oluşan zaman diyagramını oluşturunuz ve proseslerin sistemde ortalama dönme sürelerini (*average turnaround time*) hesaplayınız. (NOT: Adı geçen algoritmada işletim sisteminin çalışma modu preemptive'dir).

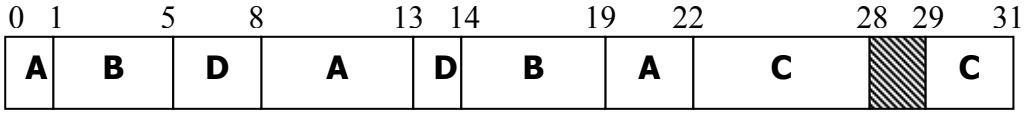
Proses	**Sisteme Giriş Zamanı**	**CPU ve I/O İşlem Süreleri**
A	0	6C-4I-3C
B	1	4C-2I-5C
C	1	6C-1I-2C
D	3	3C-5I-1C

Çözüm

Başlangıçta *(0. mSn*'de) kuyrukta sadece A prosesi olduğundan bu proses çalışmaya başlar. *1. mSn*'de B ve C prosesleri hazır kuyruğuna varmaktadır; iş sıralama algoritması *preemptive* olduğundan ve B prosesinin CPU işlem zamanı (*4mSn*) çalışmakta olan A prosesinin kalan işlem zamanından (6-1=*5mSn*) daha kısa olduğundan işletim sistemi içerik anahtarlama yapar; yani *1. mSn*'de B prosesi devreye girer. *3. mSn*'de gelen D prosesi B'yi kesemeyecektir (zira *3. mSn*'de B prosesinin 3-1=*2mSn*'lik işlem zamanı kalmıştır, oysa D prosesi *3 mSn*'lik işlem zamanına ihtiyaç duymaktadır). B prosesi *5. mSn*'de merkezi işlem birimini terk eder, *2 mSn*'lik bir I/O işleminin ardından (yani 5+2=*7.mSn*'de) hazır kuyruğuna geri gelecektir. *5. mSn*'de kuyrukta A, C ve D prosesleri bulunmaktadır. Bu üç proses arasında en kısa işlem zamanı olan D prosesi çalışmaya başlar, *8. mSn*'de işlemini bitirir (*7. mSn*'de kuyruğa tekrar ulaşan B prosesi D'yi kesemez). D prosesi 8+5=*13. mSn*'de kuyruğa tekrar varacaktır. *8. mSn*'de kuyrukta A, B ve C prosesleri vardır. En kısa işlem süresine sahip iki proses vardır : A ve B (*5 mSn*). Eşitlik durumunda eşitlik çözücü olarak FIFO algoritması kullanıldığından, daha uzun süredir kuyrukta olan A prosesi çalışır. *13. mSn*'de A prosesinin işlemi biter; bu proses 13+4=*17. mSn*'de hazır kuyruğuna tekrar ulaşacaktır. *13. mSn*'de kuyrukta B, C ve D prosesleri vardır; en kısa işlem zamanına sahip D prosesi çalışır. *14. mSn*'de D prosesi sonlanır; B prosesi devreye girer. B prosesinin sonlanması *19. mSn*'yi bulacaktır; bu sırada *17. mSn*'de kuyruğa varan A prosesi B'yi kesemez (3 > 2). *19. mSn*'de kuyrukta A ve C prosesleri kalmıştır. A prosesinin işlem süresi daha kısa olduğundan (3 < 6) bu proses çalışır ve *22. mSn*'de sonlanır. Sistemde kalan tek proses C prosesidir ve henüz hiç çalışmamıştır. Bunun nedeni işlem süresinin o ana kadar yarıştığı diğer proseslere göre hep yüksek kalmasıdır. *22. mSn*'de C prosesi çalışmaya başlar, *28. mSn*'de merkezi işlem birimini terk eder; *1 mSn*'lik I/O işlemi için bloke olur. Bu sırada hazır kuyruğunda hiçbir proses olmadığından CPU boşta kalır. *29. mSn*'de kuyruğa tekrar ulaşan C prosesi son CPU işlemini de tamamlayıp *31. mSn*'de sonlanır.

Sistemin zaman diyagramı aşağıda verilmiştir.

Zaman Diyagramı :



Proseslerin Dönme Süreleri

A → (22-0) = 22 mSn
B → (19-1) = 18 mSn
C → (31-1) = 30 mSn
D → (14-3) = 11 mSn

Ortalama Proses Dönme Süresi :
(22+18+30+11) / 4 = 81/4 = **20.25** mSn