İşletim Sistemlerine Giriş

Zamanlama (Scheduling)

Zamanlama (Scheduling)

Eğer bir bilgisayar çok programlı(multi programming) ise, sıklıkla birçok süreç aynı anda işlemciyi kullanmak için birbirleri ile yarışırlar.

Bu durum aynı anda birden fazla sürecin hazır(ready) durumuna gerçekleşir.

İşletim sisteminde, birden fazla hazır durumda bulunan bu süreçlerden hangisinin işlemciyi kullanacağına karar veren mekanizmaya *zamanlayıcı(scheduler)* denilir ve kullanılan algoritmaya da *zamanlama algoritması (scheduling algorithm)* denilir.

Zamanlama (Scheduling)

Zamanlayıcı işlemciyi çok iyi kullanmalıdır ve işlemciden maksimum performans alınmasını sağlamalıdır. Süreçler arasında geçiş işlemi son derece pahalı bir işlemdir.

*Yeni sürece geçiş işleminde, kullanıcı kipinden çekirdek kipine geçiş gerçekleşir.

*Diğer sürece atlanmadan önce, mevcut sürecin tüm bilgisi saklanmalıdır.

*Yeni sürecin bellek haritası bellek yönetim birimi (MMU) tarafından yüklenmelidir.

Zamanlama (Scheduling)

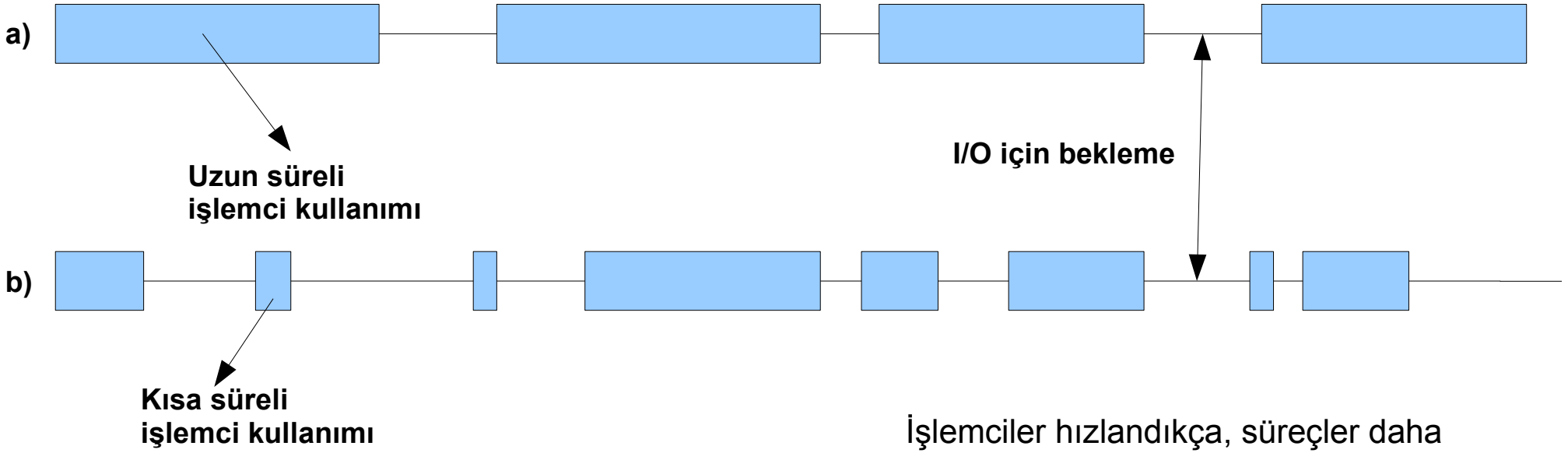
*Sonunda yeni süreç çalıştırılmalıdır.

*Ayrıca süreçler arası geçişte tüm önbellek(cache) verisi geçersiz hale gelir.

*İşlemci tarafından yapılan her istek bir kaçırma(miss) olur ve bellek ten önbelleğe yüklenme gerektirir.

Süreçlerin Davranışları

Neredeyse tüm süreçler disk(I/O) işlemleri ile doludur. İşlemci bir süre çalışır, sonra bir dosya okumak ya da yazmak için bir sistem çağrısı yapılır.



a) daki duruma, işlem istekli(compute-bound)
b) deki duruma, girdi/çıktı istekli(I/O-cound) denilir.

İşlemciler hızlandıkça, süreçler daha fazla I/O istekli hale gelirler.

Zamanlamada temel fikir, I/O istekli süreçlere hemen bir şanş verilmeli ve kısa sürede I/O isteklerini yapmalarıdır.

Zamanlama(Scheduling) Ne Zaman Yapılmalıdır ?

Zamanlamanın yapılmasına ihtiyaç duyulan zamanlar:

1. Yeni bir süreç oluşturulduğunda, ana sürecin ya da çocuk sürecin çalıştırılmasının kararı verilmelidir. Bu iki süreçte hazır durumuna gelir.

2.Bir sürecin çalışması bittiğinde ve işlemciyi kullanmayı bıraktığında karar verilmelidir.

3.Bir süreç I/O işlemi için bloklandığında başka bir süreç seçilmelidir.

4.Bir I/O kesmesi geldiğinde bir zamanlama kararı yapılmalıdır. Eğer kesme bir I/O cihazından geliyorsa, bu I/O cihazından gelecek olan veriyi bekleyen süreç çalıştırılmalıdır.

Zamanlama(Scheduling) Ne Zaman Yapılmalıdır ?

Bilgisayarlarda genelde bir donanım saati(clock) bulunur. Bu saat periyodik olarak işlemciye kesmeler gönderir.

Bu kesmeler ile işlemci zamanlama işlemlerini gerçekleştirir.

Zamanlama işlemi her saat kesmesinde ya da belirli sayıda olduğunda gerçekleştirilmelidir.

Zamanlama algoritmaları, saat kesmelerini nasıl kullandıklarına göre ikiye ayrılır;

a)Kesintisiz (non-preemptive)
b)Kesintili (preemptive)

Zamanlama(Scheduling) Ne Zaman Yapılmalıdır ?

a)Kesintisiz (non-preemptive): Bir süreç çalışması için seçilir. Kendi kodunda bulunan bir I/O isteği ile bloklanıncaya kadar ya da kendi isteği ile işlemciden çıkıncaya kadar çalışır.

b)Kesintili (preemptive): Bir süreç seçilir ve sabit, belirli bir maksimum süreye kadar çalıştırılır. Bu süre bittiğinde, süreç hala işlemcide çalışıyorsa askıya alınır ve zamanlayıcı(scheduler) başka bir süreci çalışması için seçer.

Zamanlama(Scheduling) Algoritmalarının Kategorileri

1. Toplu İş (batch): Kullanıcılar acil olarak cevap beklemedikleri için uzun süreli kesintili ya da kesintisiz zamanlama algoritmaları kullanılabilir. Süreç değişimi az olduğu için performans artar.

2. Etkileşimli sistemlerde: Kullanıcının sürekli veri alıp verdiği sistemlerdir. Kesintili zamanlama zorunludur.

3. Gerçek zaman sistemlerinde: Belirli bir amaç için kullanılan programlar olduğu için, sürelerin iyi ayarlandığı kesintili zamanlama algoritmaları kullanılmalıdır.

Zamanlama(Scheduling) Algoritmalarının Amaçları

Tüm sistemlerde;

* Adil olma: Her süreç işlemciyi adil kullanmalıdır.
* Politika Zorlama: Belirtilen politika kullanılmalıdır.
* Denge: Sistein tüm parçaları meşgul olmalıdır.

Toplu İş Sistemlerinde;

*Üretilen İş: Saat başına üretilen iş maksimum olmalıdır.

*Dönüş Süresi:Sürecin sunumu ve sonlanması arasındaki sürenin minumum olmasıdır.

*İşlemci Kullanımı: İşlemci her zaman meşgul olmalıdır.

Zamanlama(Scheduling) Algoritmalarının Amaçları

Etkileşimli Sistemlerde;

*Cevap Süresi: İsteklere hızlı şekilde cevap verilmelidir.

*Oratılı Olma: Kullanıcının beklentilerini karşılamalıdır.

Gerçek Zaman Sistemleri;

*Son teslim süresine riayet etme: Veri kaybından sakınmalıdır.

*Tahmin Edilebilirlik:Çoklu ortam sistemlerinde nitelik bozulmasından sakınmalıdır.

Toplu İş(Batch) Sistemlerde Zamanlama(Scheduling)

1. İlk gelene ilk servis yapılır (First come first served)

Kesintisiz ve en kolay zamanlama algoritmasıdır. İlk gelen süreç ilk hizmeti alır. Bu algoritmada süreçler istek sıralarına göre işlemcide çalışırlar.

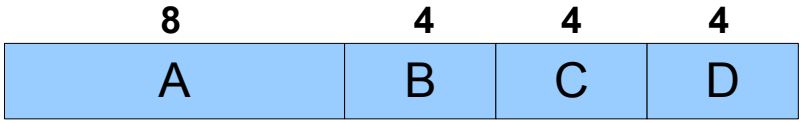
Basitçe hazır durumdaki süreçleri tutan bir kuyruk(queue) vardır. Kuyruktaki süreçler sırayla çalıştırılırlar. Çalışan süreç bloklanınca bekler. Bloklanmış süreç tekrar hazır durumuna geçtiğinde yeni bir süreç gibi davranılarak kuyruğun sonuna eklenir.

Toplu İş(Batch) Sistemlerde Zamanlama(Scheduling)

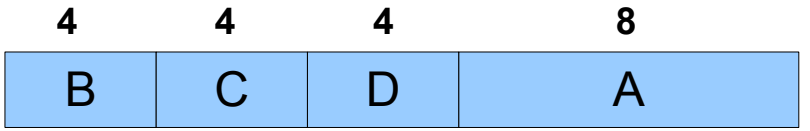
2. En Kısa Süreli İş İlk (Shortest Job First)

Kesintisiz çalışan algoritmadır. Süreçlerin toplam çalışma sürelerinin belirli olduğu düşünülür. Zamanlayıcı, kuyrukta bulunan süreçlerden en kısa sürede tamamlanacak olan süreci seçer.

Asıl sırasında çalıştırırsak;
A :8
B :12
C :16
D :20
birim süre sonra işlerini bitirirler.
Ortalama süre 14 birimdir.



En kısa süreli süreç çalışırsa;
B :4
C :8
D :12
A :20
birim süre sonra işlerini bitirirler.
Ortalama süre 11 birimdir.



Toplu İş(Batch) Sistemlerde Zamanlama(Scheduling)

3. Çalışma süresi en az kalanı çalıştırma (Shortest Remaining Time Next)

En kısa süreli iş algoritmasının, bir kesintili uyarlamasıdır. Kalan çalışma süresi en az olan seçilir. Yeni bir iş geldiğinde, mevcut işin kalan süresi ile yeni sürecin kalan süresi karşılaştırılır; küçük olan tercih edilir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

1. Dönüşümlü Zamanlama (Round-Robin Scheduling)

Her sürece, çalışabileceği maksimum süre atanır. Sürece verilen bu zaman aralığına ***quantum*** denilir.

Eğer süreç, quantumunun sonunda hala çalışıyorsa askıya alınır ve başka bir sürece geçilir.

Sürecin quantumu bitmeden süreç bloklanırsa ya da çalışması sonlanırsa başka bir sürece geçilir.

Bir listede çalışabilir süreçler tutulur. Süreç quantumunu tükettiğinde listenin sonuna eklenir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

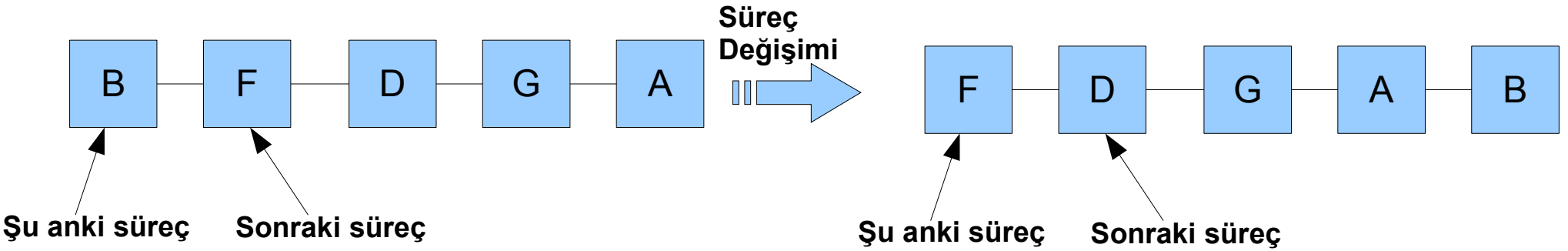
1. Dönüşümlü Zamanlama (Round-Robin Scheduling)

Dönüşümlü algoritmada en önemli konu quantumun uzunluğudur.
Bir süreçten başka bir sürece geçiş işlemide sistemde belirli bir süre almaktadır.

Örneğin;Süreçler arası değişim işleminin 1msaniye aldığını varsayalım. Quantum miktarını 4msaniye seçilirse, süreç değişimleride 1msec alır. İşlemcinin %20 si yönetimsel ek yük işlemleri için harcanır.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

1. Dönüşümlü Zamanlama (Round-Robin Scheduling)



İşlemci etkinliğini arttırmak için, quantum miktarını 100msaniye seçelim. Yönetimsel ek yük %1 olacaktır. Örneğin 10 kullanıcı bu sistemde aynı anda enter tuşuna basarak aynı işi yapmak istesinler. Tüm kullanıcılar 100msaniye bekler. En sonda bulunan kullanıcı cevabı alabilmek için yaklaşık olarak 1 saniye bekler.

Quantumu çok kısa tutmak işlemci etkinliğini azaltır, çok uzun tutmak isteklere verilen cevabı azaltır. Ortalamada 20-50 m saniye tercih edilir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

2. Öncelik Zamanlama (Priority Scheduling)

Dönüşümlü zamanlama algoritması dolaylı olarak tüm süreçlerin eşit öncelikli olduğunu farzeder. Süreçler üzerindeki dış etkilerle süreçlerin farklı önceliklerde olması gerekir. Bu durumda öncelik zamanlama algoritması kullanılır.

Her sürece bir öncelik(priority) değeri atanır ve çalışabilir süreçlerden yüksek öncelikli olanların çalışmasına izin verilir.

Örneğin, bir video filmi gösteren süreç ile arka planda email atan süreç aynı öncelikte olmaz.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

2. Öncelik Zamanlama (Priority Scheduling)

Yüksek öncelikli süreçlerin sonsuza kadar çalışmasını engellemek için, her saat sinyalinde sürecin önceliği düşürülür. Bu işlem, sürecin önceliğini en yakın rakibinin önceliğine getirdiğinde süreçler arasında geçiş işlemi yapılır.

Alternatif olarak her sürece en yüksek öncelik değeri verilir. Quantumu bittiğinde önceliği azaltılır ve kendinden sonraki sürecin çalışmasına izin verilir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

2. Öncelik Zamanlama (Priority Scheduling)

Öncelik değerleri değişken ya da durağan olabilir. Örneğin, bir askeri sistemde; generallere 100, albaylara 90, bin başılara 80,... atanabilir.

Sistem tarafından belirli sistem hedeflerinin sağlanması amacıyla süreçlerin öncelikleri değiştirilebilir. Örneğin, bazı süreçler yüksek şekilde I/O bağımlıysa ve I/O için bekliyorsa, bu süreç işlemcide çalışmak istediğinde hemen işlemcide çalışmalıdır. Yüksek I/O bağımlı süreçlerin işlemcide fazla çalıştırılması gereksiz bellek kullanmalarını sağlar.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

2. Öncelik Zamanlama (Priority Scheduling)

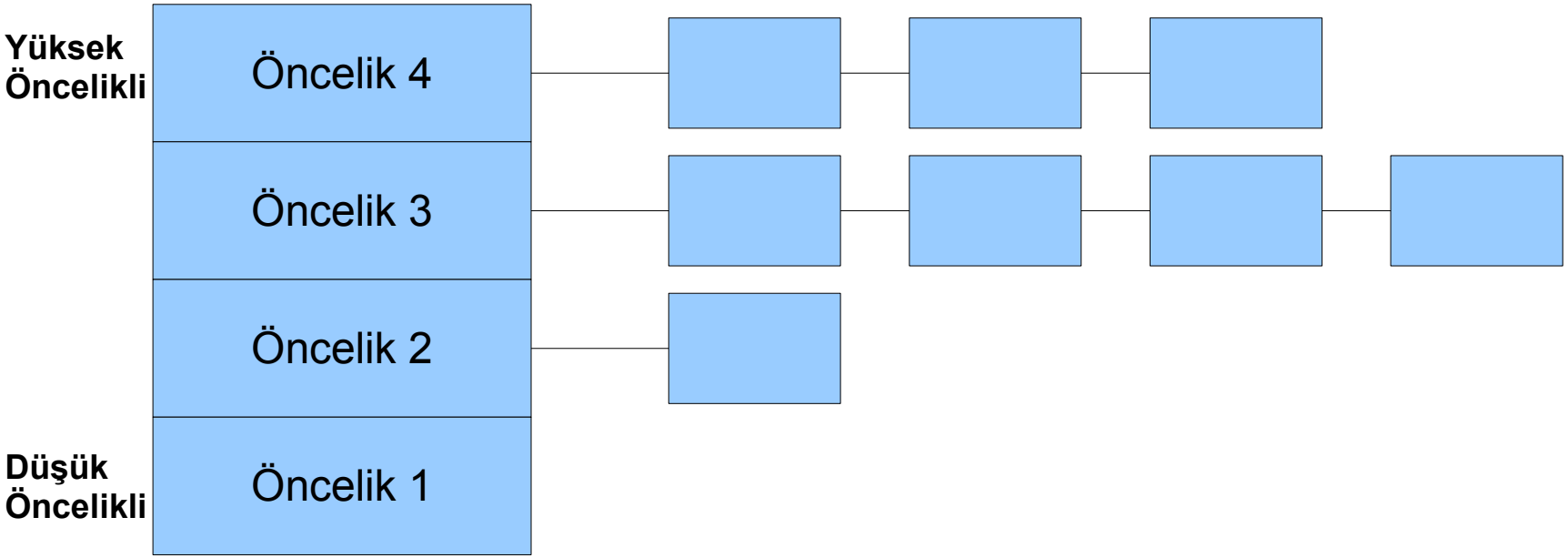
I/O bağımlı süreçlere basitçe ***1/f*** öncelik değeri atanır. ***f***, bir sürecin ***kullandığı son quantum*** kesiridir.

Bir süreç kendisine verilen 50 m saniye quantumunun 1m saniyesini kullanmışsa, önceliği 50 dir. Bloklanmadan önce 25 m saniye çalışmışsa önceliği 2 dir. Tüm quantumunu kullanmışsa önceliği 1 dir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

2. Öncelik Zamanlama (Priority Scheduling)

Yaygın olarak, süreçleri gruplayarak gruba öncelik sınıfı atanır.



Yukarıdan aşağıya doğru dönüşümlü olarak çalıştırılırlar. Yukarıdaki grubun çalışması bitmeden aşağıdakiler çalışamaz.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

2. Öncelik Zamanlama (Priority Scheduling)

Çoklu Kuyruklar (Multiple Queues)

Öncelik kuyruklarında yüksek öncelikli sınıfa 1 quantum, onun altındakine 2 quantum, onun altındakine 4 quantum şeklinde zaman aralığı atanır. Bir süreç kendisine atanan quantumun hepsini kullandıktan sonra bir alttaki kuyruğa aktarılır.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

3. En Kısa Süreç Sonraki (Shortest Process Next)

Etkileşimli sistemler genelde şu mantık ile çalışırlar, komut için bekle, komutu yürüt,...
Her komutu ayrı bir iş gibi görürsek, en kısa olanı ilk önce çalıştırırsak toplam cevap süresini minumumlaştırabiliriz.

Problem mevcut süreçlerden hangisi en kısa olandır ?

Etkileşimli Sistemlerde Zamanlama(Scheduling)

3. En Kısa Süreç Sonraki (Shortest Process Next)

Kullanılan yaklaşımda sürecin son davranışından çalışma süresinin tahmin edilmesidir. Tahmin edilen sürelerden en kısa süreye sahip olan süreç seçilir ve çalıştırılır.

Varsayalım ki, ilk tahmin edilen süre T0 olsun. Bir sonraki çalışma süresi T1 olsun. Tahminimizi ağırlıklı toplamları kullanarak güncelleyebiliriz.

$T2=aT0 + (1-a)T1$

a değerinin seçimi ile sürecin eski çalışma sürelerini hatırlamasını ya da unutmasını sağlayabiliriz.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

3. En Kısa Süreç Sonraki (Shortest Process Next)

Örneğin a=1/2 için;

T0, T0/2+T1/2, T0/4+T1/4+T2/2, T0/8+T1/8+T2/4+T3/2 ,...

3. çalışmadan sonra T0 ın ağırlığı 1/8 e düşer.

Serideki bir sonraki değerin ağırlıklı ortalama yardımıyla önceki ve şu anki değer kullanılarak hesaplanması yöntemine yaşlandırma (aging) denilir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

4. Garantili Zamanlama (Quaranteed Scheduling)

Kullanıcılara belirli sözler verme temelinde çalışır. Örneğin, sistemde n kullanıcı varsa, her biri işlemcinin 1/n nini kullanır. Sistemde her sürecin çalışma süreleri ayrıntılı olarak izlenir. Her seferinde en düşük süreli çalışmış olan süreç en yakın rakibinin süresine ulaşıncaya kadar çalıştırılır.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

5. Piyango Zamanlama (Lottery Scheduling)

Süreçlere farklı sistem kaynakları için piyango bileti verilir.(Örneğin; işlemci zamanı için). Bir zamanlama kararı verileceği zaman rastgele bir piyango bileti seçilir ve bu bilete sahip olan süreç çalıştırılır.
Bir sürecin birden fazla piyango bileti olabilir. Örneğin, bir süreç 100 biletten 20 tanesine sahipse, %20 çalışma şansı bulunmaktadır. Süreçler arasında istenilirse biletlerin paylaşımı olabilir.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

5. Adil Paylaşım Zamanlama (Fair-Share Scheduling)

Şu ana kadar bakılan algoritmalarda, süreci çalıştıranı yani sahibini düşünmedik. Birinci kullanıcı 9 süreç çalıştırmış, ikinci kullanıcı 1 süreç çalıştırmış olsun. Birinci kullanıcı işlemcinin %90 nını kullanırken, ikinci kullanıcı %10 nunu kullanabilir.

Bu modelde her kullanıcı işlemcinin belirli bir yüzdesini kullanabilir. Eğer sistemde iki kullanıcı varsa, her biri işlemcinin %50 sini kullanmalıdır. Algoritma bu kullanım yüzdesini sağlayacak şekilde süreçleri seçer.

Etkileşimli Sistemlerde Zamanlama(Scheduling)

5. Adil Paylaşım Zamanlama (Fair-Share Scheduling)

Örneğin, sistemde iki kullanıcı çalışsın. Birinci kullanıcının A,B,C,D süreçleri; ikinci kullanıcının da E süreci bulunsun. Eğer dönüşümlü zamanlama kullanılırsa süreçlerin çalışma sırası şu şekilde olabilir;

AEBECEDEAEBECEDEAEBECEDE...

İşletim Sistemlerine Giriş

Zamanlama (Scheduling)