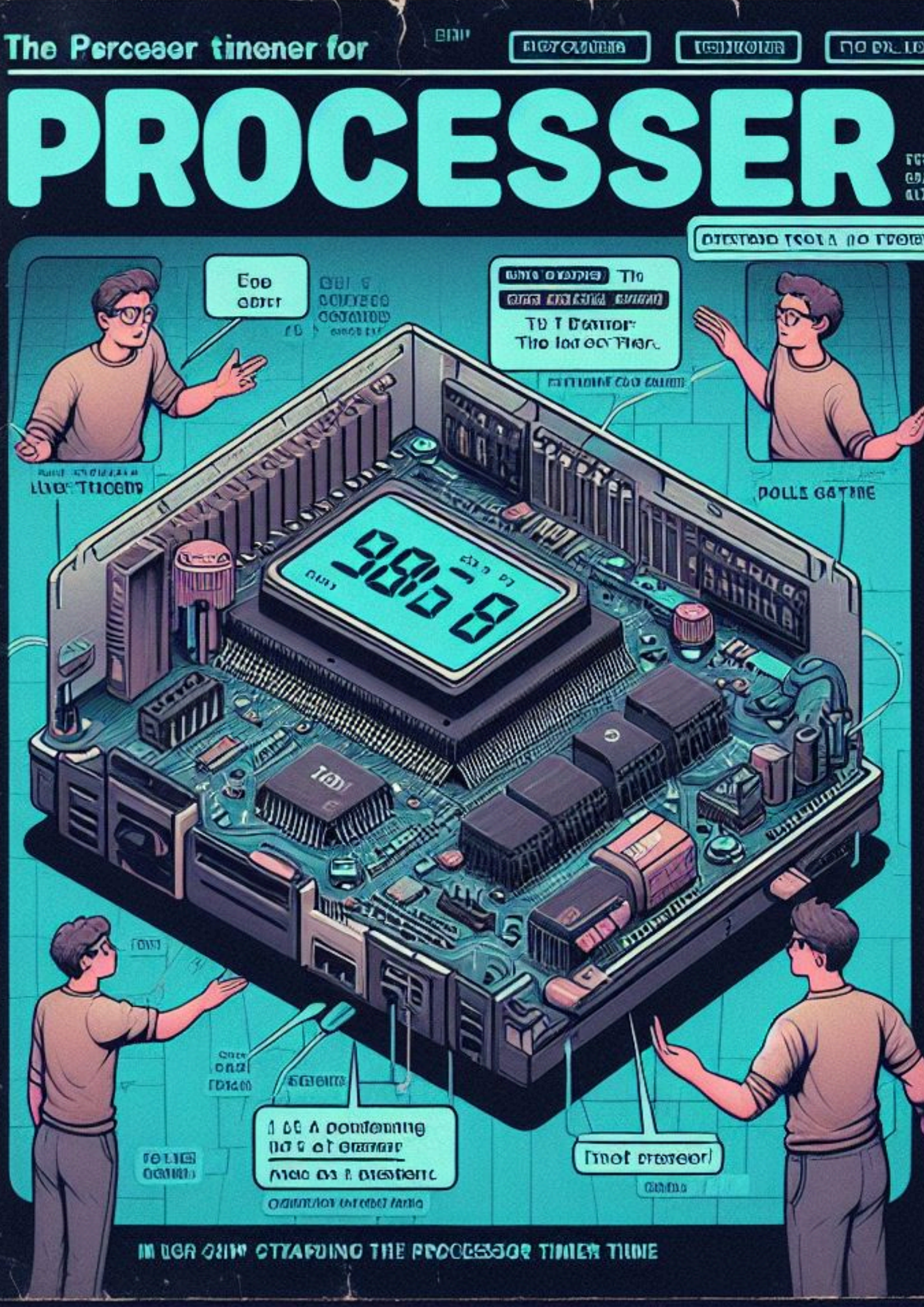

The Perceaer tinener for
PROCESSER

İçindekiler

04 First-Come, First-Served (FCFS) Scheduling

05 Shortest Job First - SJF İşlemci Zamanlama Algoritması

06 İşlemci Kullanım Süresinin Belirlenmesi
Priority Tabanlı – Scheduling Algoritması

07 Round Robin Scheduling

08 Çok Kuyruklu - İşlemci Zamanlama Algoritması

09 Çok-Seviye Geri Besleme Kuyruğu

10 Hyper-Threading

Biz Kimiz?

Özgün Düşler
Bİlgisayar mühendisliğinde okuyan 4 arkadaş olarak yola çıkmış bu alanda anlaşılması zor olan bir konuyu farklı yollar ile açıklamayı amaç edinmiş bir grubuz.

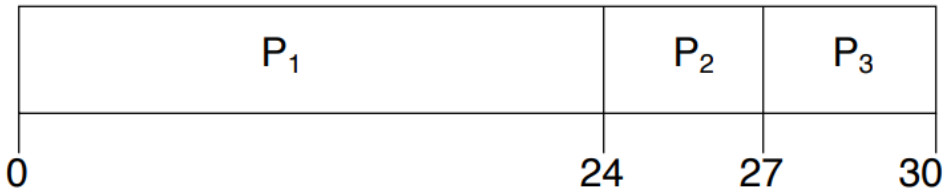
Editör
Zeynep Acar

Hazırlayanlar
Berkay Ayköse
Berna Varol
Emre Şen
Zeynep Acar

Projemiz
Ekip arkadaşlarımızla birlikte bir anket yaptık ve hep beraber bu konuyu size açıklayabilmek için bu fanzini oluşturduk.

First-Come, First-Served (FCFS) Scheduling

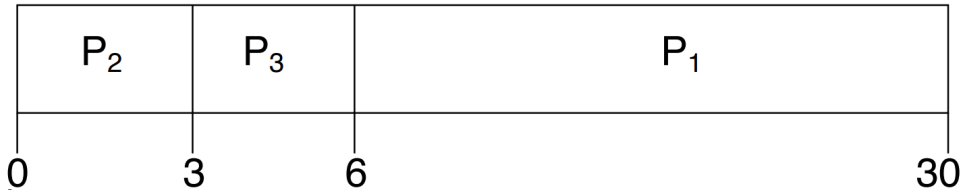
İşlemciye gelen işlerin, gelme sırasına göre ilerlediği bir zamanlama planlayıcısıdır. İşlerin uzunluğuna ya da en son kullanılma sırasına bakılmaz.
Örneğin uzunlukları sırasıyla 24,3 ve 3 olan 3 işin işlemciye geldiğini düşünelim. O halde gant şeması aşağıdaki şekildedir.



Bekleme zamanı işlerin başlama zamanlarına bakılarak hesaplanır. Ortalama başlangıç zamanları ise tüm işlerin başlangıç zamanlarını toplanıp iş sayısına bölünür. Buradaki bekleme zamanı;

Bekleme zamanları: $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
Ortalama bekleme zamanı: $(0 + 24 + 27)/3 = 17$

İşlemlerin şu şekilde geldiğini düşünelim; P2,P3,P1



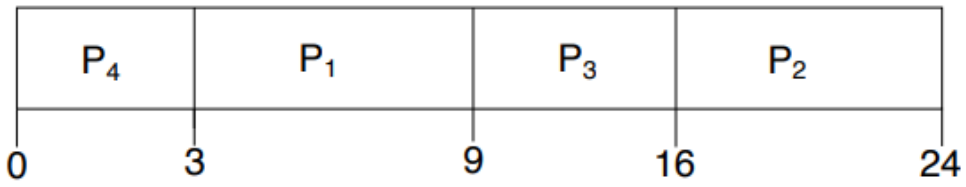
Bekleme zamanları: $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
Ortalama bekleme zamanı: $(6 + 0 + 3)/3 = 3$

Görüldüğü gibi kısa süren işlerin daha önceden işleme alınması bekleme süresinin düştüğü görülür.

En Kısa İş Önce
SJF İşlemci Zamanlama Algoritması

Sürekli en kısa işin seçildiği algoritmadır. Amaç bekleme süresinin optimal olmasıdır. İşlemcinin süreleri sadece tahmin edilebilir. Bu tahmin daha önceki işlemci süreleri kullanılarak yapılır.
İlk olarak en kısa iş seçilir ve bu şekilde en kısa seçilerek devam eder.

İşlem	İşlem Süresi
P_1	6
P_2	8
P_3	7
P_4	3



Ortalama bekleme zamanı = (3 + 16 + 9 + 0) / 4 = 7

İşlemci Kullanım Süresinin Belirlenmesi

İşlemlerin gelme sırasının ve uzunluklarının daha önceden bilinmesi mümkün olmadığından sadece tahmin edilebilir. Daha önceki işlemci kullanım süreleri kullanılarak üssel ortalama (exponential averaging) yöntemiyle tahmin edilebilir.

1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Define : $\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n.$

Priority Tabanlı – Scheduling Algoritması

Her bir işleme önem sayısına göre bir öncelik sayısı (tamsayı) atanır. Sistem bu sayılara göre iş takibi yapar. Burada en önemli olan işe en küçük sayı verilir.Açlık (starvation) düşük öncelikli işlemler hiçbir zaman çalışmayabilir. Çözüm olarak yaşlandırma (aging) yapılabilir. Bu işlem zaman geçtikçe bekleyen işlemlerin önceliğini arttırarak (katsayısını düşürerek) bütün işlerin yapılmasını sağlar.
Shortest Job First, öncelik tahmini kullanım süresi olmak kaydıyla, öncelik tabanlı bir işlemci zamanlaması algoritmasıdır.

Round Robin Scheduling

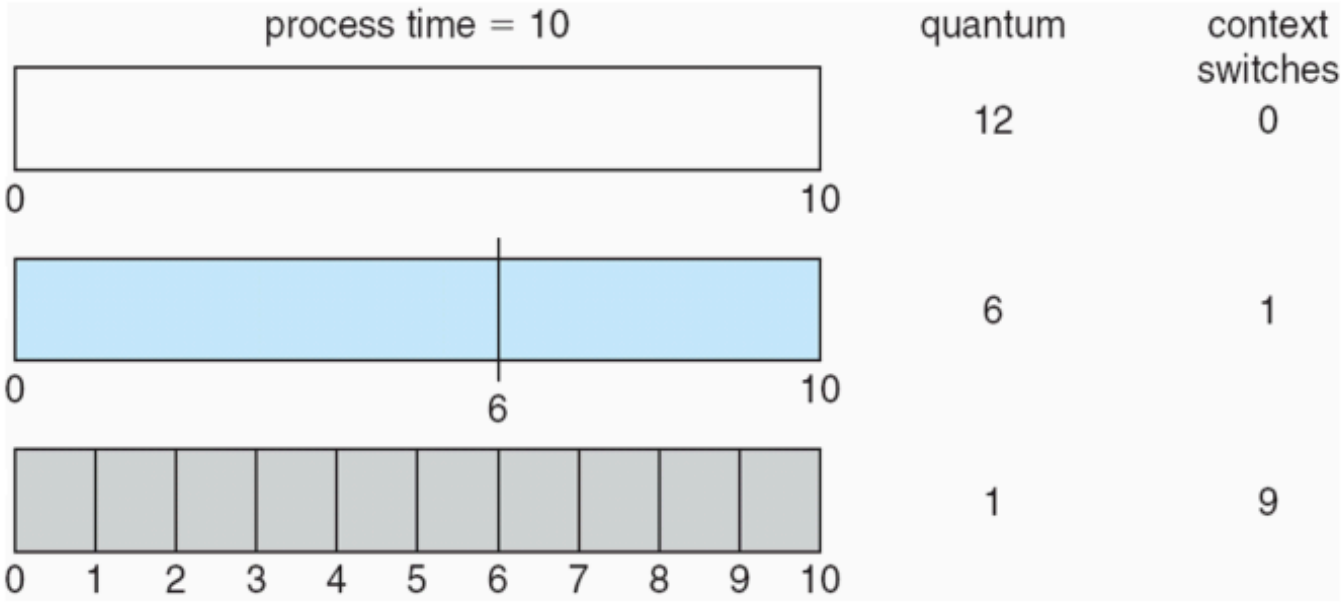
Her bir işlem işlemci zamanının küçük bir birimini alır. Buna zaman quantumu adı verilir.Bu zaman dolduğunda işlem kesilir ve hazır kuyruğunun sonuna eklenir.

Hazır kuyruğunda n tane işlem varsa ve zaman kuantumu q ise, her bir işlem CPU zamanının 1/n kadarını kullanır. Her bir işlem q birimi geçmez ve n-1 bekleme süresini geçmez.

• q büyükse; ilk gelen önce (FIFO)

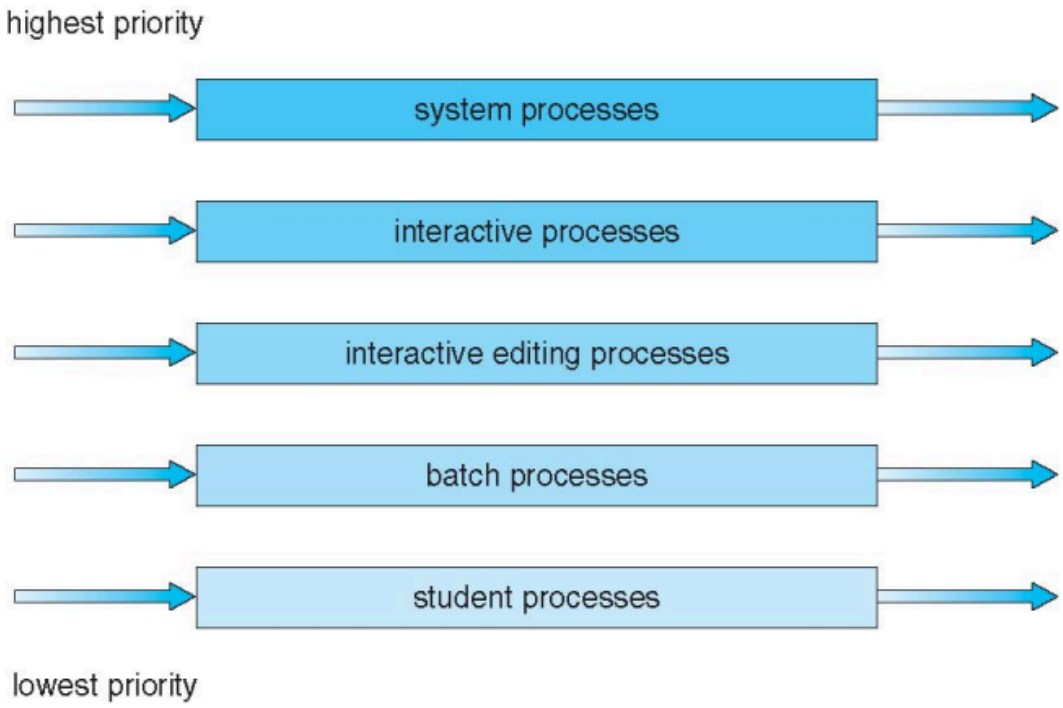
• q küçükse; q context switch süresine oranla daha büyük olmalıdır.

Aksi halde sistem verimsiz çalışır



Çok Kuyruklu - İşlemci Zamanlama Algoritması

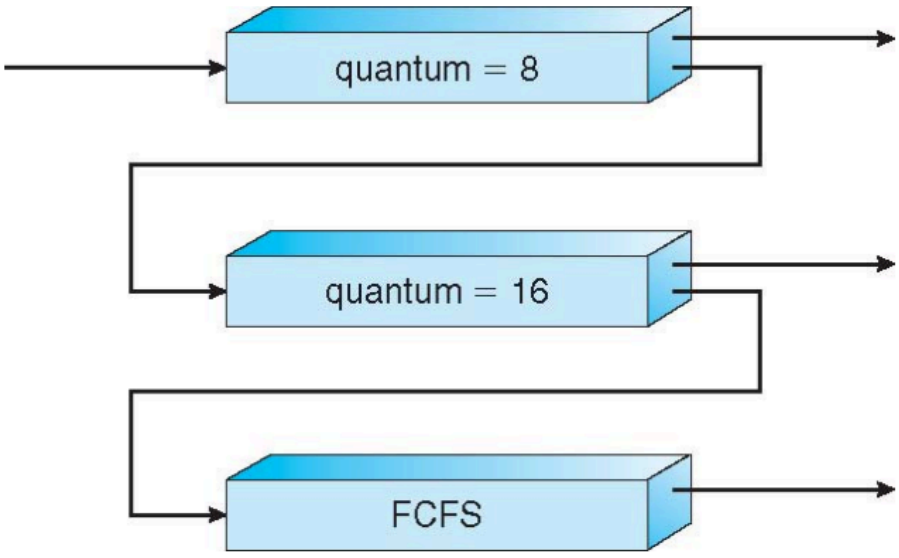
Çok kuyruklu işlemci zamanlama algoritması, işlemcideki işleri farklı öncelik seviyelerine göre ayrı kuyruklarda tutan bir zamanlama yaklaşımıdır. Bu kuyruklar genellikle işlem önceliğine göre düzenlenir. Öncelikli işler daha yüksek öncelikli kuyruklarda bulunurken, daha az öncelikli işler daha düşük öncelikli kuyruklarda bulunur.
Multilevel Queue (MLQ) sistemi genellikle farklı iş türleri veya gereksinimleri için kullanılır.
MLQ sistemi, her kuyruğun kendine özgü bir zamanlama algoritması kullanmasına izin verir. Örneğin, yüksek öncelikli kuyruklar genellikle kısa süreli işler için kullandıkları Round Robin algoritmasını kullanabilirken, daha düşük öncelikli kuyruklar uzun süreli işler için kullandıkları First Come First Served (FCFS) algoritmasını kullanabilir.
Bu sistem, farklı iş türlerinin ihtiyaçlarını daha iyi karşılayabilir ve sistem performansını artırabilir. Ancak, doğru öncelik seviyelerinin belirlenmesi ve farklı kuyruklar arasında adil bir işlemci kaynağı paylaşımının sağlanması önemlidir.



Çok-Seviye Geri Besleme Kuyruğu

Çok-seviyeli geri besleme kuyruğu (Multilevel Feedback Queue - MLFQ), işlemcideki işleri birden fazla kuyrukta organize eden ve işlemcilere farklı önceliklerde hizmet veren bir zamanlama algoritmasıdır. MLFQ, işlemcilere hizmet veren kuyruklar arasında dinamik olarak işlem taşımak için geri beslemeyi kullanır. Bu geri bildirim, işlerin önceliklerini ve işlemcilere ne kadar süre tahsis edileceğini belirlemek için kullanılır.

Birden Fazla Kuyruk Seviyesi: İşler, genellikle önceliklerine göre farklı kuyruklara atanır. Daha yüksek önceliğe sahip işler, daha yüksek öncelikli kuyruklarda bulunur ve daha hızlı hizmet alır. Öncelik seviyeleri, işin ne kadar süredir kuyrukta beklediğine veya ne kadar süre işlemciye sahip olduğuna bağlı olarak değiştirilebilir.
Geri Besleme Mekanizması: İşlerin performansına bağlı olarak kuyruklar arasında hareket eden bir geri besleme mekanizması vardır. Örneğin, bir iş uzun süreli işlemci kullanırsa veya sık sık kesintiye uğrarsa, önceliği düşük olan bir kuyruğa taşınabilir.
Zamanlama Algoritması: Her bir kuyruğun kendi zamanlama algoritması vardır. Örneğin, yüksek öncelikli kuyruklar için Round Robin, düşük öncelikli kuyruklar için ise First Come First Served (FCFS) gibi farklı algoritmalar kullanılabilir.



Hyper-Threading

Hyper-Threading (HT), Intel tarafından geliştirilen bir işlemci teknolojisidir. Bu teknoloji, işlemcinin tek bir fiziksel çekirdeği üzerinde birden fazla sanal işlemci çekirdeği oluşturarak çoklu iş parçacığı yürütme yeteneğini artırır. Hyper-Threading, işlemciyi daha verimli bir şekilde kullanarak işlemci kaynaklarının daha etkin bir şekilde kullanılmasını sağlar.
Geleneksel bir işlemci, her bir çekirdek üzerinde sadece bir iş parçacığı yürütebilir. Ancak Hyper-Threading teknolojisi kullanıldığında, her fiziksel çekirdek birden fazla mantıksal çekirdek olarak davranır. Bu, işletim sistemi ve yazılımların işlemciyi daha fazla iş parçacığı yürütmek için kullanmasına olanak tanır.

Hyper-Threading'in avantajları şunlardır:

1. Daha iyi işlemci kaynağı kullanımı: İşlemci, birden fazla iş parçacığını aynı anda çalıştırarak daha etkin bir şekilde kullanılır.
2. Daha iyi çoklu görev performansı: Birden fazla iş parçacığını aynı anda yürütebilen işlemciler, çoklu görevlerde daha iyi performans gösterir.
3. Daha hızlı tepki süreleri: Hyper-Threading, daha fazla iş parçacığını eş zamanlı olarak işleyerek sistem yanıt sürelerini iyileştirir, özellikle yoğun iş yüklerinde veya çoklu görevlerde.