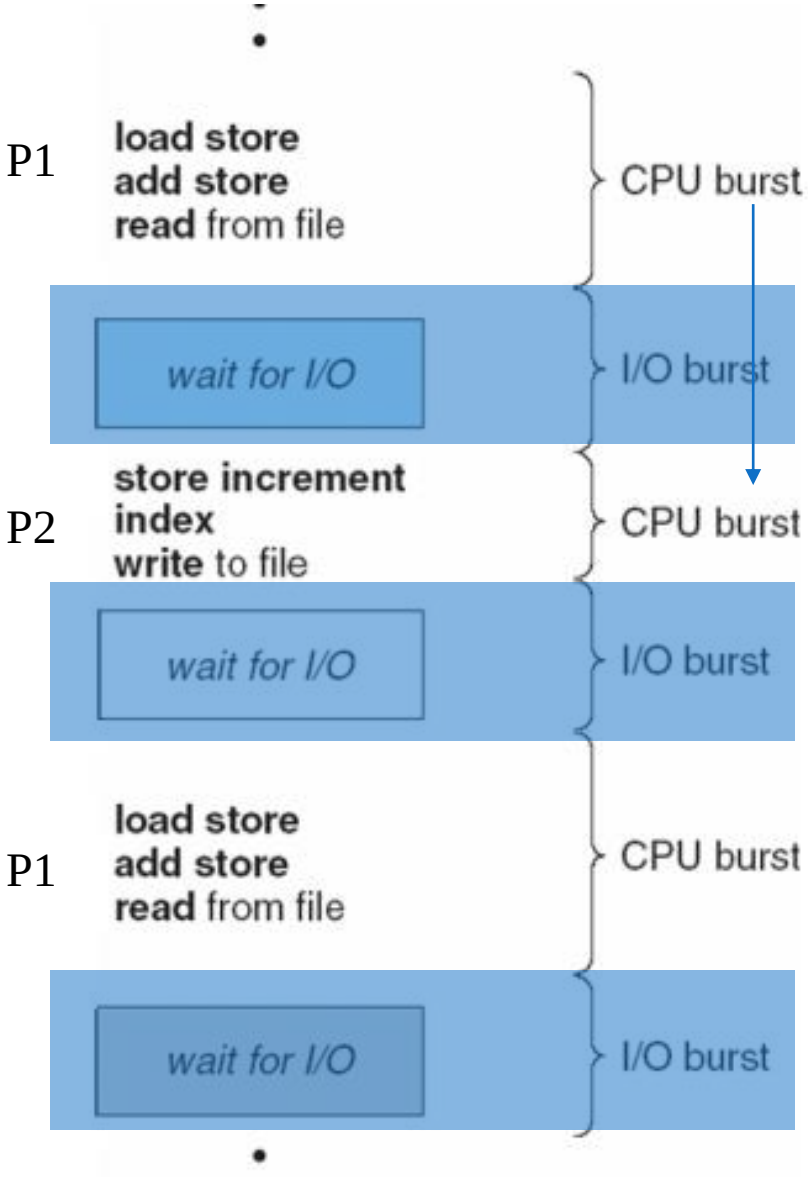
İşlemci Zamanlaması

Dr. Günay TEMÜR

Bölüm 5: İşlemci Zamanlaması

- Temel Kavramlar
- Zamanlama Kriteri
- Zamanlama Algoritmaları
- İş Parçacığı Zamanlaması
- Çok-İşlemci Zamanlaması
- Algoritmaların Değerlendirilmesi



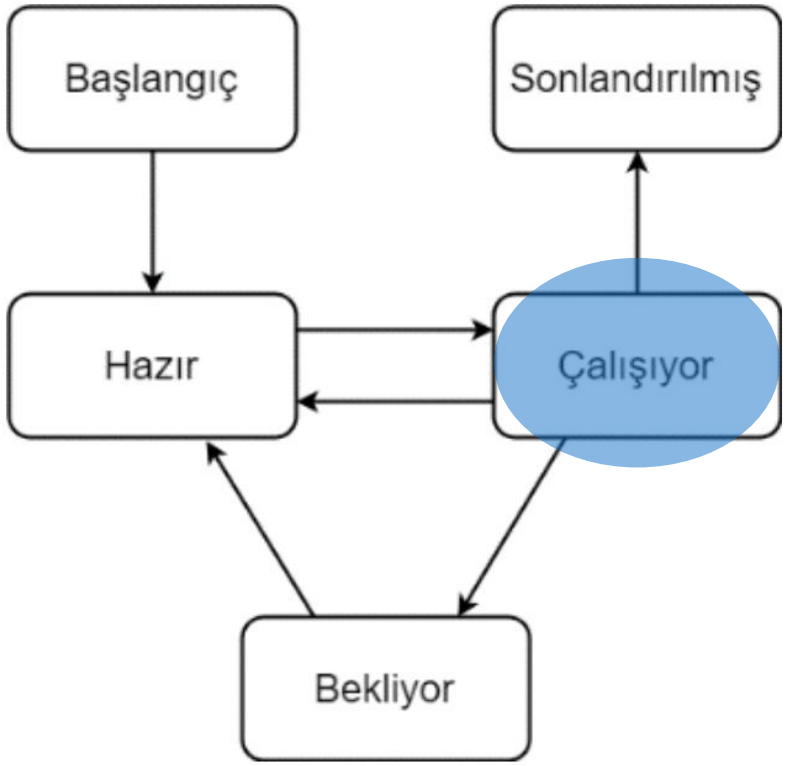
P1 için; PCB
İşlem numarası
İşlem sayacı
İşlemci zamanlama bilgileri
Hafıza yönetim bilgileri
Giriş-çıkış durum bilgileri

Temel Kavramlar

- Çoklu programlamalı sayesinde CPU kullanımının optimize edilmesi
- CPU–I/O İşlem Döngüsü – Bir işlemin çalıştırılması birbirlerini takip eden
  - (1) CPU kullanımı ve
  - (2) I/O beklemesi döngüsünden oluşur
- CPU işleme süresi dağılımı

İşlemci Zamanlayıcısı

- Çalışmaya hazır şekilde hafızada bekleyen işlemlerden birini seçerek işlemciyi ona ayırır
- İşlemci zamanlaması şu durumlarda gerçekleşebilir:
  - 1. Çalışma durumundan (running state) bekleme durumuna (waiting state) geçişte
  - 2. Çalışma durumundan hazır durumuna (ready state) geçişte
  - 3. Bekleme durumundan hazır durumuna geçişte
  - 4. İşlem sonlandığında
- 1 ve 4 durumlarında zamanlama: geçişsiz (non-preemptive) – eski işletim sistemleri (windows 3.x gibi)
- Diğer durumlarda: geçişli (preemptive)



Görevlendirici (Dispatcher)

Zamanlayıcılar:
Long – Short – Middle
Uzun vade
Kısa vade
Orta vade

- Görevlendirici modül CPU kontrolünü kısa dönem zamanlayıcı (short term scheduler) tarafından seçilen işleme devreder:
  - Ortam değiştirme (switching context)
  - Kullanıcı moduna geçme
  - Kullanıcıya ait program yeniden başlatıldığında programdaki uygun konuma atlama
- Görevlendirme gecikme süresi (dispatch latency) – Görevlendiricinin bir programı sonlandırıp diğerini başlatması için gereken süre

Zamanlama Kriterleri

- İşlemci kullanımı (CPU utilization) – İşlemciyi olabildiğince meşgul tutmak
- Üretilen iş (throughput) – birim zamanda çalışması sonlanan işlemlerin sayısı
- Devir zamanı (turnaround time) – bir işlem sonlanana kadar geçen toplam zaman
- Bekleme zamanı (waiting time) – bir işlemin hazır kuyruğunda (ready queue) toplam bekleme zamanı
- Yanıt süresi (response time) – bir isteğin gönderilmesi ile bu isteğinin yanıtının verilmesi arasında geçen zaman

Zamanlama Algoritması Optimizasyon Kriteri

- Maksimum işlemci kullanımı (Max CPU utilization)
- Maksimum üretilen iş (Max throughput)
- Minimum devir zamanı (Min turnaround time)
- Minimum bekleme zamanı (Min waiting time)
- Minimum yanıt süresi (Min response time)

İlk Gelen Önce-İşlemci Zamanlama Algoritması

First-Come, First-Served (FCFS) Scheduling

İşlem	İşlem Süresi
P_1	24
P_2	3
P_3	3

- İşlemleri şu sırada geldiğini varsayalım: P_1 , P_2 , P_3
  Zamanlama için Gantt şeması:

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

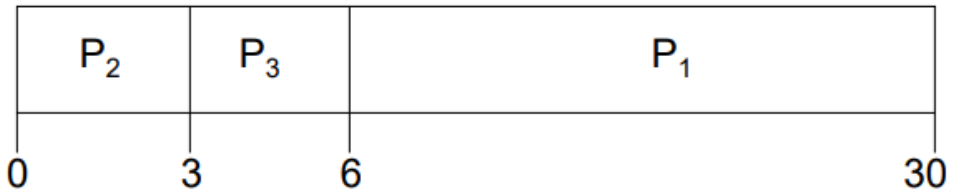
İlk Gelen Önce

Arrived
P1=0
P2=4
P3=1
P4=2

İşlemleri şu sırada geldiğini varsayalım:

P_2 , P_3 , P_1

- Zamanlama için Gantt şeması:



- Bekleme zamanları: $P_1 = 6; P_2 = 0, P_3 = 3$
- Ortalama bekleme zamanı: (6 + 0 + 3)/3 = 3
- Önceki durumdan çok daha iyi
- **Konvoy etkisi (convoy effect):** kısa işlemler uzun işlemlerin arkasında

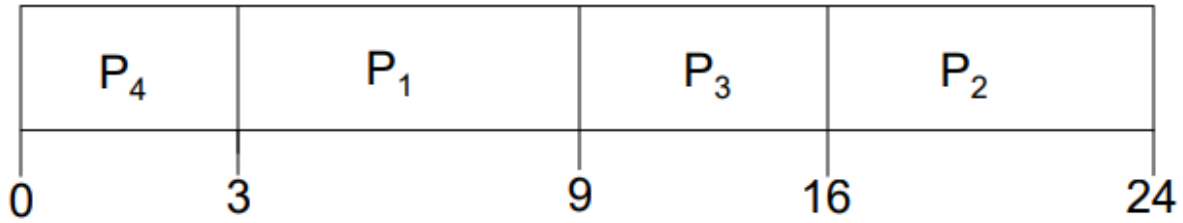
En Kısa İş Önce-İşlemci Zamanlama Algoritması

- Shortest-Job-First (SJF)
- Her işlemi sonraki işlemci kullanım süresi ile ilişkilendirilmeli
- Bu kullanım sürelerini kullanarak en kısa sürecek iş önce seçilmeli
- SJF optimal zamanlama algoritmasıdır – verilen bir iş kümesi için minimum ortalama bekleme süresini sağlar
  - Zorluk, işlemci kullanım sürelerini tahmin etmektir

SJF Örneği

İşlem	İşlem Süresi
P_1	6
P_2	8
P_3	7
P_4	3

SJF zamanlama şeması



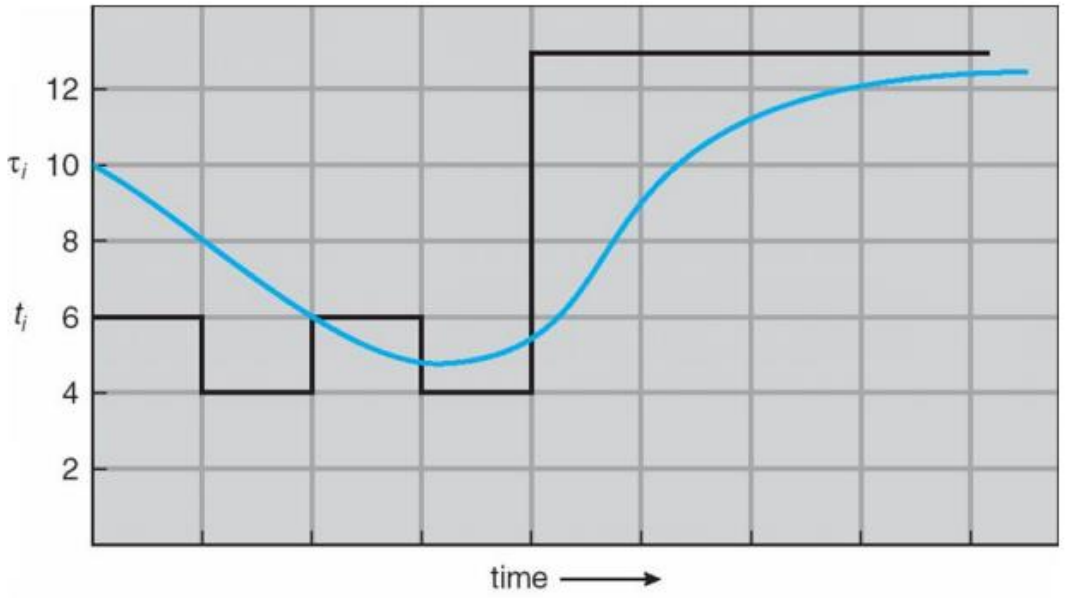
Ortalama bekleme zamanı = (3 + 16 + 9 + 0) / 4 = 7

İşlemci Kullanım Süresinin Belirlenmesi

- 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: $\tau_{n=1} = \alpha t_n + (1-\alpha)\tau_n$.

Üssel Ortalama ile Kullanım Süresi Belirleme



CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

$$\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n = (t_n + \tau_n)/2$$

En Kısa Bekleme Süresi

Process	Arrival Time	Burst Time
P1	0	8-1=7
P2	1	4-1=3-1=2
P3	2	9
P4	3	5

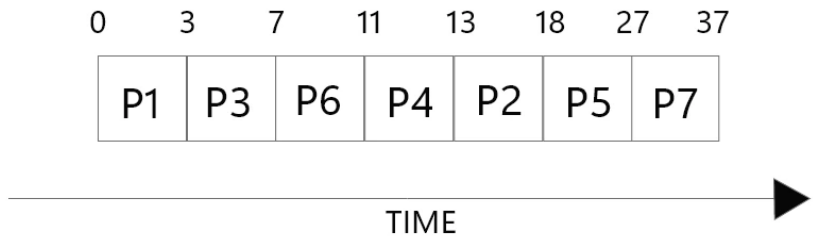
Avarage waiting time = [(10-1)+(1-1)+(17-2)+(5-3)]/4= 26/4=6.5

Öncelik Tabanlı-İşlemci Zamanlama Algoritması

- Priority Scheduling
- Her bir işleme bir öncelik sayısı (tamsayı) atanır
- İşlemci en öncelikli işleme verilir (en küçük tamsayı en yüksek öncelik)
  - Geçişli (preemptive)
  - Geçişsiz (non-preemptive)
- SJF, öncelik tahmini kullanım süresi olmak kaydıyla, öncelik tabanlı bir işlemci zamanlaması algoritmasıdır
  - Açlık Problemi (starvation) – düşük öncelikli işlemler hiçbir zaman çalışmayabilir
  - Çözüm: yaşlandırma (aging) – zaman geçtikçe bekleyen işlemlerin önceliğini arttırma

Öncelik Tabanlı Örnek

Priority Scheduling Algorithm



Process	P1	P2	P3	P4	P5	P6	P7
Burst Time	3	5	4	2	9	4	10
Priority	3	6	3	5	7	4	10
Arrival Time	0	2	1	4	6	5	7

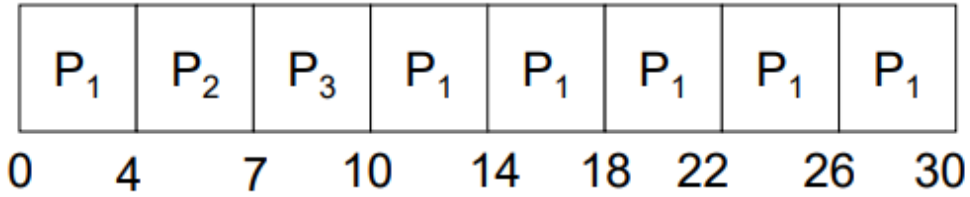
Zaman Dilimli-İşlemci Zamanlama Algoritması

- Round Robin (RR)
- Her bir işlem işlemci zamanının küçük bir birimini alır: zaman kuantumu (time quantum)
- Genellikle 10-100 millisaniye
- Bu zaman dolduğunda işlem kesilir ve hazır kuyruğunun sonuna eklenir
- Eğer hazır kuyruğunda n tane işlem varsa ve zaman kuantumu q ise, her bir işlem CPU zamanının 1/n kadarını (en fazla q birimlik zamanlar halinde) ve hiç bir işlem (n-1)q zaman biriminden fazla beklemez
- Performans
  - q büyükse-> ilk gelen önce (FIFO)
  - q küçükse-> q ortam değiştirme süresine oranla daha büyük olmalıdır. Aksi halde sistem verimsiz çalışır

Zaman Quantum = 4 olduğunda RR

İşlem	İşlem Süresi
P_1	24
P_2	3
P_3	3

Gantt şeması:



Tipik olarak, SJF'den daha yüksek ortalama *devir zamanı*, ama daha iyi *yanıt süresi*

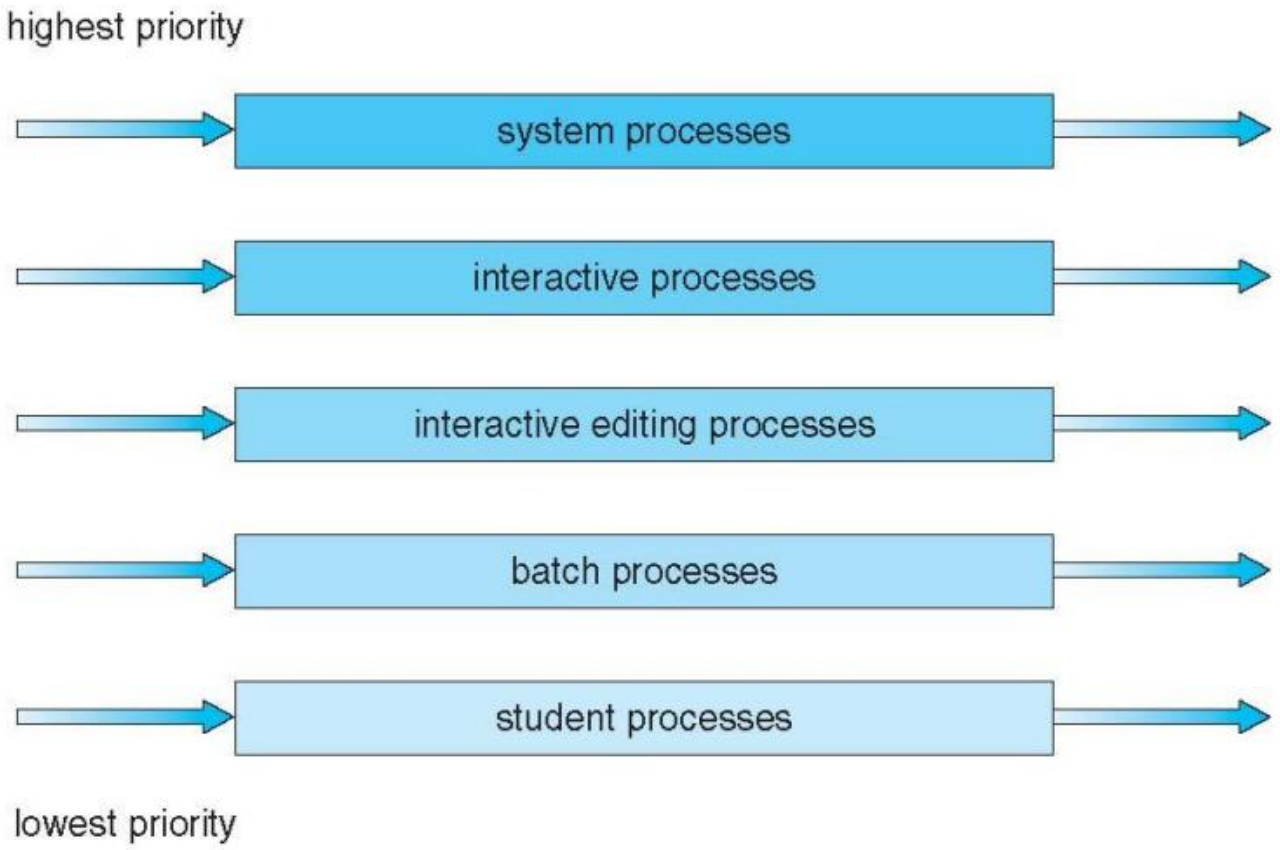
Çok Kuyruklu-İşlemci Zamanlama Algoritması

- Multilevel Queue
- Hazır kuyruğu (Ready) birden fazla kuyruğa bölünür:
  - ön plan (foreground – interaktif (interactive)
  - arka plan (background) – toplu iş (batch)
- Her kuyruk kendine ait işlemci zamanlama algoritmasına sahiptir
  - ön plan – RR
  - arka plan – FCFS

Çok Kuyruklu

- Zamanlama kuyruklar arasında yapılmalıdır
  - Sabit öncelikli zamanlama (fixed priority scheduling)
  - Örn: önce tüm ön plan işlerini çalıştırıp ardından arka plan işlerini çalıştırmak. Olası açlık (starvation)
  - Zaman dilimi – her kuyruk CPU'nun belirli bir zamanını kendi işlemlerine verebilir
  - Örn: CPU'nun %80 zamanı RR ile ön plan işlerine %20'si ise FCFS ile arka plan işlerine ayrılabilir

Çok Kuyruklu Zamanlama

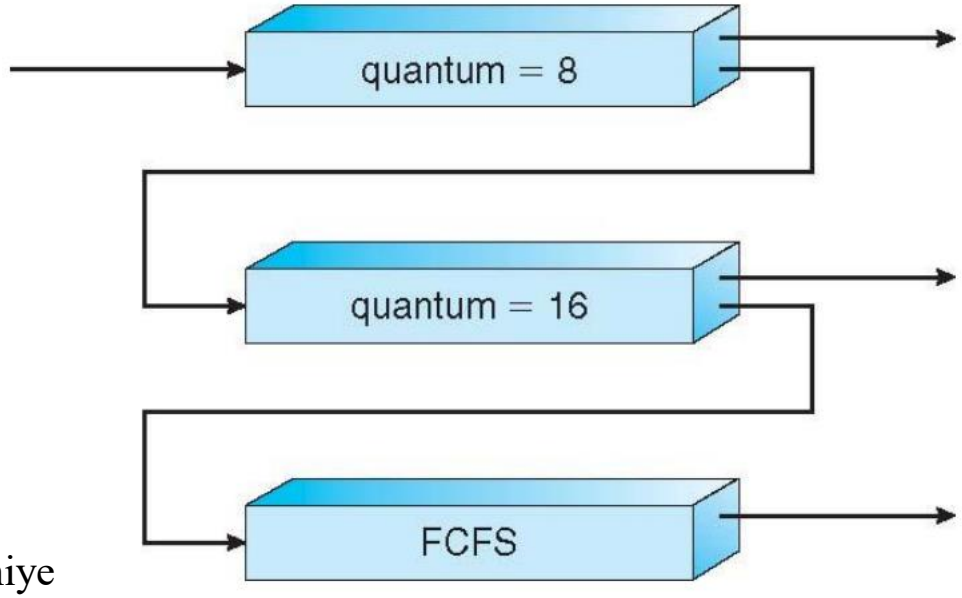


Çok – Seviye Geri Besleme Kuyruğu

- Multilevel Feedback Queue
- İşlemler kuyruklar arasında yer değiştirebilir; yaşlanma (aging) bu şekilde gerçekleştirilebilir
- Çok-seviye geri besleme kuyruğu zamanlayıcısı aşağıdaki parametrelerle tanımlanır:
  - Kuyrukların sayısı
  - Her bir kuyruk için zamanlama algoritması
  - Bir işlemin üst kuyruğa ne zaman alınacağını belirleme yöntemi
  - Bir işlemin alt kuyruğa ne zaman alınacağını belirleme yöntemi
  - Bir işlem çalıştırılmak için seçildiğinde hangi kuyruğa alınacağını belirleyen yöntem

Çok – Seviye Geri Besleme Kuyruğu Örnek

- Üç kuyruk:
  - Q0 – zaman kuantumu 8 milisaniye olan RR
  - Q1 – zaman kuantumu 16 milisaniye olan RR
  - Q2 – FCFS
- Zamanlama
  - Yeni işler Q0 kuyruğuna eklenir ve FCFS ile yönetilir
  - CPU'yu elde ettiğinde bu işe 8 milisaniye verilir. Eğer 8 milisaniyede sonlanmazsa Q1 kuyruğuna alınır
  - İş Q1 kuyruğunda yeniden zamanlanır. Sıra ona geldiğinde 16 milisaniye ek süre verilir. Hala sonlanmazsa, çalışması kesilir ve Q2 kuyruğuna alınır ve burada FCFS yöntemiyle zamanlanır



Algoritma Değerlendirmesi

- Belli bir sistem için hangi işlemci zamanlama algoritmasını seçmeliyiz?
- Pek çok algoritma ve bu algoritmaların pek çok parametresi var
- Öncelikle değerlendirme kriteri belirlenmeli – işlemci kullanımı, bekleme zamanı …
  - Örnek: İşlemci kullanımını maksimize ederken, cevap zamanını bir saniyenin altında tutmak
- Deterministik modelleme (deterministic modeling) – ön tanımlı bir iş yükünü alıp her bir algoritmanın performansını bu iş yükü açısından değerlendirmek
- Kuyruklama Modelleri (queueing models)

Kuyruklama Modelleri

- Pek çok sistemde işlem karakteristikleri günden güne değişiklik gösterir
- Ancak CPU ve I/O işlem sıklıklarının istatistiksel dağılımları çıkarılabilir
- Bu işlem sıklıkları gerçek sistemlerde ölçülebilir ve dağılımları belirlenebilir
- Benzer şekilde, işlemlerin sisteme giriş zamanlarının dağılımını da belirleyebiliriz
- Bu iki dağılımı kullanarak pek çok algoritma için ortalama olarak
  - işlemci kullanımı,
  - üretilen iş,
  - bekleme zamanı hesaplanabilir
- Dezavantaj: Matematiksel olarak analiz edilebilir olan zamanlama algoritması çok sınırlı

Simülasyonlar

- Simülasyonları çalıştırmak için bilgisayar sisteminin bir modelinin programlanması gerekir
- Yazılımdaki veri tipleri sistemin temel bileşenlerini gerçekleştirir
- Simülatör, sistem saati fonksiyonunu gören bir değişkene sahiptir
- Bu değişken her arttırıldığında, cihazların, işlemlerin ve zamanlayıcının aktivitelerine bağlı olarak sistem durumu güncellenir
- Simülasyonun çalışması için gerekli olan veri, aşağıdaki gibi dağılımları kullanan bir rastgele sayı üreticisi ile üretilir
  - Yeni işlem üretim zamanı
  - İşlemin CPU ve I/O kullanım süreleri
- Dağılımlar belli istatistiksel dağılımlara (normal veya poisson dağılımı gibi) uygun şekilde veya deneysel olarak tanımlanabilir

BİTTİ