

Temel kavramlar

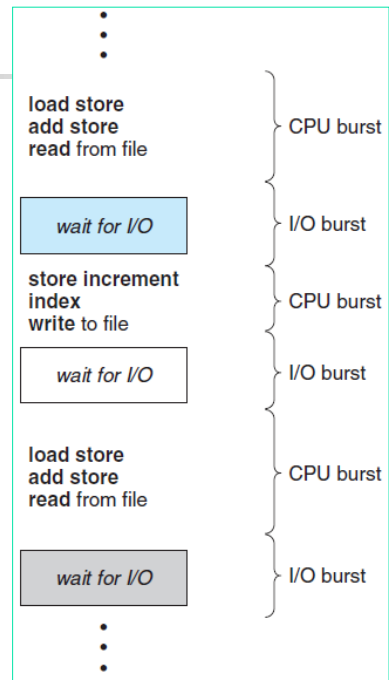
- **CPU scheduling (planlama)**, multiprogramming çalışan işletim sistemlerinin temelini oluşturur.
- CPU, process'ler arasında geçiş yaparak bilgisayarı daha verimli hale getirir.
- **Her zaman aralığında bir process'in çalıştırılması amaçlanır.**
- Tek işlemcili sistemlerde, bir anda sadece bir process çalıştırılabilir.
- CPU, process'lerde ortaya çıkacak bekleme durumlarında başka process'leri çalıştırır.
- Hafızada çok sayıda process bulundurulur.
- Bir process herhangi bir şekilde beklemeye geçtiğinde CPU başka bir process'e geçiş yapar.
- Bilgisayardaki **tüm kaynaklar** kullanılmadan önce zamana göre **planlanır**.

3

Temel kavramlar

CPU Burst Cycle ve I/O Burst Cycle

- Process çalıştırma, **CPU execution** ve **I/O wait döngüsünü içermektedir.**
- Process'ler bu iki durum arasında geçiş yaparlar.
- Process'ler **çalışmaya CPU burst ile başlarlar** ve **I/O burst** ile devam ederler.

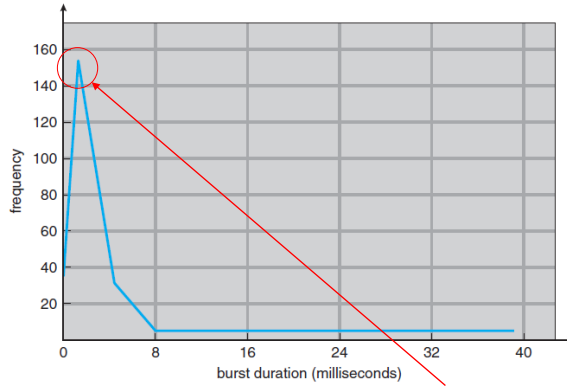


4

Temel kavramlar

CPU Burst Cycle ve I/O Burst Cycle

- **CPU burst** süresi, process'ten process'e ve bilgisayardan bilgisayara çok farklı olabilmektedir.



- **Process'ler için CPU burst süresi sıklıkla kısa olmaktadır.**
- Process'ler kısa aralıklarla durumunu değiştirmektedir.

5

Temel kavramlar

CPU Scheduler

- **CPU bekleme durumuna geçtiğinde**, işletim sistemi hazır kuyruğundan (**ready queue**) bir process'i çalıştırılmak üzere seçmek zorundadır.
- Bu seçme işlemi kısa dönem planlayıcı (**short-term scheduler** veya **CPU scheduler**) tarafından gerçekleştirilir.
- Hazır kuyruğu, ilk gelen ilk çıkar (**first-in-first-out, FIFO**) olmak zorunda değildir.
- Hazır kuyruğu, **FIFO, priority queue, ağaç, sırasız bağlı liste** şeklinde oluşturulabilir.
- Hazır kuyruğunda bekleyen tüm process'lerin CPU tarafından çalıştırılmak üzere seçilme olasılıkları vardır.
- Kuyruk içindeki kayıtlarda, **process control block (PCB)** tutulur.

6

Temel kavramlar

Preemptive Scheduling

- CPU-scheduling kararı 4 durum altında gerçekleştirilir:
 1. Bir process **çalışma** durumundan **bekleme** durumuna geçtiğinde (I/O isteği),
 2. Bir process **çalışma** durumundan **hazır** durumuna geçtiğinde (interrupt),
 3. Bir process **bekleme** durumundan **hazır** durumuna geçtiğinde (I/O tamamlanması),
 4. Bir process'in **sonlandırıldığında**.
- Eğer scheduling işlemi **1. ve 4. durumlarda gerçekleşmişse**, buna **nonpreemptive** veya **cooperative** scheduling denir.
- **2. ve 3. durumlarda gerçekleşmişse preemptive** scheduling denir.

7

Temel kavramlar

Preemptive Scheduling

- **Nonpreemptive scheduling'te**, CPU bir process'e tahsis edilmişse, bu process **sonlandırılincaya kadar, CPU'yu serbest bırakıncaya kadar veya bekler durumuna geçinceye kadar tutar**.
- Windows 3.1, nonpreemptive scheduling kullanmıştır.
- Diğer tüm Windows versiyonları preemptive scheduling kullanmıştır.
- Mac OS X işletim sistemi de preemptive scheduling kullanmaktadır.
- **Preemptive scheduling veri paylaşımı yaptığında race condition** gerçekleşir.
- **Bir process kernel verisi üzerinde değişiklik yaparken** yarıda kesilerek **başka bir process'e geçilmesi** ve aynı veriye erişim yapılması **halinde çakışma** meydana gelir.

8

Temel kavramlar

Dispatcher

- CPU scheduling işlevini gerçekleştiren bileşen **dispatcher** olarak adlandırılır.
- **Dispatcher, short-term scheduler tarafından CPU'ya atanacak process'i seçer.**
- Dispatcher aşağıdaki işlevleri içermektedir:
 - Context geçişi
 - Kullanıcı moduna geçiş
 - Programı yeniden başlatmak için kullanıcı programında uygun konuma atlama
- Dispatcher'ın çok hızlı bir şekilde geçiş yapması zorunludur.
- Process'ler arasında **geçiş süresine dispatch latency** denilmektedir.

9

Konular

- Temel kavramlar
- **Scheduling kriterleri**
- Scheduling algoritmaları
- Çoklu process veya scheduling
- Gerçek zamanlı CPU scheduling

10

Scheduling kriterleri

- CPU scheduling algoritmaları çok sayıda farklı kriterre göre karşılaştırılır:
 - **CPU utilization:** CPU'nun olabildiği kadar kullanımda olması istenir. CPU kullanım oranı %0 - %100 arasındadır. Gerçek sistemlerde bu oran %40 ile %90 arasındadır.
 - **Throughput:** Her zaman aralığında tamamlanan process sayısıdır.
 - **Turnaround time:** Bir process'in hafızaya alınmak için bekleme süresi, hazır kuyruğunda bekleme süresi, CPU'da çalıştırılması ve I/O işlemi yapması için geçen sürelerin toplamıdır.
 - **Waiting time:** Bir process'in hazır kuyruğunda beklediği süredir.
 - **Response time:** Bir process'e gönderilen isteğe cevap dönünceye kadar geçen süredir.
- **CPU utilization'ı ve throughput'u maksimum, turnaround time, waiting time ve response time'ı minimum** yapmak amaçlanır.
- Genellikle ortalama değerler optimize edilmeye çalışılır.

11

Konular

- Temel kavramlar
- Scheduling kriterleri
- **Scheduling algoritmaları**
- Çoklu process veya scheduling
- Gerçek zamanlı CPU scheduling

12

Scheduling algoritmaları

- CPU scheduling algoritmaları, **hazır kuyruğunda bekleyen process'lerden hangisinin CPU'ya atanacağını belirlerler.**
 - First-Come, First-Served Scheduling
 - Shortest-Job-First Scheduling
 - Priority Scheduling
 - Round-Robin Scheduling
 - Multilevel Queue Scheduling
 - Multilevel Feedback Queue Scheduling

13

Scheduling algoritmaları

First-Come, First-Served Scheduling

- En basit CPU scheduling algoritmasıdır ve **first-come first served (FCFS)** şeklinde çalışır.
- CPU'ya ilk istek yapan process, CPU'ya ilk atanan process'tir.
- **FIFO kuyruk yapısıyla yönetilebilir.**
- FCFS algoritmasıyla **ortalama bekleme süresi** genellikle **yüksektir.**
- **Bekleme süreleri** process'lerin **kuyruğa geliş sırasına göre çok değişmektedir.**

14

Scheduling algoritmaları

First-Come, First-Served Scheduling

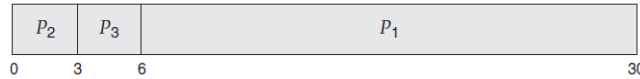
- Aşağıdaki 3 process için CPU'da çalışma süreleri ms olarak verilmiştir.

Process	Burst Time
P ₁	24
P ₂	3
P ₃	3

- Process'ler **P1, P2, P3** sırasıyla gelirse **Gantt** şeması aşağıdaki gibidir.



- Ortalama bekleme süresi $(0 + 24 + 27) / 3 = 17$ ms olur.
- P2, P1, P3** sırasıyla gelirse **Gantt** şeması aşağıdaki gibidir.



- Ortalama bekleme süresi $(0 + 3 + 6) / 3 = 3$ ms olur.

15

Scheduling algoritmaları

First-Come, First-Served Scheduling

- FCFS algoritmasında, **process'lerin çalışma süreleri çok farklıysa** ortalama **bekleme süreleri çok değişken** olur.
- Bir CPU-bound process ile çok sayıda I/O bound process varsa**, CPU-bound process CPU'da çalışırken **tüm I/O bound process'ler hazır kuyruğunda bekler**, I/O cihazları boş kalır.
- Çok sayıda küçük process'in **büyük bir process'in CPU'yu terketmesini beklemesine** **convoy effect** denilmektedir.
- Bir process'e CPU tahsis edildiğinde sonlanana veya I/O isteği yapana kadar CPU'yu elinde tutar.**
- FCFS algoritması belirli zaman aralıklarıyla **CPU'yu paylaşan time-sharing sistemler için uygun değildir.**

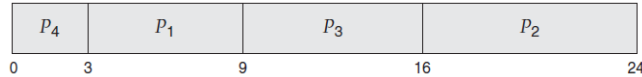
16

Scheduling algoritmaları

Shortest-Job-First Scheduling

- **Shortest-Job-First Scheduling (SJF)** algoritmasında, CPU'ya bir sonraki işlem süresi en kısa olan (shortest-next-CPU-burst) process atanır.

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



- Ortalama bekleme süresi, $(0 + 3 + 9 + 16) / 4 = 7$ ms'dir. FCFS kullanılsaydı 10,25 ms olurdu $((0 + 6 + 14 + 21) / 4)$.
- SJF algoritması minimum ortalama bekleme süresini elde eder.

17

Scheduling algoritmaları

Shortest-Job-First Scheduling

- SJF algoritmasındaki en büyük zorluk, sonraki çalışma süresini tahmin etmektir.
- Long-term (job) scheduling için kullanıcının belirlediği süre alınabilir.
- SJF algoritması genellikle long-term scheduling için kullanılır.
- SJF algoritması short-term scheduling seviyesinde kullanılamaz.
- Short-term scheduling'te CPU'da sonraki çalışma süresini bilmek mümkün değildir.
- Short-term scheduling'te sonraki çalışma süresi tahmin edilmeye çalışılır.
- Sonraki çalışma süresinin önceki çalışma süresine benzer olacağı beklenir.

18

Scheduling algoritmaları

Shortest-Job-First Scheduling

- Sonraki CPU burst süresi, **önceki** CPU burst sürelerinin **üstel ortalama (exponential average) değeri olarak tahmin edilir.**

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

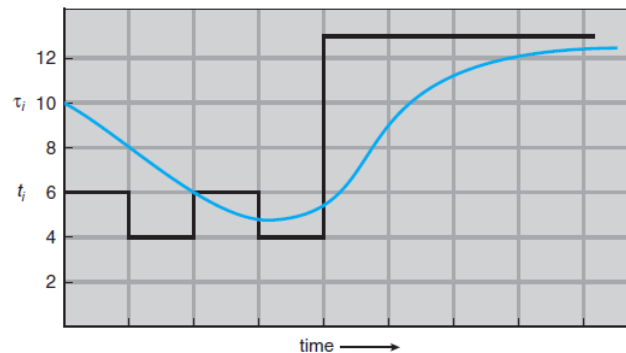
- Burada, $0 < \alpha < 1$ (genellikle $1/2$), τ_{n+1} sonraki tahmin edilen süreyi, t_n ise n . CPU burst süresini gösterir.
- SJF algoritması preemptive veya nonpreemptive olabilir.
- **Çalışmakta olan process'ten daha kısa süreye sahip yeni bir process geldiğinde, preemptive SJF çalışmakta olanı keser, nonpreemptive SJF çalışmakta olanın sonlanmasına izin verir.**
- Preemptive SJF, **shortest-remaining-time-first scheduling** olarak adlandırılır.

19

Scheduling algoritmaları

Shortest-Job-First Scheduling

- $\alpha = 1/2$ ve $\tau_0 = 10$ için exponential average görülmektedir.



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

20

Scheduling algoritmaları

Priority Scheduling

- Shortest-job-first (SJF) algoritması, **priority scheduling** algoritmalarının özel bir durumudur.
- **CPU en yüksek önceliğe sahip process'e atanır.**
- Eşit önceliğe sahip olanlar ise FCFS sırasıyla atanır.
- **SJF algoritması tahmin edilen CPU-burst süresine göre önceliklendirme yapar.**
- SJF algoritmasında, CPU burst süresi azaldıkça öncelik artar, CPU burst süresi arttıkça öncelik azalır.

21

Scheduling algoritmaları

Priority Scheduling

- Aşağıda 5 process için öncelik değerine göre gantt şeması verilmiştir.

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2



- Ortalama bekleme süresi $(1 + 6 + 16 + 18) / 4 = 8,2$ ms olur.

22

Scheduling algoritmaları

Priority Scheduling

- Önceliklendirme kriterleri aşağıdakilerden bir veya birkaç tanesi olabilir:
 - Zaman sınırı
 - Hafıza gereksinimi
 - Açılan dosya sayısı
 - I/O burst ve CPU burst oranı
 - Process'in önemi
- Priority scheduling preemptive veya nonpreemptive olabilir.
- **Preemptive yönteminde**, bir process hazır kuyruğuna geldiğinde, çalışmakta olan process'ten daha öncelikli ise, **çalışmakta olan kesilir**.
- **Nonpreemptive yönteminde**, bir process hazır kuyruğuna geldiğinde, çalışmakta olan process'ten daha öncelikli bile olsa, **çalışmakta olan durum değiştirene kadar devam eder**.

23

Scheduling algoritmaları

Priority Scheduling

- Priority scheduling algoritmasında, **CPU sürekli yüksek öncelikli process'leri çalıştırabilir** ve bazı processler sürekli hazır kuyruğunda bekleyebilir (*indefinite blocking, starvation*).
- **Sınırsız beklemeyi engellemek için düşük öncelikli process'ler kuyrukta beklerken öncelik seviyesi artırılır** (Örn. her 15 dakikada 1 artırılır).
- Öncelik değeri artırılarak **en düşük önceliğe sahip process'in** bile belirli bir süre sonunda çalışması sağlanır.

24

Scheduling algoritmaları

Round-Robin Scheduling

- Round-robin (RR) scheduling, genellikle time-sharing sistemlerde kullanılır.
- Hazır kuyruğundaki process'ler belirli bir zaman aralığında (time slice) CPU'ya sıralı atanır.
- Zaman aralığı genellikle 10 ms ile 100 ms aralığında seçilir.
- Time slice aralığından daha kısa sürede sonlanan process CPU'yu serbest bırakır.
- Round-robin scheduling ile ortalama bekleme süresi genellikle uzundur.

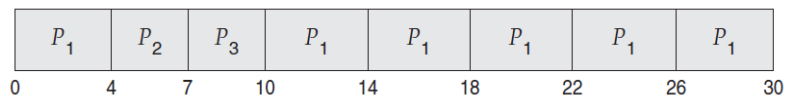
25

Scheduling algoritmaları

Round-Robin Scheduling

- Aşağıda 3 process için CPU-burst time ve gantt şeması verilmiştir.
- Örnekte time slice = 4 ms olarak alınmıştır.

Process	Burst Time
P_1	24
P_2	3
P_3	3



- P_1 için $10 - 4 = 6$, P_2 için 4, P_3 için 7 ms bekleme süresi vardır.
- Ortalama bekleme süresi ise $17 / 3 = 5,66$ ms'dir.
- q time slice süresiyle n process çalışan sistemde, bir process için en fazla bekleme süresi $(n - 1) * q$ olur.

26

Scheduling algoritmaları

Round-Robin Scheduling

- Time slice süresi **çok büyük olursa** çalışma **FCFS yöntemine benzer**.
- Time slice süresi **çok küçük olursa context switch işlemi çok fazla yapılır**.
- Context switch süresi overhead olur ve çok fazla context switch yapılması istenmez.
- Time slice süresi, context switch süresinin genellikle **10 katı** alınır.
- CPU'nun %10 süresi context switch için harcanır.

27

Scheduling algoritmaları

Multilevel Queue Scheduling

- **Multilevel Queue Scheduling (MQS)** algoritmasında, process'ler farklı gruplar halinde sınıflandırılır.
- Örneğin process'ler **foreground** (interaktif) ve **background** (batch) olarak 2 gruba ayrılabilir.
- Foreground process'lerde response-time kısa olması gereklidir ve background process'lere göre önceliklidir.
- **Multilevel queue scheduling** algoritması **hazır kuyruğunu parçalara böler** ve kendi aralarında önceliklendirir.
- Process'ler bazı özelliklerine göre (hafıza boyutu, öncelik, process türü, ...) bir kuyruğa atanır.
- Her kuyruk kendi scheduling algoritmasına sahiptir.

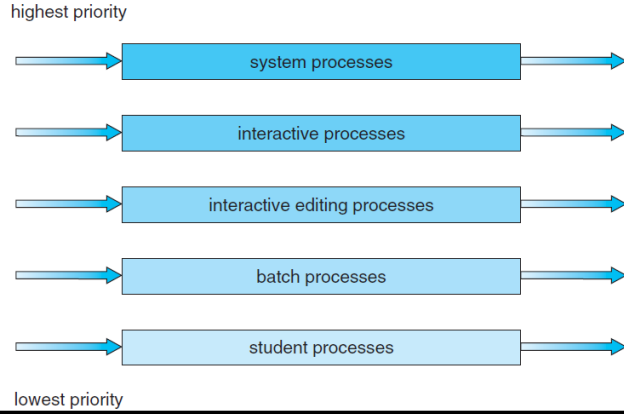
28

Scheduling algoritmaları

Multilevel Queue Scheduling

- Her kuyruğa öncelik derecesine göre time slice atanabilir.
- Her kuyruğa diğerlerine göre **mutlak öncelik tanımlanabilir ve kendisinde process varken düşük öncelikli kuyruğa geçilmez.**

1. System processes
2. Interactive processes
3. Interactive editing processes
4. Batch processes
5. Student processes



Scheduling algoritmaları

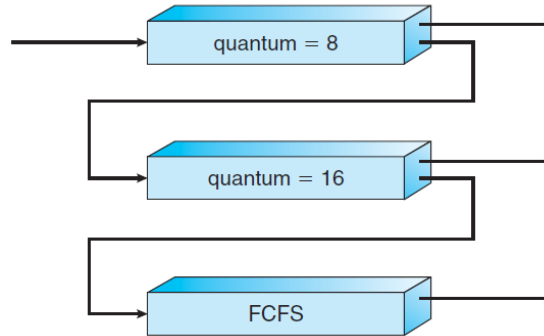
Multilevel Feedback Queue Scheduling

- **Multilevel Queue Scheduling (MQS)** algoritmasında, process'ler farklı kuyruklar arasında geçiş yapabilirler.
- Bu yöntemde, I/O bound ve interaktif process'ler yüksek öncelikli kuyruğa atanır.
- **Düşük öncelikli kuyrukta çok uzun süre bekleyen process'ler yüksek öncelikli kuyruğa aktarılır** (indefinite lock engellenir).
- Hazır kuyruğuna gelen process öncelikle en yüksek öncelikli kuyruğa alınır.
- En yüksek öncelikli kuyruk tamamen boşalırsa ikinci öncelikli kuyruğa geçilir.

Scheduling algoritmaları

Multilevel Feedback Queue Scheduling

- Şekilde üstteki kuyruk en yüksek önceliğe, en alttaki kuyruk en düşük önceliğe sahiptir.
- Kuyruklarda time slice (quantum) süreleri farklı olabilir.



31

Konular

- Temel kavramlar
- Scheduling kriterleri
- Scheduling algoritmaları
- Multi-processor scheduling
- Gerçek zamanlı CPU scheduling

32

Multi-processor scheduling

- Çok işlemci kullanılan sistemlerde yük paylaşımı (load sharing) yapılabilir, ancak scheduling çok daha karmaşık hale gelir.
- Asymmetric multiprocessing yaklaşımında, CPU'lerden birisi (master) scheduling algoritmaları, I/O işlemleri ve diğer sistem aktivitelerini yönetir. Diğer CPU'lar kullanıcı kodlarını çalıştırır.
- Symmetric multiprocessing yaklaşımında, her CPU kendi scheduling algoritmasına sahiptir ve master CPU yoktur.
- Tüm CPU'lar ortak hazır kuyruğuna sahip olabilir veya ayrı ayrı hazır kuyruğu olabilir.
- Birden fazla CPU'nun paylaşılan veri yapısına erişimi engellenmelidir.
- Birden fazla CPU'nun aynı process'i çalıştırması engellenmelidir.
- Windows, Linux ve Mac OS X işletim sistemleri SMP desteğini sağlarlar.

33

Multi-processor scheduling

İşlemci ile atanan process ilişkisi

- Bir process başka bir işlemciye aktarıldığında, eski işlemcideki cache bellek bilgileri aktarılmaz.
- Yeni aktarılan işlemcinin cache bellek bilgileri oluşana kadar hit rate oranı çok düşük kalır.
- Bir process çalışmakta olduğu işlemci ile ilişkilendirilir (processor affinity) ve sonraki çalışacağı işlemci de aynı olur.
- Bazı sistemlerde, process bir işlemciye atanır, ancak aynı işlemcide çalışmayı garanti etmez (soft affinity).
- Bazı sistemlerde, process bir işlemciye atanır ve her zaman aynı işlemcide çalışmayı garanti eder (hard affinity).
- Linux işletim sistemi soft affinity ve hard affinity desteğine sahiptir.

34

Multi-processor scheduling

Yük dengeleme

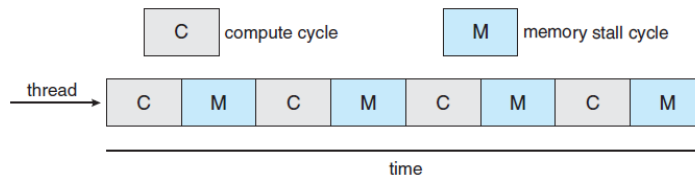
- **Yük dengeleme (load balancing)**, SMP sistemlerde tüm işlemciler üzerinde iş yükünü dağıtarak verimi artırmayı amaçlar.
- Her işlemcinin kendi kuyruğına sahip olduğu sistemlerde, yük dengeleme iyi yapılmazsa bazı işlemciler boş beklerken diğer işlemciler yoğun çalışabilir.
- Ortak kuyruk kullanan sistemlerde yük dengelemeye ihtiyaç olmaz.
- **Yük dağılımı için iki yöntem kullanılır: push migration ve pull migration.**
- **Push migration** yönteminde, **bir görev** işlemcilerin iş yükünü kontrol eder ve **boş olanlara dolu olan diğer işlemcilerdeki process'leri aktarır.**
- **Pull migration** yönteminde, boş kalan işlemci dolu olan diğer işlemcilerde **bekleyen bir process'i kendi üzerine alır.**

35

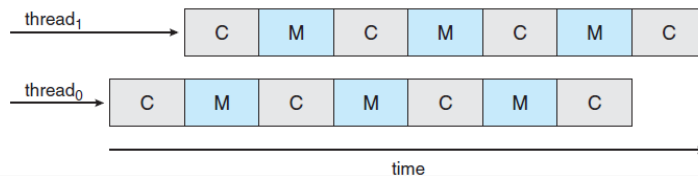
Multi-processor scheduling

Multicore sistemler


- İşletim sistemi için her core ayrı bir işlemci olarak görülür.
- İşlemcinin hafıza erişimi uzun süre alır (**memory stall**).



- Şekildeki işlemci, %50 süreyi hafızayı beklerken geçirmektedir.
- Bir core'a birden fazla thread atanarak aralarında geçiş yapılır.




36



Konular

- Temel kavramlar
- Scheduling kriterleri
- Scheduling algoritmaları
- Multi-processor scheduling
- Gerçek zamanlı CPU scheduling

37



Gerçek zamanlı CPU scheduling

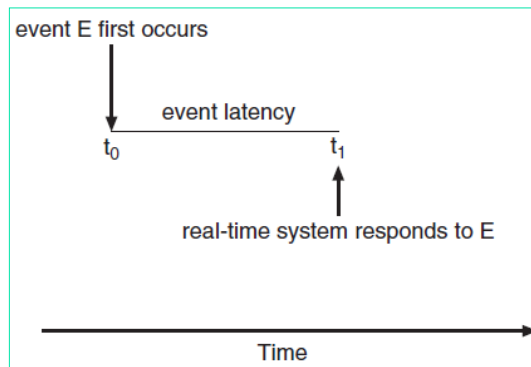
- Gerçek zamanlı sistemler iki gruba ayrılır:
 - Soft real-time sistemler
 - Hard real-time sistemler
- Soft real-time sistemler, zaman kritik process'lere diğerlerine göre öncelik verir, ancak **çalışma süresine garanti vermez.**
- Hard real-time sistemler, zaman kritik process'leri **deadline süresinde çalıştırmayı garanti eder.**

38

Gerçek zamanlı CPU scheduling

Minimizing latency

- Sistemde bir olay gerçekleştiğinde, olabildiği kadar kısa sürede gerekli işlemin yapılması zorunludur.
- Bir olay oluştuktan sonra işlemin gerçekleşmesine kadar bir süre geçer (event latency).

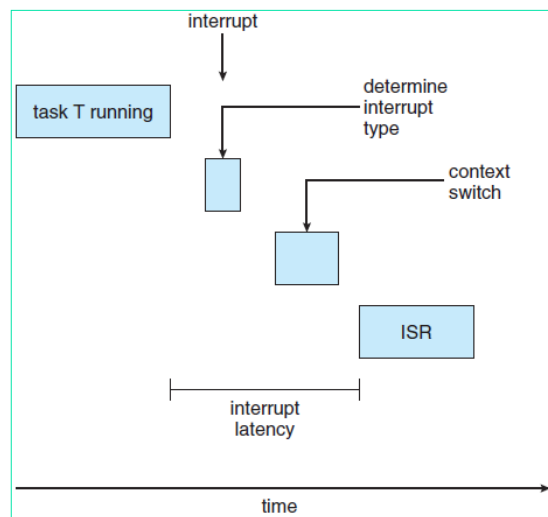


39

Gerçek zamanlı CPU scheduling

Minimizing latency

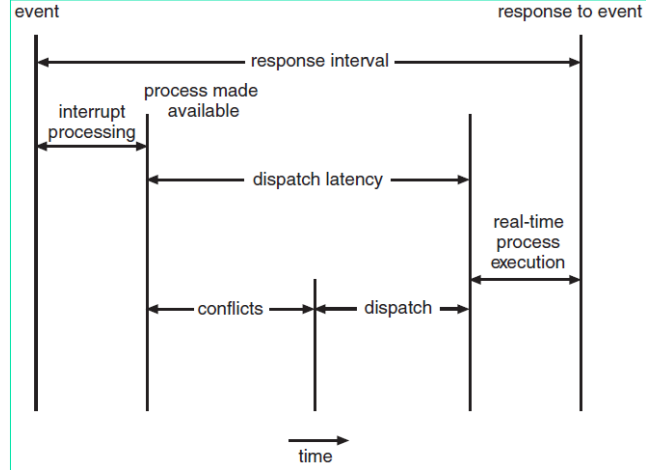
- Gerçek zamanlı sistemlerin performansını interrupt latency ve dispatch latency etkiler.
- Interrupt latency, CPU'ya interrupt gelmesi ile CPU'nun istenen işleme başlaması için geçen süredir.



Gerçek zamanlı CPU scheduling

Minimizing latency

- Bir process'in durdurularak diğer process'in başlatılması için geçen süreye **dispatch latency** denir.
- **Conflict** aşamasında yeni process için kaynak aktarımı veya bir process'in durdurulması gerçekleştirilir.



Gerçek zamanlı CPU scheduling

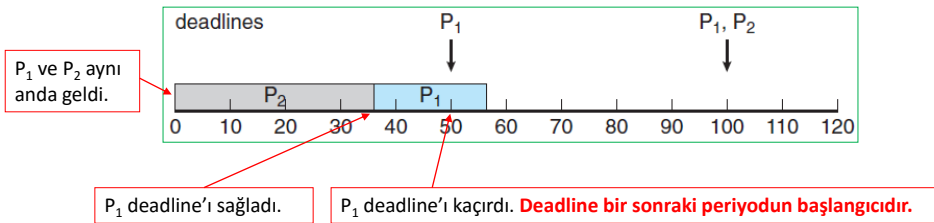
Priority-Based Scheduling

- Gerçek zamanlı işletim sistemleri **öncelik tabanlı algoritma** kullanmak zorundadır.
- Bir process CPU'da çalışırken yüksek öncelikli process geldiğinde kesilir ve gelen process çalıştırılır.
- Windows 32 seviyeli önceliklendirme yapar. **16-31 arasındaki öncelik değerlerini gerçek zamanlı process'lere ayırmıştır.**
- Bu şekildeki çalışma ile (preemptive, priority-based scheduler) soft real-time garanti edilir.
- Bir process için **deadline'dan önce çalışma garantisi yoktur.**
- Aynı öncelik seviyesinde bekleyen process'ler varsa preemptive yapılmaz.

Gerçek zamanlı CPU scheduling

Rate-Monotonic Scheduling

- **Rate-Monotonic Scheduling** algoritmasında, her process sisteme geldiğinde **periyot süresiyle ters orantılı** (sisteme gelme sıklığı ile doğru orantılı) **öncelik seviyesi atanır**.
- **Periyot süresi kısaltıkça** öncelik seviyesi artar, **artıkça** öncelik seviyesi azalır. **CPU'yu kullanma sıklığı artan process'lere öncelik verilir**.
- **P_2 daha önceliklidir** (CPU'ya gelme sıklığı göz önüne alınmamıştır.).
- **$p_1 = 50$** (periyot süresi), **$t_1 = 20$** (çalışma süresi), **$p_2 = 100$** , **$t_2 = 35$** .

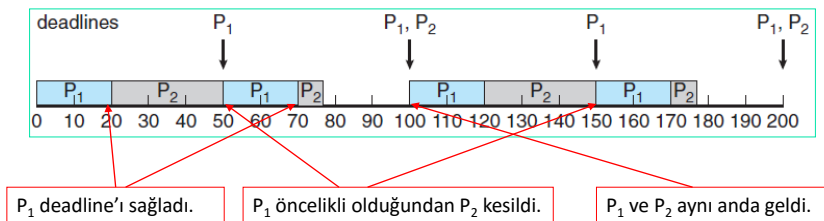


43

Gerçek zamanlı CPU scheduling

Rate-Monotonic Scheduling

- Şekilde **P_1 daha önceliklidir** (CPU'ya gelme sıklığı fazladır.).
- **$p_1 = 50$** (periyot süresi), **$t_1 = 20$** (çalışma süresi), **$p_2 = 100$** , **$t_2 = 35$** .

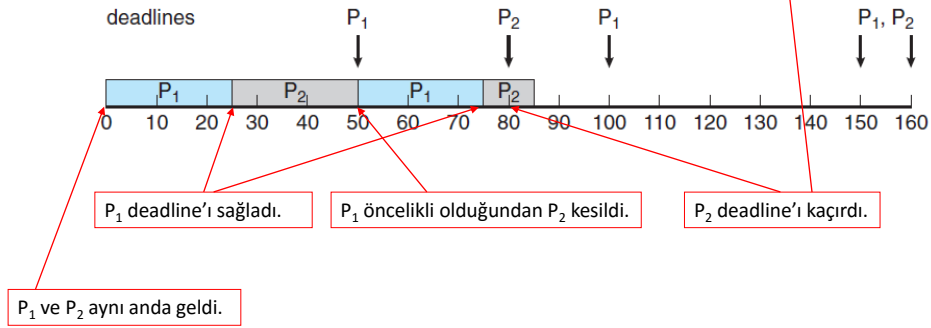


44

Gerçek zamanlı CPU scheduling

Rate-Monotonic Scheduling

- Şekilde P_1 daha önceliklidir (CPU'ya gelme sıklığı fazladır.).
- $p_1 = 50$ (periyot süresi), $t_1 = 25$ (çalışma süresi), $p_2 = 80$, $t_2 = 35$.

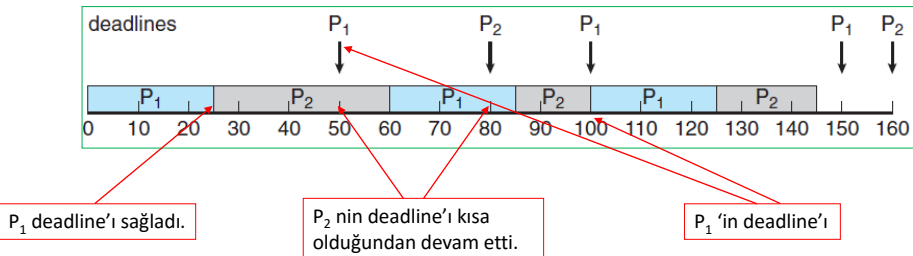


45

Gerçek zamanlı CPU scheduling

Earliest-Deadline-First Scheduling

- Earliest-Deadline-First Scheduling** algoritmasında, her process'e deadline'a göre öncelik seviyesi atanır.
- Deadline'ı kısa olanın öncelik seviyesi yüksektir.
- Şekilde P_1 daha önceliklidir (CPU'ya gelme sıklığı fazladır.).
- $p_1 = 50$ (periyot süresi), $t_1 = 25$ (çalışma süresi), $p_2 = 80$, $t_2 = 35$.



46