Bölüm 5: Çizelgeleme

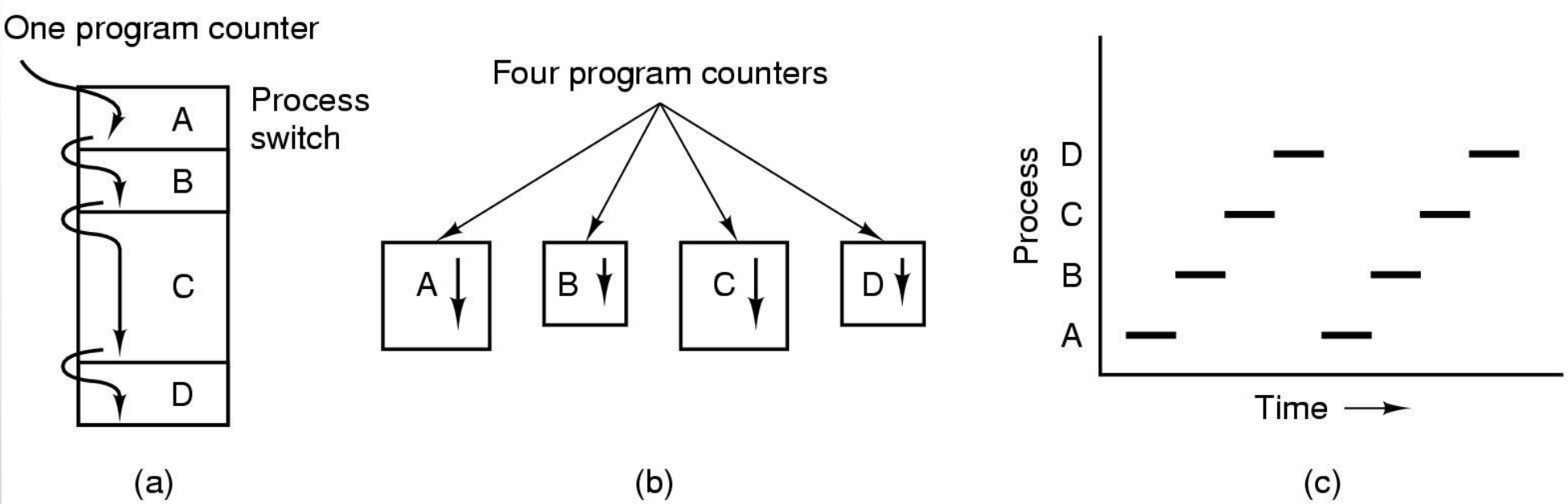
İşletim Sistemleri

Sözde Paralellik

- Tüm modern bilgisayarlar aynı anda birçok iş yapar.
- Tek işlemcili bir sistemde, herhangi bir anda, işlemci sadece bir işlem yürütebilir.
- Ancak çoklu programlama sisteminde işlemci, her biri onlarca veya yüzlerce ms boyunca çalışan işlemler arasında hızlıca geçiş yapar.
- Sözde paralellik kullanıcılar için çok faydalıdır. Ancak; yönetimi bir o kadar zordur.

Çoklu Programlama Süreç Modeli

(a) Dört programın çoklu programlanması. (b) Birbirinden bağımsız dört ardışık sürecin kavramsal modeli. (c) Aynı anda bir program etkindir.



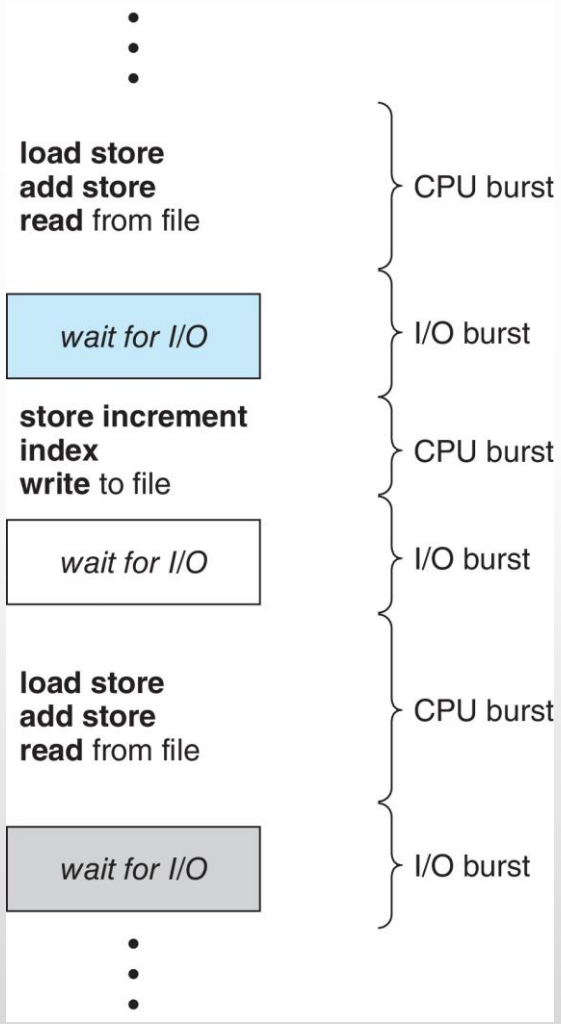
Çizelgeleme

- Bir sonraki adımda hangi süreç çalıştırılacak?
- İşlemci bir süreci sonlanana kadar çalıştırmalı mı, yoksa farklı süreçler arasında geçiş yapmalı mı?

- Süreç değiştirme pahalı
  - Kullanıcı modu ve çekirdek modu arasında geçiş
  - Geçerli süreç kaydedilir
  - Bellek haritası (memory map) kaydedilir
  - Önbellek temizlenir ve yeniden yüklenir

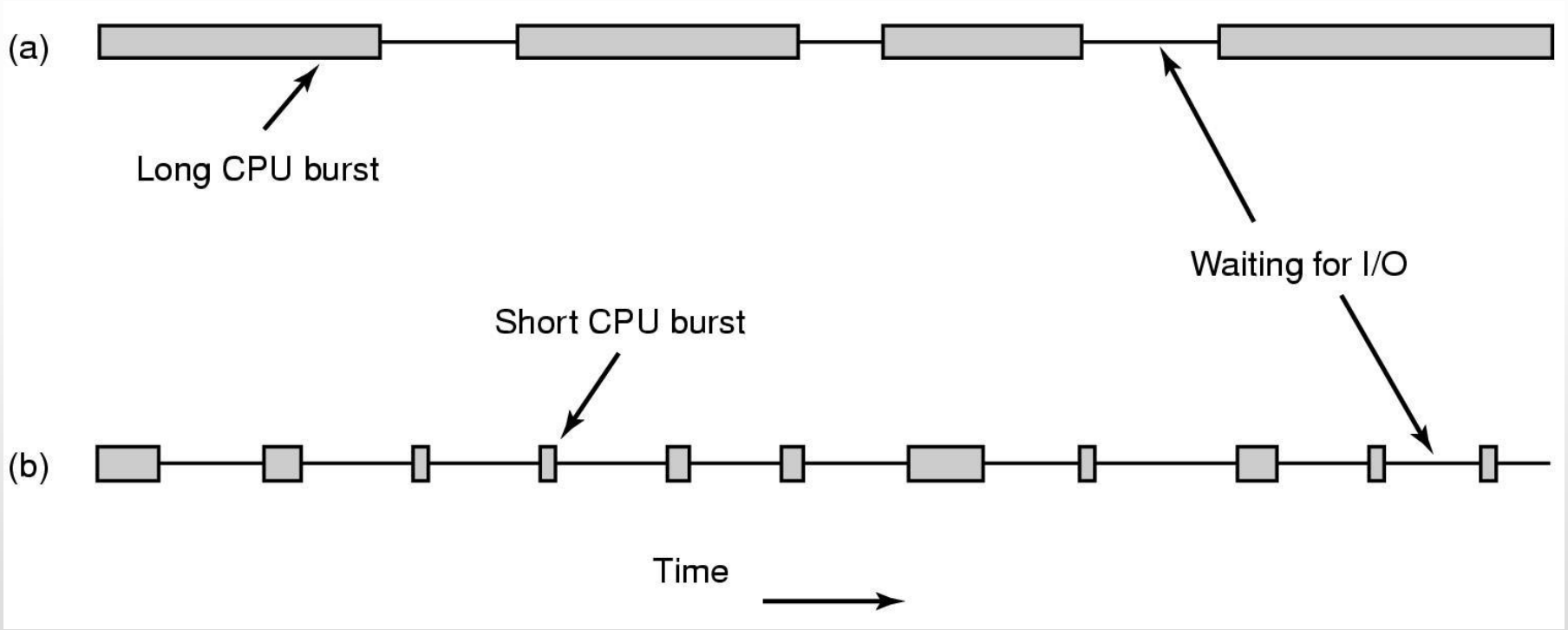
Çizelgeleme

- Çoklu programlama ile işlemci maksimum düzeyde kullanılır
- CPU yürütme döngüsünü, G/Ç beklemesi takip eder
- CPU kaynak dağıtımı önemli!



İşlemci Kullanımı

(a) CPU'ya bağlı (bound) bir işlem. (b) G/Ç'ye bağlı bir işlem.



Not

- CPU hızlandığında, süreçler daha fazla G/Ç bağlı olurlar
- Bir G/Ç bağlı süreç çalışmak istediğinde, hızlı bir şekilde değişiklik alması gerekir.

İşlemci Çizelgeleyici

- Hazır kuyruğundaki süreçler arasından seçim yapar ve bunlardan birine bir işlemci çekirdeği tahsis eder.
  - Kuyruk elemanları çeşitli şekillerde sıralanmış olabilir
- Çizelgeleme kararları bir süreç,
  - Çalışır durumundan bekleme durumuna geçtiğinde,
  - Çalışır durumundan hazır durumuna geçtiğinde,
  - Bekleme durumundan hazır durumuna geçtiğinde,
  - Sonlandığında, alınır.

Kavramlar

- Önleyici (preemptive) algoritma
  - Bir süreç, zaman aralığının sonunda hala çalışıyor durumunda ise, askıya alınır ve başka bir süreç çalıştırılır.
- Önleyici olmayan (non-preemptive) algoritma
  - Çalıştırmak için bir süreç seçilir ve bloke olana kadar veya gönüllü olarak işlemciyi serbest bırakana kadar çalışmasına izin verilir.

- Windows, MacOS, Linux, ve UNIX gibi tüm modern işletim sistemleri önleyici algoritmalar kullanır.
- Preemptive algoritma süreçler arasında yarış durumuna neden olabilir.

Görev Dağıtıcısı

- Görev dağıtıcısı, çizelgeleyici tarafından seçilen sürece işlemcinin kontrolünü verir
  - Bağlam anahtarlama
  - Kullanıcı moduna geçiş
  - Kullanıcı programında uygun konuma atlama
- Görev dağıtıcısı gecikmesi: dağıtıcının bir süreci durdurup başka bir süreci başlatması için geçen süre

Kriterler

- CPU kullanımı – CPU'yu mümkün olduğunca meşgul tutun
- Verim – zaman birimi başına yürütülmesini tamamlayan süreçlerin sayısı
- Geri dönüş süresi - belirli bir işlemi yürütmek için geçen süre
- Bekleme süresi - bir işlemin hazır kuyruğunda beklediği süre
- Yanıt süresi – bir talebin gönderilmesinden ilk yanıtın üretilmesine kadar geçen süre.

Çizelgelemenin Hedefleri

- Tüm sistemler
  - Adalet - her işleme CPU'dan adil bir pay vermek
  - Politika uygulama - belirtilen politikanın yürütüldüğünü görme
  - Denge - sistemin tüm parçalarını meşgul tutmak
- Toplu sistemler
  - Verim – birim zamanda yapılan işi maksimize etmek
  - Geri dönüş süresi – başlatma ve sonlanma arası süreyi minimum
  - CPU kullanımı - CPU'yu her zaman meşgul tutmak

Çizelgelemenin Hedefleri

- Etkileşimli sistemler
  - Yanıt süresi - isteklere hızla yanıt verilmeli
  - Orantılılık - kullanıcıların beklentilerini karşılamalı
- Gerçek zamanlı sistemler
  - Son teslim zamanı (deadline) - veri kaybı olmamalı
  - Öngörülebilirlik - multimedya sistemlerinde kalite düşmemeli

Çizelgeleme Kategorileri

- Farklı ortamlar farklı çizelgeleme algoritmalarına ihtiyaç duyar
- Toplu (batch)
  - Hala yaygın olarak kullanılıyor
  - Önleyici olmayan algoritmalar süreç geçişlerini azaltır
- Etkileşimli
  - Önleyici algoritma gerekli
- Gerçek zamanlı
  - Süreçler hızlı çalışır ve bloke olur

Toplu Sistemlerde Çizelgeleme

- İlk gelen alır (first-come first-served)
  - Süreçler, kuyruğa geldikleri sırayla yürütülür. Basit ve uygulaması kolay. Uzun işlerde düşük performans, daha kısa işler için bekleme süresine neden olur.
- Önce en kısa iş (shortest job first)
  - Kısa yürütme süresine sahip süreçlere, uzun olanlara göre öncelik verilir. CPU kullanımını üst düzeye çıkarır. Süreçlerin yürütme süresini tahmin etmek zor.
- Sonraki en kısa kalan süre (shortest remaining time next)
  - En kısa kalan süreye sahip işler önce yürütülür. Dinamik yaklaşım, yürütme süresi değiştikçe kendini ayarlar. Her iş için kalan süreyi takip etme ve kuyrukta sık sık değişiklik yapma yükü var.

FCFS Çizelgeleme

Süreç	Gelme zamanı	Servis süresi	Başlama zamanı	Sonlanma zamanı	Geri dönüş süresi	Ağırlıklı geri dönüş süresi
A	0	1	0	1	1	1
B	1	100	1	101	100	1
C	2	1	101	102	100	100
D	3	100	102	202	199	1.99

FCFS Çizelgeleme

- Uzun süreçler kısa süreçlerin çok fazla beklemesine neden olur
  - Konvoy etkisi
- G/Ç bağlı süreçler az CPU kullanmasına rağmen sonlanana kadar diğer süreçlere şans vermezler
- Tahmin edilebilir (deterministic) değil

FCFS Örnek Senaryo

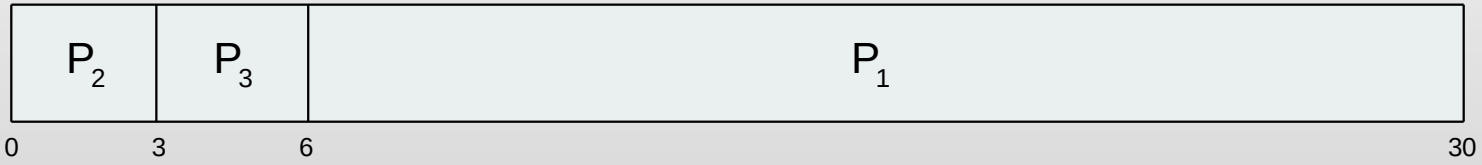
Süreç	Servis süresi
P1	24
P2	3
P3	3

- Süreçlerin çalışma sırası: **P1 -> P2 -> P3**
- Bekleme süresi: P1 = 0; P2 = 24; P3 = 27
- Ortalama bekleme süresi: (0 + 24 + 27) / 3 = 17

FCFS Örnek Senaryo

Süreç	Servis süresi
P1	24
P2	3
P3	3

- Süreçlerin çalışma sırası: **P2 -> P1 -> P3**
- Bekleme süresi: P1 = 6; P2 = 0; P3 = 3
- Ortalama bekleme süresi: (6 + 0 + 3) / 3 = 3



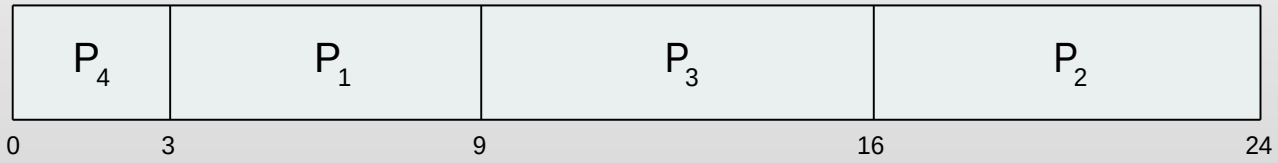
SJF Çizelgeleme

- Her süreç, servis süresi ile ilişkilendirilir
- Çizelgeleme aşamasında servis süreleri göz önüne alınır
- Optimal bir algoritmadır
  - Ortalama bekleme süresi minimumdur
- Servis süresi nasıl hesaplanacak?
  - Kullanıcıya sorulabilir
  - Tahmin

SJF Örnek Senaryo

Süreç	Servis süresi
P1	6
P2	8
P3	7
P4	3

- Bekleme süresi: P1 = 3; P2 = 16; P3 = 9; P4 = 0
- Ortalama bekleme süresi: (3 + 16 + 9 + 0) / 4 = 7



SJF Çizelgeleme

(a) Orijinal sıra.

8	4	4	4
A	B	C	D

(b) En kısa süreci birinci sırada yürütme

4	4	4	8
B	C	D	A

FCFS SJF Karşılaştırma

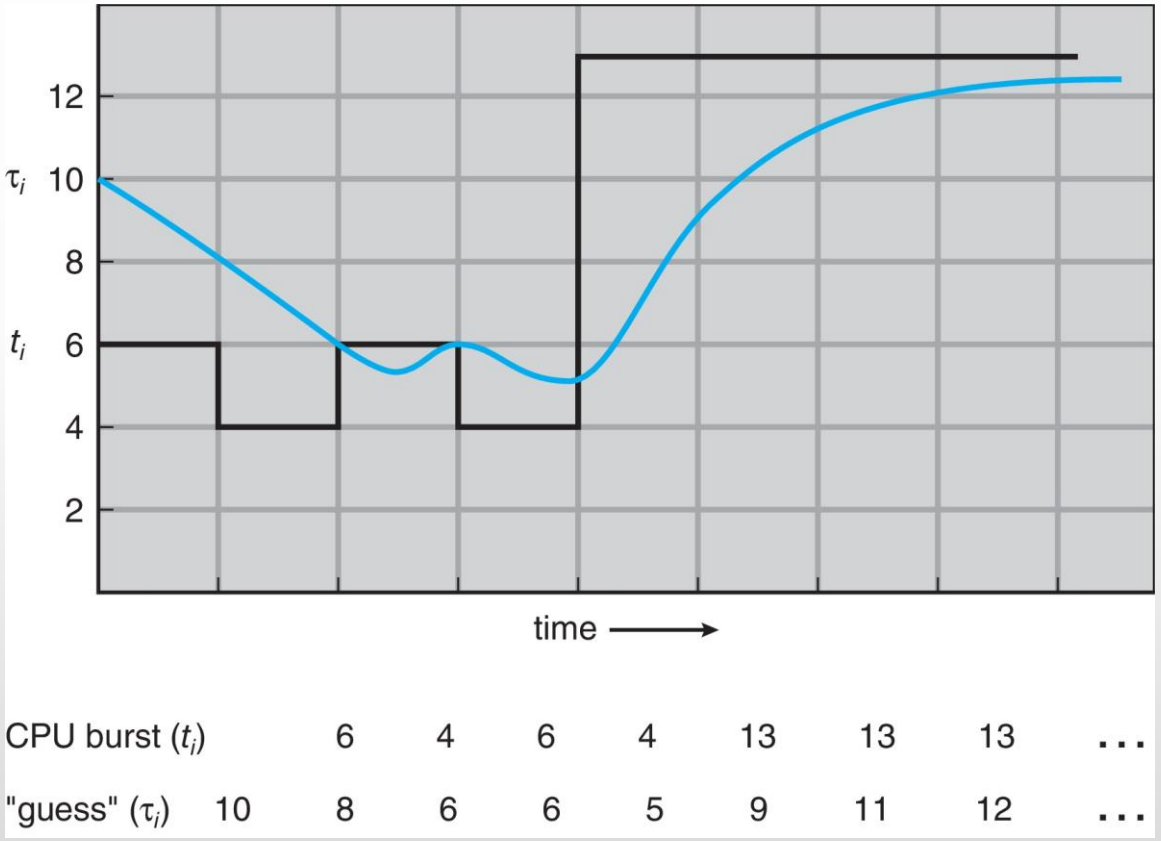
	Süreç	A	B	C	D	E	ortalama
	Geliş	0	1	2	3	4	
	Servis	4	3	5	2	4	
SJF	Bitiş	4	9	18	6	13	
	Dönüş	4	8	16	3	9	8
	Ağırlıklı	1	2.67	3.1	1.5	2.25	2.1
FCFS	Bitiş	4	7	12	14	18	
	Dönüş	4	6	10	11	14	9
	Ağırlıklı	1	2	2	5.5	3.5	2.8

SRTN Çizelgeleme

- SJF algoritmasının preemptive versiyonu
- Sıradaki yürütme için en kısa süreye sahip süreç seçilir
- Önleyici (pre-emptive): yeni sürecin çalışma süresini mevcut işin kalan süresiyle karşılaştır
  - Süreçlerin çalışma sürelerinin önceden bilinmesi gerekiyor
  - Sürecin önceki servis sürelerinden tahmin edilebilir
  - Üstel ortalama (Exponential averaging) yöntemi
    - $s_0 = x_0$
    - $s_t = ax_t + (1 - a)\, s_{t-1},\ t > 0$
    - $0 < a < 1$

Üstel Ortalama

- .

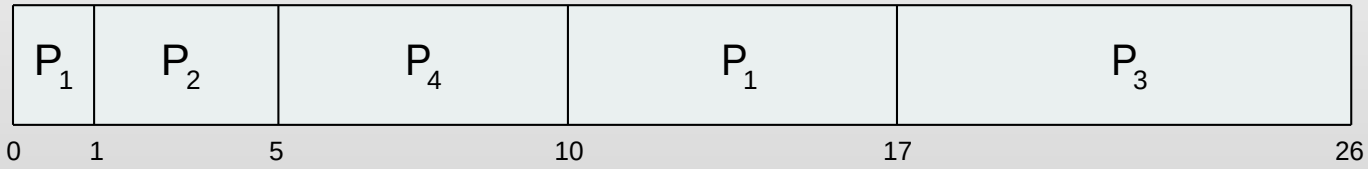


CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

SRTN Örnek Senaryo

Süreç	Geliş süresi	Servis süresi
P1	0	8
P2	1	4
P3	2	9
P4	3	5

- Başlama süresi: P1 = 10; P2 = 1; P3 = 17; P4 = 5
- Ortalama bekleme süresi: [(10-1)+(1-1)+(17-2)+(5-3)] / 4 = 6.5



Sonuç

- Her algoritmada, CPU tahsisi farklı kriterlere göre belirlenir.
- FCFS ve SJF önleyici değildir, yani bir iş başlatıldıktan sonra tamamlanıncaya kadar devam eder.
- SRTN, kalan yürütme süreleri değiştikçe işler arasında geçiş yapılmasına izin veren önleyici bir algoritmadır.
- Algoritma seçimi, işletim sisteminin ve üzerinde çalıştığı sistemin özel gereksinimlerine bağlıdır.

İnteraktif Sistemlerde Çizelgeleme

- Sıralı planlama (Round-robin scheduling)
- Öncelik zamanlaması (Priority scheduling)
- Çoklu kuyruk (Multiple queues)
- Sonraki en kısa süreç (Shortest process next)
- Garantili planlama (Guaranteed scheduling)
- Piyango planlaması (Lottery scheduling)
- Adil paylaşım planlaması (Fair-share scheduling)

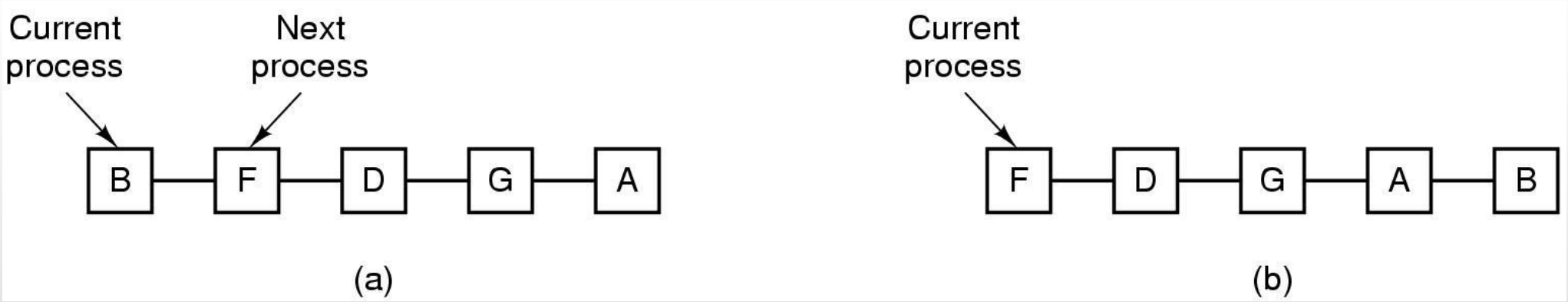
Sıralı Planlama

- Süreçler döngüsel bir sırada yürütülür
- Sırası gelen sabit bir zaman dilimi (time quantum) işlemciyi kullanır.
- Uygulaması basit bir algoritma
- Hem G/Ç'ye bağlı hem de CPU'ya bağlı süreçleri etkili bir şekilde yönetir.
- Zaman diliminin çok küçük olması durumunda yüksek ek maliyet ve kötü yanıt süresine neden olabilir.
- Zaman dilimi bağlam anahtarlama süresinden büyük olması gerekir

Sıralı Planlama

(a) Çalıştırılabilir süreçlerin listesi.

(b) B, kuantumunu kullandıktan sonra çalıştırılabilir süreçlerin listesi.



Sıralı Planlama Örnek Senaryo

- Kuantum = 20

Süreç	P1	P2	P3	P4
Süre	53	17	68	24

- Gantt çizelgesi

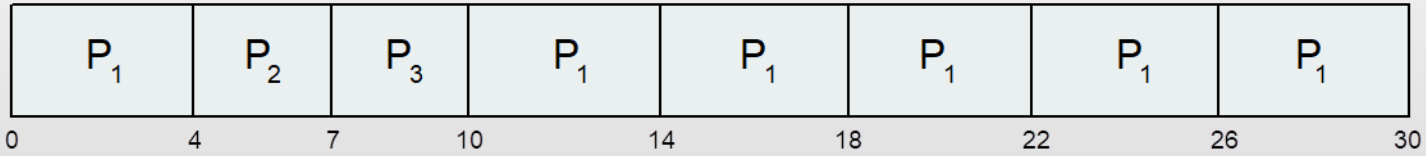
P1	**P2**	**P3**	**P4**	**P1**	**P3**	**P4**	**P1**	**P3**	**P3**
20	37	57	77	97	117	121	134	154	162

- SJF'den yüksek geri dönüş süresi, daha iyi yanıt (response) süresi

Sıralı Planlama Örnek Senaryo

Süreç	Servis süresi
P1	24
P2	3
P3	3

- Kuantum = 4
- Gantt çizelgesi



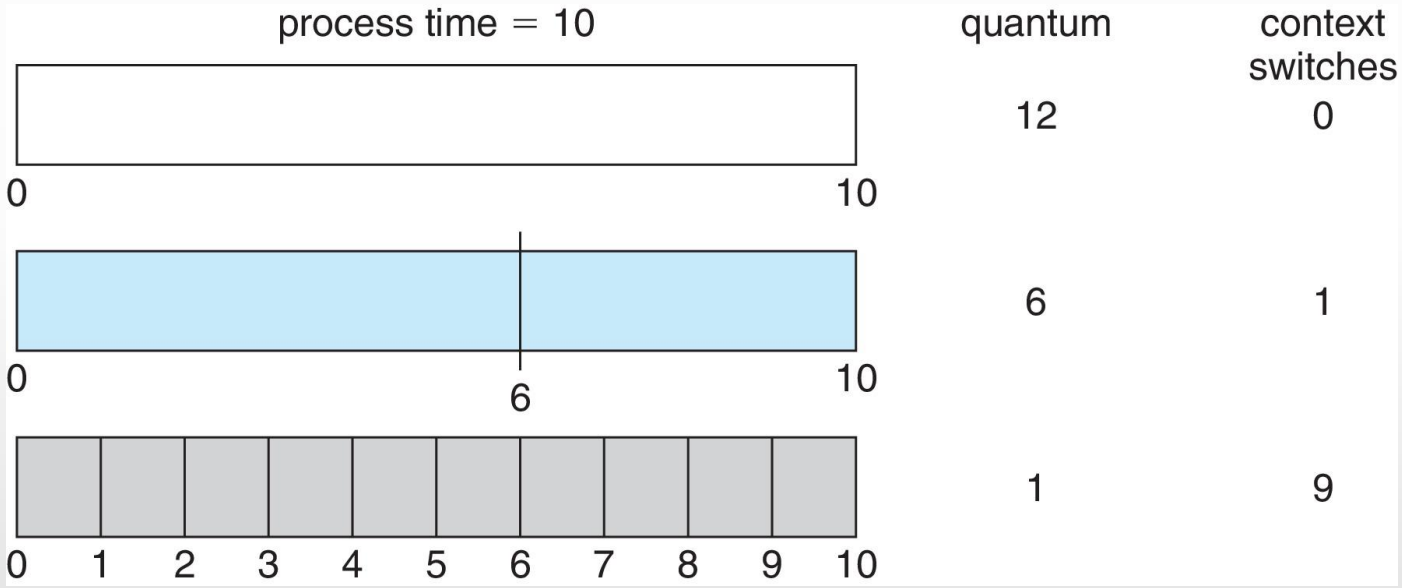
Bağlam Anahtarlama (context switch)

- Sıralı planlamanın performansı büyük ölçüde zaman kuantumunun boyutuna bağlıdır.
- Zaman kuantumu
  - Çok büyük ise FCFS ile benzer
  - Çok küçük:
    - Donanım: Süreç paylaşımı
    - Yazılım: bağlam anahtarlama, yüksek ek maliyet, düşük CPU kullanımı
  - Bağlam anahtarlama masrafına göre büyük olmalıdır

Bağlam Anahtarlama

- Kuantum 4 milisaniye ve bağlam anahtarlama 1 milisaniye ise CPU zamanının %20'si boşa gider
- Kuantum 100 milisaniye ise, boşa harcanan zaman %1'dir, ancak sistem daha az duyarlı olur
- 20-50 milisaniye civarında bir kuantum makul

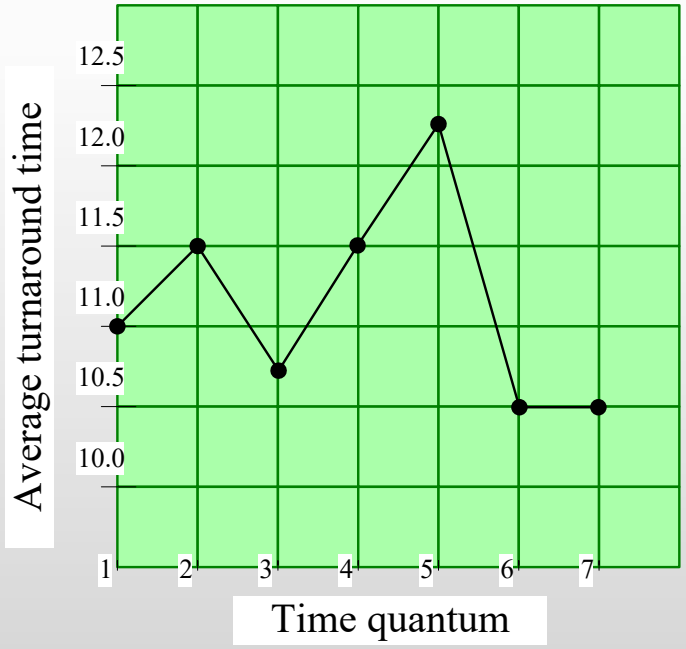
Bağlam Anahtarlama ve Zaman Kuantumu

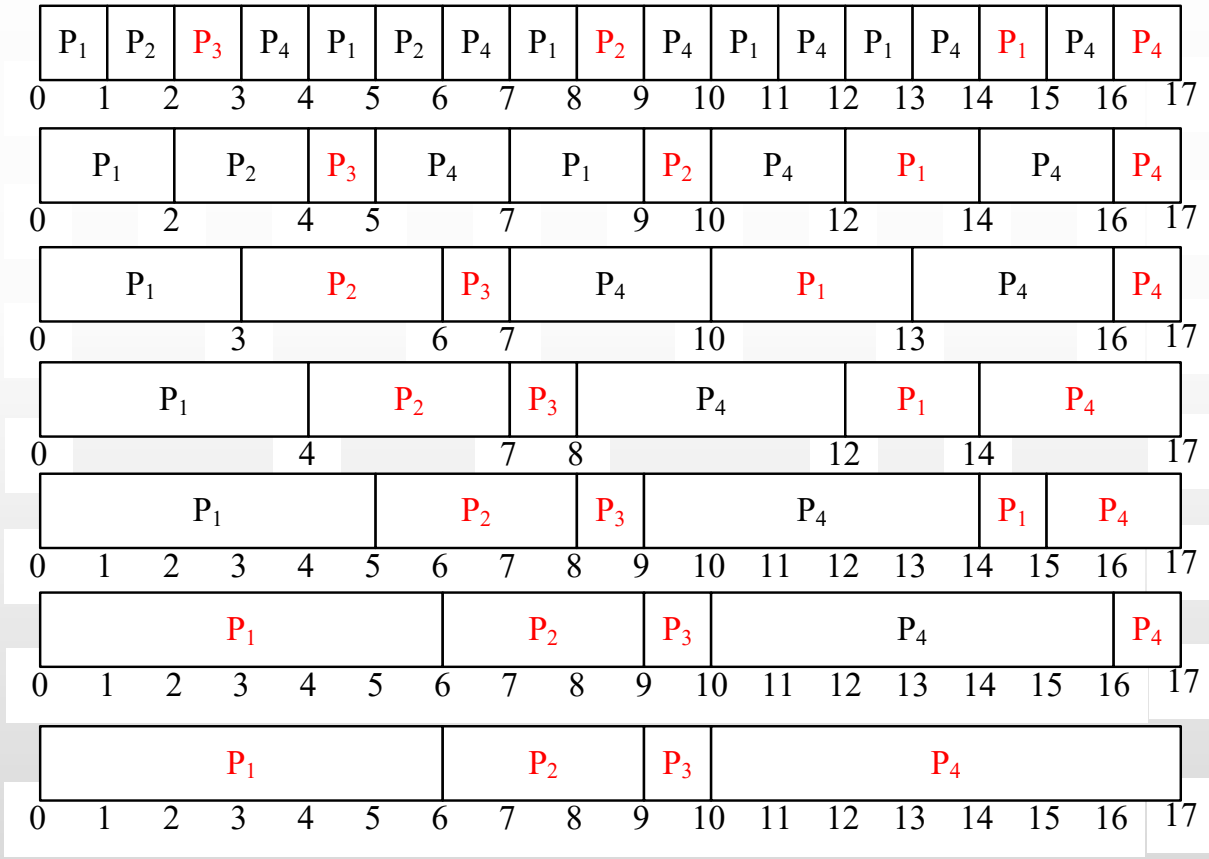


Kuantuma Göre Dönüş Süreleri

- q arttıkça dönüş süresi iyileşir mi?
- İşlemci kullanma zamanlarının %80'i q'dan kısa olmalıdır.

Process	Time
P_1	6
P_2	3
P_3	1
P_4	7



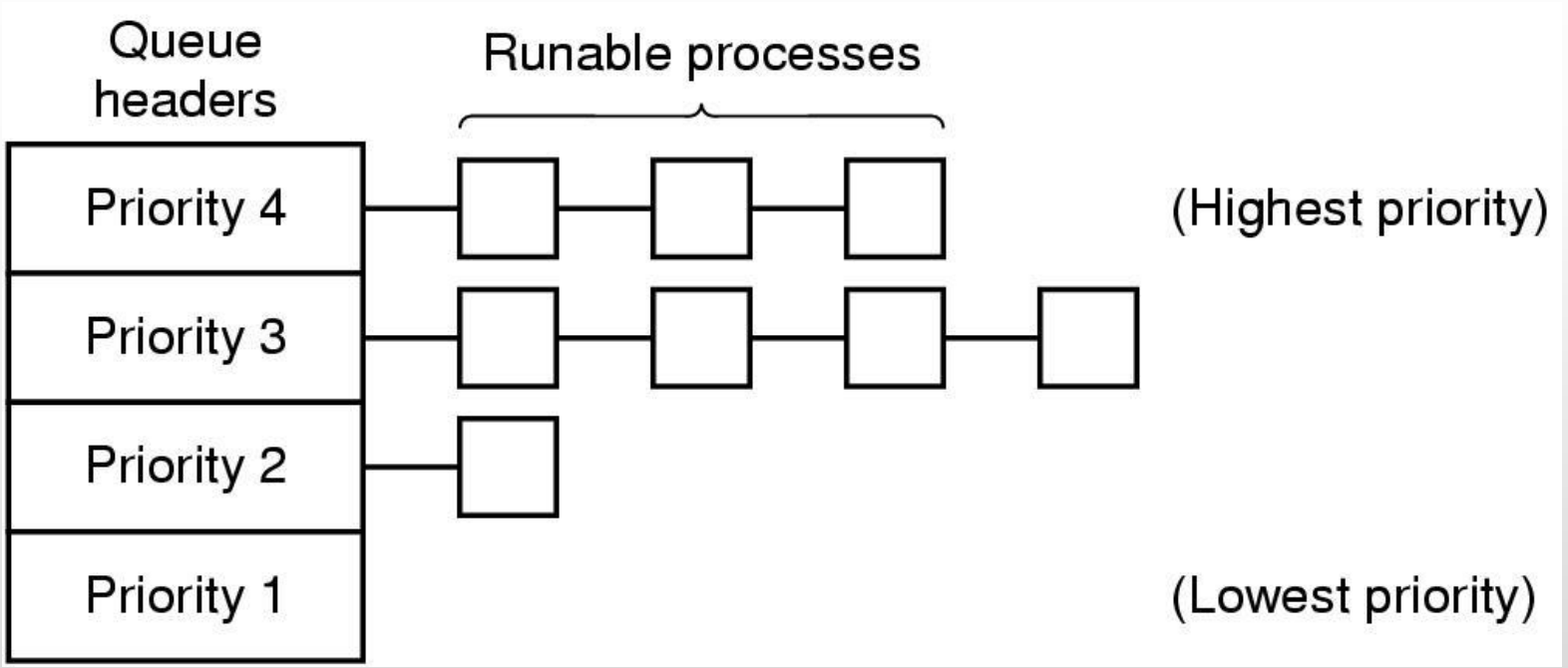


Öncelik Zamanlaması

- Süreçler önceliğini belirten sayısal değerler ile ilişkilendirilir
- Süreçler önceliğe göre yürütülür,
  - yüksek öncelikli süreçler daha önce
- Önemli süreçlere öncelik sağlar, genel sistem yanıt verebilirliğini artırır.
- Açlığa (starvation) neden olabilir
  - Düşük öncelikli süreçlere hiç sıra gelmeyebilir
- Bunu önlemek için yaşlanma (aging) mekanizması
  - Bekleyen süreçlerin önceliğini arttır

Öncelik Zamanlaması

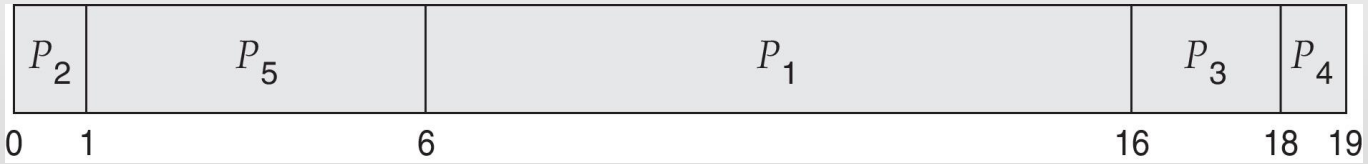
- Her önceliğin bir sayısal karşılığı vardır
- Tüm öncelikler eşitse, FCFS gibi davranır.



Öncelik Zamanlaması Örnek Senaryo

Süreç	Servis süresi	Öncelik
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

- Gantt çizelgesi

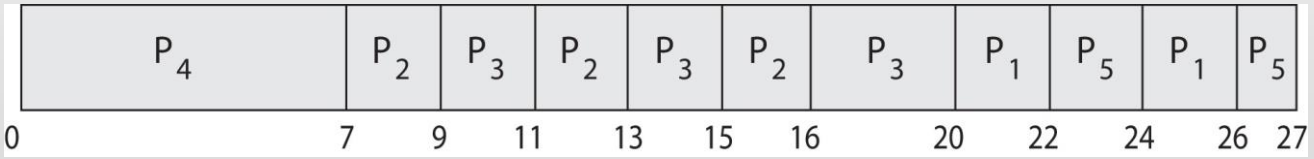


- Ortalama bekleme süresi: 8.2

Öncelik Zamanlaması ve Sıralı Çizelgeleme

Süreç	Servis süresi	Öncelik
P1	4	3
P2	5	2
P3	8	2
P4	7	1
P5	3	3

- Gantt çizelgesi (kuantum = 2)

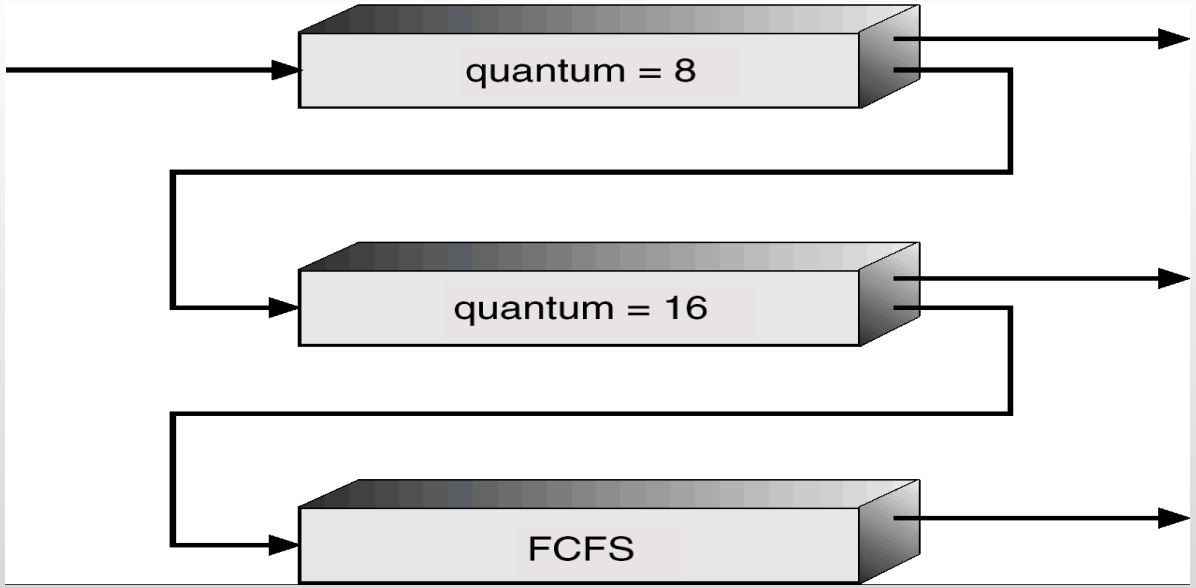


Çoklu Kuyruk

- Süreçler farklı öncelik sınıflarına ayrılır ve ayrı kuyruklara yerleştirilir.
- Önceliği ve yaşlanmayı yönetir, ek yükü azaltarak sistem performansını artırır.
- Uygulaması karmaşıktır, sık sıra değişiklikleri nedeniyle daha yüksek yanıt süresine neden olabilir.
- CPU'ya bağlı sürece büyük kuantum verilir
- Etkileşimli sürece küçük kuantum verilir.
- En yüksek sınıf bir kuantum için koşar; bir sonraki en yüksek sınıf iki kuantum için koşar, vb. Bir süreç kuantumunu tükettiğinde, bir sonraki sınıfa taşınır.

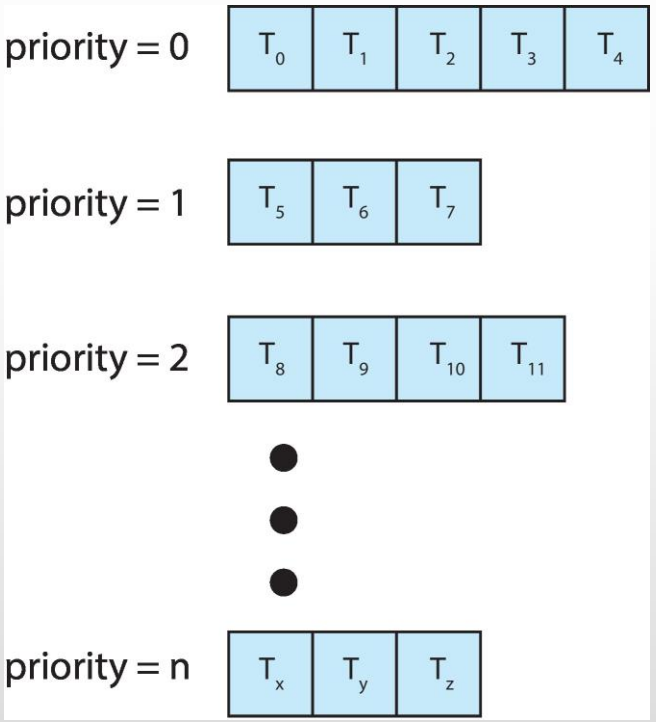
Çoklu Kuyruk Örnek Senaryo

- Üç kuyruk:
  - Q0 – zaman kuantumu 8 milisaniye, FCFS
  - Q1 – zaman kuantumu 16 milisaniye, FCFS
  - Q2 – FCFS



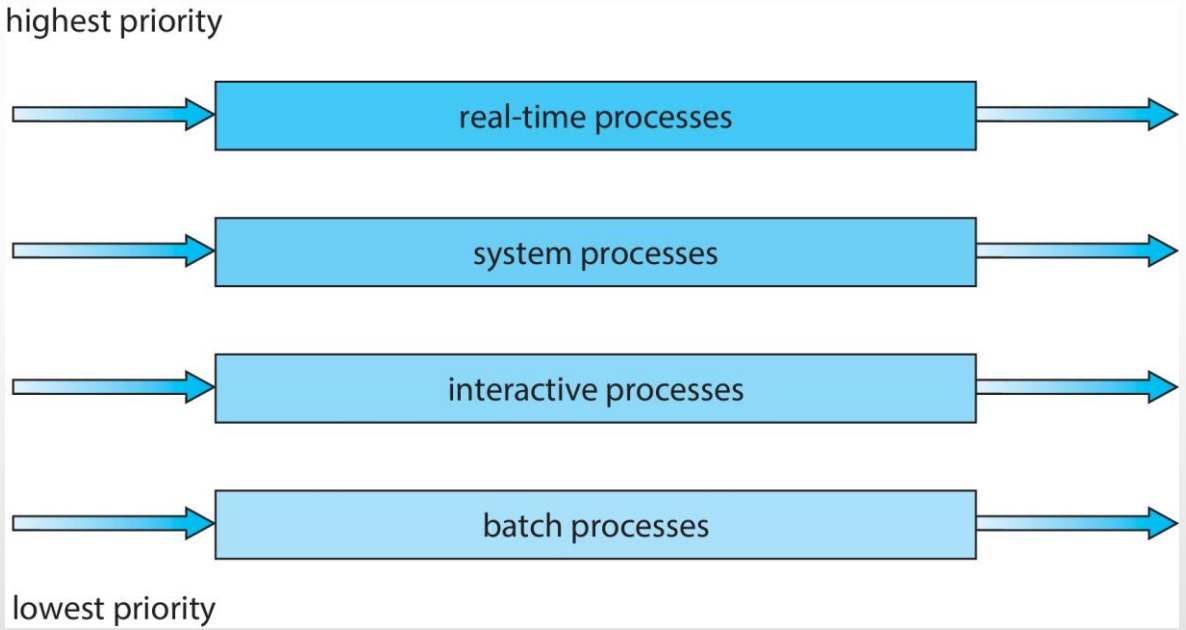
Çoklu Kuyruk

- Her öncelik seviyesi için farklı kuyruklar vardır
- Süreç çalıştırıldıktan sonra farklı kuyruğa atanabilir
- Her seviyede kullanılan algoritma değişebilir
- Terminalde her satırbaşı (enter tuşu) yazıldığında terminale ait işlem en yüksek öncelik sınıfına taşınıyor.



Çoklu Seviye Kuyruk

- .



Sonraki En Kısa Süreç

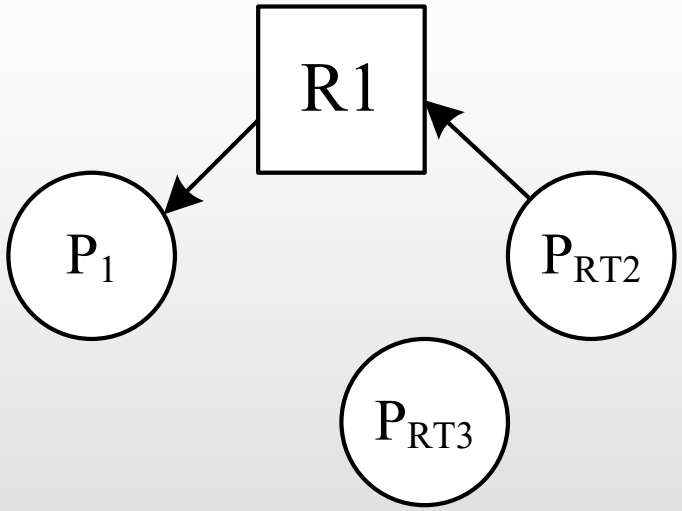
- Süreçler, sonraki sürecin kalan süresinin uzunluğuna göre çizelgelenir.
- CPU kullanımını en üst düzeye çıkarır
- Ortalama bekleme süresini azaltır.
- İnteraktif sistemlerde bir sürecin kalan süresini tahmin etmek zordur.
- Geçmiş davranışa dayalı olarak tahminde bulunulabilir
  - T0, T0 / 2 + T1 / 2, T0 / 4 + T1 / 4 + T2 / 2..
  - Geçerli ve önceki tahminin ağırlıklı ortalaması alınarak

Öncelik Ters Çevirme

- Yüksek öncelikli süreç, daha düşük öncelikli bir sürecin bitmesini beklemesi gerekebilir
  - Düşük öncelikli süreç sistem kaynağına erişmiş olabilir
  - Kritik bölgede olabilir

Öncelik Ters Çevirme

- Öncelik: P1 < PRT3 < PRT2
- PRT3, P1'i önler; PRT2, P1'i bekler
- PRT2, PRT3'ü bekler



Garantili Çizelgeleme

- Önceden belirlenmiş bir programa göre çizelgeleme yapar.
- Gerçek zamanlı sistemler için uygun, öngörülebilir bir davranış sağlar.
- Sınırlı esneklik, öngörülemeyen iş yükleri durumunda düşük performansa neden olabilir.
- Katı gerçek zamanlıya karşı yumuşak gerçek zamanlı
  - Katı: Bir fabrikada robot kontrolü
  - Yumuşak: CD çalar
- Algoritmalar statik (çalışma süreleri önceden bilinen) veya dinamik (çalışma zamanı kararları) olabilir.

Piyango Çizelgeleme

- Süreçlere bilet atanır ve yürütme için rastgele seçilir.
- İşlemci süresi için saniyede birkaç kez çekiliş yapılır
- Önceliği ve yaşlandırmayı yönetir
- Ek maliyeti azaltır ve sistem performansını artırır.
- Uygulanması karmaşık
- Dengesiz bilet dağılımı durumunda öngörülemez davranır
- "Daha önemli" süreçler için daha fazla çekiliş hakkı tanıyarak önceliklerin değiştirilebilmesine izin verir.

Adil Paylaşım Çizelgeleme

- Süreçler son kullanım geçmişlerine göre çizelgelenir.
- Süreçler arasında kaynakları dengeler,
- Genel sistem yanıt verebilirliğini artırır.
- Kullanım geçmişlerinin izlenmesi, ek yüke neden olabilir.

Değerlendirme

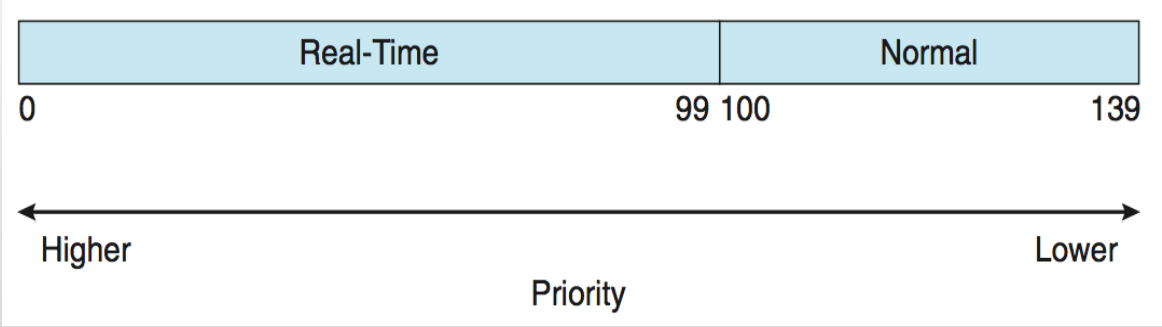
- İşletim sisteminin özel gereksinimlerine bağlı olarak her algoritmanın kendi güçlü ve zayıf yönleri vardır.
- Round-Robin, Priority, Multiple Queues ve Fair-Share çizelgelemelerinin tümü önleyicidir ve CPU zaman paylaşımına izin verir.
- Shortest Process Next, Guaranteed, Lottery çizelgelemeleri öncelikli değildir ve CPU kullanımını en üst düzeye çıkarmayı amaçlar.
- Algoritma seçimi, işletim sisteminin yanıt verebilirlik, öngörülebilirlik, kaynak tahsisi ve adalet gibi hedeflerine bağlıdır.

POSIX Gerçek Zamanlı Çizelgeleme

- POSIX.1b standardı
- API, gerçek zamanlı iş parçacıklarını yönetmek için işlevler sağlar
- Gerçek zamanlı iş parçacıkları için iki seçenek tanımlar:
  - SCHED_FIFO - FIFO kuyruğuna sahip FCFS stratejisi kullanır. Eşit önceliğe sahip iş parçacıkları için zaman dilimleme yoktur
  - SCHED_RR - Eşit önceliğe sahip iş parçacıkları için zaman dilimleme vardır
- Çizelgeleme ilkesini öğrenmek ve değer atamak için:
- `pthread_attr_getsched_policy(pthread_attr_t *attr, int *policy)`
- `pthread_attr_setsched_policy(pthread_attr_t *attr, int policy)`

Linux Çizelgeleme

- Completely fair scheduler (CFS) kullanır
- Her bir sürecin belirli bir önceliği vardır
- En yüksek çizelgeleme sınıfındaki en yüksek öncelikli görevi seçer
- Kuantum temelli sabit zaman dilimleri atamak yerine CPU kullanım oranına dayalıdır
- İki çizelgeleme sınıfı vardır, genişletilebilir
  - Varsayılan, gerçek zamanlı
- Gerçek zamanlı görevlerin statik öncelikleri vardır



Windows Çizelgeleme

- Önceliğe dayalı önleyici (priority based preemptive) çizelgeleme kullanır
- Sonraki adımda en yüksek öncelikli iş parçacığı çalışır
- İş parçacığı (1) bloke olana kadar, (2) zaman dilimi bitene kadar, (3) daha yüksek öncelikli bir iş parçacığı gelene kadar çalışır.
- Gerçek zamanlı iş parçacıkları, diğerlerini bloke edebilir
- 32 seviyeli öncelik şeması kullanır
  - Değişken sınıf 1-15, gerçek zamanlı sınıf 16-31
- Öncelik 0, bellek yönetimi iş parçacığıdır
- Her öncelik için kuyruk tutulur
- Çalıştırılabilir iş parçacığı yoksa, boş (idle) iş parçacığını çalıştırır

Windows Öncelikler

- .

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

SON