Bölüm 2.1: Yazılım ve İşletim Sistemleri

Bilgisayar ve Programlamaya Giriş

Yazılım | İşletim sistemi | Kaynak yönetimi | Sistem türleri | Lisanslama

Sercan KÜLCÜ · Giresun Üniversitesi

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/computer/slides/Bolum_02_01_Yazilim_ve_Isletim_Sistemleri.html

GİRİŞ

Öğrenme Çıktıları

Yazılım türleri

Sistem ve uygulama yazılımlarını ayırt etmek.

İşletim sistemi

İşletim sisteminin kullanıcı, uygulama ve donanım arasındaki rolünü açıklamak.

Kaynak yönetimi

Süreç, bellek, dosya ve aygıt yönetimini yorumlamak.

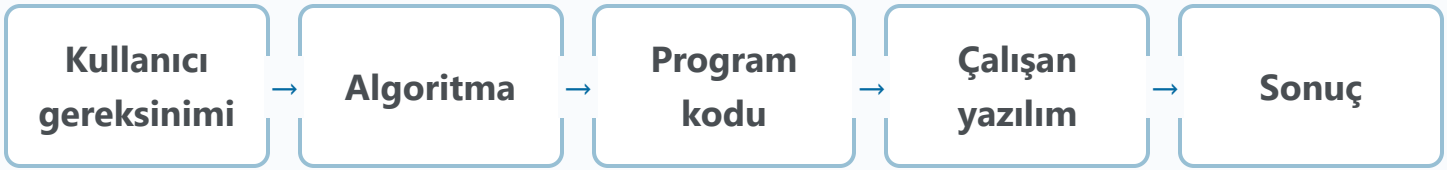
Sistem seçimi

Farklı işletim sistemi türlerini gereksinimlere göre karşılaştırmak.

YAZILIM

Yazılım Nedir?

Yazılım, bir bilgisayarın gerçekleştireceği işlemleri tanımlayan programlar, veriler ve ilgili belgelerin bütünüdür.



Donanım işlemi fiziksel olarak yürütür; yazılım hangi işlemin, ne zaman ve nasıl yapılacağını belirler.

YAZILIM

Yazılımın Kalite Özellikleri

İşlevsellik
Beklenen görevleri doğru yerine getirme.

Kullanılabilirlik
Kolay öğrenilme ve etkili kullanım.

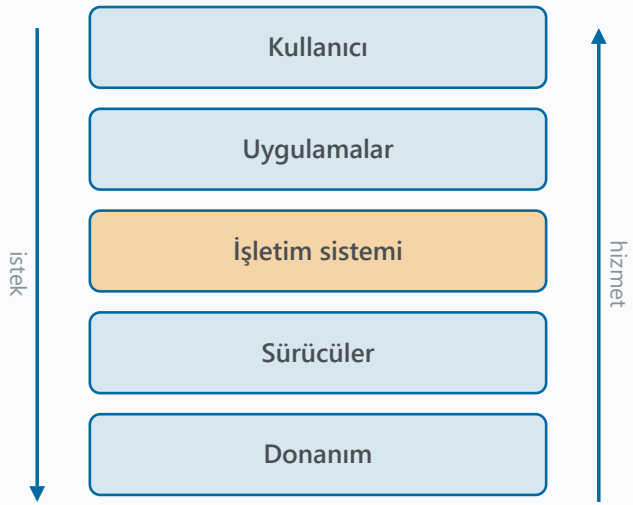
Performans
Zaman ve kaynakları verimli kullanma.

Güvenilirlik
Hata durumlarında öngörülebilir davranma.

Güvenlik
Yetkisiz erişim ve değişikliklere karşı koruma.

Bakım kolaylığı
Düzeltme ve geliştirmelerin yönetilebilir olması.

Yazılım ve Donanım İlişkisi



- Kullanıcı yalnızca en üst katmanlarla etkileşir.
- Her katman, altındaki ayrıntıları gizleyerek üst katmana daha kolay hizmetler sunar (soyutlama).
- Aynı uygulama, sürücüler sayesinde farklı donanımlarda çalışabilir.

İstek yukarıdan aşağı iner, sonuç (hizmet) aşağıdan yukarı döner.

YAZILIM

Yazılım Türleri

Sistem yazılımı

Donanımı yönetir ve diğer yazılımlara çalışma ortamı sağlar.

Uygulama yazılımı

Kullanıcının belirli işlerini yerine getirir.

Programlama araçları

Kod yazma, çevirme, test ve hata ayıklama işlemlerini destekler.

Ürün yazılımı (firmware)

Bir cihazın temel işlevlerini yöneten gömülü yazılımdır.



YAZILIM

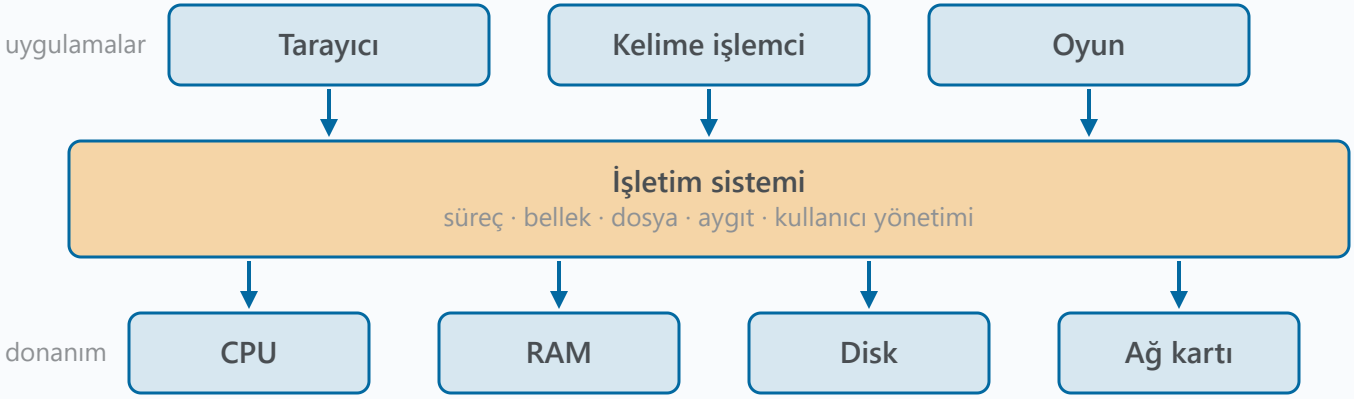
Sistem ve Uygulama Yazılımları

Tür	Örnekler	Temel amaç
Sistem	İşletim sistemi, sürücü, yardımcı araç	Bilgisayarı ve kaynakları yönetmek
Uygulama	Kelime işlemci, tarayıcı, oyun	Kullanıcı görevini gerçekleştirmek
Ara yazılım	Veri tabanı bağlantı katmanı, mesaj kuyruğu	Uygulamaların sistemle ve birbiriyle haberleşmesini kolaylaştırmak
Programlama araçları	Derleyici, tümleşik geliştirme ortamı (IDE), hata ayıklayıcı	Yeni yazılım üretmek

İŞLETİM SİSTEMİ

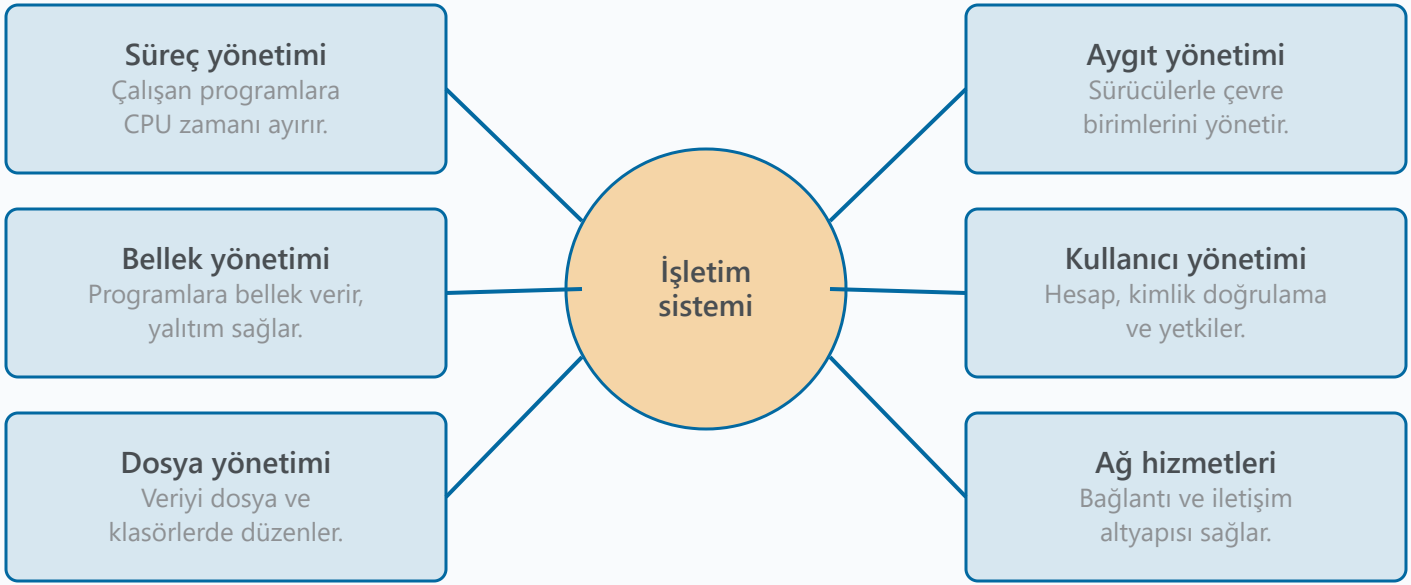
İşletim Sistemi Nedir?

İşletim sistemi, donanım kaynaklarını yöneten ve uygulamalara ortak hizmetler sunan temel sistem yazılımıdır.

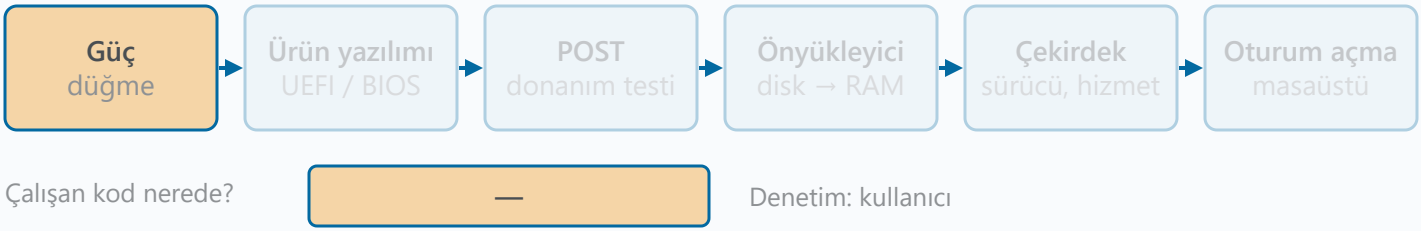


- Donanım ayrıntılarını uygulamalardan soyutlar.
- Birden fazla programın kaynakları güvenli biçimde paylaşmasını sağlar.
- Kullanıcıya grafiksel veya komut satırı arayüzü sunar.

İşletim Sisteminin Görevleri

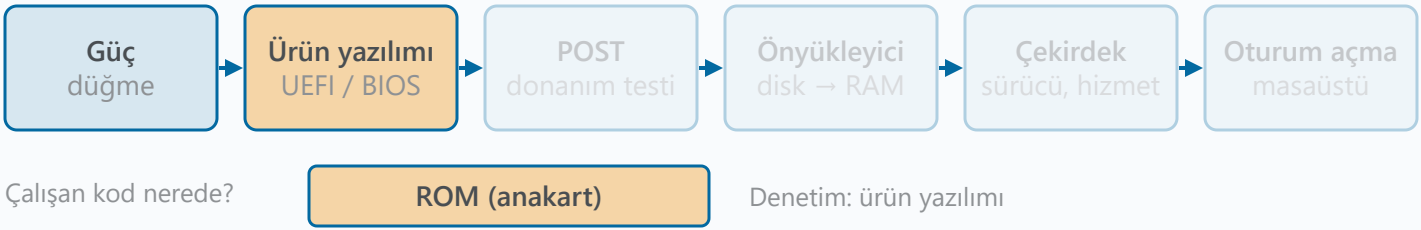


Bilgisayar Nasıl Açılır? Önyükleme Adım Adım



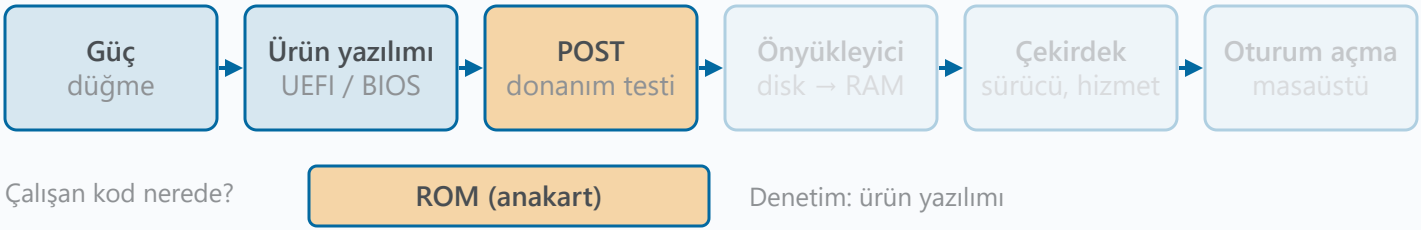
Güç düğmesine basılır; gerilimler kararlı olunca güç kaynağı anakarta “güç hazır” sinyali verir. RAM boştur; işletim sistemi henüz diskte bekler.

Bilgisayar Nasıl Açılır? Önyükleme Adım Adım



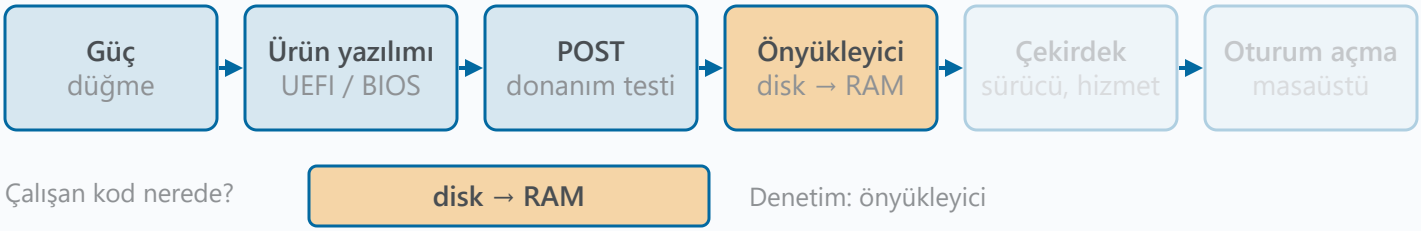
İşlemci, anakarttaki kalıcı bellekte (ROM/flash) duran **ürün yazılımını** (UEFI ya da eski adıyla BIOS) çalıştırmaya başlar.

Bilgisayar Nasıl Açılır? Önyükleme Adım Adım



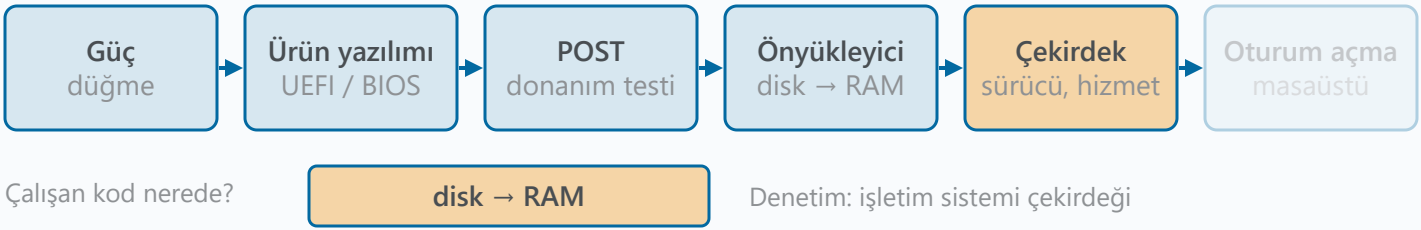
POST (açılışta öz sınama): bellek, klavye, disk gibi temel donanım denetlenir. Hata varsa sesli uyarı ya da ileti verilir.

Bilgisayar Nasıl Açılır? Önyükleme Adım Adım



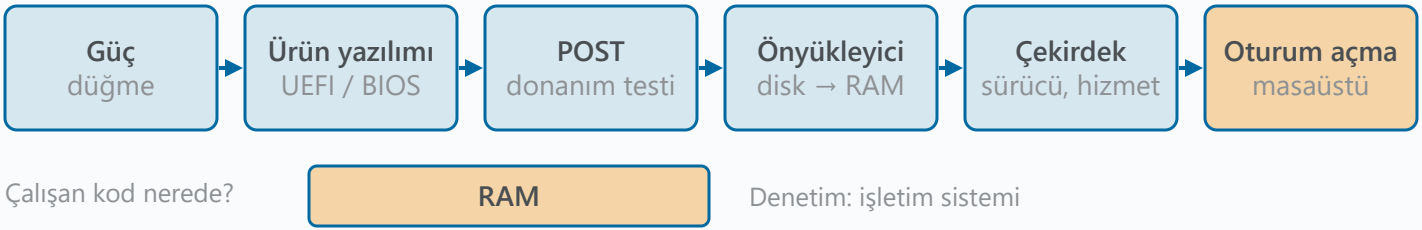
Ayarlardaki önyükleme sırası (ör. önce USB, sonra SSD) denenir; takılı olmayan aygıt atlanır. Diskteki küçük **EFI sistem bölümünden** (FAT32) **önyükleyici** (ör. Windows Boot Manager: bootmgfw.efi, GRUB) RAM'e yüklenir.

Bilgisayar Nasıl Açılır? Önyükleme Adım Adım



Önyükleyici **çekirdeği** (Windows'ta winload.efi aracılığıyla ntoskrnl.exe) RAM'e yükleyip denetimi ona verir. Çekirdek sürücüleri yükler, ilk süreçleri ve sistem hizmetlerini başlatır.

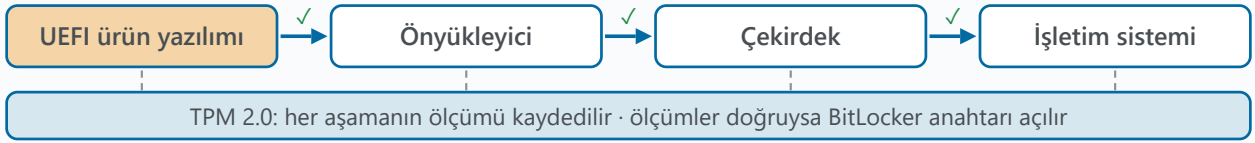
Bilgisayar Nasıl Açılır? Önyükleme Adım Adım



Oturum açma ekranı gelir; kimlik doğrulandıktan sonra kullanıcının masaüstü ve başlangıç uygulamaları açılır.

İŞLETİM SİSTEMİ · GÜVENLİK

Güvenli Önyükleme ve TPM



UEFI

BIOS'un yerini alan modern ürün yazılımı; büyük disk, hızlı açılış ve dijital imza denetimi sağlar.

Güvenli Önyükleme

Yalnızca güvenilir dijital imzası olan önyükleyici ve çekirdek çalışır; sahte ya da değiştirilmiş önyükleme kodunu engeller.

TPM 2.0

Şifreleme anahtarlarını ve açılış ölçümlerini saklayan güvenlik yongası (anakartta, işlemcide veya ürün yazılımında); BitLocker gibi araçlar kullanır.

Windows 11

Kurulum için UEFI + Güvenli Önyükleme desteği ve TPM 2.0 ister; ayrıntılar Microsoft'un güncel gereksinim sayfasında yer alır.

İki bölümleme düzeni vardır. **MBR**: BIOS ile birlikte, 2 TB'a kadar disk, en çok 4 birincil bölüm. **GPT**: UEFI ile birlikte, 2 TB üzeri disk, çok sayıda bölüm.

İŞLETİM SİSTEMİ BİLEŞENLERİ

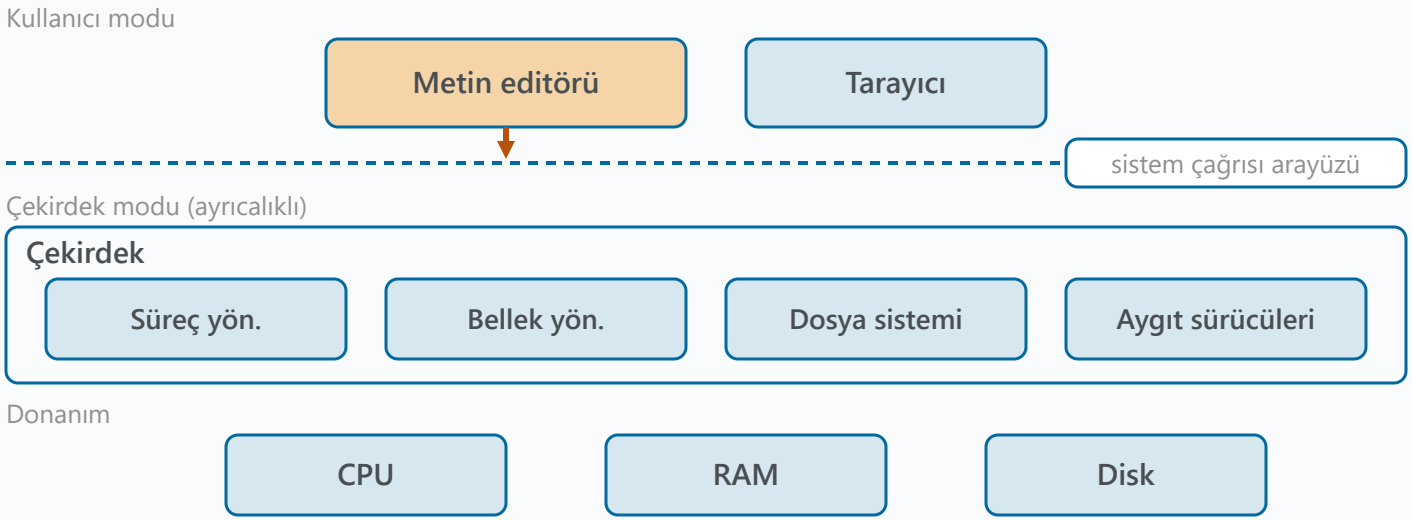
Çekirdek (Kernel)

Çekirdek, işletim sisteminin ayrıcalıklı çalışan merkezî bölümüdür. Uygulamalar donanıma doğrudan erişmek yerine sistem çağrılarıyla çekirdekten hizmet ister.



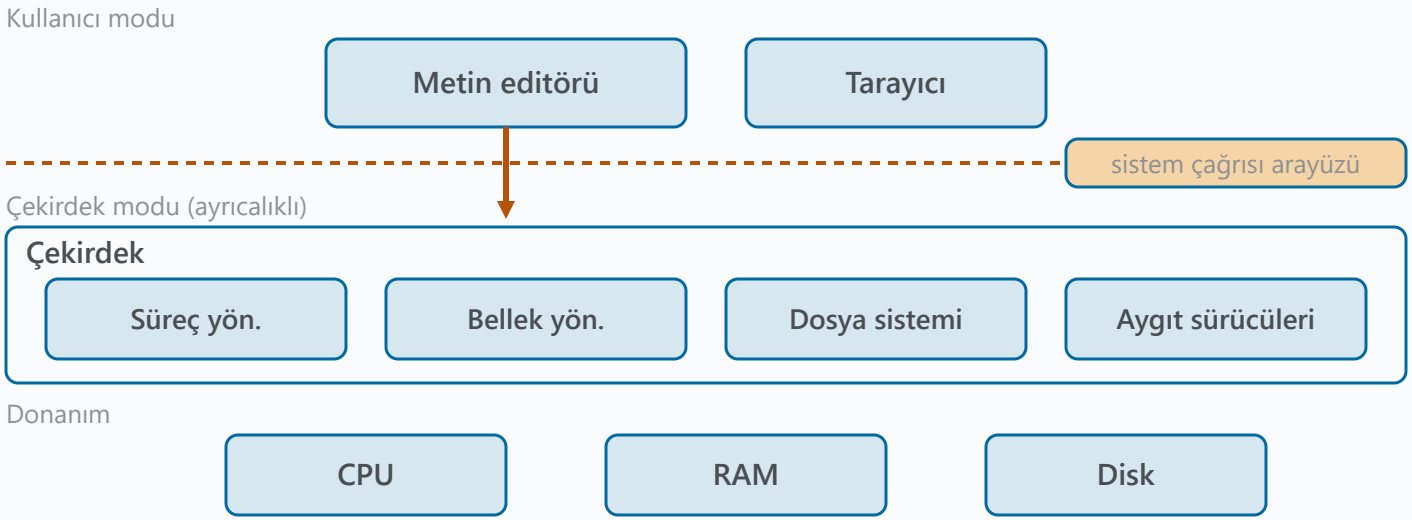
Çekirdekteki bir hata tüm sistemi etkileyebilir; bu nedenle erişim denetimi ve yalıtım önemlidir.

Sistem Çağrısı Adım Adım: Dosya Okuma



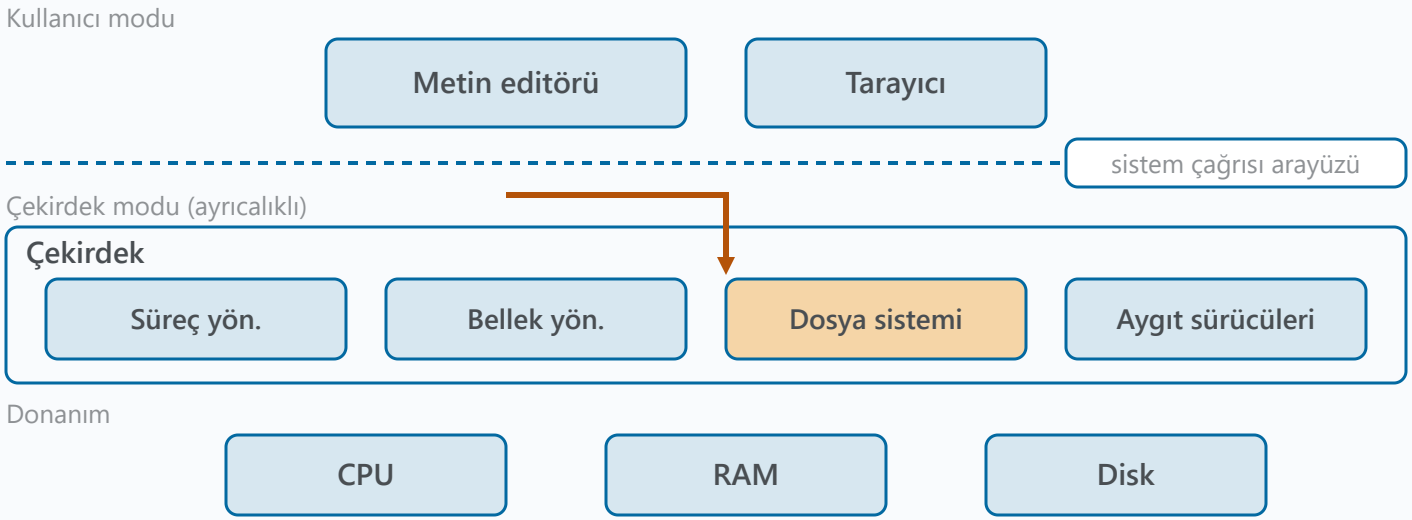
Metin editörü **rapor.txt** dosyasını okumak ister. Diske kendisi erişemez; **read** sistem çağrısını yapar.

Sistem Çağrısı Adım Adım: Dosya Okuma



İşlemci **kullanıcı modundan çekirdek moduna** geçer. Artık yalnızca çekirdek kodu çalışır.

Sistem Çağrısı Adım Adım: Dosya Okuma



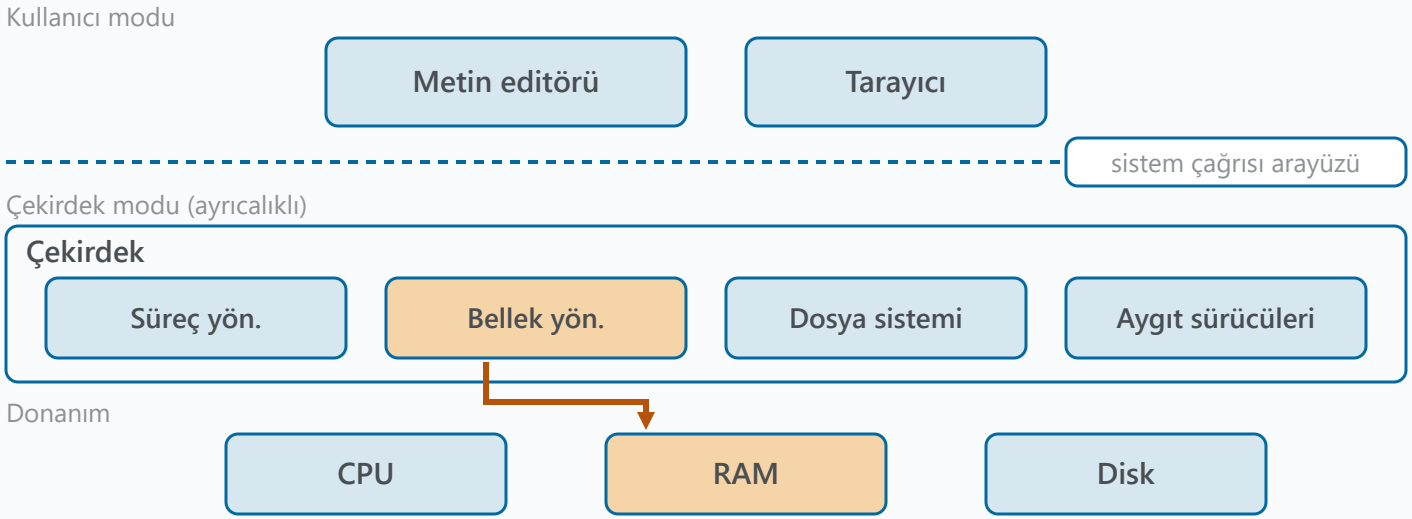
Dosya sistemi dosyanın diskteki yerini bulur ve kullanıcının **okuma izni** olup olmadığını denetler.

Sistem Çağrısı Adım Adım: Dosya Okuma



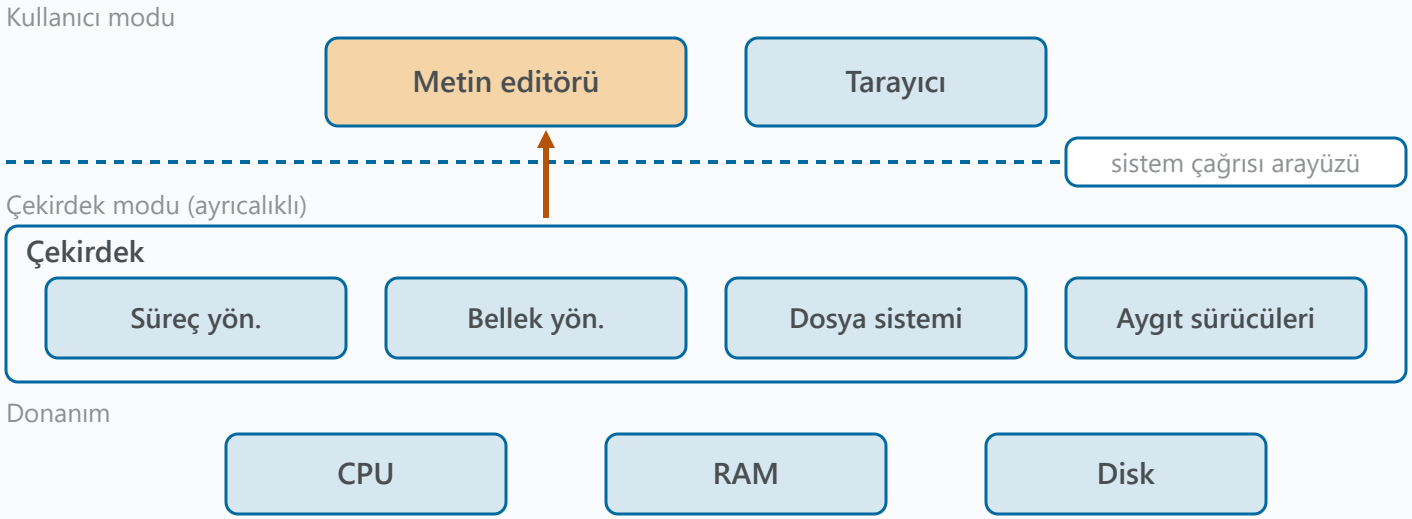
Disk sürücüsü diske okuma komutu verir. Disk işi bitince **kesme** (interrupt) ile işlemciye haber verir.

Sistem Çağrısı Adım Adım: Dosya Okuma



Bellek yönetimi: çekirdek okunan veriyi RAM'de editörün kendi bellek alanına kopyalar.

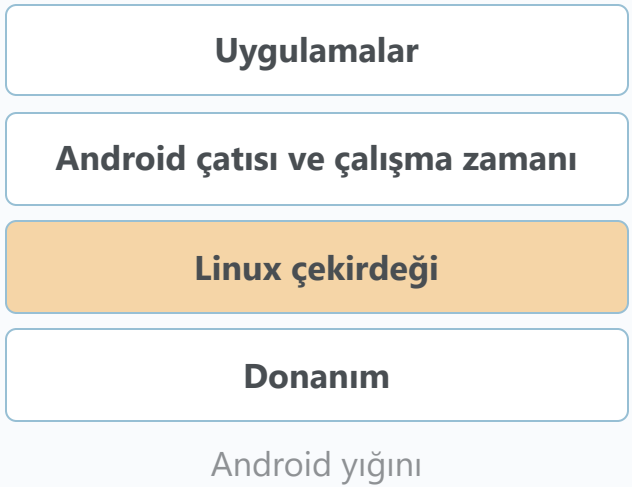
Sistem Çağrısı Adım Adım: Dosya Okuma



Çağrı döner, işlemci tekrar **kullanıcı moduna** geçer. Editör veriyi ekranda gösterir. Uygulama donanımı hiç doğrudan görmedi.

Çekirdek Türleri

Tür	Fikir	Örnek
Monolitik çekirdek	Süreç, bellek, dosya sistemi ve sürücüler tek ayrıcalıklı çekirdekte; hızlı ama büyük	Linux (modüller yüklenebilir)
Mikro çekirdek	Çekirdekte yalnızca temel işler; sürücü ve dosya sistemi kullanıcı modunda; yalıtımı güçlü	MINIX 3, QNX
Hibrit çekirdek	Mikro çekirdek tasarımını alıp hız için bazı hizmetleri çekirdekte tutar	Windows NT çekirdeği, macOS/iOS (XNU)



Android yığını

Linux yalnızca bir çekirdektir; Linux dağıtımı çekirdek + araçlar + masaüstüdür. **Android**, Linux çekirdeğini kullanır ancak üstündeki katmanlar Android'e özgüdür.

KAYNAK YÖNETİMİ

Süreç Yönetimi

Program
Diskte saklanan komut ve veri bütünüdür.

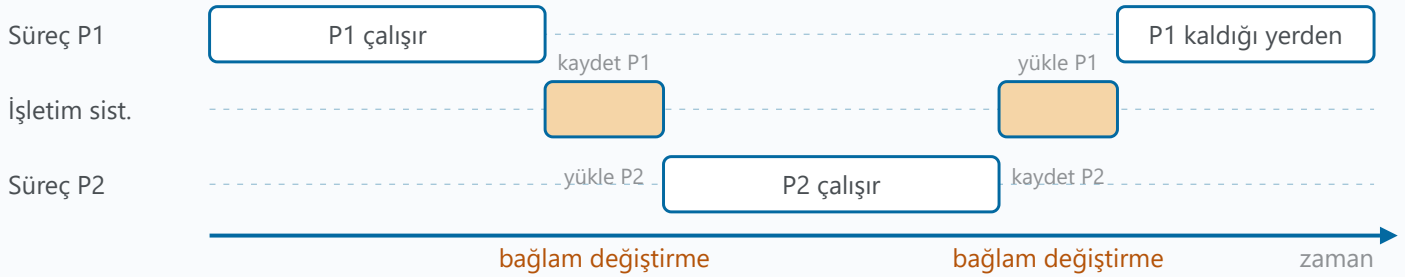
Süreç
Çalışmakta olan program örneğidir.

İş parçacığı
Süreç içindeki yürütme akışıdır.

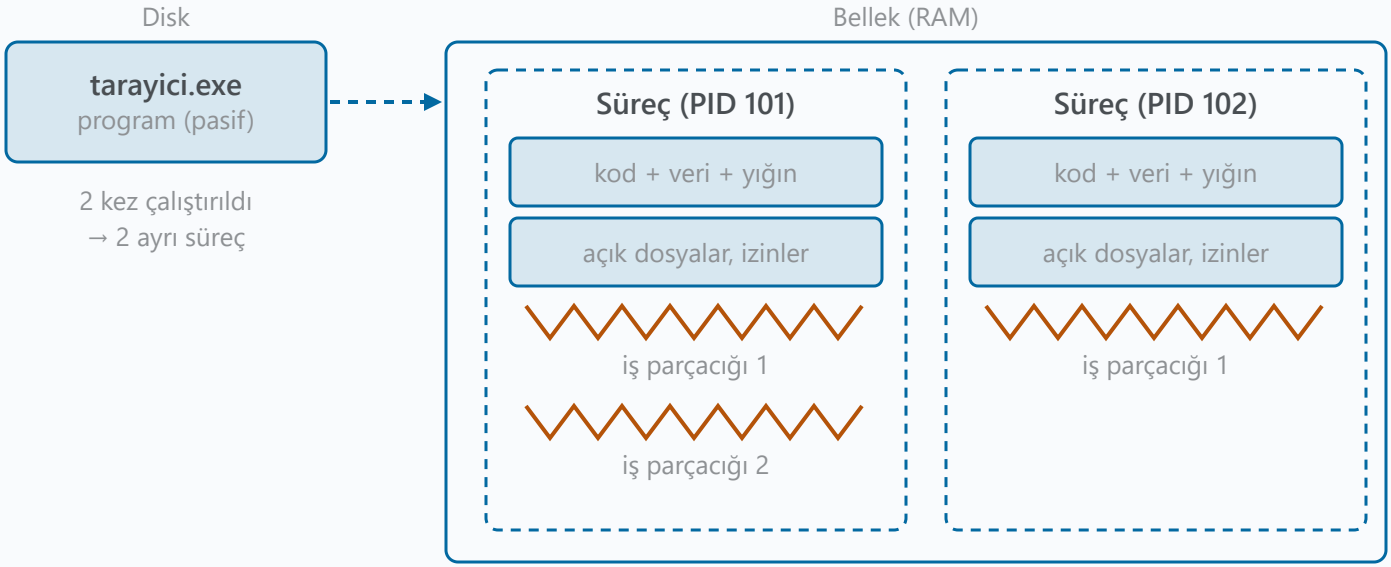
Çizelgeleme
CPU'nun hangi sürece ne zaman verileceğini belirler.

Bağlam değiştirme
CPU durumunu bir süreçten diğerine aktarır.

Eş zamanlılık
Birden fazla işin ilerlemesini düzenler.



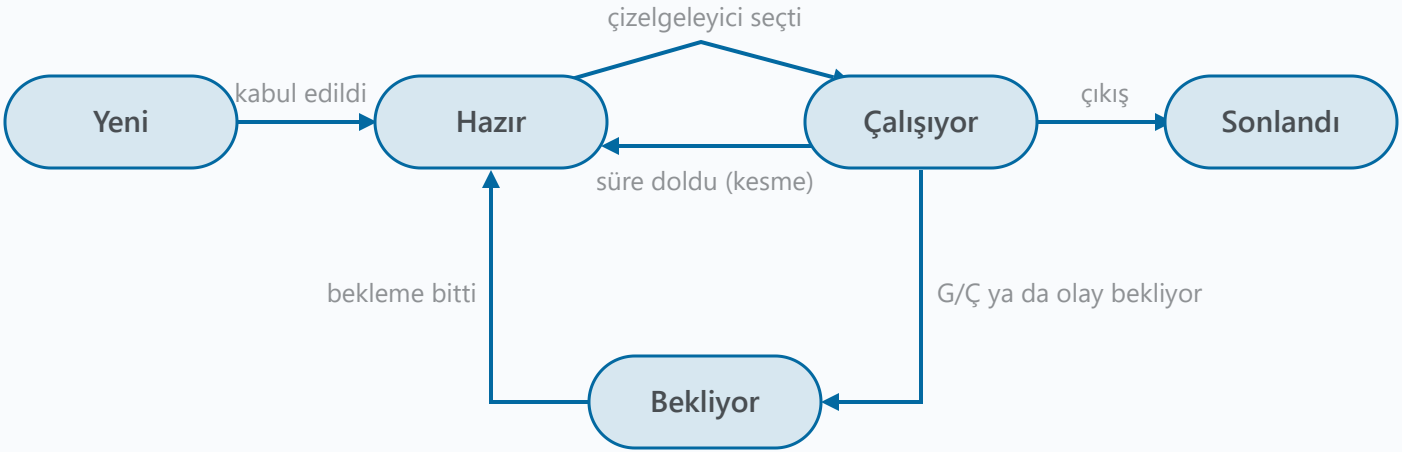
Program, Süreç ve İş Parçacığı



Program diskte duran dosyadır; süreç onun çalışan örneğidir ve bir süreç kimliğiyle (PID) tanınır. Her sürecin ayrı bellek alanı vardır, iş parçacıkları ise süreç içindeki belleği paylaşır.

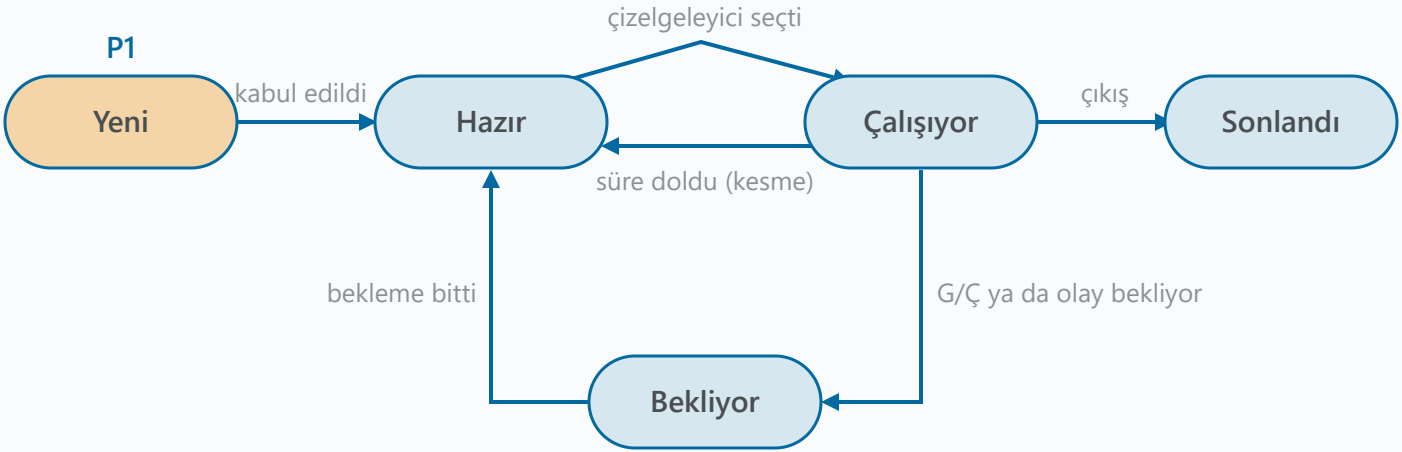
KAYNAK YÖNETİMİ

Süreç Durumları



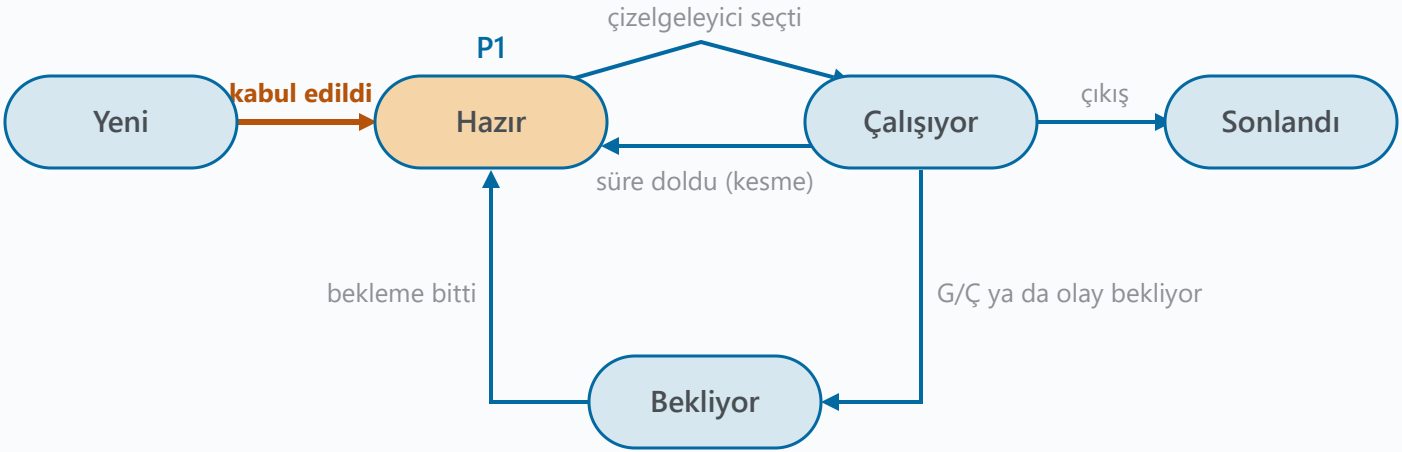
- Tek çekirdekte aynı anda yalnızca **bir** süreç **Çalışıyor** durumundadır; diğer hazır süreçler sırada bekler.
- **Bekliyor** durumundaki süreç CPU'ya verilmez; beklediği olay (disk, klavye, ağ) gerçekleşince yeniden **Hazır** olur.

Süreç Durum Geçişleri Adım Adım



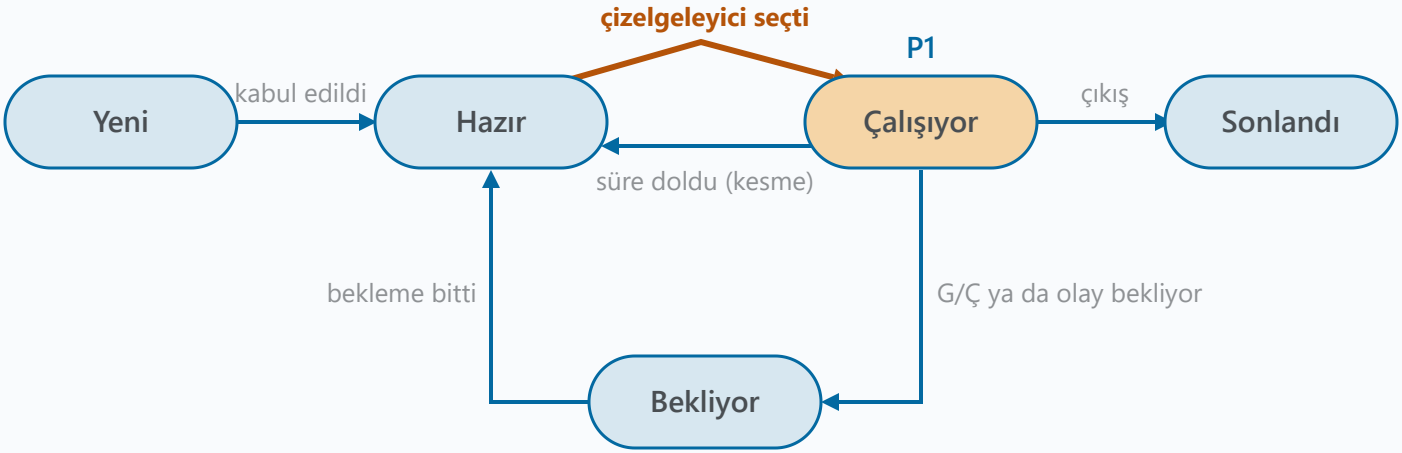
P1 süreci oluşturuluyor (ör. kullanıcı bir programa çift tıkladı): durum **Yeni**.

Süreç Durum Geçişleri Adım Adım



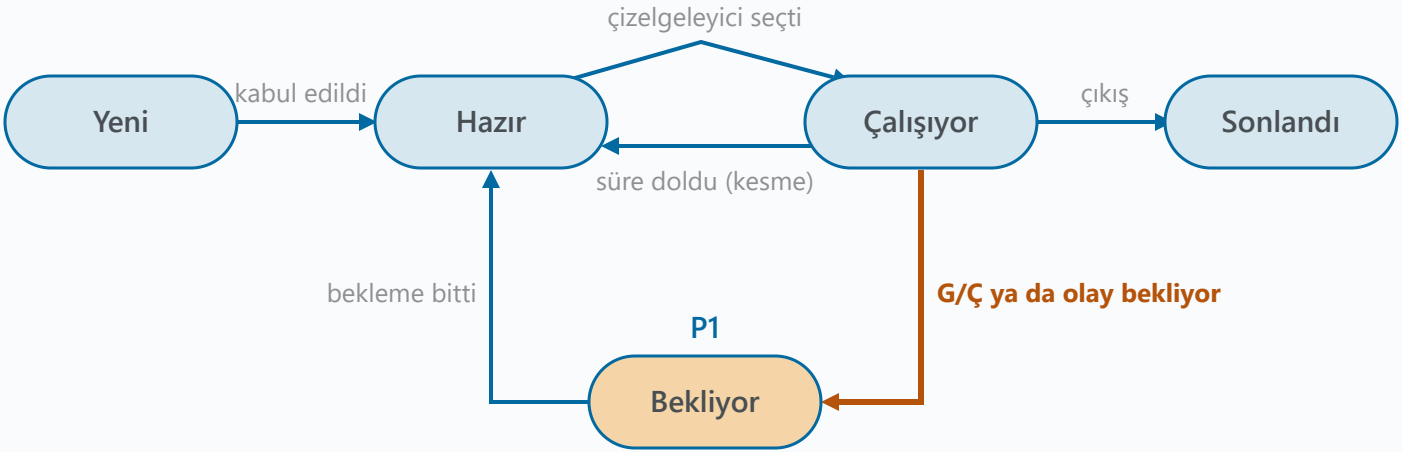
Yeni → **Hazır**: İşletim sistemi süreci **kabul eder**; bellek ayrılır ve süreç hazır kuyruğuna girer.

Süreç Durum Geçişleri Adım Adım



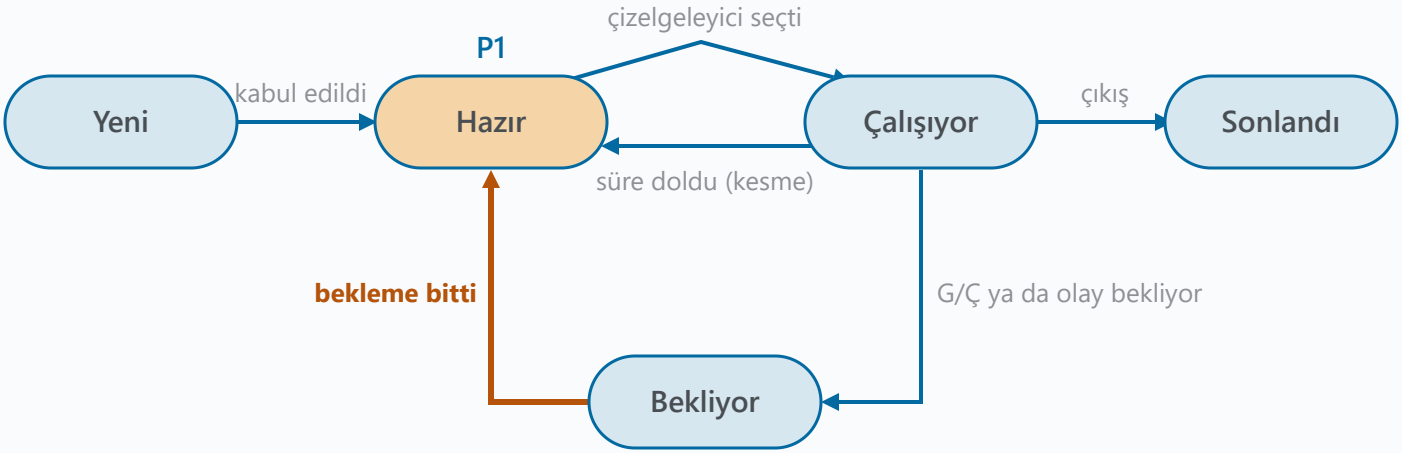
Hazır → **Çalışıyor**: **Çizelgeleyici** P1'i seçer; bağlam yüklenir ve CPU P1'in komutlarını yürütür.

Süreç Durum Geçişleri Adım Adım



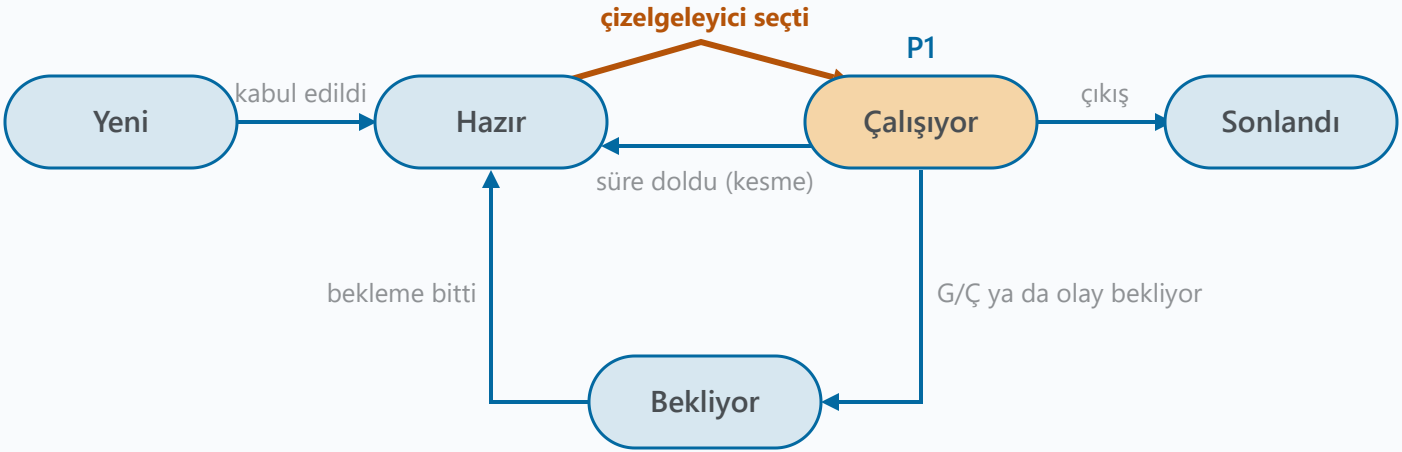
Çalışıyor → **Bekliyor**: P1 diskten veri ister. Veri gelene kadar CPU'yu boşa tutmaması için **Bekliyor** durumuna geçer.

Süreç Durum Geçişleri Adım Adım



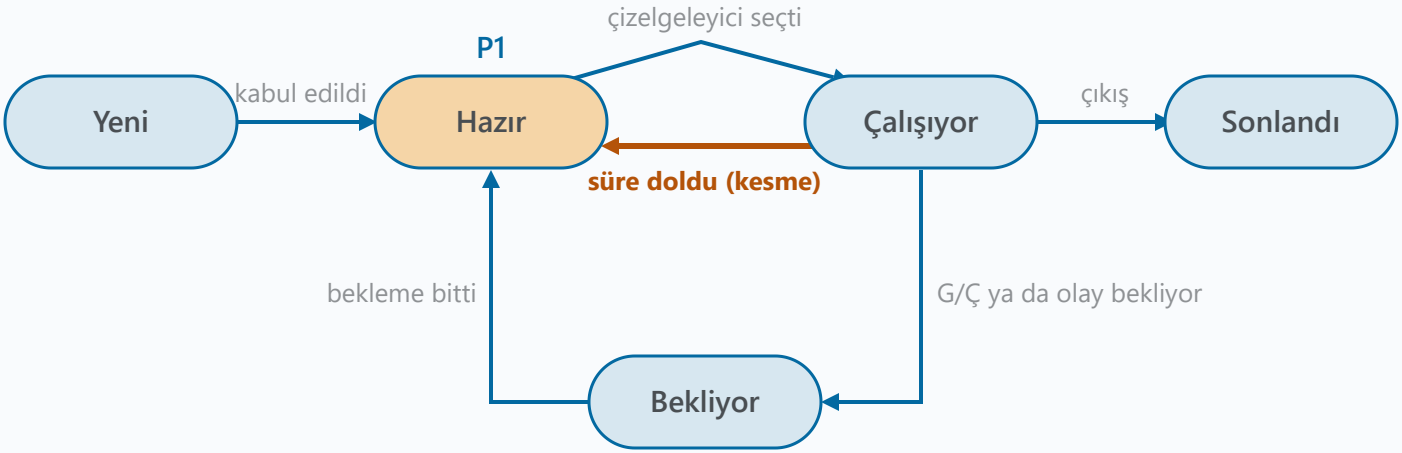
Bekliyor → **Hazır**: Disk işi bitti (kesme geldi). P1 hemen çalışmaz; önce **Hazır** kuyruğuna döner.

Süreç Durum Geçişleri Adım Adım



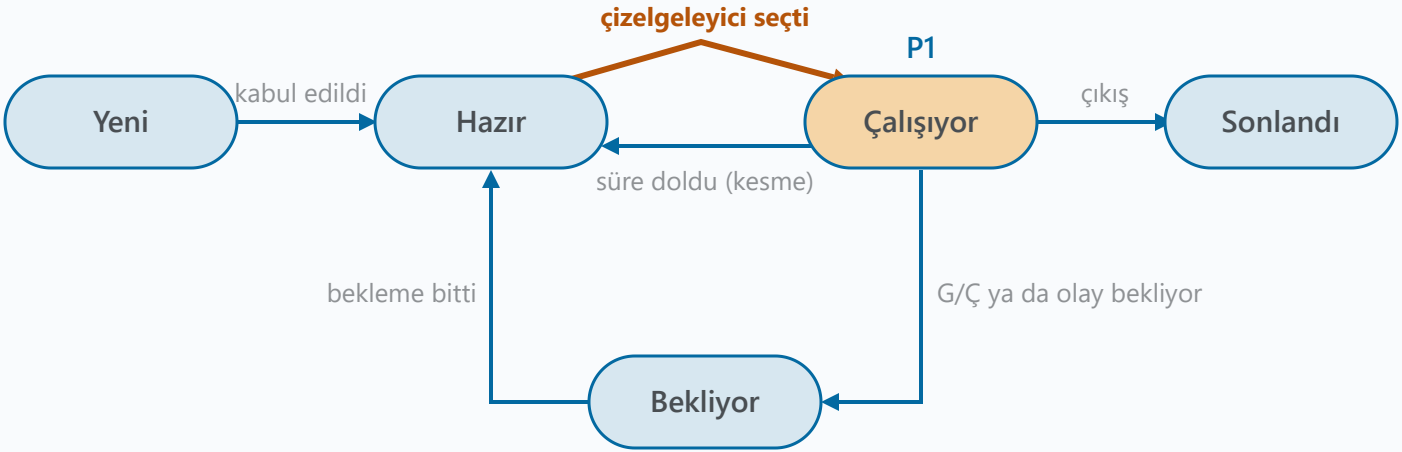
Hazır → **Çalışıyor**: Sırası gelince P1 yeniden seçilir ve kaldığı yerden devam eder.

Süreç Durum Geçişleri Adım Adım



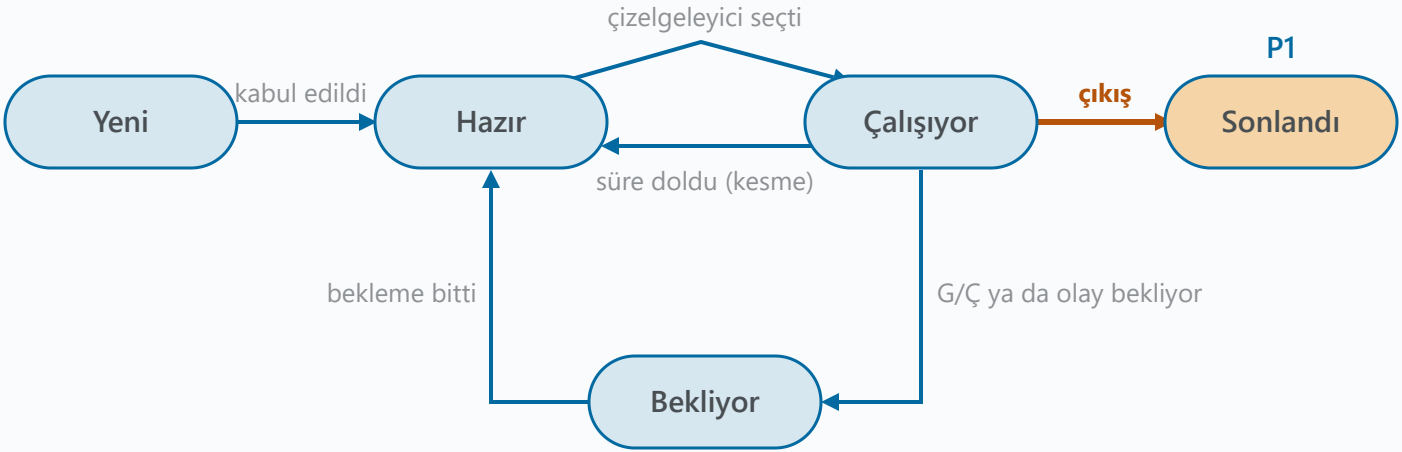
Çalışıyor → **Hazır**: P1'in **zaman dilimi doldu**; zamanlayıcı kesmesiyle CPU başka sürece verilir.

Süreç Durum Geçişleri Adım Adım



Hazır → **Çalışıyor**: P1 tekrar seçilir ve işinin kalanını yürütür.

Süreç Durum Geçişleri Adım Adım



Çalışıyor → **Sonlandı**: P1 işini bitirip **çıkar**; işletim sistemi belleğini ve kaynaklarını geri alır.

KAYNAK YÖNETİMİ · ADIM ADIM

İki Süreç, Tek CPU: Durum İzleme



İki süreç, tek çekirdek. P1 önce 3 ms hesap yapacak, sonra diskten okuyacak (3 ms), sonra 2 ms daha hesaplayacak. P2 yalnız 4 ms hesap. İkisi de **Hazır**; çizelgeleyici P1'i seçer.

Adım 1 / 8

```
1 P1: hesapla 3 ms
2 P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
4 P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 0` `P1 = Hazır` `P2 = Hazır` `CPU = boş`

İki Süreç, Tek CPU: Durum İzleme



t = 0–3: **P1 Çalışıyor**, P2 hazır kuyruğunda bekler.


```
▶ P1: hesapla 3 ms
2 P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
4 P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 3` `P1 = Çalışıyor` `P2 = Hazır` `CPU = P1`

İki Süreç, Tek CPU: Durum İzleme



t = 3: P1 diskten okuma ister (sistem çağrısı). Disk yavaştır; P1 **Bekliyor** durumuna geçer ve CPU'yu bırakır. Bağlam değiştirilir.

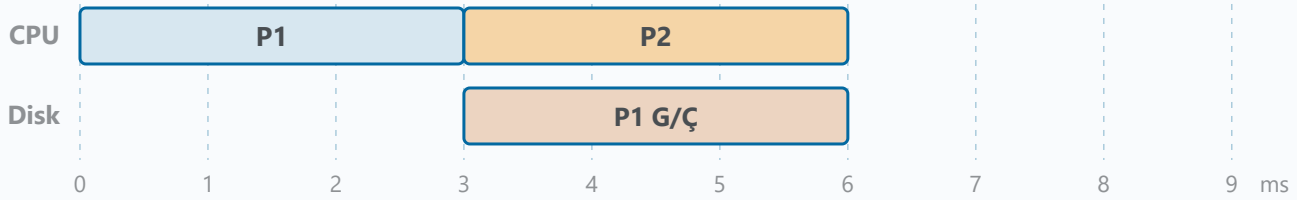
Adım 3 / 8

```
1 P1: hesapla 3 ms
▶ P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
4 P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 3` `P1 = Bekliyor` `P2 = Hazır` `CPU = boş`

İki Süreç, Tek CPU: Durum İzleme



t = 3–6: **P2 Çalışıyor**; aynı anda disk P1'in verisini okuyor. CPU ve disk **paralel** çalışıyor.

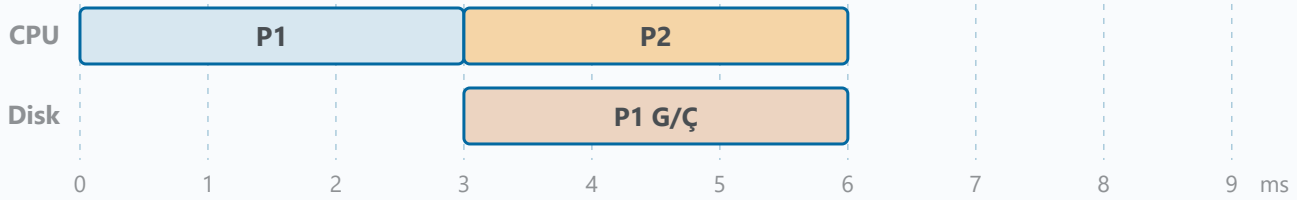
Adım 4 / 8

```
1 P1: hesapla 3 ms
2 P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
▶ P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 6` `P1 = Bekliyor` `P2 = Çalışıyor` `CPU = P2`

İki Süreç, Tek CPU: Durum İzleme



t = 6: Disk işini bitirir, **kesme** ile haber verir. P1 **Hazır** olur ama hemen çalışmaz: CPU'da P2 var.

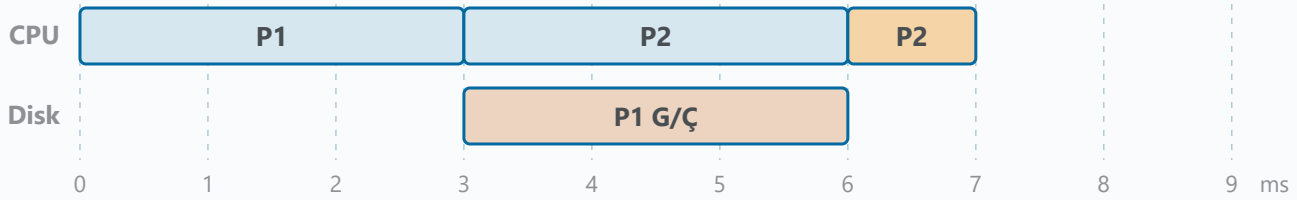
Adım 5 / 8

```
1 P1: hesapla 3 ms
▶ P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
4 P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 6` `P1 = Hazır` `P2 = Çalışıyor` `CPU = P2`

İki Süreç, Tek CPU: Durum İzleme



t = 7: P2 4 ms'sini tamamlar ve **Sonlandı**. Hazır kuyruğunda yalnız P1 kaldı.

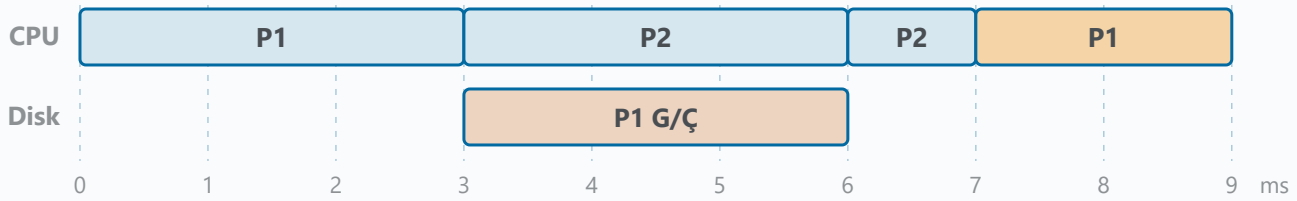
Adım 6 / 8

```
1 P1: hesapla 3 ms
2 P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
▶ P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 7` `P1 = Hazır` `P2 = Sonlandı` `CPU = boş`

İki Süreç, Tek CPU: Durum İzleme



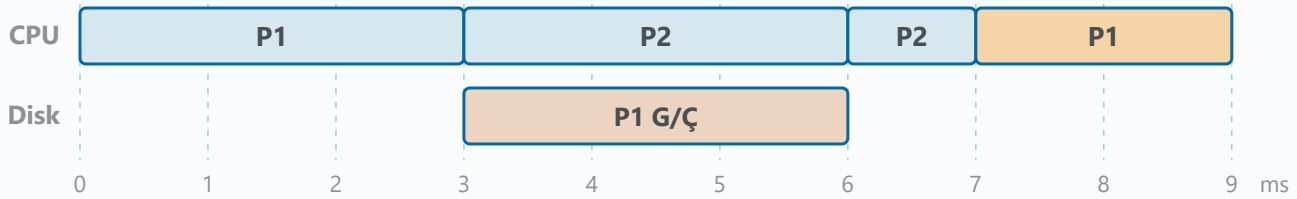
t = 7–9: **P1 Çalışıyor**, kalan 2 ms'yi bitirip **Sonlandı**.


```
1 P1: hesapla 3 ms
2 P1: diskten oku (G/Ç 3 ms)
▶ P1: hesapla 2 ms, çık
4 P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

`t = 9` `P1 = Sonlandı` `P2 = Sonlandı` `CPU = boş`

İki Süreç, Tek CPU: Durum İzleme



Toplam **9 ms**; CPU hiç boş kalmadı. P1 diski beklerken CPU boş kalsaydı 3 + 3 + 2 + 4 = **12 ms** sürerdi: G/Ç beklerken başka süreci çalıştırmak (çoklu programlama) zaman kazandırır.

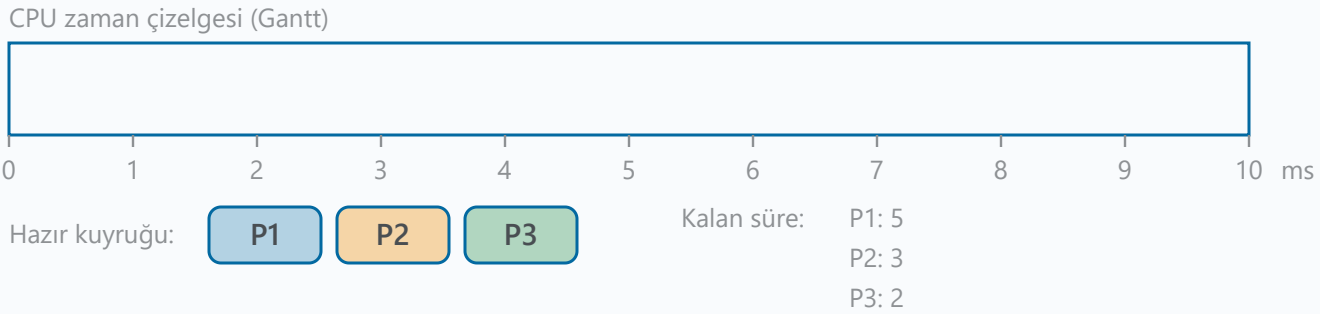
Adım 8 / 8

```
1 P1: hesapla 3 ms
2 P1: diskten oku (G/Ç 3 ms)
3 P1: hesapla 2 ms, çık
4 P2: hesapla 4 ms, çık
```

DEĞİŞKENLER

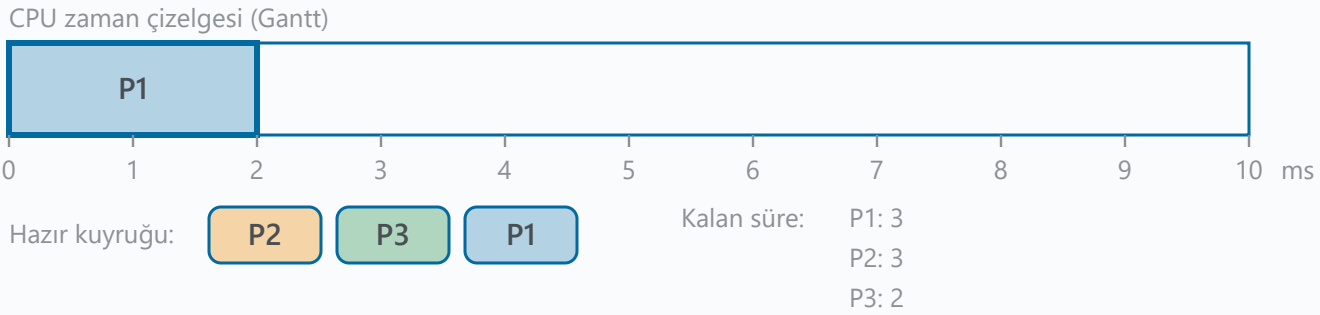
`t = 9` `P1 = Sonlandı` `P2 = Sonlandı` `CPU = boş`

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



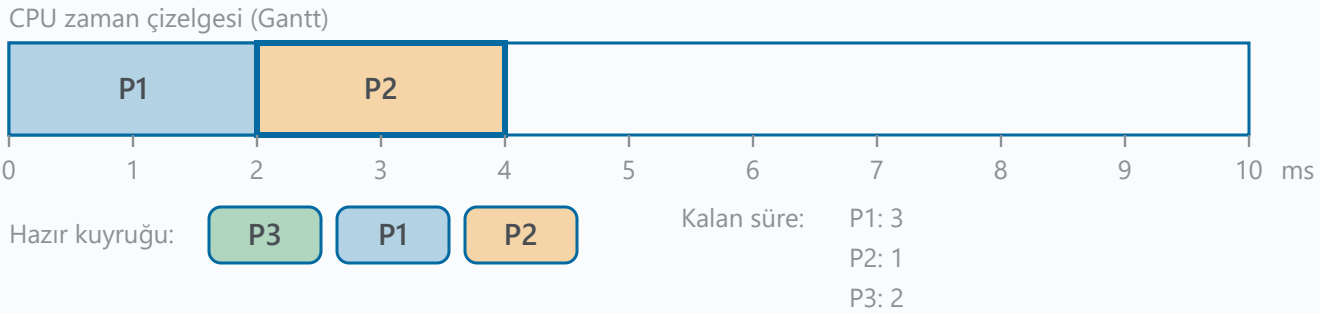
Üç süreç hazır: P1 5 ms, P2 3 ms, P3 2 ms iş istiyor. **Döner sıralı** (round robin) çizelgeleme, her birine sırayla en çok **2 ms** verir.

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



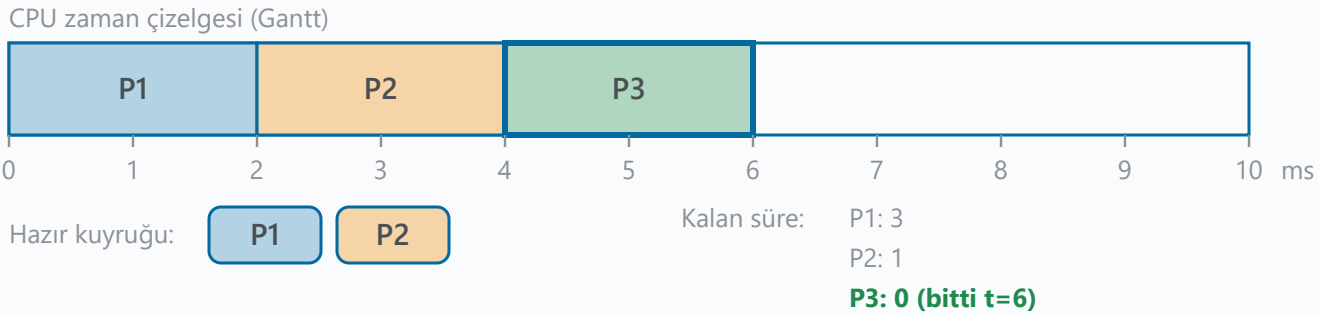
t = 0–2: CPU **P1**'de. Zaman dilimi doldu; P1 kuyruğun sonuna gider (kalan 3 ms). Bağlam değiştirilir.

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



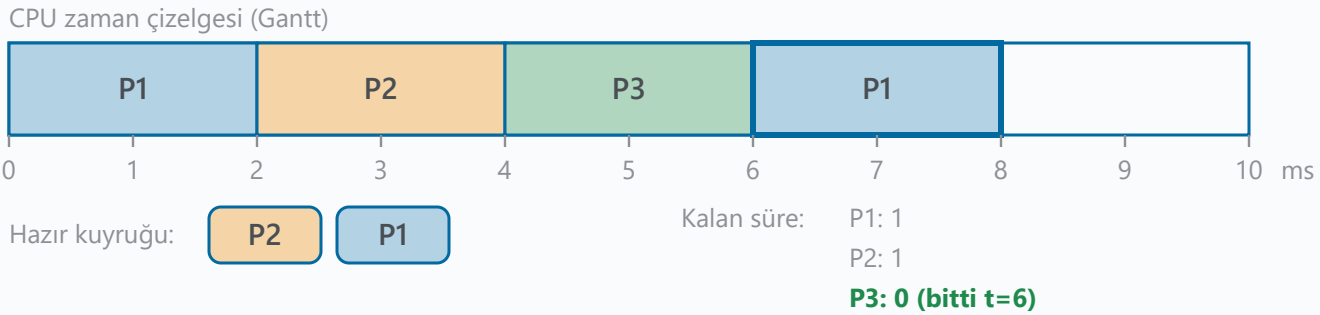
t = 2–4: CPU **P2**'de. Zaman dilimi doldu; P2 kuyruğun sonuna gider (kalan 1 ms). Bağlam değiştirilir.

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



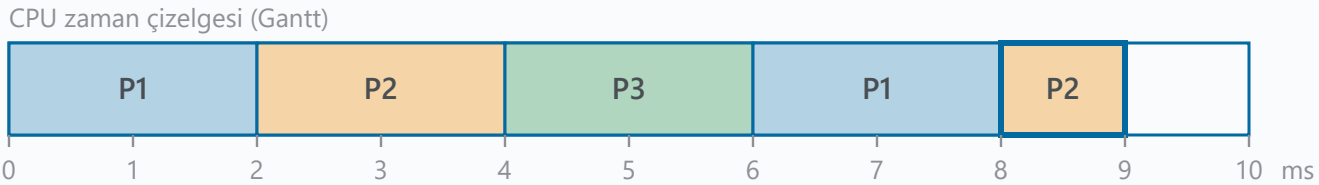
t = 4–6: CPU **P3**'te. P3 işini bitirdi (t = 6) ve sistemden çıkar.

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



t = 6–8: CPU **P1**'de. Zaman dilimi doldu; P1 kuyruğun sonuna gider (kalan 1 ms). Bağlam değiştirilir.

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



Hazır kuyruğu: **P1**

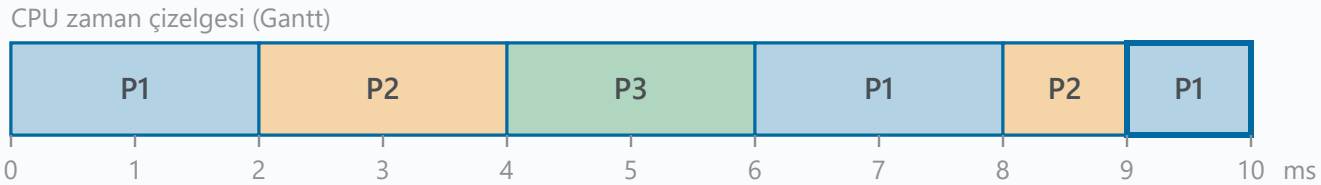
Kalan süre: P1: 1

P2: 0 (bitti t=9)

P3: 0 (bitti t=6)

t = 8–9: CPU **P2**'de. P2 işini bitirdi (t = 9) ve sistemden çıkar.

Zaman Paylaşımı: Döner Sıralı CPU Dağıtımı



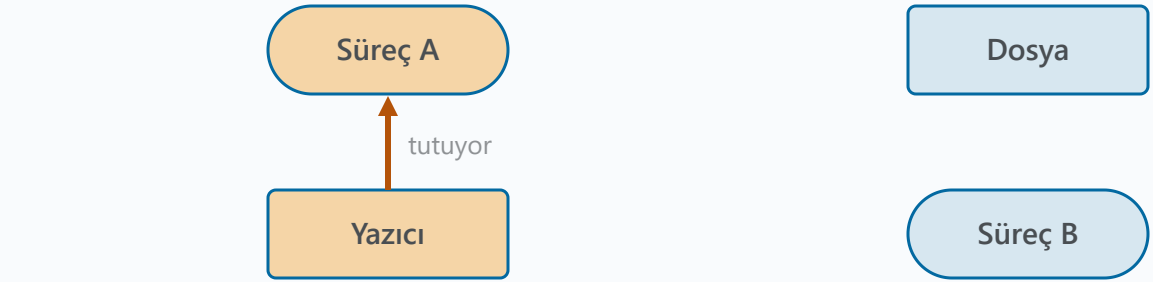
Hazır kuyruğu: boş

Kalan süre: **P1: 0 (bitti t=10)**
P2: 0 (bitti t=9)
P3: 0 (bitti t=6)

t = 9–10: CPU **P1**'de. P1 işini bitirdi (t = 10) ve sistemden çıkar. Bitiş: P3 = 6, P2 = 9, P1 = 10 ms. Hızlı geçişler sayesinde üçü de "aynı anda" çalışıyor gibi görünür.

KAYNAK YÖNETİMİ · ADIM ADIM

Kilitlenme (Deadlock)



düz ok: kaynak sürecin elinde · kesikli ok: süreç kaynağı bekliyor

Süreç A **yazıcıyı** aldı ve kullanıyor (düz ok).

Adım 1 / 5

Karşılıklı dışlama
tek sahip

Tut ve bekle
bırakmadan bekler

Ön alınamazlık
zorla alınamaz

Döngüsel bekleme
halka biçiminde

Dört koşulun **hepsi** gerekir. Çözüm fikri: kaynakları herkes aynı **sırayla** istesin (önce Yazıcı, sonra Dosya); döngüsel bekleme oluşmaz.

Kilitlenme (Deadlock)



düz ok: kaynak sürecin elinde · kesikli ok: süreç kaynağı bekliyor

Süreç B **dosyayı** kilitledi ve kullanıyor.

Karşılıklı dışlama
tek sahip

Tut ve bekle
bırakmadan bekler

Ön alınamazlık
zorla alınamaz

Döngüsel bekleme
halka biçiminde

Dört koşulun **hepsi** gerekir. Çözüm fikri: kaynakları herkes aynı **sırayla** istesin (önce Yazıcı, sonra Dosya); döngüsel bekleme oluşmaz.

Kilitlenme (Deadlock)



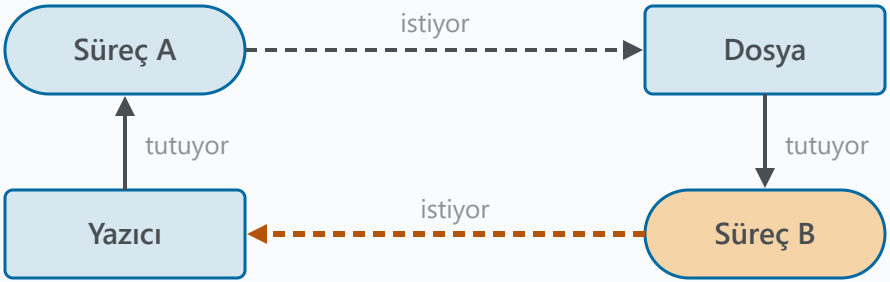
düz ok: kaynak sürecin elinde · kesikli ok: süreç kaynağı bekliyor

Süreç A, yazıcıyı bırakmadan **dosyayı** istiyor (kesikli ok); dosya B'de olduğu için bekler.

Karşılıklı dışlama	**Tut ve bekle**	**Ön alınamazlık**	**Döngüsel bekleme**
tek sahip	bırakmadan bekler	zorla alınamaz	halka biçiminde

Dört koşulun **hepsi** gerekir. Çözüm fikri: kaynakları herkes aynı **sırayla** istesin (önce Yazıcı, sonra Dosya); döngüsel bekleme oluşmaz.

Kilitlenme (Deadlock)



düz ok: kaynak sürecin elinde · kesikli ok: süreç kaynağı bekliyor

Süreç B de dosyayı bırakmadan **yazıcıyı** istiyor; yazıcı A'da olduğu için bekler.

Karşılıklı dışlama
tek sahip

Tut ve bekle
bırakmadan bekler

Ön alınamazlık
zorla alınamaz

Döngüsel bekleme
halka biçiminde

Dört koşulun **hepsi** gerekir. Çözüm fikri: kaynakları herkes aynı **sırayla** istesin (önce Yazıcı, sonra Dosya); döngüsel bekleme oluşmaz.

Kilitlenme (Deadlock)



düz ok: kaynak sürecin elinde · kesikli ok: süreç kaynağı bekliyor

A → Dosya → B → Yazıcı → A: **döngüsel bekleme**. İkisi de bırakmadığı için ikisi de sonsuza dek bekler: **kilitlenme**.

Karşılıklı dışlama
tek sahip

Tut ve bekle
bırakmadan bekler

Ön alınamazlık
zorla alınamaz

Döngüsel bekleme
halka biçiminde

Dört koşulun **hepsi** gerekir. Çözüm fikri: kaynakları herkes aynı **sırayla** istesin (önce Yazıcı, sonra Dosya); döngüsel bekleme oluşmaz.

KAYNAK YÖNETİMİ

Bellek Yönetimi

Ayırma
Her programa gereksinimine göre bellek alanı sağlama.

Koruma
Bir sürecin başka sürecin belleğini izinsiz değiştirmesini önleme.

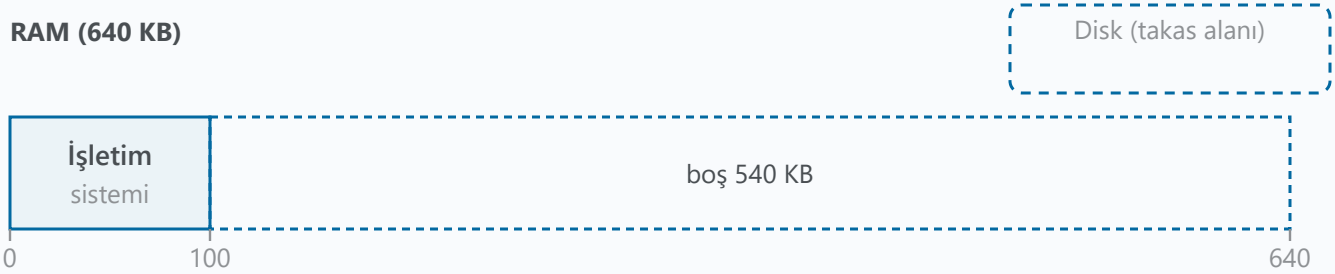
Sanal bellek
Programlara fiziksel RAM'den bağımsız adres alanı sunma.

Sayfalama
Belleği sabit boyutlu parçalarla yönetme.

Takas
Gerektiğinde bazı verileri geçici olarak diske taşıma.

Paylaşım
Ortak kod veya veriyi kontrollü biçimde kullanma.

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas

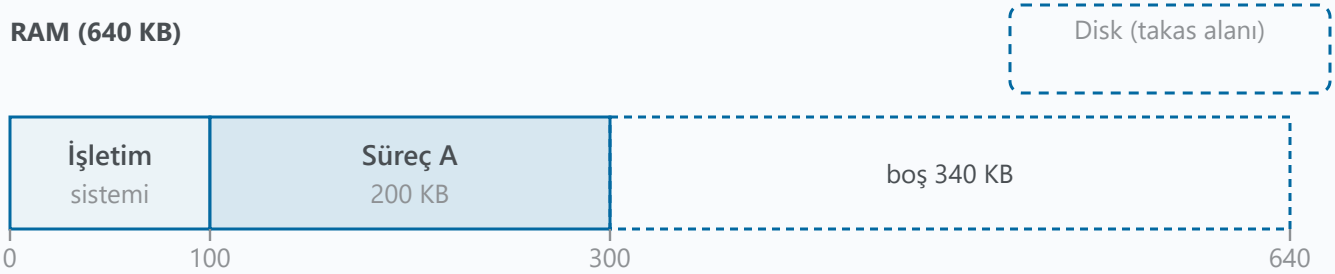


Taban / limit yazmaçları: henüz kullanıcı süreci yok

RAM'in başı çekirdeğe ayrılmıştır; kalan alan boştur.

Başlangıç: RAM'in başında **işletim sistemi (çekirdek)** vardır; geri kalanı boştur.

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas

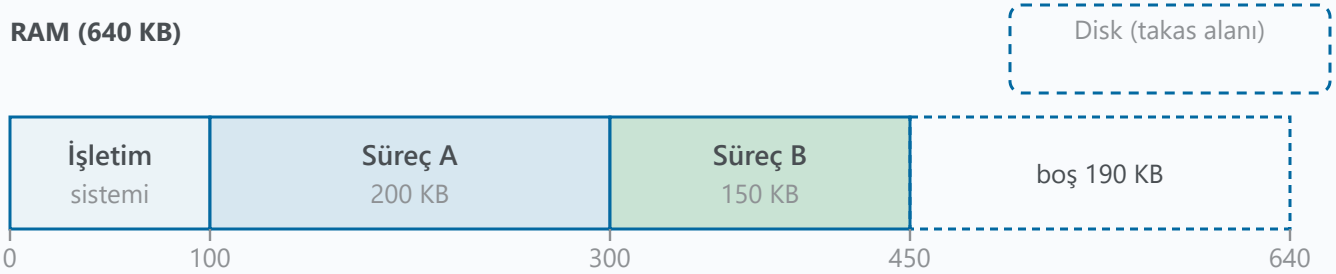


Süreç A → taban = 100, limit = 200

Taban: bölgenin başlangıcı. Limit: bölgenin uzunluğu (KB).

Ayırma: A için 100–300 arası 200 KB ayrılır. Basit modelde bölge **taban** ve **limit** yazmaçlarıyla tanımlanır (gerçekte sayfalama kullanılır).

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas

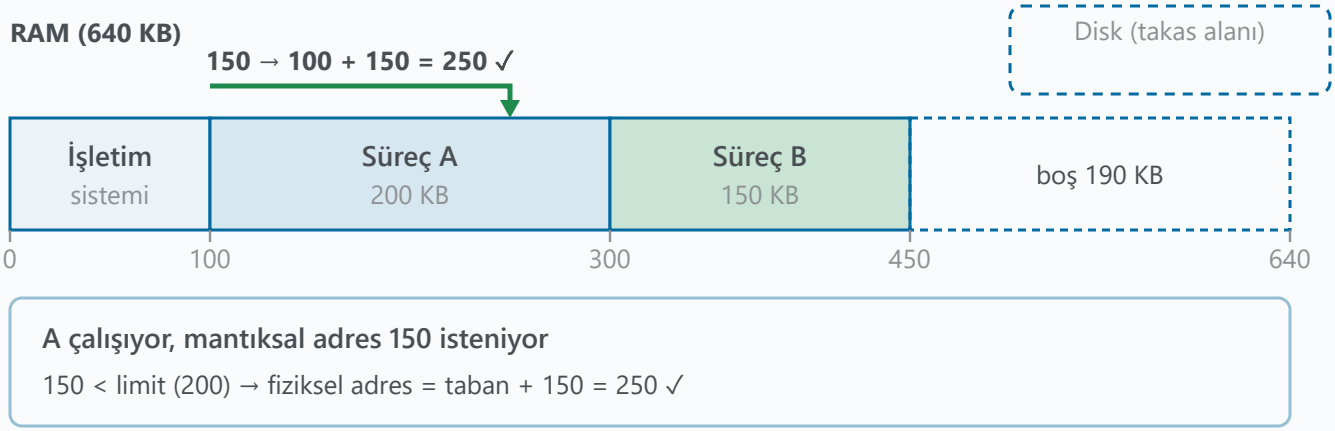


A: taban = 100, limit = 200 B: taban = 300, limit = 150

Her sürecin kendi bölgesi vardır; bölgeler çakışmaz.

B için 300–450 arası 150 KB ayrılır. Her sürecin yazmaç çifti ayrıdır; süreç değişince işletim sistemi bunları yükler.

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas

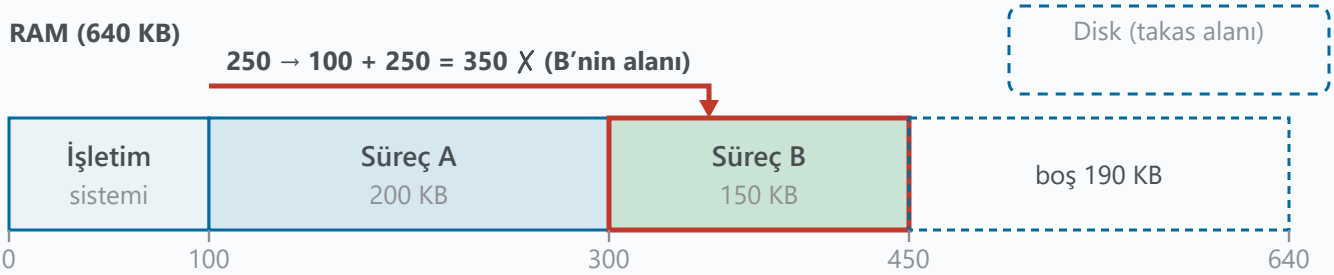


A çalışıyor, mantıksal adres 150 isteniyor

150 < limit (200) → fiziksel adres = taban + 150 = 250 ✓

Geçerli erişim: A'nın adresi 150, limitin (200) altındadır; donanım **taban + 150 = 250** fiziksel adresine gider.

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas

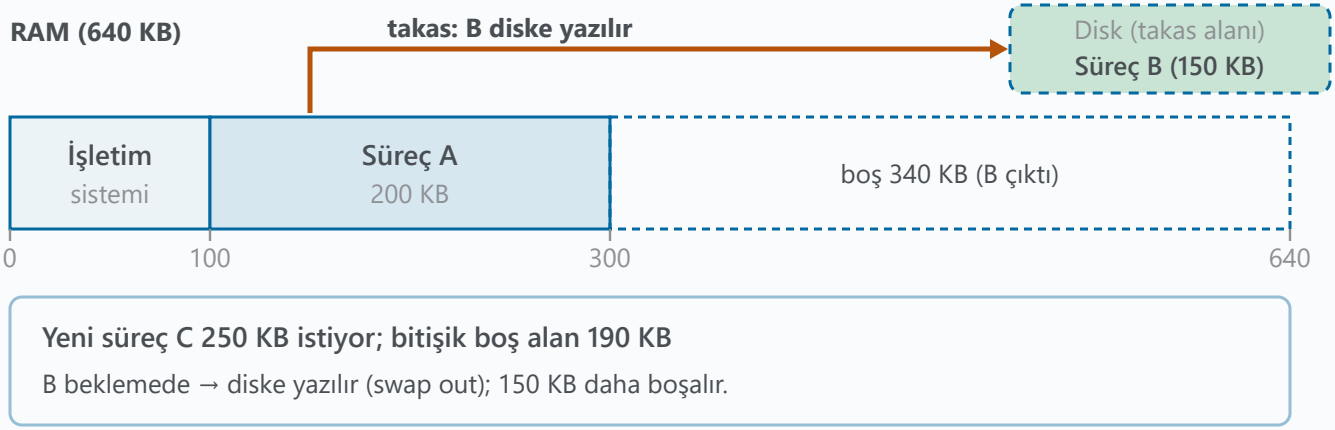


A çalışıyor, mantıksal adres 250 isteniyor

250 ≥ limit (200) → ihlal ✗: donanım kesmesi, erişim engellenir

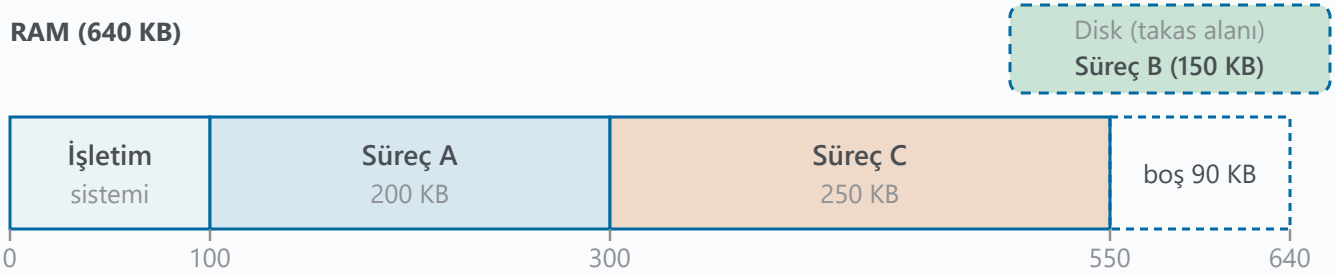
Koruma ihlali: A 250 ister, 250 ≥ 200 olduğu için donanım erişimi keser; denetim çekirdeğe geçer. Gerçek bir sistemde A "erişim ihlali / segmentation fault" ile sonlandırılır; örnekte A'nın bölgesi yerinde kalır. B'nin belleği korunur.

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas



Takas: C'ye yer yoktur (190 KB < 250 KB). Beklemedeki B diske yazılır; RAM'de 340 KB bitişik boş alan oluşur. Disk RAM'den çok yavaştır.

Bellek Yönetimi Adım Adım: Ayırma, Koruma, Takas

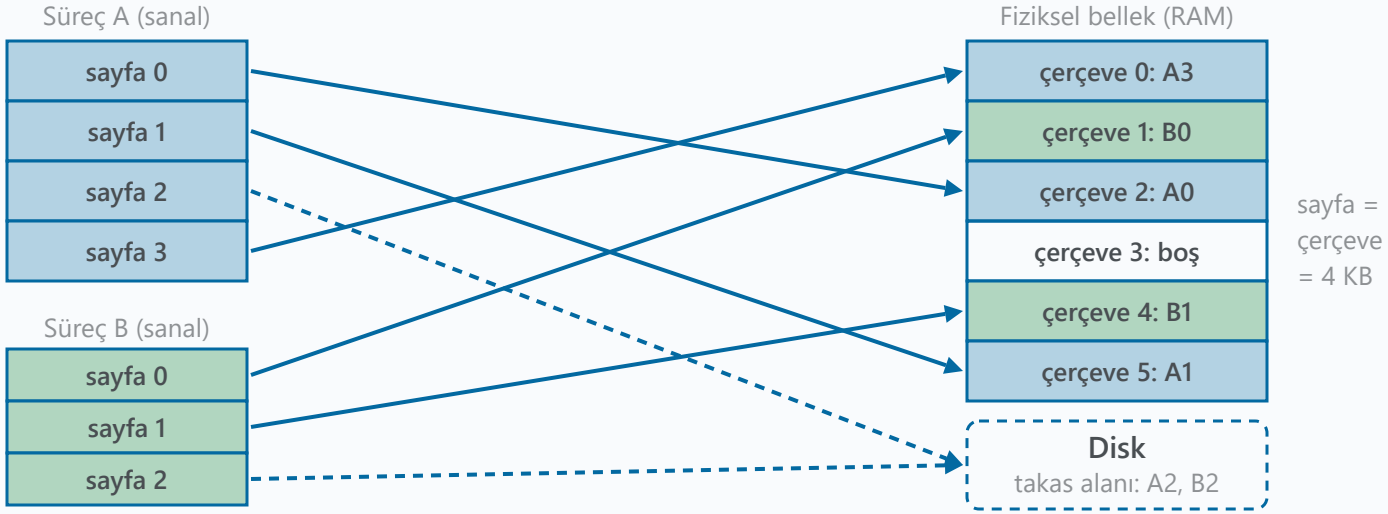


Süreç C → taban = 300, limit = 250

B diskte bekler; çalışacağı zaman geri yüklenir (swap in).

C, 300–550 arasına yüklenir. B diskte bekler ve yeniden çalışacağında RAM'e geri alınır; yer yoksa başka bir süreç diske yazılır.

Sanal Bellek ve Sayfalama



- Her süreç 0'dan başlayan kendi **sanal adres** alanını görür; bellek 4 KB'lık **sayfalara** bölünür ve sayfa tablosu her sayfanın hangi **çerçevede** olduğunu tutar.
- Sayfalar RAM'de ardışık olmak zorunda değildir; az kullanılan sayfa **diske** (takas) çıkarılabilir.
- Takas alanı Windows'ta **sayfa dosyası**, Linux ve macOS'ta **swap** adını alır. Diskten okuma (SSD ≈ 0,1 ms) RAM'den (50–100 ns) kat kat yavaştır; takas çok artarsa bilgisayar belirgin biçimde yavaşlar.

Adres Çevirisi Adım Adım: 4100 ve 8200

Sanal adres

4100

Sayfa tablosu (Süreç A)

0	çerçeve 2
1	çerçeve 5
2	diskte (yok)
3	çerçeve 0

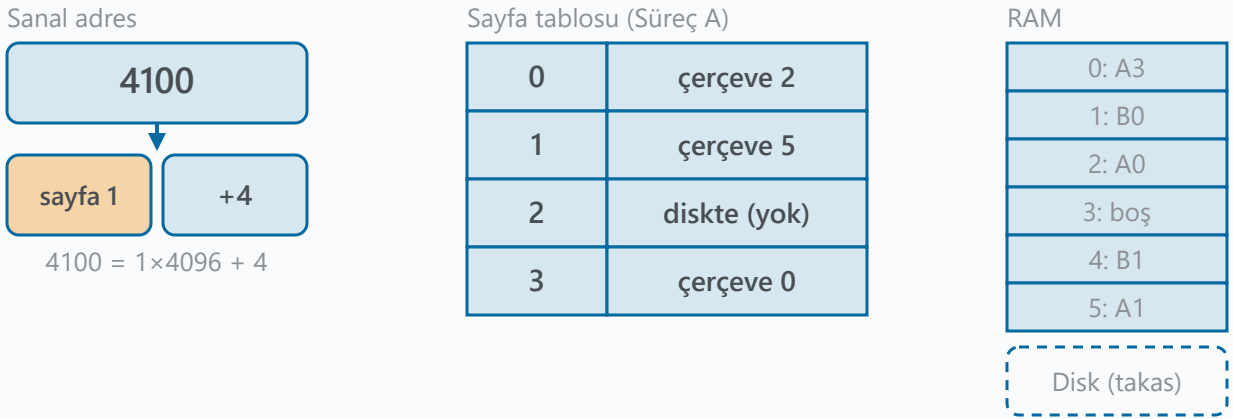
RAM

- 0: A3
- 1: B0
- 2: A0
- 3: boş
- 4: B1
- 5: A1

Disk (takas)

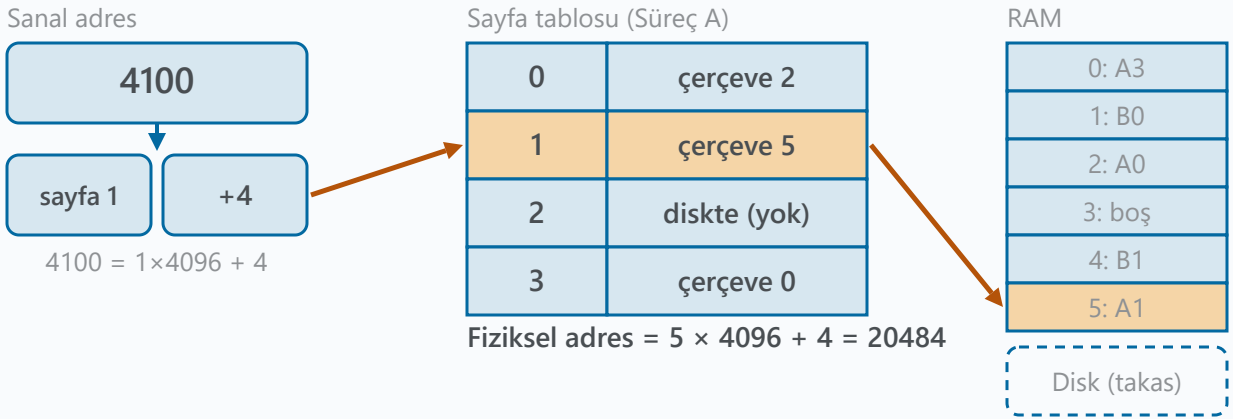
Süreç A, **4100** sanal adresindeki veriyi okumak istiyor. İşlemcideki bellek yönetim birimi (MMU) bu adresi çevirmeli.

Adres Çevirisi Adım Adım: 4100 ve 8200



Adres ikiye ayrılır: **sayfa = 4100 / 4096 = 1**, **bağıl konum = 4100 mod 4096 = 4**.

Adres Çevirisi Adım Adım: 4100 ve 8200



Tablo: sayfa 1 → çerçeve 5. **Fiziksel adres = 5 × 4096 + 4 = 20484**.

Adres Çevirisi Adım Adım: 4100 ve 8200

Sanal adres

8200

Sayfa tablosu (Süreç A)

0	çerçeve 2
1	çerçeve 5
2	diskte (yok)
3	çerçeve 0

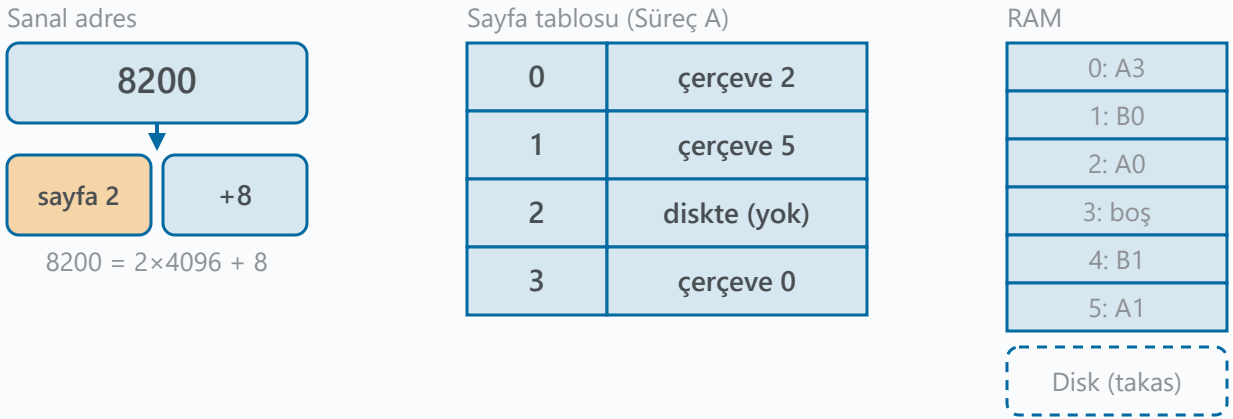
RAM

- 0: A3
- 1: B0
- 2: A0
- 3: boş
- 4: B1
- 5: A1

Disk (takas)

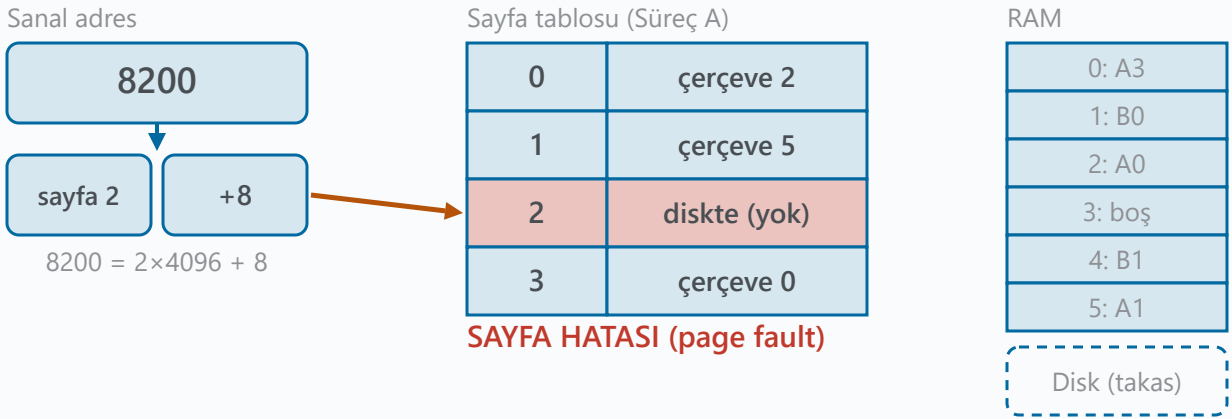
Süreç A, **8200** sanal adresindeki veriyi okumak istiyor. İşlemcideki bellek yönetim birimi (MMU) bu adresi çevirmeli.

Adres Çevirisi Adım Adım: 4100 ve 8200



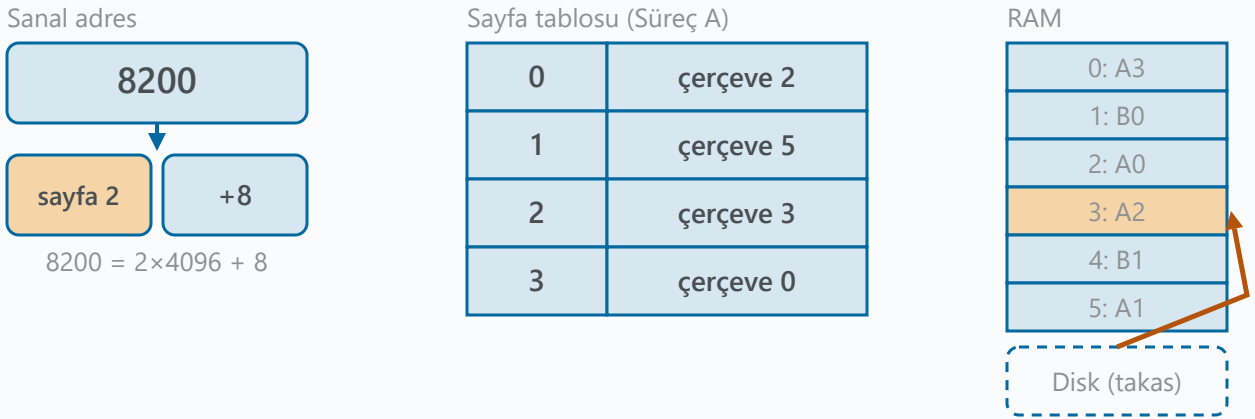
Adres ikiye ayrılır: **sayfa = 8200 / 4096 = 2**, **bağıl konum = 8200 mod 4096 = 8**.

Adres Çevirisi Adım Adım: 4100 ve 8200



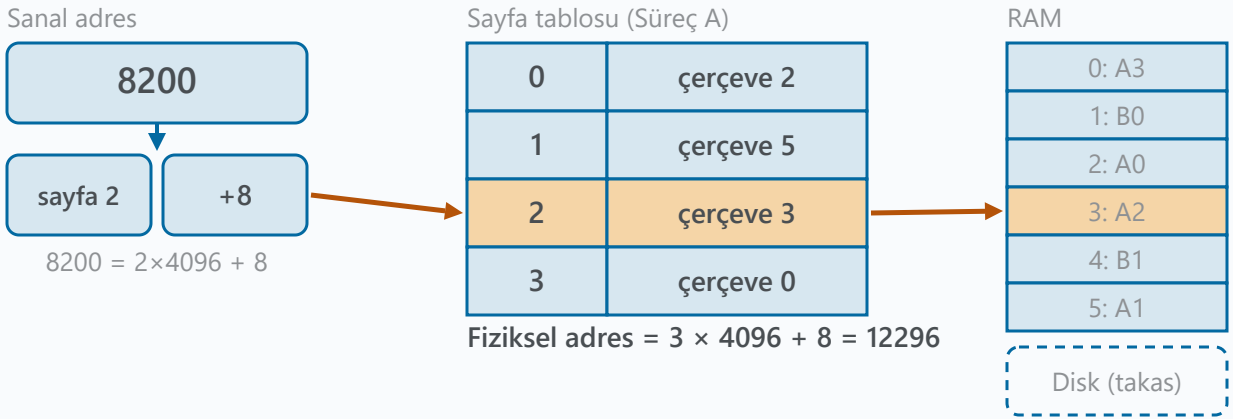
Tabloda sayfa 2 **RAM'de değil** → sayfa hatası. Program durdurulur, işletim sistemi devreye girer.

Adres Çevirisi Adım Adım: 4100 ve 8200



İşletim sistemi sayfayı diskten boş **çerçeve 3**'e yükler ve sayfa tablosunu günceller.

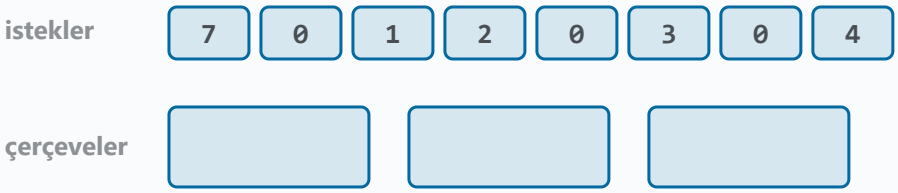
Adres Çevirisi Adım Adım: 4100 ve 8200



Tablo: sayfa 2 → çerçeve 3. **Fiziksel adres = 3 × 4096 + 8 = 12296**. Komut yeniden çalışır ve bu kez başarılı olur.

KAYNAK YÖNETİMİ · ADIM ADIM

Sayfa Yer Değiştirme (LRU) Adım Adım



Sürecin RAM'de yalnız **3 çerçevesi** var; sayfaları sırayla 7, 0, 1, 2, 0, 3, 0, 4 diye istiyor. RAM dolunca hangi sayfa diske çıkarılacak? **LRU** (en uzun süredir kullanılmayan) kuralı.

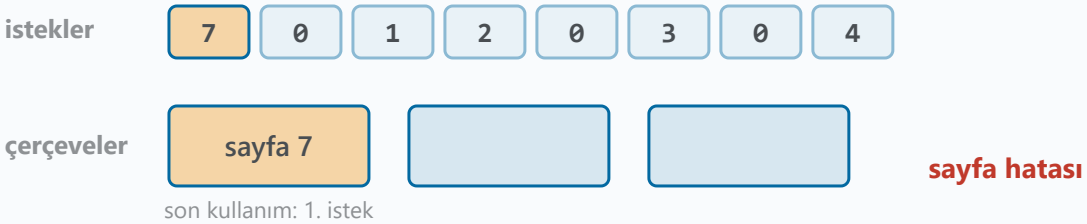
Adım 1 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

istek = **–** çerçeveler = **boş** hata = **0**

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 7 RAM'de yok → **sayfa hatası**; boş çerçeve var, diskten oraya yüklenir.

Adım 2 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
▶   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 7` `çerçeveler = 7` `hata = 1`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 0 RAM'de yok → **sayfa hatası**; boş çerçeve var, diskten oraya yüklenir.

Adım 3 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
▶   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 0` `çerçeveler = 7, 0` `hata = 2`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 1 RAM'de yok → **sayfa hatası**; boş çerçeve var, diskten oraya yüklenir.

Adım 4 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
▶   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 1` `çerçeveler = 7, 0, 1` `hata = 3`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 2 yok → **sayfa hatası**; boş çerçeve yok. En eski kullanım sayfa 7'de → o çıkarılır, 2 yüklenir.

Adım 5 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
▶   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 2` `çerçeveler = 2, 0, 1` `hata = 4`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 0 zaten RAM'de → **isabet**; yalnız son kullanım zamanı güncellenir.

Adım 6 / 10

```
▶ sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 0` `çerçeveler = 2, 0, 1` `hata = 4`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 3 yok → **sayfa hatası**; boş çerçeve yok. En eski kullanım sayfa 1'de → o çıkarılır, 3 yüklenir.

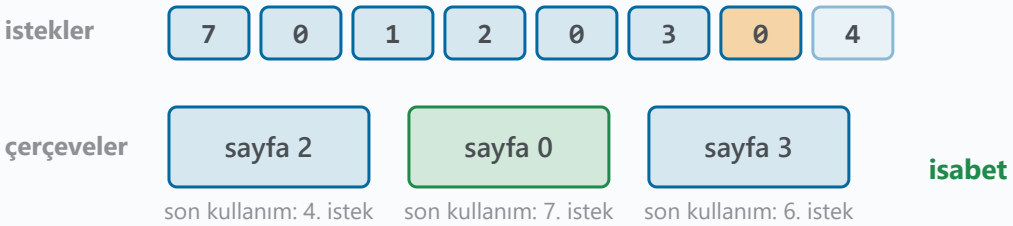
Adım 7 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
▶   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 3` `çerçeveler = 2, 0, 3` `hata = 5`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 0 zaten RAM'de → **isabet**; yalnız son kullanım zamanı güncellenir.

Adım 8 / 10

```
▶ sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 0` `çerçeveler = 2, 0, 3` `hata = 5`

Sayfa Yer Değiştirme (LRU) Adım Adım



Sayfa 4 yok → **sayfa hatası**; boş çerçeve yok. En eski kullanım sayfa 2'de → o çıkarılır, 4 yüklenir.

Adım 9 / 10

```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
▶   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = 4` `çerçeveler = 4, 0, 3` `hata = 6`

Sayfa Yer Değiştirme (LRU) Adım Adım



8 istekte **6 sayfa hatası**. Aynı dizide FIFO (ilk gelen ilk çıkar) 7 hata verir: FIFO, sık kullanılan sayfa 0'ı da sırası gelince çıkarır. Her hata yavaş bir disk okuması demektir.

Adım 10 / 10

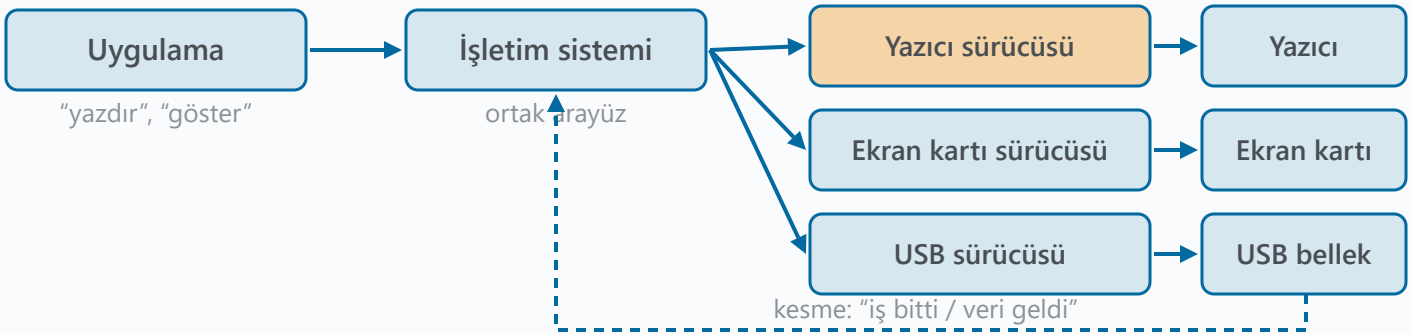
```
1 sayfa RAM'de mi?  Evet → isabet
2 Hayır → sayfa hatası
3   boş çerçeve var mı?  Evet → yükle
4   Hayır → en eski kullanılanı çıkar, yükle
```

DEĞİŞKENLER

`istek = –` `çerçeveler = 4, 0, 3` `hata = 6`

Aygıtlar ve Sürücüler

Donanım aygıtlarının komutları ve özellikleri farklıdır. Aygıt sürücüsü, işletim sisteminin ortak isteklerini donanıma özgü işlemlere çevirir.



Tak ve çalıştır

Uyumlu aygıtı algılayıp uygun sürücüyü yapılandırma.

Kesme

Aygıtın işlemciye olay bildirmesi.

Tampon

Hızları farklı bileşenler arasında geçici veri tutma.

KAYNAK YÖNETİMİ

Dosya Sistemi

Dosya sistemi, verinin depolama aygıtında nasıl adlandırıldığını, düzenlendiğini ve bulunduğunu tanımlar.

Dosya

Adı ve içeriği olan veri birimi.

Klasör

Dosya ve alt klasörleri gruplar.

Yol

Dosyanın klasör yapısındaki konumunu belirtir.

Meta veri

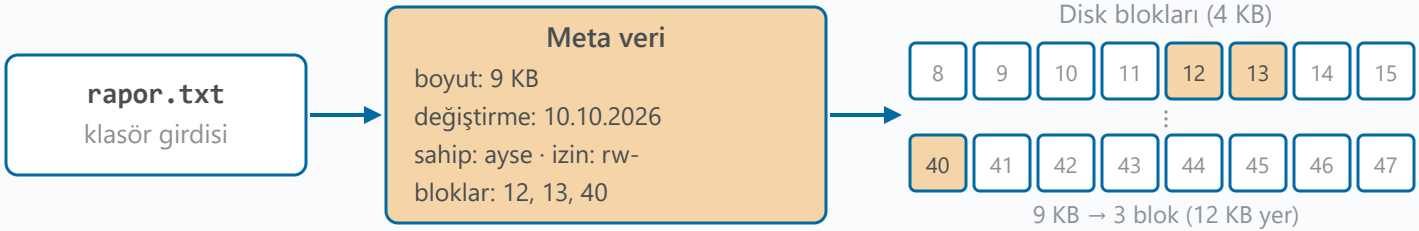
Boyut, tarih, sahip ve izin gibi özellikler.

İzinler

Okuma, yazma ve çalıştırma yetkileri.

Günlükleme (journaling)

Beklenmeyen kesintilerden sonra tutarlılığı destekler.

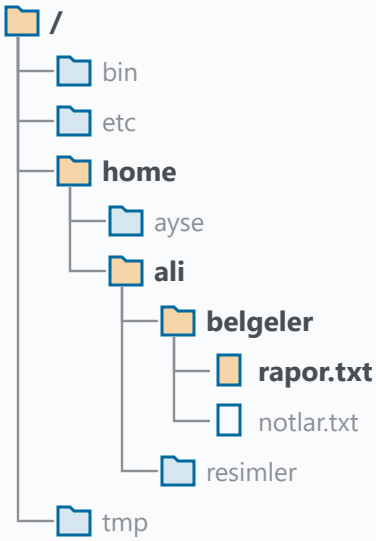


Dosya Sistemi Türleri

Dosya sistemi	Platform	Günlükleme	Ne zaman kullanılır?
NTFS	Windows (macOS yalnızca okur)	Var	Windows sistem diski; büyük dosya, izin ve şifreleme gerektiğinde
FAT32	Hemen her aygıt ve işletim sistemi	Yok	Eski aygıtlar, kamera, basit bellek kartı; **tek dosya en çok 4 GB**
exFAT	Windows, macOS ve güncel Linux	Yok	Büyük dosyalı USB bellek ve SD kart; platformlar arası taşıma
ext4	Linux	Var	Linux sistem ve veri diski
APFS	macOS, iOS, iPadOS	Klasik günlük yok; yazınca kopyalama ile tutarlılık	Apple cihazlarının varsayılan dosya sistemi

Örnek: 5 GB'lık bir video dosyası FAT32 biçimli USB belleğe kopyalanamaz; bellek exFAT veya NTFS'ye biçimlendirilmelidir (biçimlendirme içeriği siler).

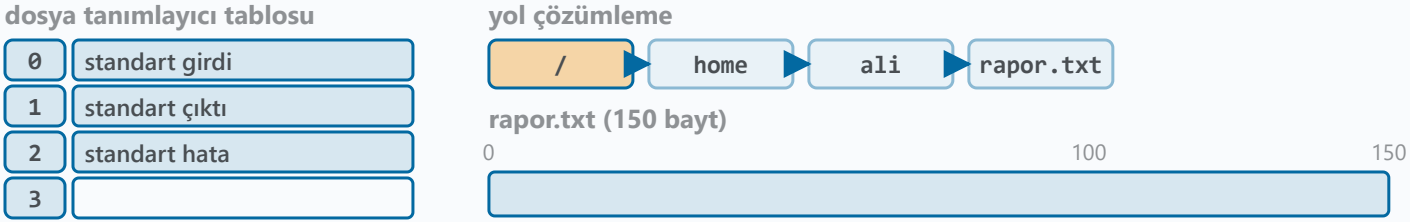
Dosya Sistemi Ağacı ve Yol




```
/home/ali/belgeler/rapor.txt
```

- Dosya sistemi ağaç biçimindedir: **kök** en üstte, klasörler dallar, dosyalar yapraklardır.
- **Mutlak yol** kökten başlar; **göreli yol** geçerli klasöre göredir (ali'deyken: `belgeler/rapor.txt`).
- Windows'ta aynı fikir sürücü harfiyle yazılır: `C:\Users\Ali\Documents\rapor.txt`.

Dosya Açma ve Okuma Adım Adım



Editör **open** sistem çağrısı yapar. Çekirdek yolu parça parça çözer: önce **kök** klasör /.

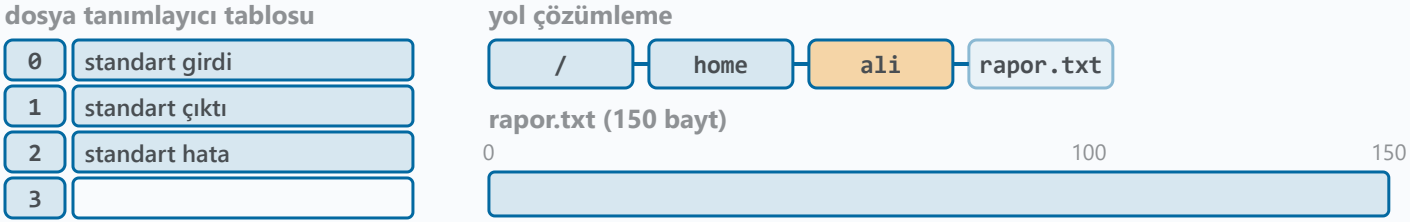
Adım 1 / 8

```
▶ fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = –` `konum = –` `n = –`

Dosya Açma ve Okuma Adım Adım



Kökte **home**, onun içinde **ali** klasörü aranır. Her klasör, içindeki adları ve bunların diskteki yerini tutan bir listedir.

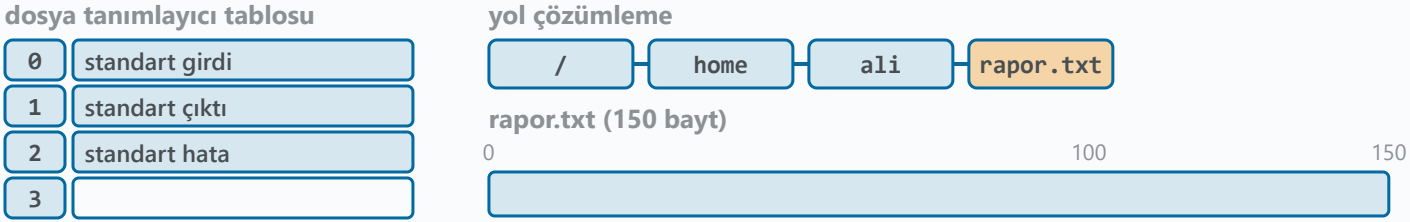
Adım 2 / 8

```
▶ fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = –` `konum = –` `n = –`

Dosya Açma ve Okuma Adım Adım



ali içinde **rapor.txt** bulunur. Meta veriye bakılır: boyut 150 bayt; kullanıcının **okuma izni** var mı? Var. (Yoksa open hata döner, dosya açılmaz.)

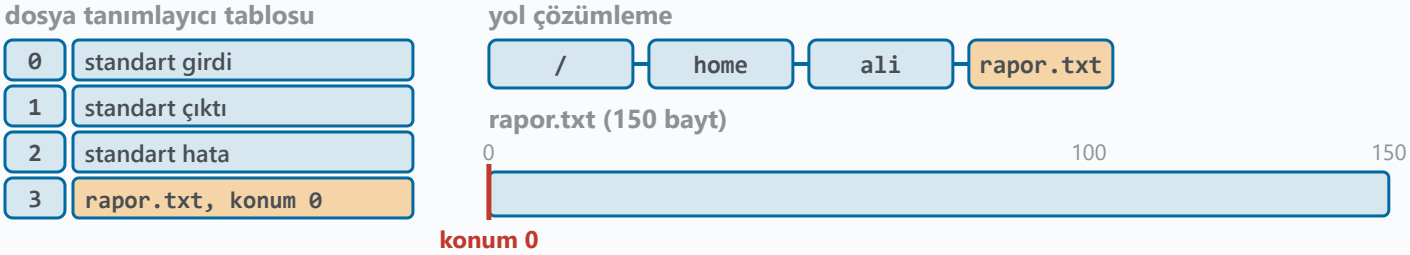
Adım 3 / 8

```
▶ fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = –` `konum = –` `n = –`

Dosya Açma ve Okuma Adım Adım



Çekirdek açık dosya kaydını oluşturur ve sürece en küçük boş numarayı verir: **fd = 3**. 0, 1 ve 2 her süreçte standart girdi, çıktı ve hata için açıktır.

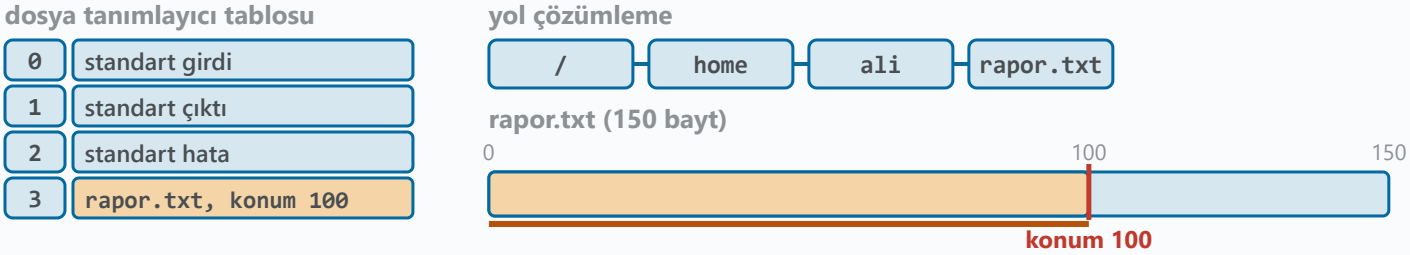
Adım 4 / 8

```
▶ fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = 3` `konum = 0` `n = –`

Dosya Açma ve Okuma Adım Adım



read 100 bayt ister; 0–99 arası okunur, **n = 100**. Konum kendiliğinden 100'e ilerler; uygulama adresi bilmez, yalnız fd'yi verir.

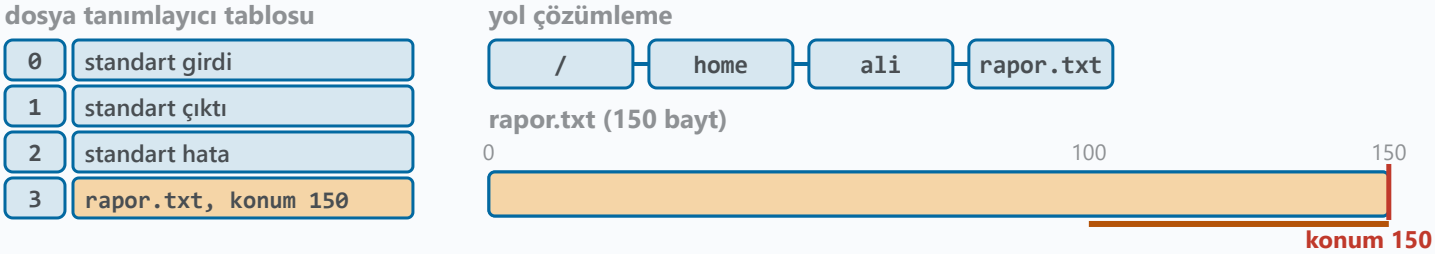
Adım 5 / 8

```
1 fd = open("/home/ali/rapor.txt", OKU)
▶ n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = 3` `konum = 100` `n = 100`

Dosya Açma ve Okuma Adım Adım



Yine 100 bayt istenir ama yalnız 50 bayt kaldı: **n = 50**, konum 150.

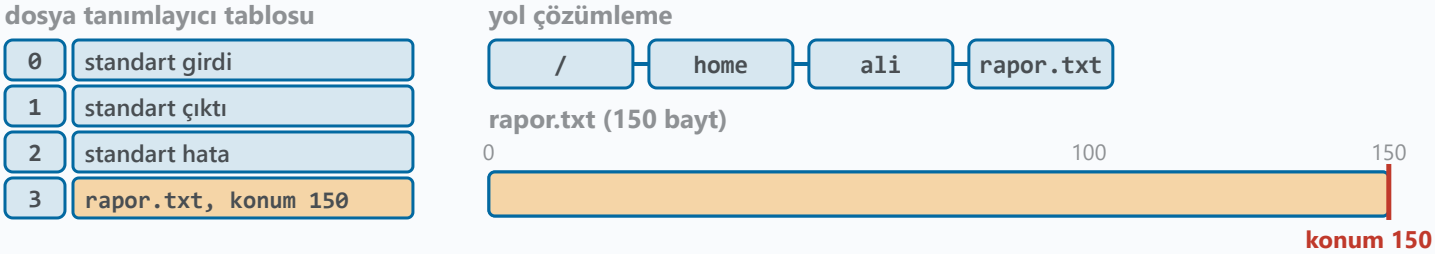
Adım 6 / 8

```
1 fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
▶ n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = 3` `konum = 150` `n = 50`

Dosya Açma ve Okuma Adım Adım



Üçüncü read **n = 0** döndürür: dosya sonu. Program okumayı burada bitirir.

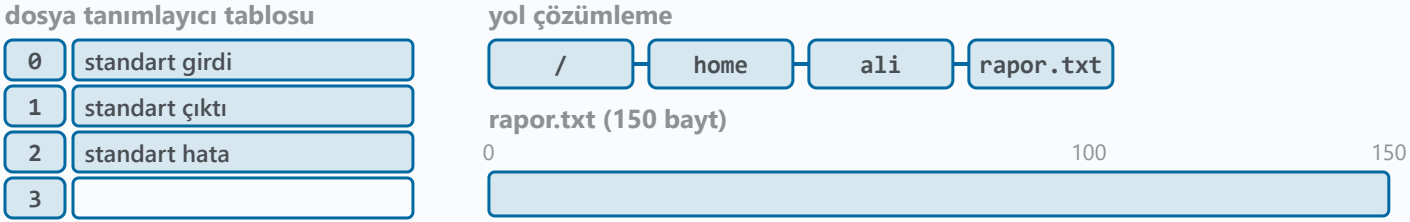
Adım 7 / 8

```
1 fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
▶ n = read(fd, tampon, 100)
5 close(fd)
```

DEĞİŞKENLER

`fd = 3` `konum = 150` `n = 0`

Dosya Açma ve Okuma Adım Adım



close: kayıt silinir, 3 numarası yeniden boşalır. Kapatılmayan dosyalar süreç bitince işletim sistemince kapatılır; ama uzun çalışan programda tanımlayıcılar tükenebilir. Windows'ta aynı fikrin adı tanıtıcıdır (handle).

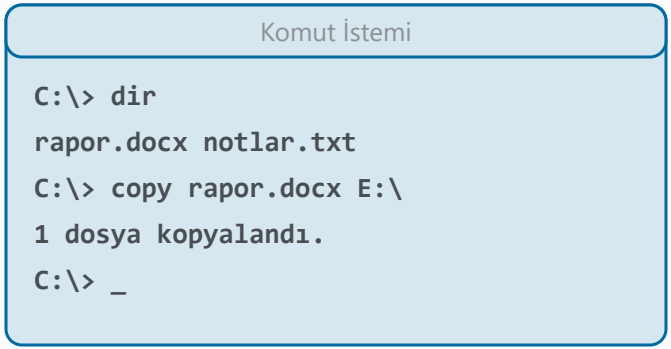
Adım 8 / 8

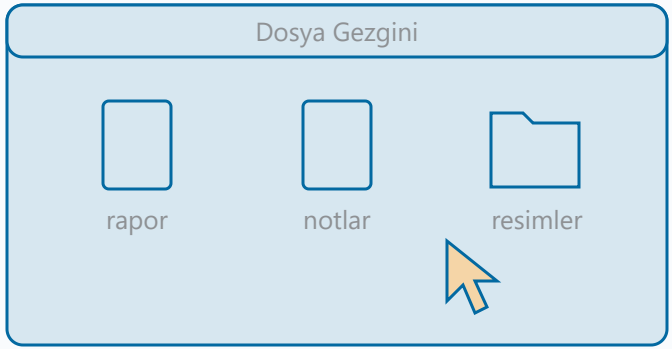
```
1 fd = open("/home/ali/rapor.txt", OKU)
2 n = read(fd, tampon, 100)
3 n = read(fd, tampon, 100)
4 n = read(fd, tampon, 100)
▶ close(fd)
```

DEĞİŞKENLER

`fd = –` `konum = –` `n = 0`

Kullanıcı Arayüzleri





GUI (grafik kullanıcı arayüzü)

Pencere, simge, menü ve fareyle etkileşim.

CLI

Metin komutlarıyla hızlı, tekrarlanabilir, otomasyona uygun.

Dokunmatik arayüz

Dokunma, kaydırma ve hareketlerle etkileşim.

Erişilebilirlik

Ekran okuyucu, büyütme, yüksek kontrast.

Komut Satırı Temelleri

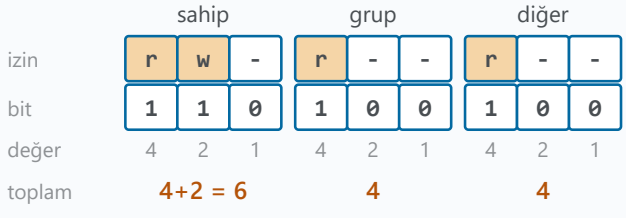
İş	Linux / macOS	Windows (cmd)
Geçerli klasörü göster	`pwd`	`cd` (parametresiz)
Klasör içeriğini listele	`ls`	`dir`
Klasör değiştir	`cd klasör`	`cd klasör`
Klasör oluştur	`mkdir yeni`	`mkdir yeni`
Dosya kopyala	`cp a.txt b.txt`	`copy a.txt b.txt`
Dosya sil	`rm a.txt`	`del a.txt`

chmod izin biçimi

Üç grup için **r**=4 (oku), **w**=2 (yaz), **x**=1 (çalıştır) toplanır: sahip · grup · diğerleri.

`chmod 644 dosya`

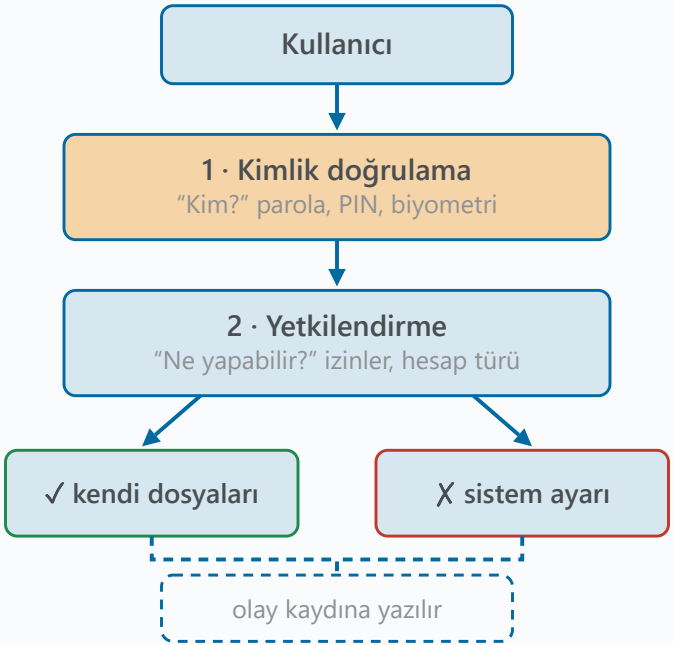
6 = rw- (sahip) · 4 = r-- (grup) · 4 = r-- (diğer)



`rm` ve `del` dosyayı Geri Dönüşüm Kutusu'na taşımaz; komutu çalıştırmadan önce yolu ve adı kontrol edin.

GÜVENLİK

Kullanıcı ve Yetki Yönetimi



- **Standart hesap** günlük kullanımda sistem değişikliklerini sınırlar; **yönetici hesabı** sistem genelinde değişiklik yapabilir.
- **En az ayrıcalık:** kullanıcıya yalnızca işi için gerekli yetki verilir.
- **Kayıt tutma**, önemli olayların sonradan incelenebilmesini sağlar.
- Örnekte standart kullanıcı kendi dosyasını açabilir, sistem ayarını değiştiremez.

İŞLETİM SİSTEMİ TÜRLERİ

Kişisel Bilgisayar Sistemleri

Windows

Geniş donanım ve uygulama ekosistemi.

macOS

Apple donanımıyla bütünleşik masaüstü sistemi.

Linux dağıtımları

Farklı masaüstü ve kullanım amaçlarına göre özelleştirilebilir sistemler.

ChromeOS

Web ve bulut tabanlı kullanıma odaklanan sistem.

Mobil İşletim Sistemleri

Android

Farklı üreticilerin cihazlarında çalışan geniş mobil ekosistem.

iOS / iPadOS

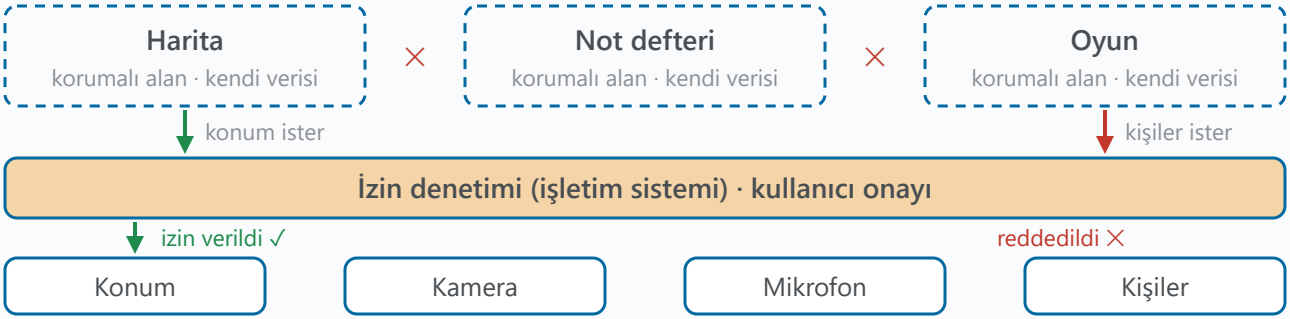
Apple mobil cihazları için bütünleşik sistem.

Mobil öncelikler

Dokunmatik arayüz, enerji yönetimi, sensörler ve kablosuz bağlantı.

Uygulama yalıtımı

Uygulamaların erişimini izinler ve korumalı alanlarla sınırlar.



İŞLETİM SİSTEMİ TÜRLERİ

Sunucu İşletim Sistemleri

Çok kullanıcılı çalışma
Çok sayıda kullanıcı ve hizmeti aynı anda yönetir.

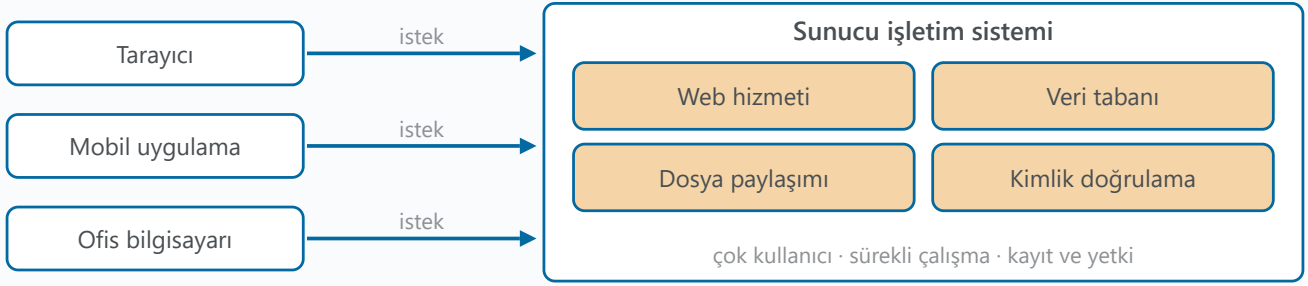
Ağ hizmetleri
Web, veri tabanı, dosya ve kimlik hizmetleri sunar.

Süreklilik
Uzun süre kesintisiz çalışma ve uzaktan yönetim hedeflenir.

Güvenlik
Yetki, kayıt, güncelleme ve ağ denetimi ön plandadır.

Örnekler
Linux dağıtımları, Windows Server ve Unix türevleri.

Otomasyon
Komut satırı ve yapılandırma araçlarıyla tekrarlanabilir yönetim.



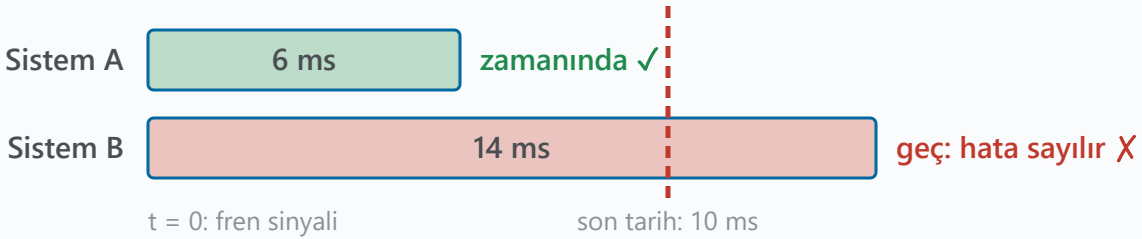
Gömülü ve Gerçek Zamanlı Sistemler

Gömülü işletim sistemi

Belirli bir cihaz görevi için sınırlı işlemci, bellek ve enerjiyle çalışır.
Örnek: yönlendirici, otomobil kontrol birimi, tıbbi cihaz.

Gerçek zamanlı işletim sistemi

Bir işlemin yalnızca doğru sonucu değil, belirlenen süre içinde sonucu üretmesini hedefler.
Örnek: endüstriyel kontrol ve güvenlik sistemleri.



İşletim Sistemi Seçimi

Gereksinim	Değerlendirilecek ölçüt
Uygulama	Gerekli yazılımların ve sürümlerin bulunması
Donanım	Sürücü ve mimari desteği
Güvenlik	Güncelleme süresi, yetki modeli ve yönetim araçları
Kullanılabilirlik	Arayüz, erişilebilirlik ve öğrenme gereksinimi
Maliyet	Lisans, destek, eğitim ve bakım toplamı

LİSANSLAMA

Açık Kaynak Yazılım

Kaynak kodu

Lisans koşulları çerçevesinde incelenebilir.

Değiştirme

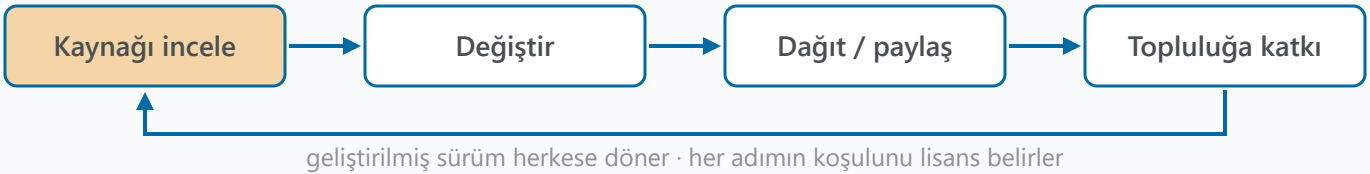
Lisans izin veriyorsa ihtiyaçlara göre uyarlanabilir.

Dağıtım

Değiştirilmiş veya özgün sürümün paylaşımı lisansa bağlıdır.

Topluluk

Geliştirme ve destek bireyler, kurumlar ve şirketler tarafından yürütülebilir.



Açık kaynak, otomatik olarak ücretsiz veya desteksiz demek değildir. Hakları ve yükümlülükleri lisans belirler.

Ticari ve Kapalı Kaynak Yazılım

Kullanım hakkı

Yazılım satın alınmaz; çoğunlukla belirli koşullarla kullanım lisansı alınır.

Kaynak kodu

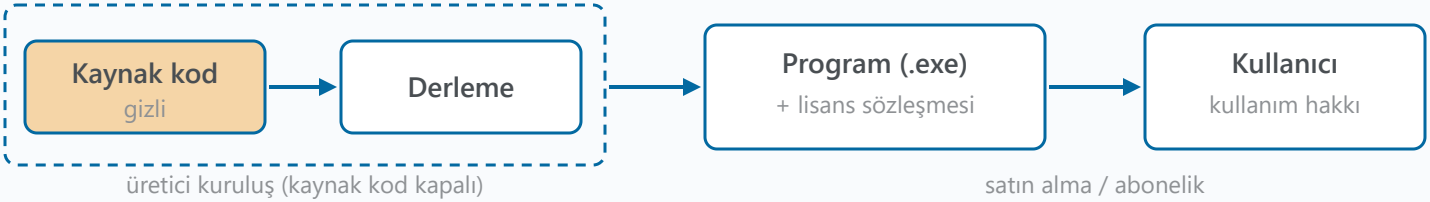
Genellikle geliştirici kuruluş tarafından kapalı tutulur.

Destek

Resmî destek ve hizmet düzeyi sözleşmeleri sunulabilir.

Maliyet

Lisans, abonelik, bakım ve yükseltme ücretleri olabilir.



Lisans Modellerini Karşılaştırma

Ölçüt	Açık kaynak	Ticari / kapalı kaynak
Kaynak erişimi	Lisansla açık	Genellikle kapalı
Özelleştirme	Lisans kapsamında mümkün	Üreticinin sunduğu seçeneklerle sınırlı
Destek	Topluluk veya ücretli sağlayıcı	Üretici veya yetkili ortak
Maliyet	Lisans ücretsiz olabilir; işletme maliyeti devam eder	Lisans veya abonelik gerekebilir
Lisans örneği	GPL, MIT, Apache	Son kullanıcı lisans sözleşmesi
Ürün örneği	Linux, Firefox, LibreOffice	Windows, Microsoft Office, Adobe Photoshop

Ücretsiz dağıtılan her program da açık kaynak değildir; belirleyici olan lisans koşullarıdır.

BAKIM

Güncelleme ve Destek

Güvenlik güncellemesi
Bilinen açıkları azaltır.

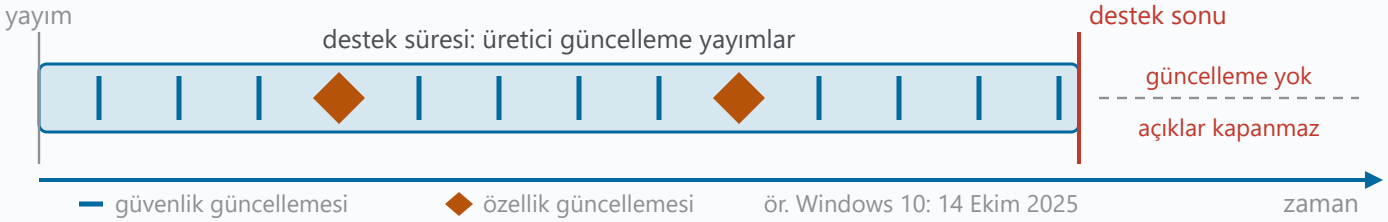
Özellik güncellemesi
Yeni işlev ve arayüz değişiklikleri getirir.

Sürücü güncellemesi
Donanım uyumluluğu veya kararlılığını geliştirebilir.

Destek süresi
Üreticinin güncelleme sağladığı zaman aralığıdır.

Yedekleme
Büyük değişikliklerden önce geri dönüş imkanı sağlar.

Test
Kurumsal ortamlarda güncellemeler yaygın dağıtımdan önce denenir.



İŞLETİM SİSTEMİ

Sanallaştırma ve Konteynerler

Sanal makine

Bir fiziksel bilgisayarda yalıtılmış işletim sistemi örnekleri çalıştırır.

Hipervizör

Sanal makinelere işlemci, bellek ve aygıt kaynakları sağlar.

Konteyner

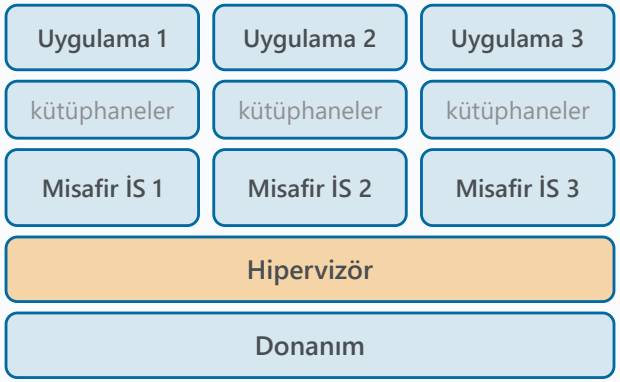
Uygulamaları ve bağımlılıklarını süreç düzeyinde yalıtır; çekirdeği paylaşır.

Kullanım

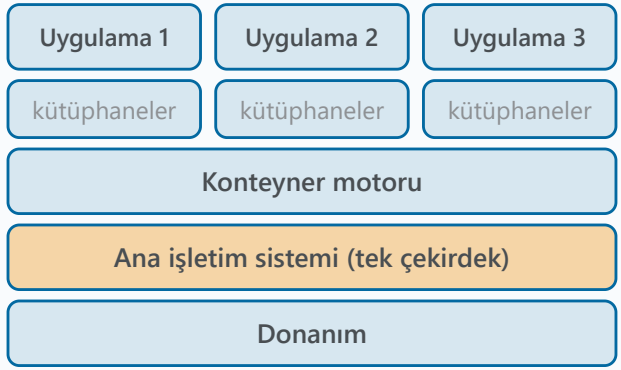
Test, sunucu birleştirme, bulut ve taşınabilir uygulama dağıtımı.

Sanal Makine ile Konteyner Karşılaştırması

Sanal makineler



Konteynerler



Sanal makine

Tam bir misafir işletim sistemi taşır: güçlü yalıtım, ama daha çok bellek ve daha yavaş açılış.

Konteyner

Yalnızca uygulama ve bağımlılıkları; çekirdek ortak: hafif ve hızlı, yalıtım daha zayıf.

ÖZET

Bölüm Özeti

- Yazılım, donanımın gerçekleştireceği işlemleri tanımlar.
- İşletim sistemi uygulamalarla donanım arasında yönetim ve soyutlama katmanı oluşturur.
- Çekirdek süreç, bellek, aygıt ve dosya kaynaklarını yönetir.
- Kişisel, mobil, sunucu, gömülü ve gerçek zamanlı sistemlerin öncelikleri farklıdır.
- İşletim sistemi seçiminde uygulama, donanım, güvenlik, destek ve toplam maliyet birlikte değerlendirilir.
- Açık ve kapalı kaynak haklarını kullanılan lisans belirler.

Kontrol Soruları

1. Sistem yazılımı ile uygulama yazılımı arasındaki fark nedir?
2. Çekirdek neden ayrıcalıklı modda çalışır?
3. Program ve süreç kavramları nasıl ayrılır?
4. Sanal bellek hangi problemi çözer?
5. Sunucu ve mobil işletim sistemlerinin öncelikleri nasıl değişir?
6. Açık kaynak yazılım neden her zaman ücretsiz olmak zorunda değildir?

Alıştırmalar

1. Döner sıralı çizelgeleme, zaman dilimi 2 ms: **P1 = 4, P2 = 2, P3 = 3 ms**. Gantt çizelgesini çizin ve her sürecin bitiş zamanını bulun.
2. Sayfa boyutu 4096 bayt. Sanal adres **12300** hangi sayfada ve sayfa içinde hangi bağıl konumdadır? Sayfa 3, çerçeve 1'deyse fiziksel adres nedir?
3. Klavyeden girdi bekleyen bir süreç hangi durumdadır? Tuşa basılınca hangi duruma, sonra hangi duruma geçer? Geçişleri diyagramda gösterin.
4. Güvenli Önyükleme açıkken değiştirilmiş bir önyükleyici neden çalışmaz? Güven zincirindeki imza denetimiyle açıklayın.
5. Windows ve Linux gereken bir test ortamı için sanal makine mi konteyner mi seçersiniz? Nedenini çekirdek paylaşımıyla açıklayın.
6. Bir uygulamanın dosyaya yazmasını; sistem çağrısı, çekirdek modu, dosya sistemi ve disk sürücüsü adımlarıyla sırayla anlatın.

ÖZET

Kaynaklar

- Silberschatz, Galvin ve Gagne, Operating System Concepts.
- Tanenbaum ve Bos, Modern Operating Systems.
- The Linux Foundation ve Free Software Foundation lisans kaynakları.

Son

Yazılım ve İşletim Sistemleri

Sonraki sunum: **Windows İşletim Sistemi** · **Çalışma soruları** · Simülatör: **Süreç Çizelgeleme** · **Terim sözlüğü**