Bölüm 2: Yapılar

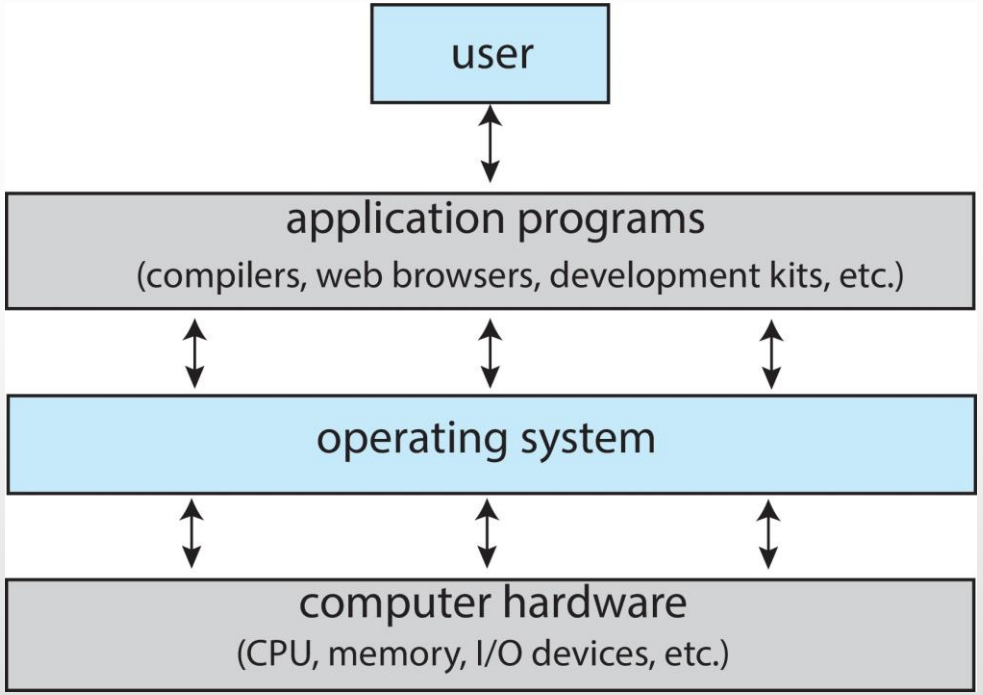
İşletim Sistemleri

İşletim Sistemi

- Modern bilgisayar çok karmaşıktır.
- Uygulama programcısının her detayı bilmesi imkansızdır.
- Kaynakları daha iyi, daha basit, ve daha sade yönetebilmek için bir katman
- Çeşitli işletim sistemleri; Windows, Linux, MacOS
- Kullanıcılar, kabuk (Shell) veya GKA (GUI) sayesinde bilgisayar ile etkileşime girer.

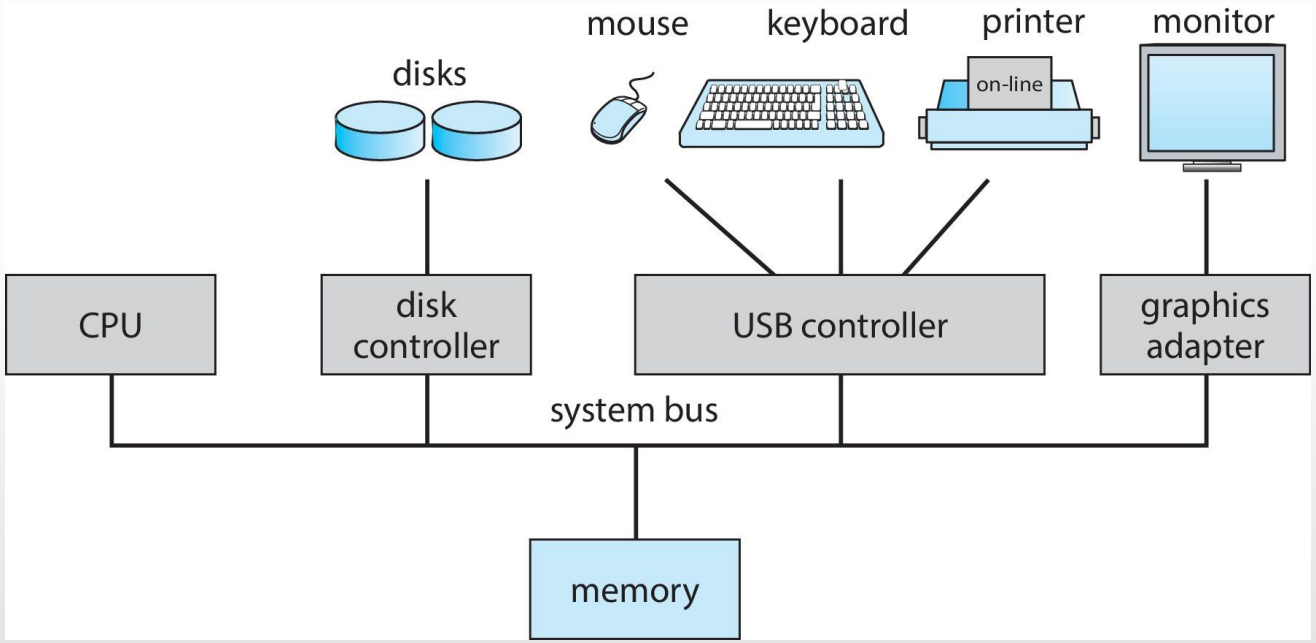
Modern Bilgisayarın Bileşenleri

- .

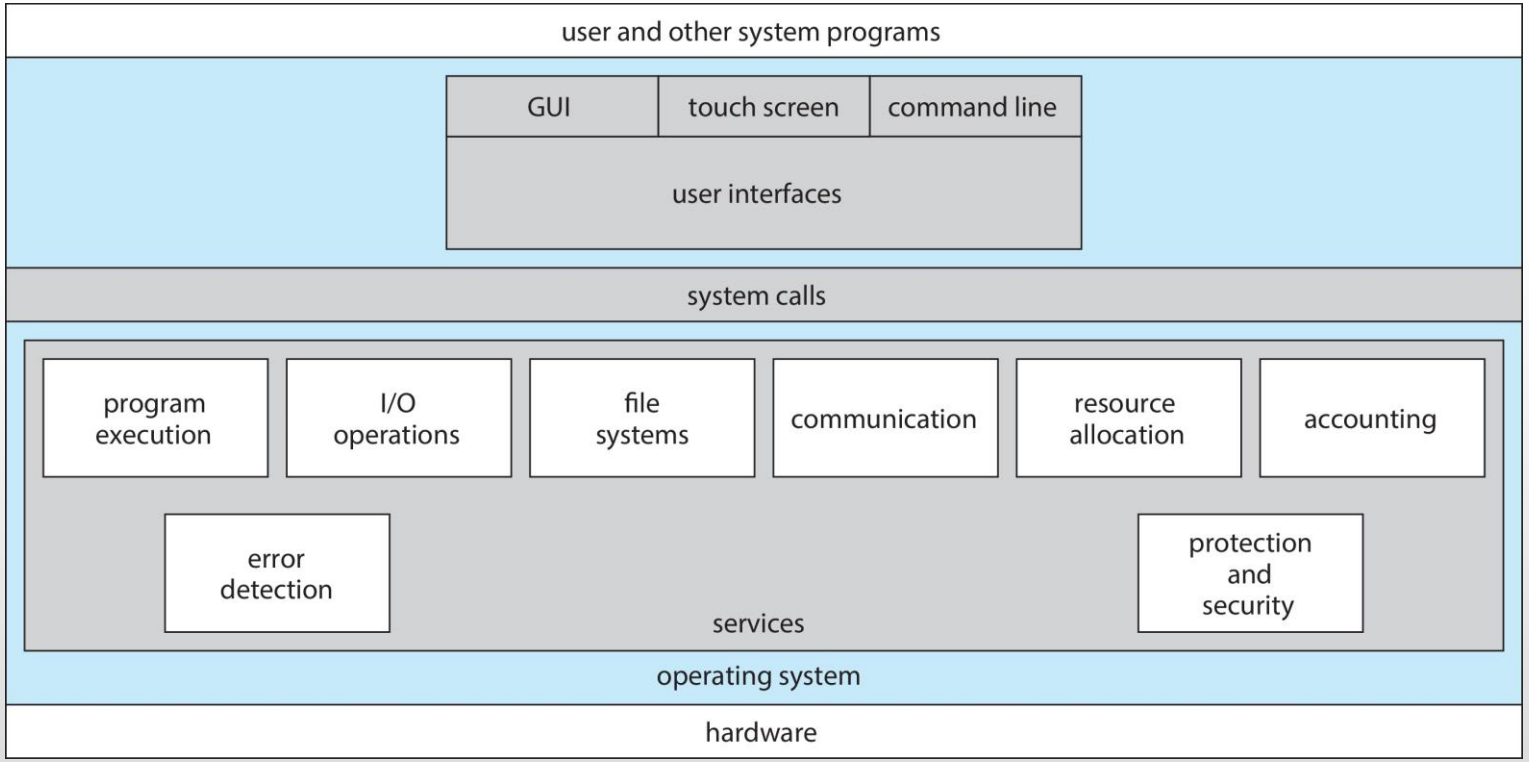


Modern Bilgisayarın Donanım Bileşenleri

- İşlemci
- Ana bellek
- Disk
- Yazıcı
- Klavye
- Fare
- Ekran
- Ağ arayüzleri
- G/Ç cihazları



İşletim Sistemi Servisleri



Kabuk (Shell) ve GKA (Grafik Kullanıcı Arayüzü)

- Kabuk: Kullanıcıların işletim sistemiyle etkileşimini sağlayan komut satırı arayüzü.
- GUI (Grafik Kullanıcı Arayüzü): Simgeler, pencereler ve fare gibi grafik öğeleri kullanarak kullanıcıların işletim sistemiyle etkileşime girmesini sağlayan görsel bir arayüz.
- Kabuk ve GUI, çekirdek işletim sisteminin değil, kullanıcı arabiriminin parçasıdır.
- Kabuk ve GUI, kullanıcının işletim sistemiyle etkileşime girmesi için bir araç sağlar, ancak temel sistem işlevlerinden veya hizmetlerinden herhangi birini sağlamazlar.

Kabuk (Shell) ve GKA (Grafik Kullanıcı Arayüzü)

- Aynı işletim sisteminde aynı anda birden çok kabuk ve GUI kullanılabilir.
- Kullanıcının işletim sistemi deneyimini kişiselleştirmesine ve onu kendi özel ihtiyaçları için daha verimli ve etkili hale getirmesine olanak tanır.
- Kabuk, belirli görevleri gerçekleştirmek için daha verimli ve etkili bir yol sağlarken, GUI, işletim sistemiyle etkileşimde bulunmak için daha kullanıcı dostu ve sezgisel bir yol sağlar.
- Kabuk ve GKA, işletim sisteminin teknik yetenekleri veya tercihleri ne olursa olsun daha geniş bir kullanıcı yelpazesi tarafından erişilebilir ve kullanılabilir olmasını sağlar.

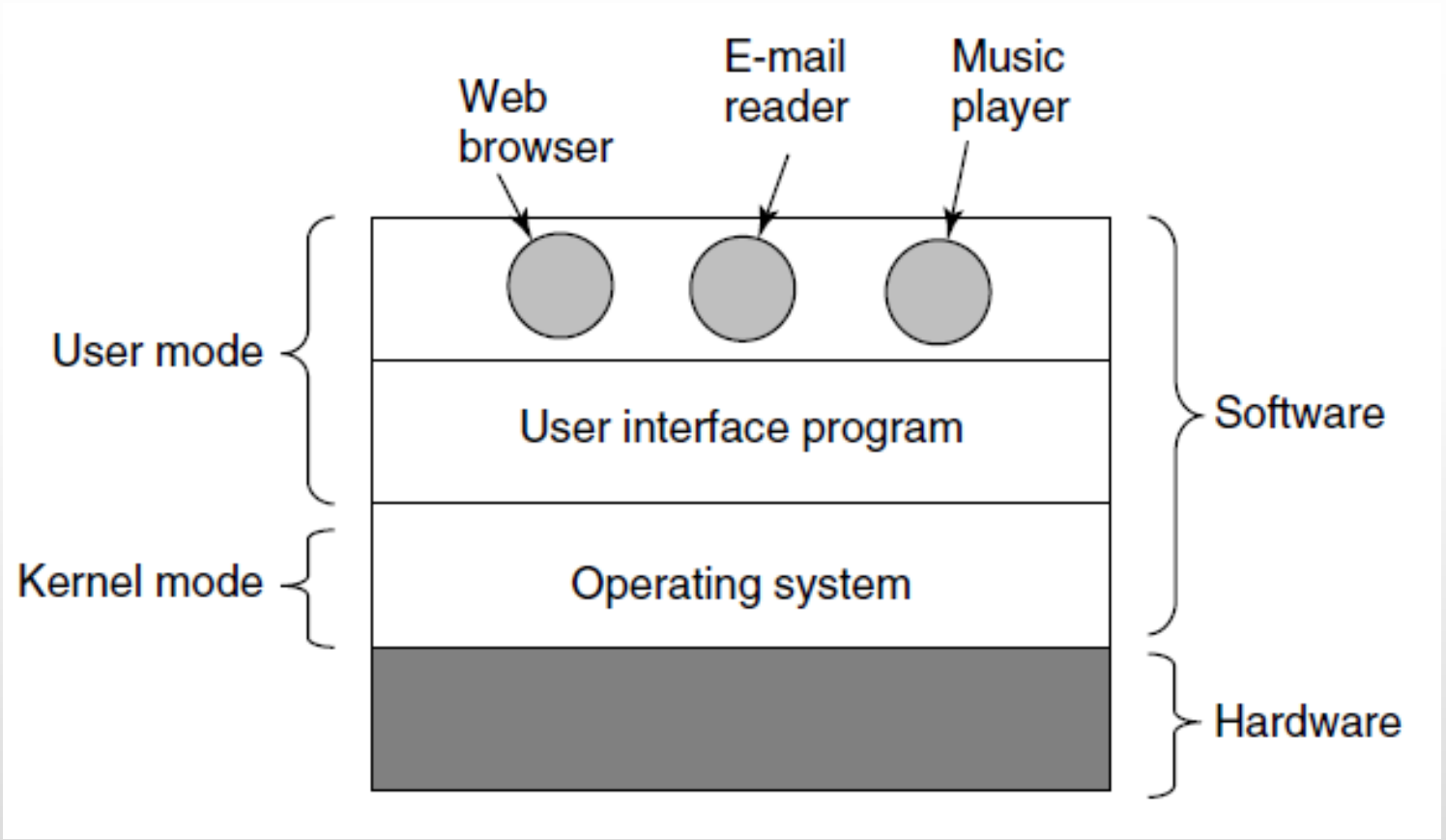
Aygıt Yöneticisi

- Aygıt yöneticisi, bilgisayara bağlı donanım aygıtlarını yönetmekten sorumlu işletim sisteminin bir bileşenidir.
- Cihazlar hakkında bilgi sağlar ve kullanıcının cihazları yapılandırmasına, güncellemesine ve sorunlarını gidermesine olanak tanır.
- Donanım aygıtlarının sorunsuz çalışması ve işletim sistemi ile uyumluluğunun sağlanmasında kilit rol oynar.
- Bilgisayarda çalışan ayrı bir uygulama veya program değildir.
- Cihaz türü, üreticisi ve sürümünün yanı sıra cihazla ilgili sorunlar veya problemler hakkında bilgi sağlar.

İşletim Sisteminin Konumu

- Bilgisayar donanımı ve yazılım arasında bir arayüzdür.
- Donanımın yazılım tarafından nasıl kullanılacağını yönetir.
- Donanım, fiziksel olarak mevcut olan bileşenleri (örneğin CPU, RAM, diskler) temsil eder.
- Yazılım ise, bilgisayarın yapabileceği işlemleri yürütmek için yazılmış kodları içerir.
- İşletim sistemi,
  - yazılımın donanımı kullanmasını kontrol eder,
  - donanımın kullanımını optimize eder ve
  - sistemin güvenliğini sağlar.

İşletim Sistemi Konumu



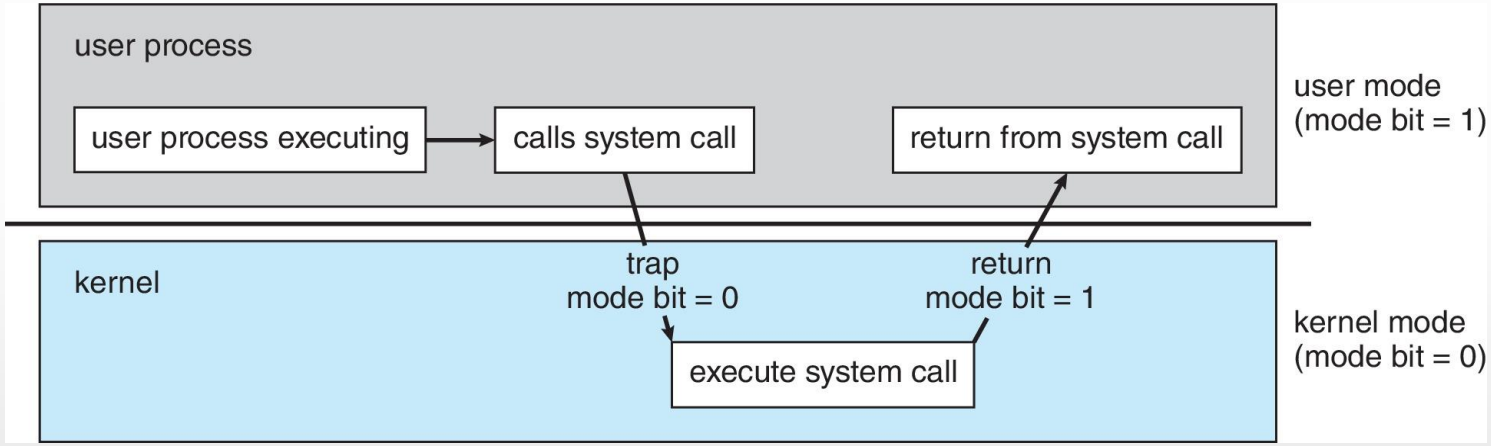
Çekirdek Modu ve Kullanıcı Modu

- Çoğu bilgisayarın iki çalışma modu vardır:

- İşletim sistemi, tüm donanıma tam erişime sahip olan ve herhangi bir talimatı yürütebilen çekirdek modunda çalışır.
- Yazılımın geri kalanı, sınırlı erişim ve kapasiteye sahip kullanıcı modunda çalışır.
- Kabuk ve GKA, kullanıcı modu yazılımının en düşük seviyesidir.

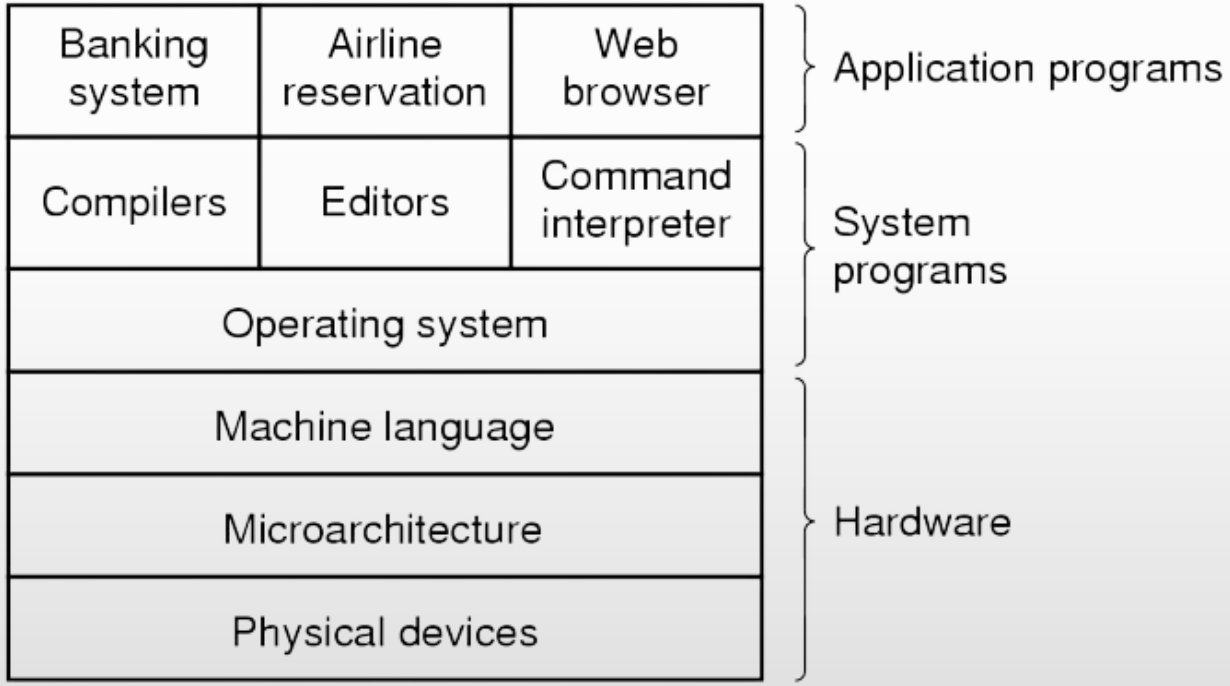
Çekirdek Modu ve Kullanıcı Modu

- .



İşletim Sistemi Konumu

- .



Fiziksel Aygıtlar, Mikro Mimari, Makine Dili

- Fiziksel aygıtlar, bir bilgisayarın merkezi işlem birimi (CPU), bellek, depolama ve giriş/çıkış aygıtları gibi donanım bileşenleridir.
- Mikro mimari, veri yolu, bellek hiyerarşisi ve kontrol birimi dahil olmak üzere bilgisayarın işlemcisinin özel tasarımıdır.
- Makine dili, doğrudan bilgisayarın donanımı tarafından yürütülebilen ikili koddan oluşan en düşük seviyeli programlama dilidir.
- İşletim sistemi, aygıt sürücüleri aracılığıyla fiziksel aygıtlarla iletişim kurar ve bunları yönetir.
- Aygıt sürücüleri, üst düzey istekleri fiziksel aygıtlar tarafından yürütülebilecek düşük düzeyli komutlara çevirir.

Fiziksel Aygıtlar, Mikro Mimari, Makine Dili

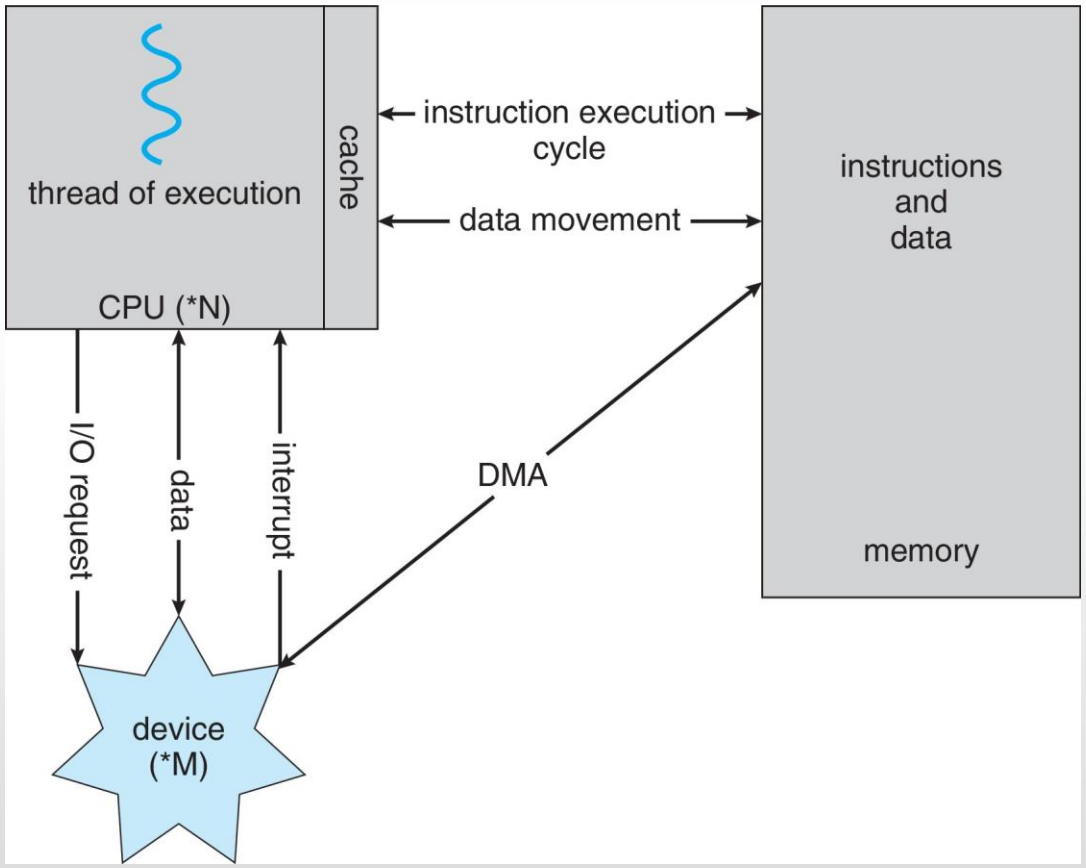
- İşletim sistemi, uygulamalar ile mikro mimari arasındaki arabirimi sağlayarak, uygulamaların bilgisayar kaynaklarına etkin bir şekilde erişmesine ve bunları kullanmasına izin verir.
- İşletim sistemi, üst düzey uygulamaları doğrudan bilgisayarın donanımı tarafından yürütülebilen makine diline çevirir.
- Donanım yapılandırması veya mikro mimarisi ne olursa olsun uygulamaların bilgisayarda yürütülmesine izin verir.
- İşletim sisteminin tasarımı ve uygulaması, donanım tarafından kullanılan makine dilinin yanı sıra bilgisayarın belirli mikro mimarisini ve fiziksel cihazlarını dikkate almalı ve hesaba katmalıdır.

Genişletilmiş Makine Olarak İşletim Sistemi

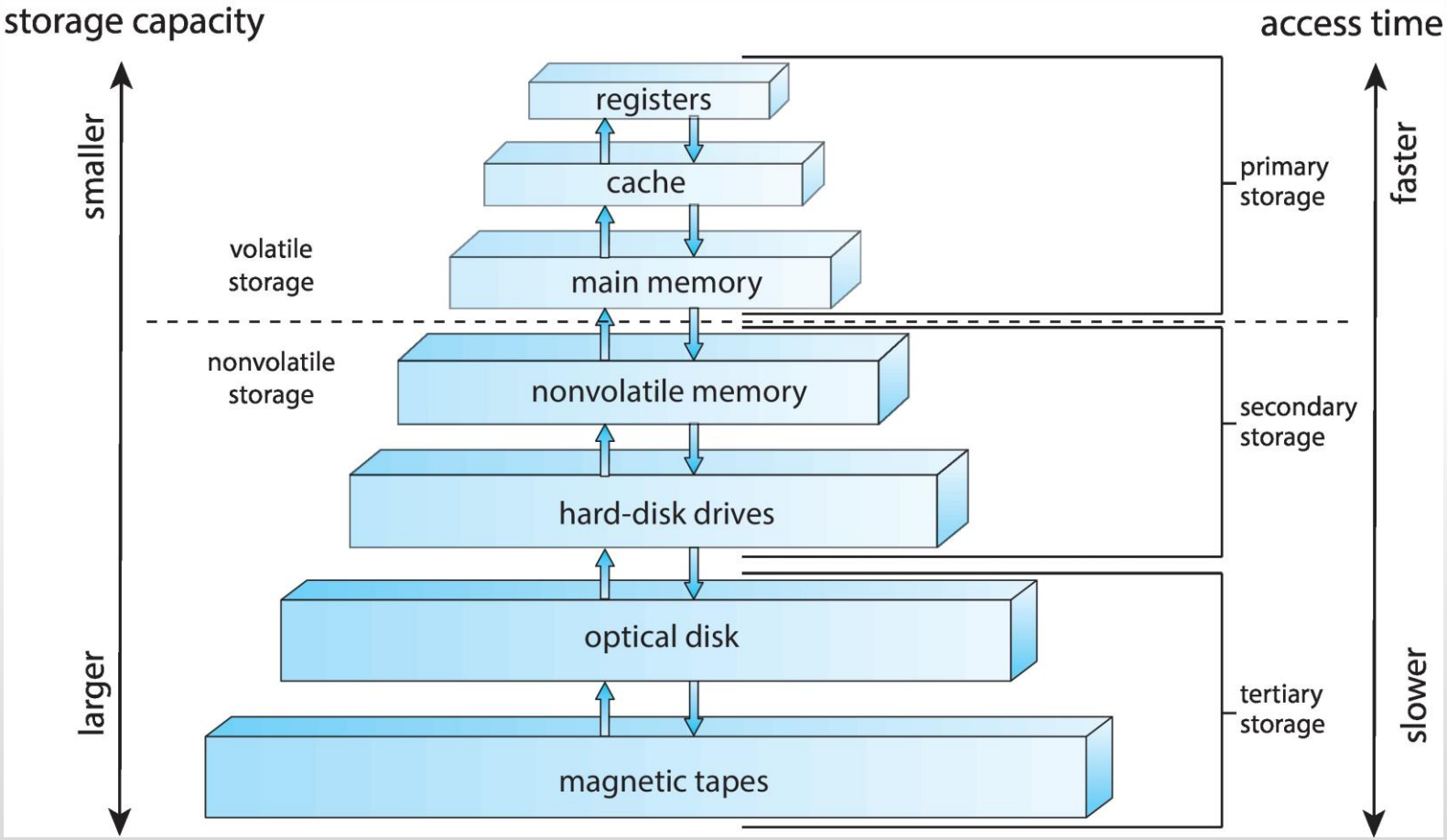
- Soyutlama:
  - İşlemci – Süreç
  - Depolama – Dosya
  - Bellek – Adres uzayı

- 4 tip çalışana ihtiyaç var:
  - Donanım tasarımcısı
  - Çekirdek tasarımcısı
  - Uygulama geliştirici
  - Son kullanıcı

Von Neumann Mimarisi



Bellek Hiyerarşisi



Bellek Hiyerarşisi

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Aygıt Sürücüsü

- İşletim sistemi denetleyiciyle konuşur. (komut verir, yanıt alır)
- Denetleyici üreticileri, her işletim sistemi için bir sürücü sağlar
- Sürücü, çekirdek modunda çalışır
- Denetleyici, sürücüyle iletişim kurmak için yazmaçlar kullanır.
- Üç iletişim modu
  - Sorgulama (polling)
  - Kesmeler (interrupt)
  - DMA

Sorgulama, Kesilme ve Direkt Bellek Erişimi

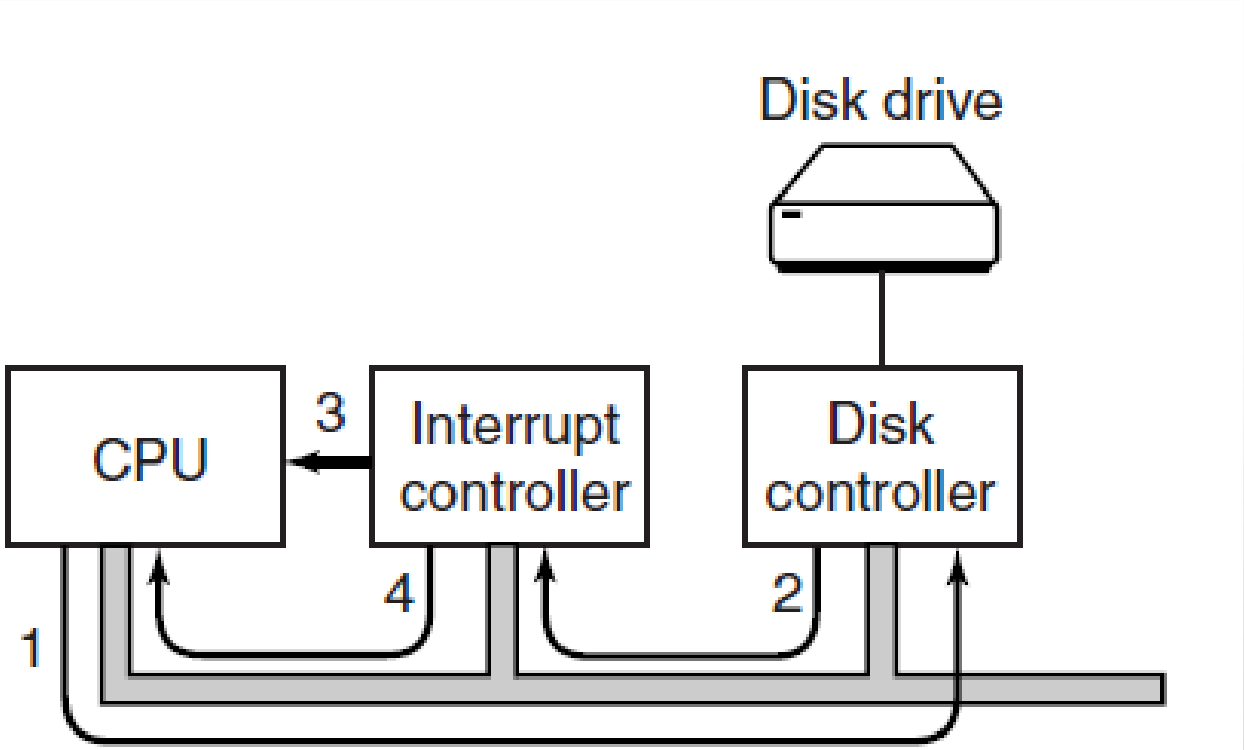
- **Sorgulama** (polling), aygıtın hazır olup olmadığını görmek için aygıtın durumu kontrol edilir. İşletim sistemi, bir döngü veya zamanlayıcı kullanarak aygıtı sürekli olarak denetler.
- **Kesilmeler** (interrupt), aygıtın ilgilenilmesi gerektiğini işletim sistemine bildirme yöntemidir. Bir aygıt bir kesme oluşturduğunda, işletim sistemine bir sinyal gönderir, işletim sistemi mevcut görevini durdurur ve kesmeyi işler. Sorgulamadan daha verimli ve duyarlı bir yöntemdir.
- **DMA**, işlemciyi dahil etmeden doğrudan bir cihaz ile ana bellek arasında veri aktarma yöntemidir. DMA, büyük miktarda veri aktarımı için daha hızlı ve daha verimli bir yöntem sağlar.

G/Ç Cihazları - Sorgulama

- Sürücü, denetleyiciye komut verir
- Sürücü, aygıt hazır olana kadar sorgular
  - Örneğin, kabul etmeye hazır olana kadar yazıcı denetleyicisini sorgula, karakter gönder
- Yüksek CPU kullanımına neden olur
- Programlanmış G/Ç olarak adlandırılır, artık kullanılmıyor

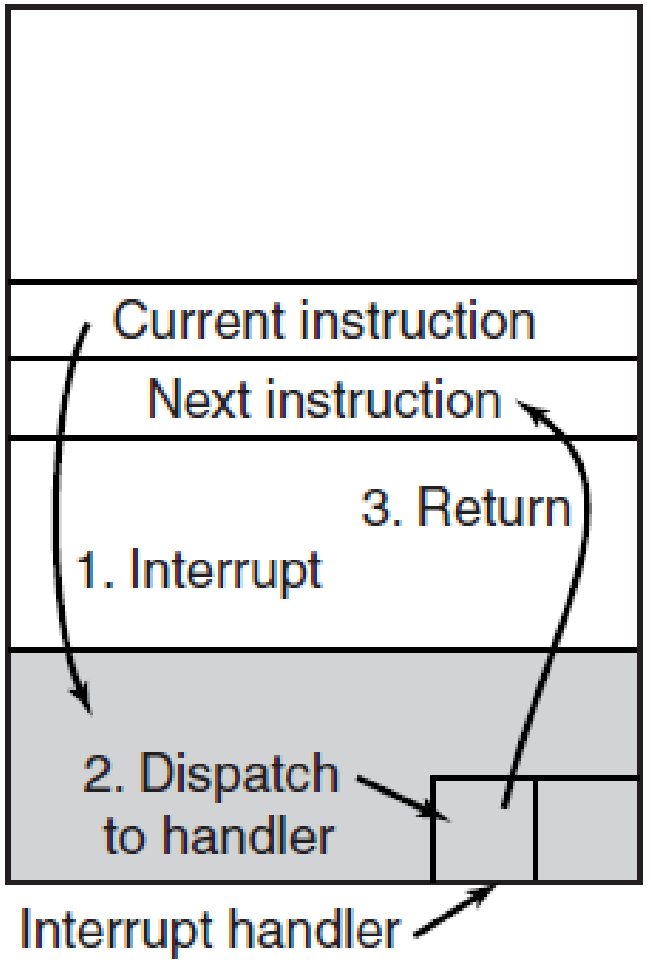
G/Ç Cihazları - Kesme

G/Ç cihazını başlatma ve bir kesme alma adımları



G/Ç Cihazları - Kesme

- Kesme işleme,
  - kesmeyi alma
  - kesme işleyicisini çalıştırma
  - kullanıcı programına dönme
- G/Ç işlemi bittiğinde kesme üret
- İşlem yapılırken işlemcinin başka işler yapmasına izin verir.



G/Ç Cihazları - DMA

- Özel (denetleyici) yonga var
- Bellek ile veri transferinde işlemci kullanmaktan kaçınır
- İşlemci, yongaya aktarım hakkında gerekli bilgileri verir
- Yonga işlem bittiğinde kesme üretir.

Bilgisayarın Ayağa Kalkması

- BIOS: temel giriş/çıkış sistemi
- Ana kartta yer alır, düşük seviye G/Ç yazılımı

1. Bellek, klavye ve diğer temel cihazları kontrol eder
2. Önyükleme aygıtını belirler (disket, CD-ROM, disk)
3. Önyükleme aygıtının ilk sektörü belleğe okunur

- Sektör, hangi bölümün aktif olduğunu kontrol etmek için program içerir

4. Ardından, ikincil bir önyükleyici belleğe okunur
5. İşletim sistemi aktif bölümden okunur.

Zamanlayıcı (timer)

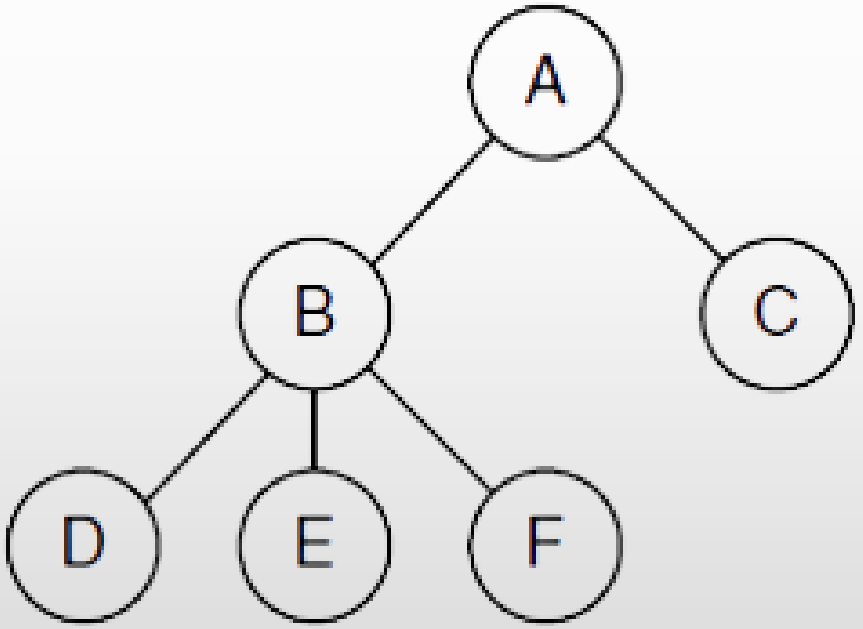
- Sonsuz döngüyü önlemek için kullanılır
- Belirli bir süre sonra kesilme yaratır
  - Fiziksel saat (clock) tarafından azaltılan bir sayaç tutulur
  - İşletim sistemi sayacı ayarlar (ayrıcalıklı talimat)
  - Sayaç sıfır olduğunda kesilme üretilir
  - İşletim sistemi kontrolü tekrar kazanır
  - Kendisine ayrılan süreyi aşan görevleri sonlandırır

Süreçler

- İşletim sistemi tarafından yürütülen programlardır.
- Bir programın çalıştırılabilmesi için gerekli tüm bilgiyi tutan konteyner olarak düşünülebilir.
- Süreç tablosunda tutulurlar. İşletim sistemi tarafından atanmış kaynaklar (örneğin bellek, CPU) ile ilişkilidir.
- Diğer süreçlerle konuşabilirler (IPC).
- Bellekte saklanır ve yürütülürler. Adres uzayı ile ilişkilidir.
- Adres uzayı: 0-4G, yürütülebilir program, programın verileri ve yığını
- Yazmaçlar, dosyalar, alarmlar, ilgili süreçler…

Süreçler

Süreç ağacı. A süreci, B ve C olmak üzere 2 çocuk süreç başlatır. B süreci D, E ve F olmak üzere 3 çocuk süreç başlatır.



Adres Uzayı

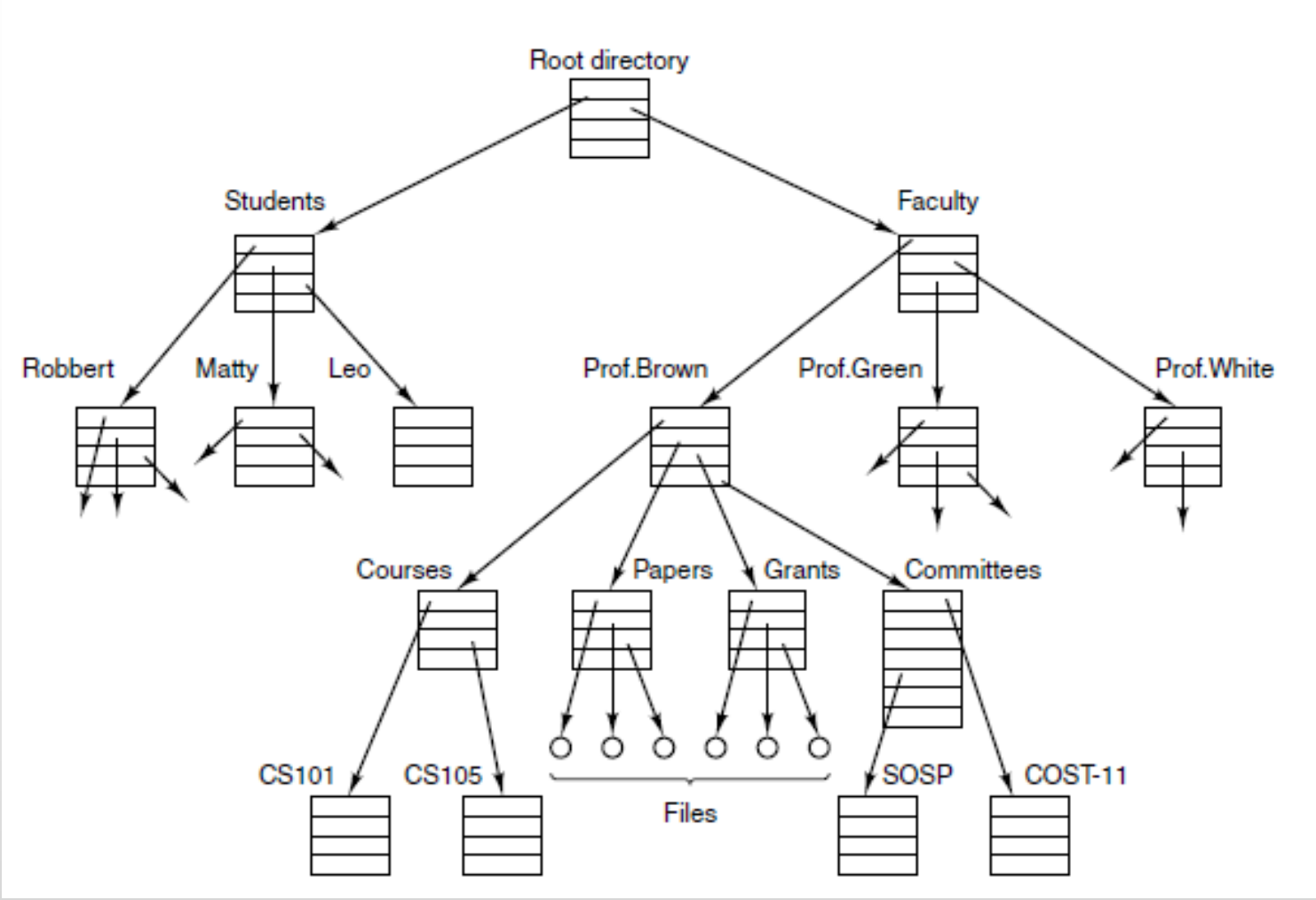
- Kavram olarak bir süreç tarafından kullanılan bellek
- İşletim sistemi, bellekte aynı anda birden çok sürece izin verir
- Bazı işlemler, fiziksel olarak mevcut olandan daha fazla belleğe ihtiyaç duyar, bu durumda sanal bellek devreye girer.

Dosyalar

- Veri depolama birimleridir.
- Dosyalar, veri, metin, resim, video, ses ve diğer türlerde olabilir.
- İşletim sistemi tarafından yönetilir ve veri depolama işlemleri gerçekleştirilir.
- İşletim sistemi tarafından belirlenen dizin yapısına göre saklanır.
- Kullanıcılar tarafından erişilebilir, okunabilir, yazılabilir veya silinebilir.
- Blok tabanlı (disk), karakter tabanlı (yazıcı, modem) olabilir.

Dosyalar

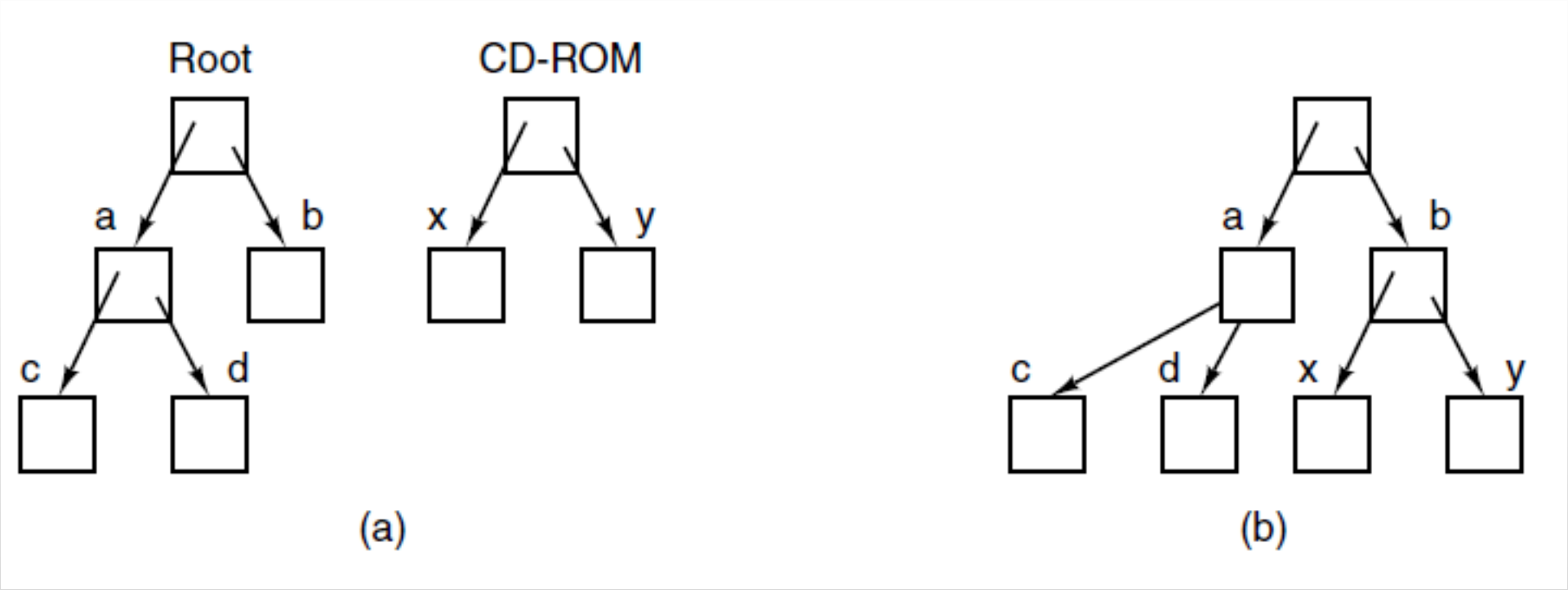
- Örnek dosya sistemi.



Dosyalar

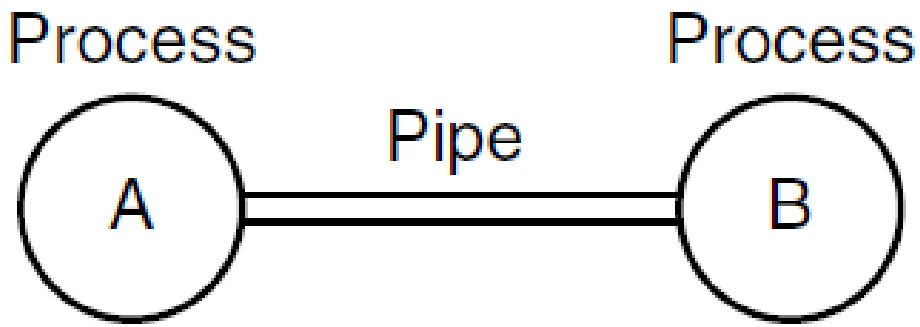
(a) Bağlamadan (mount) önce, CD-ROM'daki dosyalara erişilemez.

(b) Bağlandıktan sonra, dosya hiyerarşisinin bir parçasıdırlar.



Dosyalar

- 2 süreç boru (pipe) ile bağlanmış



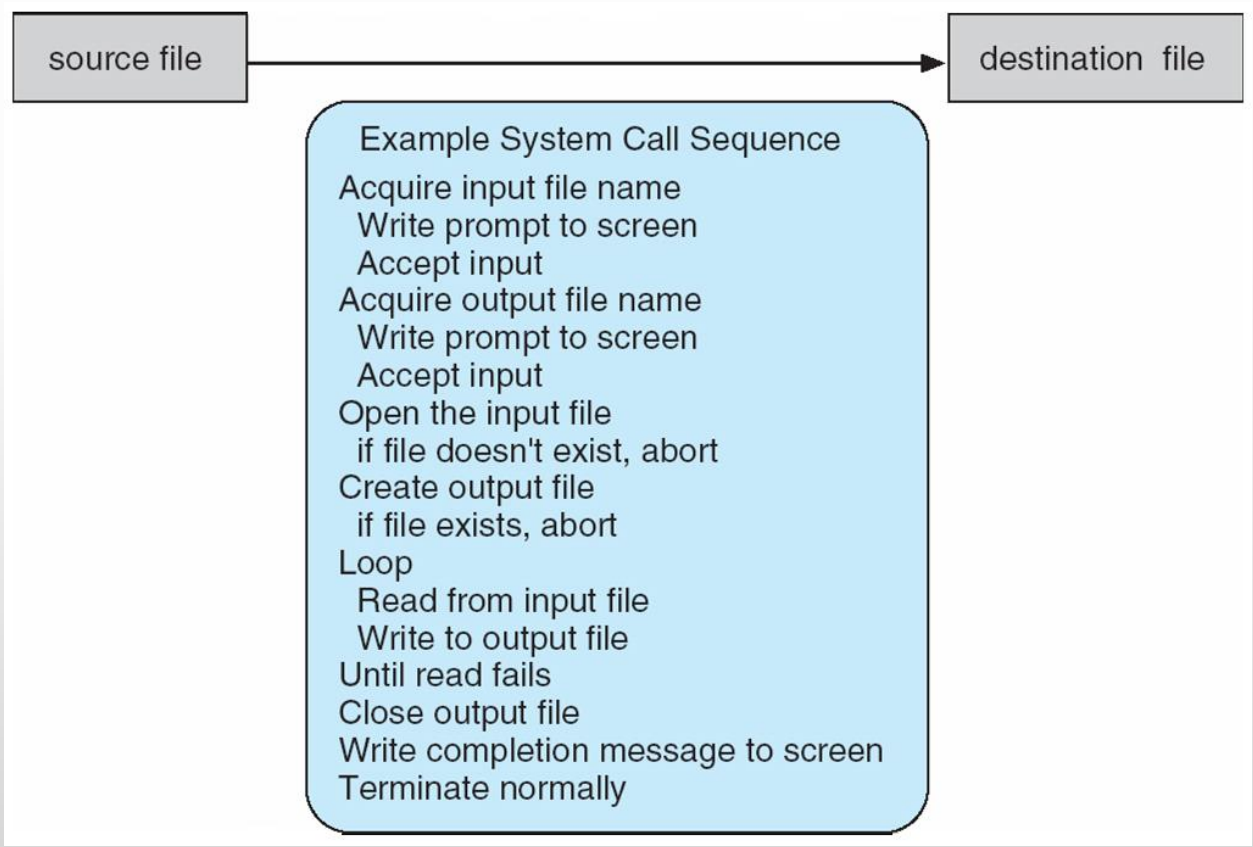
- Boru, iki süreci birbirine bağlayan ve veri gönderip alarak iletişim kurmalarını sağlayan tek yönlü bir veri kanalıdır.
- Anonim (anonymous), ebeveyn ve çocuk süreçler gibi ilgili süreçler arasında IPC için kullanılır.
- Adlandırılmış (named), iki ayrı uygulama arasındaki ilgisiz süreçler arasında IPC için kullanılır.

Sistem Çağrıları

- Sistem çağrıları, işletim sistemi tarafından sağlanan hizmetlere erişmek için kullanılır.
  - Örneğin, dosya işlemleri, bellek yönetimi, zaman hizmetleri, vb.
- İşletim sistemi tarafından tanımlanmış bir arayüze göre gerçekleştirilir.
- Sistem çağrıları, uygulama programları tarafından kullanılır ve işletim sistemi tarafından yürütülür.
- Sistem çağrıları sistemden sisteme değişir, ancak temel kavramlar benzerdir.

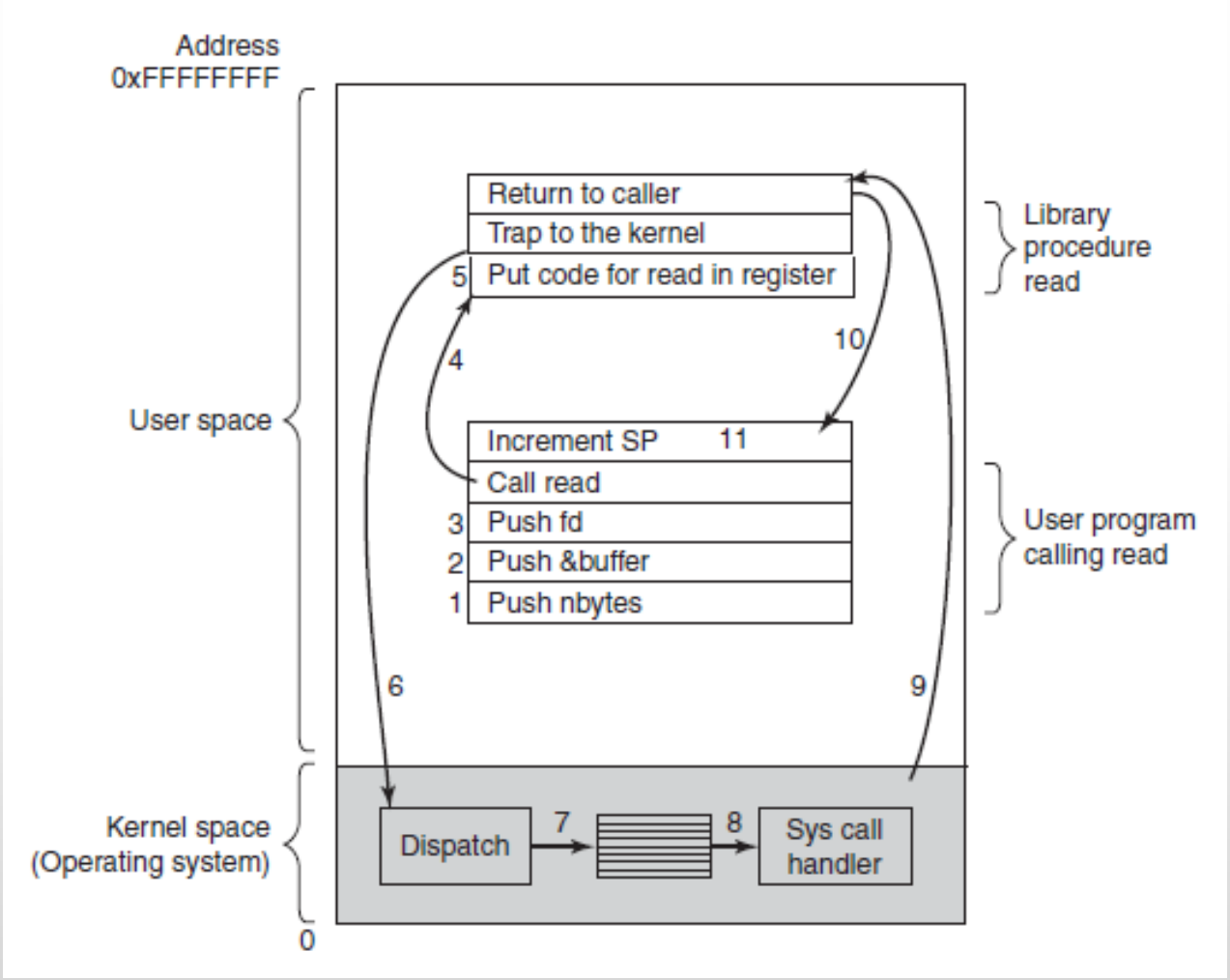
Örnek Sistem Çağrısı

- Bir dosyanın içeriğinin başka bir dosyaya aktarılması

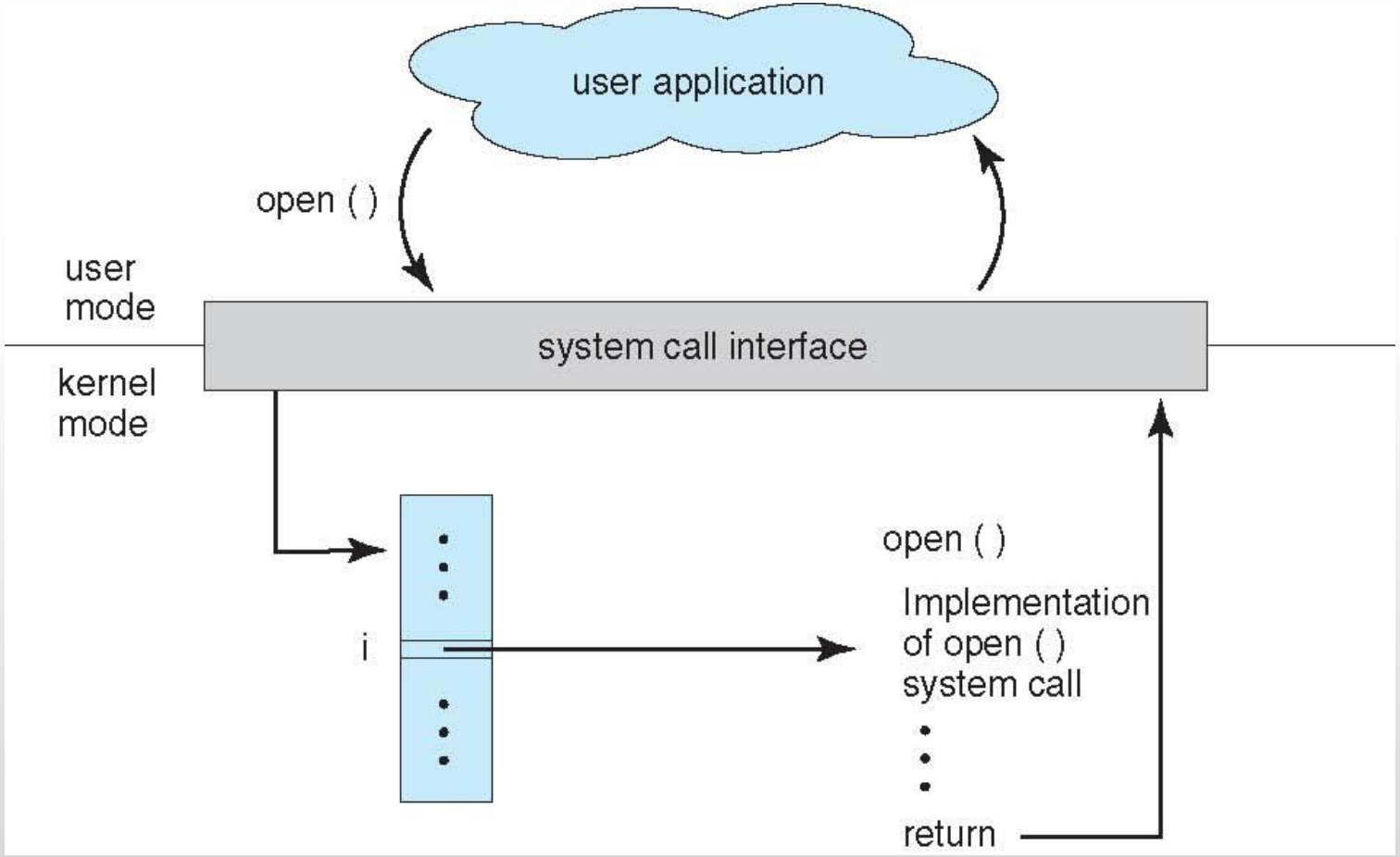


Sistem Çağrıları

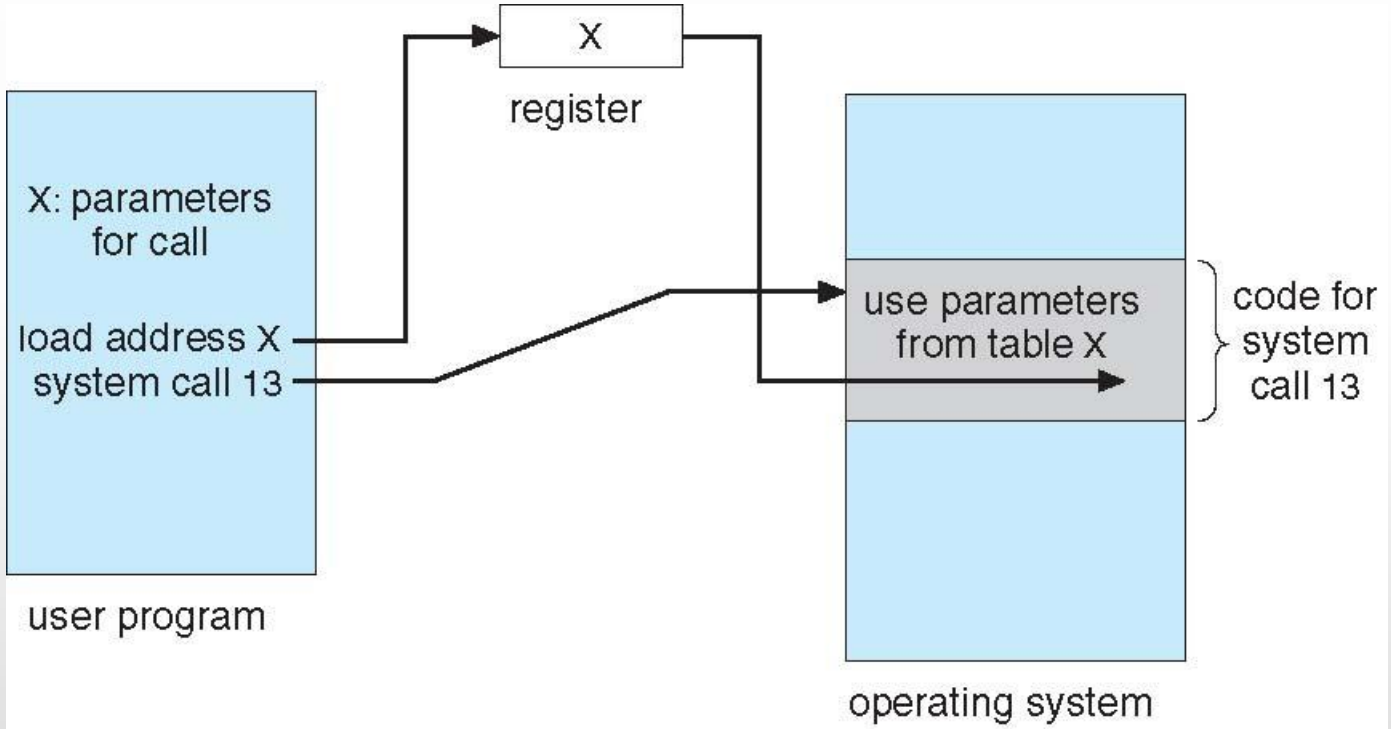
read(fd, buffer, nbytes) sistem çağrısının adım adım gösterimi



Sistem Çağrıları



Sistem Çağrıları – Parametre Aktarımı



Sistem Çağrıları – Süreç Yönetimi

Başlıca POSIX sistem çağrıları. Hata durumunda -1 döner.

pid: işlem kimliği.

s: geri dönüş kodu

Çağrı	**Tanım**
pid = fork()	Ebeveyn ile aynı bir alt süreç oluşturur
pid = waitpid(pid, &statloc, options)	Çocuk sürecin sonlanmasını bekler
s = execve(name, argv, environp)	Bir sürecin ana görüntüsünü değiştirir
exit(status)	Süreci yürütmeyi durdurur ve durumu geri döndürür

Sistem Çağrıları – Dosya Yönetimi

- fd: dosya tanıtıcı,
- n: bayt sayısı,
- position: dosya içinde göreli konum (offset).

Çağrı	Tanım
fd = open(file, how,...)	Okumak, yazmak veya her ikisi için bir dosya açar
s = close(fd)	Açık bir dosyayı kapatır
n = read(fd, buffer, nbytes)	Bir dosyadan arabelleğe veri okur
n = write(fd, buffer, nbytes)	Arabellekten bir dosyaya veri yaz
position = lseek(fd, offset, whence)	Dosya işaretçisini hareket ettirir
s = stat(name, &buf)	Bir dosyanın durum bilgisini alır

Sistem Çağrıları – Dizin Yönetimi

- .

Çağrı	Tanım
s = mkdir(name, mode)	Yeni bir dizin oluşturur
s = rmdir(name)	Boş bir dizini kaldırır
s = link(name1, name2)	name1'i işaret eden yeni bir name2 adında girdi oluşturur
s = unlink(name)	Bir dizin girdisini kaldırır
s = mount(special, name, flag)	Bir dosya sistemini bağlar
s = umount(special)	Bir dosya sisteminin bağlantısını keser

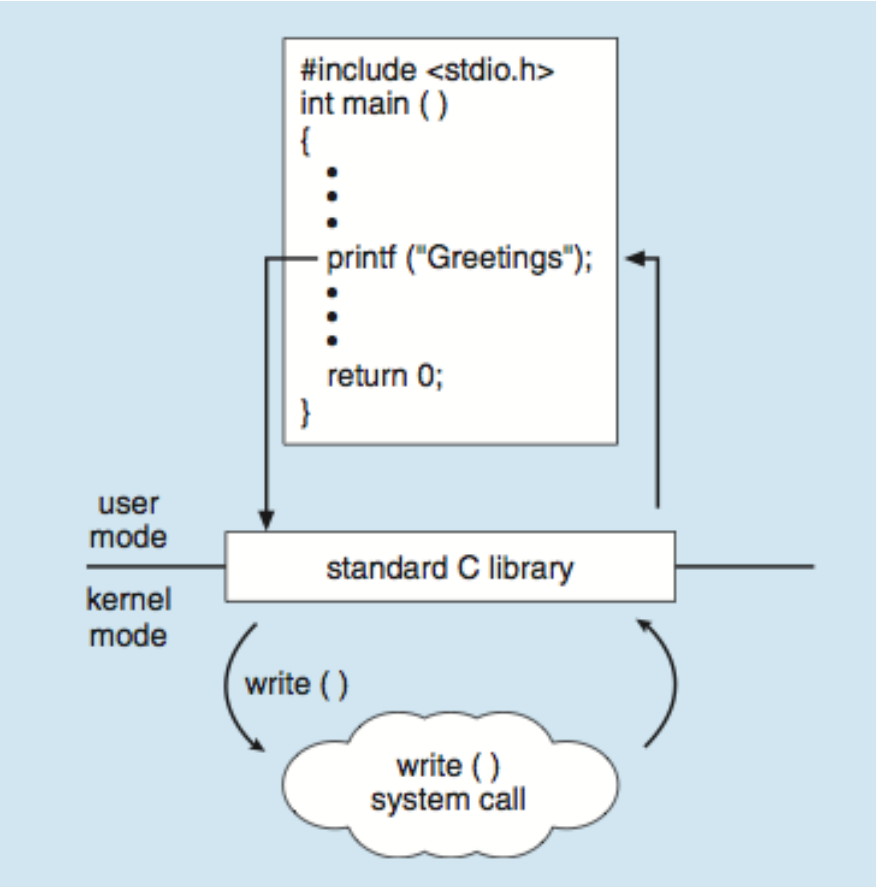
Sistem Çağrıları

seconds: geçen süre

Çağrı	Tanım
s = chdir(dirname)	Çalışma dizinini değiştirir
s = chmod(name, mode)	Bir dosyanın koruma bitlerini değiştirir
s = kill(pid, signal)	Bir sürece sinyal gönderir
seconds = time(&seconds)	1 Ocak 1970'ten bu yana geçen süreyi döndürür

Standart C Kütüphanesi

- .



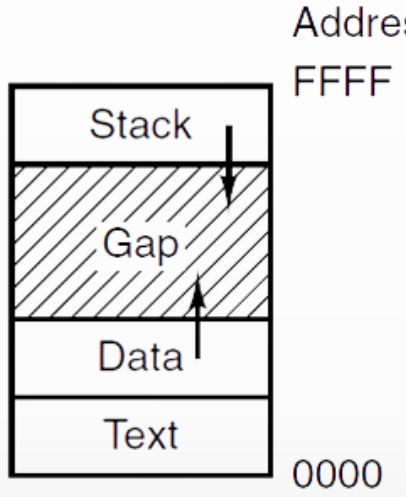
Süreç Yönetimi

```
#define TRUE 1

while (TRUE) {                                  /* repeat forever */
        type_prompt( );                         /* display prompt on the screen */
        read_command(command, parameters);      /* read input from terminal */

        if (fork( ) != 0) {                     /* fork off child process */
             /* Parent code. */
             waitpid(–1, &status, 0);           /* wait for child to exit */
        } else {
             /* Child code. */
             execve(command, parameters, 0);    /* execute command */
        }
}
```

Süreçlerin Bellek Yönetimi

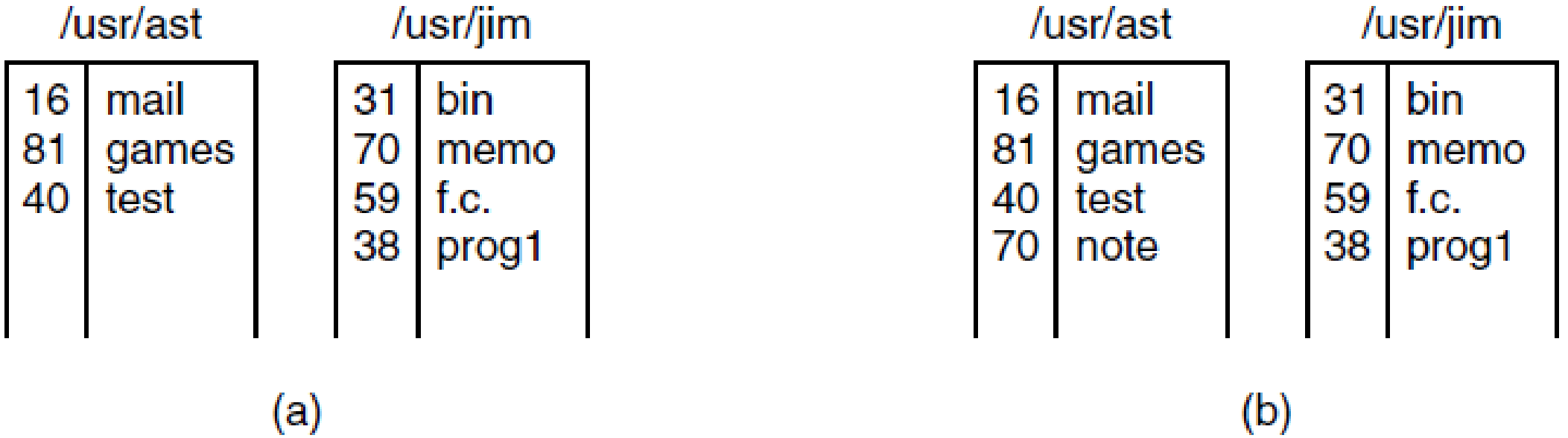


- Süreçler 3 kesime sahiptir. metin, veri, yığın
- **Yığın**, bir işlev çağrısı için gerekli olan parametreler, yerel değişkenler ve işlev dönüş adresleri gibi geçici verileri depolar.
- **Veri**, genel ve statik değişkenlerin yanı sıra dinamik olarak ayrılmış belleği depolar. Tüm iş parçacıkları arasında paylaşılır.
- **Metin**, sürecin yönergelerini uygulayan makine kodunu tutar. Salt okunurdur ve işlemin tüm iş parçacıkları arasında paylaşılır. Talimatları yürütmek için CPU tarafından kullanılır.

Dizin Yönetimi

(a) usr/jim/memo'yu ast'nin dizinine bağlamadan önce.

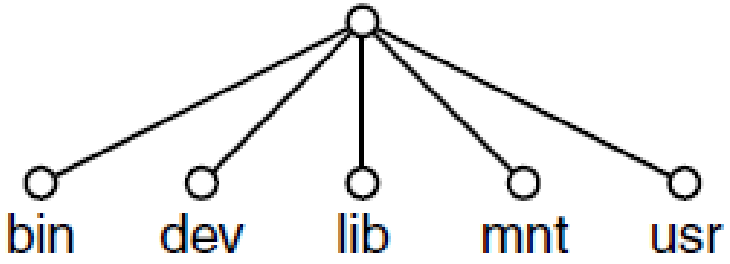
(b) bağlandıktan sonra.



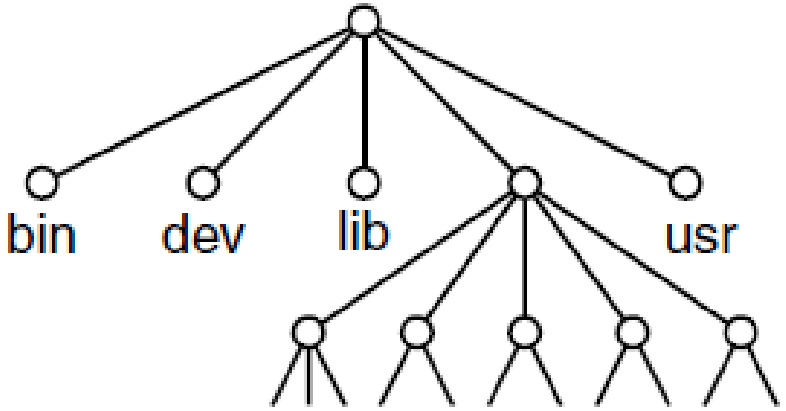
Dizin Yönetimi

(a) Bağlamadan önce dosya sistemi

(b) Bağlamadan sonra



(a)

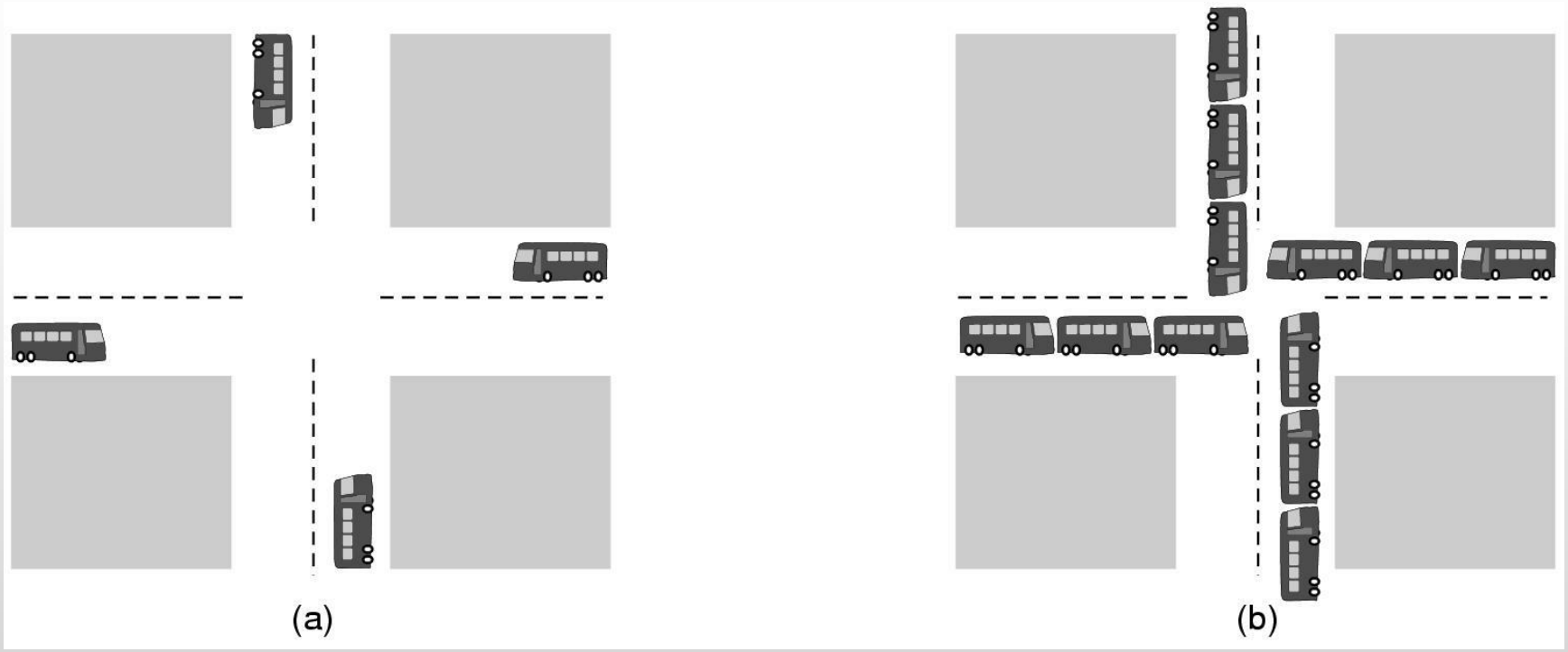


(b)

Kilitlenme (Deadlock)

(a) Potansiyel kilitlenme

(b) Gerçekleşmiş kilitlenme



The Windows Win32 API

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory

The Windows Win32 API

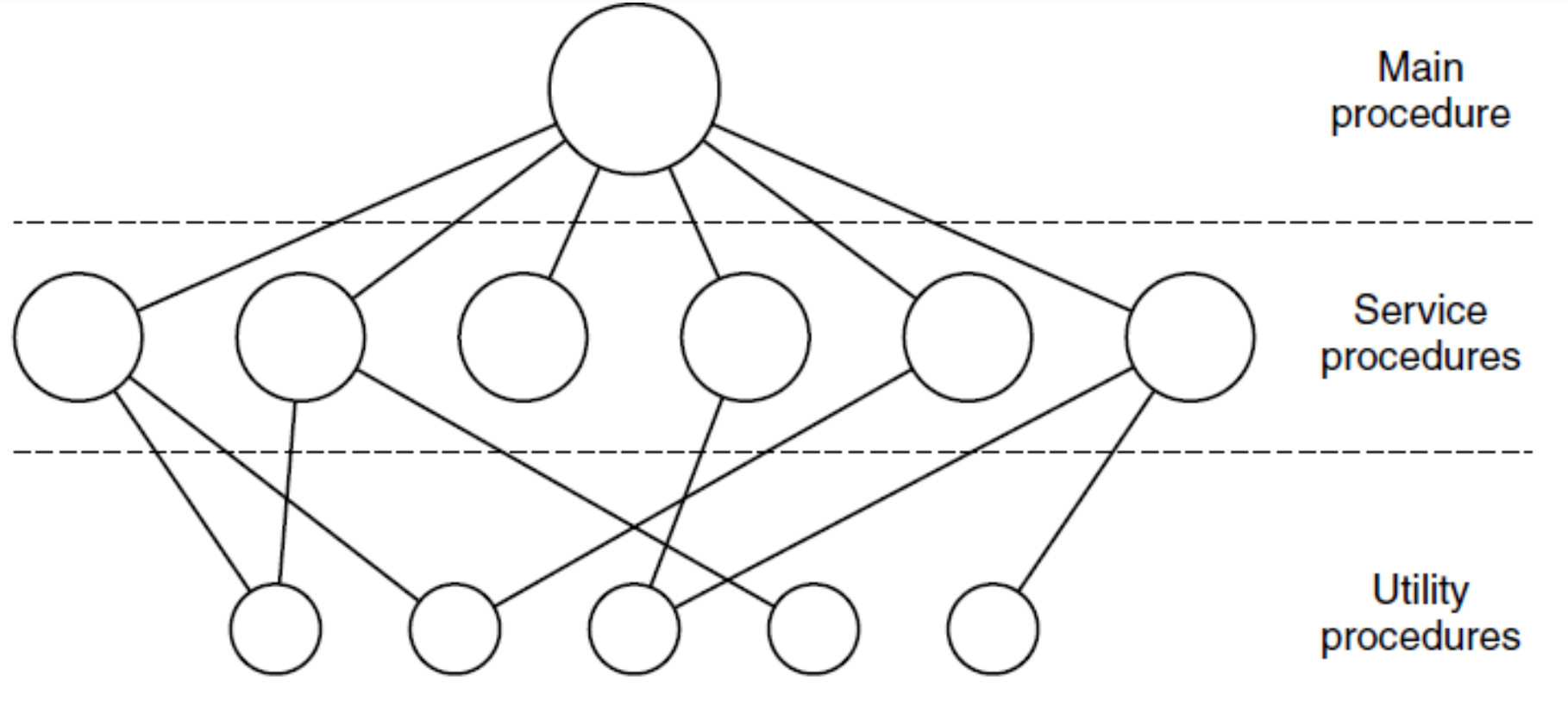
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Monolitik Sistem

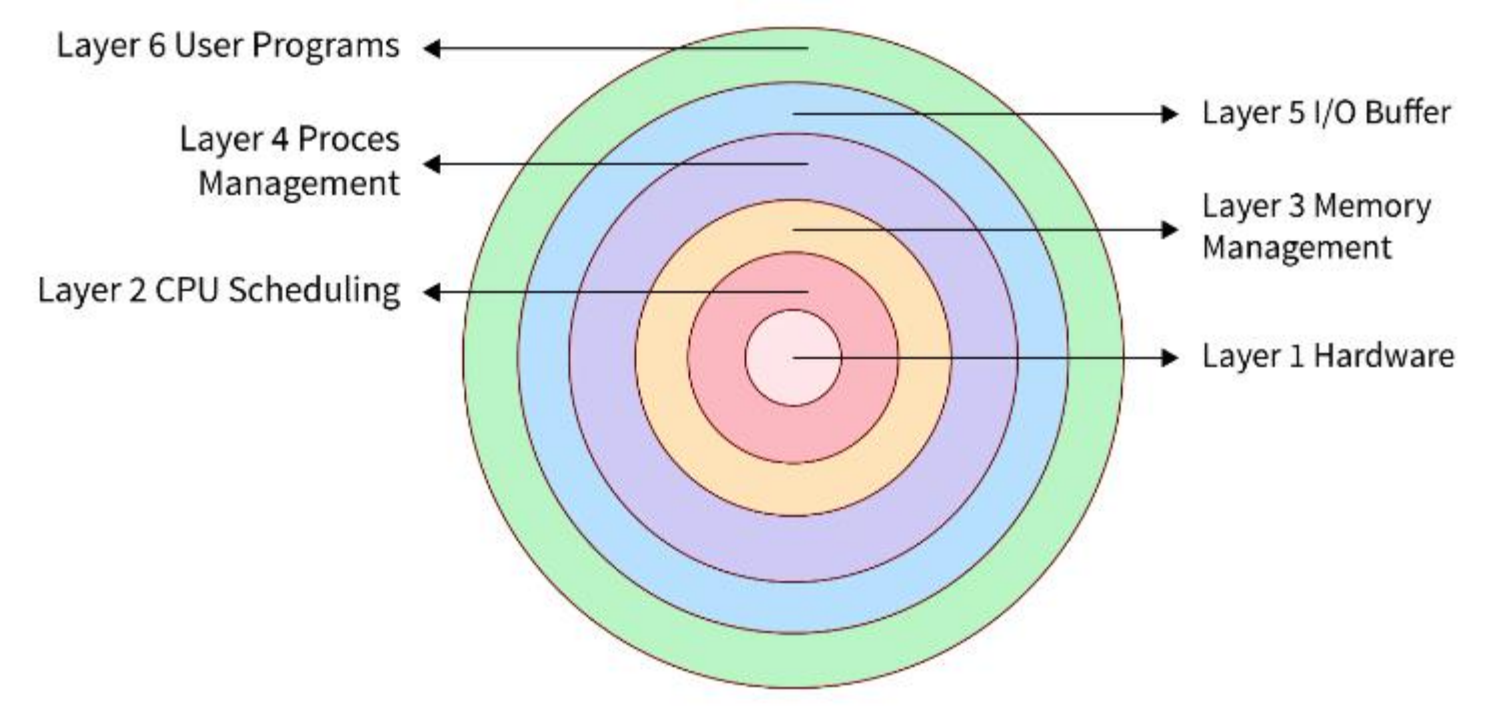
İşletim sisteminin temel yapısı

- İstenen hizmet prosedürünü başlatan bir ana program.
- Sistem çağrılarını gerçekleştiren bir dizi hizmet prosedürü.
- Hizmet prosedürlerine yardımcı olan bir dizi yardımcı prosedür.

Monolitik Sistem Yapısı



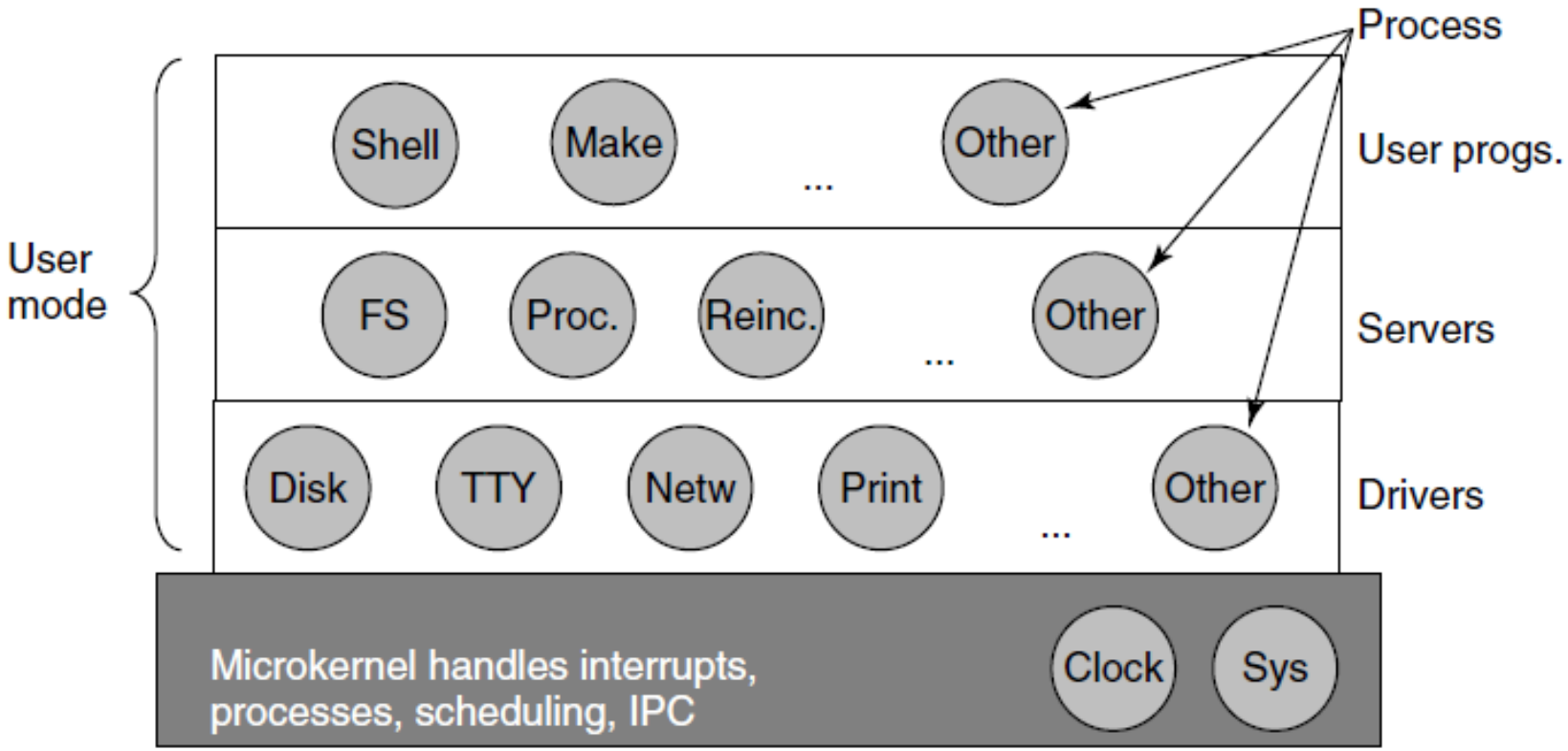
Katmanlı Sistem Yapısı



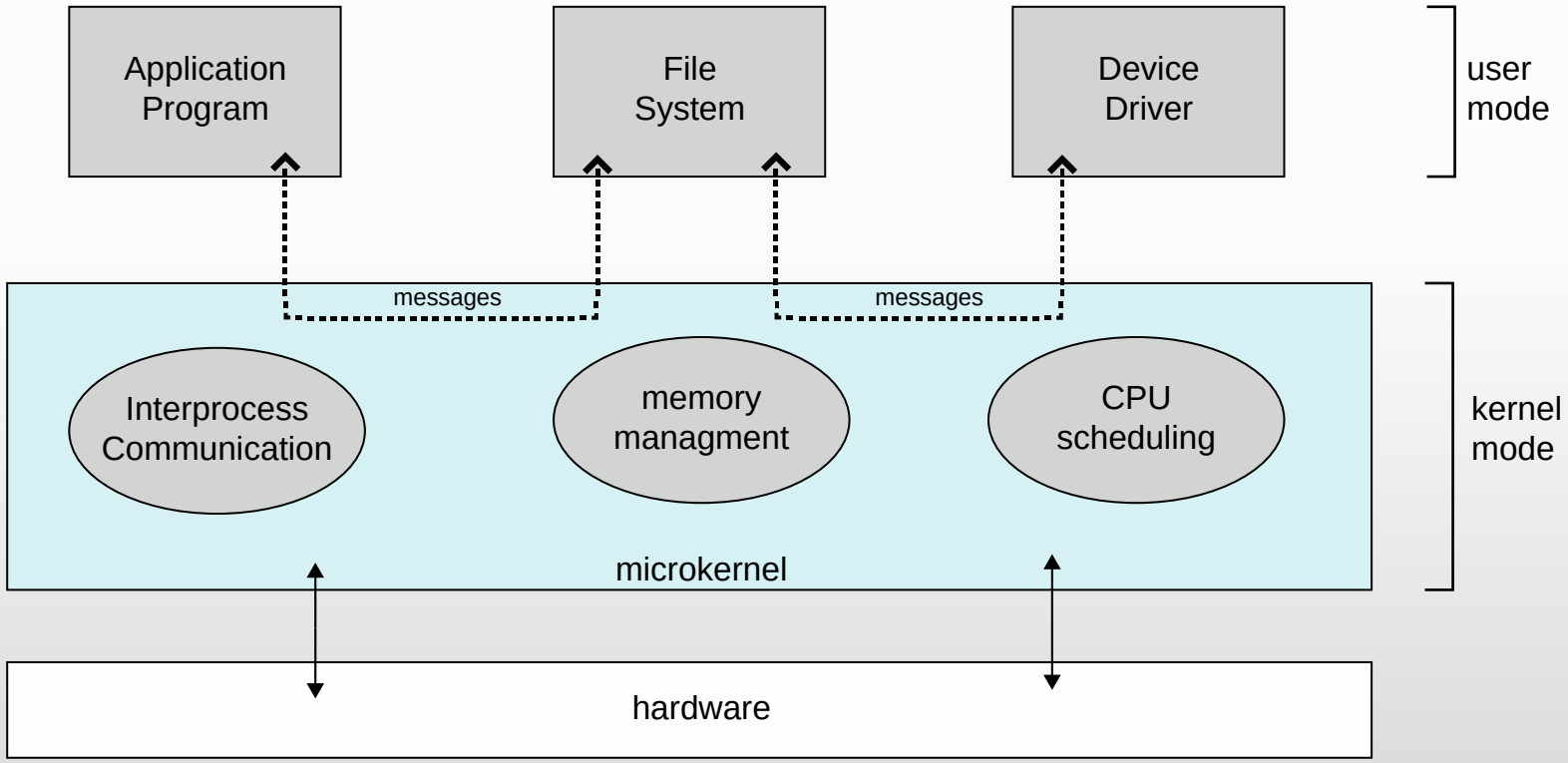
Mikro Çekirdek

- Çekirdekte az sayıda sürecin yürütülmesine izin verilir
- Hataların etkilerini en aza indirir
  - Sürücüdeki bir hatanın sistemi çökertmesi istenmez
- Mekanizma çekirdekte, ilke (policy) çekirdeğin dışındadır
  - Mekanizma, süreçler önceliklerine göre çizelgelenir (kernel)
  - İlke, süreç öncelikleri kullanıcı modunda tanımlanır (user)

Mikro Çekirdek Sistem Yapısı



Mikro Çekirdek Sistem Yapısı



Andrew Tanenbaum (Monolitik çekirdek)

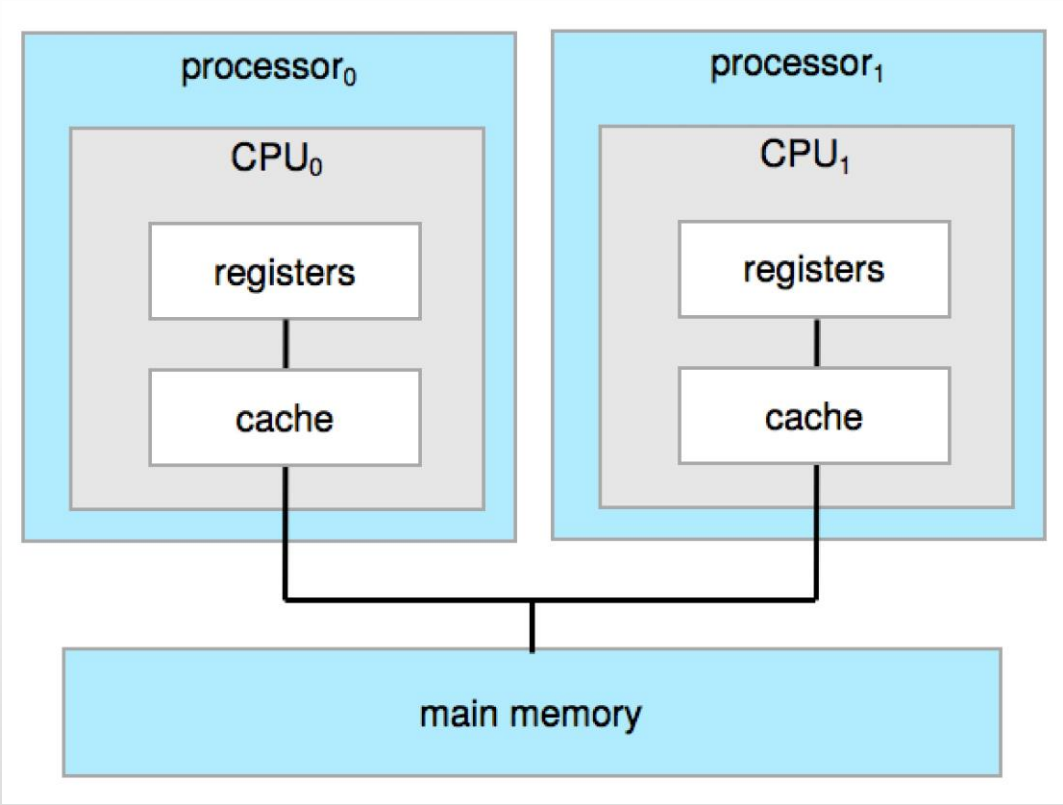
- Monolitik çekirdeklerin tasarımı ve uygulanması daha basittir ve daha bütünleşik bir sistem sağlar.
- Monolitik çekirdeklerdeki doğrudan işlev çağrıları, süreçler arası iletişim gerektiren mikro çekirdeklere kıyasla daha iyi performans sağlar.
- Çekirdekteki hatalar tüm sistemi çökertebileceğinden yekpare çekirdekler daha güvenilirdir, ancak her şeyin tek bir modülde olması hataları bulmayı ve düzeltmeyi kolaylaştırır.

Linus Torvalds (Mikro Çekirdek)

- Mikro çekirdekler daha modüler, esnek ve ölçeklenebilirdir ve daha kolay korunabilir ve geliştirilebilir.
- Mikro çekirdekler, herhangi bir bileşendeki bir hatanın verebileceği hasarı sınırlayarak daha kararlı ve güvenli bir sisteme yol açar.
- Modern bilgisayar mimarileri, mikro çekirdeklerin performans sorunlarının üstesinden gelebilir ve mikro çekirdekler, uygun şekilde tasarlandıkları takdirde daha iyi performans sunabilir.
- Linux'un açık kaynak geliştirme modeli, hızla gelişmesine ve yaygın olarak kullanılmasına yardımcı oldu.

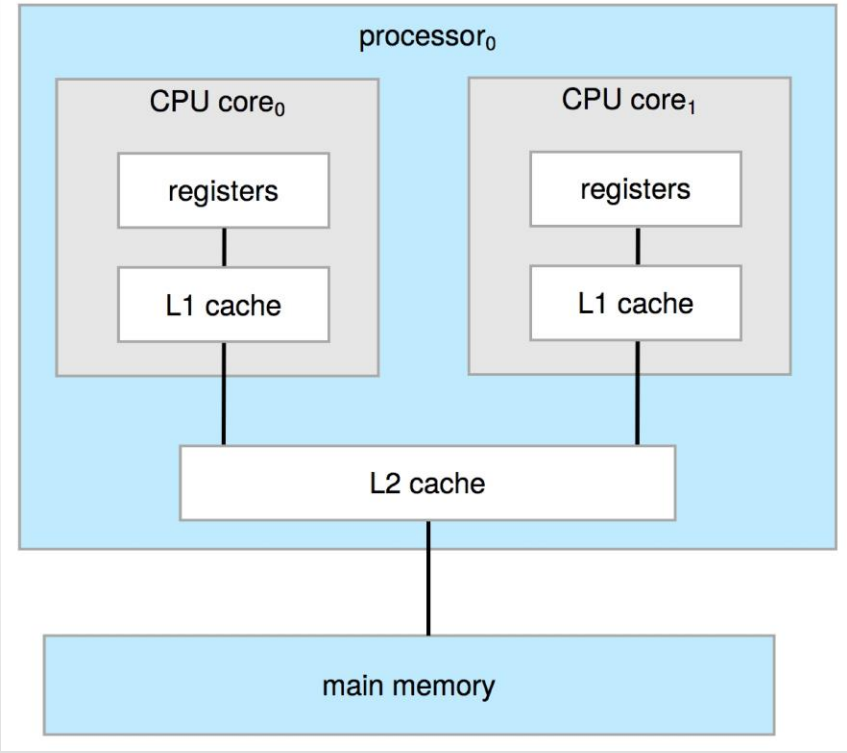
Simetrik Çoklu İşlemci Mimarisi

- .

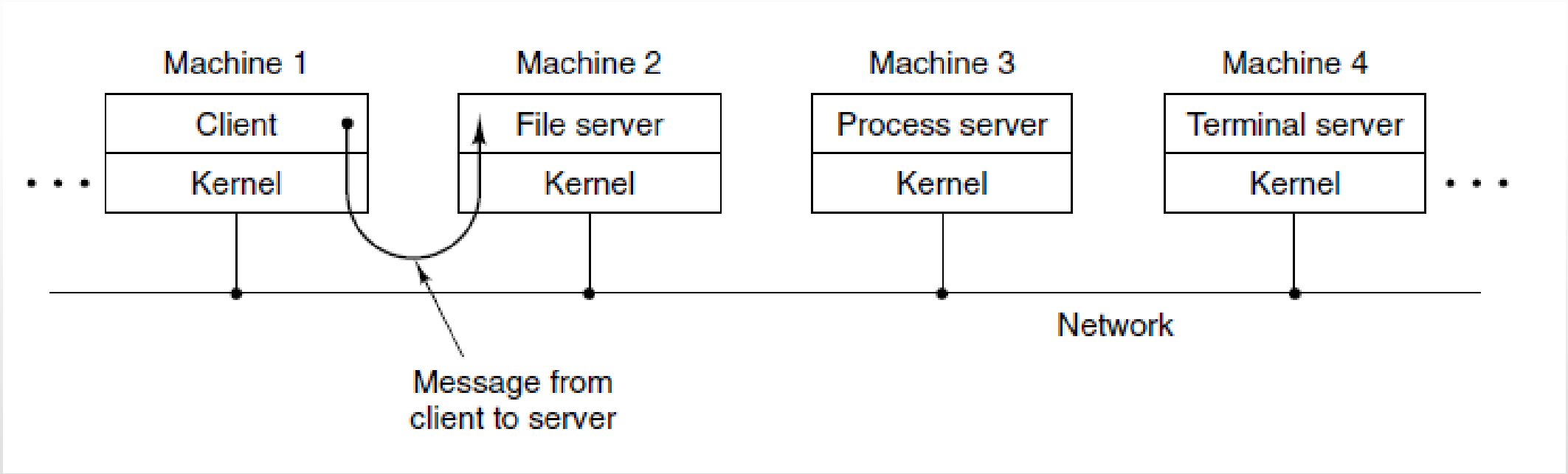


Çift Çekirdekli İşlemci Mimarisi

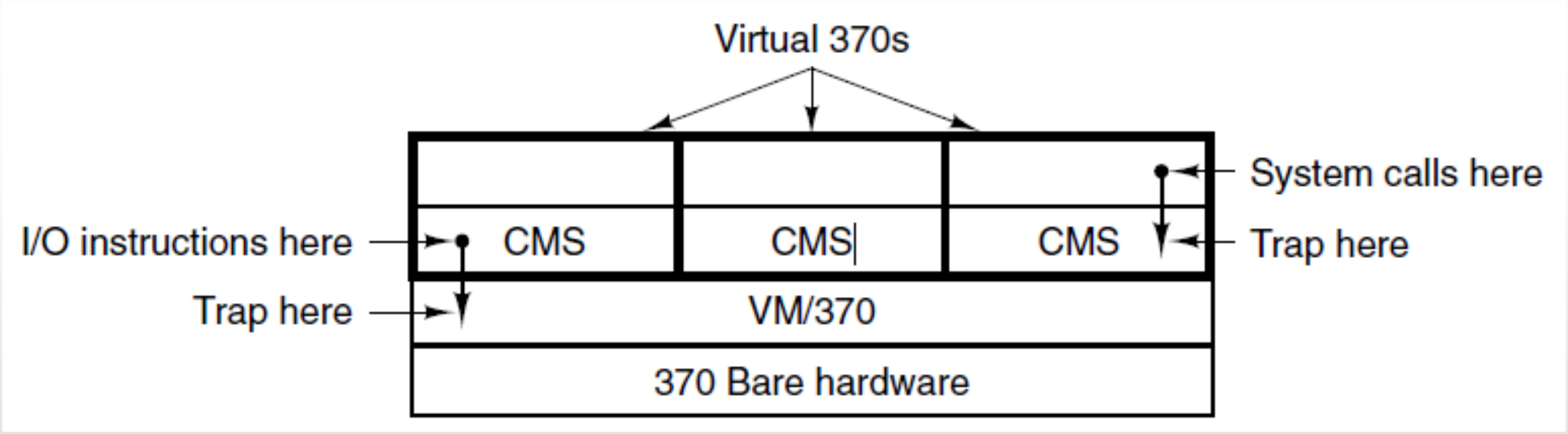
- .



İstemci Sunucu Modeli

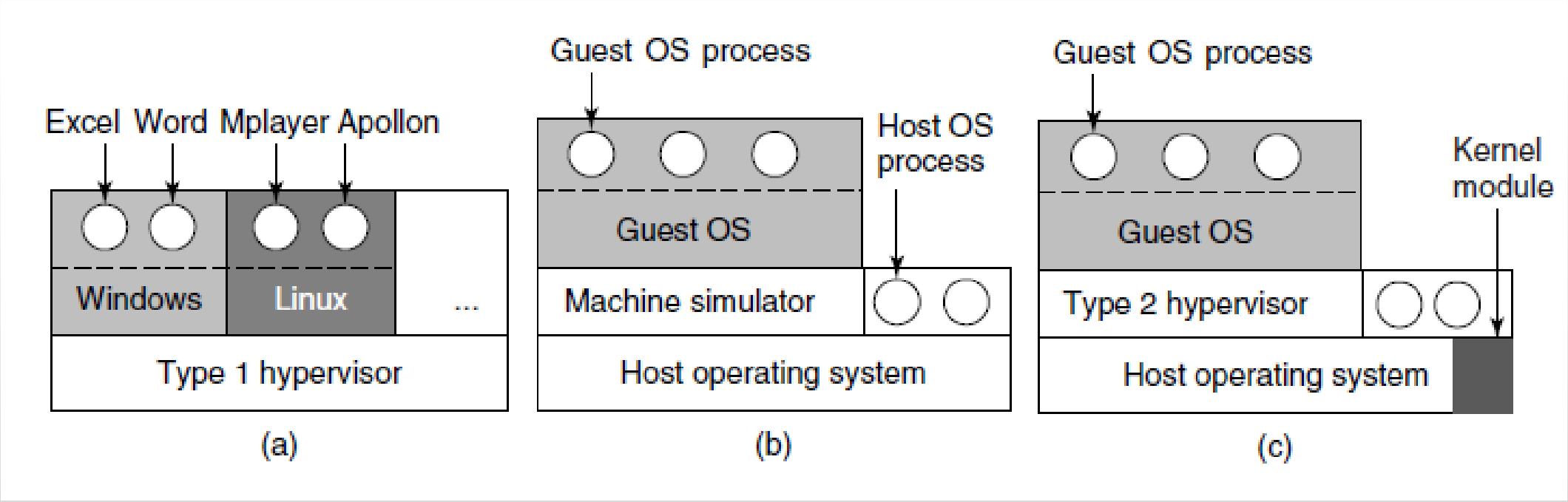


Sanal Makine Yapısı

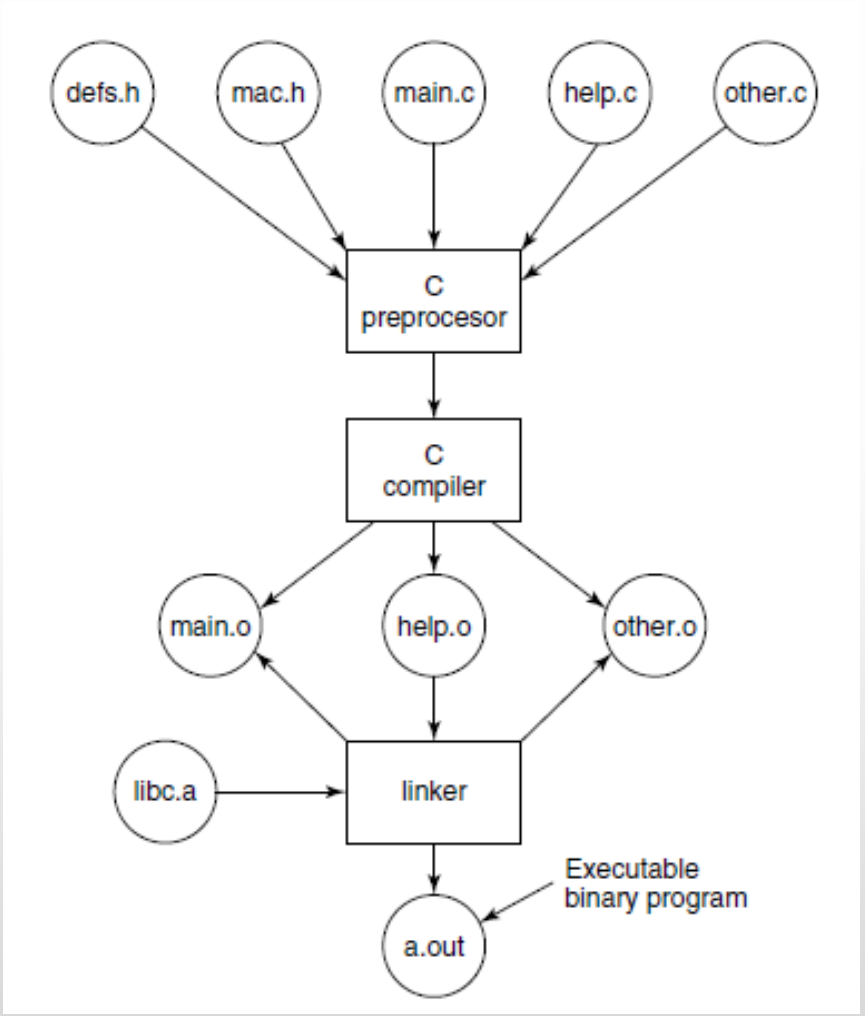


Sanal Makine Yapısı

(a) Tip 1 hipervizör. (b) Yalın tip 2 hipervizör. (c) Pratik tip 2 hipervizör.



Yürütülebilir Dosya Oluşturma

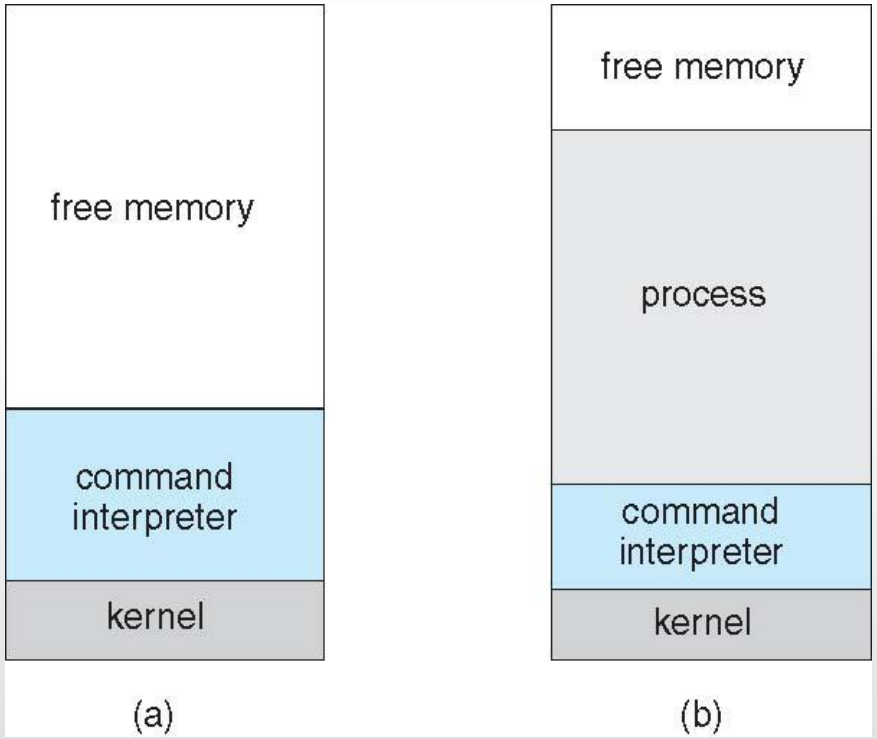


Yürütülebilir Dosya Oluşturma

- C preprocessor (önişlemci):
  - Başlığı alır, makroları genişletir, koşullu derlemeyi ele alır.
- Compiler (derleyici)
  - .c -.o , kaynak koda göre nesne dosyalarını oluşturur.
- Linker (bağlayıcı)
  - .o uzantılı nesne dosyalarını birleştirerek yürütülebilir dosyayı oluşturur.

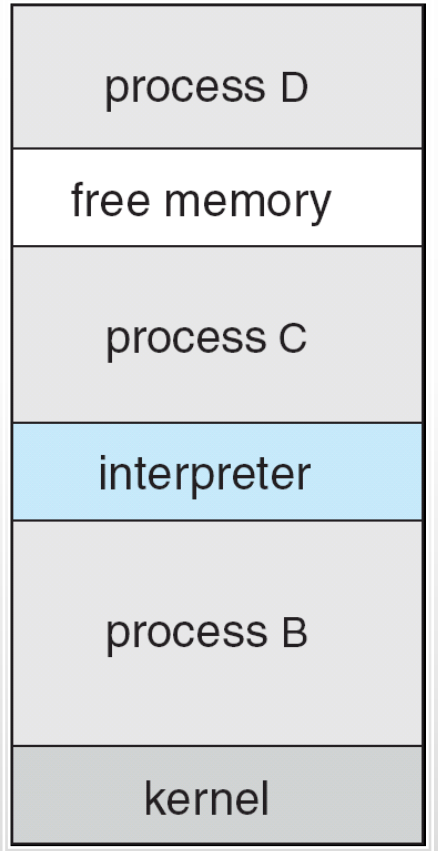
MS-DOS

- Tek görevli
- Sistem ayağa kalktığında kabuk çağrılır
- Süreç yaratılmaz
- Tek bellek alanı
- Program belleğe üzerine yazılarak yüklenir
- Program sonlandığında kabuk yeniden yüklenir
- (a) Sistem ayağa kalkarken
- (b) Program çalışırken



FreeBSD

- UNIX türevi
- Çok görevli
- Kullanıcı giriş yapar ve seçilen kabuk yüklenir
- Kabuk fork() komutu ile süreç yaratır
- exec() komutu ile program süreç içerisine yüklenir
- Kabuk programın sonlanmasını bekler veya kullanıcı komutları ile çalışmaya devam eder.



SON