

Bölüm 2

İşletim Sistemi Yapıları





İşletim Sistemi Kullanıcı Servisleri

- İşletim sistemi servislerinin bir grubu kullanıcıya doğrudan faydalı olan fonksiyonlar sunar:
 - **Kullanıcı arayüzü** – Neredeyse tüm işletim sistemleri bir kullanıcı arayüzüne (UI) sahiptir
 - Değişiklik gösterir: Komut satırı (CLI), Grafiksel Kullanıcı Arayüzü (GUI)
 - **Program çalıştırma** – Sistem, bir programı hafızaya yükleyebilmeli, çalıştırabilmeli ve normal veya anormal (hata durumunda) bir şekilde sonlandırabilmeli
 - **I/O işlemleri** - Çalışan bir program I/O işlemi gerektirebilir – bir dosyaya ya da I/O cihazını kullanmayı gerektirebilir





Kullanıcı Servisleri – Dosya Sistemi

- **Dosya sistemi değişiklikleri** – Programlar aracılığıyla
 - ▶ dosyalar okunabilir,
 - ▶ dosyalara yazabilir,
 - ▶ dosyalar/dizinler oluşturulabilir,
 - ▶ dosyalar/dizinler aranabilir,
 - ▶ dosyalar/dizinler silinebilir,
 - ▶ dosyalar/dizinler listelenebilir,
 - ▶ dosya veya dizinlerin erişim izinleri (permission) değiştirilebilir.





Kullanıcı Servisleri – İletişim ve Hata

- **İletişim** – İşlemler bilgi alış verişinde bulunabilirler – aynı bilgisayar üzerinde veya ağ üzerinde bulunan farklı bilgisayarlar üzerinde
 - İletişim **ortak hafıza (shared memory)** ile veya **mesaj gönderimi (message passing)** ile sağlanabilir.
 - Paketler işletim sistemi tarafından taşınır
- **Hata tespiti** – İşletim sistemi sürekli olarak olası hataları takip eder
 - Hatalar CPU, hafıza donanımı, I/O cihazları veya kullanıcı programı kaynaklı olabilir
 - Her bir hata tipi için, işletim sistemi uygun olan işlemi gerçekleştirerek bilgisayarın doğru ve tutarlı şekilde çalışmasına devam etmesini sağlamalıdır
 - Hata ayıklamak için sunulan mekanizmalar, sistemin verimli şekilde kullanımı için kullanıcıya ve programcıya sağlanmalıdır





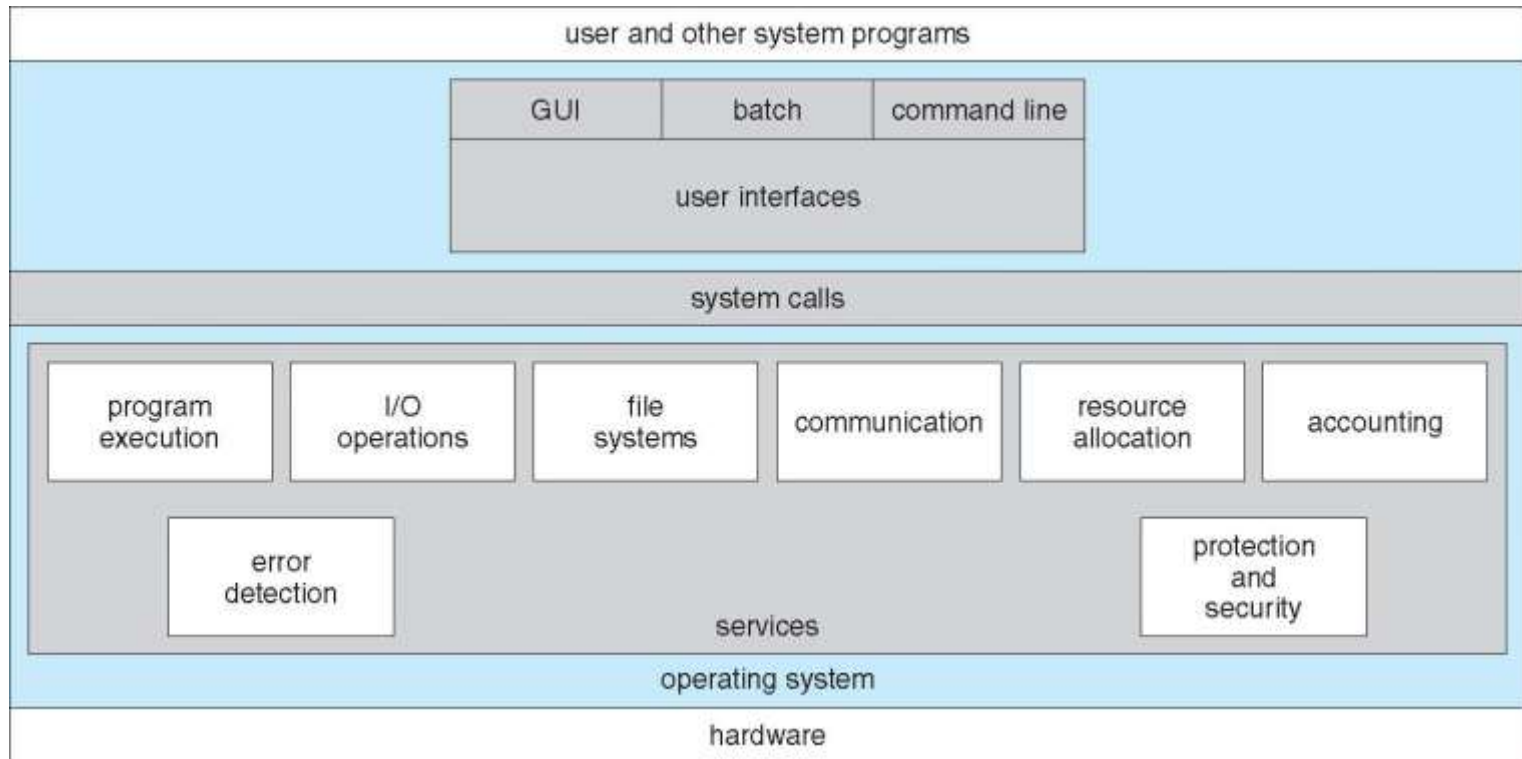
Kaynak Paylaşım Servisleri

- İşletim sistemi servislerinin bir diğer grubu kaynakların paylaşımını sağlayarak bilgisayarın etkili bir şekilde kullanımını sağlar
 - **Kaynak paylaşdırma** – Birden fazla kullanıcı ya da birden fazla iş aynı anda çalıştırıldığında kaynaklar her birine adil şekilde paylaşdırılmalıdır
 - ▶ Perk çok farklı tipte kaynak
 - Bazıları özel pay alma kodları kullanır: CPU zamanı, ana hafıza, dosya kayıt birimi
 - Bazıları ise isteme ve iade etme kodları kullanır: I/O cihazları
 - **Hesap tutma** – Hangi kullanıcının hangi tip sistem kaynağının ne kadarını kullandığını takip etmek
 - **Koruma ve güvenlik** – Çok kullanıcılı bir sistemde bir bilginin sahibi bu bilginin kim tarafından kullanılabileceğini kontrol etmek ister, aynı anda çalışan işlemlerin birbirine müdahalesi engellenmelidir
 - ▶ **Koruma** - sistem kaynaklarına her tür erişimin kontrol altında tutulmasını gerektirir
 - ▶ **Güvenlik** - sisteme dışarıdan erişmek isteyenlerin kimlik doğrulamasını yapmaktan I/O cihazlarının geçersiz erişim isteklerine engel olunmasına kadar çeşitlilik gösterir
 - ▶ Bir zincir en fazla en zayıf halkası kadar güçlüdür!





İşletim Sistemi Servislerine Genel Bakış





İşletim Sistemi İşlem Arayüzü- CLI

- Komut Satırı Arayüzü - Command Line Interface (CLI)
- Veya **komut yorumlayıcısı (command interpreter)** direk komut girişini sağlar
 - Bazen çekirdeğin parçası olarak gerçekleştirilir, bazen sistem programı olarak.
 - Bazen farklı özellikleri barındıran farklı versiyonları bulunur – **kabuklar (shells)**
 - Temel olarak, kullanıcıdan bir komut alır ve bunu çalıştırır
 - Bazen komutlar kabuğun bir parçasıdır, bazen programların adıdır
 - Eğer ikincisi ise, yeni komutların eklenmesi kabuğun güncellenmesini gerektirmez





Kullanıcı İşletim Sistemi Arayüzü - GUI

- Kullanıcı dostu **masaüstü (desktop)** benzetmesi kullanan arayüz
 - Genellikle fare, klavye ve bilgisayar ekranı kullanılır
 - **Simgeler (icons)** dosyaları, programları, eylemleri vs. ifade eder
 - Nesneler üzerinde fare tıklamaları pek çok eylemi tetikler: bilgi sağlama, fonksiyon çalıştırma, izin açma vs.
 - İlk olarak Xerox PARC ile kullanılmıştır
- Pek çok sistem CLI ve GUI arayüzlerini birlikte sunar
 - Microsoft Windows grafiksel arayüze ek olarak CLI komut kabuğu sunar
 - Apple Mac OS X “Aqua” GUI arayüzüne ek olarak alt katmanlarda UNIX çekirdeği ve kabuklarını bulundurur
 - Solaris CLI arayüzüne ek olarak opsiyonel olarak GUI arayüzleri kullanılabilir (Java Desktop, KDE)





Bourne Kabuğu – Komut Yorumlayıcısı

```

Terminal
File Edit View Terminal Tabs Help
fd0      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
sd0      0.0    0.2    0.0    0.2  0.0  0.0    0.4  0  0
sd1      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
          extended device statistics
device   r/s    w/s    kr/s    kw/s wait actv  svc_t  %w  %b
fd0      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
sd0      0.6    0.0   38.4    0.0  0.0  0.0    8.2  0  0
sd1      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
(root@pbg-nv64-vn)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vn)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
12:53am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vn)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
4:07pm up 17 day(s), 15:24, 3 users, load average: 0.09, 0.11, 8.66
User      tty          login@ idle   JCPU   PCPU   what
root      console      15Jun07 18days 1      /usr/bin/ssh-agent -- /usr/bi
n/d
root      pts/3        15Jun07 18      4      w
root      pts/4        15Jun07 18days w
(root@pbg-nv64-vn)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#

```





The Mac OS X GUI





System Çağrıları

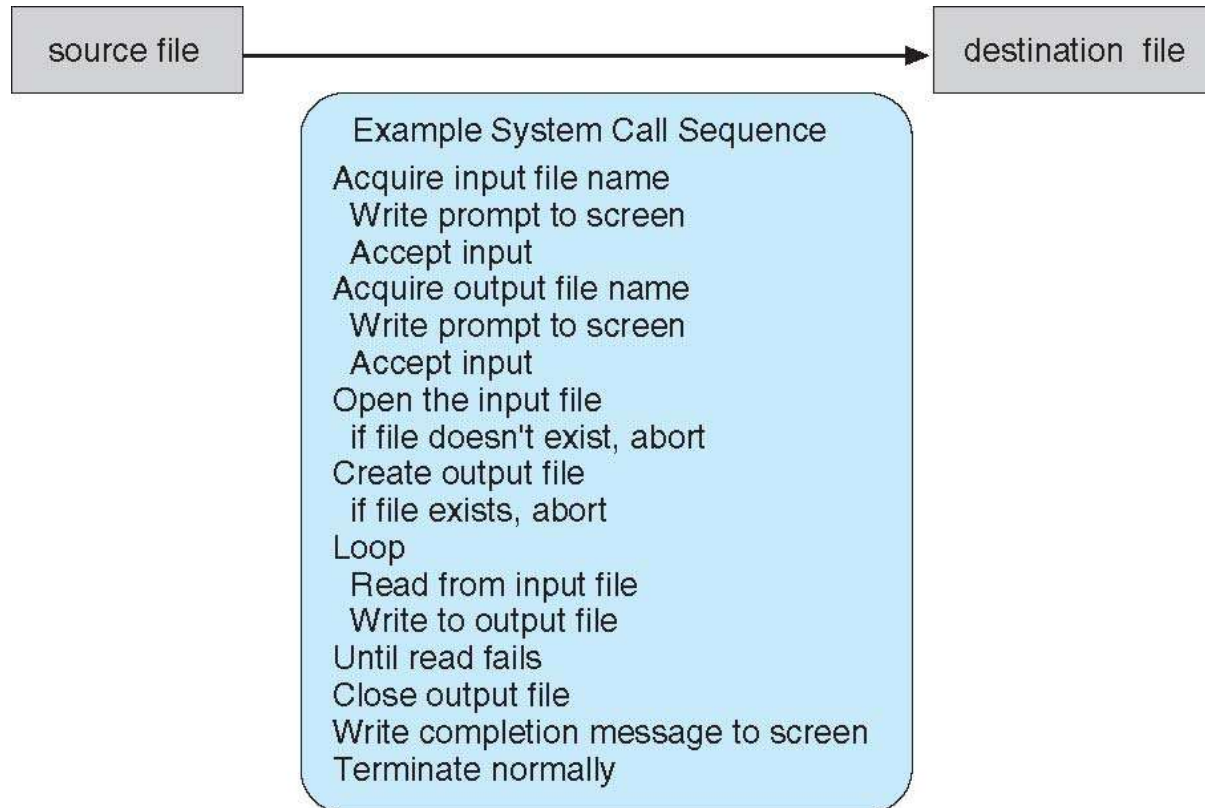
- İşletim sistemi tarafından sunulan servisler için programlama arayüzüdür
- Genellikle üst seviye dillerde yazılır (C veya C++)
- Genellikle sistem çağrıları direk çağırılmazlar. Bunun yerine üst seviye **Uygulama Programı Arayüzü (Application Program Interface - API)** aracılığıyla kullanılırlar
- En çok kullanılan API'ler:
 - Windows için Win32 API
 - POSIX tabanlı sistemler için (UNIX, Linux, ve Mac OS X'in hemen hemen bütün versiyonları) POSIX API
 - Java Sanal Makinası (JVM) için Java API
- Sistem çağrıları yerine neden API'ler kullanılır? taşınabilirlik ve basitlik





Sistem Çağrısı Örneği

- Bir dosyanın içeriğini başka bir dosyaya kopyalayan sistem çağrıları





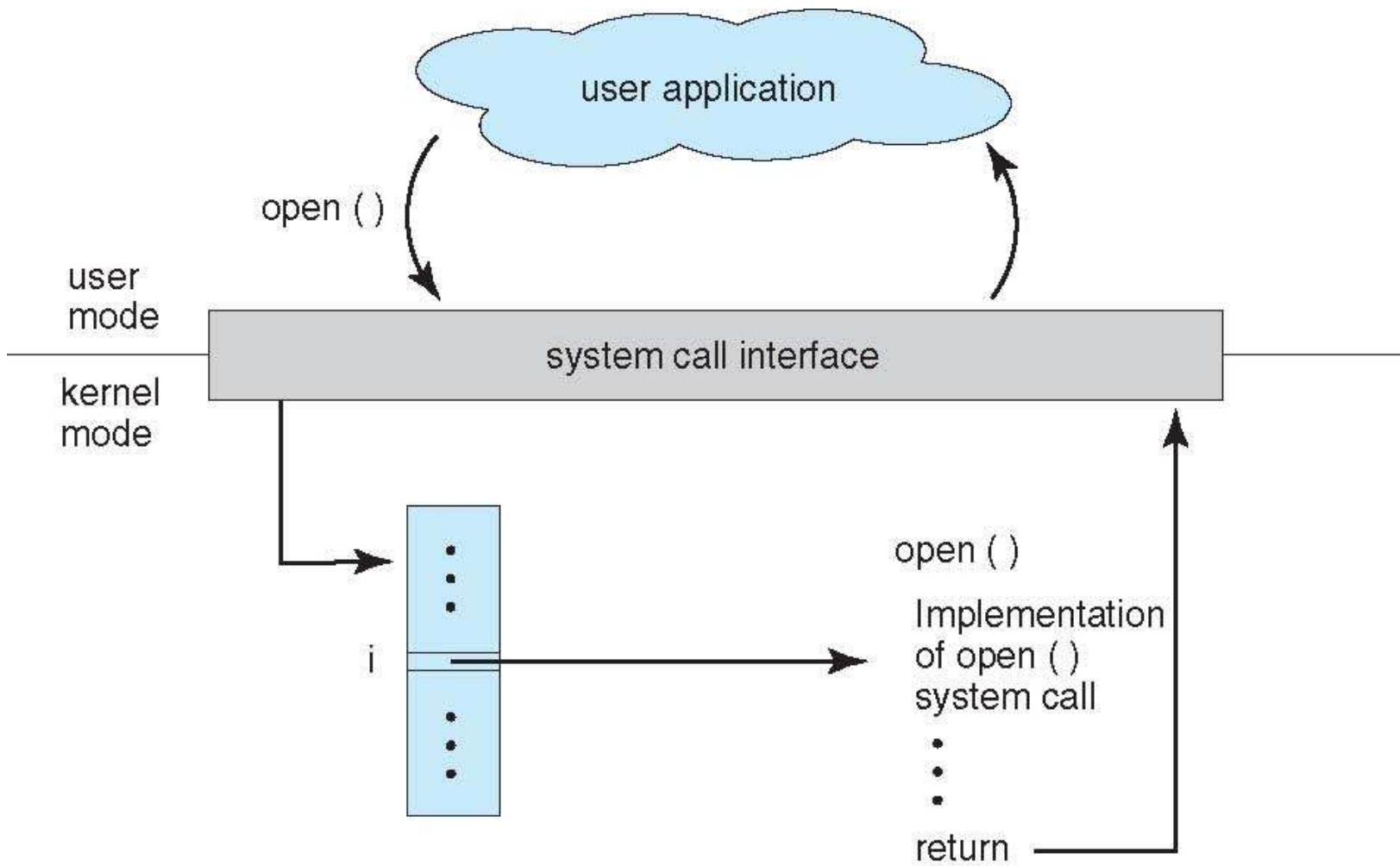
Sistem Çağrısı Gerçekleştirimi

- Genellikle her bir sistem çağrısı ile bir sayı ilişkilendirilir
 - Sistem çağrısı arayüzü bu sayılarla indekslenmiş bir tablo tutar
- Sistem çağrısı arayüzü, işletim sistemi kabuğunda gerçekleştirilmiş sistem çağrısını çağırır ve eğer dönen bir bilgi varsa bu bilgi ile sistem çağrısı durumunu geri döndürür
- Sistem çağrısını çağırان uygulama sistem çağrısının nasıl gerçekleştirildiğini bilmek zorunda değildir.
 - İşletim sistemi arayüzünün pek çok detayı API ile programcıdan gizlenir





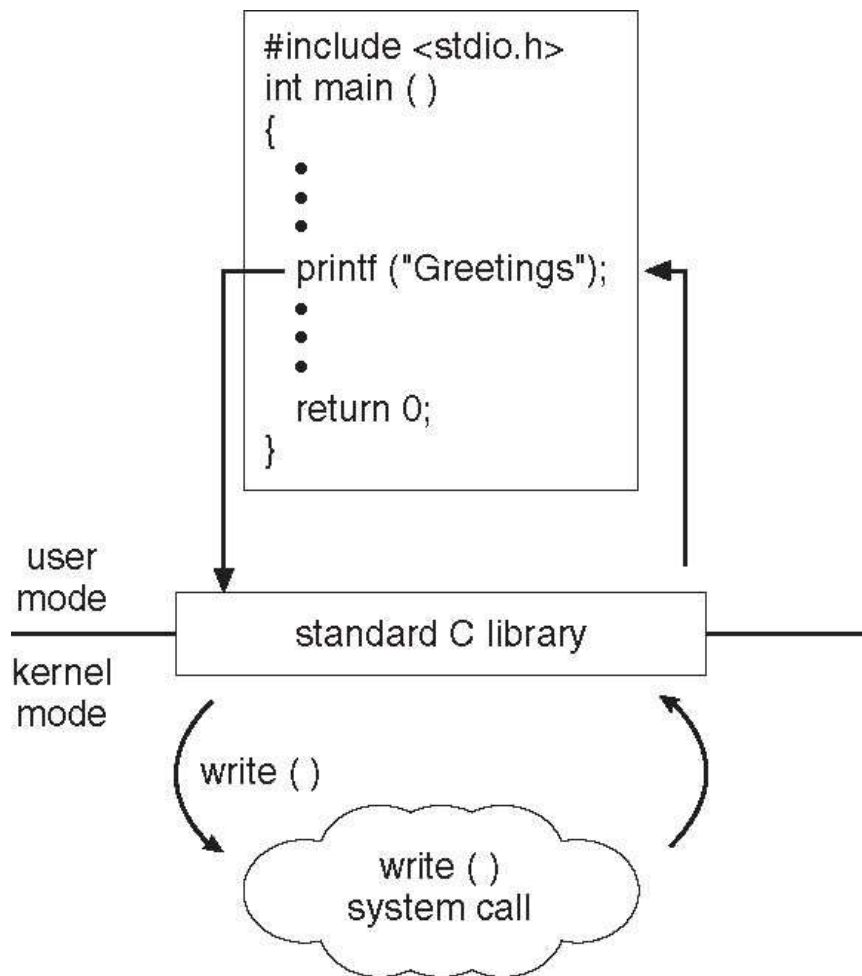
API – Sistem Çağrısı – OS İlişkisi





Standart C Kütüphanesi Örneği

- printf() kütüphane fonksiyonunu çağıran program örneği – printf fonksiyonu arka planda write() sistem çağrısını kullanır





Sistem Çağrısı Çeşitleri

- İşlem kontrolü
- Dosya yönetimi
- Cihaz yönetimi
- Bilgi sağlama
- İletişim
- Koruma





Windows ve Unix Sistem Çağrısı Örnekleri

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()





Sistem Programları

- Sistem programları program geliştirmek ve çalıştırmak için rahat bir ortam sağlarlar.
- Aşağıdaki gruplara ayrılabilirler
 - Dosya manipülasyonu
 - Durum bilgisi
 - Dosya değiştirme
 - Programlama dili desteği
 - Program yükleme ve çalıştırma
 - İletişim
 - Uygulama programları
- İşletim sisteminin pek çok kullanıcısı işletim sistemini, sağlanan sistem programları aracılığıyla bilir ve sistem çağrılarından habersizdir





Sistem Programları (devam)

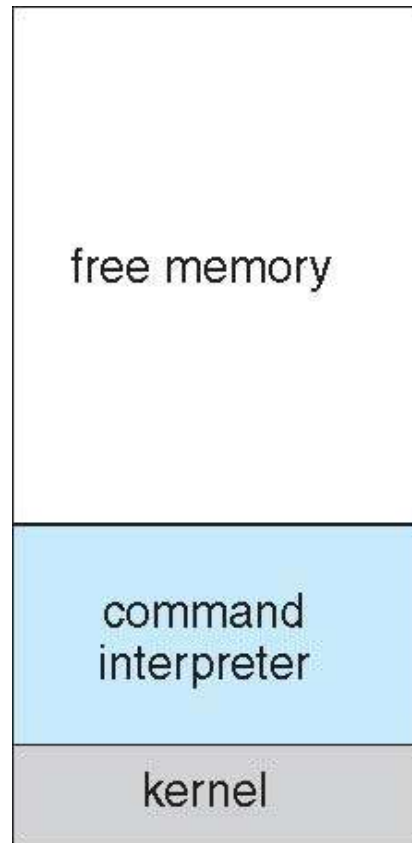
■ Durum bilgisi

- Bazıları sistemden aşağıdaki bilgileri ister
 - Tarih
 - Saat
 - Kullanılabilir hafıza miktarı
 - Kullanılabilir disk alanı
 - Kullanıcı sayısı
- Diğerleri performans, kayıtlar (logging) ve hata ayıklama (debugging) bilgileri ister
- Tipik olarak, bu programlar elde edilen bilgiyi uygun formata getirip terminale veya diğer çıktı cihazlarına yazdırır
- Bazı sistemler konfigürasyon bilgisini tutmak ve kullanmak için bir kayıt ortamı (registry) kullanır





MS-DOS ile Program Çalıştırma



(a)



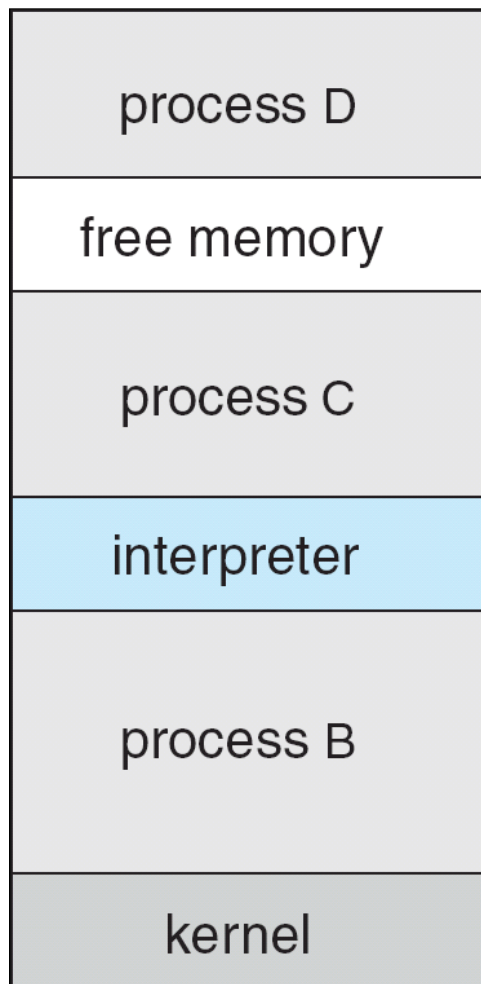
(b)

(a) Sistem başlangıcında (b) Bir program çalışırken





FreeBSD ile Çoklu Program Çalıştırma





İşletim Sistemi Tasarımı ve Gerçekleştirimi

- *Kullanıcı hedefleri ve Sistem hedefleri*
 - **Kullanıcı hedefleri** – işletim sistem rahat kullanılmalı, kolayca öğrenilmeli, tutarlı, güvenli ve hızlı olmalı
 - **Sistem hedefleri** – işletim sistemi kolay tasarlanmalı ve gerçekleştirilmeli, bakımı kolay yapılmalı. Esnek olmalı, tutarlı, hatasız ve verimli çalışmalı





İlkeler ve Mekanizma

- İlkeler ve mekanizmayı birbirinden ayırmak önemli bir prensiptir
 - İlke (policy): Ne yapılmalı?
 - Mekanizma (Mechanism): Nasıl yapılmalı?
- Mekanizmalar bir şeyin nasıl yapılacağını belirler. İlkeler ise ne yapılması gerektiğine karar verir
 - İlkelerin mekanizmadan ayrılması maksimum esneklik sağlar
 - Daha sonra ne ilkeler değişiklik gösterirse sistemin güncellenmesini kolaylaştırır





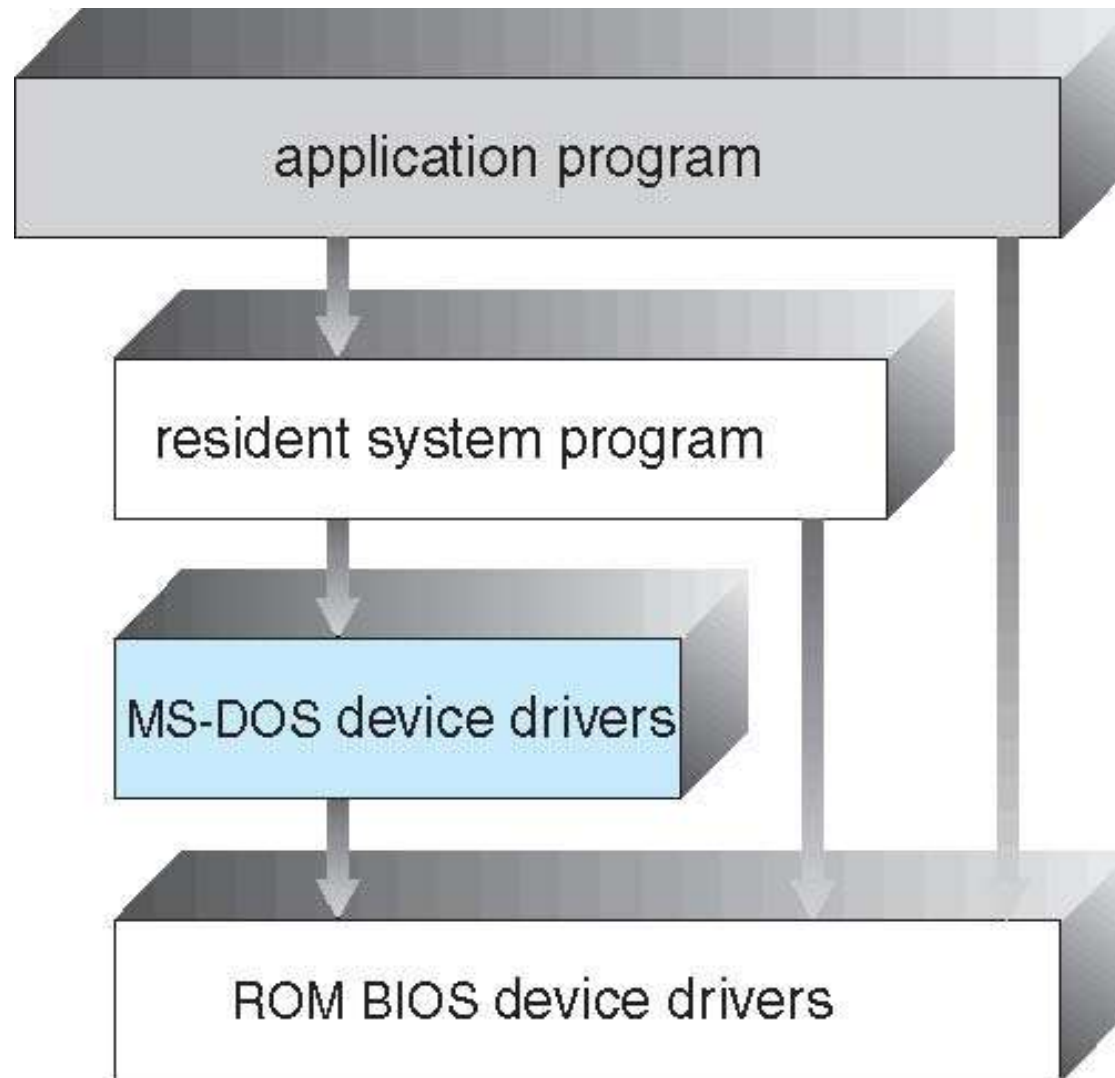
Basit Yapı

- MS-DOS – an az alanda en çok fonksiyonu sağlamak için yazılmıştır
 - Modüllere ayrılmamıştır
 - MS-DOS belli bir yapıya sahip olsa da arayüzleri ve fonksiyonlarının saviyeleri iyi ayrılmamıştır





MS-DOS Katman Yapısı





Katmanlı Yaklaşım

- İşletim sistemi belli sayıda **katmana (layers)** ayrılır.
- Her katman alt seviyedeki diğer katman(lar)ın üzerine yerleşir.
- En alt katman (layer 0) donanım katmanıdır.
- En üst katman ise (layer N) kullanıcı arayüzüdür.
- Sistemin modüler olması için katmanlar şu kriteri sağlayacak şekilde seçilirler:
 - Üst katmanlar sadece altındaki katman(lar)ın fonksiyonlarını ve servisleri kullanmalıdır





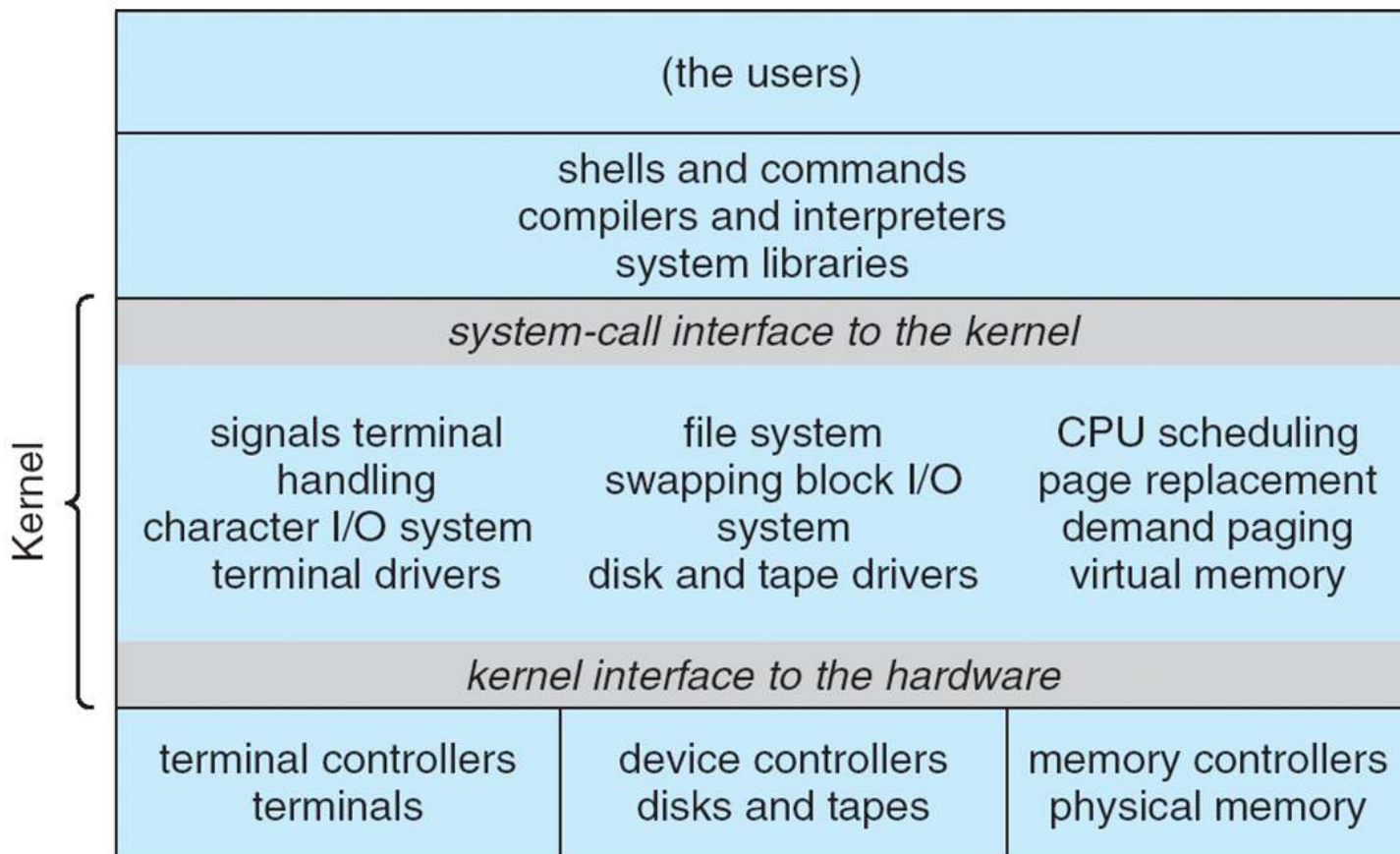
UNIX

- UNIX : İki tane birbirinden ayrılabilir parçaya sahip
 - Sistem programları
 - Çekirdek
 - ▶ Fiziksel donanımın üstünde ve sistem çağrısı arayüzünün altında herşey
 - ▶ Dosya sistemi, CPU zamanlaması, hafıza yönetimi ve diğer işletim sistemi fonksiyonlarını sağlıyor
 - ▶ Tek seviyede çok sayıda fonksiyonu barındırıyor



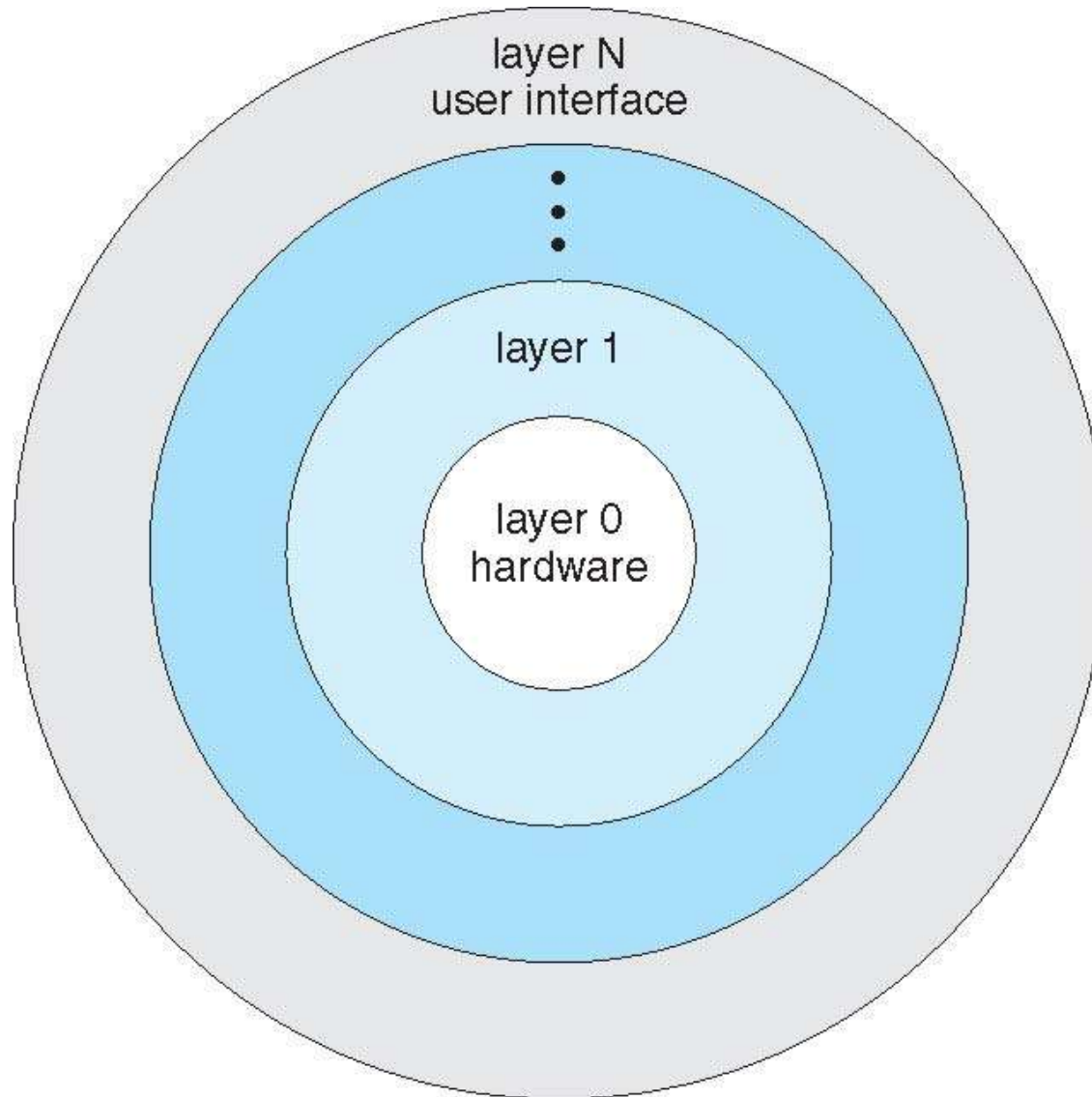


Geleneksel UNIX Sistem Yapısı





Katmanlı İşletim Sistemi





Mikroçekirdek Sistem Yapısı

- **Mikro-çekirdek (microkernel) Sistem Yapısı**
- Çekirdekte gerçekleşen şeylerden taşınabilir olan herşeyi «kullanıcı» alanına (moduna) taşıyor
- Modüller arasındaki iletişim mesaj gönderme (message passing) yoluyla gerçekleşiyor
- Faydalar:
 - Mikroçekirdeğin özelliklerini arttırmak daha kolay
 - İşletim sistemini yeni mimarilere geçirmek daha kolay
 - Daha tutarlı (çekirdek modunda çok daha az kod çalışıyor)
 - Daha güvenli
- Zararlar:
 - Çekirdek modu ile kullanıcı modu arasında iletişimin getirdiği aşırı yüklenmenin getirdiği performans sorunu





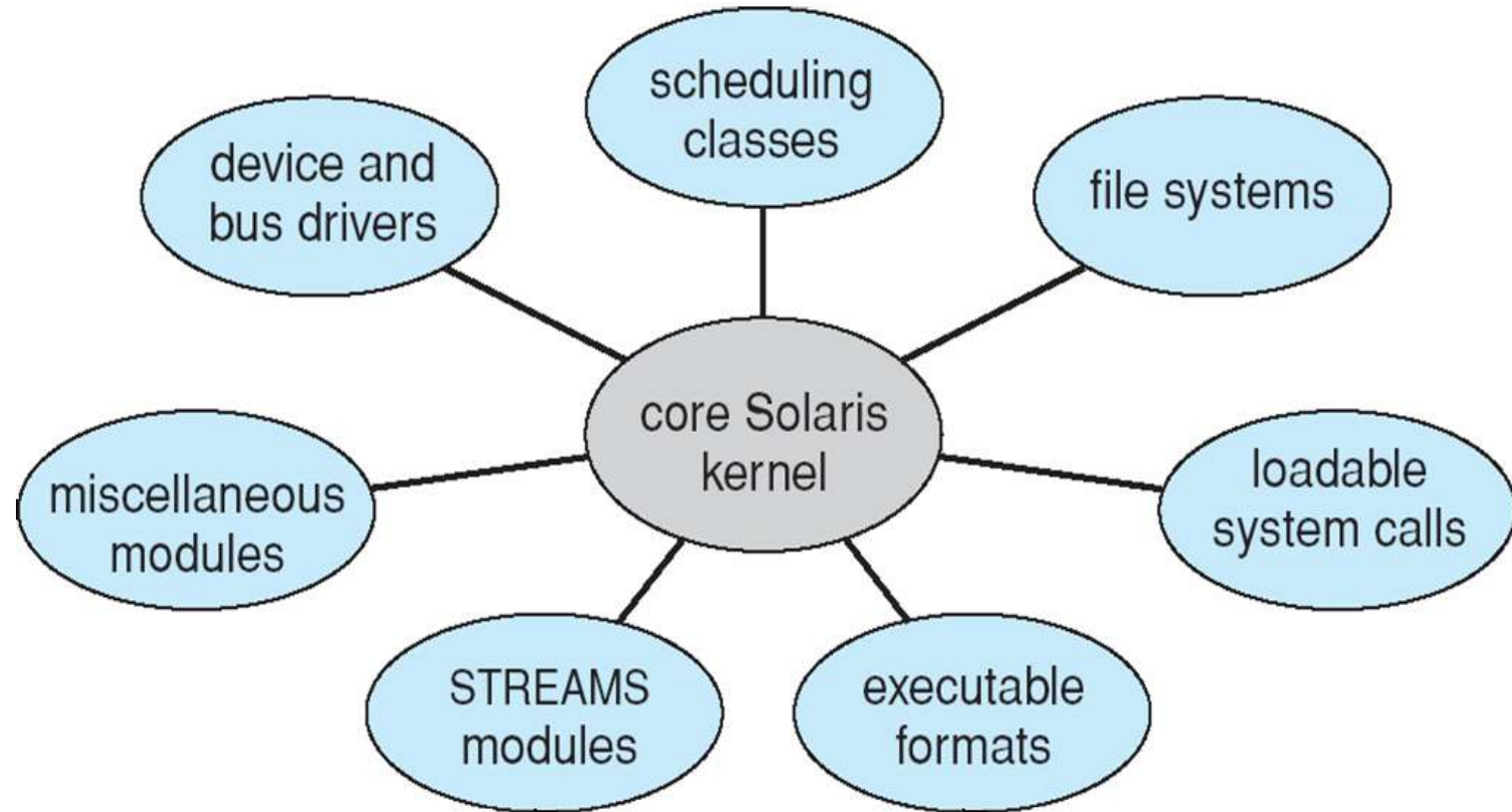
Modüller

- Perk çok işletim sistemi **çekirdek modüllerini** kullanır
 - Nesneye yönelik yaklaşımı kullanır
 - Tüm temel bileşenler ayrıdır
 - Birbirleriyle belirli arayüzler üzerinden iletişim kurarlar
 - Her biri gerektiğinde çekirdeğe yüklenebilir durumdadır
- Katmanlara benzerler ama daha çok esneklik sağlarlar





Solaris Modüler Yaklaşımı





Sanal Makinalar

- **Sanal makinalar (virtual machines)** katmanlı yaklaşımı kullanır
- Donanımı ve işletim sistemi çekirdeğini donanım katmanıymış gibi kullanırlar
- Sanal makinalar alt seviyedeki donanım ne olursa olsun, üst seviyedeki programlara aynı arayüzü sunarlar
- İşletim sisteminin **ev sahibi (host)** işlemlere kendi işlemcisi ve sanal hafızası varmış izlenimi verir
- Tüm **konuklara (guest)** üzerinde çalışılan bilgisayarın sanal bir kopyası sağlanır





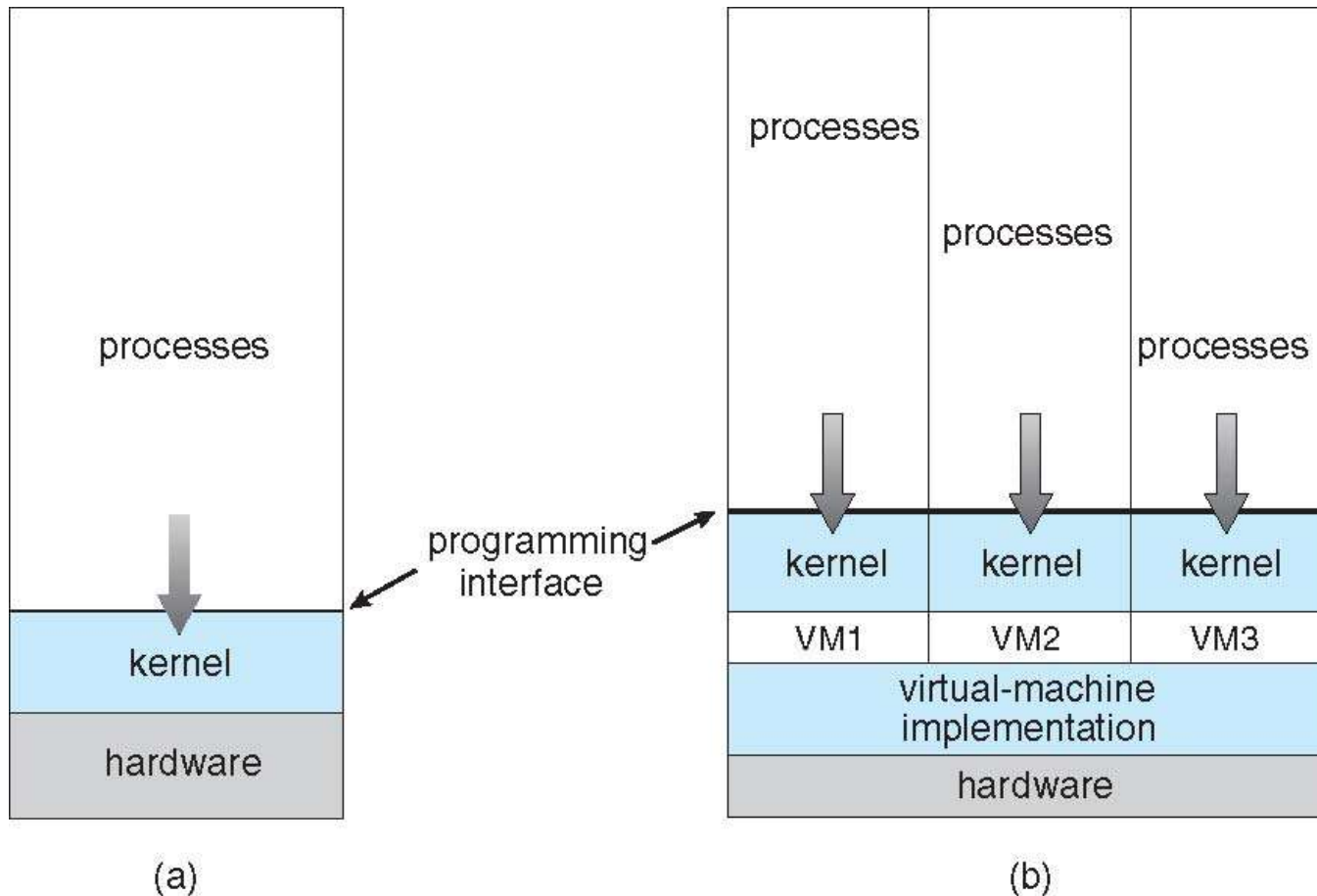
Sanal Makinalar - Tarihçe ve Faydalar

- İlk olarak IBM anabilgisayarlarında (mainframes) 1972 yılında kullanılmaya başlandı
- Temel olarak, birden fazla çalıştırma ortamı (farklı işletim sistemi) aynı donanımı paylaşabilir ve birbirlerinden korunurlar
- Dosya paylaşımı sağlanabilir (kontrollü bir şekilde)
- Bilgisayar ağları aracılığıyla birbirleriyle ve diğer bilgisayar sistemleriyle etkileşime geçebilirler
- Sistem geliştirmek ve test etmek için kullanışlıdır
- Az kullanılan birden fazla işletim sistemini bir araya getirerek sistem kaynaklarının daha etkili kullanımını sağlar
- **“Açık Sanal makina formatı”** (Open Virtual Machine Format) – Sanal makinaların farklı sanal makine (host) platformlarında çalışabilmesini sağlayan standart





Sanal Makinalar (devam)

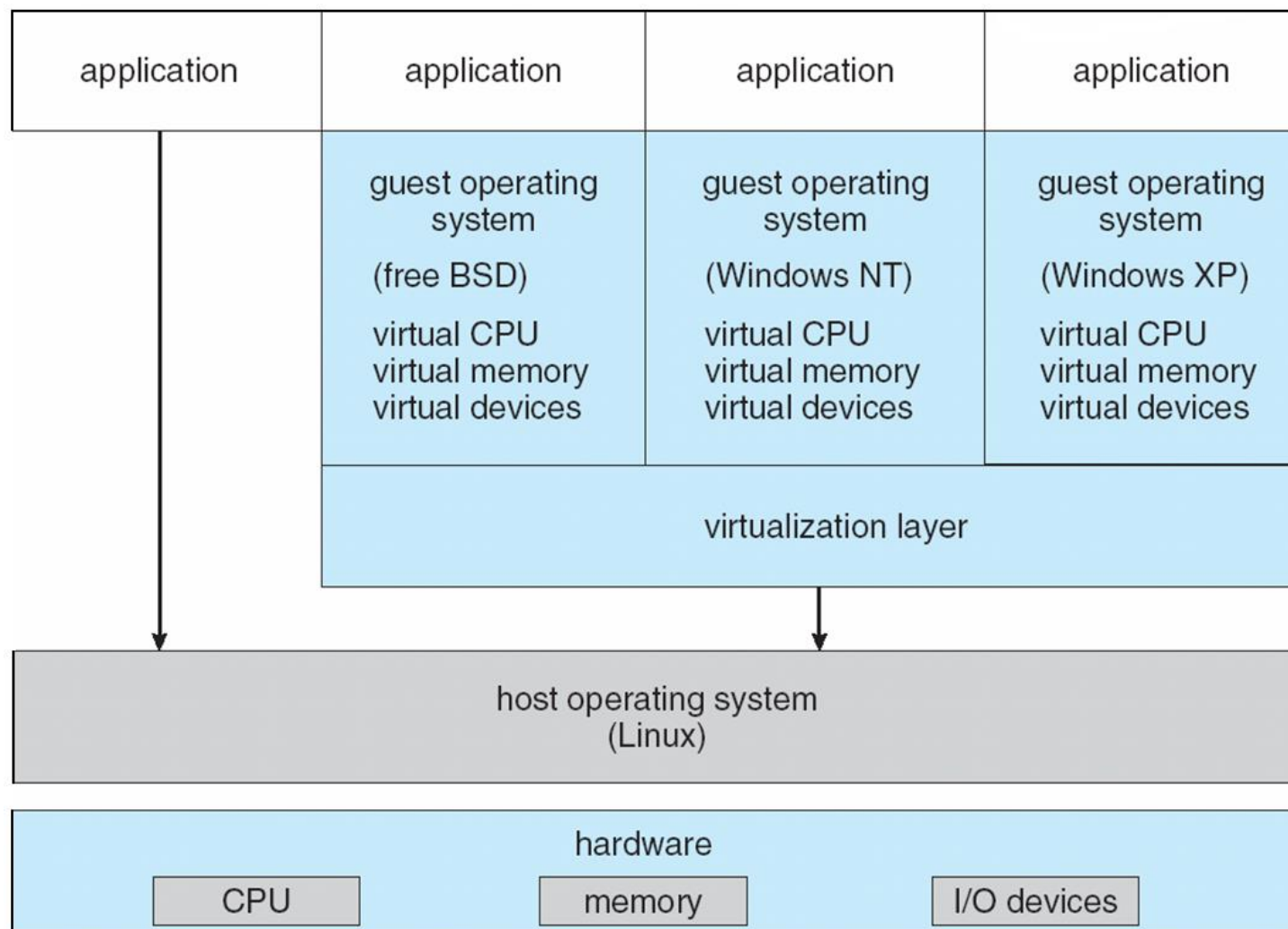


(a) Sanal olmayan makine (b) sanal makina





VMware Architecture





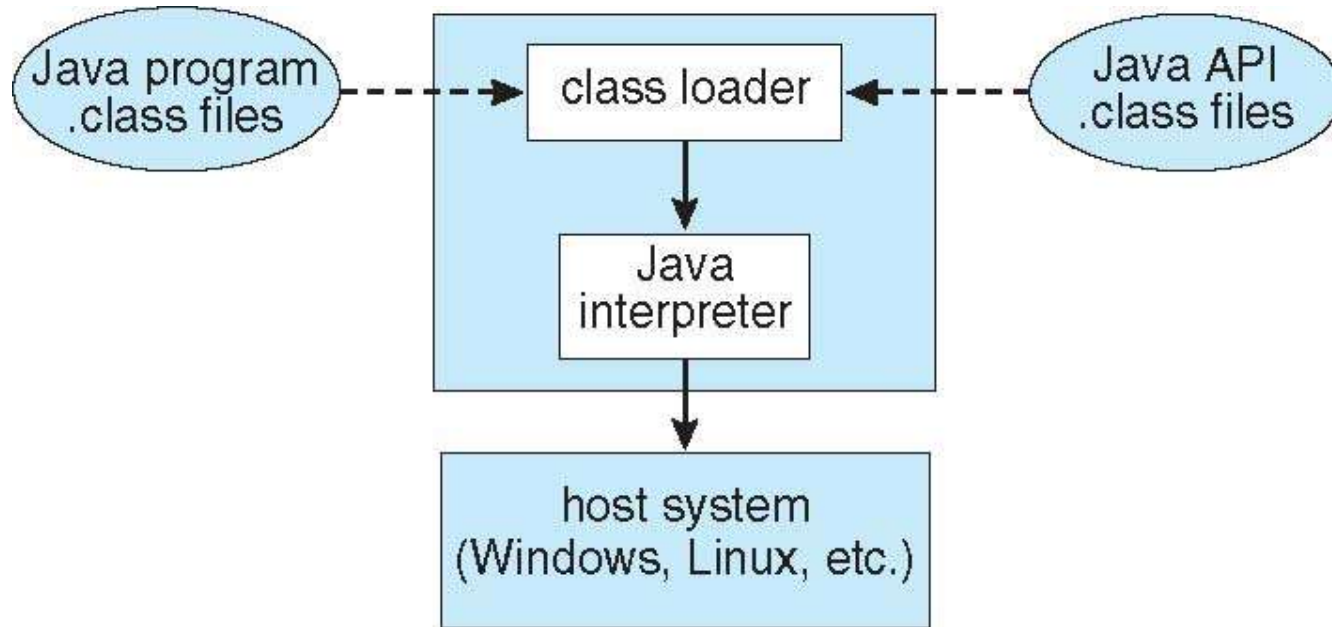
Java

- Java:
 1. Programlama dili spesifikasyonu
 2. Uygulama programlama arayüzü (API)
 3. Sanal makine spesifikasyonu





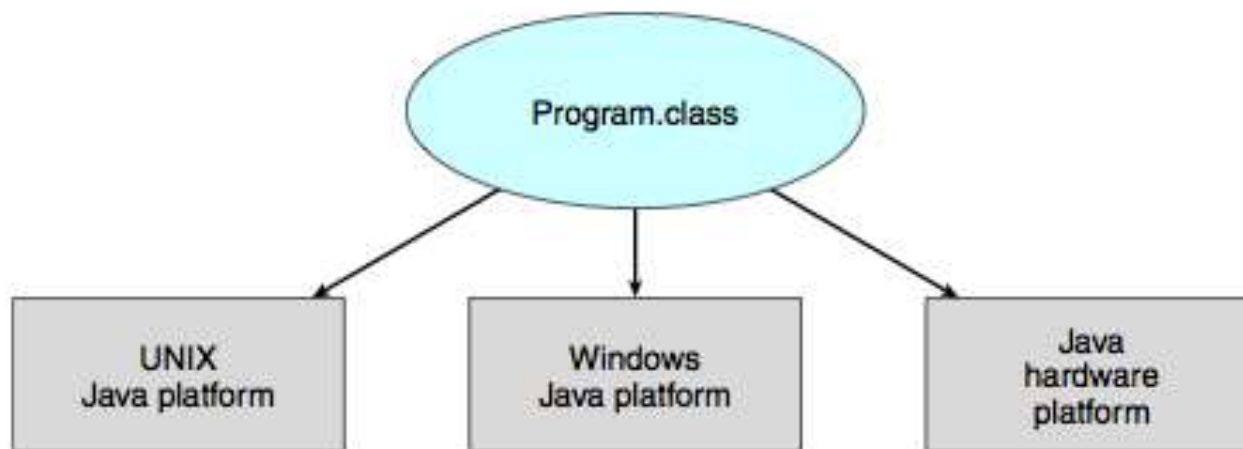
Java Sanal Makinası





JVM - Taşınabilirlik

Java'nın farklı platformlarda taşınabilirliği





Java Geliştirme Aracı

