Bölüm 2
İşletim Sistemi Yapıları

Bölüm 2: İşletim Sistemi Yapıları

- İşletim Sistemi Servisleri
- Kullanıcı İşletim Sistemi Arayüzü
- Sistem Çağrıları
- Sistem Çağrısı Tipleri
- Sistem Programları
- İşletim Sistemi Tasarım ve Gerçekleştirimi
- İşletim Sistemi Yapısı
- Sanal Makinalar
- İşletim Sistemi Oluşturma
- Sistem Önyükleme

Hedefler

- İşletim sisteminin, kullanıcılara, işlemlere ve diğer sistemlere sağladığı servisleri açıklamak
- Bir işletim sistemini gerçeklemenin farklı şekillerini açıklamak
- İşletim sistemlerinin nasıl kurulduğunu, özelleştirildiğini ve önyüklendiğini açıklamak

İşletim Sistemi Kullanıcı Servisleri

- İşletim sistemi servislerinin bir grubu kullanıcıya doğrudan faydalı olan fonksiyonlar sunar:
  - **Kullanıcı arayüzü (Shell)** – 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.

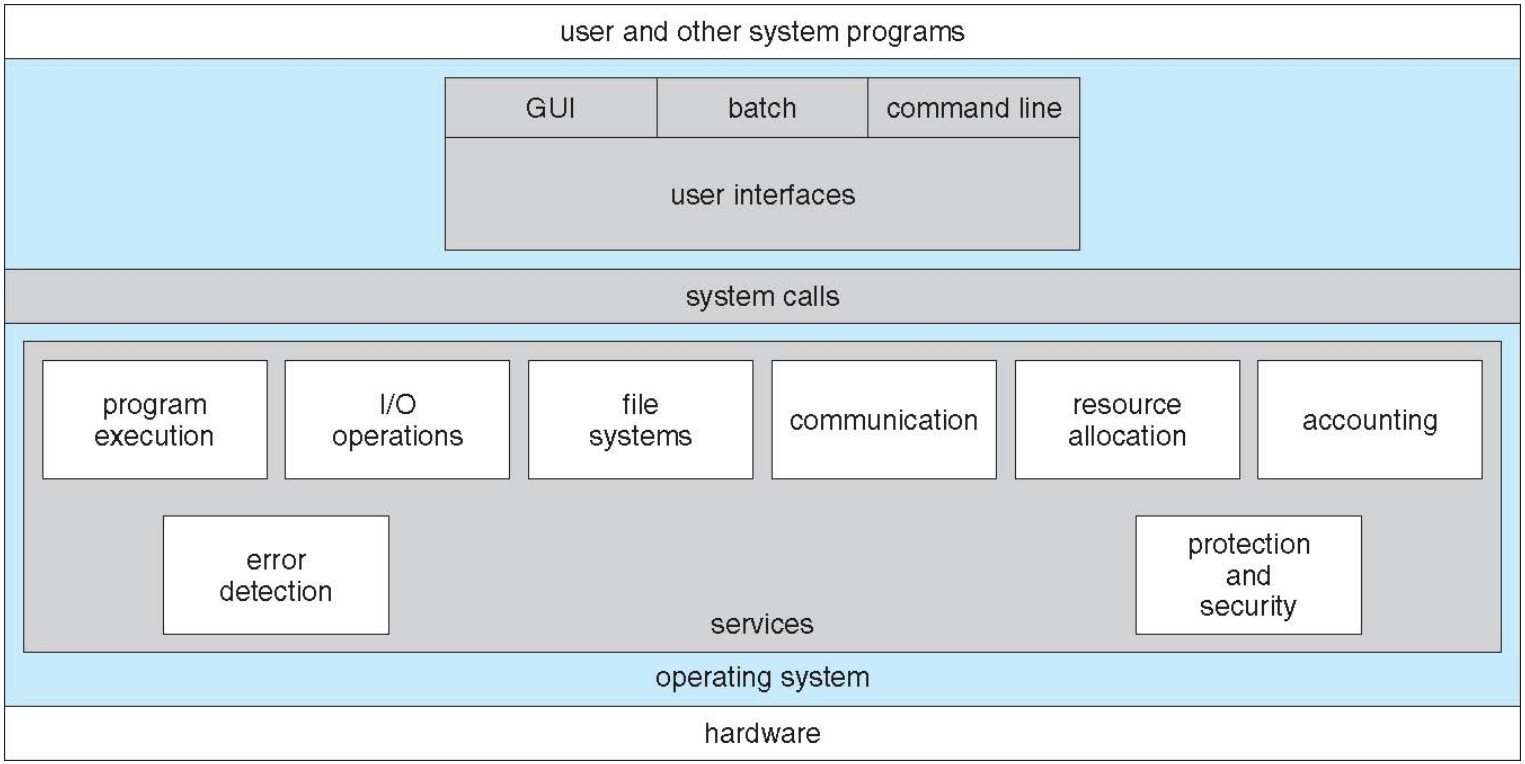
Kullanıcı Servisleri – İletişim ve Hata

- **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 paylaştırma** – Birden fazla kullanıcı ya da birden fazla iş aynı anda çalıştırıldığında kaynaklar her birine adil şekilde paylaştırılmalıdır
    - Pek 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 atlında 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



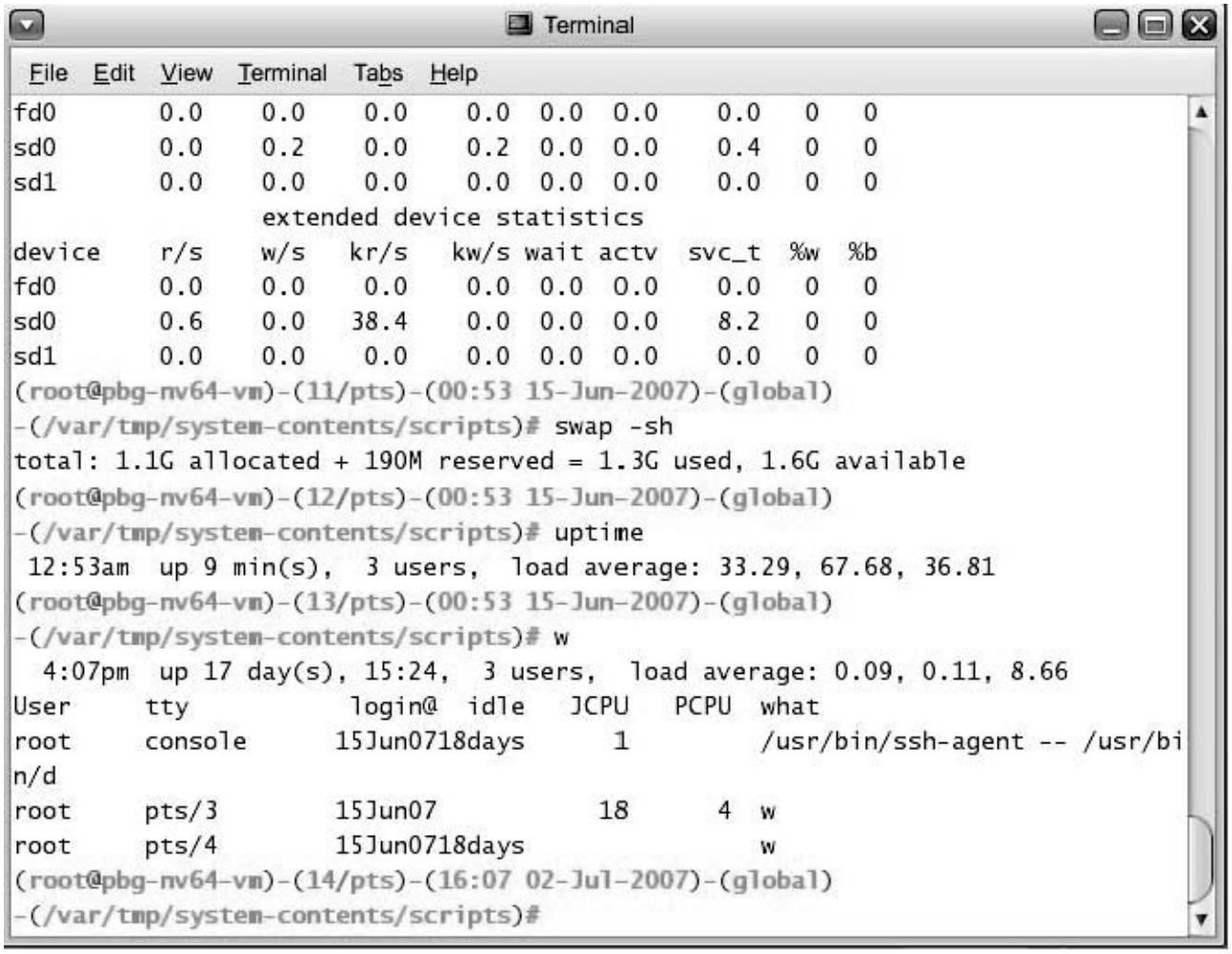
İşletim Sistemi İşlem Arayüzü- CLI

- 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, dizin 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 kullanabilir (Java Desktop, KDE)

Bourne Kabuğu – Komut Yorumlayıcısı



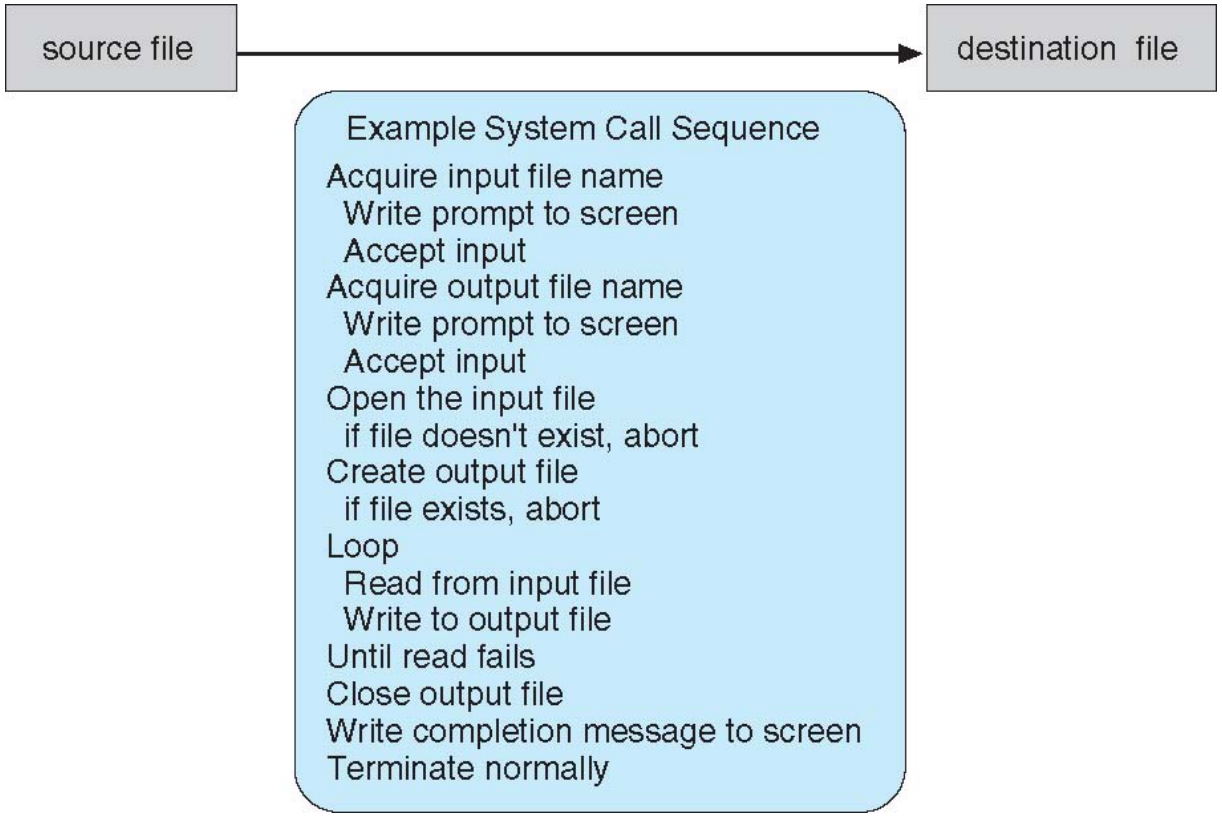
The Mac OS X GUI

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

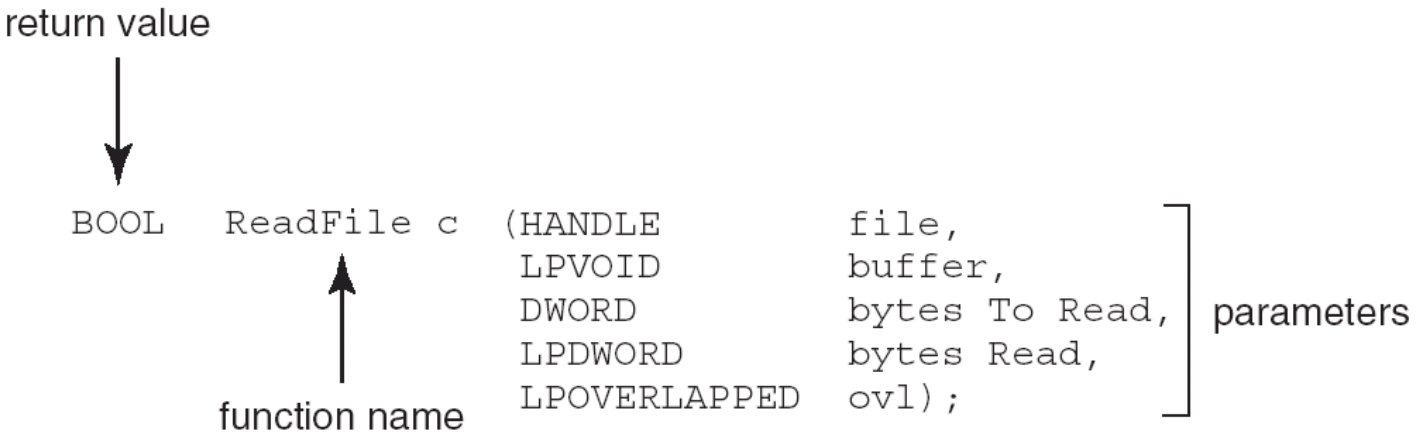
Sistem Çağrısı Örneği

- Bir dosyanın içeriğini başka bir dosyaya kopyalayan sistem çağrıları



Standart API Örneği

- Windows API'deki ReadFile() fonksiyonunu ele alalım
- Bir dosyadan okuma yapmak için kullanılan fonksiyondur

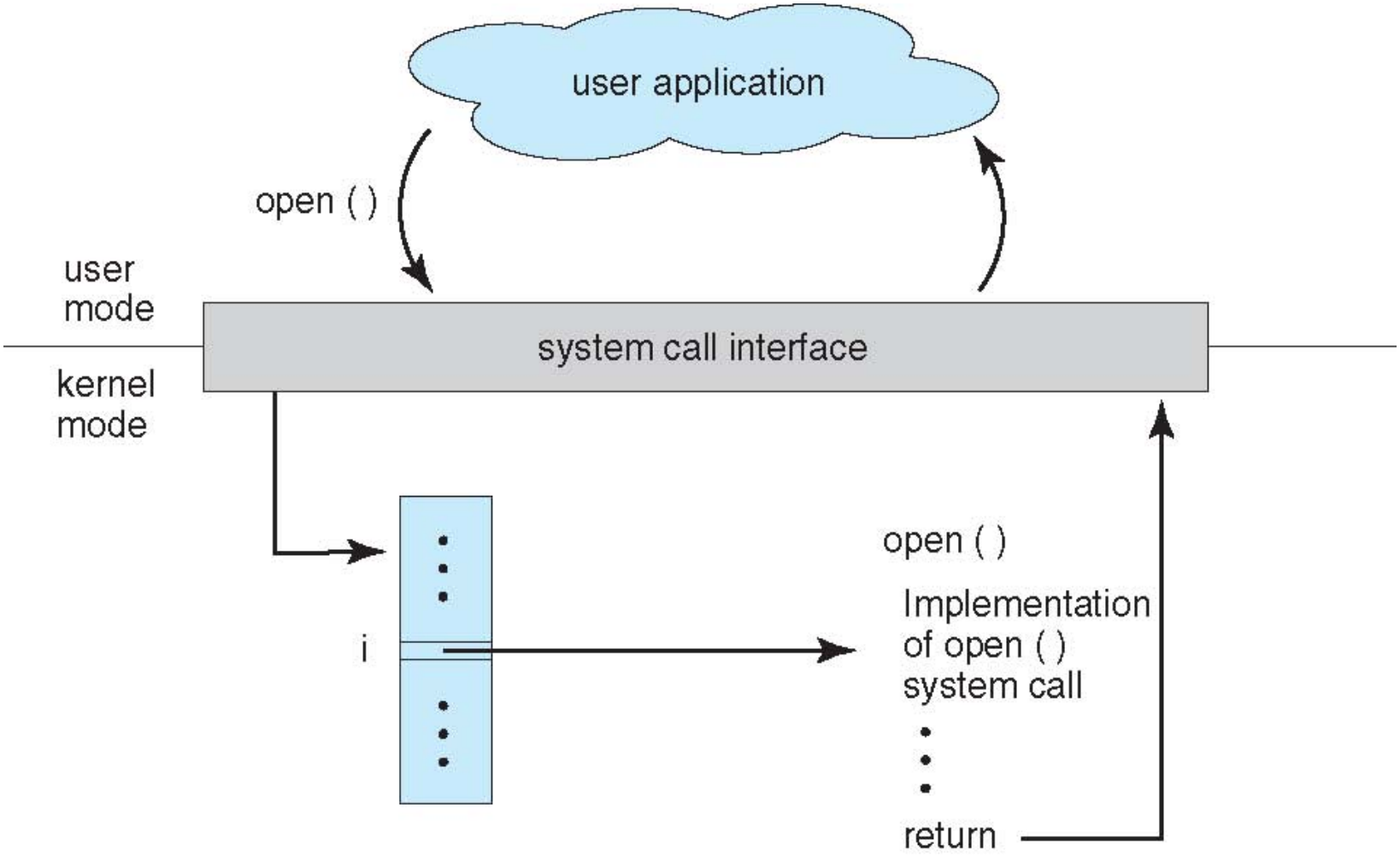


- ReadFile() fonksiyonuna gönderilen parametrelerin açıklaması
  - HANDLE file — okunacak dosya
  - LPVOID buffer — okunan ve yazılacak bilgilerin tutulduğu tampon bellek
  - DWORD bytesToRead — tampon bellekten okunacak bilginin kaç byte olduğu
  - bytesRead—en son okumada kaç byte'lık bilgi okunduğu
  - LPOVELPDWORDRLAPPED ovl —üst üste bindirilmiş I/O'nun kullanılıp kullanılmayacağını belirtir

Sistem Çağrısı Gerçekleştirimi

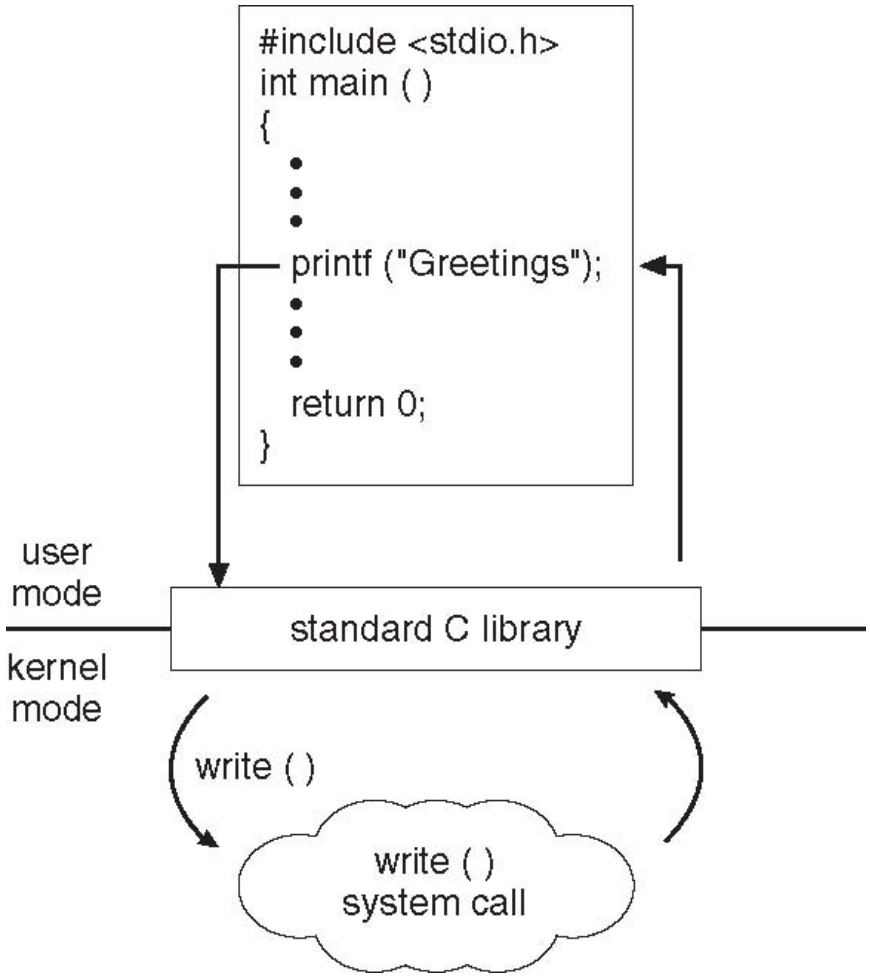
- Genellikle her bir sistem çağrısı ile bir sayı ilişkilendirilir
  - Sistem çağrısı arayüzü bu sayılarla indekslenmiş bir tablo tutar
- Sistem çağrısı arayüzü, işletim sistemi kabuğunda gerçekleştirilmiş sistem çağrısını çağırır ve eğer dönen bir bilgi varsa bu bilgi ile sistem çağrısı durumunu geri döndürür
- Sistem çağırısını çağıran uygulama sistem çağrısının nasıl gerçekleştirildiğini bilmelidir
  - API'ye uymalı ve işletim sisteminin çağrı ile ne yapacağını bilmelidir
  - İşletim sistemi arayüzünün pek çok detayı API ile programcıdan gizlenir
    - Çalışma zamanı destek kütüphanesi ile yönetilir – derleyici ile birlikte gelen kütüphaneler içine gömülmüş fonksiyonlardır

API – Sistem Çağrısı – OS İlişkisi



Standart C Kütüphanesi Örneği

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

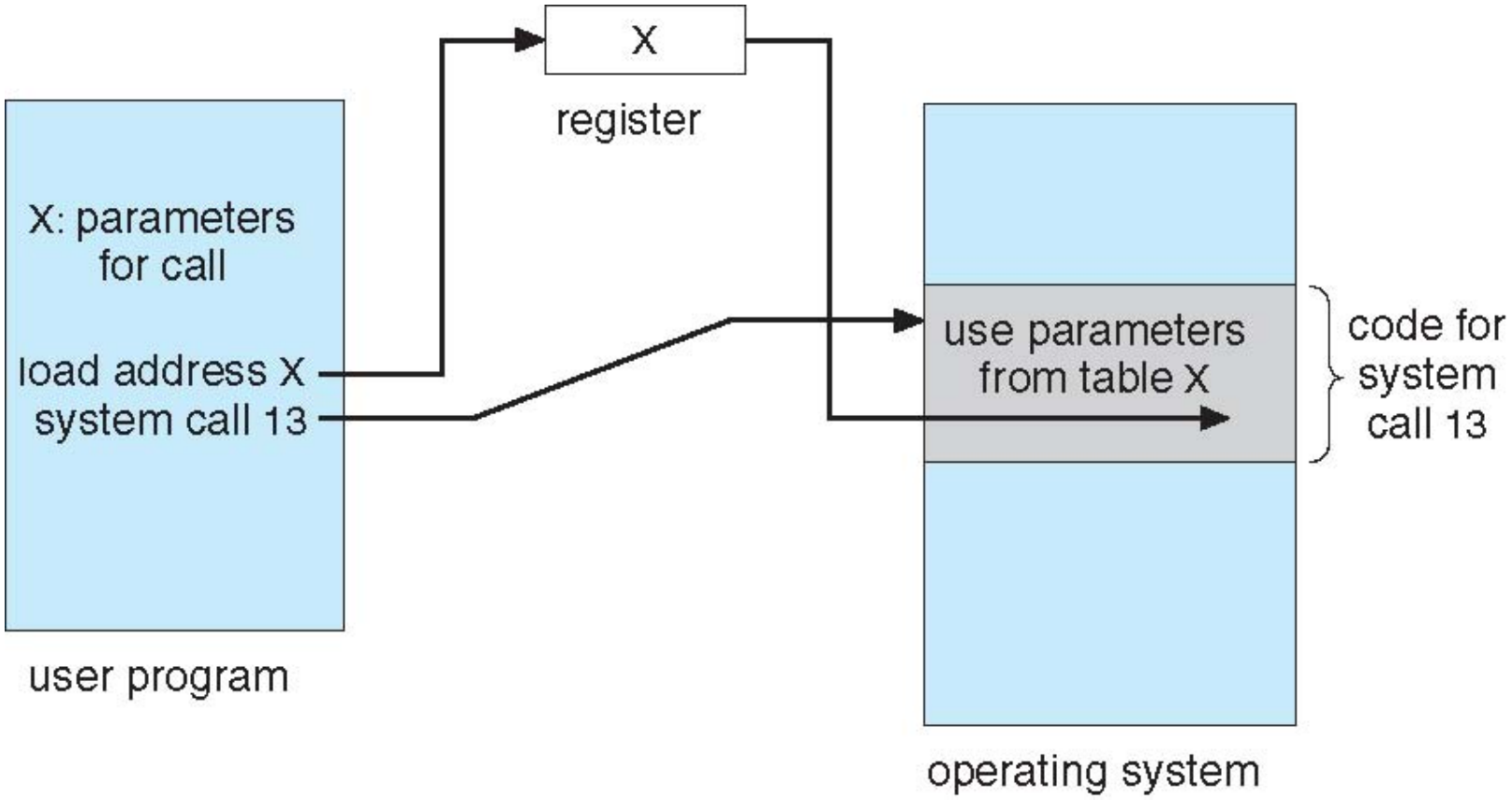


Sistem Çağrısına Parametre Gönderimi

- Genellikle, istenen sistem çağrısının numarasından fazlası gönderilir
  - Gönderilecek bilginin tipi ve miktarı işletim sistemine ve yapılacak çağrıya göre değişir

- İşletim sistemine parametre göndermeyi sağlayan üç genel yöntem
  - En basiti: parametreleri **yazmaçlar (registers)** içinde göndermek
    - Bazı durumlarda, yazmaç sayısından daha çok parametre göndermek gerekebilir
  - Parametreler **hafızada bir blokta veya tabloda** tutulur ve blokun adresi yazmaç ile gönderilir
    - Bu yaklaşım Linux ve Solaris tarafından kullanılmaktadır
  - Parameterler program tarafından **yığına (stack)** atılır ve işletim sistemi tarafından yığından çekilir
  - Blok ve yığın yöntemlerinde, gönderilen parametrelerin sayısı ve büyüklüğü konusunda herhangi bir sınır yoktur

Tablo ile Parametre Gönderimi



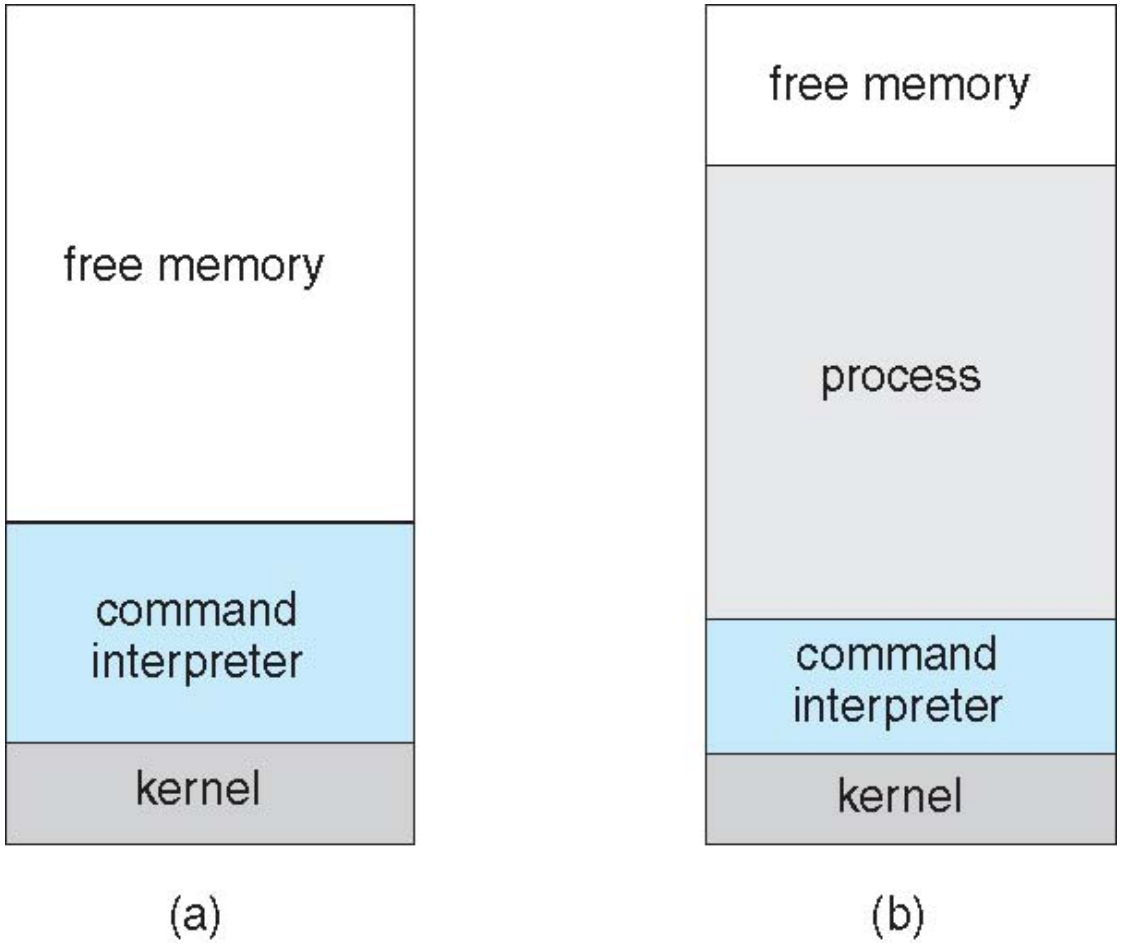
Sistem Çağrısı Çeşitleri

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

Windows ve Unix Sistem Çağrısı Örnekleri

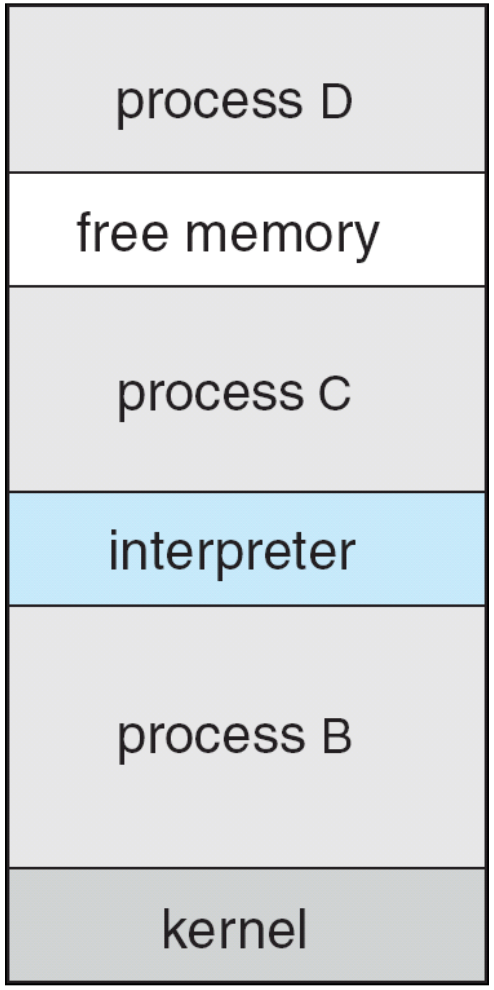
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information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

MS-DOS ile Program Çalıştırma



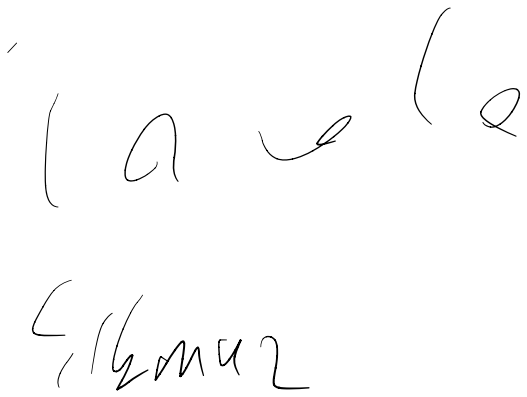
(a) Sistem başlangıcında (b) Bir program çalışırken

FreeBSD ile Çoklu Program Çalıştırma



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

- **Program geliştirmek ve çalıştırmak** için rahat bir ortam sağlamak
  - Bazıları sistem çağrıları yapan basit arayüzler
  - Bazıları ise çok daha karmaşık programlar
- **Dosya yönetimi**
  - Dosya oluşturma
  - Dosya silme
  - Dosya kopyalama
  - Dosya yeniden adlandırma
  - Dosya yazdırma
  - Dosyaları listeleme
  - Genel olarak dosyaları ve dizinleri değiştirme

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 kullanmakl için bir kayıt ortamı (registry) kullanır

Sistem Programları (devam)

- **Dosya değiştirme**
  - Dosyaları oluşturmak ve değiştirmek için metin düzenleyicileri (text editors)
  - Dosyaların içeriğinde arama yapmak ve dosya içeriklerini dönüştürmek için özel komutlar
- **Programlama dili desteği** – derleyiciler, assemblers, hata ayıklayıcılar (debuggers) and yorumlayıcılar (interpreters) bazen işletim sistemi ile birlikte sunulur
- **Program yükleme ve çalıştırma** – çeşitli yükleyiciler, üst seviye diller ve makina dili için hata ayıklama sistemleri
- **İletişim** – İşlemler, kullanıcılar ve bilgisayar sistemleri arasında sanal bağlantılar oluşturmak için mekanizma sunar
  - Kullanıcıların birbirlerinin ekranına mesaj yollamasını, web sayfalarında gezinmesini, elektronik posta göndermesini, uzak bilgisayarlara bağlanmasını ve makinalar arasında dosya göndermesini sağlar

İşletim Sistemi Tasarımı ve Gerçekleştirimi

- İdeal işletim sistemi tasarımı ve gerçekleştirimine dair ideal bir çözüm yok. Ancak başarılı olduğu gözlemlenen yaklaşımlar var
- İşletim sistemlerinin içsel yapısı çok farklılıklar gösteriyor
- Hedefleri ve özellikleri tanımlayarak başla
- Donanım seçimi ve sistem tipi önemli
- *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

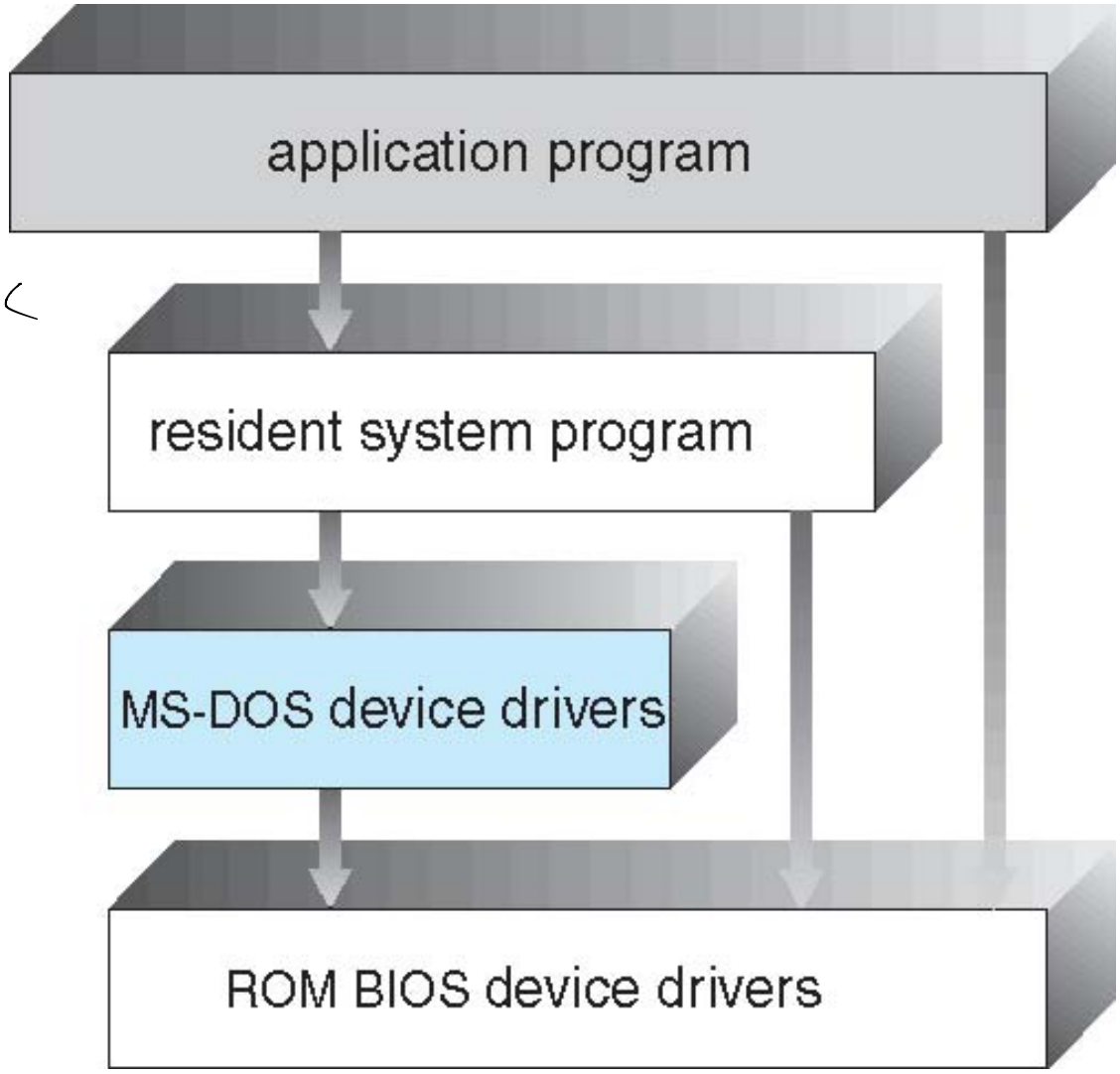
  İlke (policy): Ne yapılmalı?
  Mekanizma (Mechanism): Nasıl yapılmalı?

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

MS-DOS Katman Yapısı



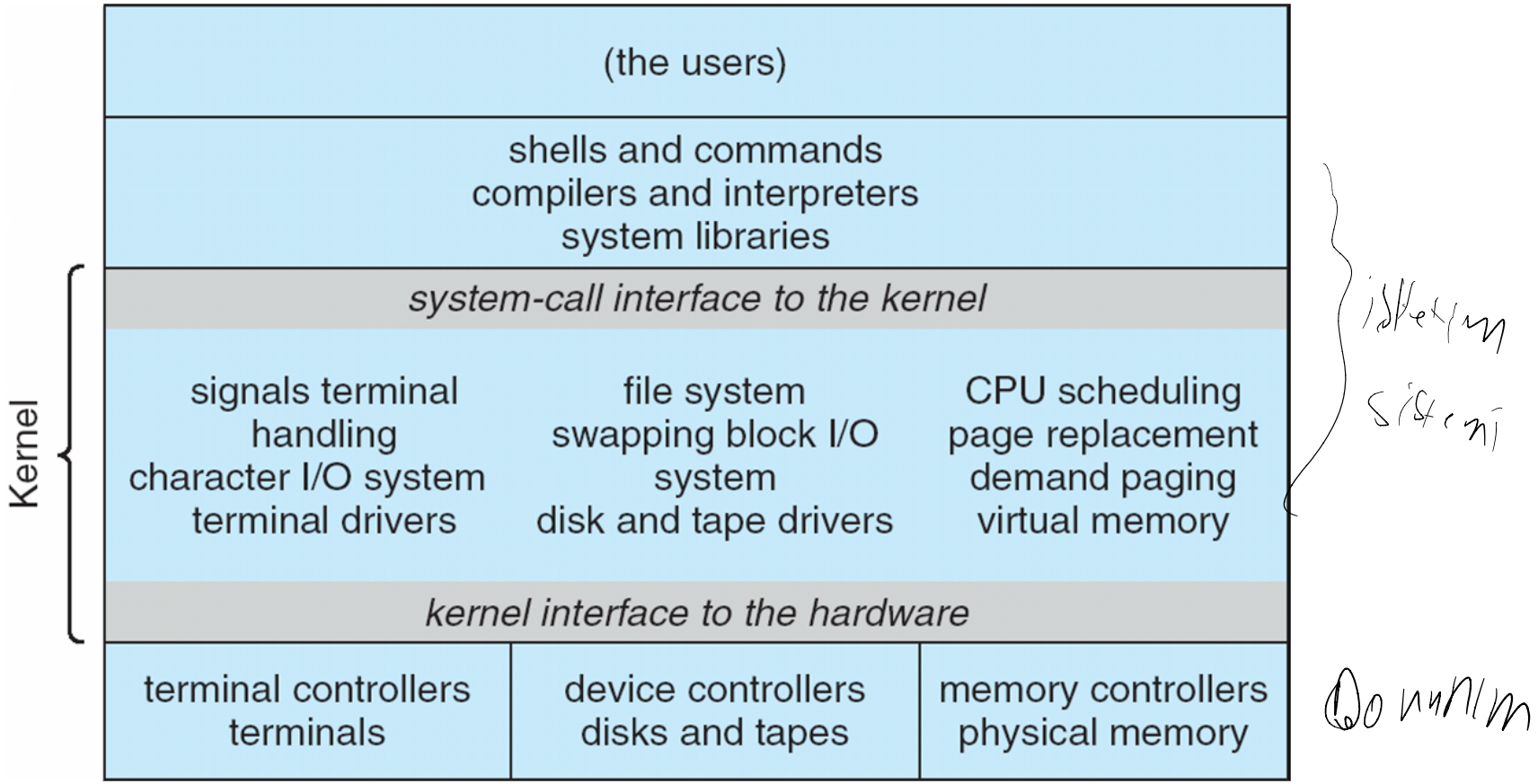
Katmanlı Yaklaşım

- 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

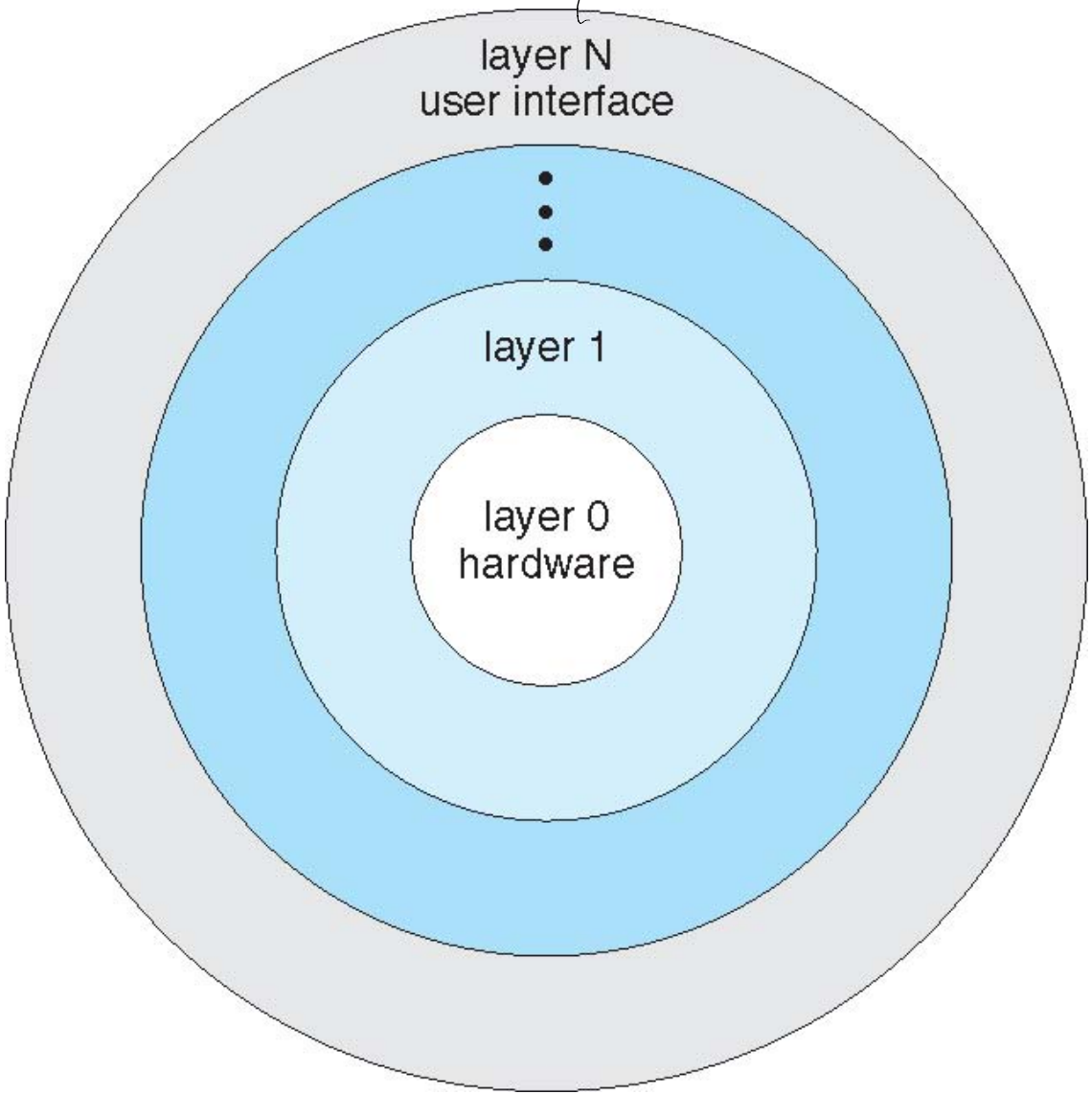
UNIX

- UNIX sınırlı bir yapılandırmaya sahip: İki adet 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



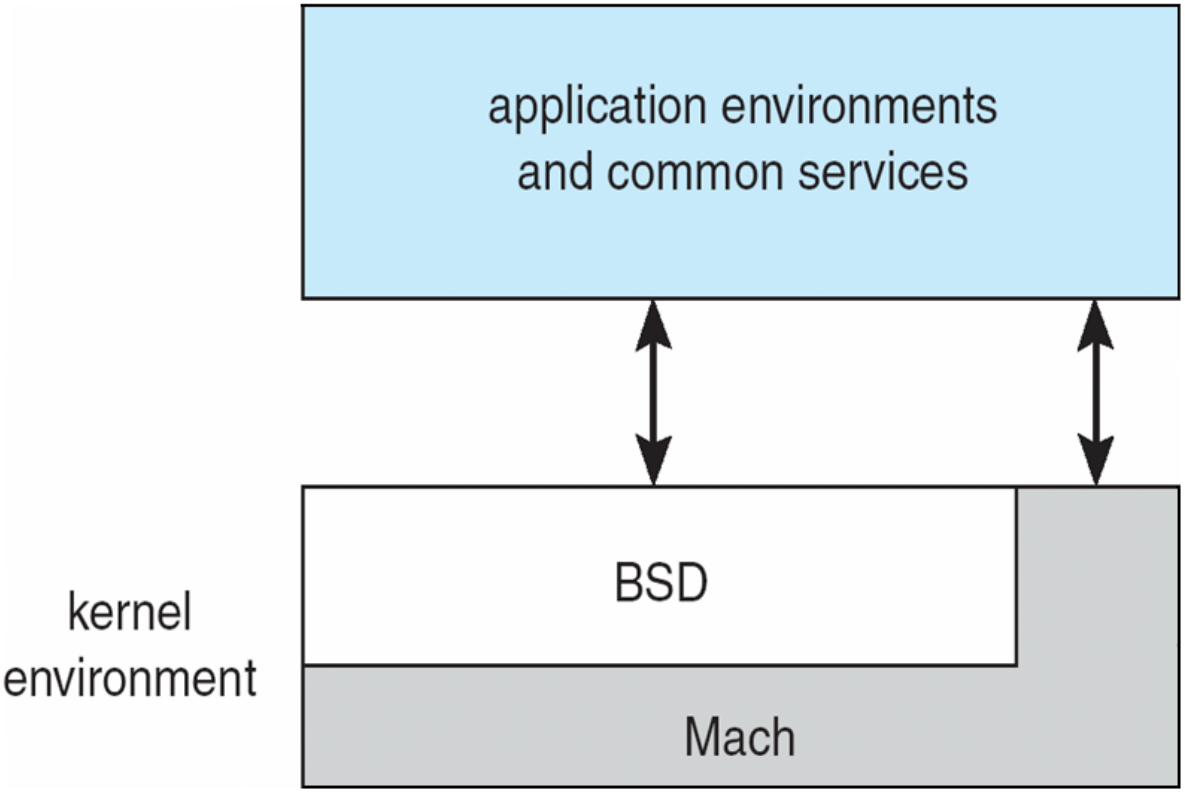
Katmanlı İşletim Sistemi



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

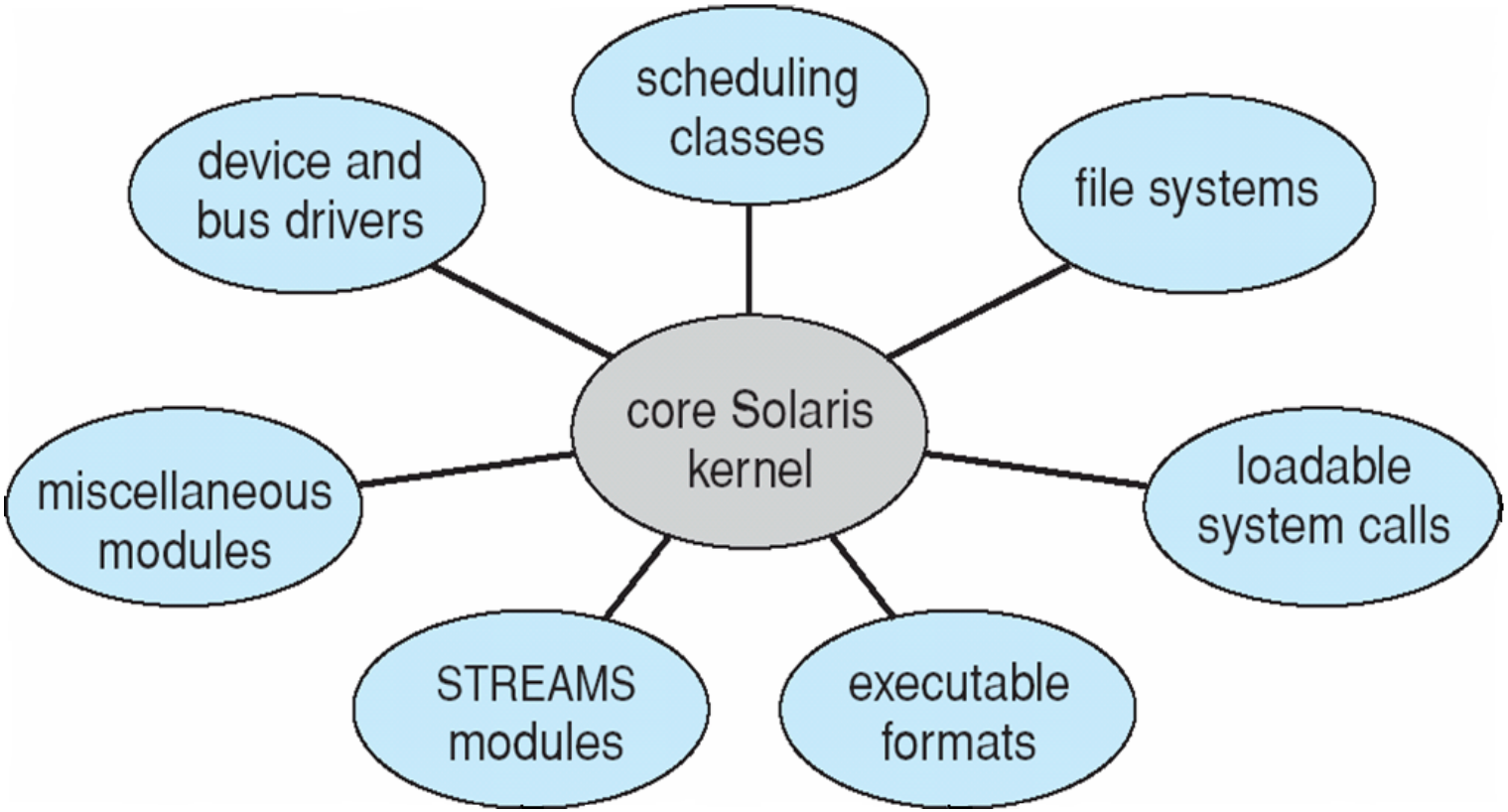
Mac OS X Yapısı – Hibrit



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



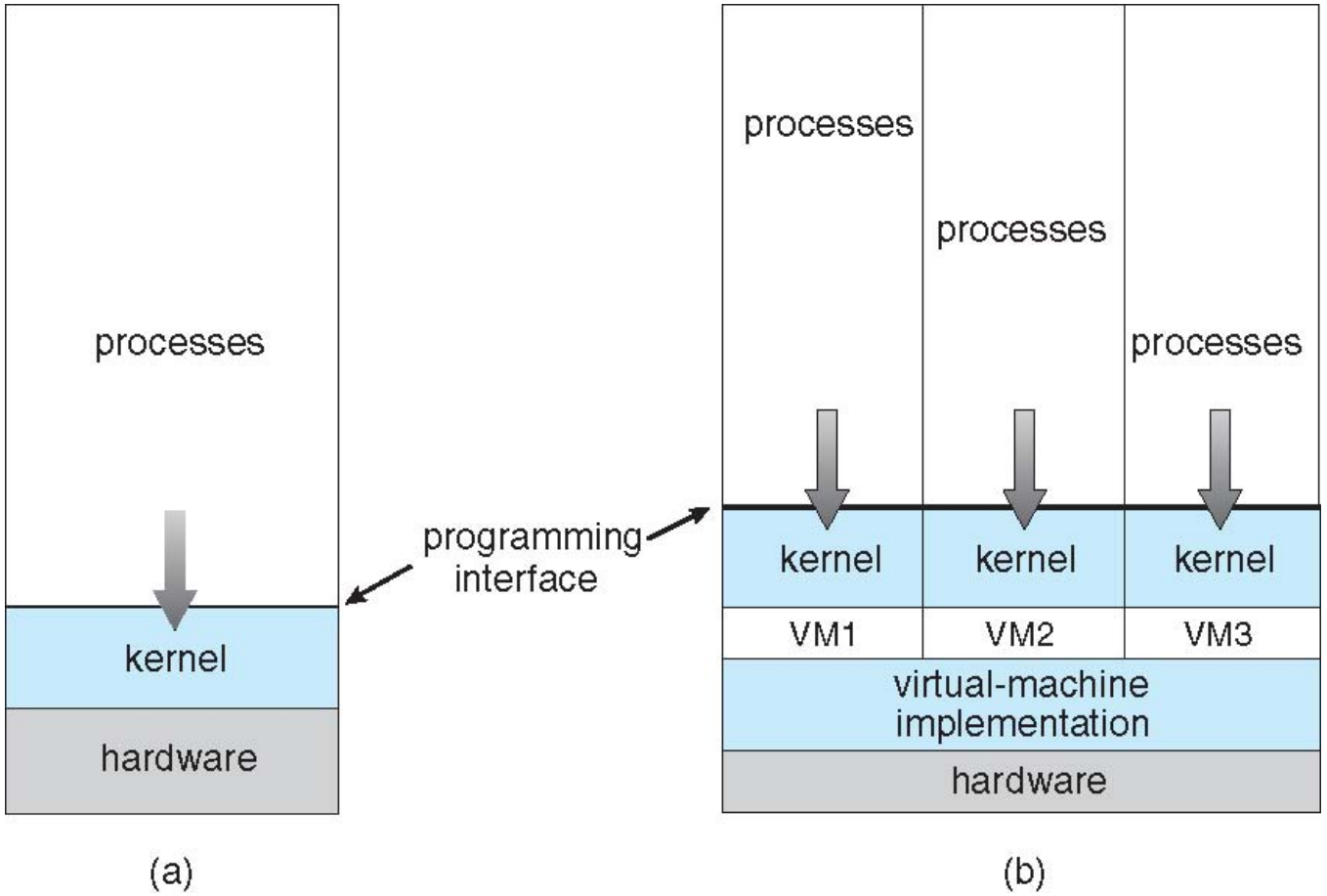
Sanal Makinalar

- **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)

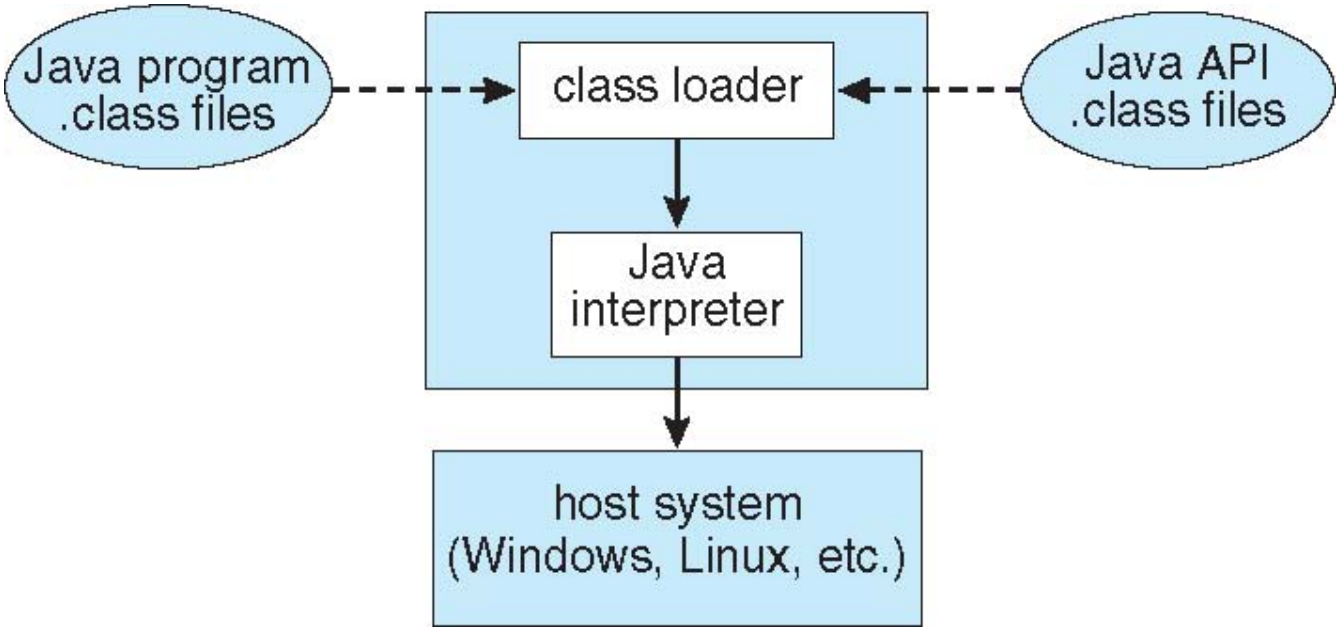


(a) Sanal olmayan makine (b) sanal makina

Java

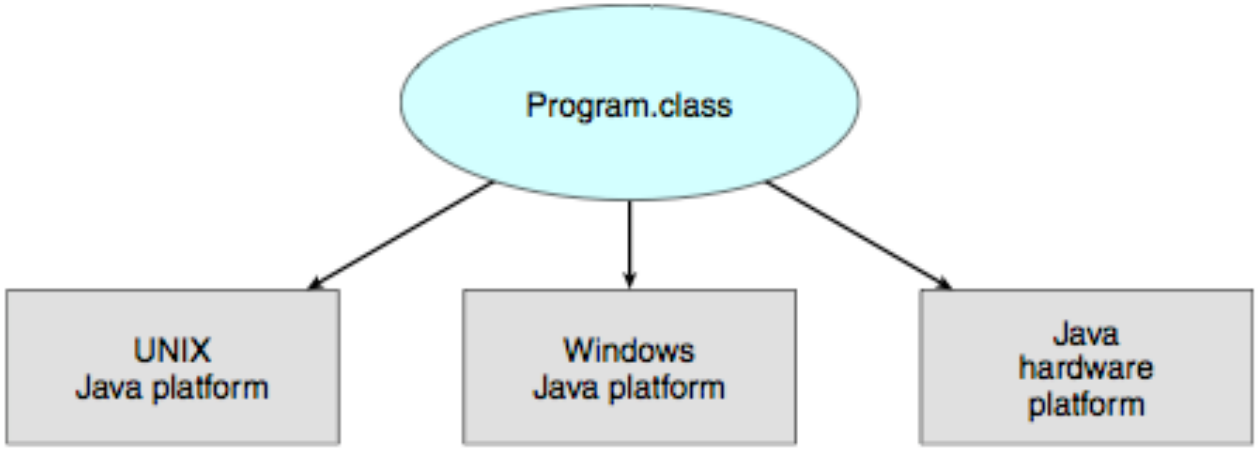
- Java:
  1. Programlama dili spesifikasyonu
  2. Uygulama programlama arayüzü (API)
  3. Sanal makine spesifikasyonu

Java Sanal Makinası

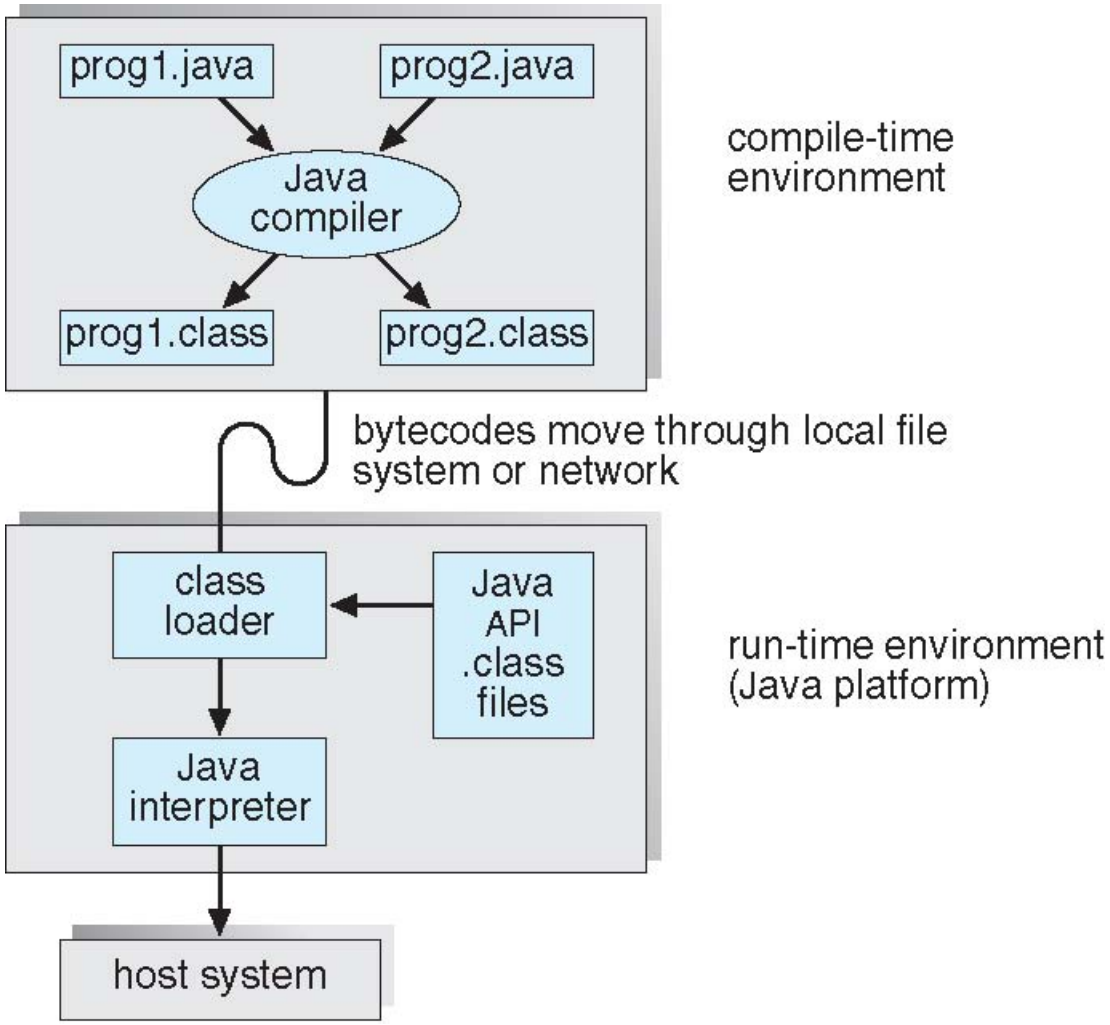


JVM - Taşınabilirlik

Java'nın farklı platformlarda taşınabilirliği



Java Geliştirme Aracı



Sistemin Yüklenmesi

- Donanımın işletim sistemini başlatabilmesi için işletim sisteminin donanım tarafından erişilebilir olması gerekir
  - ROM'da tutulan **BIOS** bilgisayar ile ilgili ön kontrolleri yapar ve işletim sistemini yükleyecek olan **önyükleyici programı (bootstrap program, bootstrap loader)** çalıştırır
  - Neden ROM?
    - RAM dahi sistem başlangıcında bilinmeyen bir durumda
    - ROM'a bilgisayar virüsleri kolay bulaşamaz
  - **Önyükleyici program** – çekirdeği bulup hafızaya yükleyen ve çekirdeğin çalışmasını başlatan program
  - Büyük işletim sistemlerinde yükleme iki adımda gerçekleşir:
    - Diskte yeri belirli olan **yükleme bloğundan (boot block)** önyükleyici program yüklenir (Örnek: Linux'de GRUB)
    - Ön yükleyici program çekirdeği hafızaya yükler ve çalışmasını başlatır