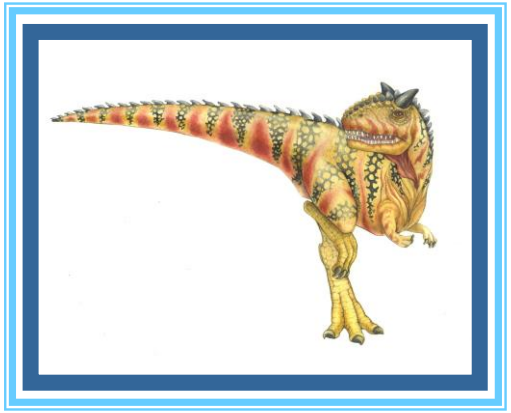
Bölüm 2: İşletim Sistemi Yapıları



Doç.Dr. Ahmet ZENGİN

Bölüm 2: İşletim Sistemlerinin Yapısı

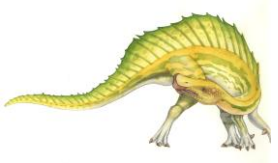
- İşletim Sistemi Servisleri
- Kullanıcı Arayüzü
- Sistem Çağrıları
- Sistem Çağrısı Tipleri
- Sistem Programları
- İşletim Sistemi Tasarımı ve Uygulaması
- İşletim Sistemi Yapısı
- Sanal Makineler
- İşletim Sistemlerinde Hata Ayıklama
- İşletim Sistemi Üretimi
- Sistem Önyükleme

Hedefler

- Bir işletim sisteminin, kullanıcılara, processlere, ve diğer sistemlere sağladığı hizmetleri / servisleri tanıtmak
- İşletim sistemlerinin farklı yöntemlerle yapılandırılmasının tartışılması
- İşletim sistemlerinin kurulum, özelleştirilme ve önyükleme işlemlerinin açıklanması

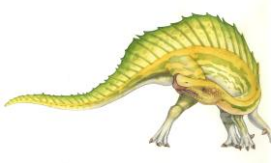
İşletim Sistemi Servisleri

- İşletim sistemleri, programların çalışması için kullanıcılara ve programlara uygun bir çevre sağlar.
- İşletim sistemleri kullanıcılara yardımcı olacak servisler sunar. Bunlardan bazıları:
  - **User interface (Kullanıcı Arayüzü)**- Neredeyse tüm işletim sistemleri kullanıcı arayüzüne sahiptir(UI). Kullanıcının işletim sistemine komut vermesini sağlar.
    - **Komut Satırı (CLI)**, **Grafiksel Kullanıcı Arayüzü(GUI)**, **Çalıştırılabilir Dosyalar(Batch) vb.**
  - **Program execution (Uygulama Çalıştırma)**- İşletim sistemi, programları belleğe yükleyerek çalıştırabilmeli ve çalışmakta olan programları normal ya da anormal durumlarda (hata belirterek) sonlandırabilmelidir.
  - **I/O operations (Giriş/Çıkış İşlemleri)**- Çalışan bir program girdi/çıktı işlemleri yapmak isteyebilir (dosyadan ya da bir giriş/çıkış aygıtından).
  - **File-system manipulation (Dosya-Sistem Değişiklikleri)**-. Programlar, dosya ve dizinlerde okuma, yazma, silme, oluşturma, arama yapma, bilgi listeleme ve erişim izinlerini yönetme gibi işlemler gerçekleştirmek isteyebilir. Dosya sistemleri özellikle bu konuyla ilgilenir.



İşletim Sistemi Servisleri(Devam)

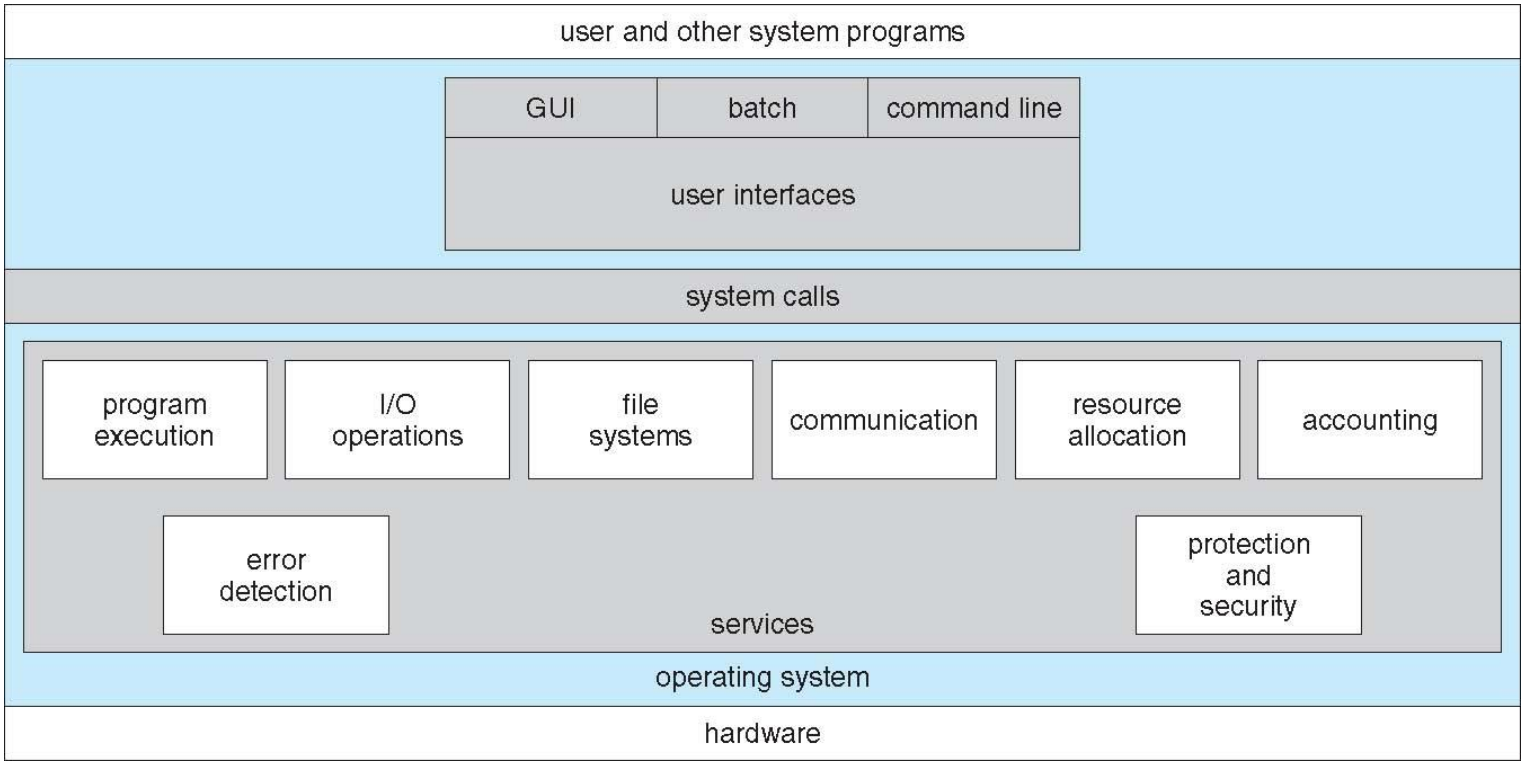
- **Communications (İletişim)** – Sistemdeki bir *process*(işlem) network üzerinden başka bir işlem ya da bilgisayarla iletişime geçmek isteyebilir.
  - Bu iletişim, *shared memory* veya *message passing* yöntemlerinden biri ile gerçekleştirilir.
- **Error detection (Hata Tespiti)**– İşletim sistemi, oluşabilecek hatalara karşı sürekli tetikte olmalıdır.
  - İşlemcide, fiziksel bellekte, bağlı durumdaki bir giriş/çıkış aygıtında ya da kullanıcı programında hata ile karşılaşılabilir.
  - İşletim sistemi her türlü hataya karşı önlem almalıdır.
  - Hata ayıklama araçları, kullanıcı ve programcının sistemi etkin olarak kullanabilmesine çok büyük katkı sağlar.



İşletim Sistemi Servisleri(Devam)

- Bazı işletim sistemi servisleri de sistemin kendi kendine etkin bir biçimde kaynak paylaşımı yapabilmesini sağlar:
  - **Resource allocation (Kaynak Tahsisi) -** Birçok kullanıcı veya görevin (*task*) aynı anda çalıştığı sistemlerde, her biri için kaynak ayrılmasını sağlar.
  - (İşlemci, bellek ve sabit disk gibi) Pek çok aygıt için özel bir kaynak ayırma işlemi uygulanırken, (giriş/çıkış aygıtları gibi) bazı aygıtlarda ise daha genel bir kaynak ayırma işlemi uygulanır.
  - **Accounting –** Kullanıcıların ne kadar kaynak kullandığını takip edilmelidir.
  - **Protection and security (Koruma ve Güvenlik) -** Çok sayıda işlem eş zamanlı işletilirken, bir işlemin diğer işlemin işleyişine veya işletim sisteminin işleyişine karışamıyor olması gerekir. Bu yüzden bir işlem, başka bir işlemin bellek bölgesine erişemez.
    - **Koruma**, sistem kaynaklarına erişim kontrollü biçimde gerçekleşmesidir.
    - **Güvenlik,** dışarıdan gelen izinsiz talepleri engelleme gibi konuları içerir.
    - Bir sistemin güvenlik ve koruma durumu sürekli olmalıdır. Zincir en zayıf halkası kadar güçlüdür.

İşletim Sistemi Servislerine Bakış

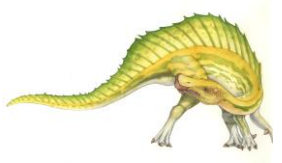


Kullanıcı Arayüzü - CLI

- Komut satırı arayüzü ya da **komut yorumlayıcısı** doğrudan sisteme komut verme imkanı sağlar.
  - Komut satırından doğrudan ya da sistem servisleri vasıtasıyla işletim sistemi çekirdeğine (kernele) ulaşılır.
  - Çoğu kez çoklu yapılar kullanılır – kabuk(**shells)**
  - Temel olarak, kullanıcıdan komut alınır ve çalıştırılır.
    - Bazen bu girdi bir program adı bazen de yerleşik bir komut olabilir.
      - » Komutlar parametreler ile değişik davranış gösterir.

Kullanıcı Arayüzü - GUI

- Kullanıcı dostu, bilgi ve görsellerle donatılmış arayüz.
  - Fare, klavye ve monitör.
  - Programları, dosyaları vb. temsil eden simgeler mevcuttur.
  - Fare yardımı ile çeşitli eylemler gerçekleştirilir(bilgi getirme, seçenekler, fonksiyon çalıştırma, dizin açma)
  - İlk kez Xerox PARC tarafından kullanılmıştır.
- Günümüzdeki çoğu sistem hem CLI (komut arayüzü), hem GUI(grafik arayüz) barındırır.
  - Microsoft Windows, grafik arayüzlüdür ve CLI shell barındırır.
  - Apple Mac OS X'de "Aqua" GUI arayüzü ile UNIX kerneli altında shell barındırır.
  - Solaris CLI ve opsiyonel olarak GUI barındırır(Java Desktop, KDE).



Shell (Kabuk) Komutu Yorumlayıcı

Terminal

File Edit View Terminal Tabs Help

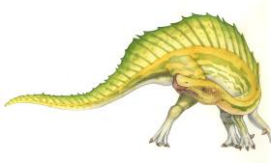
```
fd0       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
sd0       0.0    0.2    0.0    0.2  0.0  0.0    0.4   0   0
sd1       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
                 extended device statistics
device    r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
fd0       0.0    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-vm)-(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-vm)-(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-vm)-(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      15Jun0718days      1         /usr/bin/ssh-agent -- /usr/bi
n/d
root     pts/3        15Jun07           18      4  w
root     pts/4        15Jun0718days                w
(root@pbg-nv64-vm)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

The Mac OS X Arayüzü

Sistem Çağrıları

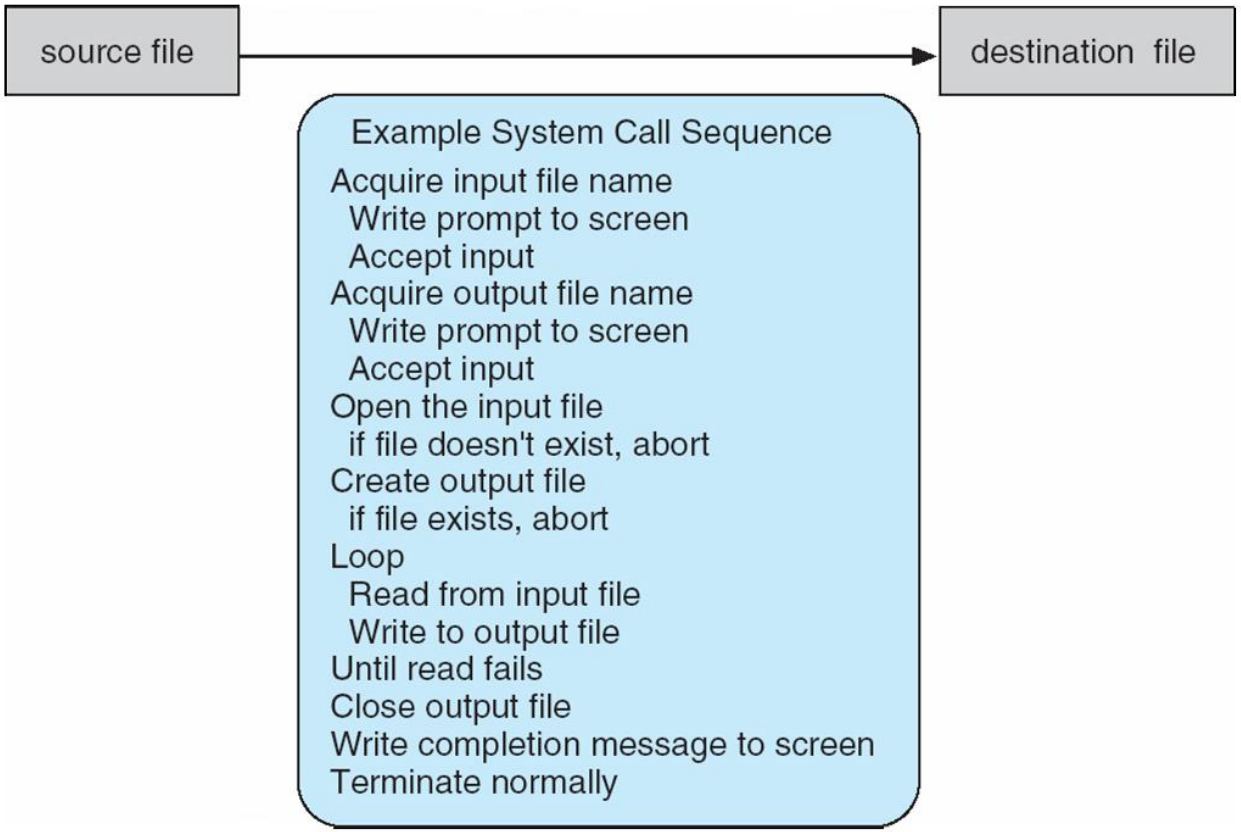
- İşletim sistemi, sistem çağrı servislerini programlama imkanı sunar.
- Bu servisleri programlamak için çoğunlukla yüksek seviyeli diller kullanılır(C ya da C++).
- Programlar, bu çağrı servislerine direk ulaşmak yerine çoğunlukla API (*Application Program Interface*) üzerinden iletişim kurar.
- En sık kullanılan üç API, Win32 API (Windows için),POSIX tabanlı sistemler için POSIX API (UNIX, Linux ve MacOS X 'in tüm versiyonlarında mevcuttur) ve Java API (Java Sanal Makinesi için)'dir.
- Neden doğrudan sistem çağrısı yapmak yerine API kullanılır?

  (Not: Yazı boyunca sistem çağrısı adı kullanılacaktır.)



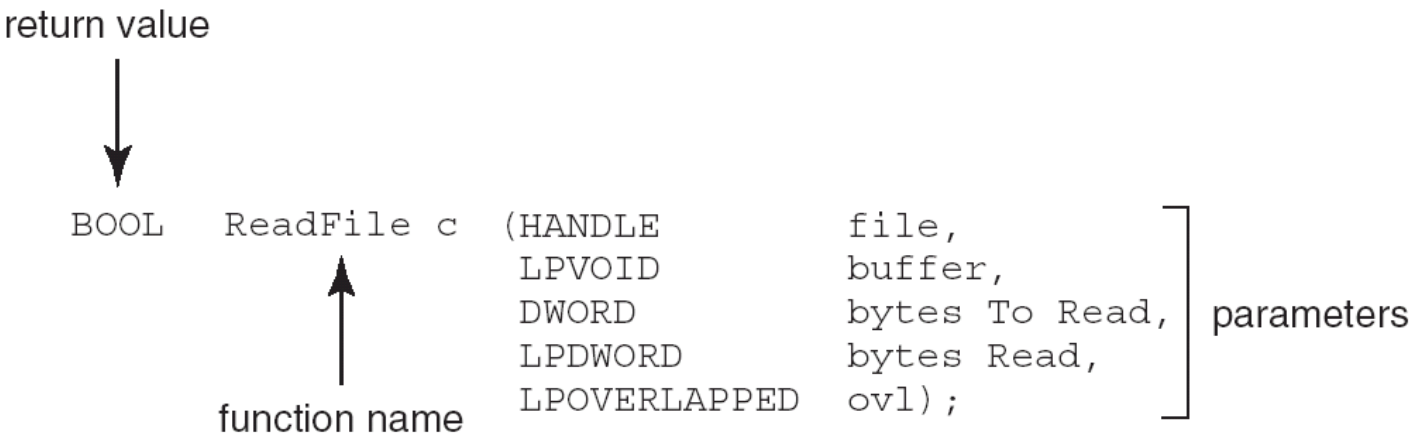
Sistem Çağrısı Örneği

- Bir dosyadan diğerine içerik kopyalamak için yapılan sistem çağrısı.



Standard API Örneği

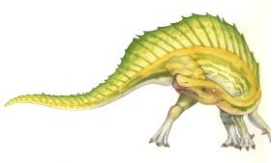
- ReadFile() fonksiyonunu göz önüne alalım
- Win32 API— dosyadan okuma yapan bir fonksiyon



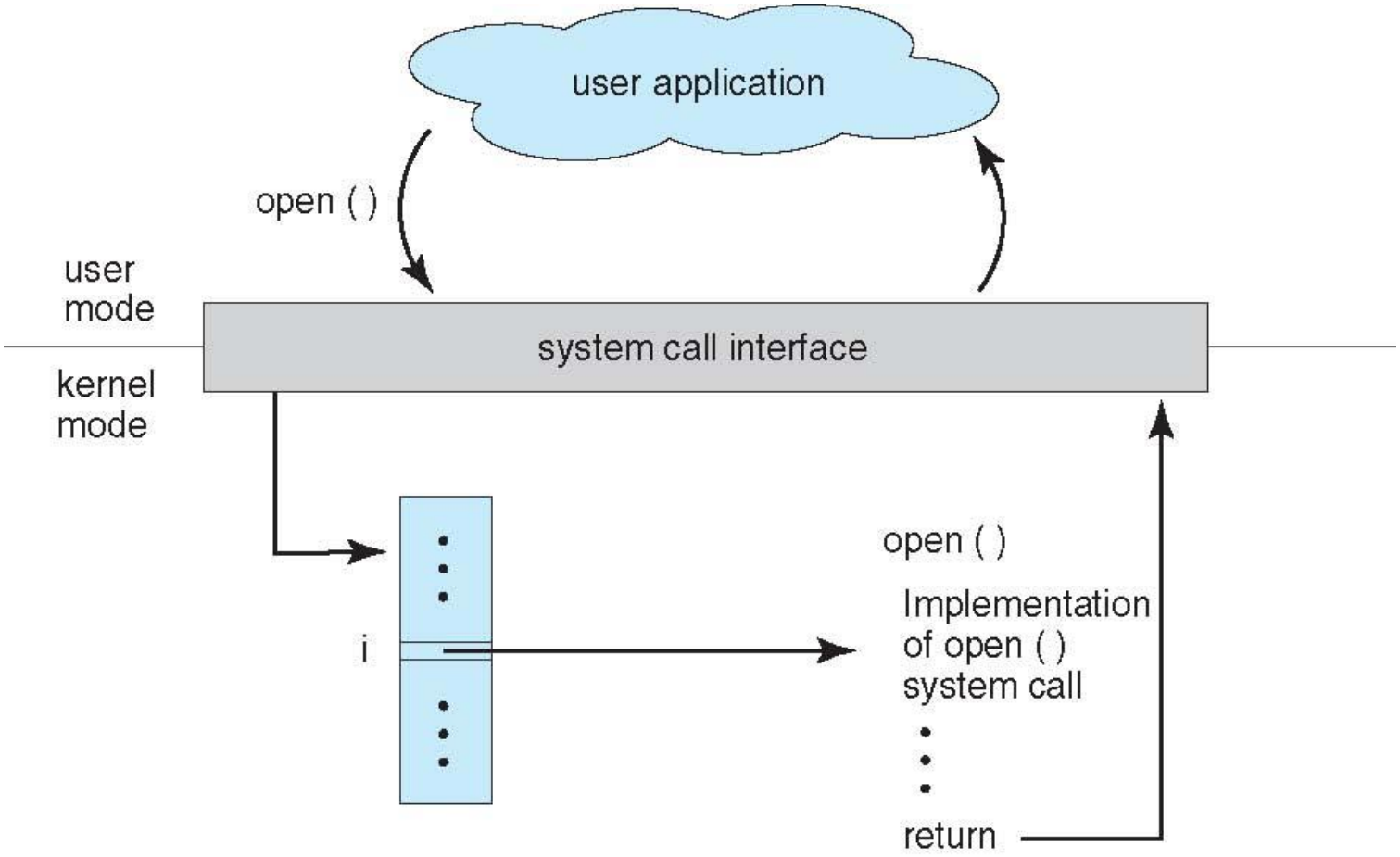
- ReadFile()'a geçen parametrelerin bir tanımı
  - HANDLE file—okuma yapılacak dosya
  - LPVOID - okunan verinin yazılacağı bellek alanı
  - DWORD bytesToRead- okunacak veri boyutu
  - LPDWORD bytesRead— en son okunan bayt miktarı
  - LPOVERLAPPED ovl—çakışmalı(overlapped) I/O kullanılıp kullanılmadığını gösterir

Sistem Çağrısı Yapmak

- Her sistem çağrısının kendine ait bir numarası vardır.
  - Sistem çağrısı arayüzü, bu numaraları indexlenmiş bir tablo halinde tutar.
- Sistem çağrı arayüzü, işletim sistemi kernelindeki ilgili sistem çağrısını çağrırır ve sistem çağrısının durumunu ve dönüş değerini döndürür.
- Sistem çağrısında bulunan program ya da istemci, sistem çağrısının nasıl gerçekleştiğiyle ilgilenmez.
- Yalnızca API ile iletişim kurar ve işletim sisteminin döndüreceği sonucu bekler.
  - Bu sürecin bir çok detayı API sayesinde programcıdan gizlenmiştir.
    - İşlemler, çalışma zamanı destek kütüphanesi tarafından yönetilir.(Gerekli derleyici ve fonksiyonlar bu kütüphanelerde yerleşik olarak bulunur.)

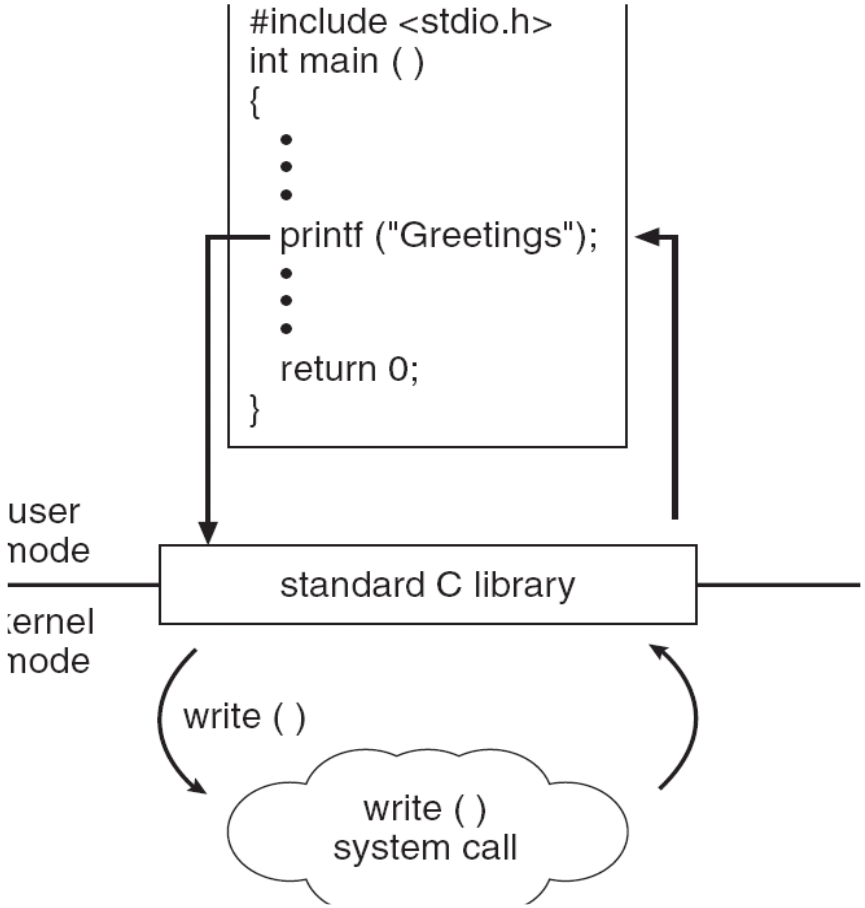


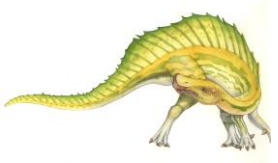
API – Sistem Çağırısı– OS İlişkisi



Standard C Kütüphanesi Örneği

- C programı, printf() fonksiyonu ile C kütüphanesine "write() sistem çağrısı" yapması için çağrıda bulunuyor.

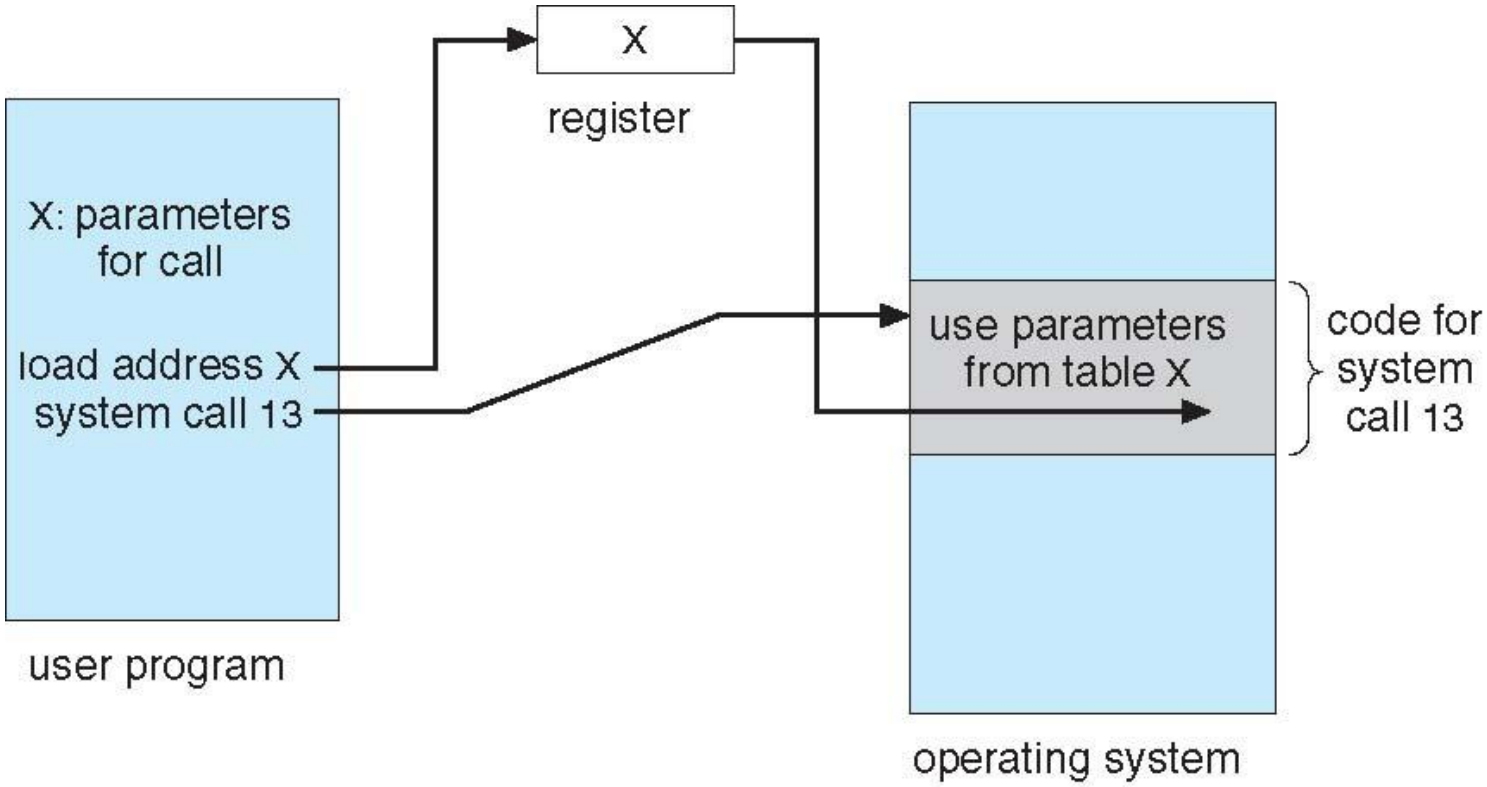




Sistem Çağrılarına Parametre Geçişi

- Genellikle, istenilen sistem çağırısına ulaşmak için basit bir kimlik numarasından daha fazlası gerekir.
  - Gereken bilgi türü ve miktarı işletim sistemine ve çağrıya göre değişir.
- İşletim sistemine parametre göndermek için 3 temel metot kullanılır:
  - Basit: parametreleri register a yazma.
    - Bazı durumlarda, parametre miktarı register dan fazla olabilir.
  - Parametreler, bellek içerisinde blok ya da tabloda tutulur , ve blok adresleri register a yazılır.
    - Bu yaklaşım Linux ve Solaris'ten alınmıştır.
  - Parametreler, program tarafından, yığın (*stack)* üzerine eklenir (*push) ve* işletim sistemi yığın üzerinden alınır (*pop*).
  - Blok ve yığın metotları, sayılara ya da geçilen parametre uzunluklarına bir limit koymaz.

Tablo ile Parametre Geçişi

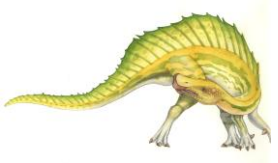


Sistem Çağrı Tipleri

- Process Kontrolü
  - Bitirme, iptal etmek
  - Yükleme, çalıştırma
  - create process(işlem oluşturma), terminate process(işlem sonlandırma)
  - get process attributes (işleme değerlerini getirme), set process attributes (işleme değer atama)
  - Belirli bir süre bekleme
  - Bekleme event'ı (olayı), sinyal event'ı
  - Bellek ayırma, belleği serbest bırakma
- File Management (Dosya yönetimi)
  - Dosya oluşturmak, silmek
  - Dosya açmak, kapatmak
  - Dosyadan okuma, dosyaya yazma, konum değiştirme
  - Dosya özelliklerini (attribute) değiştirmek

Sistem Çağrı Tipleri (Devam)

- Device Management (Aygıt yönetimi)
  - Aygıt talep etme, aygıtı serbest bırakma
  - Okuma, yazma, konum değiştirme
  - Aygıt değerlerini getirme, aygıt değerlerini değiştirme
  - Aygıt ekleme, kaldırma
- Information maintenance (Bilgilendirme Hizmeti)
  - Zamanı ya da tarihi getirme, değiştirme
  - Sistem verisini getirme, değiştirme
  - İşlem, dosya ya da aygıt değerlerini getirme, değer atama
- Communications (İletişim)
  - Bağlantı oluşturmak, silmek
  - Mesaj alma, gönderme
  - Durum bilgisi transferi
  - Uzak aygıta bağlanmak, bağlantıyı sonlandırmak



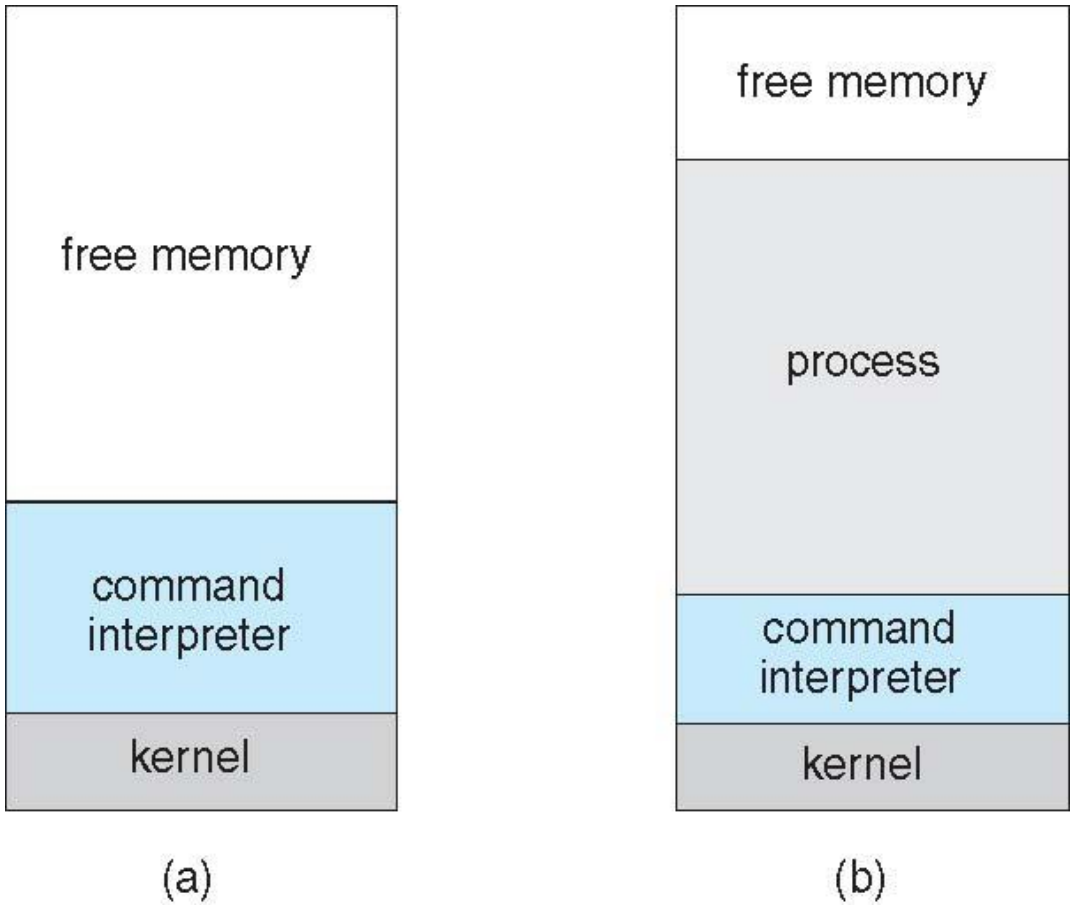
Windows ve Unix Sistem Çağrı Örnekleri

	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()

Örnek: MS-DOS

- Aynı anda tek bir işlem yapabilir (Single-tasking)
- Sistem başlatılırken shell çağırılır.
- Programı başlatmak için basit bir yöntem kullanılır.
  - İşlem oluşturulmaz.
- Bellek tek bölümden oluşur.
- Loads program into memory, overwriting all but the kernel
- Programı kapat -> shell reloaded

MS-DOS Yürütme



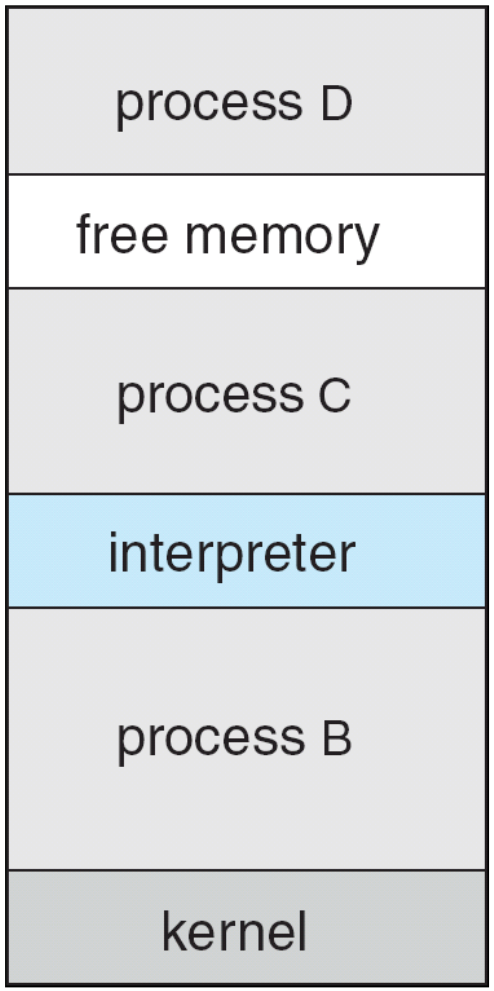
(a) Sistem ilk başlatıldığında

(b) (b) Çalışmakta olan program

Örnek: FreeBSD

- Unix türevi
- Aynı anda birden fazla işlem yapabilme (Multitasking)
- Kullanıcı girişi -> kabuk
- Kabuk (Shell) prosesi oluştururken fork() sistem çağrısını çalıştırır
  - Prosese programı yüklerken exec() sistem çağrısını çalıştırır.
  - Kabuk prosesin kullanıcı komutlarıyla sonlanmasını yada devam etmesinin bekler
- Prosesler 0 koduyla(hatasız) veya > 0 koduyla (hata kodu) sonlanır

FreeBSD Üzerinde Birden Fazla İşlem Çalışırken

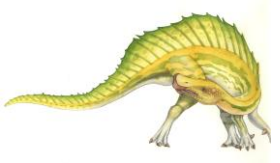


Sistem Programları

- Sistem programları program geliştirme ve çalıştırma için ortam sağlar. Sistem programlarını şu başlıklara ayırabiliriz:
  - Dosya işleme
  - Durum bilgisi
  - Dosya değişiklikleri
  - Programlama dili desteği
  - Program yükleme ve çalıştırma
  - İletişim
  - Uygulama programları
- Kullanıcıların çoğu gerçek sistem çağrıları yerine sistem programlarını kullanılır.

Sistem Programları

- Sistem programları program geliştirme ve çalıştırma için ortam sağlar.
  - Bu programların bir kısmı basit bir arayüz ile sistem çağrısında bulunurken diğer kısmı ise çok daha karmaşık işlemler gerçekleştirir.
- **Dosya Yönetimi** – Oluşturma, silme, kopyalama, yeniden adlandırma, yazdırma, listeleme gibi dosya ve dizinler üzerinde gerçekleştirilen işlemlerdir.
- **Durum Bilgisi**
  - Gün, saat, kullanılabilir disk alanı, disk kapasitesi vb. bilgiler.
  - Performans, sistem günlüğü, hata ayıklama bilgileri
  - Genel olarak, bu programlar, çıktıyı biçimler ve terminalde yada diğer çıkış birimlerinde yazdırır
  - Çoğu sistem, geçmiş durum bilgilerini yeniden kullanmak amacıyla kayıt defterinde saklar.



Sistem Programları (Devam)

- **Dosya işlemleri**
  - Dosya oluşturmak ve oluşturulan dosyalar üzerinde değişiklikler yapmak üzere metin editörleri mevcuttur.
  - Dosya içeriğinde arama yapmak ya da içerik üzerinde değişiklikler yapmak için özel komutlar barındırır.
- **Programlama Dili Desteği** - Derleyiciler, yorumlayıcılar, hata ayıklayıcılar ve çeviriciler sağlar.
- **Program yükleme ve çalıştırma** – Tam yükleyiciler, bağlantı editörleri, mutlak yükleyiciler, yüksek seviyeli diller ve makine dili için derleyiciler…
- **İletişim** - Bilgisayarlar, kullanıcılar ve işlemler arasında sanal bağlantılar kuran bir mekanizma sağlar.
  - Kullanıcıların birbirlerine mesaj göndermesi, web sayfalarına ulaşması, e-mail ve dosya gönderip alması gibi işlemlerine izin verir.

İşletim Sistemi Tasarımı ve Uygulaması

- İşletim sistemi tasarımı ve uygulaması kolay olmamakla birlikte geçerliliğini koruyan başarılı tasarımlar mevcuttur.
- İşletim sistemlerinin iç yapıları büyük farklılıklar gösterebilir.
- Bu farklılıkların başında, işletim sisteminin tasarlanma amacındaki ve istenilen özelliklerdeki farklılıklar gelir.
- İşletim sistemi tasarımında seçilen donanım önemli bir etkiye sahiptir.
- Sistemden Beklenenler ve Kullanıcının Beklentileri
  - Kullanıcının beklentileri– işletim sisteminin, güvenilir, hızlı, güvenli, öğrenilmesi kolay, kullanımı kolay ve rahat olması.
  - Sistemden Beklenenler–Tasarımı kolay, uygulanabilir, sürdürülebilir ayrıca esnek, güvenilir, hatasız ve kaynakları verimli kullanan bir yapıda olması.

İşletim Sistemi Tasarımı ve Uygulaması (Devam)

- Şu iki kavramın birbirinden ayrılması çok önemlidir.

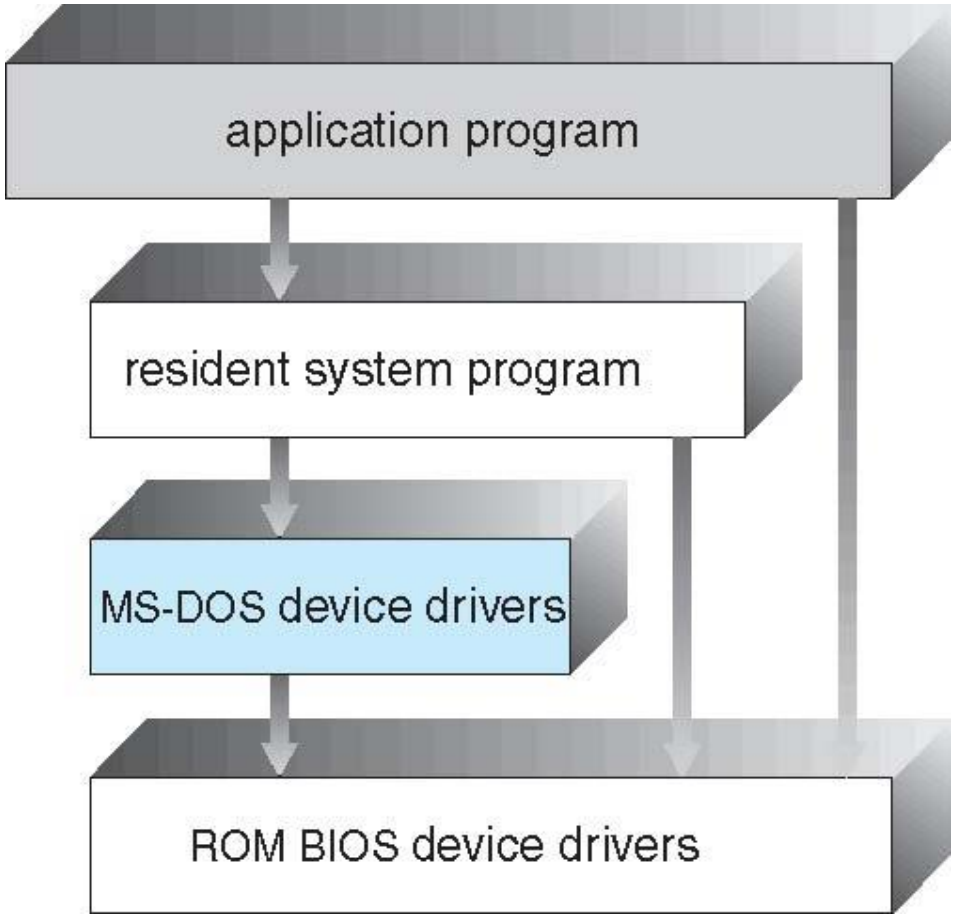
  Anlayış/Politika: Ne yapılacak?
  İşleyiş: Nasıl yapılacak?

- İşleyiş bir işin nasıl gerçekleştirileceğini belirtirken; anlayış, ne yapılacağını belirtir.
  - Anlayışın işleyişten ayrılması çok önemli bir ilkedir, bu ilkenin uygulanması ileride anlayış değiştirilmek istendiğinde bize çok büyük esneklik sağlar.

Basit Yapı

- MS-DOS – En az bellek kullanıp, maksimum fonksiyonelliği sağlamak üzere yazılmıştır.
  - Modüler değildir.
  - Bazı yapılara sahip olmasına karşın arayüzleri ve fonksiyonları ayrık(modüler) değildir.

MS-DOS Katman Yapısı

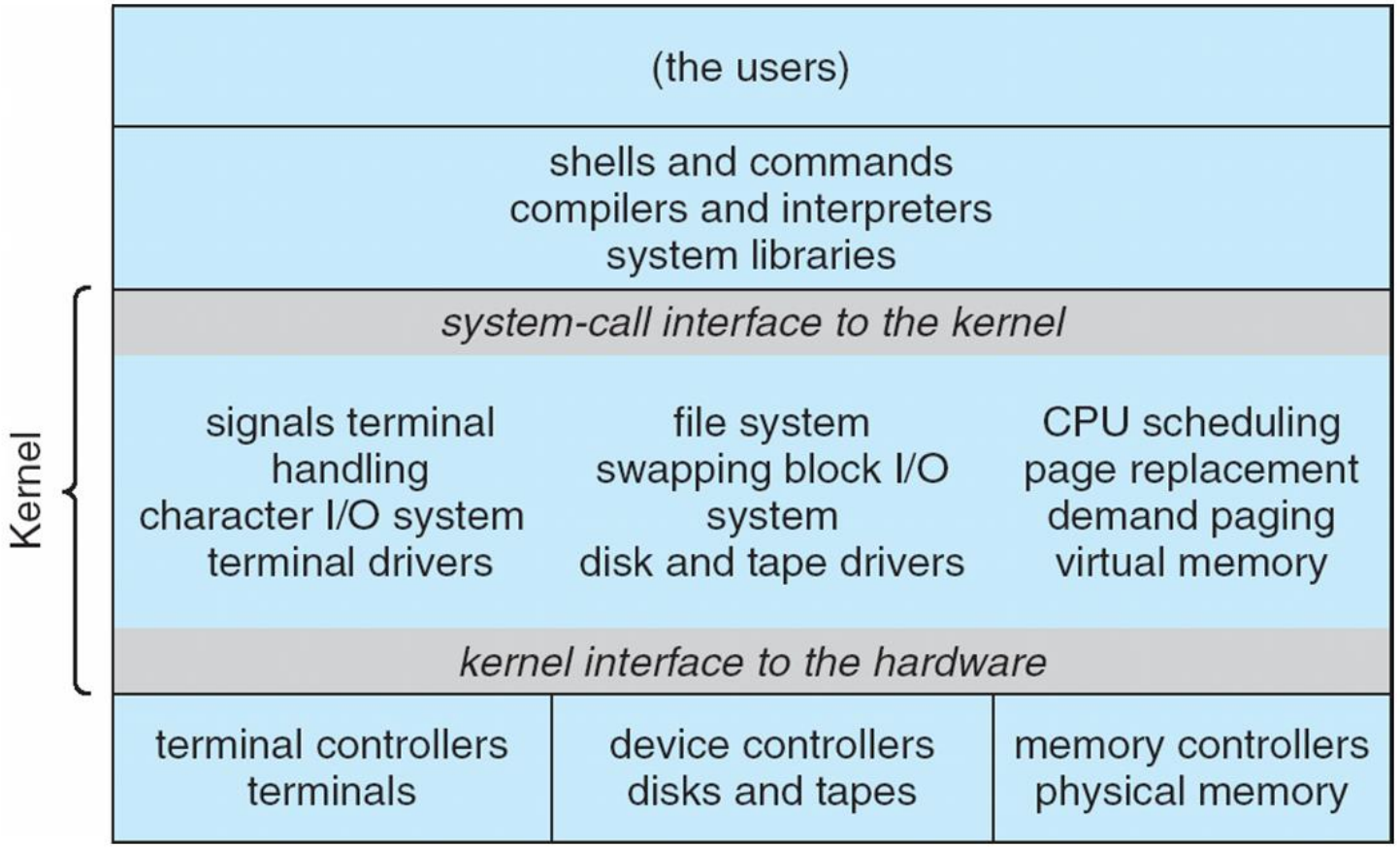


Katmanlı Yaklaşım

- İşletim sistemleri katmanlara bölünmüştür, her katman alt katmanların üzerine inşa edilmiştir. En alt katman (0. katman), donanım; en üst katman (N. katman) ise kullanıcı arayüzüdür.

- Modülerlik ile, seçilen katman yalnızca alt katmanlara ait servis ve fonksiyonları kullanır.

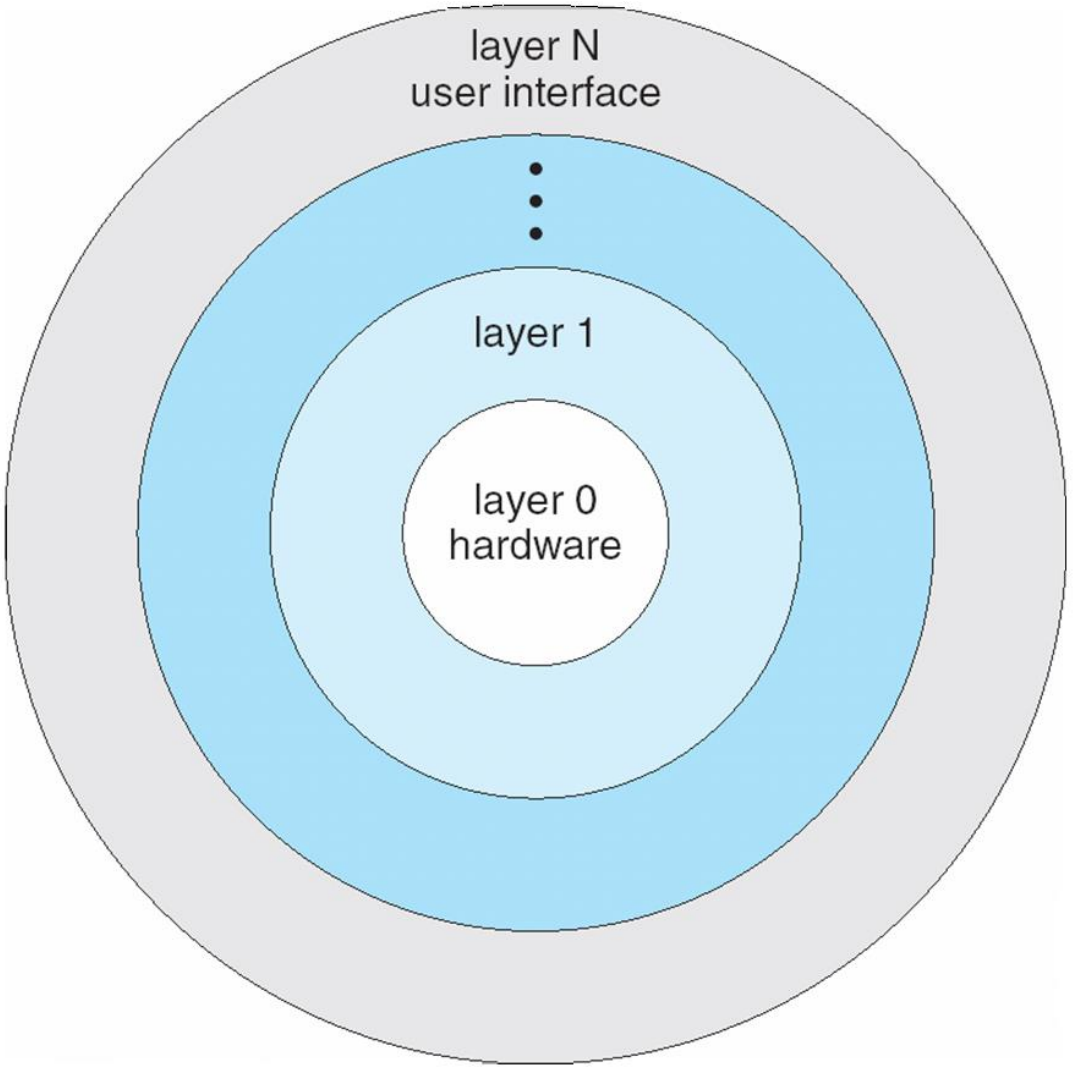
Geleneksel UNIX Sistem Yapısı



UNIX

- UNIX – donanım sınırlaması nedeniyle orijinal UNIX kısıtlı bir yapıya sahiptir. UNIX OS'i iki kısıma ayırabiliriz:
  - Sistem Programları
  - Çekirdek (Kernel)
    - Sistem çağrıları ile donanım arasındaki her şeyi ifade eder.
    - Bellek yönetimi, kaynak ayrılması, işlemci planlaması ve diğer işletim sistemi görevlerini yapar, ilk katman için çok sayıda fonksiyon yerine getirir.

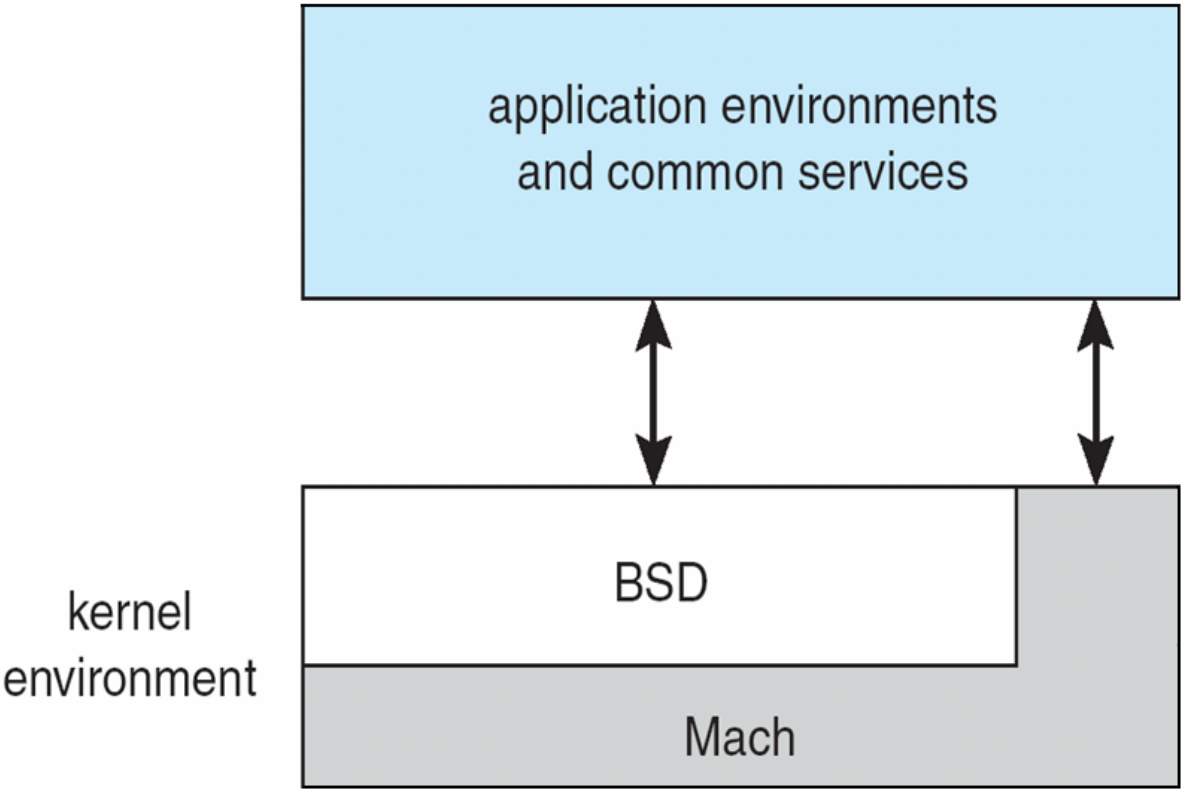
Katmanlı İşletim Sistemi

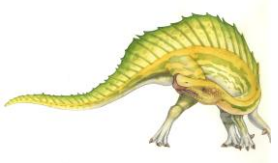


Microkernel Sistem Yapısı

- Kernel'deki 'fazlalıkları' olabildiğince kullanıcı (*user*) alanına taşır.
- Kullanıcı modülleri arasında iletişim, mesaj geçişi yöntemi ile sağlanır.
- Artıları:
  - Mikrokerneli genişletmek kolaydır.
  - İşletim sistemini yeni mimarilere taşıması kolaydır.
  - Kernel modda çalışan kod azaldığı için daha güvenilirdir.
  - Daha güvenlikli
- Eksileri:
  - Kullanıcı alanı ile kernel alanı arasındaki iletişimden kaynaklanan performans kaybı

Mac OS X Yapısı

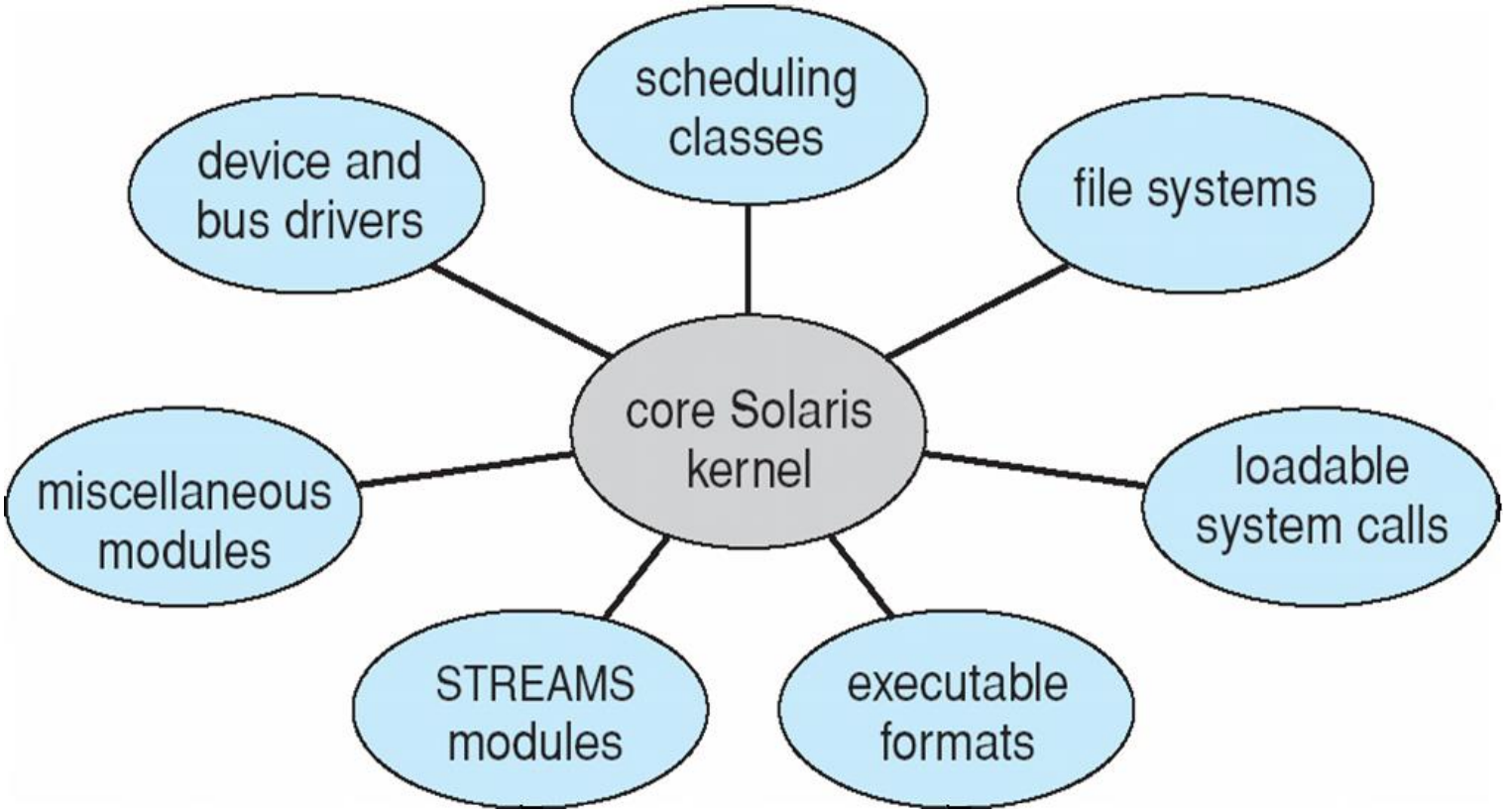




Modüler Yapı

- Günümüz işletim sistemleri kernel modülleri uygulamaktadır.
  - Nesne yönelimli yaklaşım hakimdir.
  - Her bir çekirdek bileşeni bağımsızdır.
  - Her modül birbiriyle bilinen arayüzler üzerinden konuşur.
  - Her biri kernel içerisinde ihtiyaç duyulması halinde yüklenebilirdir.
- Genel olarak, katmanlı yapıya benzer fakat çok daha esnektir.

Solaris Modüler Yaklaşımı



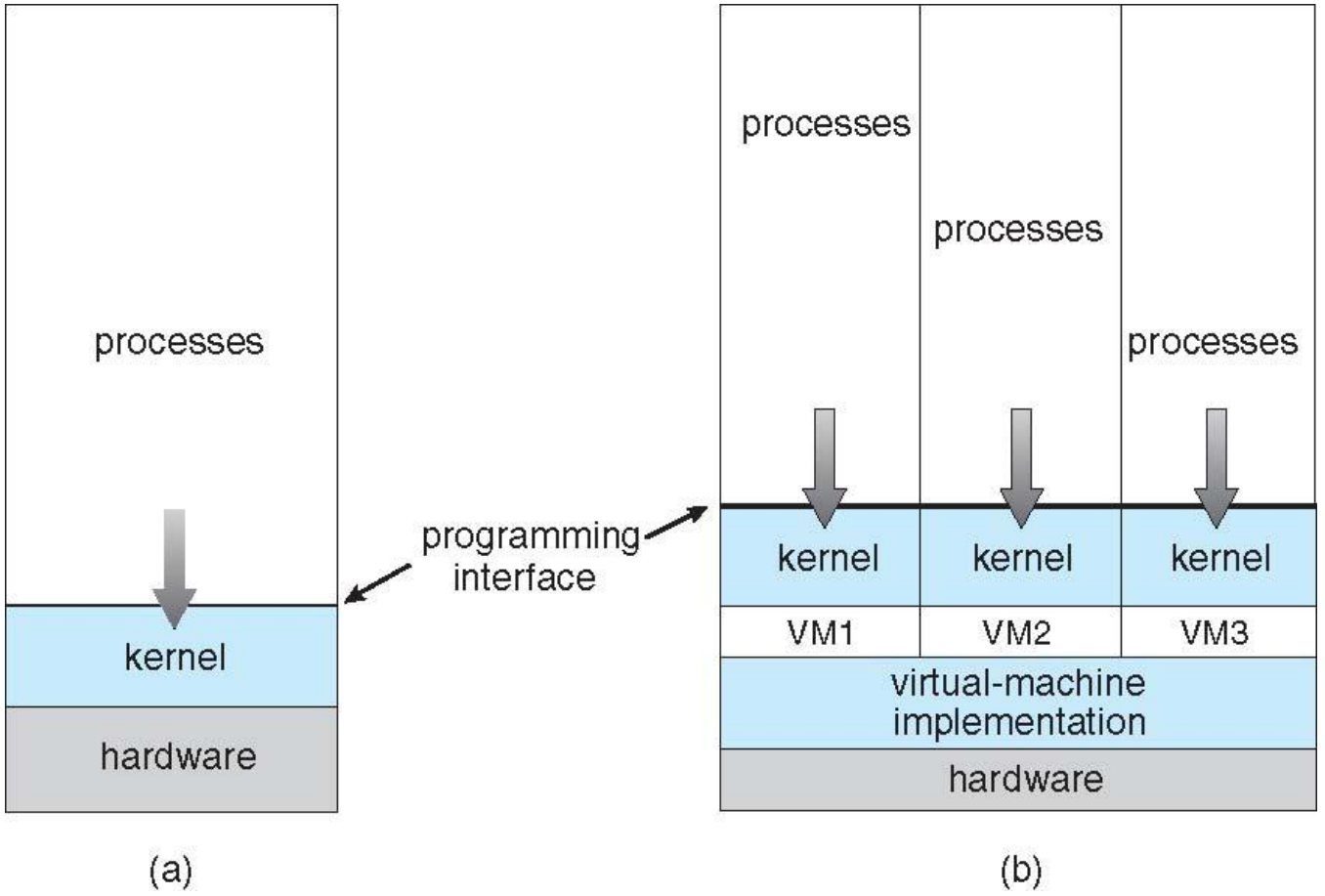
Sanal Makineler

- Sanal bir makine (**virtual machine)** katmanlı yaklaşımının mantıksal (logical) bir sonucudur.
- Donanım + İşletim Sistemi Çekirdeği = Donanım
- Sanal bir makine alttaki donanıma benzer bir arayüz sağlar
- Ev sahibi işletim sistemi bir illüzyon yapar.
  - Kendi işlemcisi ve sanal belleği olan bir proses oluşturur
- Her bir konuk işletim sistemine donanımın sanal bir kopyası sağlanır.

Sanal Makineler Tarihi ve Faydaları:

- İlk kez 1972 yılında IBM'in anabilgisayarlarında kullanıldı.
- Tek bir fiziksel donanım üzerinde farklı işletim sistemleri yürütme imkanı sunar.
- Kurulan her işletim sistemini diğerlerinden izole eder.
- Dosya paylaşımına kontrollü olarak izin verilebilir.
- Ağ altyapısı yardımıyla birbirleriyle veya diğer fiziksel sistemlerle haberleşir
- Test ve geliştirme işlemleri için kullanışlıdır.
- Daha düşük kaynak kullanan sistemlerin daha az meşgul sistemler üzerinde çalışma olanağı sağlar
- "Açık Sanal Makine Biçimi", çok sayıda farklı sanal makine çalıştırmanıza izin verir.

Sanal Makineler (Devam)



(a) sanal olmayan makine (b) sanal makine

Para-virtualization

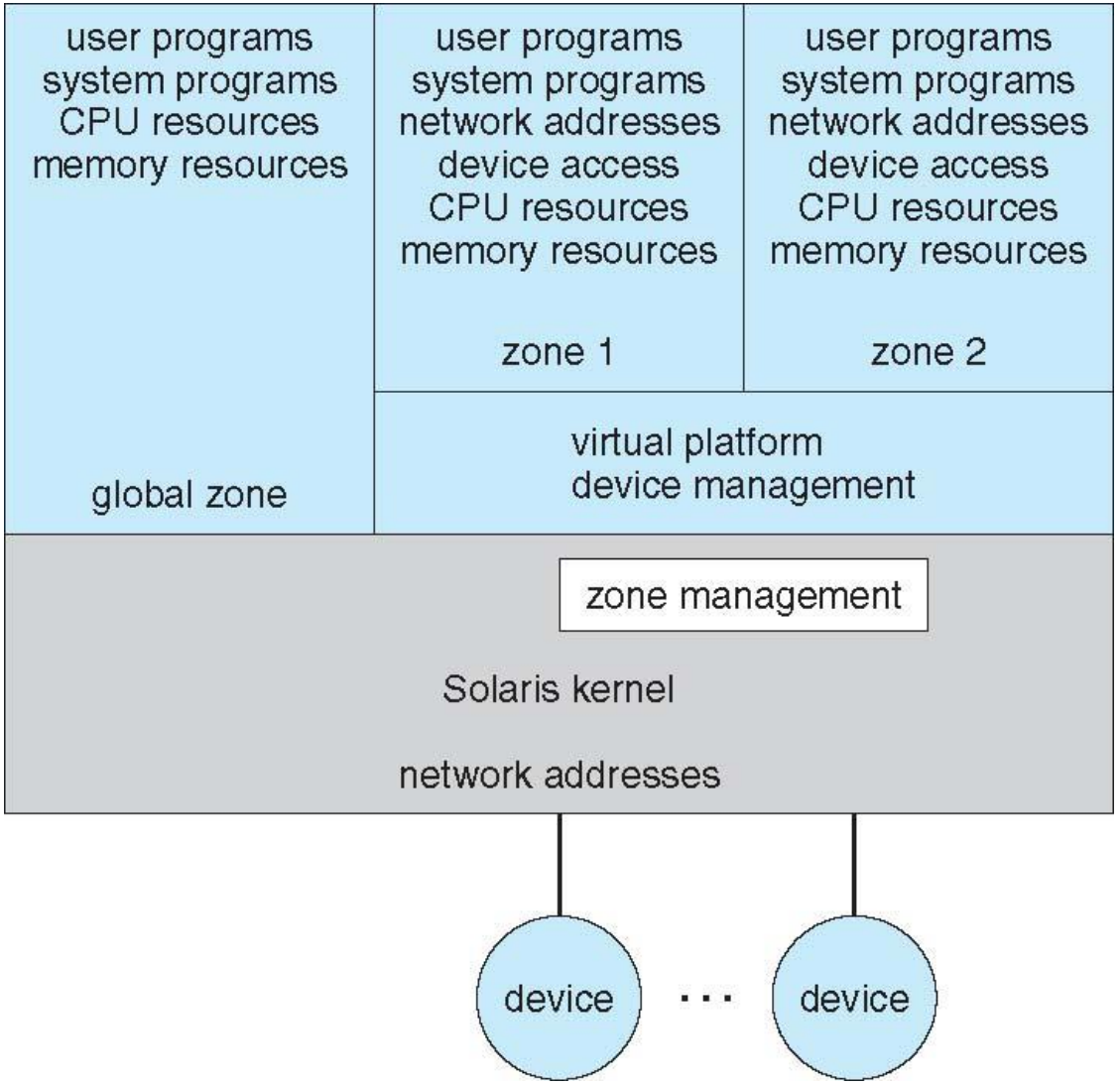
- Donanıma benzer ama tıpkısının aynısı olmayan konuk işletim sistemleri
- Donanım üzerinde çalıştırabilmek için konuk işletim sistemi de modifiye edilmelidir.

- Konuk bir işletim sistemi olabilir, ya da "taşıyıcı - container" içinde çalışan Solaris 10 uygulamaları da olabilir.

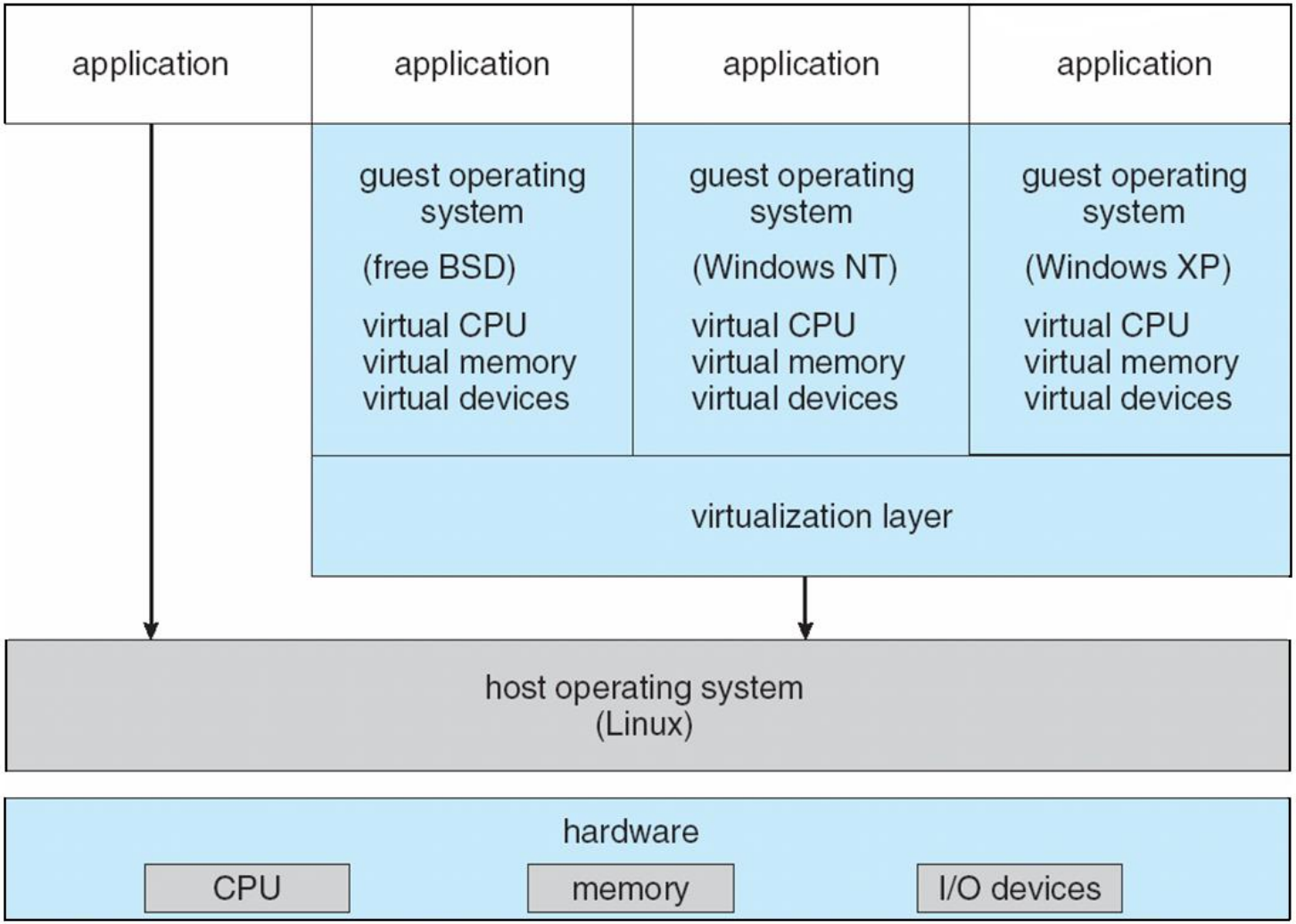
Sanallaştırma

- Sanallaştırma zordur– temel işletim sisteminin bir yedeğinin alınması gerekir.
  - Genellikle, kullanıcı modunda çalışır. Sanal kullanıcı modu ve sanal çekirdek modu oluşturulur.
- Gerçek işletim sisteminden daha yavaş çalışma problemi olabilir.
- Donanım desteği gerekir.
  - Daha fazla destek -> daha iyi sanallaştırma
  - i.e. AMD “host” ve “guest” modları sağlar.

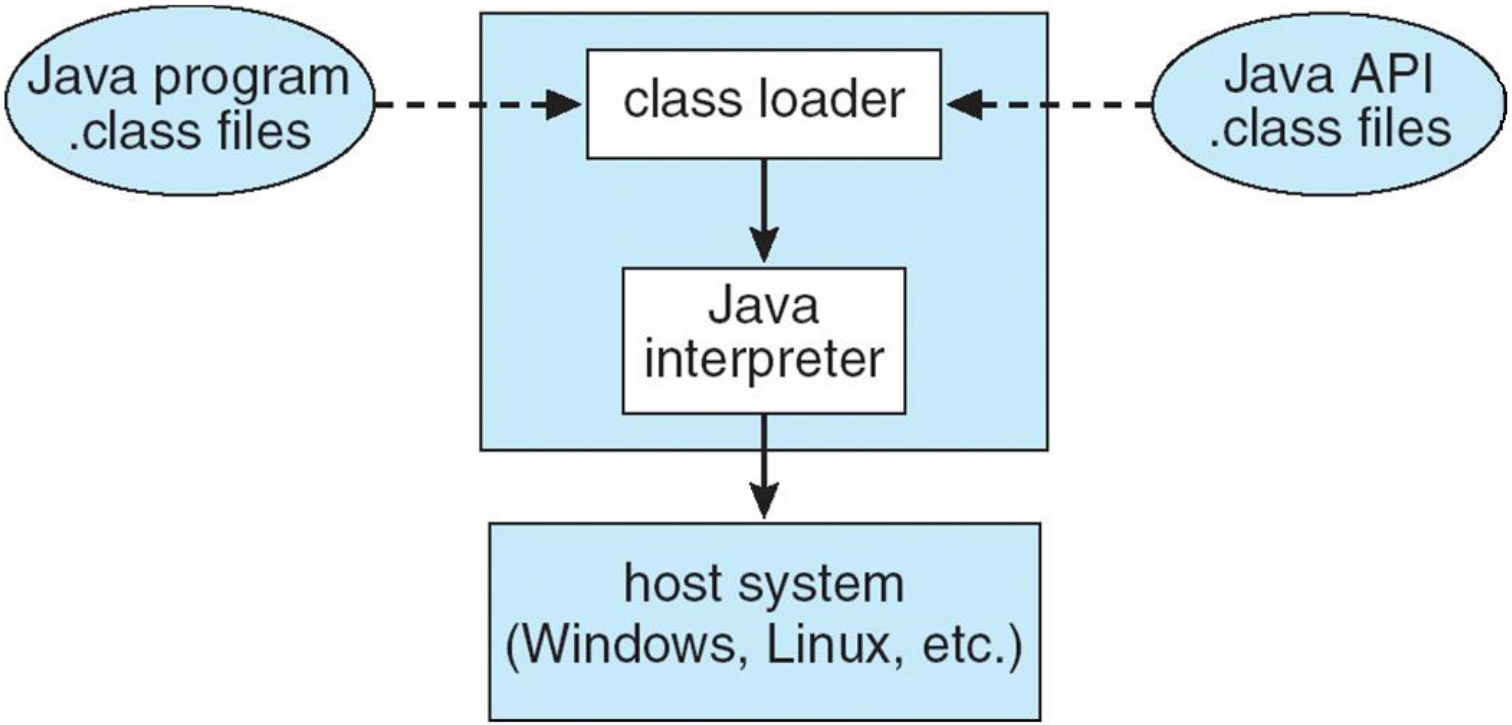
Solaris 10 with Two Containers



VMware Mimarisi

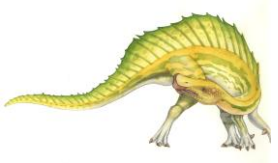


Java Sanal Makinesi



İşletim sisteminde Hata Ayıklama

- **Hata yakalama - Debugging** hataları ve böcekleri(**bug**) bulma ve ayıklama işlemidir.
- İşletim sistemleri hata bilgilerini içeren **log dosyaları** oluşturur.
- Bir uygulamanın hatası prosesin bellek görüntüsünü tutan bir dosya oluşturabilir (**core dump** file). capturing memory of the process
- İşletim Sistemi hatası çekirdek bellek görüntüsünü taşıyan bir dosya üretebilir (**crash dump** file)
- Çöküşlerden sonra, sistem performansı optimize edilir.
- Kernighan's Kuralı: "Hata ayıklama kod yazmaktan iki kat zordur. Bu yüzden, kodu zekice tasarlayıp yazabilirsin, ama hata ayıklamak için yeteri kadar akıllı olmayabilirsin "
- FreeBSD, Mac OS X, Solaris içindeki DTrace aracı, kod çalışırken veriyi takip ederek hataları yakalar



Solaris 10 dtrace Sistem Çağrısı Gönderiyor

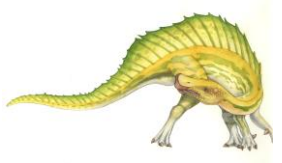
```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
  0 -> XEventsQueued                        U
  0   -> _XEventsQueued                     U
  0     -> _X11TransBytesReadable           U
  0     <- _X11TransBytesReadable           U
  0     -> _X11TransSocketBytesReadable     U
  0     <- _X11TransSocketBytesreadable     U
  0     -> ioctl                            U
  0       -> ioctl                          K
  0         -> getf                         K
  0           -> set_active_fd              K
  0           <- set_active_fd              K
  0         <- getf                         K
  0         -> get_udatamodel               K
  0         <- get_udatamodel               K
...
  0         -> releasef                     K
  0           -> clear_active_fd            K
  0           <- clear_active_fd            K
  0           -> cv_broadcast               K
  0           <- cv_broadcast               K
  0         <- releasef                     K
  0       <- ioctl                          K
  0     <- ioctl                            U
  0   <- _XEventsQueued                     U
  0 <- XEventsQueued                        U
```

Operating System Generation

- İşletim sistemleri her donanım üzerinde çalışabilecek biçimde tasarlanmıştır; sistem, her bilgisayarda çalışacak biçimde konfigüre edilmelidir.
- Bu amaçla SYSGEN programı, donanım sistemine ilişkin bilgi edinir.
- *Booting* – bilgisayarın kernel yüklenerek başlatılması işlemidir.
- *Bootstrap program* – Rom bellekte saklanan bir kod parçasıdır ki bu kod parçası kernelin yerini belirleyip, belleğe yükler ve kerneli başlatır

Sistem Önyükleme

- İşletim sistemi, donanımın üzerinde hazır olmalıdır ki, donanım onu başlatabilsin.
  - **bootstrap loader**(küçük bir kod parçası), kerneli bulur, belleğe yükler ve çalıştırır.
  - Bazen iki aşamada gerçekleşir sabit konumdaki **boot block, bootsrap loader**'i yükler ve bootstrap çalışır.
  - Sisteme güç verildiğinde çalışmaya belli bir bellek konumundan başlanır.
    - Firmware, ilk önyükleme kodlarını tutmak için kullanılır.



Bölüm 2 Sonu