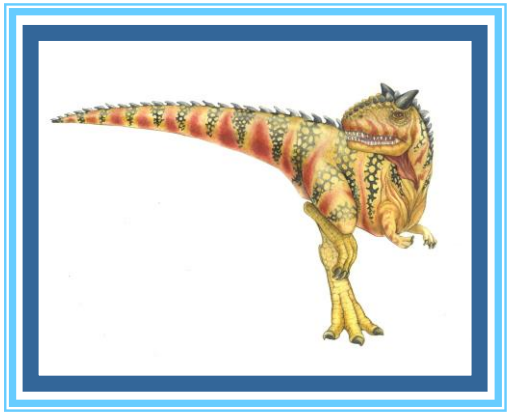
Bölüm 2: İşletim Sistemi Yapıları



Dr. Öğretim Üyesi Abdullah SEVİN

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 tanımak
- İşletim sistemi hizmetlerini sağlamak için sistem çağrılarının nasıl kullanıldığını göstermek
- İşletim sistemlerini tasarlamak için monolitik, katmanlı, mikrokernel, modüler ve karma stratejileri karşılaştırmak
- İşletim sistemi performansını izlemek için araçları tanımak
- İşletim sistemlerinin kurulum, özelleştirilme ve önyükleme işlemlerinin açıklanması

İşletim Sistemi Servisleri

- İşletim sistemleri, programların çalışması için kullanıcılara ve programlara uygun bir ortam sağlar.
- İşletim sistemleri kullanıcılara yardımcı olacak hizmetler 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)**, **Toplu iş (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.

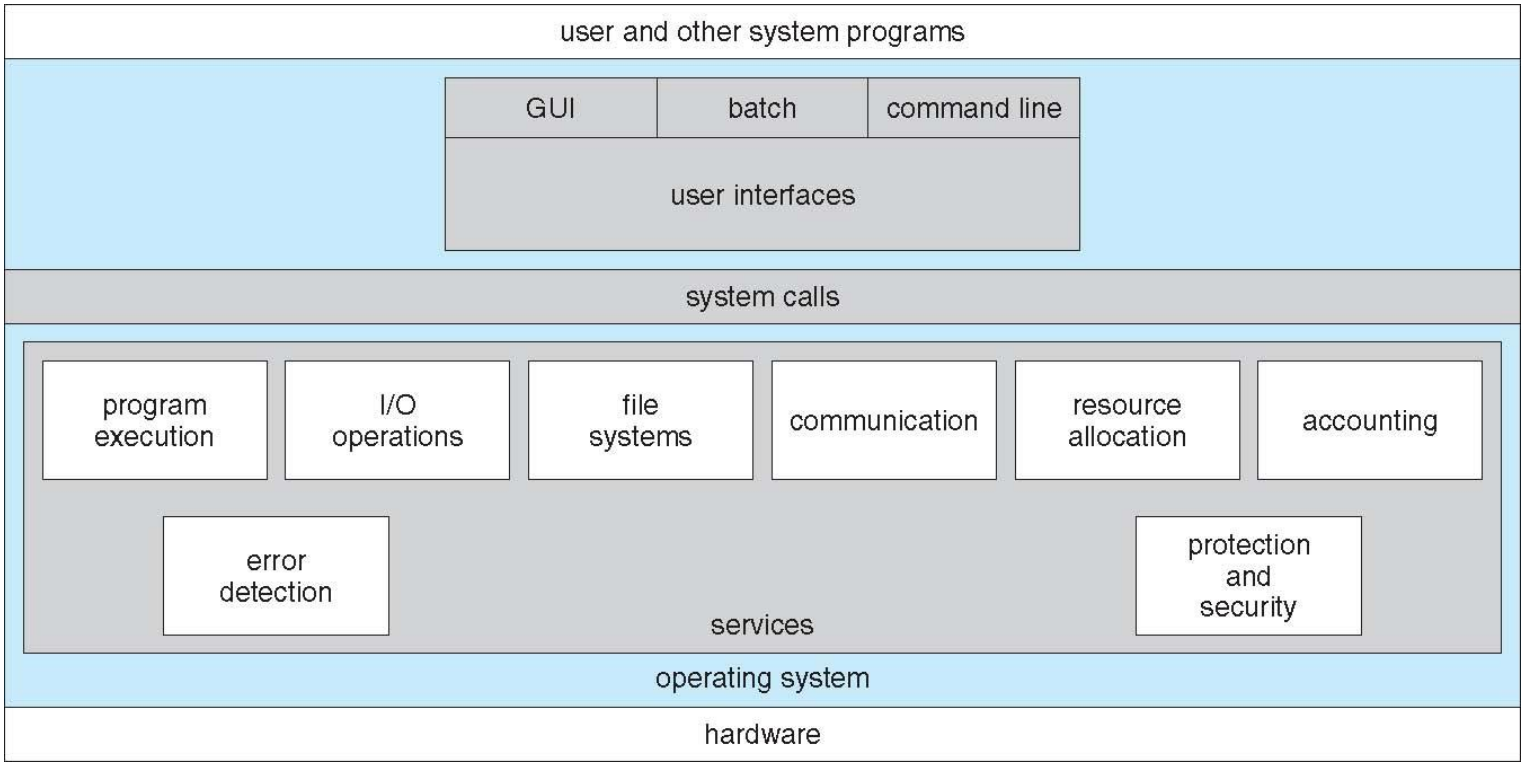
İşletim Sistemi Servisleri(Devam)

- **Communications (Haberleşme)** – Sistemdeki bir *proses* ağ üzerinden başka bir prosese ya da aynı bilgisayardaki prosese bilgi paylaşımı gerçekleştirmek 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 (Doğru ve tutarlı hesaplama için).
  - Hata ayıklama araçları, kullanıcı ve programcının sistemi etkin olarak kullanabilmesine çok büyük katkı sağlar.

İşletim Sistemi Servisleri(Devam)

- Bazı işletim sistemi servisleri de, kaynak paylaşımını gerçekleştirerek sistemin kendi kendine etkin bir biçimde işlemesini 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/Logging (Kayıtlar-hesaplar) –** Kullanıcıların ne kadar ve ne tür kaynak kullandığını takip edilmelidir.
  - **Protection and security (Koruma ve Güvenlik) -** Çok sayıda proses 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



Komut-Satırı (Command Lıne Interface- CLI)

- Komut satırı ya da **komut yorumlayıcısı** doğrudan sisteme komut verme imkanı sağlar.
- Linux, UNIX ve Windows dahil olmak üzere çoğu işletim sistemi, komut yorumlayıcısına bir işlem başlatıldığında veya bir kullanıcı ilk kez oturum açtığında (etkileşimli sistemlerde) çalışan özel bir program olarak davranır.
- Çoklu komut yorumlayıcısının seçilebileceği sistemlerde, yorumlayıcılar kabuk (shell) olarak bilinir.
- Kabuk bir işletim sisteminin hizmetlerine erişim için bir kullanıcı arabirimidir.
- Örneğin, UNIX ve Linux sistemlerinde, bir kullanıcı C kabuğu, Bourne-Again kabuğu, Korn kabuğu ve diğerleri gibi birkaç farklı kabuk arasından seçim yapabilir.
- İşletim sistemi çekirdeğinin çevresindeki en dış katman olduğu için bir kabuk adı verilir.

Komut-Satırı (Command Lıne Interface- CLI)

- Komut satırından doğrudan ya da sistem servisleri vasıtasıyla işletim sistemi çekirdeğine (kernele) ulaşılır.
- Bazen birçok özellik gerçekleştirilir– kabuk (**shells)**
- Temel olarak, kullanıcıdan komut alınır ve çalıştırılır.
  - Bazen bu komutlar bir program adı bazen de yerleşik bir komut olabilir.
    - Komutlar parametreler ile değişik davranış gösterir.
  - Yeni özellikler eklemek kabuk modifikasyonu gerektirmez

Komut Yorumlayıcısı

1. root@r6181-d5-us01:~ (ssh)

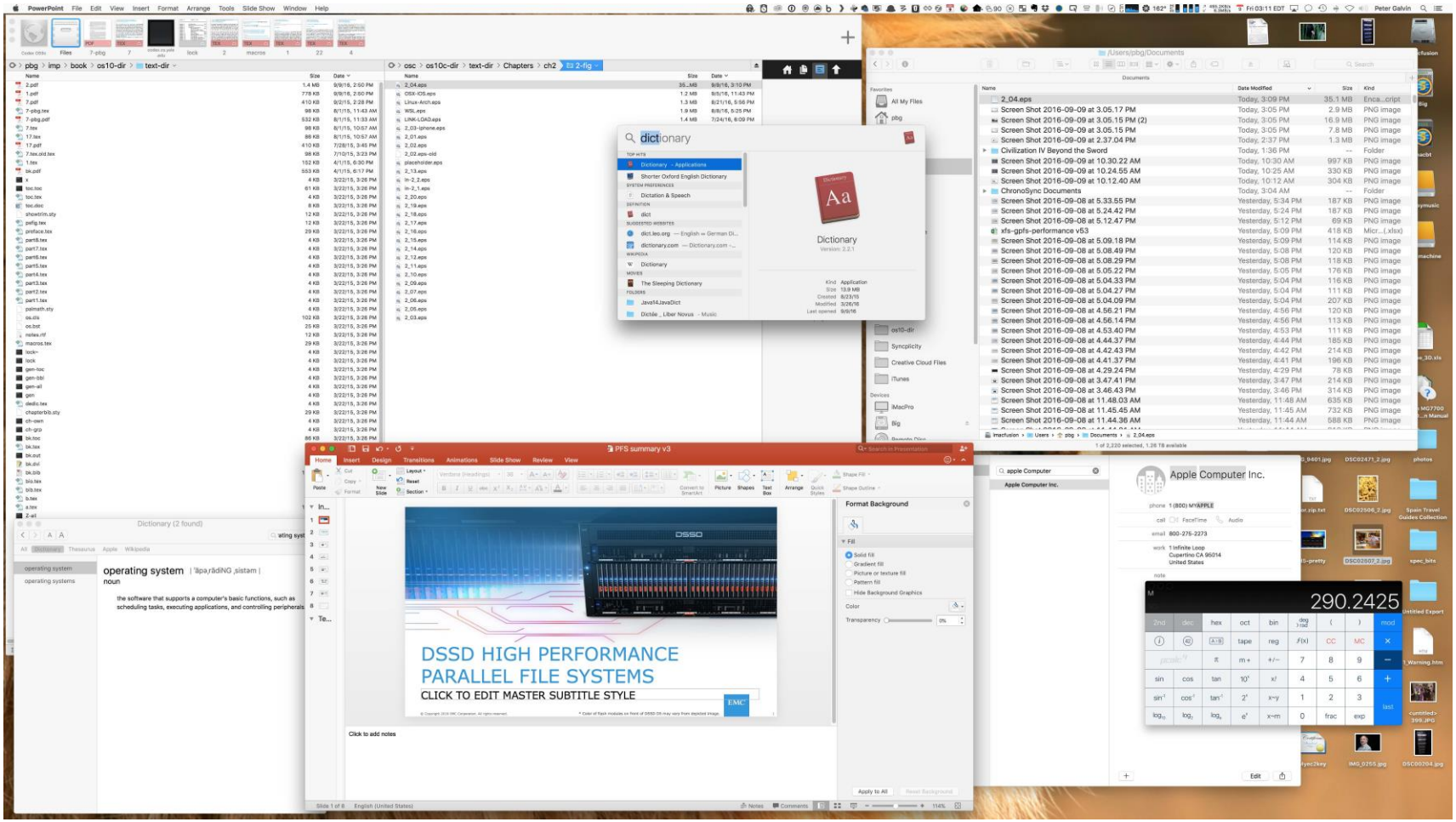
root@r6181-d5-u... ⌘1 | ssh ⌘2 | root@r6181-d5-us01... ⌘3

```
Last login: Thu Jul 14 08:47:01 on ttys002
iMacPro:~ pbg$ ssh root@r6181-d5-us01
root@r6181-d5-us01's password:
Last login: Thu Jul 14 06:01:11 2016 from 172.16.16.162
[root@r6181-d5-us01 ~]# uptime
 06:57:48 up 16 days, 10:52,  3 users,  load average: 129.52, 80.33, 56.55
[root@r6181-d5-us01 ~]# df -kh
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_ks-lv_root
                       50G   19G   28G  41% /
tmpfs                 127G  520K  127G   1% /dev/shm
/dev/sda1             477M   71M  381M  16% /boot
/dev/dssd0000         1.0T  480G  545G  47% /dssd_xfs
tcp://192.168.150.1:3334/orangefs
                       12T  5.7T  6.4T  47% /mnt/orangefs
/dev/gpfs-test         23T  1.1T   22T   5% /mnt/gpfs
[root@r6181-d5-us01 ~]#
[root@r6181-d5-us01 ~]# ps aux | sort -nrk 3,3 | head -n 5
root      97653 11.2  6.6 42665344 17520636 ?   S<Ll Jul13 166:23 /usr/lpp/mmfs/bin/mmfsd
root      69849  6.6  0.0      0     0 ?        S    Jul12 181:54 [vpthread-1-1]
root      69850  6.4  0.0      0     0 ?        S    Jul12 177:42 [vpthread-1-2]
root       3829  3.0  0.0      0     0 ?        S    Jun27 730:04 [rp_thread 7:0]
root       3826  3.0  0.0      0     0 ?        S    Jun27 728:08 [rp_thread 6:0]
[root@r6181-d5-us01 ~]# ls -l /usr/lpp/mmfs/bin/mmfsd
-r-x------ 1 root root 20667161 Jun  3  2015 /usr/lpp/mmfs/bin/mmfsd
[root@r6181-d5-us01 ~]#
```

Kullanıcı Arayüzü - GUI

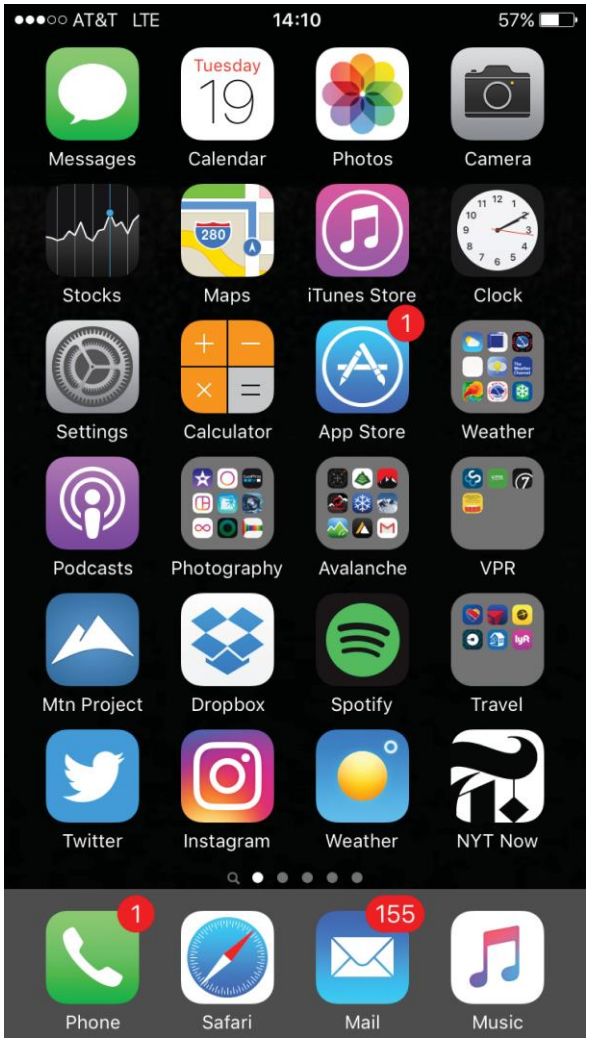
- Kullanıcı dostu, bilgi ve görsellerle donatılmış arayüz.
  - Genellikle Fare, klavye ve monitör.
  - Programları, dosyaları vb. temsil eden simgeler (**Icons**) mevcuttur.
  - Fare yardımı ile çeşitli eylemler gerçekleştirilir(bilgi getirme, seçenekler, fonksiyon çalıştırma, dizin açma)
  - İlk kez Xerox PARC tarafından kullanılmıştır.
- Günümüzdeki çoğu sistem hem CLI (Command-line interface/komut satırı arayüzü), hem GUI(grafik arayüz) barındırır.
  - Microsoft Windows, grafik arayüzlüdür ve CLI (Bourne-Shell) barındırır.
  - Apple Mac OS X'de "Aqua" GUI arayüzü ile UNIX çekirdeği altında çalışır.
  - Unix ve Linux CLI ve opsiyonel olarak GUI barındırır(Java Desktop, KDE).

The Mac OS X Arayüzü



Touchscreen Interfaces

- Dokunmatik ekranlar yeni bir arayüze ihtiyaç duyarlar
  - Mouse yok yada istenmez
  - Eylemler ve seçimler el hareketlerine dayalıdır
  - Sanal klavye (yazım için)
- Ses komutları vb.



Sistem Çağrıları

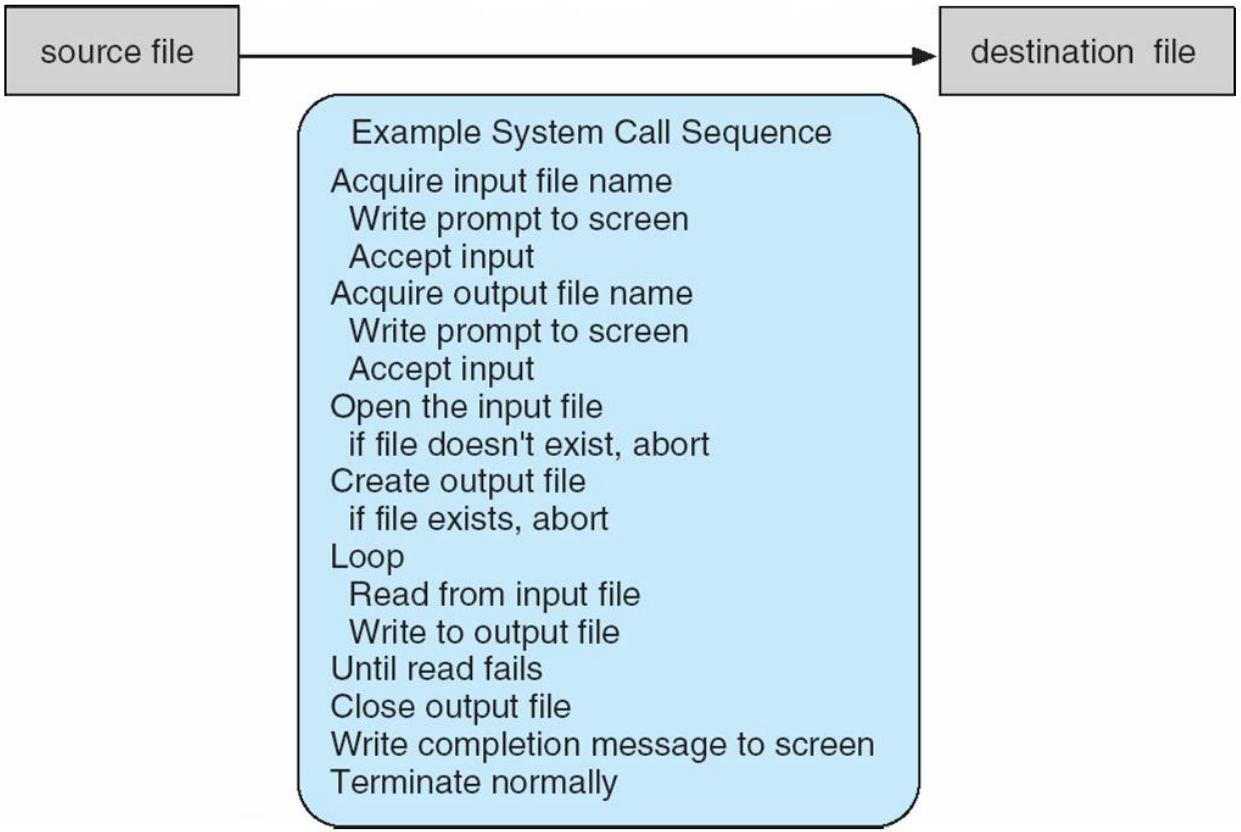
- Sistem çağrıları, servisler için bir arayüzdür. Sistem çağrıları sayesinde yazılımcı doğrudan donanıma müdahale etmez. Donanım üzerinde gerçekleştireceği işlemi sistem çağrısı kullanarak gerçekleştirir. Bu sayede olası sistem hatalarından kaçınılmış olur.
- Bu sistem çağrıları işletim sisteminden işletim sistemine farklılık gösterir. Sistem çağrıları makineye bağımlıdır.
- Çoğunlukla yüksek seviyeli diller kullanılır (C ya da C++).
- Eğer kullanıcı modunda çalışan bir kullanıcı programı bir servise ihtiyaç duyarsa, örneğin bir dosyayı okumak isterse, bir kütüphane prosedürü ile sistem çağrısı komutu (system call instruction) çalıştırarak kontrolü işletim sistemine vermelidir.
- İşletim sistemin gelen prosedürün parametrelerine bakarak sistem çağrısına karşılık gelen işlemi gerçekleştirir ve elde edilen değeri geri döndürür.

API (Application Programming Interface)

- Sistem çağrılarının, sistemden sisteme değişiyor olması yazılımcının işini zorlaştırır. Bu yüzden programcılar, doğrudan sistem çağrılar yerine API kullanmayı tercih ederler.
- API, bir kütüphane yada ortamın (framework) sunduğu yararlı işlemler yapan foksiyonlarına verilen genel addır.
- 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.
- API ise daha geneldir.Her sistem çağrısı bir API'dir fakat her API bir sistem çağrısı değildir.
- Windows'un sistem çağrıları API'ler aracılığıyla dolaylı olarak çağrılmaktadır. Kullanıcı programları direk sistem çağrılarına ulaşamaz onun yerine API'ler aracılığı ile sistem çağrılarını aktive ederler.
- Neden doğrudan sistem çağrısı yapmak yerine API kullanılır?
  - API'ler programın her sistem üzerinde çalışabilmesini sağlar.
  - API kullanımı, her sist. sistem çağrılarını ezberlemeyi gerektirmez
  - Birden fazla sistem işlemini kısa fonksiyonlarla yapabilmeyi sağlar.

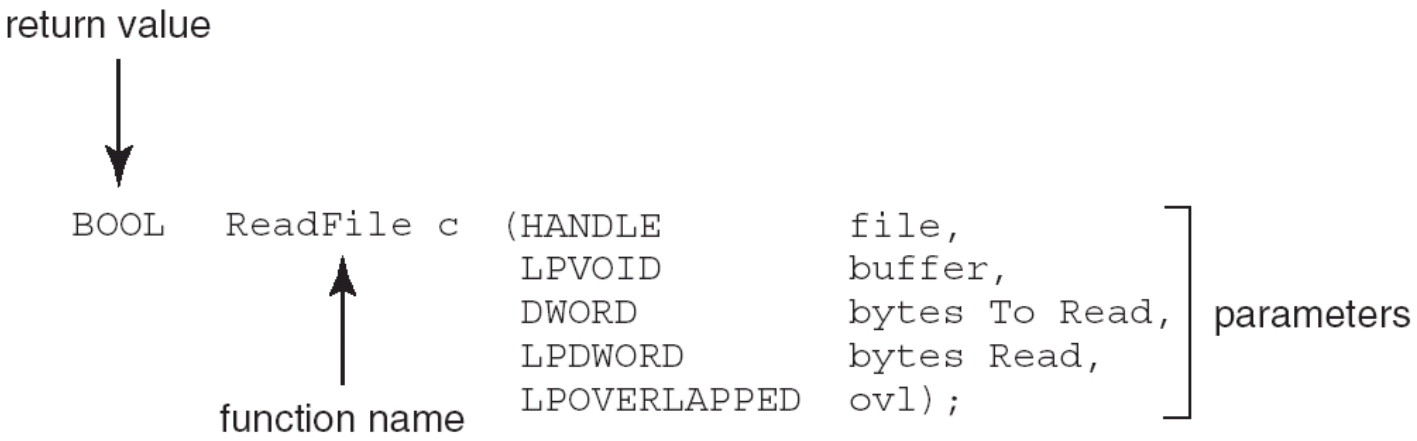
Sistem Çağrısı Örneği

- Bir dosyadan diğerine içerik kopyalamak için yapılan sistem çağrısı.



Standard API Örneği

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

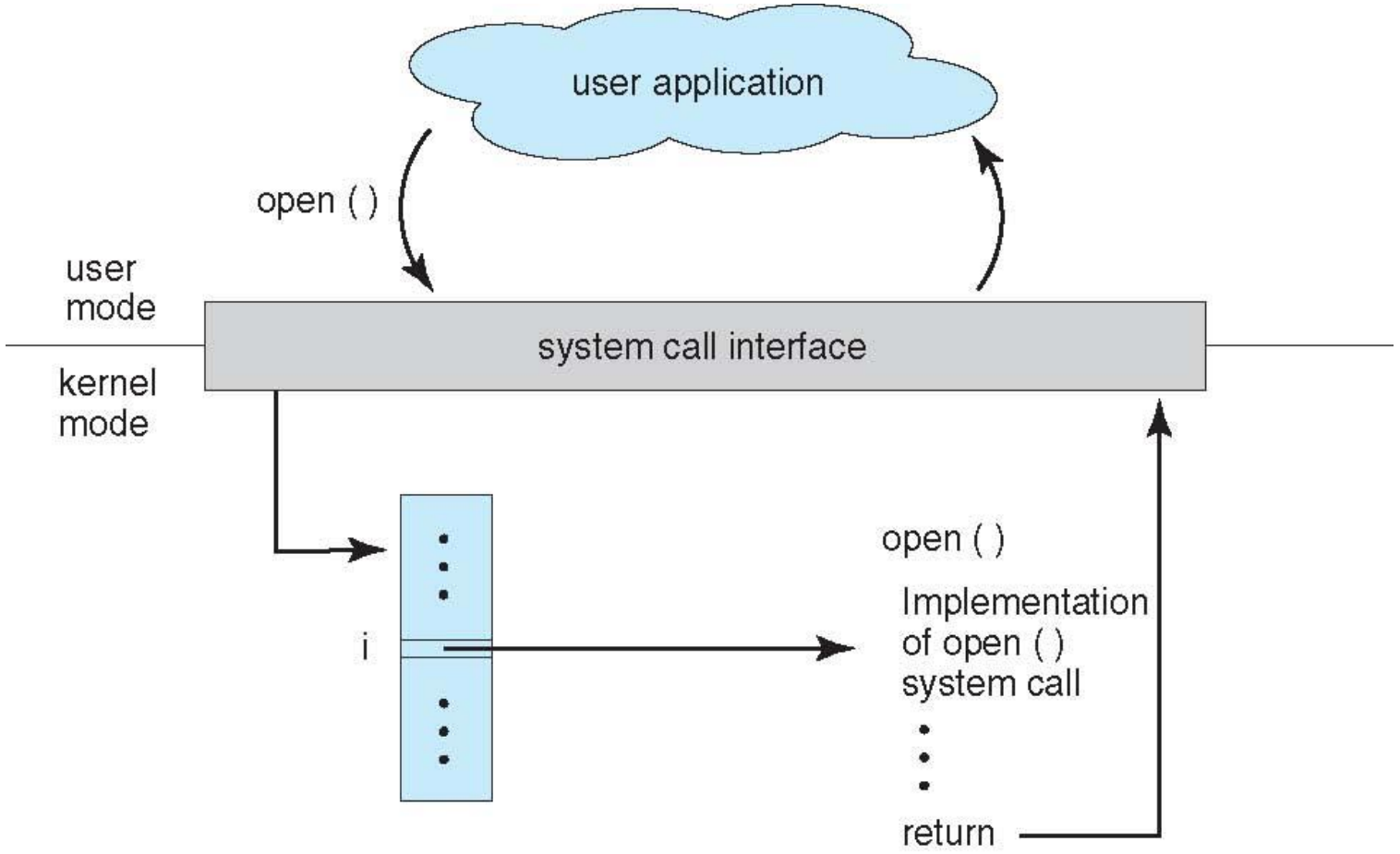


- ReadFile()'a geçen parametrelerin bir tanımı
  - HANDLE file—okuma yapılacak dosya
  - LPVOID - okunan verinin yazılacağı bellek alanı
  - DWORD bytesToRead- okunacak veri boyutu
  - LPDWORD bytesRead— en son okunan bayt miktarı
  - LPOVERLAPPED ovl—çakışmalı(overlapped) I/O kullanılıp kullanılmadığını 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.
    - Çalışma zamanında yönetim için destek kütüphaneleri kullanılır. Fonksiyonlar (Gerekli derleyici ile) bu kütüphanelerde yerleşik olarak bulunur.

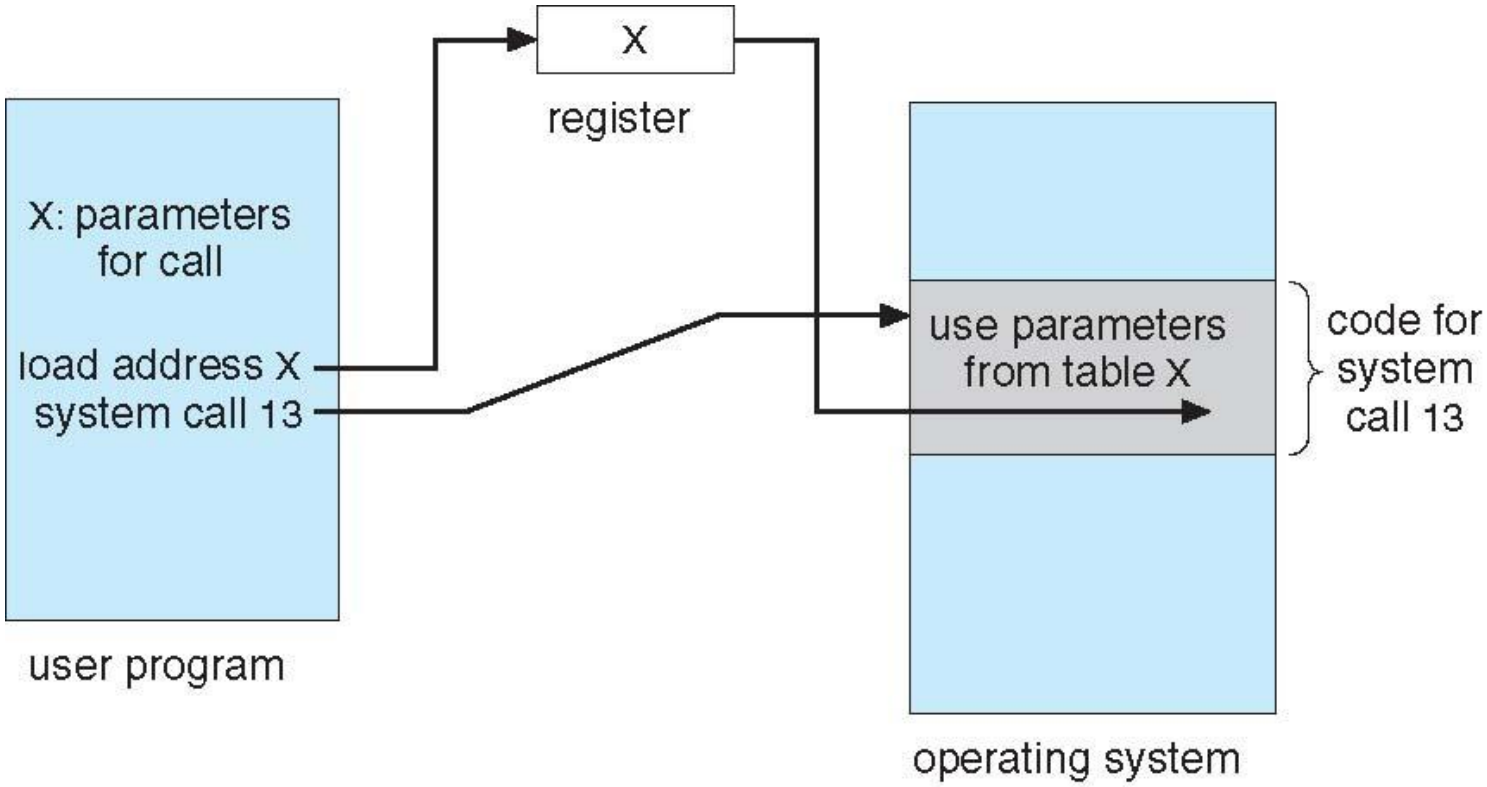
API – Sistem Çağırısı– OS İlişkisi



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:
  - En Basiti: 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, Hata varsa belleği boşalt
- 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 (Haberleşme)
  - Bağlantı oluşturmak, silmek
  - Mesaj alma, gönderme
  - Durum bilgisi transferi
  - Uzak aygıta bağlanmak, bağlantıyı sonlandırmak

Sistem Çağrı Tipleri (Devam)

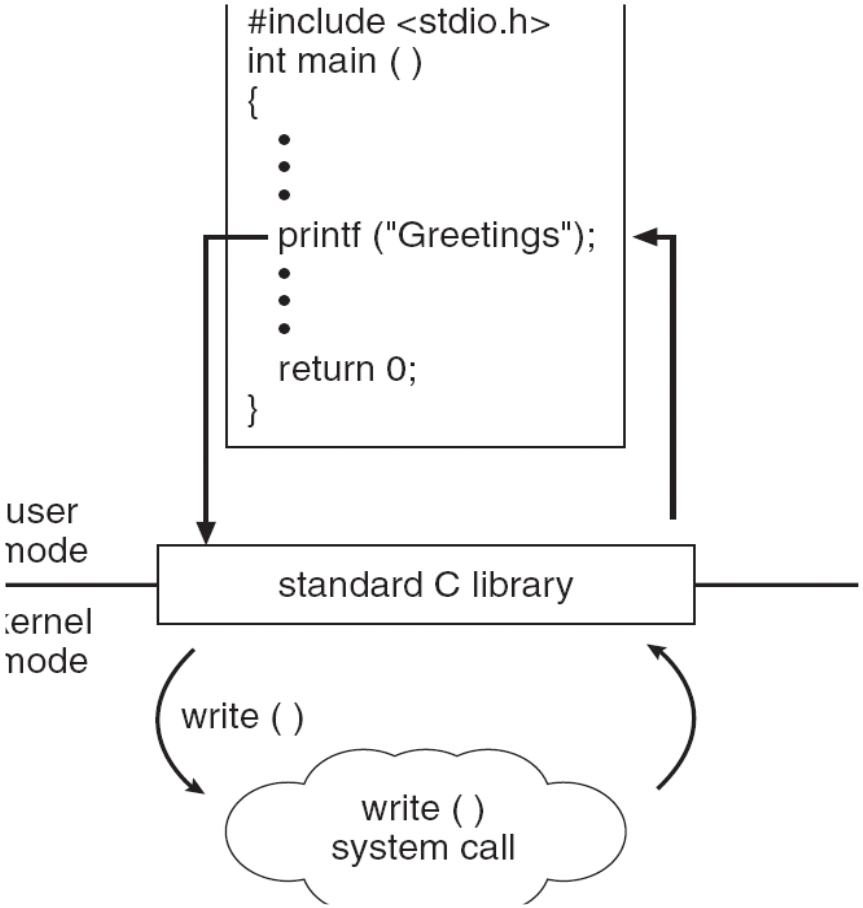
- Koruma
  - Kaynaklara erişimi kontrol etme
  - İzinleri al ve ayarla
  - Kullanıcı erişimine izin ver ve reddet

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()

Standard C Kütüphanesi Örneği

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



Sistem Servisleri

- Sistem programları, program/uygulama geliştirme ve çalıştırma ortamı sağlar. Sistem programlarını şu başlıklara ayırabiliriz:
  - Dosya işleme
  - Durum bilgisi
  - Dosya değişiklikleri
  - Programlama dili desteği
  - Program yükleme ve çalıştırma
  - Haberleşme
  - Uygulama programları
- Çoğu kullanıcının işletim sistemi hakkındaki görüşleri, gerçek sistem çağrıları değil, sistem programları tarafından tanımlanır

Sistem Servisleri

- 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 çıktıyı biçimler ve terminalde yada diğer çıkış birimlerinde yazdırır
  - Çoğu sistem, geçmiş durum bilgilerini yeniden kullanmak amacıyla kayıt defterinde saklar.

Sistem Servisleri (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.

Sistem Servisleri (Devam)

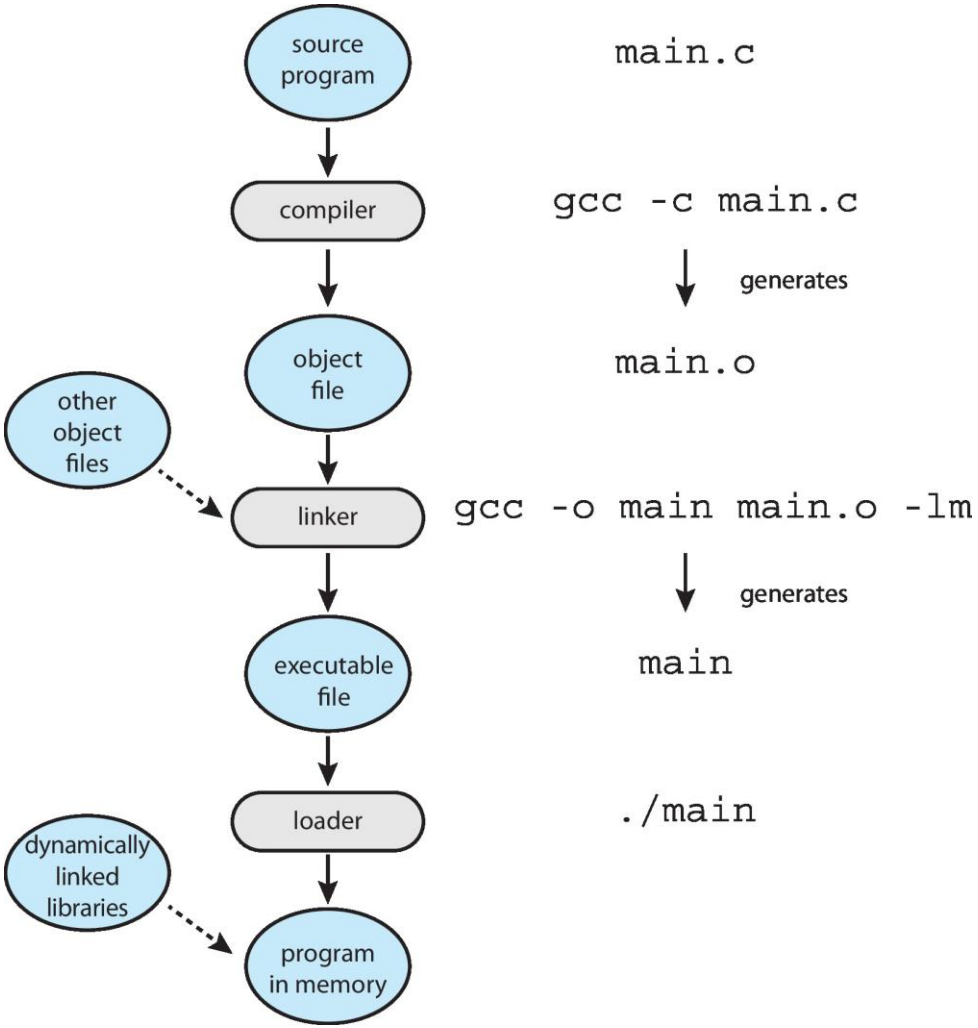
- **Arkaplan Servisleri**
  - Önyükleme sırasında başlatılanlar:
    - Sistem başlangıcında yüklenir sonra yok edilir
    - Sistem başlangıcından kapatılmasına kadar devam eder
  - Disk kontrolü, proses çizelgeleme, hata kaydı, yazdırma vb. özellikler sunar
  - Kullanıcı modunda çalışır (Çekirdek modunda değil)
  - Hizmetler, alt-sistemler olarak bilinirler

- **Uygulama Programları**
  - Sisteme ait olmayan uyg.
  - Kullanıcı tarafından yürütülür
  - Komut satırından, fare veya parmak dürtmesi ile çalıştırılabilir

Bağlayıcılar ve Yükleyiciler

- Kaynak kodu, herhangi bir fiziksel bellek konumuna yüklenecek şekilde tasarlanmış nesne dosyalarına derlenir - **relocatable object file**
- Bağlayıcı (**Linker**) bunları tek bir (ikili-binary) yürütülebilir dosya haline getirir
- Yürütülmesi için yükleyici(**loader**)tarafından belleğe getirilmelidir
  - Yeniden yerleştirme (**Relocation)**, program parçalarına son adresleri atar ve programdaki kod ve verileri bu adreslerle eşleşecek şekilde ayarlar
- Modern genel amaçlı sistemler kütüphaneleri yürütülebilir dosyalara bağlamaz
  - Aksine, **Dynamically Linked Libraries** (Windows, DLL'ler), gerektiği zaman yüklenir,
- Nesne, çalıştırılabilir dosyalar standart formatlara sahiptir, bu yüzden işletim sistemi bunların nasıl yükleneceğini ve başlatılacağını bilir

Bağlayıcı ve Yükleyicinin Rolü



Uygulamalar Neden İşletim Sistemine Özgüdür?

- Uygumalar genelde bir sisteme göre derlenir ve diğer işletim sistemlerinde genellikle çalıştırılamaz
- Her işletim sistemi kendine özgü sistem çağrıları vardır
  - Kendi dosya formatları, vb
- Uygulamalar birçok işletim sisteminde çalışadabilir
  - Python, Ruby gibi yorumlanmış dillerle yazılmışsa, (birçok OS'de yorumlayıcısı vardır)
  - Çalışan uygulamayı (Java gibi) içeren bir VM varsa bu dilde yazılmış uygulama
  - Standart dil (C gibi) kullanın, her bir işletim sisteminde ayrı ayrı derleyin ve her birinde çalışın
- **Application Binary Interface** (**ABI**), API'nin mimari eşdeğeridir, ikili kodun farklı bileşenlerinin belirli bir mimaride, CPU'da vb. belirli bir işletim sistemi için nasıl arayüz oluşturabileceğini tanımlar.

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.

  İlke: Ne yapılacak?
  Mekanizma: Nasıl yapılacak?

- Mekanizma bir işin nasıl gerçekleştirileceğini belirtirken; ilke, ne yapılacağını belirtir.
  - ilkenin mekanizmadan ayrılması çok önemlidir, bunun uygulanması ileride ilke değiştirilmek istendiğinde bize çok büyük esneklik sağlar.
- Bir işletim sisteminin belirlenmesi ve tasarlanması, yazılım mühendisliğinin son derece önemli bir görevidir.

Gerçekleştirme/Uygulanması

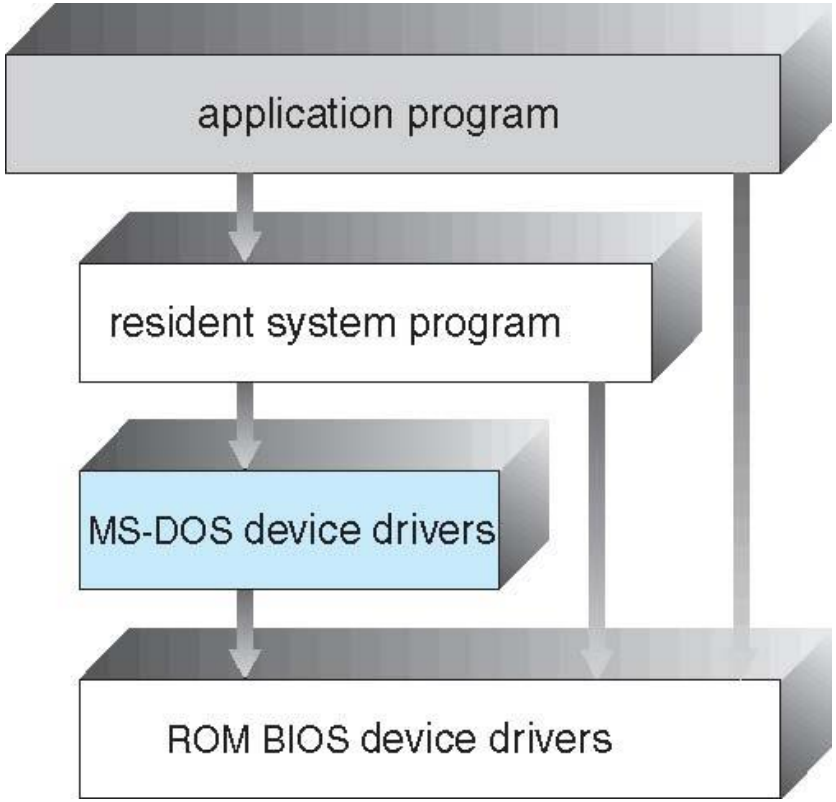
- Çok değişkendir
  - İlk İşletim Sistemleri Assembly dilinde idi
  - Sonrasında sistem prog dilleri ile Algol, PL/1 gibi
  - Şimdi C, C++
- Aslında birçok dilin karışımı
  - En düşük seviye Assembly ile
  - Ana gövde C ile
  - Sistem programları C, C++, script dilleri PERL, Python
- Daha yüksek seviyeli diller diğer donanımlara bağlanmak daha kolay olabilir ama daha yavaş
- Emülatörler (Benzetici) yabancı bir donanım üzerinde çalışmasına olanak sağlar.

İşletim Sistemi Yapısı

- Genel amaçlı İşletim Sistemi çok geniştir
- Çeşitli yollarla yapılandırılabilir
  - Basit yapı-MS-DOS
  - Daha kompleks yapılar – UNIX
  - Katmanlı yapı – Soyutlama
  - Mikro çekirdek -

Basit Yapı

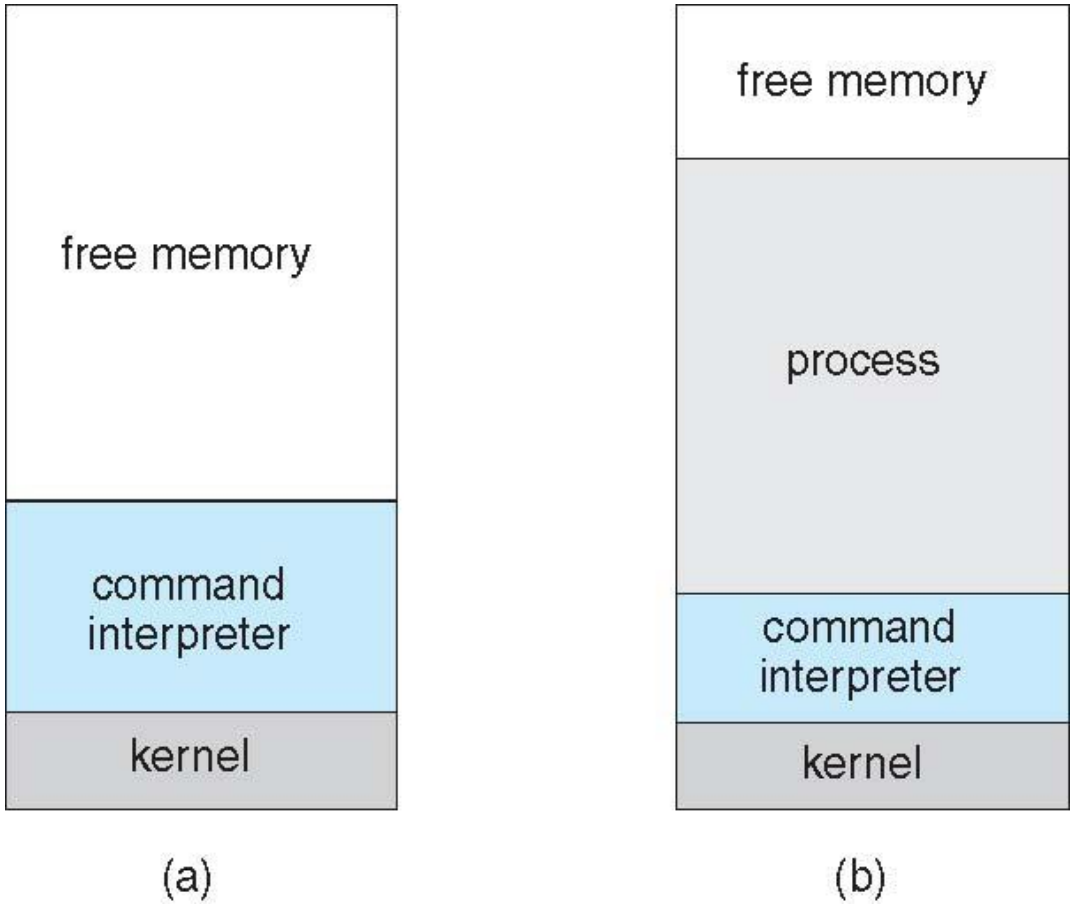
- MS-DOS (**M**icro**S**oft **D**isk **O**perating **S**ystem) – En az bellek kullanıp, maksimum fonksiyonelliği sağlamak üzere yazılmıştır.
  - Modüler değildir.
  - Bazı yapılara sahip olmasına karşın arayüzleri ve fonksiyonları düzgün bir şekilde gruplandırılmamış.



Örnek: MS-DOS

- Aynı anda tek bir işlem yapabilir (Single-tasking)
- Sistem başlatılırken shell çağırılır.
- Programı başlatmak için basit bir yöntem kullanılır.
  - Proses oluşturulmaz.
- Bellek tek bölümden oluşur.
- Programları hafızaya yükler
- Programı kapat -> shell yeniden yüklenir

MS-DOS Yürütme



(a) Sistem ilk başlatıldığında

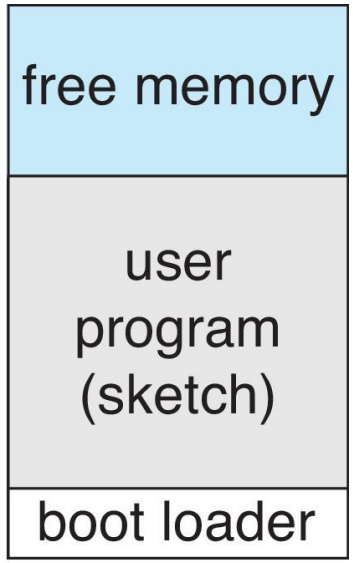
(b) Çalışmakta olan program

Example: Arduino

- Aynı anda tek bir işlem yapabilir (Single-tasking)
- İşletim sistemi yok
- USB üzerinden yüklenen programlar (sketch) flash belleğe yazılır
- Tek bellek alanı
- Önyükleyici programı yükler
- Program çıkışı -> kabuk yeniden yüklenir

(a)

At system startup



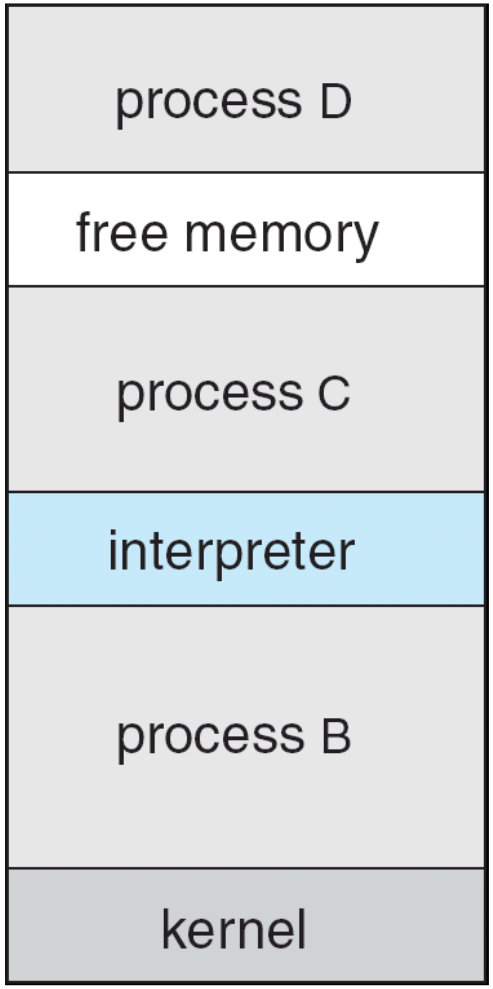
(b)

running a program

Örnek: FreeBSD

- FreeBSD (Free Berkeley Software Distribution) x86 Uyumlu, AMD64, IA-64, PC-98 ve UltraSPARC® mimarileri için bir işletim sistemidir. Berkeley'deki Kaliforniya Üniversitesi'nde geliştirilmiş UNIX® türevi olan BSD'yi temel almıştır.
- Unix türevi
- Aynı anda birden fazla işlem yapabilme (Multitasking)
- Kullanıcı girişi -> kabuk başlatılır
- Kabuk (Shell) prosesi oluştururken fork() sistem çağrısını çalıştırır
  - Proses içine program yüklenirken exec() sistem çağrısını çalıştırır.
  - Kabuk, prosesin kullanıcı komutlarıyla sonlanmasını yada devam etmesinin bekler
- Prosesler 0 koduyla(hatasız) veya > 0 koduyla (hata kodu) sonlanır
- FreeBSD Türkiye'de çoğunlukla Metin2, Silkroad gibi birçok PvP oyun sunucusunda tercih edilir.

FreeBSD Üzerinde Birden Fazla İşlem Çalışırken

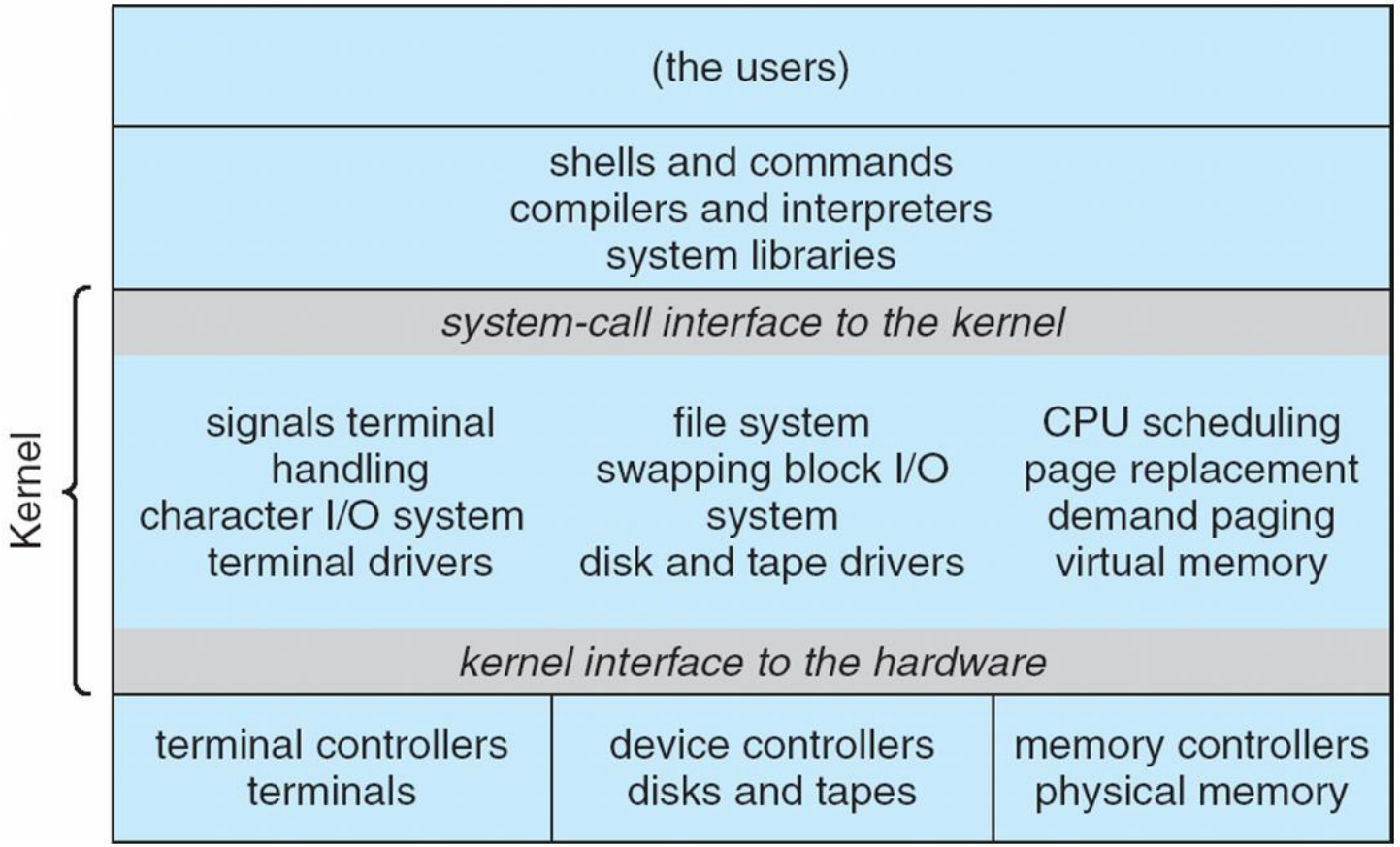


Monolithic Structure – Original UNIX

- Tekparça (Monolithic)
- **UNIX**, 1969 yılında,Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk ve Joe Ossanna tarafından Bell Laboratuvarları'nda geliştirilmiş, çok kullanıcılı (multiuser), çok görevli yapıyı destekleyen (multitasking) bir bilgisayar işletim sistemidir.
- 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.

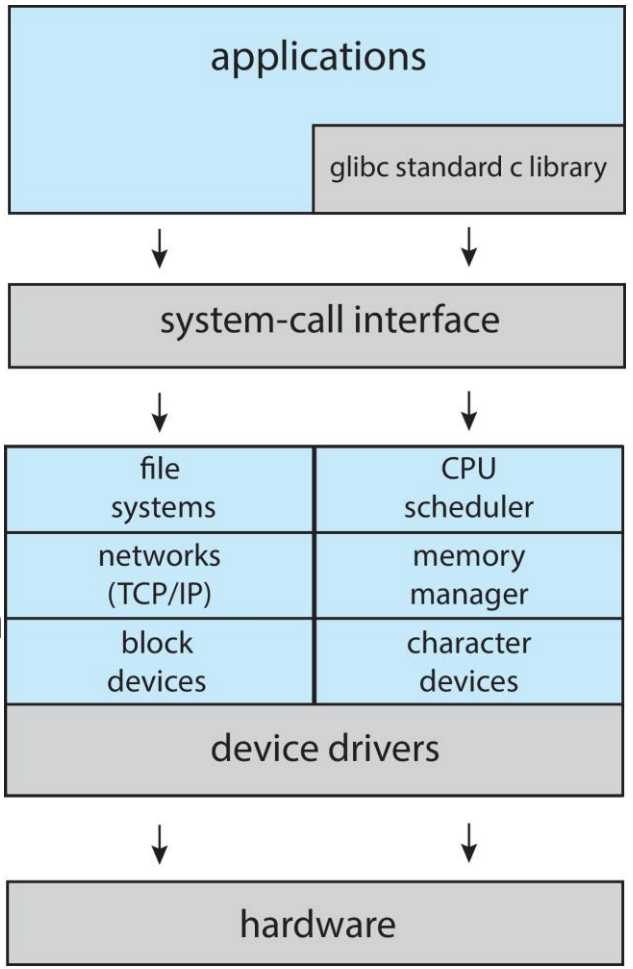
Geleneksel UNIX Sistem Yapısı

Basit ama tamamen katmanlı değil



Linux System Structure

- Monolitik artı modüler tasarım
- **Linux** bilgisayar işletim sistemlerinin en temel parçası olan çekirdek yazılımlarından bir tanesidir.
- GNU Genel Kamu Lisansı ile sunulan ve Linux Vakfı çatısı altında geliştirilen bir özgür yazılım projesidir.
- Linux ismi ilk geliştiricisi olan Linus Torvalds tarafından 1991 yılında verilmiştir.
- Günümüzde süper bilgisayarlarda, akıllı cihazların ve internet altyapısında kullanılan cihazların işletim sistemlerinde yaygın olarak kullanılmaktadır.
- Bunlardan en popüler olanı Google tarafından geliştirilen Android işletim sistemidir.

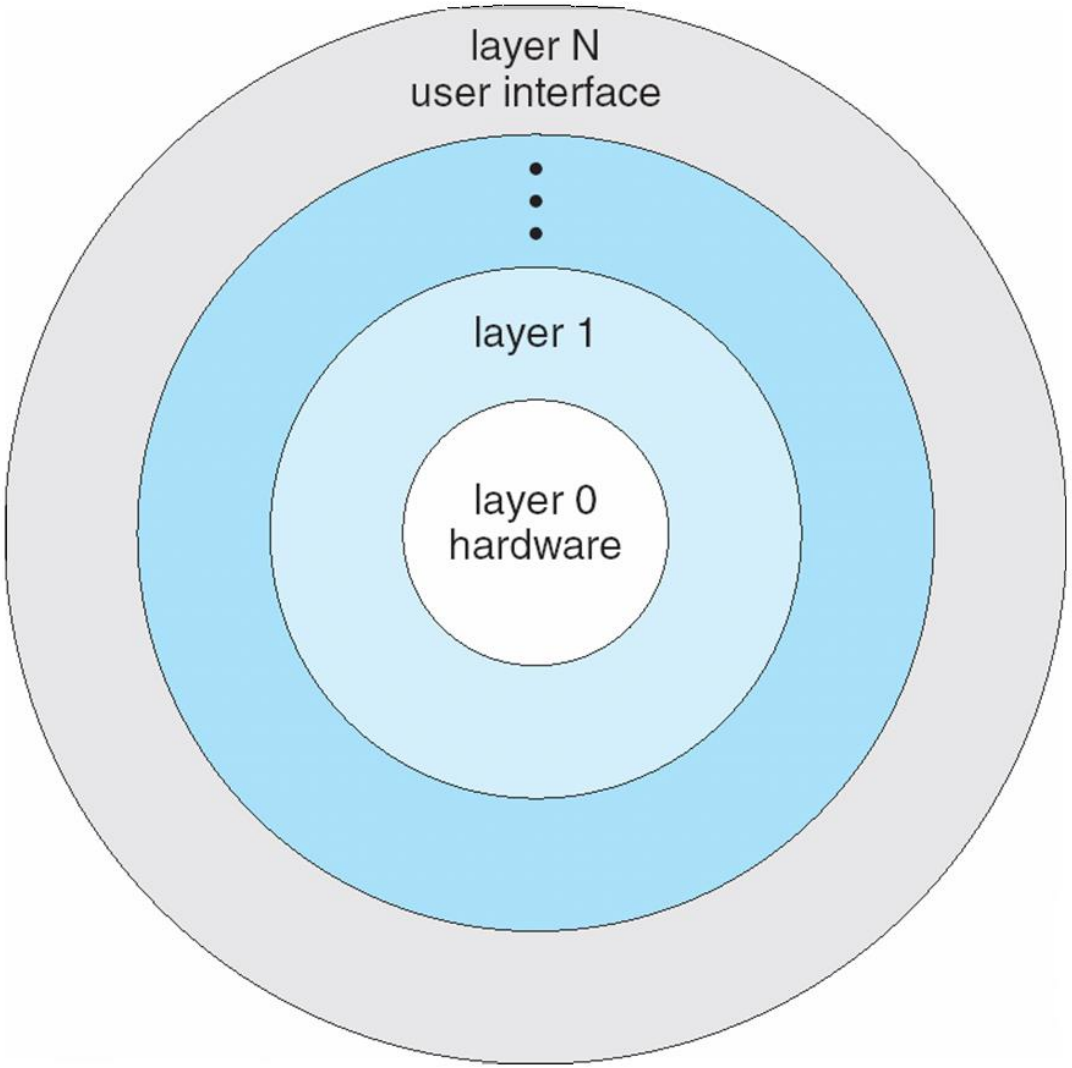


Katmanlı Yaklaşım

- 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.

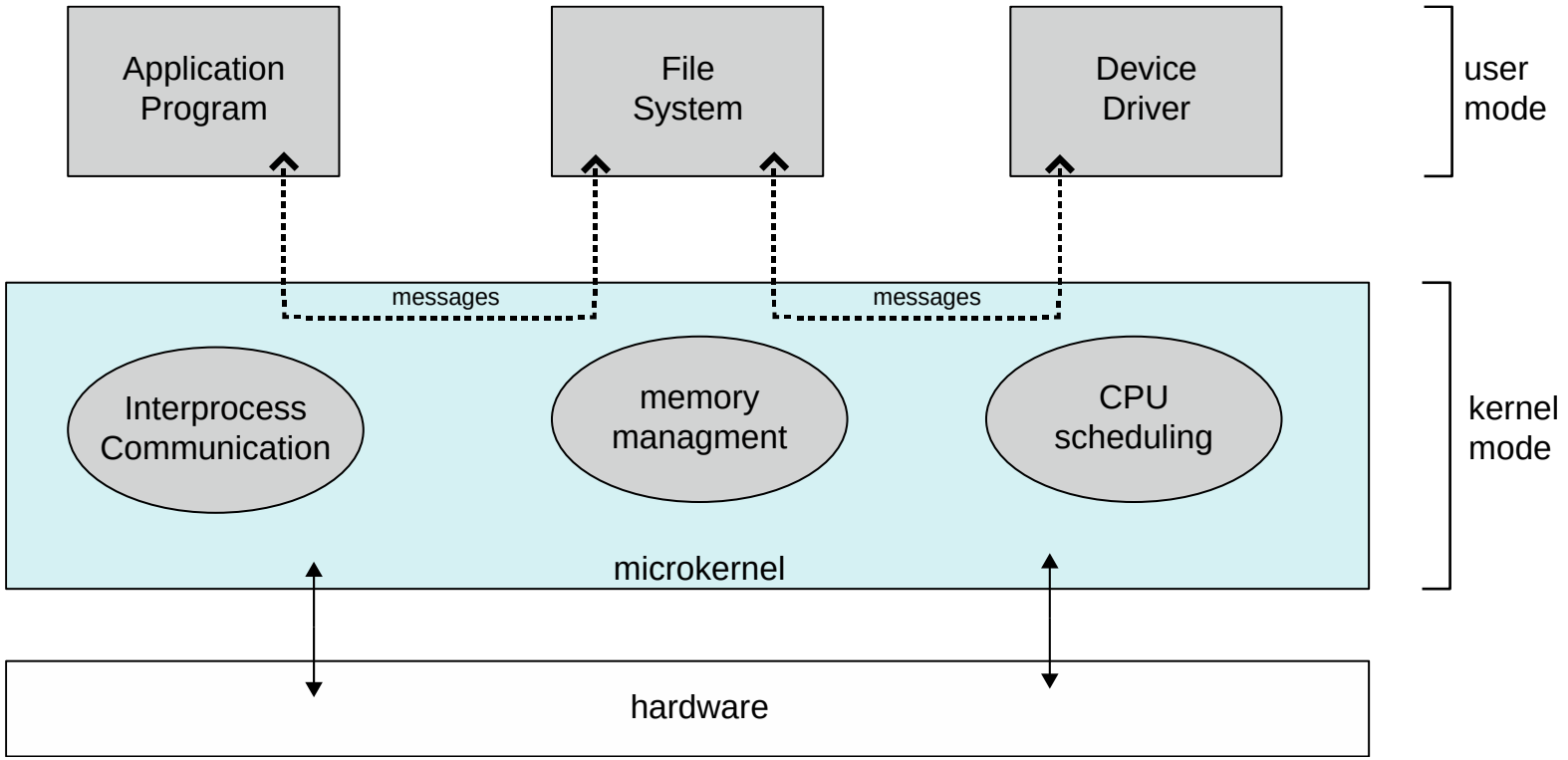
Katmanlı İşletim Sistemi



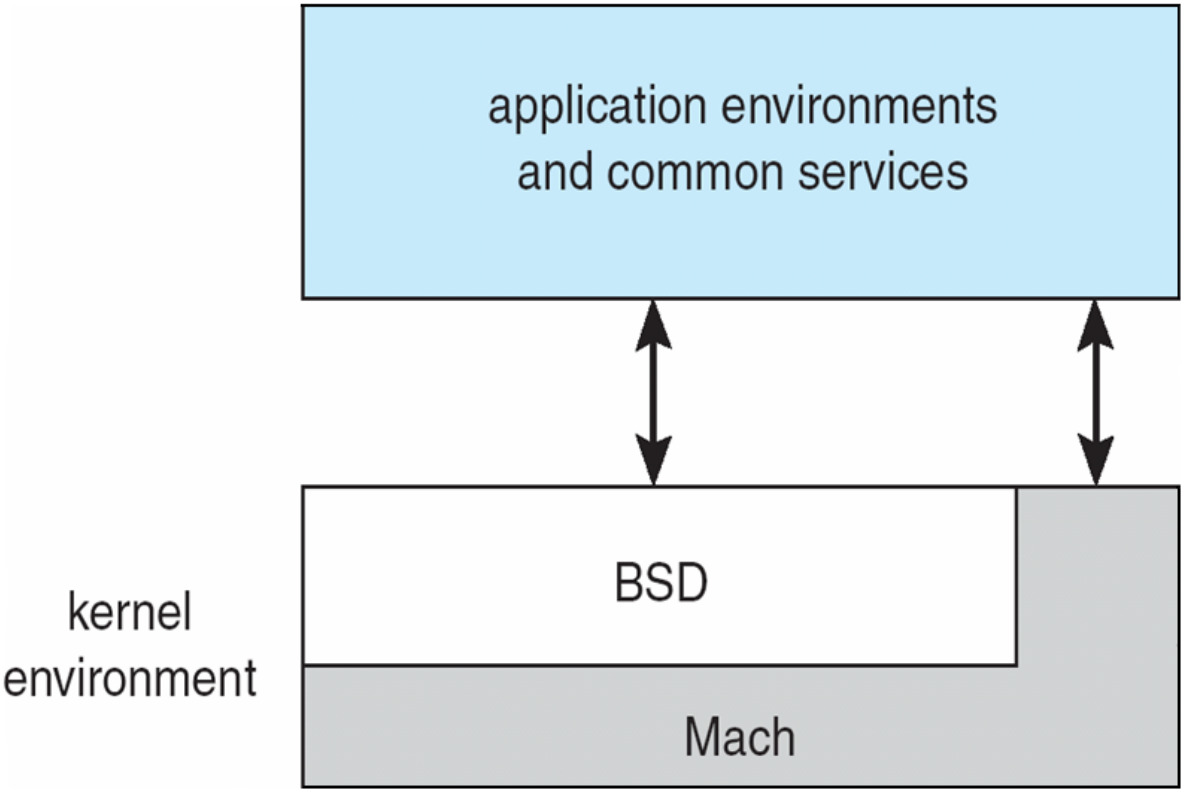
Mikro Çekirdek Sistem Yapısı

- Kernel'deki 'fazlalıkları' olabildiğince kullanıcı (*user*) alanına taşır.
- Kullanıcı modülleri arasında iletişim, mesaj geçişi yöntemi ile sağlanır.
- Mach bir mikro çekirdek örneği
- Mac OS X çekirdeği (Darwin) kısmen Mach'a dayanıyor
- Artıları:
  - Mikro çekirdeği 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

Microkernel System Structure



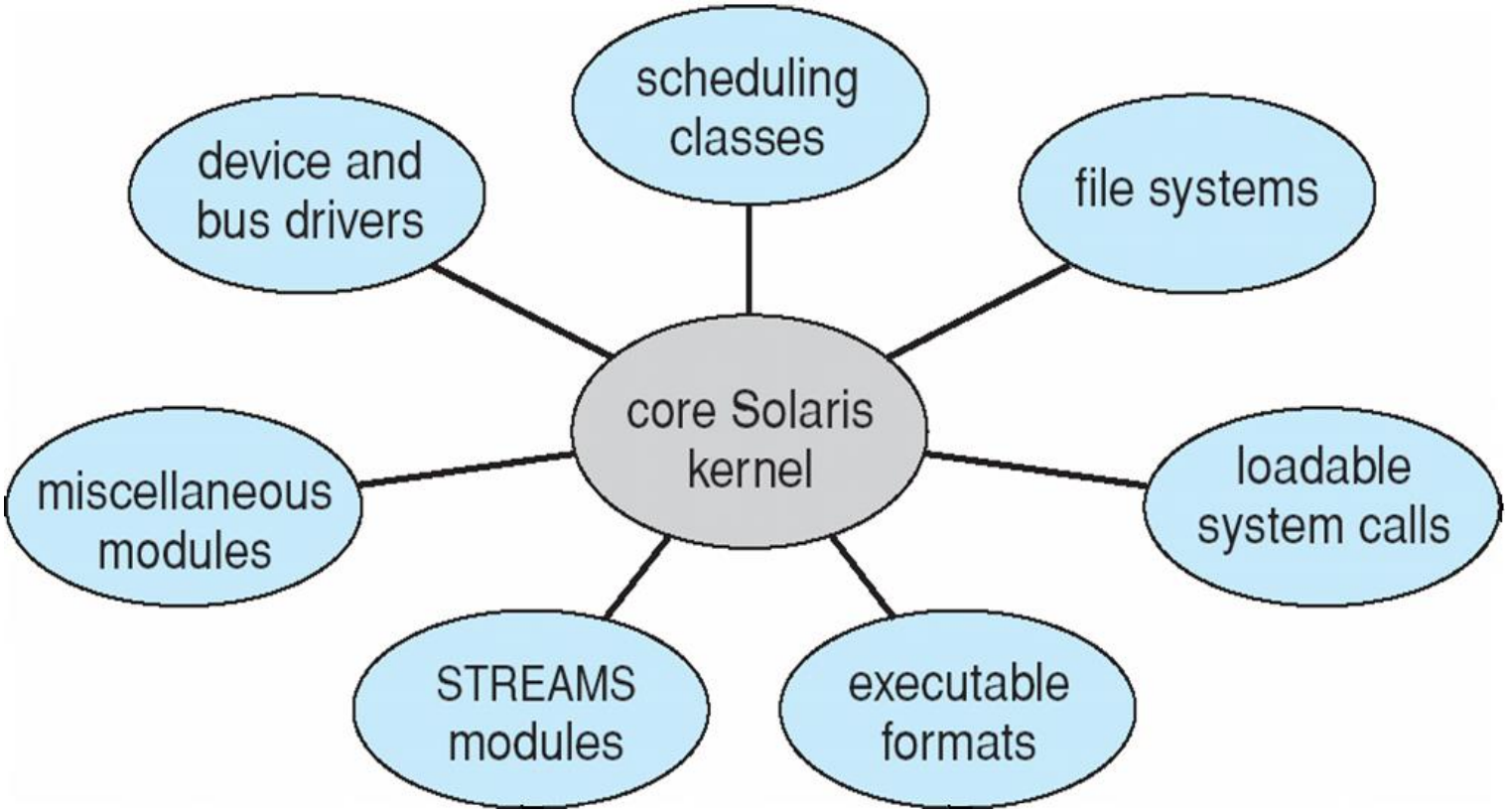
Mac OS X Yapısı



Modüler Yapı

- Günümüz işletim sistemlerinin çoğunda yüklenebilir kernel modülleri uygulamaktadır.
  - Nesne yönelimli yaklaşım hakimdir.
  - Her bir çekirdek bileşeni bağımsızdır.
  - Her modül birbiriyle bilinen arayüzler üzerinden konuşur.
  - İhtiyaç duyulması halinde herbir modül yüklenebilir.
- Genel olarak, katmanlı yapıya benzer fakat çok daha esnektir.
  - Linux, Solaris, etc

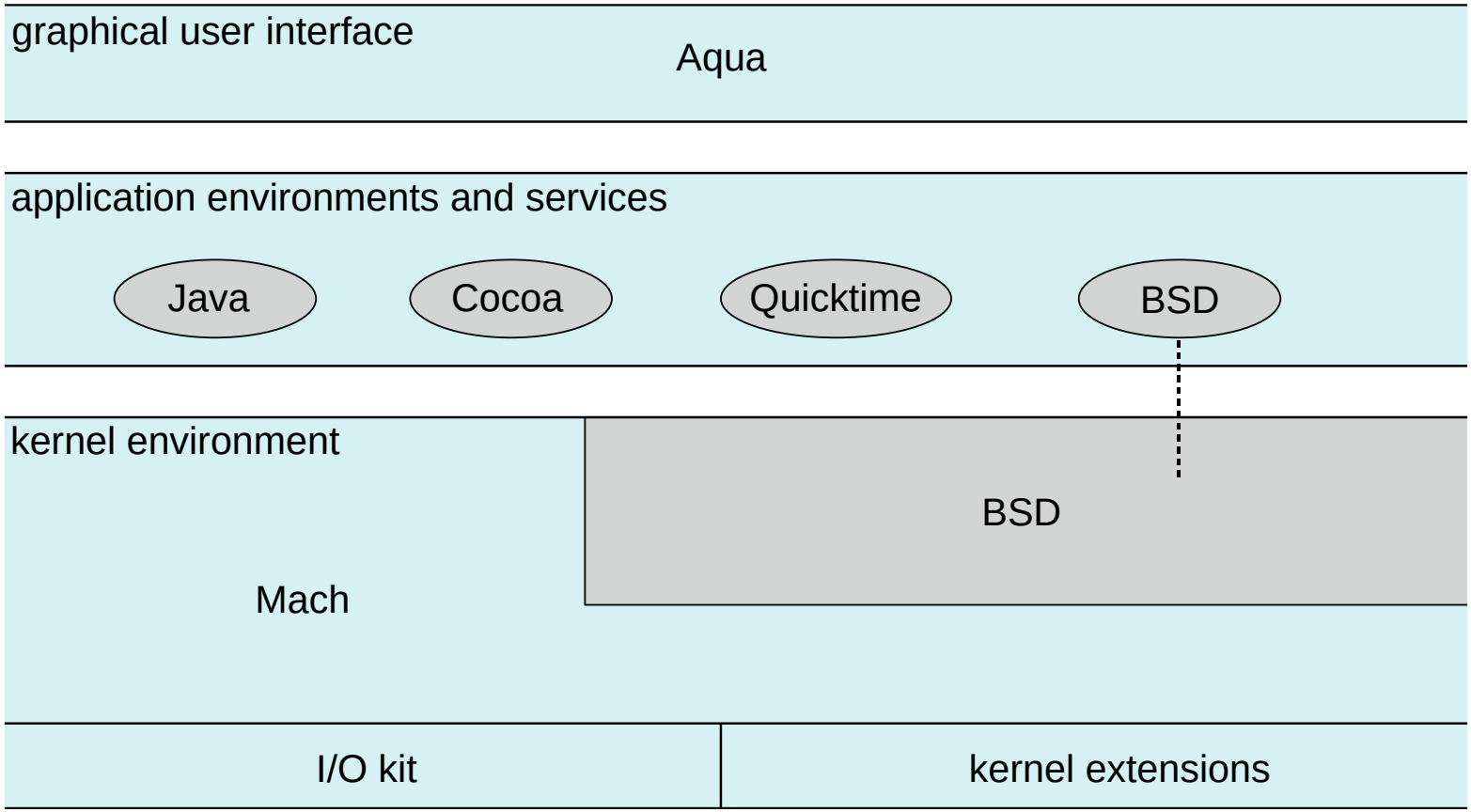
Solaris Modüler Yaklaşımı



Hibrit Sistemler

- Modern işletim sistemler tek bir saf modül değildir
  - Performans, güvenlik ve kullanılabilirlik gibi özellikler için çeşitli yaklaşımlar birlikte kullanılır
  - Linux ve Solaris çekirdekleri tek parçadır fakat eklenebilir diğer modüller ile hibrit bir yapıya dönüşür
  - Windows genelde tek parça, sadece farklı alt-sistemlerde kişiselleştirilebilir
- Apple Mac OS X hibrit, katmanlı Aqua kullanıcı arayüzü ile Cocoa programlama ortamı

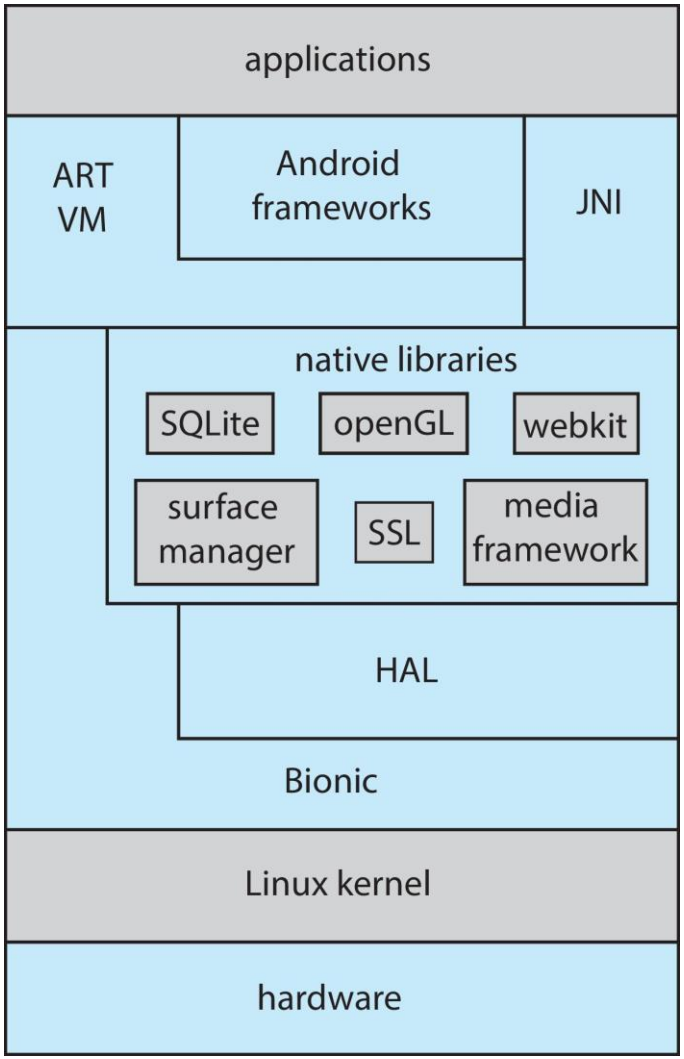
Mac OS X yapısı



Android

- Open Handset Alliance tarafından geliştirildi (çoğunlukla Google)
  - Açık kaynak
- Linux çekirdeği temel alındı ancak modifiye edildi
  - Proses, bellek, aygıt sürücüsü yönetimi sağlar
  - Güç yönetimi ek
- Runtime environment (Çalışma zamanı ortamı), çekirdek kütüphaneleri ve Dalvik sanal makinesini içerir.
  - Uygulamalar Java plus Android API de geliştirilir
- Kütüphaneler web tarayıcısı (webkit), veritabanı (SQLite), multimedyalar için çerçeveler içerir

Android Architecture



İşletim Sistemini Oluşturma ve Önyükleme

- işletim sistemleri, Genel olarak çeşitli çevre birimlerine sahip bir sistem sınıfı üzerinde çalışacak şekilde tasarlanmıştır
- Yaygın olarak, işletim sistemi zaten satın alınan bilgisayarda yüklü gelir
  - Ancak diğer bazı işletim sistemlerini kurulabilir ve yüklenebilir
  - Bir işletim sistemini sıfırdan oluşturuyorsanız
    - İşletim sistemi kaynak kodunu yaz
    - Çalıştırılacağı sistem için işletim sistemini yapılandırın.
    - İşletim sistemini derleyin
    - İşletim sistemini yükle
    - Bilgisayarı ve yeni işletim sistemini önyüklemesini yapın

Building and Booting Linux

- Linux kaynak kodunu indirin (http://www.kernel.org)
- Çekirdeği “make menuconfig” ile yapılandırın
- “Make” kullanarak çekirdeği derleyin
  - `vmlinuz`, çekirdek imajı üretir
  - Çekirdek modüllerini “`make modules`” ile derleyin
  - Çekirdek modüllerini “`make modules_install`” aracılığıyla `vmlinuz'a` kurun.
  - “`make install`” ile sisteme yeni çekirdeği yükleyin

İşletim Sistemi Üretimi

- 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 (Önyükleme) program* – ROM bellekte saklanan bir kod parçasıdır ki bu kod parçası kernelin yerini belirleyip, belleğe yükler ve çalıştırır.

Sistem Önyükleme

- Donanımın başlatabilmesi için işletim sistemi donanıma uygun hale getirilmelidir
  - **bootstrap loader**(küçük bir kod parçası), kernelin yerini belirler, 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.
  - **Unified Extensible Firmware Interface** (**UEFI**) bir bilgisayarın donanım yazılımını (firmware) işletim sistemine (OS) bağlayan bir yazılım programı için bir belirtimdir. UEFI'nın sonunda BIOS'un yerini alması bekleniyor.
  - Çekirdek yüklenir ve sistem çalışır

İşletim sisteminde Hata Ayıklama

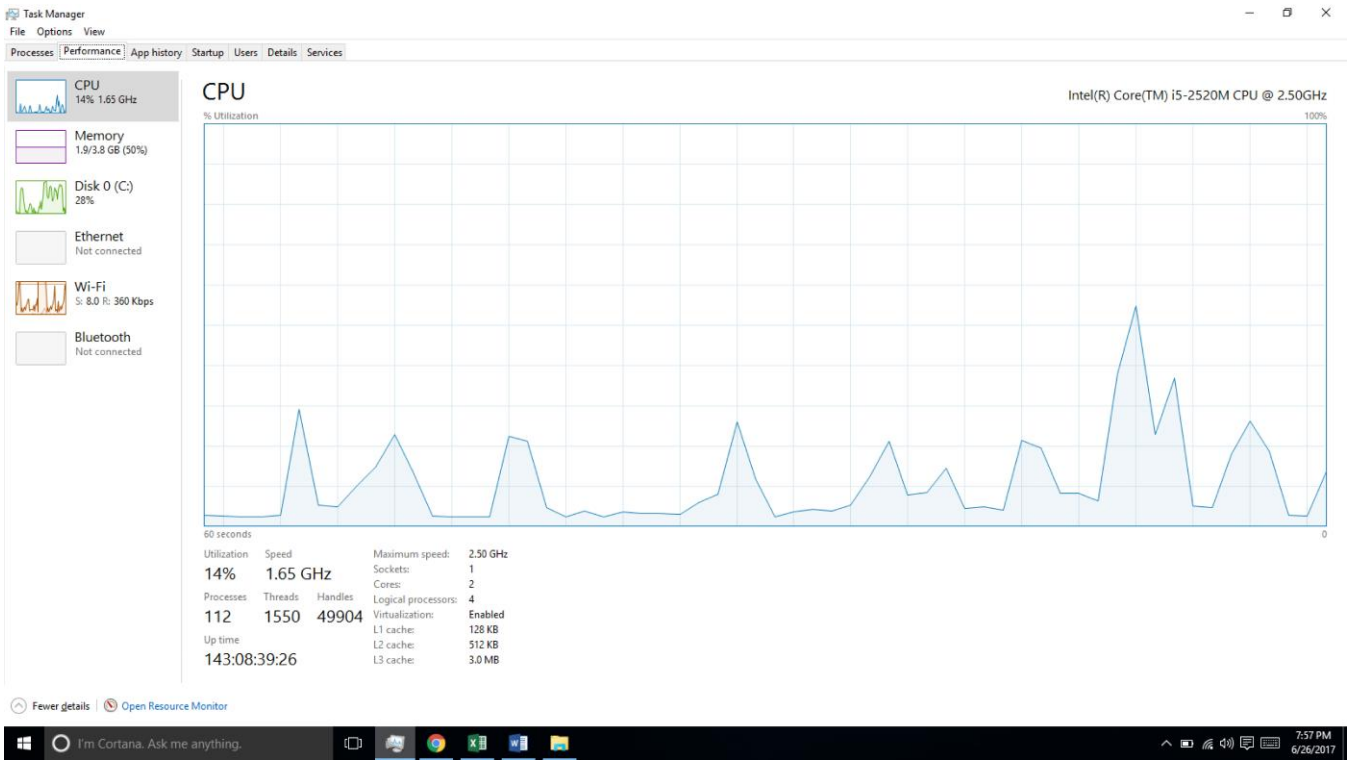
- **Hata yakalama - Debugging** hataları ve arıza (**bug**) bulma ve ayıklama işlemidir.
- İşletim sistemleri hata bilgilerini içeren **log dosyaları** oluşturur.
- Bir uygulamanın hatası sonucunda, prosesin bellek görüntüsünü tutan bir dosya oluşturabilir (**core dump** file).
- İşletim Sistemi hatasında, çekirdeğin bellek görüntüsünü taşıyan bir dosya üretebilir (**crash dump** file)
- Çöküşlerden sonra, sistem performansı optimize edilir.
  - Aktiviteler analiz için takip tablosunda tutulur (Trace listing)
  - Profillerde, istatistiksel eğilimleri tespit etmek için periyodik olarak komutların örnekleri alınır
- 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
```

Performans Ayarı

- Darboğazları kaldırarak performansı artırın
- İşletim sistemi, bilgi işlem araçlarını sağlamalı ve sistem davranışı ölçütlerini göstermelidir.
- Örneğin, "Top/Üst" programlar veya Windows Görev Yöneticisi



Tracing

- Belirli bir olaylar için veri toplar (Sistem çağrısı çağrılmasındaki adımlar gibi)
- Tools include
  - strace - bir proses tarafından çağrılan sistemi çağrılarını izler
  - gdb - kaynak düzeyinde hata ayıklayıcısı
  - perf - Linux performans araçları
  - tcpdump - ağ paketlerini toplar

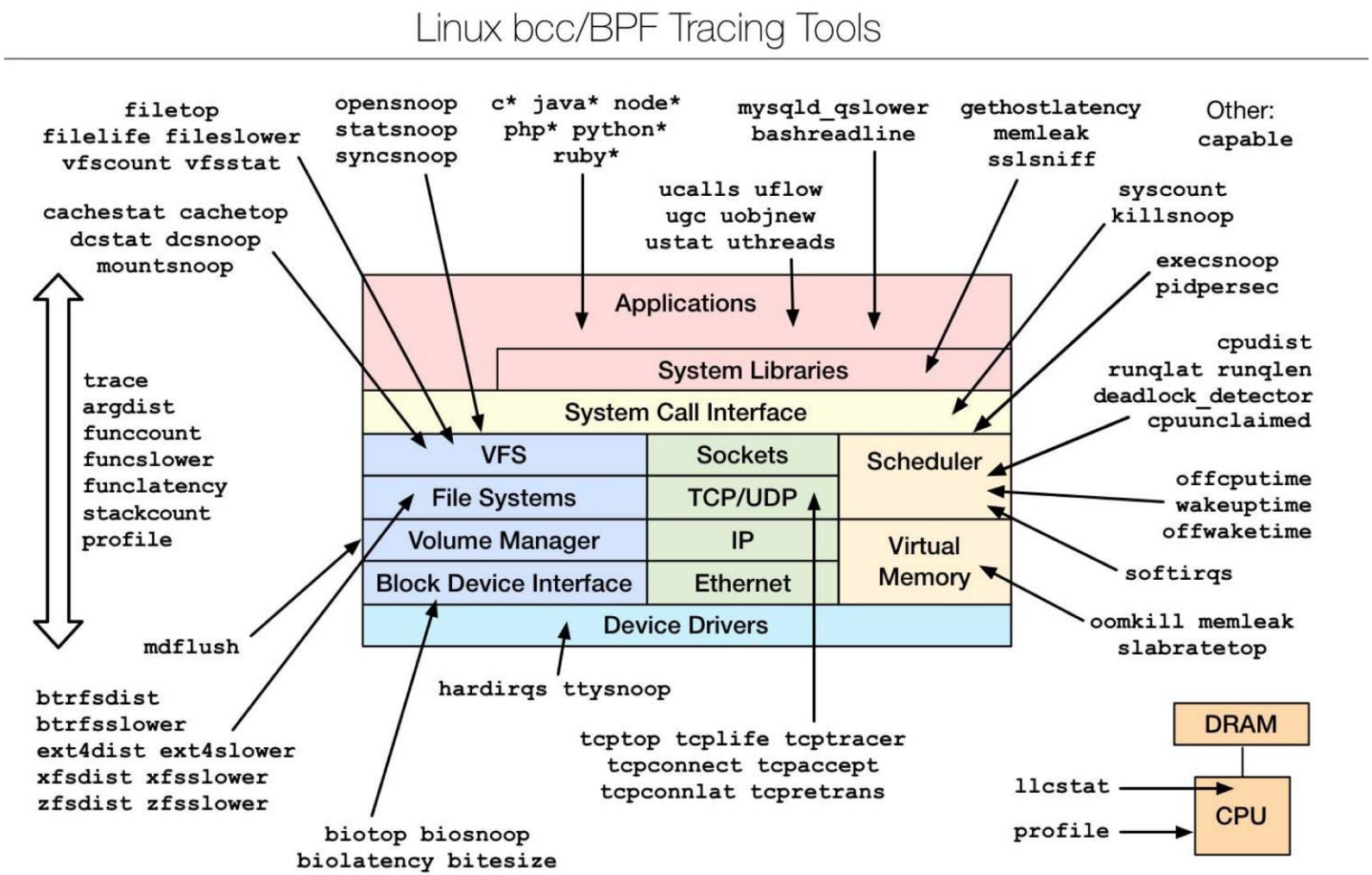
BCC

- Hem kullanıcı hem de bir sistemin eylemlerini anlayan bir araç seti olmadan kullanıcı düzeyinde ve çekirdek kodu arasındaki etkileşimler/hataların ayıklanması neredeyse imkansızdır
- BCC (BPF Derleyici Koleksiyonu) Linux için izleme özellikleri sağlayan zengin bir araç takımıdır.
  - Ayrıca orijinal DTrace ‘ e bakın
- Örneğin, disksnoop.py disk G / Ç etkinliğini izler.

```
TIME(s)            T      BYTES      LAT(ms)
1946.29186700      R      8              0.27
1946.33965000      R      8              0.26
1948.34585000      W      8192           0.96
1950.43251000      R      4096           0.56
1951.74121000      R      4096           0.35
```

- Diğer birçok araç (sonraki slayt)

Linux bcc/BPF Tracing Tools



Bölüm 2 Sonu