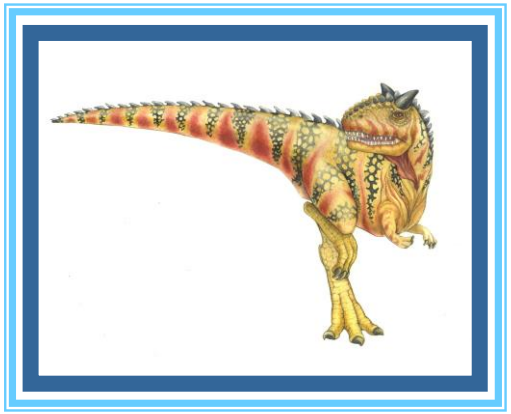
Bölüm 2b: İşletim Sistemi Tasarım ve Uygulaması



Doç.Dr. Ahmet ZENGİN

İçindekiler

- Tasarım ve Uygulama
- İşletim sistemi Yapısı
- Bir OS Derleme ve Başlatma
- OS Hata Ayıklama

Amaçlar

- İşletim sistemleri tasarlamak için monolitik, katmanlı, mikro çekirdekli, modüler ve hibrit stratejileri karşılaştırma ve ayırtedici yönlerini bulmak
- Bir işletim sistemi önyükleme işlemini görmek
- İşletim sistemi performansını izlemek için araçlar uygulamak
- Linux çekirdeği ile etkileşim için çekirdek modülleri tasarlama ve uygulamak

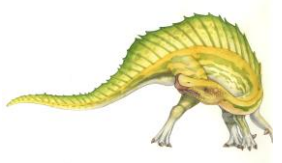
Tasarım ve Uygulama

- 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.
- OS nin şartnamesi ve tasarımı **yazılım mühendisliğinin** en zor ve en çok çaba gerektiren bir işidir.

Politika ve Mekanizma

Şu iki kavramın birbirinden ayrılması çok önemlidir.

- **Politika:** **Ne** yapılacak?
  - Örmek: her 100 sm sonra kesme
- **Mekanizma**: **Nasıl** yapılacak?
  - Örnek: zamanlayıcı
- 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.
    - Örnek: 100'ü 200 yaparken



Uygulama

- Çok varyasyon
  - Erken dönem işletim sistemleri Assembly dilinde
  - Ardından Algol, PL/1 gibi sistem programlama dilleri
  - Şimdi C, C++
- Aslında, genellikle dillerin bir karışımı
  - En düşük seviyelerde assembly
  - C'de ana gövde
  - C, C++'da sistem programları, PERL, Python gibi script dilleri, kabuk scriptleri
- Diğer donanımlara taşınması daha kolay daha yüksek seviyeli dil
  - Ama daha yavaş
- **Emülasyon** bir işletim sisteminin yerel olmayan donanımlarda çalışmasına izin verir

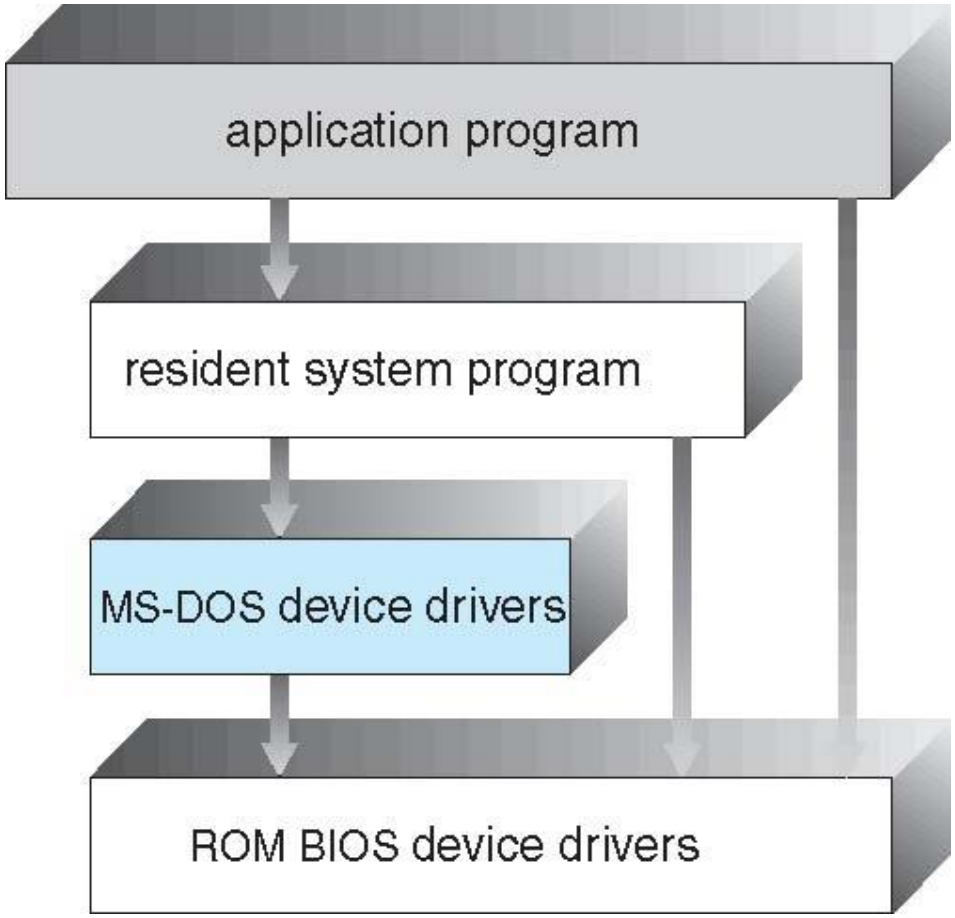
İşletim Sistemi Yapıları

- Genel amaçlı bir OS çok büyük bir program
- 4 ana gruba ayrılır:
  - Basit yapılı– MS-DOS
  - Daha karmaşık– UNIX
  - Katmanlı– bir soyutlama şekli
  - Mikro çekirdek– Mach

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

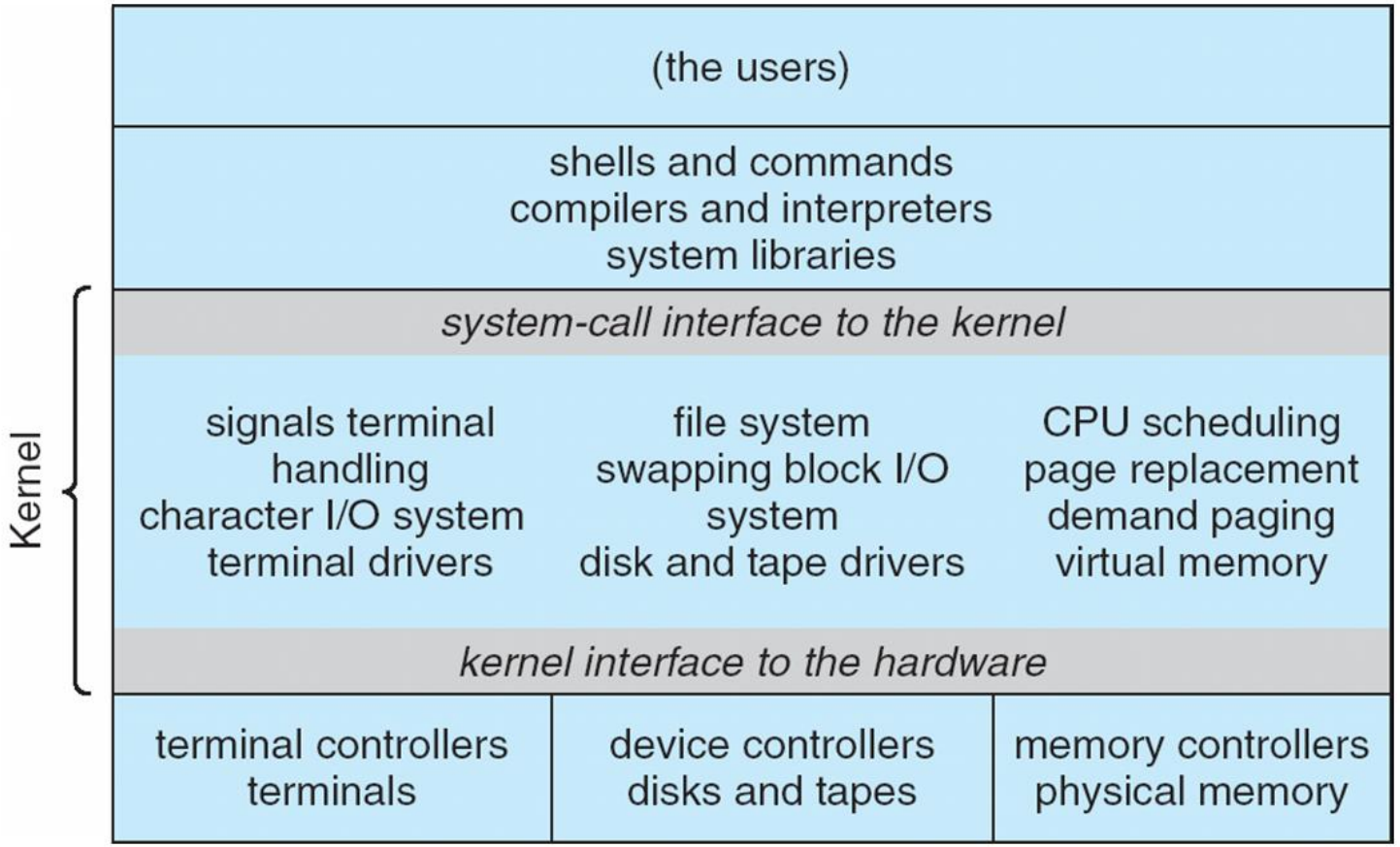


Monolitik Yapı – Orginal UNIX

- UNIX – donanım işlevselliği ile sınırlı, orijinal UNIX işletim sistemi sınırlı bir yapıya sahiptir.
- UNIX OS, ayrılabilir iki bölümden oluşur
  - Sistem programları
  - Çekirdek
    - Sistem çağrısı arayüzünün altındaki ve fiziksel donanımın üzerindeki her şey
    - Dosya sistemi, CPU sıralama, bellek yönetimi ve diğer işletim sistemi işlevlerini sağlar; bir seviye için çok sayıda fonksiyon

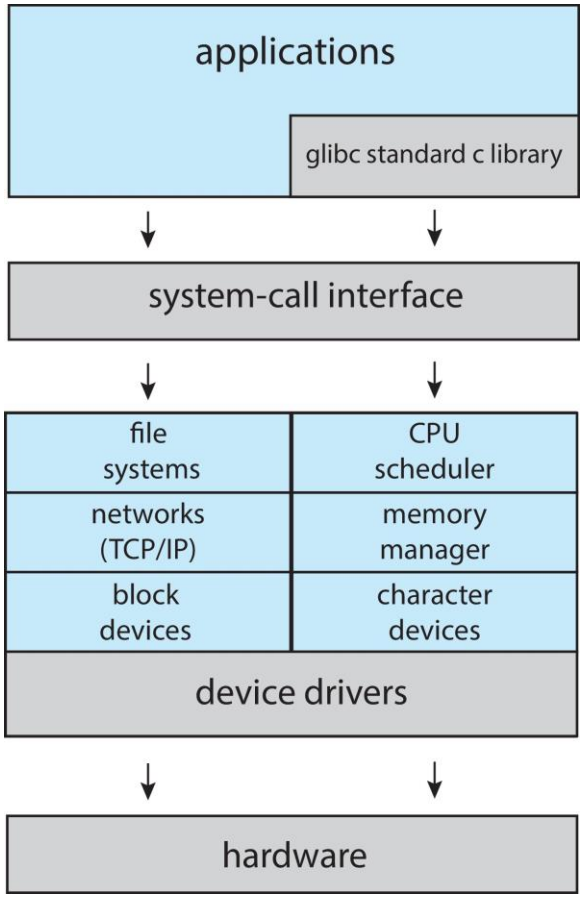
Geleneksel UNIX Sistem Yapısı

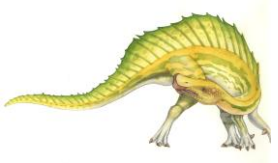
Basit ötesi, ama tam manasıyla katmanlı değil



Linux Sistem Yapısı

Monolitik artı modüler tasarım

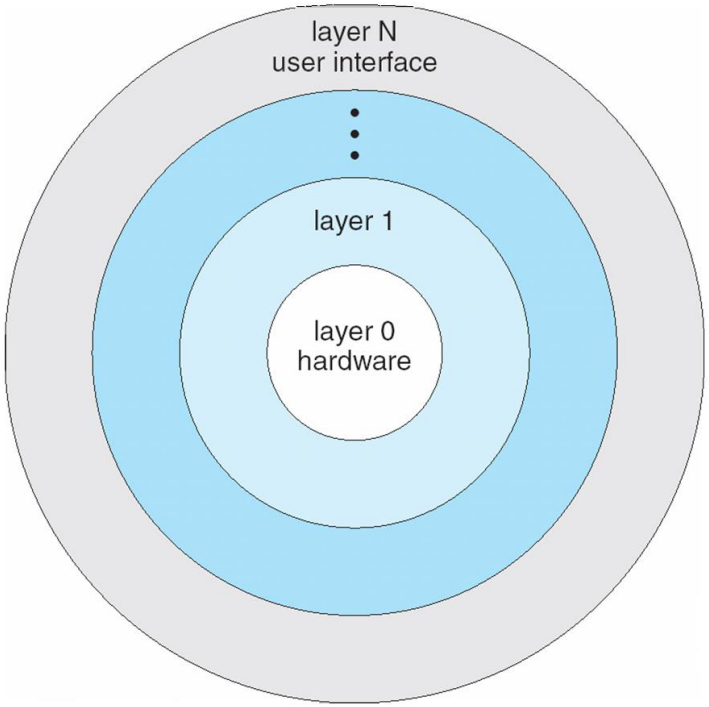




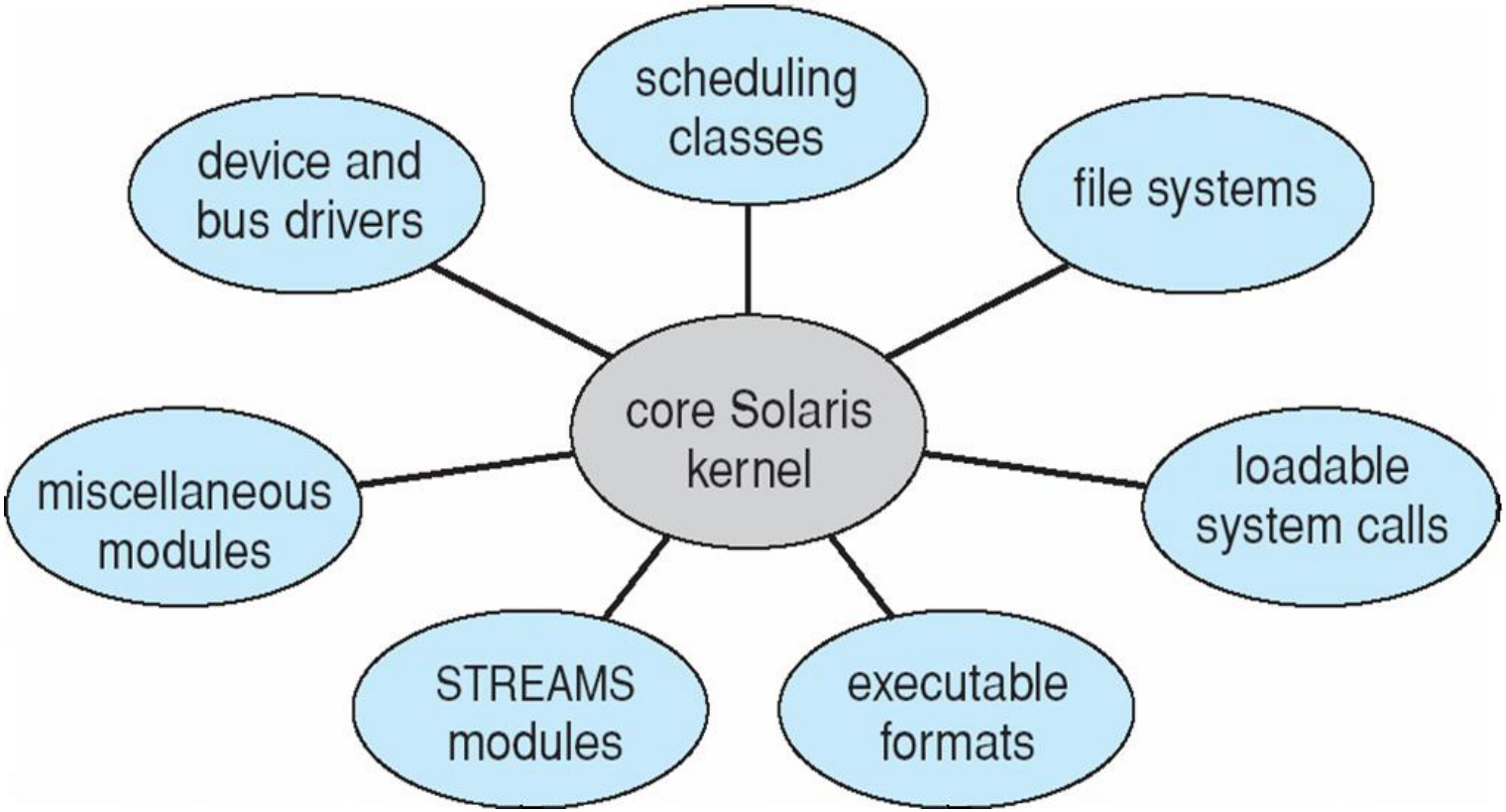
Katmanlı Yaklaşım

- 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.



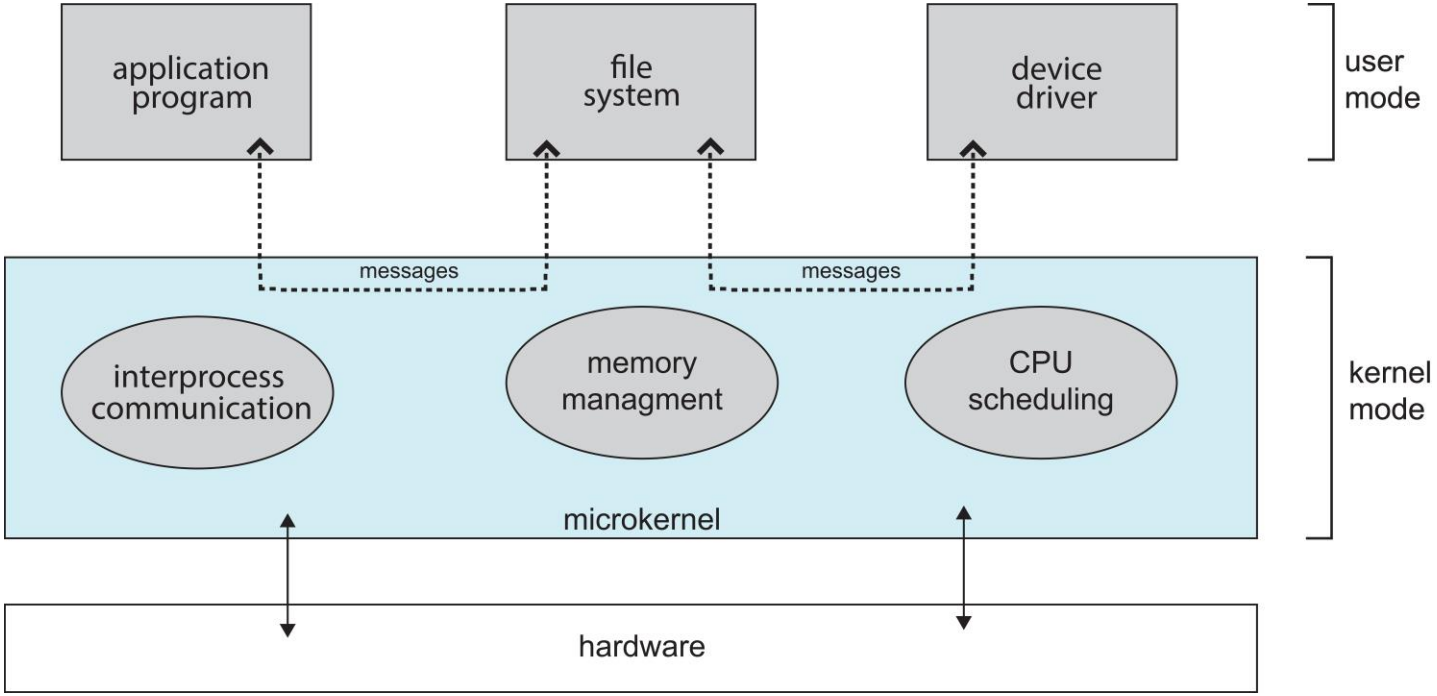
Solaris Modüler Yaklaşımı



Mikroçekirdek Sistem Yapısı

- Kernel'deki 'fazlalıkları' olabildiğince kullanıcı alanına taşır.
- **Mach** mikroçekirdek örneği
  - Mac OS X çekirdeği (**Darwin**) kısmen Mach'a dayanır
- Kullanıcı modülleri arasında iletişim, **mesaj iletimi** yöntemi ile sağlanır.
- Artıları:
  - Mikroçekirdeği genişletmek kolaydır.
  - İşletim sistemini yeni mimarilere taşıması kolaydır.
  - Çekirdek modda çalışan kod azaldığı için daha güvenilirdir.
  - Daha güvenlikli
- Eksileri:
  - Kullanıcı alanı ile çekirdek alanı arasındaki iletişimden kaynaklanan performans kaybı

Mikroçekirdek Sistem Yapısı



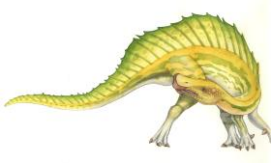
Modüler Yapı

- Günümüz işletim sistemleri yüklenebilir çekirdek modülleri kulllanmaktadır.
  - Nesne yönelimli yaklaşım hakimdir.
  - Her bir çekirdek bileşeni bağımsızdır.
  - Her modül birbiriyle bilinen arayüzler üzerinden konuşur.
  - Her biri çekirdek içerisine ihtiyaç duyulması halinde yüklenebilir.

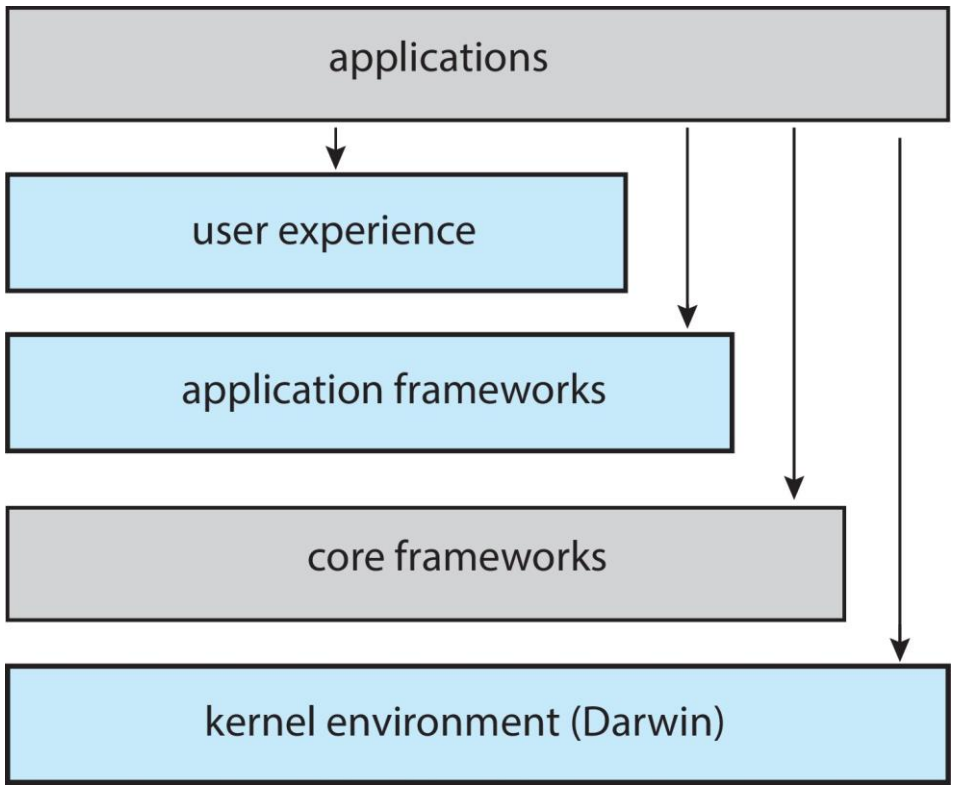
- Genel olarak, katmanlı yapıya benzer fakat çok daha esnektir.
  - Linux, Solaris, vb.

Hibrit Sistemler

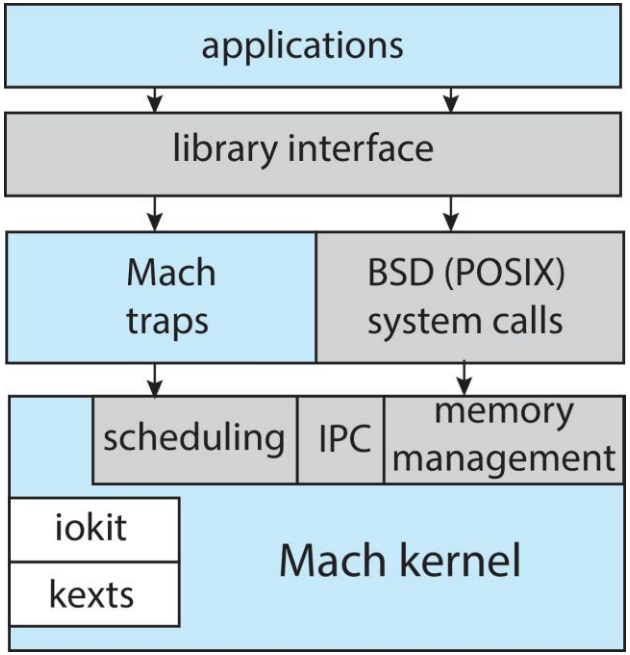
- Çoğu modern işletim sistemi saf bir model değildir.
  - Hibrit, performans, güvenlik ve kullanılabilirlik ihtiyaçlarını karşılamak için birden fazla yaklaşımı birleştirir
  - Çekirdek adres alanında Linux ve Solaris çekirdekleri,
    - monolitik, ayrıca dinamik yükleme işlevselliği ile modüler
  - Windows çoğunlukla monolitik, artı farklı alt sistem özellikleri mikroçekirdek olarak uygulanmış
- Apple Mac OS X hibrit, katmanlı, **Aqua** UI plus **Cocoa** programlama ortamı
  - Aşağıda, Mach mikroçekirdeği ve BSD Unix parçalarından oluşan çekirdek, ayrıca G/Ç kiti ve dinamik olarak yüklenebilir modüller (**çekirdek uzantıları** olarak adlandırılır) bulunmaktadır.



macOS ve iOS Yapısı



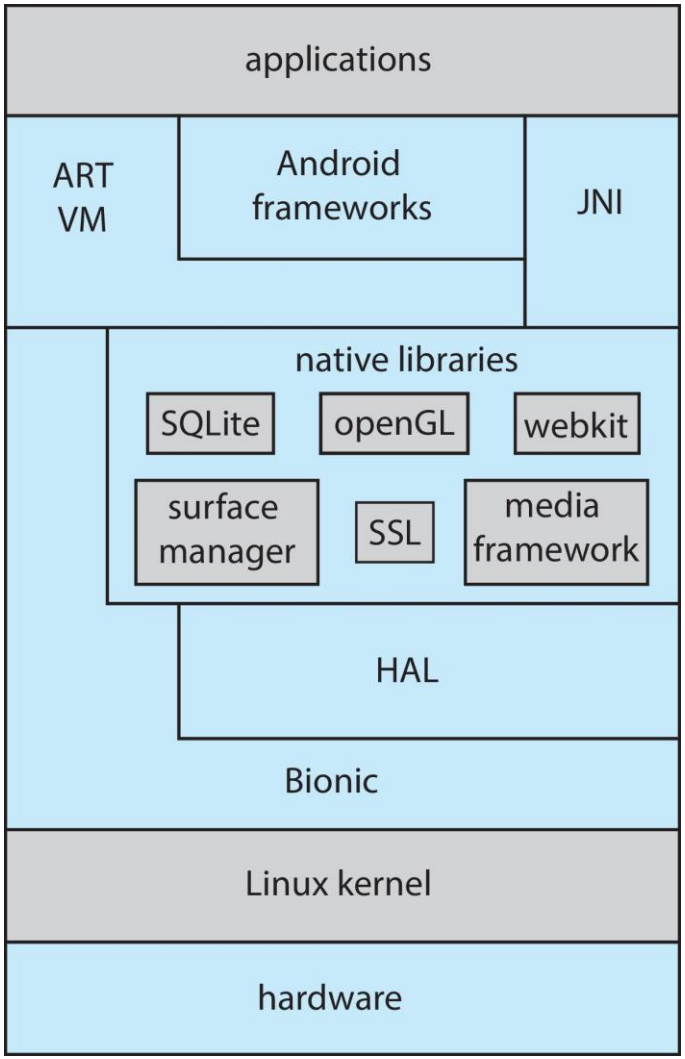
Darwin


applications
library interface
Mach traps
BSD (POSIX) system calls
scheduling
IPC
memory management
iokit
kexts
Mach kernel

Android

- Open Handset Alliance (çoğunlukla Google) tarafından geliştirildi
  - Açık kaynak
- IOS'a benzer
- Linux çekirdeğine dayalı ancak değiştirilmiş
  - Proses, bellek, aygıt sürücüsü yönetimi sağlar
  - Güç yönetimini ekler
- Çalışma zamanı ortamı, ana kütüphane kümesini ve Dalvik sanal makinesini içerir
  - Java artı Android API'sinde geliştirilen uygulamalar
    - Java bayt koduna derlenen Java sınıfı dosyaları yürütülebilir dosyaya çevrilir daha sonra Dalvik VM'de çalıştırılır
- Kütüphaneler, web tarayıcısı (webkit), veritabanı (SQLite), multimedya, daha küçük libc için çerçeveler içerir

Android Mimarisi

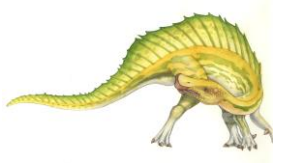


OS İnşası ve Boot Etme

- Genellikle çeşitli çevre birimlerine sahip bir sistem sınıfında çalışmak üzere tasarlanmış işletim sistemleri
- Genellikle satın alınan bilgisayarda yüklü olan işletim sistemi
  - Ancak diğer bazı işletim sistemlerini derleyip kurabilir
  - Sıfırdan bir işletim sistemi oluşturuyorsanız
    - İşletim sistemi kaynak kodunu yazın
    - Üzerinde çalışacağı sistem için işletim sistemini yapılandırın
    - İşletim sistemini derleyin
    - İşletim sistemini yükleyin
    - Bilgisayarı ve yeni işletim sistemini boot edin

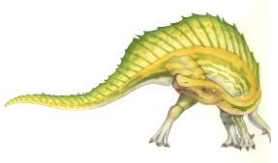
Linux'u Derleme ve Boot

- Linux kaynak kodunu indirin (http://www.kernel.org)
- Çekirdeği konfigüre edin "`make menuconfig`"
- Çekirdeği derleyin "`make`"
  - Çekirdek imajı `vmlinuz`,
  - Çekirdek modüllerini derleyin "`make modules`"
  - Çekirdek modüllerini "`make ile` kurun `vmlinuz modules_install`"
  - Sistem üzerinde yeni çekirdeği kurun "`make install`"



Sistem Boot

- Açma kapama düğmesine basıldığında, çalışma sabit bir bellek bölgesinde çaşlar
- OS donanım tarafından erişilebilmeli ve böylece donanım sistemi başlatabilmeli
  - Kodun küçük bir parçası – **bootstrap loader**, **BIOS**, **ROM** veya **EEPROM** da saklanır, çekirdeği konumlandrır, belleğe yükler ve başlatır
  - Bazen iki adımlı bir süreç,
  - Modern sistemler BIOS u **Unified Extensible Firmware Interface** (**UEFI**) ile değiştirir
- Ortak bootstrap yükleyicisi, **GRUB**, çekirdeği birden fazla diskten seçerek yükler
- Çekirdek yüklendiğinde sistem **çalışır/running**

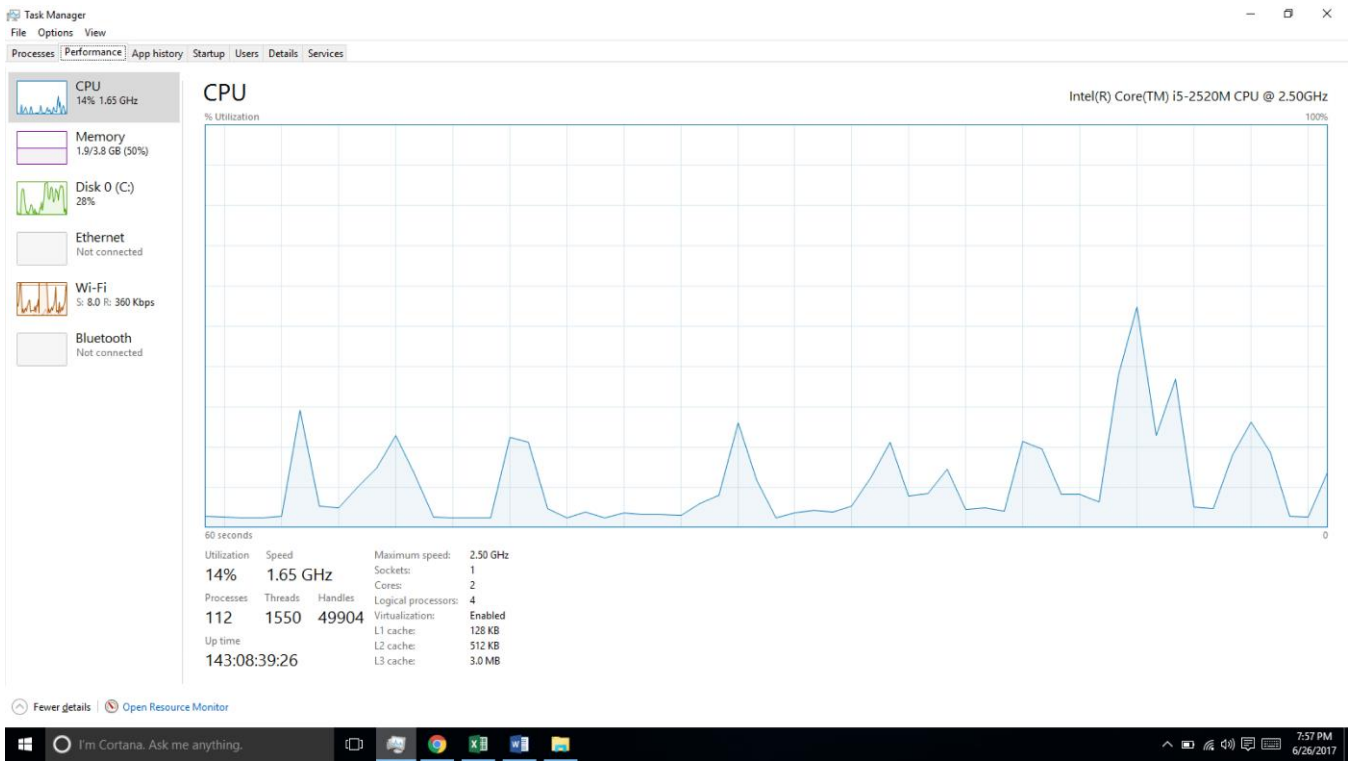


İşletim Sisteminde Hata Ayıklama

- **Hata ayıklama**, hataları veya **böcekleri-bugs** bulup düzeltmektir
- İşletim sistemi, hata bilgilerini içeren günlük dosyaları(**log files)** oluşturur
- Bir uygulamanın başarısızlığı, prosesin belleğini kaydeden **çekirdek döküm(core dump)** dosyası oluşturur.
- İşletim sistemi hatası, çekirdek belleğini içeren **kilitlenme dökümü(crash dump)** dosyası oluşturur

Kernighan Yasası: "Hata ayıklamak, ilk etapta kodu yazmaktan iki kat daha zordur. Bu nedenle, kodu olabildiğince akıllıca yazsanız bile, hata ayıklayacak kadar akıllı olmayabilirsiniz."

Performance Artırma



İzleme(Tracing)

- Bir istem çağrısı talebinin adımları gibi belirli bir olay için veri toplama,
- Araçlar
  - strace – bir proses tarafından çağrılan sistem çağrılarını izler
  - gdb – kaynak seviyeli hata ayıklayıcı
  - perf – Linux performans araçları
  - tcpdump – ağ paketlerini toplar

Bölüm 2b Sonu