İŞLETİM SİSTEMLERİ

İşletim Sistemi Yapıları

Prof.Dr. Ahmet ZENGİN

Öğrenme Hedefleri

Bu konuyu çalıştıktan sonra:

- İşletim Sistemi Servisleri
- Kullanıcı ve İşletim Sistemi Arayüzü
- Sistem Çağrıları
- Sistem Servisleri
- Bağlayıcılar(Linkers) ve Yükleyiciler (Loaders)
- Neden Uygulamalar İşletim Sistemine özgü?

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:

- **Kullanıcı Arayüzü:** Neredeyse tüm işletim sistemleri kullanıcı arayüzüne (UI) sahiptir.
  - **Komut Satırı Arayüzü (Command Line Interface),**
  - **Grafiksel Kullanıcı Arayüzü(Graphical User Interface),**
  - **Dokunmatik Ekran,**
  - **Çalıştırılabilir Dosyalar(Batch) vb.**
- **Uygulama Çalıştırma:** İşletim sistemi, programları belleğe yükleyerek çalıştırabilmeli ve çalışmakta olan programları normal ya da anormal durumlarda (hata belirterek) sonlandırabilmelidir.
- **I/O İşlemleri:** Çalışan bir program girdi/çıktı işlemleri yapmak isteyebilir (dosyadan ya da bir giriş/çıkış aygıtından).

İşletim Sistemleri Servisleri (devam)

- **Dosya-Sistemi Kullanma:** 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.

- **İletişim:** Prosesler aynı bilgisayar üzerinden veya ağ üzerinden bilgi değişimi yapabilir.
    - Bu iletişim, paylaşımlı bellek (*shared memory)* veya *mesaj iletimi (message passing)* yöntemlerinden biri ile gerçekleştirilir.

İşletim Sistemi Servisleri (devam)

Hata Tespiti– İşletim sistemi, oluşabilecek hatalara karşı sürekli tetikte olmalıdır.

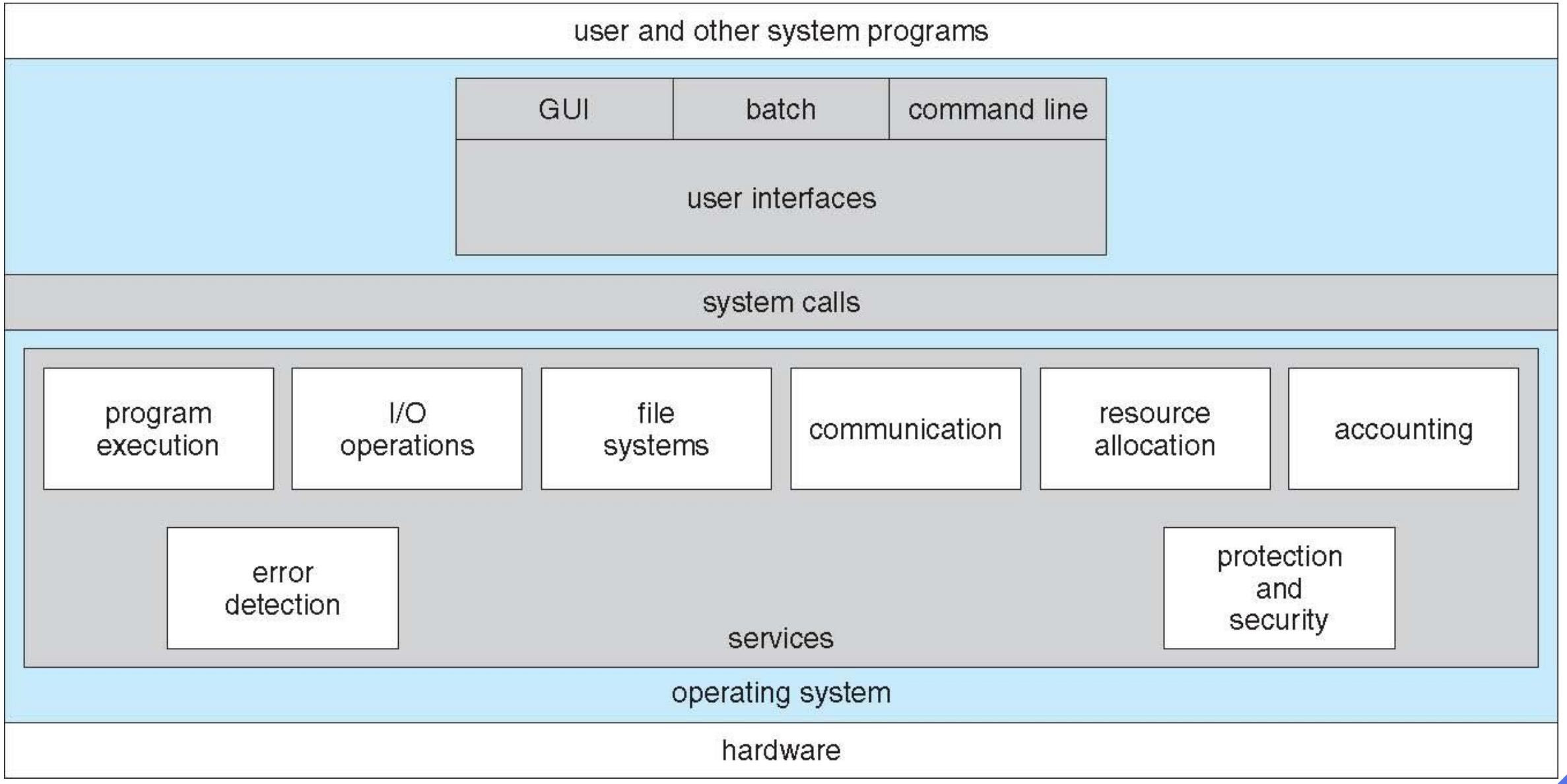
- İşlemcide, fiziksel bellekte, bağlı durumdaki bir giriş/çıkış aygıtında ya da kullanıcı programında hata ile karşılaşılabilir.
- İşletim sistemi her türlü hataya karşı önlem almalıdır.
- Hata ayıklama araçları, kullanıcı ve programcının sistemi etkin olarak kullanabilmesine çok büyük katkı sağlar.

İşletim Sistemi Servisleri (devam)

Bazı işletim sistemi servisleri de sistemin kendi kendine etkin bir biçimde kaynak paylaşımı yapabilmesini sağlar:

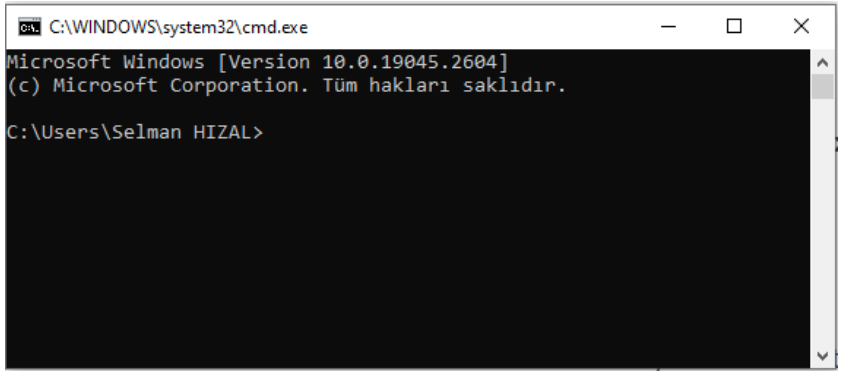
- **Kaynak Tahsisi -** Birçok kullanıcı veya görevin (*task*) aynı anda çalıştığı sistemlerde, her biri için kaynak ayrılmasını sağlar.
- (İşlemci, bellek ve sabit disk gibi) Pek çok aygıt için özel bir kaynak ayırma işlemi uygulanırken, (giriş/çıkış aygıtları gibi) bazı aygıtlarda ise daha genel bir kaynak ayırma işlemi uygulanır.
- **Loglama–** Kullanıcıların ne kadar kaynak kullandığını takip edilmelidir.
- **Koruma ve Güvenlik -** Çok sayıda proses eş zamanlı işletilirken, bir prosesin diğer bir prosesin işleyişine veya işletim sisteminin işleyişine karışamıyor olması gerekir. Bu yüzden bir proses, başka bir prosesin bellek bölgesine erişemez.
  - **Koruma**, sistem kaynaklarına erişimin 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



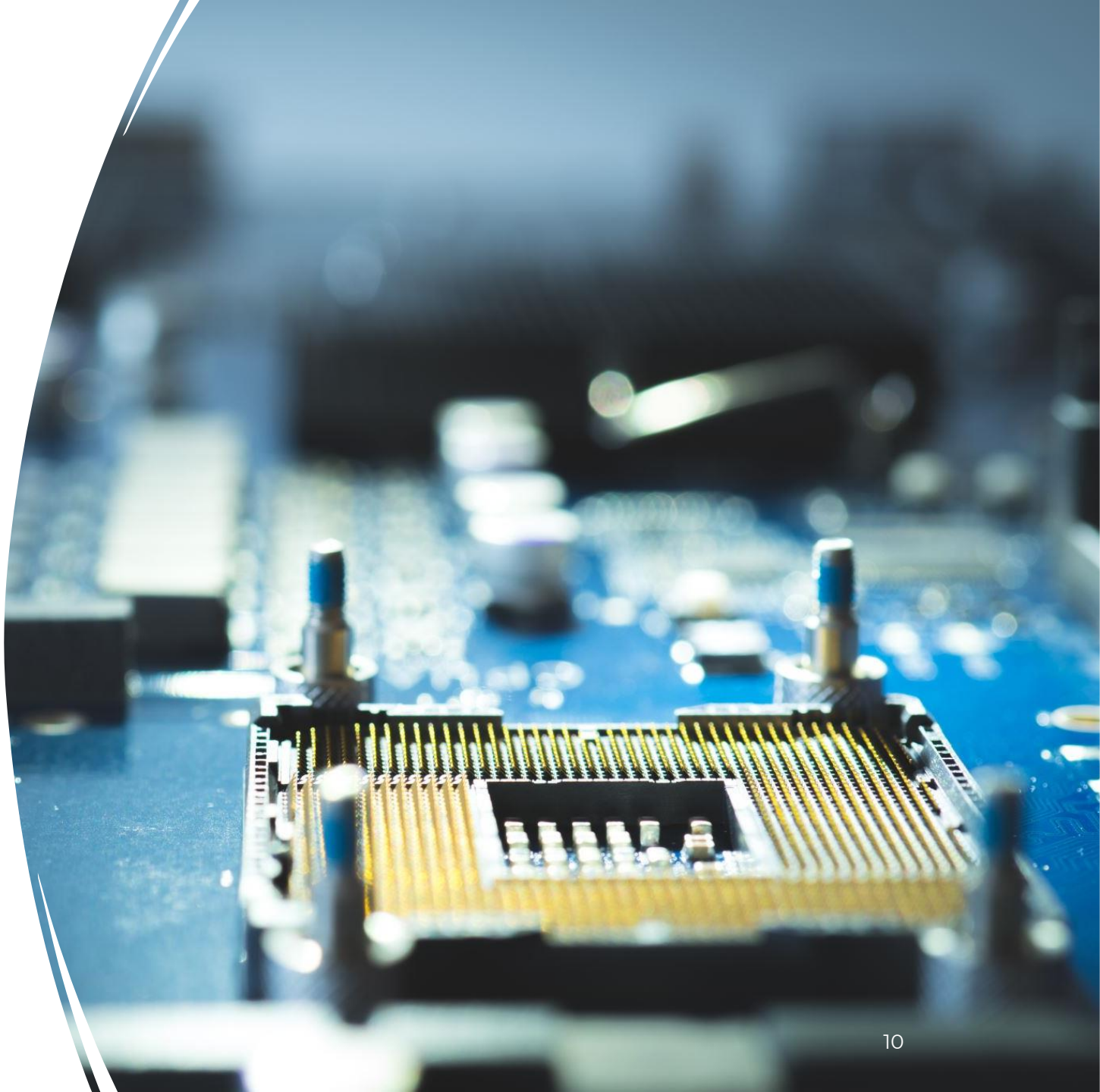
Kullanıcı İşletim Sistem Arayüzü

- CLI - Command Line Interface – Komut Satırı Arayüzü
  - Doğrudan komut girişi
- GUI – graphical user interface – grafik kullanıcı arayüzü
- Dokunmatik ekran arayüzleri
- Toplu işlem

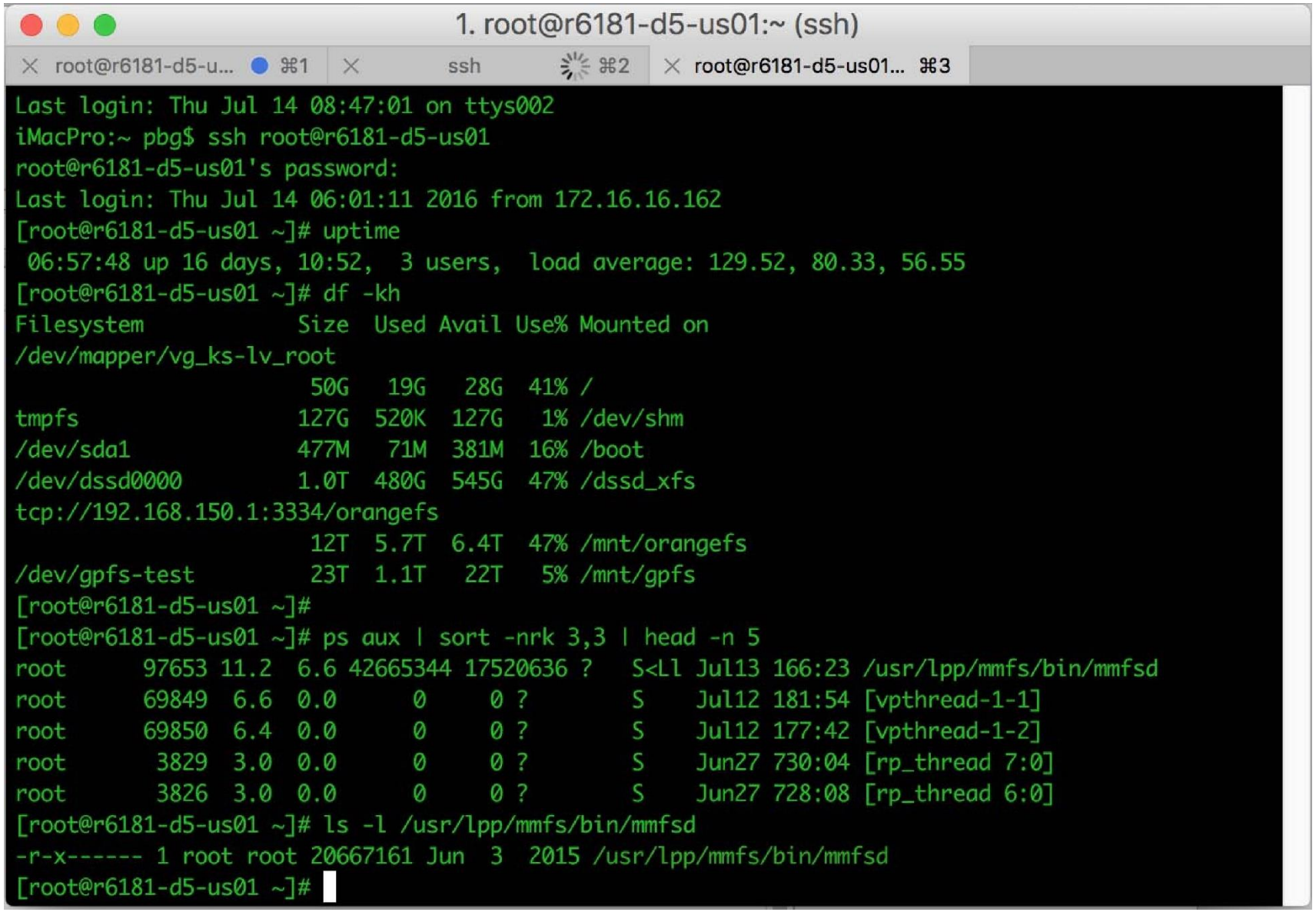


Kullanıcı Arayüzü - CLI

- Komut satırı arayüzü ya da **komut yorumlayıcısı** doğrudan sisteme komut verme imkanı sağlar.
    - Komut satırından doğrudan ya da sistem servisleri vasıtasıyla işletim sistemi çekirdeğine (kernel'e) 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.



Bourne Shell Komut Yorumlayıcısı



Bourne kabuğu (sh), komut dosyası oluşturmak için kullanılan bir UNIX kabuğu veya komut işlemcisidir. 1977 yılında AT&T'den Stephen Bourne tarafından geliştirildi ve Mashey kabuğunun (sh) yerine UNIX Sürüm 7'de tanıtıldı.

Bourne kabuğu, komut istemleriyle kullanılan yürütülebilir program adı "sh" ve dolar simgesi "$" ile de bilinir.

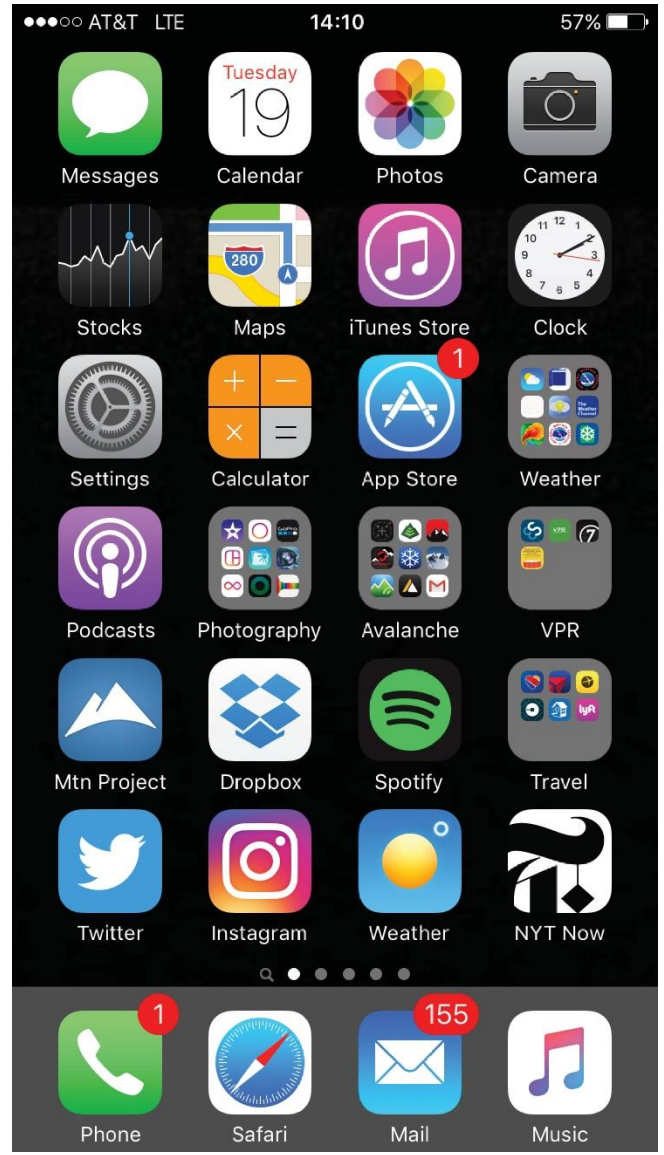
https://en.wikipedia.org/wiki/Bourne_shell

Kullanıcı Arayüzü - GUI

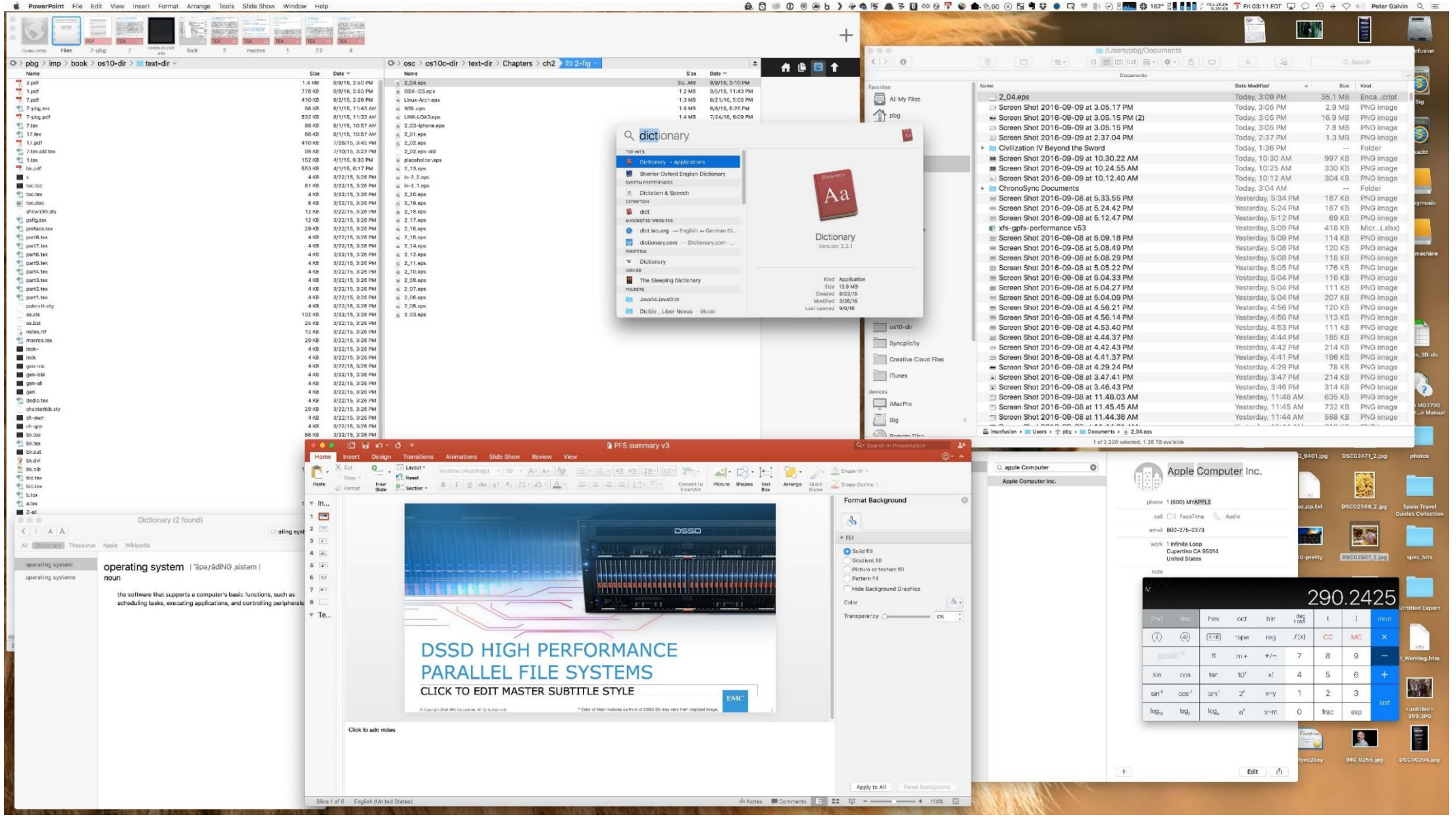
- Kullanıcı dostu, bilgi ve görsellerle donatılmış arayüz.
  - Fare, klavye ve monitör.
  - Programları, dosyaları vb. temsil eden **simgeler** mevcuttur.
  - Fare yardımı ile çeşitli eylemler gerçekleştirilir(bilgi getirme, seçenekler, fonksiyon çalıştırma, dizin açma)
  - İlk kez Xerox PARC tarafından 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.
  - Unix ve Linux CLI'ya sahiptir ayrıca opsiyonel GUI kullanır (CDE, KDE, GNOME)
  - Solaris CLI ve opsiyonel olarak GUI barındırır (Java Desktop, KDE).

Dokunmatik Arayüzler

- Dokunmatik ekranlı cihazlar yeni arabirimler gerektiriyor
  - Fare mümkün değil veya istenmiyor
  - Hareketlere dayalı eylemler ve seçim
  - Metin girişi için sanal klavye
- Sesli komutlar



The Mac OS X GUI



https://developer.apple.com/

Sistem Çağrıları

- İşletim sistemi tarafından sağlanan servislere programlama arayüzü

- 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.

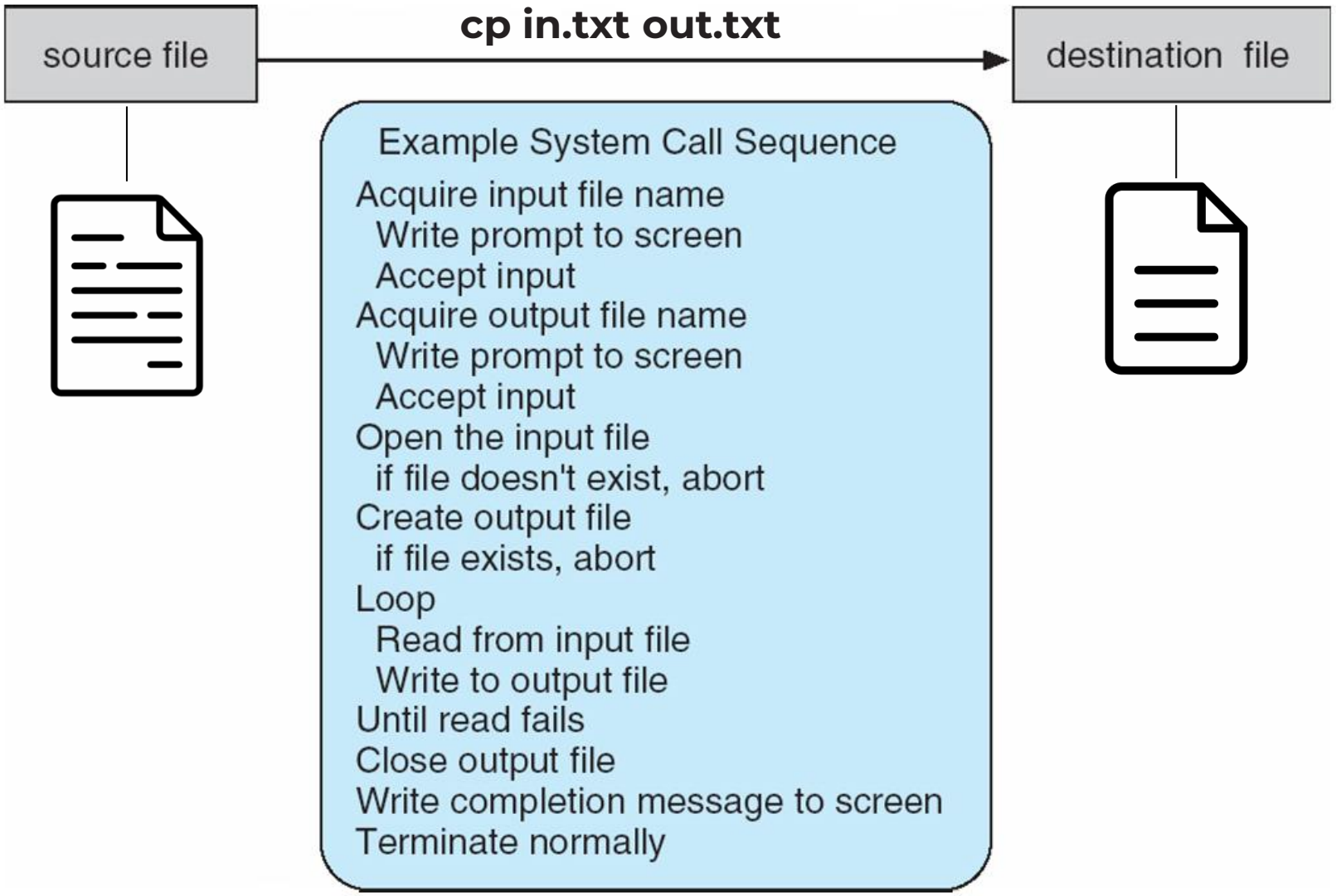
- **En sık kullanılan 3 API,**
    1. Win32 API (Windows için),
    2. POSIX tabanlı sistemler için POSIX API (UNIX, Linux ve MacOS X 'in tüm versiyonlarında mevcuttur) ve
    3. Java API (Java Sanal Makinesi için)'dir.

- Neden doğrudan sistem çağrısı yapmak yerine API kullanılır?

  (Not: Yazı boyunca sistem çağrısı adı kullanılacaktır.)

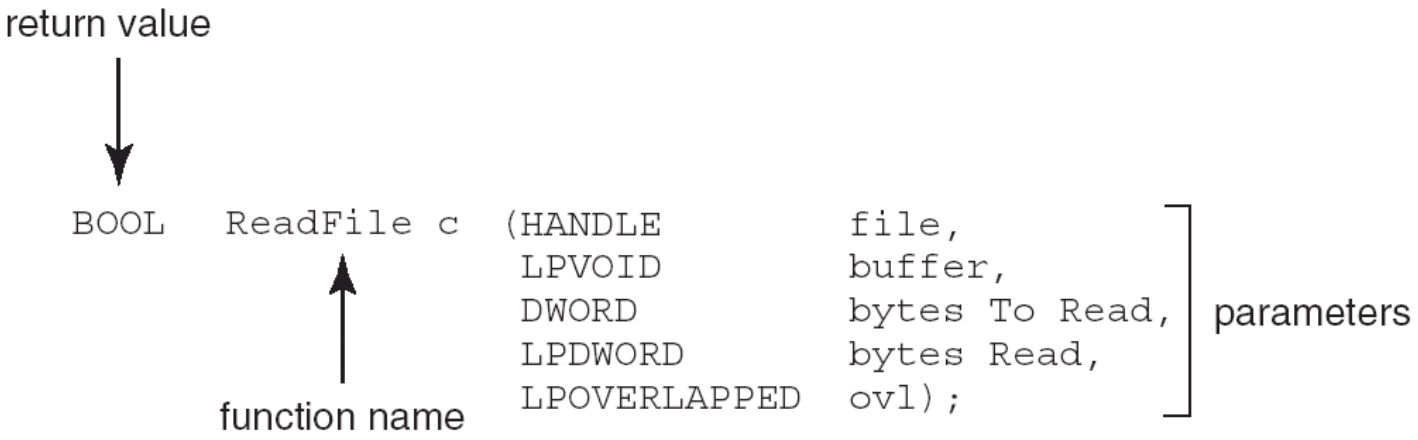
Sistem Çağrısı Örneği

- Bir dosyadan diğerine içerik kopyalamak için UNIX'de yapılan sistem çağrısı.



Standard API Örneği

- **ReadFile( )** fonksiyonunu göz önüne alalım Win32 API: dosyadan okuma yapan bir



ReadFile( )'a geçen parametrelerin bir tanımı

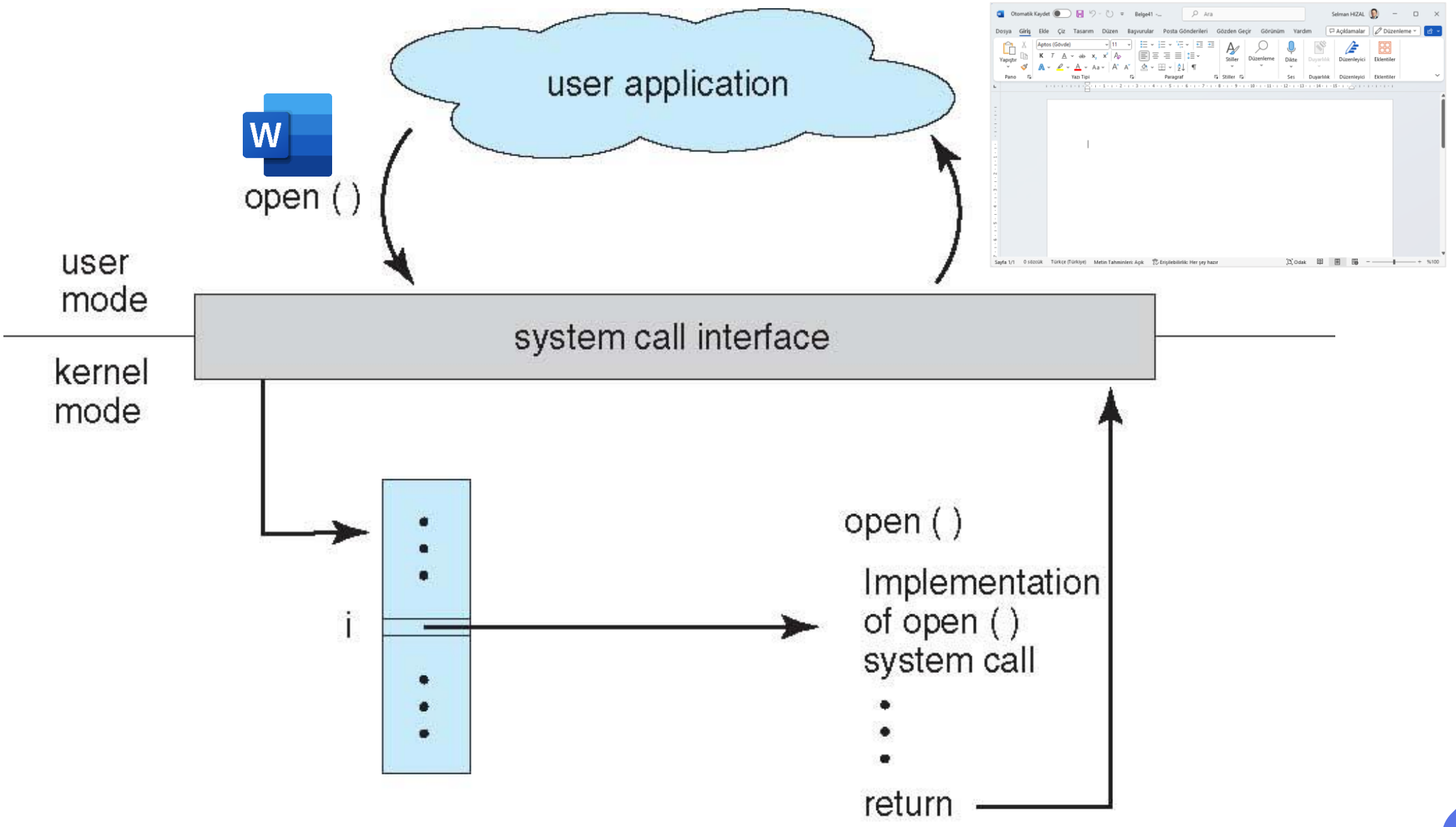
- HANDLE file—okuma yapılacak dosya
- LPVOID - okunan verinin yazılacağı bellek alanı
- DWORD bytesToRead- okunacak veri boyutu
- LPDWORD bytesRead— en son okunan bayt miktarı
- LPOVERLAPPED ovl—çakışmalı(overlapped) I/O kullanılıp kullanılmadığını gösterir

https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile

Sistem Çağrısı Uyarlaması

- Her sistem çağrısının kendine ait bir numarası vardır.
  - **Sistem çağrısı arayüzü**, bu numaraları bir tablo halinde tutar.
- Sistem çağrı arayüzü, işletim sistemi çekirdeğindeki 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)

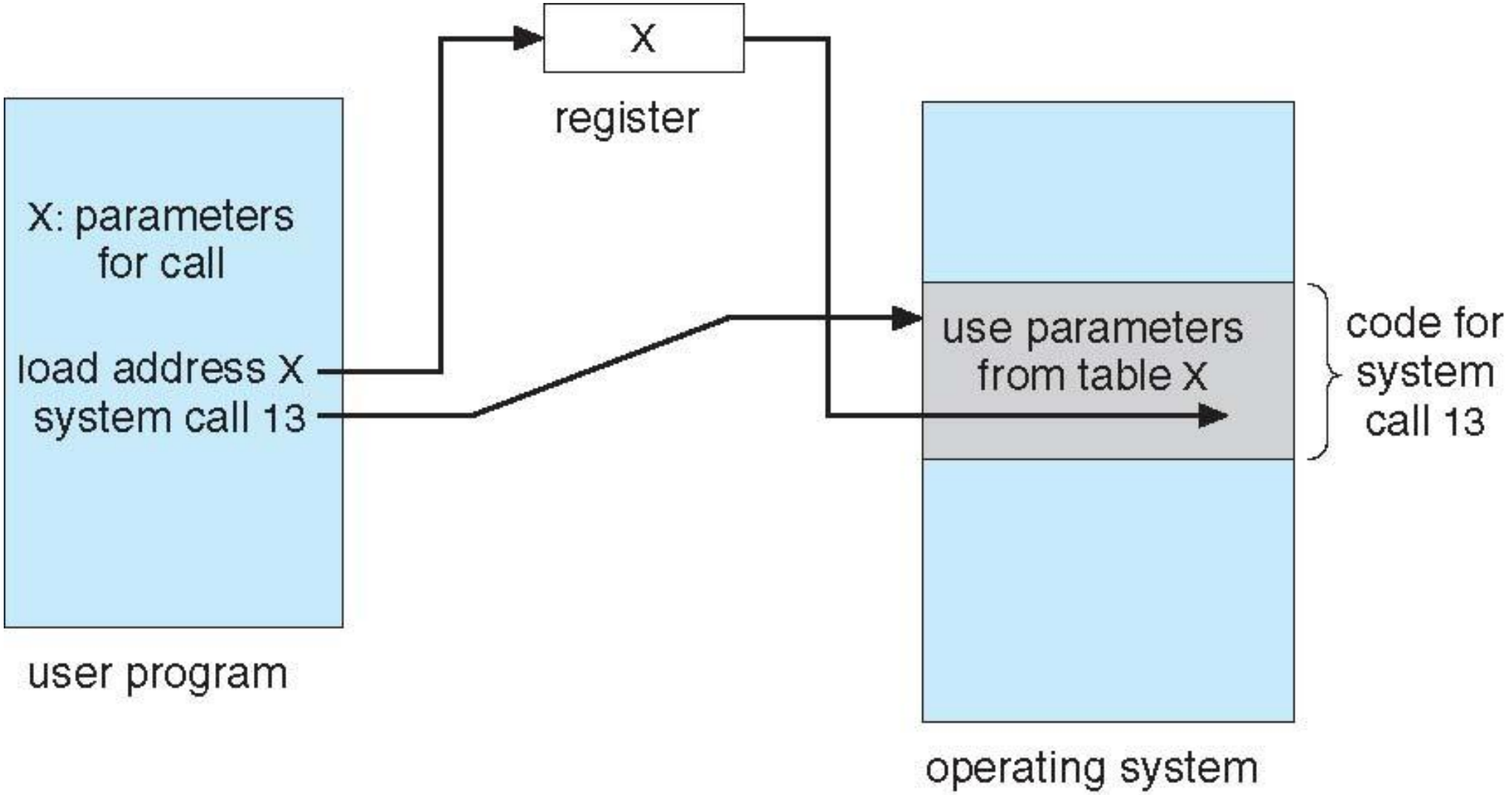
API – Bir Dosyayı Açan Sistem Çağrısı



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 kaydedicilere (register) yazma.
    - Bazı durumlarda, parametre miktarı kaydedicilerden fazla olabilir.
  - Parametreler, bellek içerisinde blok ya da tabloda tutulur ve blok adresleri kaydediciye yazılır.
    - Bu yaklaşım Linux ve Solaris'ten alınmıştır.
  - Parametreler, program tarafından, **yığın** (*stack)* üzerine eklenir (**push***) ve* işletim sistemi yığın üzerinden alır (**pop**).
  - Blok ve yığın metotları, sayılara ya da geçilen parametre uzunluklarına bir limit koymaz.

Tablo ile Parametre Geçişi



Sistem Çağrı Tipleri

- **Process kontrolü**
  - Prosesleri oluşturma, sonlandırma
  - İptal etme
  - Yükleme, çalıştırma
  - Proses özniteliklerini alma, ve belirleme
  - Belirli bir süre bekleme
  - Olay bekleme devam ettirme
  - Bellek ayırma, bellek bölgesini geri iade
  - Hata anında belleği boşaltma
  - **Bug (hata)** bulan **hata ayıklayıcı**, **adım adım** çalıştırma
  - Prosesler arasında paylaşılan veriye erişim **kilitleri**

Sistem Çağrı Tipleri(devam)

- **Dosya yönetimi**
  - Dosya oluşturmak, silmek
  - Dosya açmak, kapatmak
  - Dosyadan okuma, dosyaya yazma, konum değiştirme
  - Dosya özelliklerini (attribute) değiştirmek
- **Aygıt yönetimi**
  - Aygıt talep etme, aygıtı serbest bırakma
  - Okuma, yazma, konum değiştirme
  - Aygıt özelliklerini getirme, aygıt değerlerini değiştirme
  - Aygıt ekleme, kaldırma

Sistem Çağrı Tipleri(devam)

- **Bilgilendirme Hizmeti**
  - Zamanı ya da tarihi getirme, değiştirme
  - Sistem verisini getirme, değiştirme
  - İşlem, dosya ya da aygıt değerlerini getirme, değer atama
- **İletişim**
  - Bağlantı oluşturmak, silmek
  - **Ana bilgisayar** adına veya **proses adına mesaj geçiş modeli** varsa mesaj gönder, al
    - **İstemci sunucu** modeli
  - **paylaşılan bellek modeli**, bellek bölgeleri oluşturur ve bunlara erişim sağlar
  - Durum bilgisi transferi
  - Uzak aygıta bağlanmak, bağlantıyı sonlandırmak

Sistem Çağrı Tipleri(devam)

- **Koruma**
  - Kaynaklara erişimi kontrol
  - İzinleri yönetme
  - Kullanıcı erişimine izin yada engelleme

Windows ve Unix Sistem Çağrı Örnekleri

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

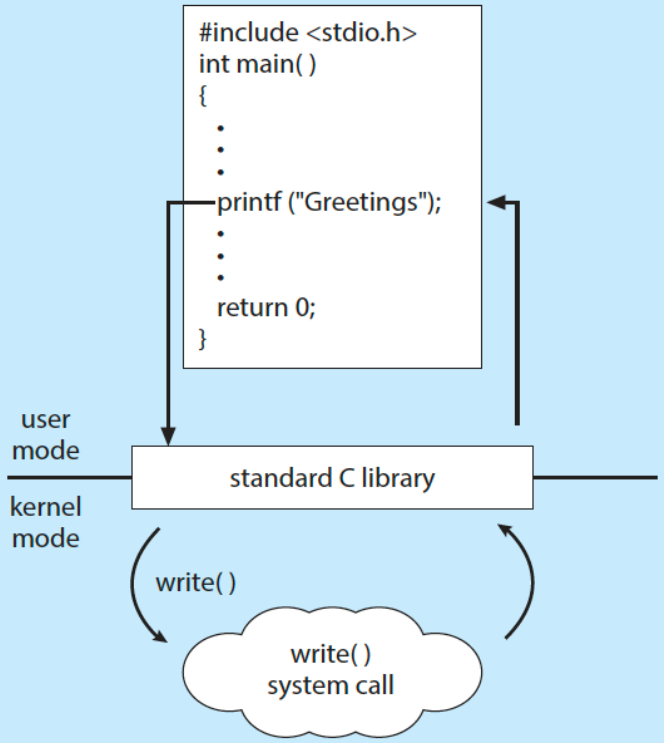
	Windows	Unix
Process control	`CreateProcess()`	`fork()`
	`ExitProcess()`	`exit()`
	`WaitForSingleObject()`	`wait()`
File management	`CreateFile()`	`open()`
	`ReadFile()`	`read()`
	`WriteFile()`	`write()`
	`CloseHandle()`	`close()`
Device management	`SetConsoleMode()`	`ioctl()`
	`ReadConsole()`	`read()`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

Standard C Kütüphanesi Örneği

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

THE STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the `write()` system call. The C library takes the value returned by `write()` and passes it back to the user program:

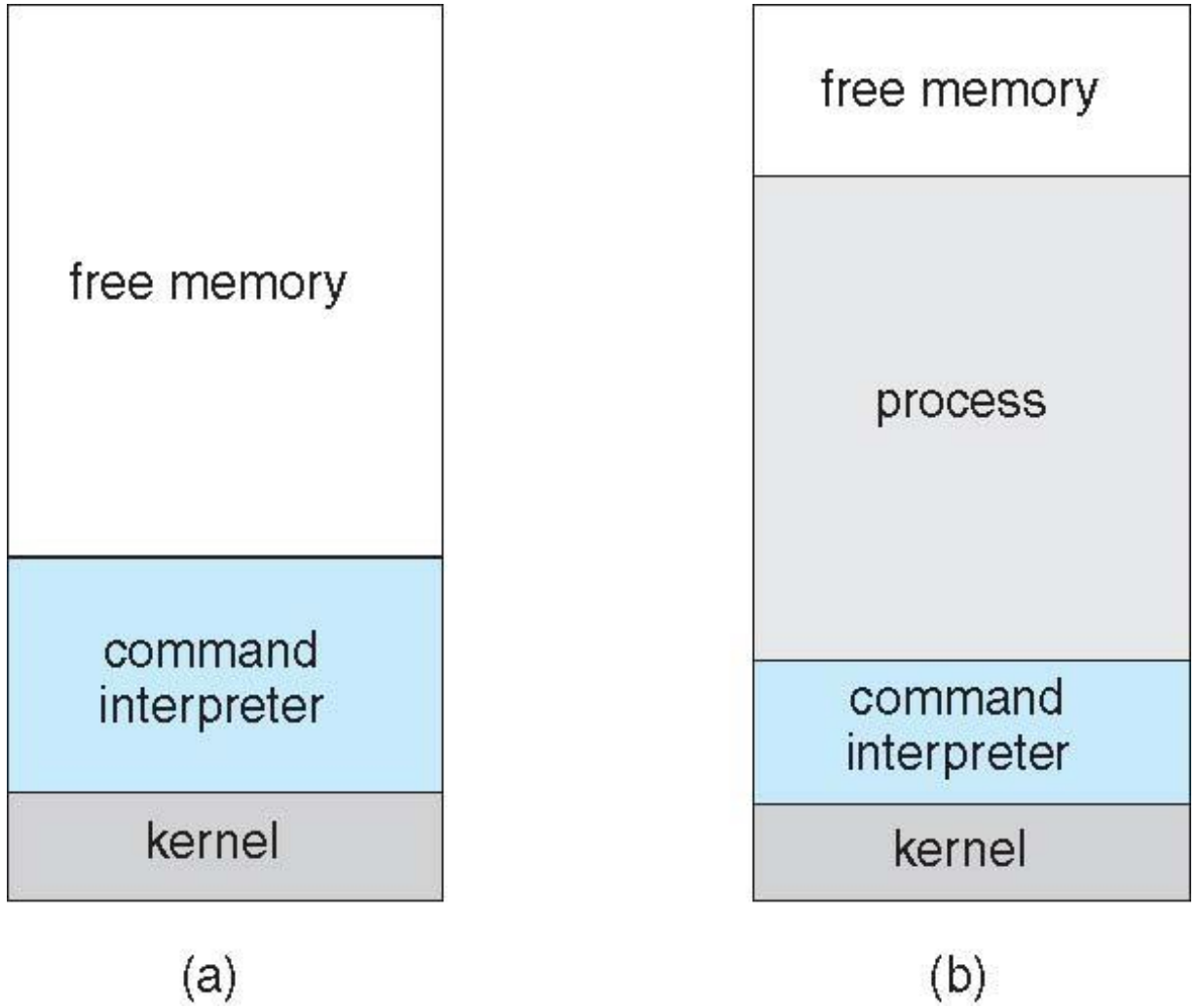


Örnek: Microsoft Disk İşletim Sistemi (MicroSoft Disk Operating System) MS-DOS

- 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

https://tr.wikipedia.org/wiki/MS-DOS

MS-DOS Yürütme



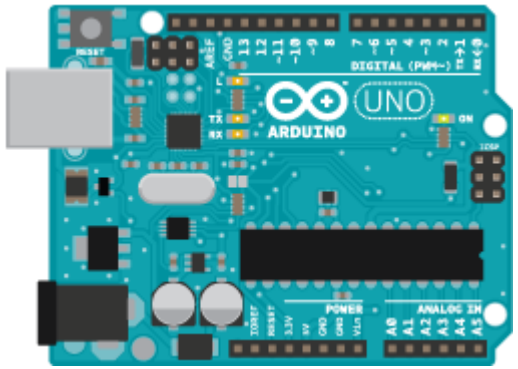
(a) Sistem ilk başlatıldığında

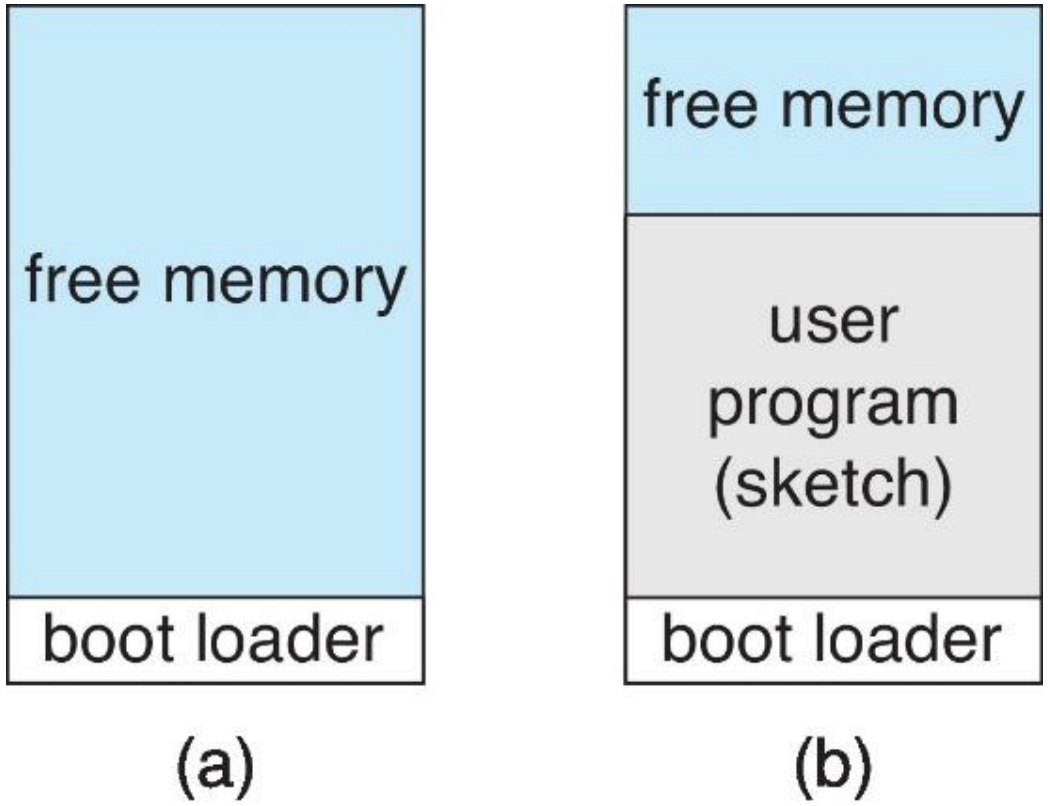
(b) (b) Çalışmakta olan program

Örnek: Arduino

Arduino, ışık, sıcaklık ve barometrik basınçtaki değişiklikler gibi çeşitli olaylara yanıt veren giriş sensörleri ile birlikte bir mikro denetleyiciden oluşan basit bir donanım platformudur.

- Tek görev
- İşletim sistemi yok
- USB aracılığıyla flash belleğe yüklenen programlar
- Tek hafıza alanı
- Önyükleyici programı yükler
- Program çıkışı -> kabuk yeniden yüklenir



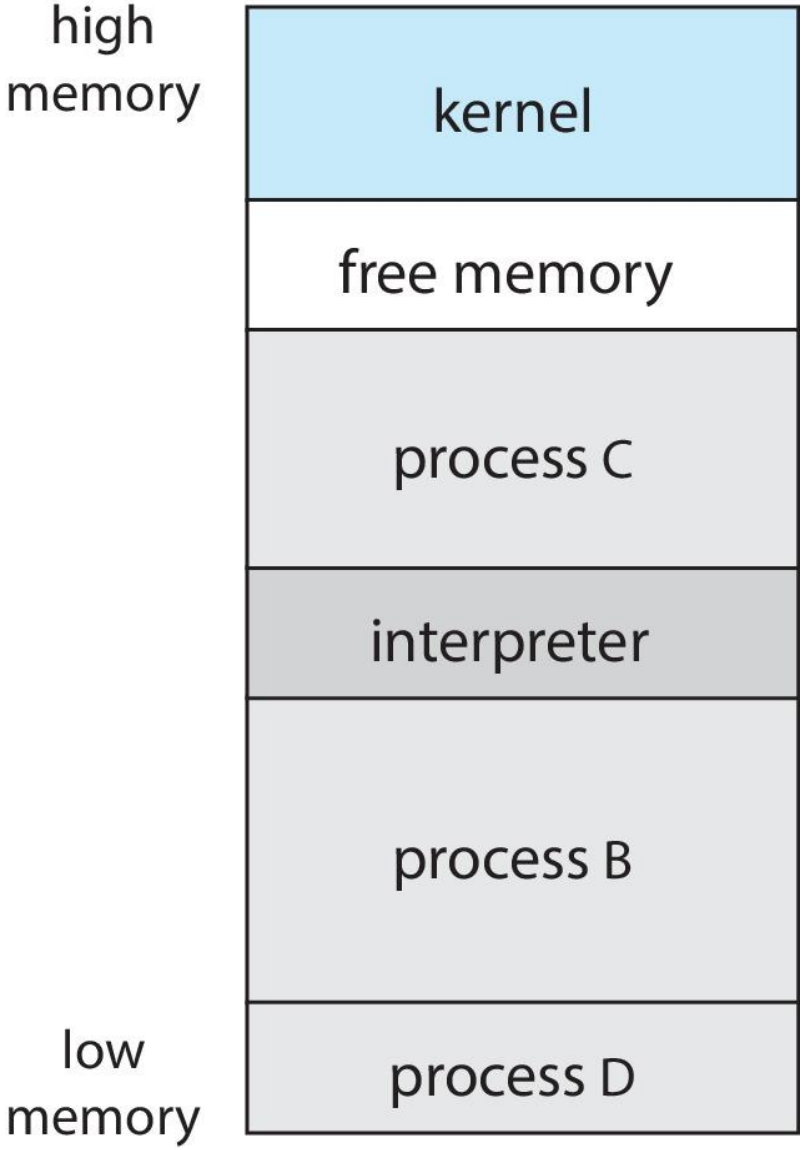


https://www.arduino.cc/

Örnek: FreeBSD

- Unix türevi
- Aynı anda birden fazla program (multitasking)
- Kullanıcı girişi -> kullanıcının kabuk seçimini talep eder
- 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
  - Kod = 0 - hatasız
  - Kod > 0 - hata kodu

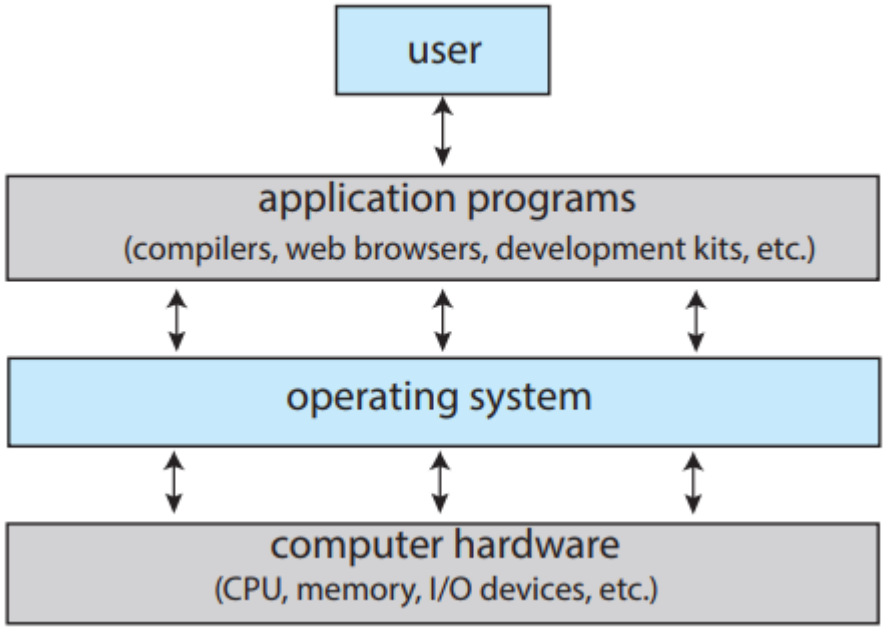
  İle sonlanır



https://www.freebsd.org/

Sistem Servisleri

- En alt seviyede donanım bulunur. Ardından işletim sistemi gelir, ardından sistem hizmetleri ve son olarak da uygulama programları gelir.
- Sistem programları program geliştirme ve çalıştırma için ortam sağlar.
- Sistem servislerini şu başlıklara ayırabiliriz:
  - Dosya yönetimi
  - Durum bilgisi
  - Programlama dili desteği
  - Program yükleme ve çalıştırma
  - İletişim
  - Arkaplan servisleri
  - Uygulama programları



- Kullanıcıların çoğu gerçek sistem çağrıları yerine sistem programlarını kullanılır.

Sistem Servisleri(devam)

- 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 **kaydedicide** 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)

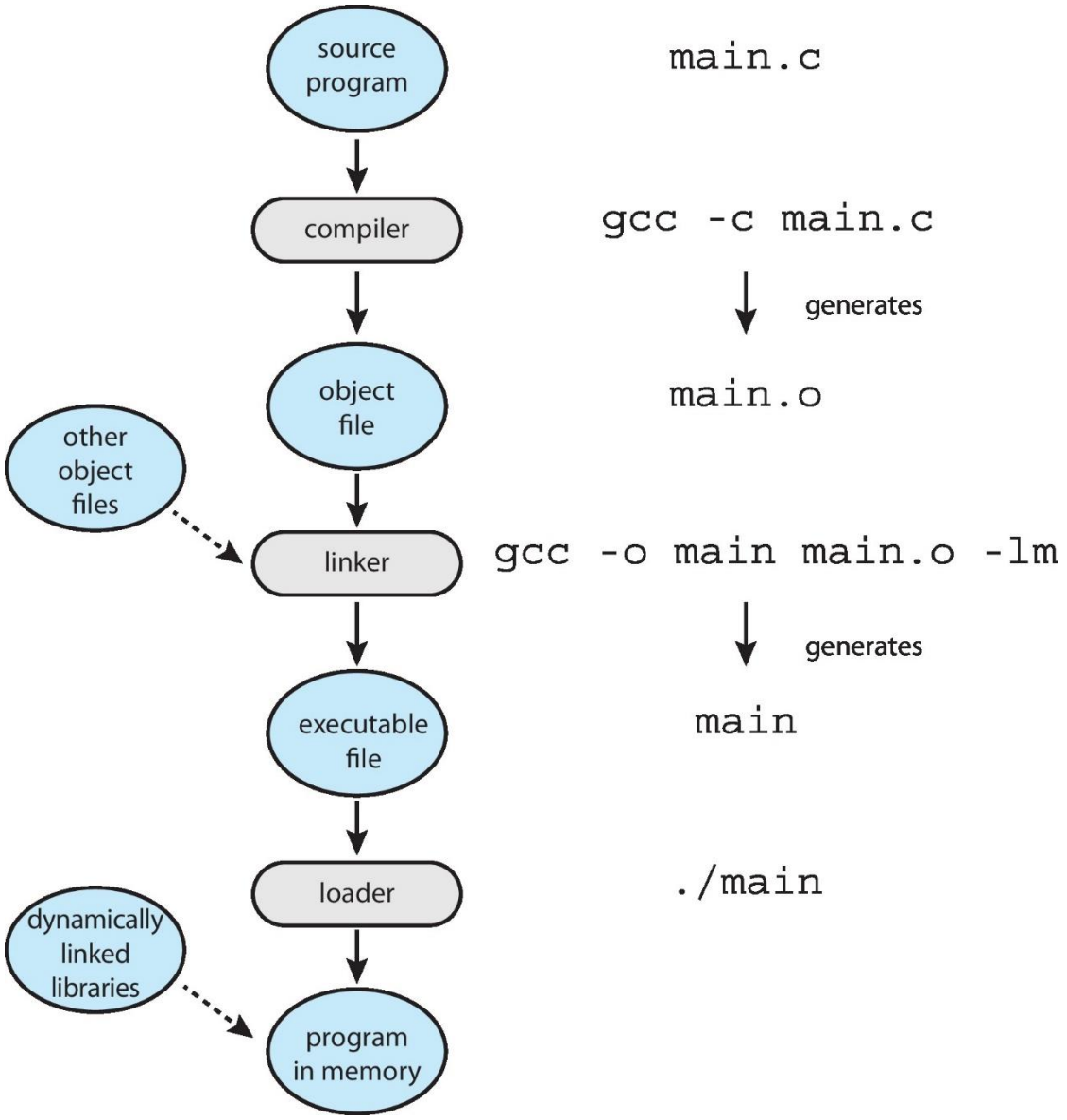
- **Arka Plan Servisleri**
  - Önyükleme zamanında başlar
    - Bazıları sistem başlangıcında, ardından sonlanır
    - Bazıları sistem önyüklemesinden kapatmaya kadar
  - Disk kontrolü, süreç planlaması, hata günlüğü, yazdırma gibi olanaklar sağlar
  - Çekirdek bağlamında değil kullanıcı bağlamında çalışır
  - **Servisler, alt sistemler, daemonlar** olarak bilinir

- **Uygulama programları**
  - Sistemle ilgili değil
  - Kullanıcılar tarafından çalıştırılır
  - Genellikle işletim sisteminin bir parçası olarak kabul edilmez
  - Komut satırı, fare tıklaması, parmak dokunması ile başlatılır

Bağlayıcılar (Linkers) ve Yükleyiciler (Loaders)

- Herhangi bir fiziksel bellek konumuna yüklenmek üzere tasarlanmış nesne dosyalarında derlenmiş kaynak kodu – **yeniden konumlandırılabilir nesne dosyası**
- **Bağlayıcı**, bunları tek bir ikili **yürütülebilir** dosyada birleştirir
  - Ayrıca kütüphaneleri getirir
- Program, ikili yürütülebilir dosya olarak ikincil depolamada bulunur
- Yürütülmesi için **yükleyici** tarafından belleğe getirilmelidir
  - **Yer değiştirme(relocation)**, program parçalarına son adresleri atar ve programdaki kod ve verileri bu adreslerle eşleşecek şekilde ayarlar
- Modern genel amaçlı sistemler, kitaplıkları yürütülebilir dosyalara bağlamaz
  - Bunun yerine, **dinamik olarak bağlantılı kitaplıklar** (Windows'ta, **DLL'lerde**) gerektiği gibi yüklenir, aynı kitaplığın aynı sürümünü kullanan herkes tarafından paylaşılır (bir kez yüklenir)
- Nesne, yürütülebilir dosyaların standart biçimleri vardır, bu nedenle işletim sistemi bunları nasıl yükleyeceğini ve başlatacağını bilir

Bağlayıcı (linker) ve Yükleyicinin (loader)Rolü




```c
#include <stdio.h>

int main() {
    int number = 42;
    float floatValue = 3.14;
    char character = 'A';
    char string[] = "Hello, world!";

    // Printing integer
    printf("Integer: %d\n", number);

    // Printing float
    printf("Float: %f\n", floatValue);

    // Printing character
    printf("Character: %c\n", character);

    // Printing string
    printf("String: %s\n", string);

    return 0;
}
```

https://sourceforge.net/projects/mingw/files/latest/download
https://en.wikipedia.org/wiki/Linker_(computing)

Uygulamalar Neden İşletim Sistemine Özgü

1. **API Farklılıkları**: Her işletim sistemi farklı bir API (Application Programming Interface - Uygulama Programlama Arayüzü) setine sahiptir. Bu API'ler, uygulamaların işletim sistemine erişmesini ve sistem kaynaklarını kullanmasını sağlar. Dolayısıyla, bir uygulama, çalıştırılacağı işletim sisteminin API'lerine uygun olarak geliştirilmelidir.
2. **Sistem Kaynaklarına Erişim**: İşletim sistemleri, bellek, dosya sistemi, ağ, donanım cihazları gibi sistem kaynaklarını yönetir. Farklı işletim sistemleri, bu kaynaklara erişmek ve yönetmek için farklı yöntemler sağlar. Bu nedenle, bir uygulama, çalıştığı işletim sisteminin kaynak yönetim modeline uygun olarak tasarlanmalıdır.
3. **Donanım Farklılıkları**: Farklı işletim sistemleri, farklı donanım platformlarında çalışabilir. Örneğin, bir uygulamanın Windows işletim sistemi sürümü, Linux işletim sistemi sürümü veya macOS işletim sistemi sürümü olabilir. Bu nedenle, uygulamanın donanım farklılıklarını dikkate alması ve uyumlu olması gerekmektedir.
4. **Grafiksel Arabirimler**: İşletim sistemleri genellikle grafiksel kullanıcı arabirimleri (GUI) sağlar. Ancak, farklı işletim sistemleri farklı GUI kütüphanelerini ve araçlarını kullanır. Bu nedenle, uygulamaların GUI öğeleri işletim sistemine özgü olabilir.
5. **Performans ve Güvenlik**: İşletim sistemleri, farklı performans ve güvenlik özellikleri sağlar. Uygulamalar, bu özellikleri göz önünde bulundurarak geliştirilmelidir. Örneğin, bir uygulama, Windows işletim sistemi için geliştirilen bir güvenlik modeliyle uyumlu olmalıdır.

Bu nedenlerden dolayı, uygulamaların genellikle işletim sistemine özgü olması gerekmektedir. Ancak, çapraz platform geliştirme araçları ve teknolojileri kullanılarak, aynı uygulamanın farklı işletim sistemlerinde çalışabilir hale getirilmesi mümkündür.