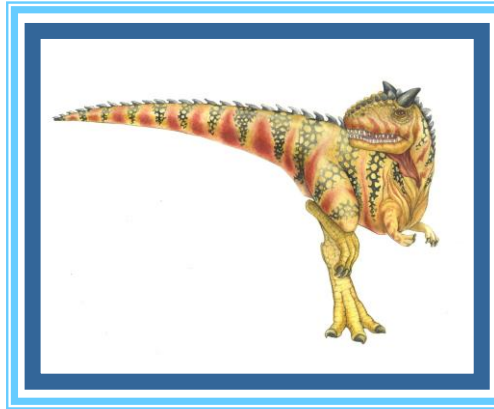


Bölüm 2: İşletim Sistemi Servisleri





İçindekiler

- İşletim Sistemi Servisleri
- Kullanıcı ve İşletim Sistemi Arayüzü
- Sistem Çağrılar
- Sistem Servisleri
- Bağlayıcılar(Linkers) ve Yükleyiciler (Loaders)
- Neden Uygulamalar OS ye özgü

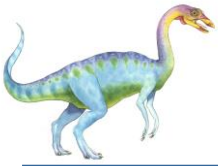




Hedefler

- 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 sahiptir(UI).
 - ▶ **Komut Satırı (CLI), Grafiksel Kullanıcı Arayüzü(GUI), 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/çıkırtı 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 *meaj 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.





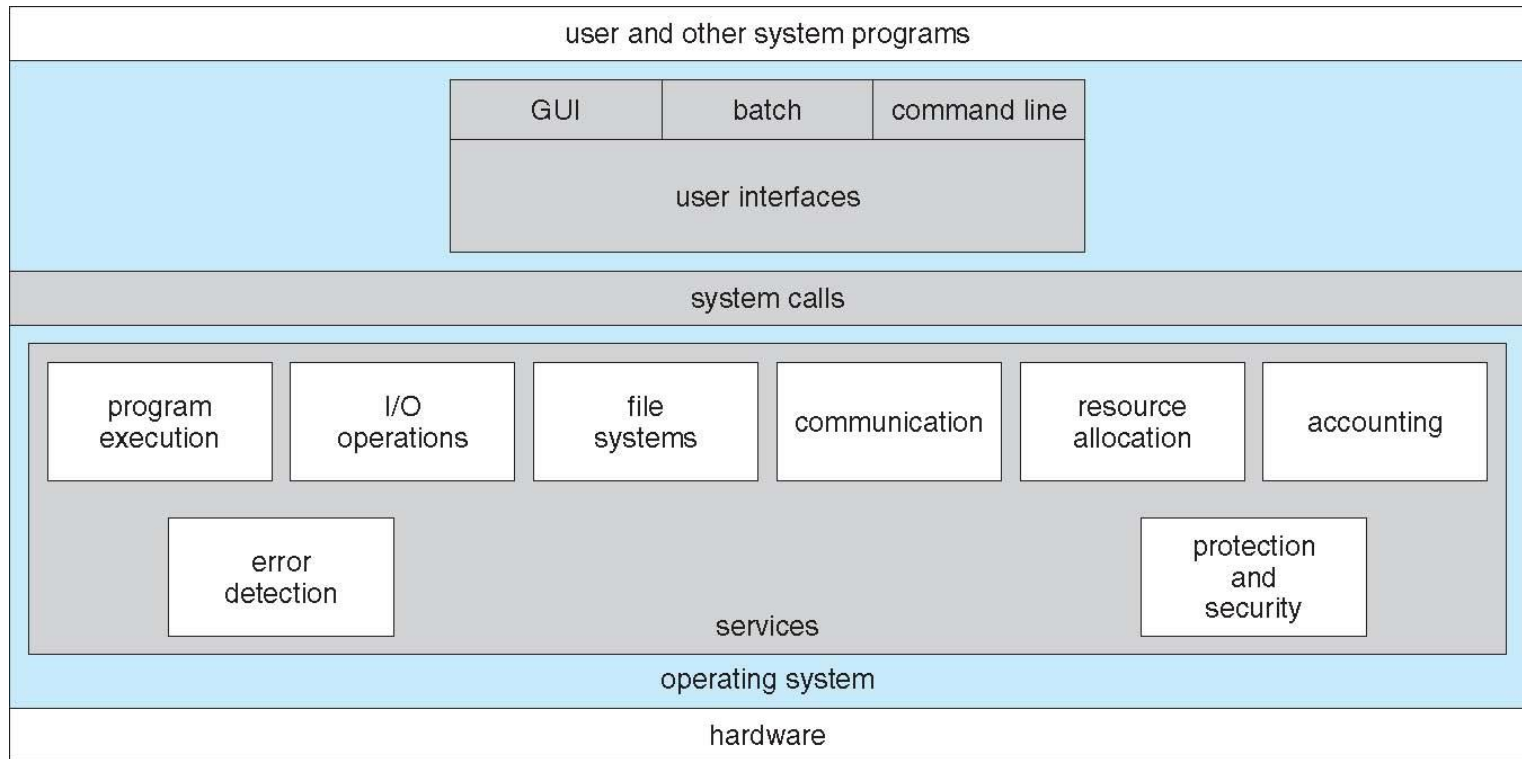
İşletim Sistemi Servisleri (devam)

- 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 karışmıyor 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

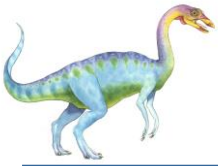




Kullanıcı İşletim Sistem Arayüzü

- CLI -- command line interpreter – komut satırı yorumlayıcısı
 - 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 (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.

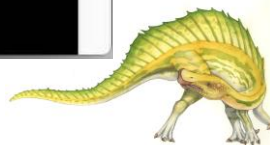


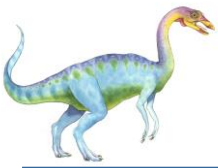


Bourne Shell Komut Yorumlayıcısı

```
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/orangeifs
                          12T     5.7T   6.4T  47% /mnt/orangeifs
/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

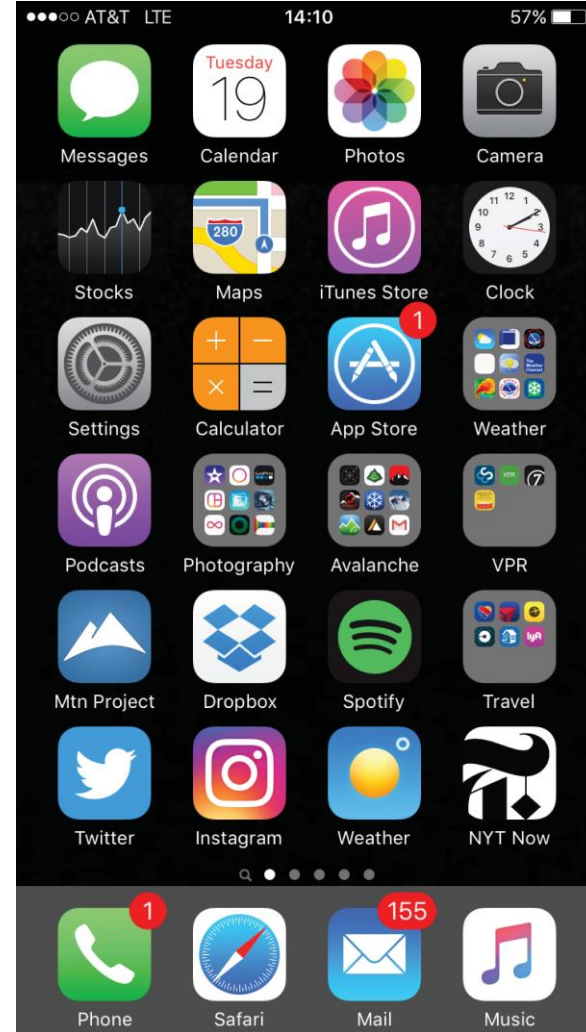
- Kullanıcı dostu, bilgi ve görsellerle donatılmış arayüz.
 - Fare, klavye ve monitör.
 - Programları, dosyaları vb. temsil eden **simgeler** mevcuttur.
 - Fare yardımı ile çeşitli eylemler gerçekleştirilir(bilgi getirme, seçenekler, fonksiyon çalıştırma, dizin açma)
 - İlk kez Xerox PARC tarafından 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).





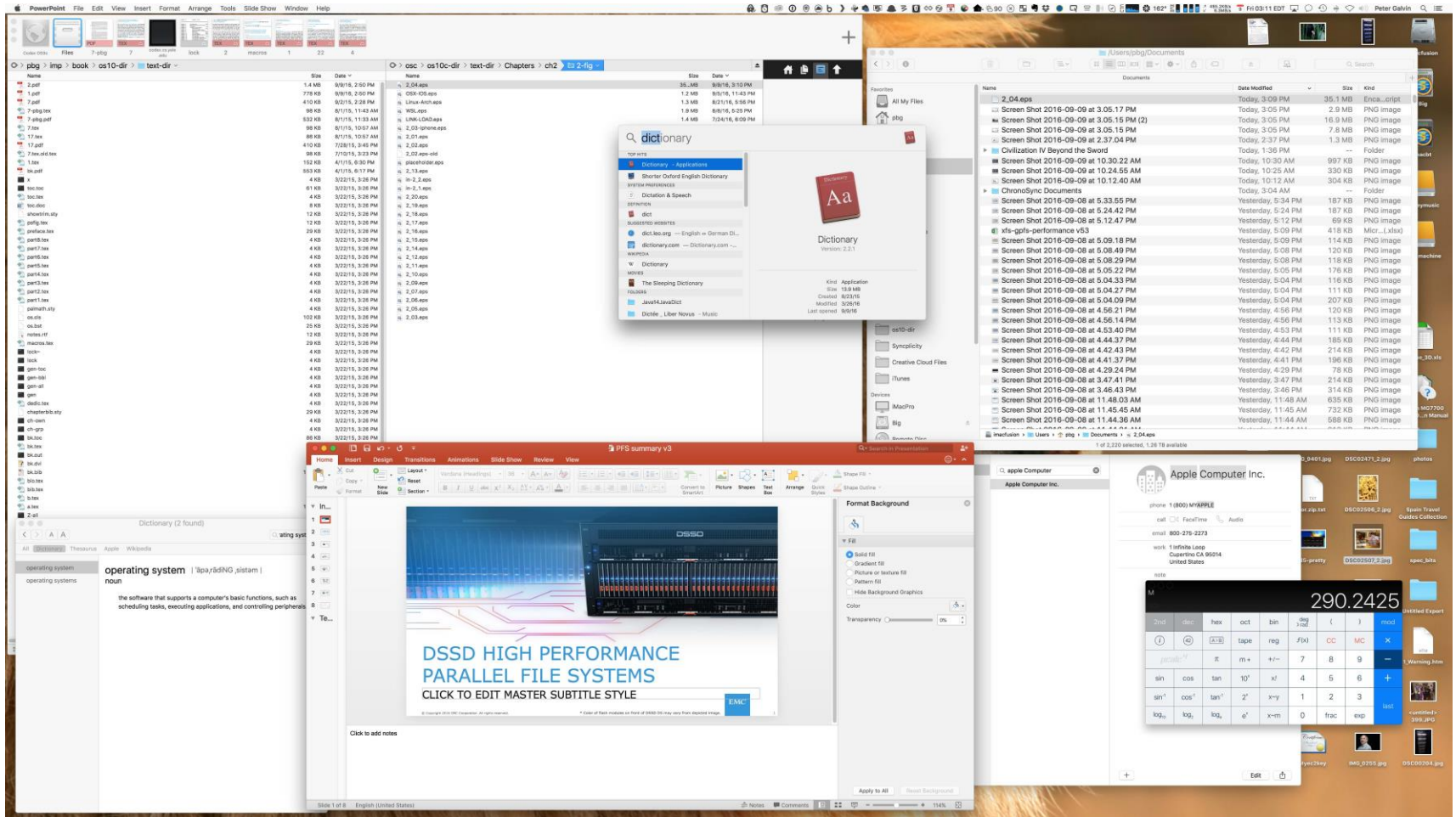
Dokunmatik Arayüzler

- 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





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 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?

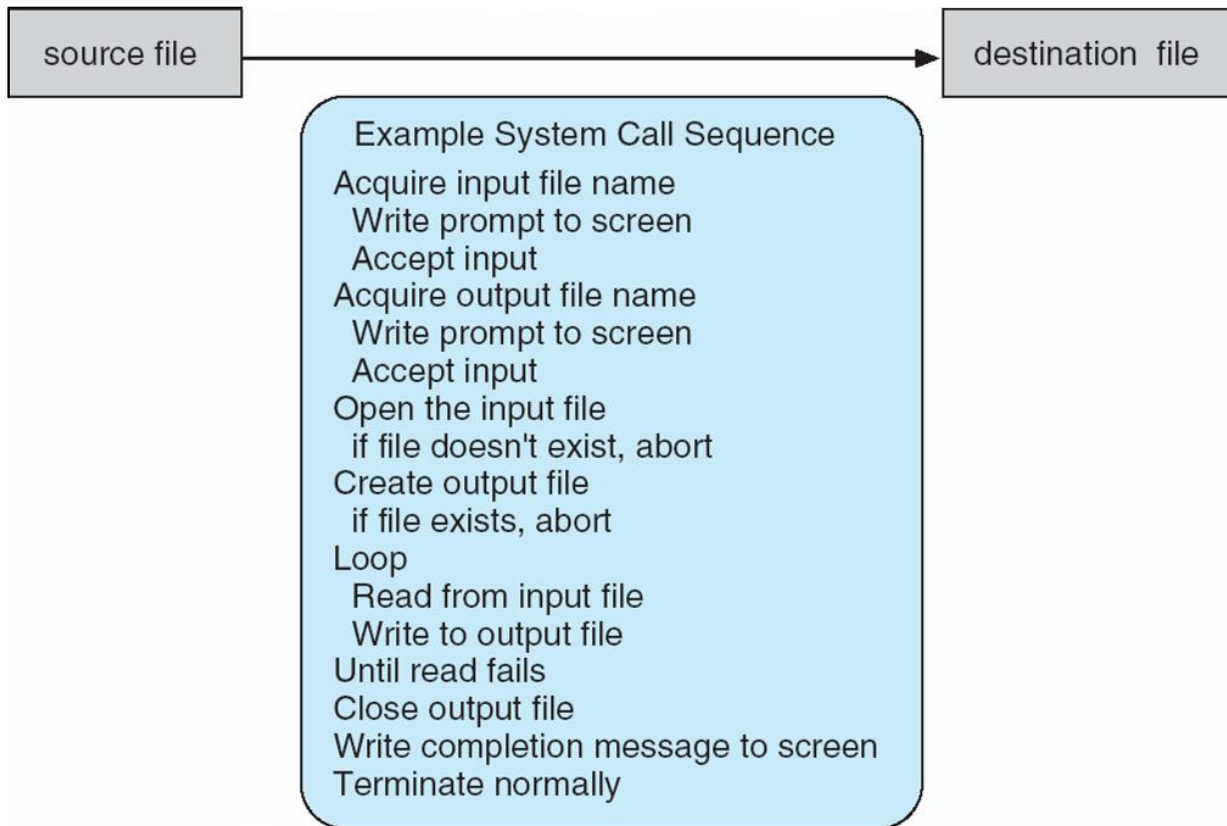
(Not: Yazı boyunca sistem çağrısı adı kullanılacaktır.)

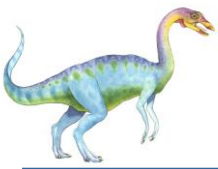




Sistem Çağrısı Örneği

- Bir dosyadan diğerine içerik kopyalamak için yapılan sistem çağrısı.





Standard API Örneği

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

return value

↓

```
BOOL    ReadFile c (HANDLE    file,  
                  LPVOID    buffer,  
                  DWORD    bytes To Read,  
                  LPDWORD    bytes Read,  
                  LPOVERLAPPED ovl);
```

↑

function name

parameters

- 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

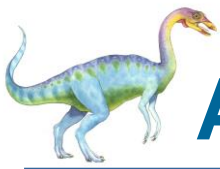




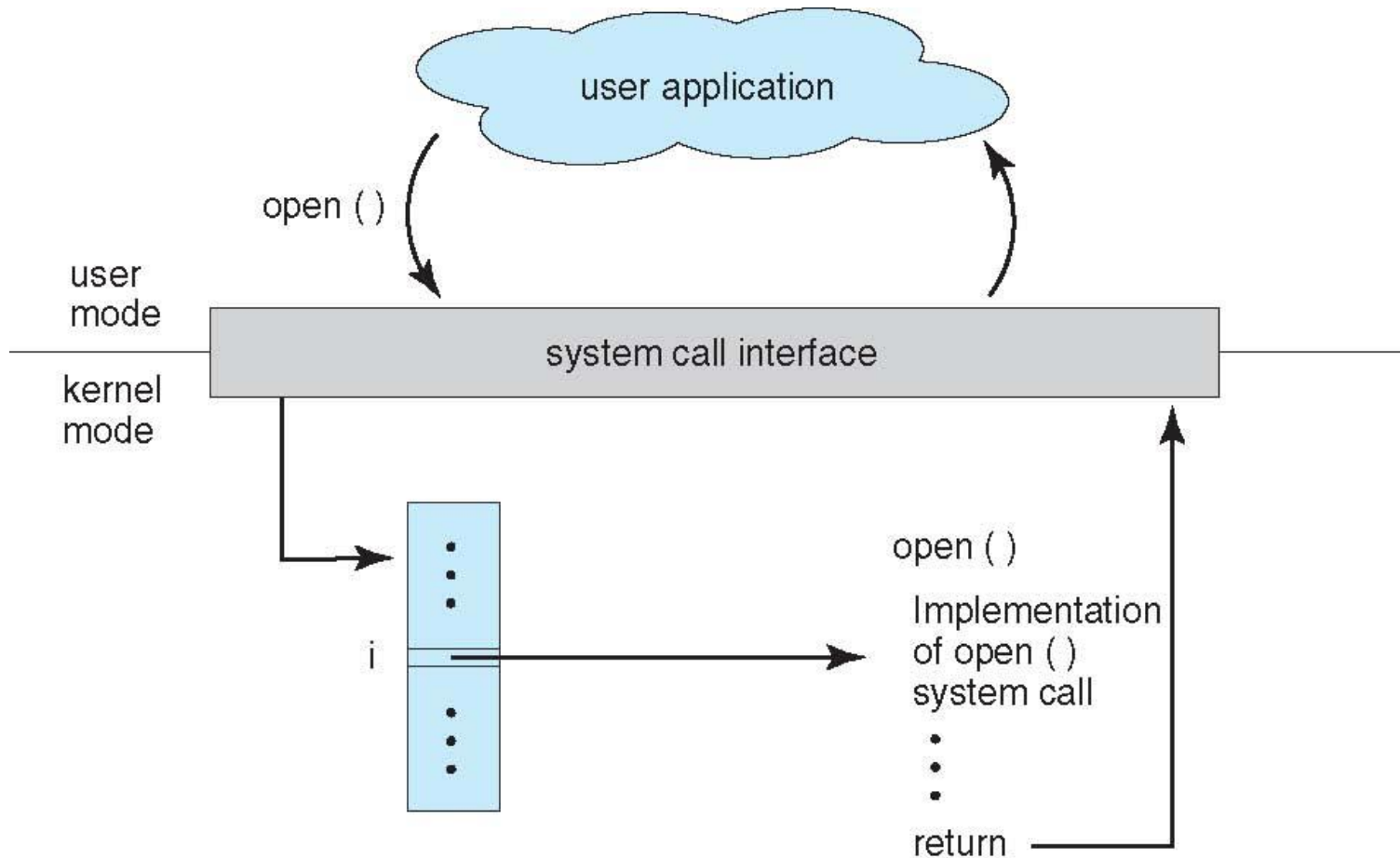
Sistem Çağrısı Uyarlaması

- 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 çağırır 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





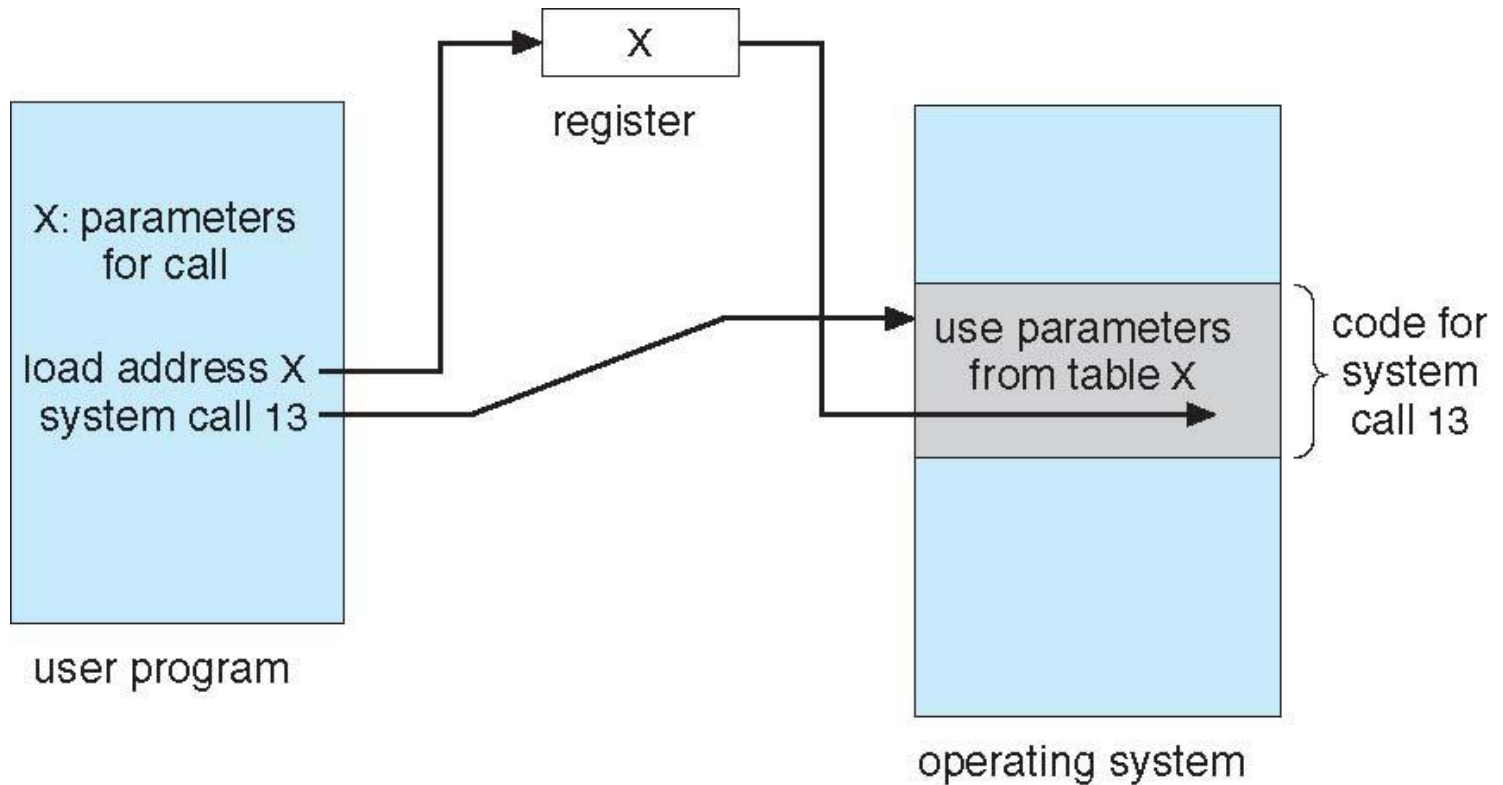
Sistem Çağrılarına Parametre Geçişi

- 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
- **Böcekleri** 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





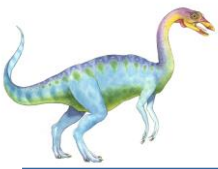
Windows ve Unix Sistem Çağrı Örnekleri

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

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

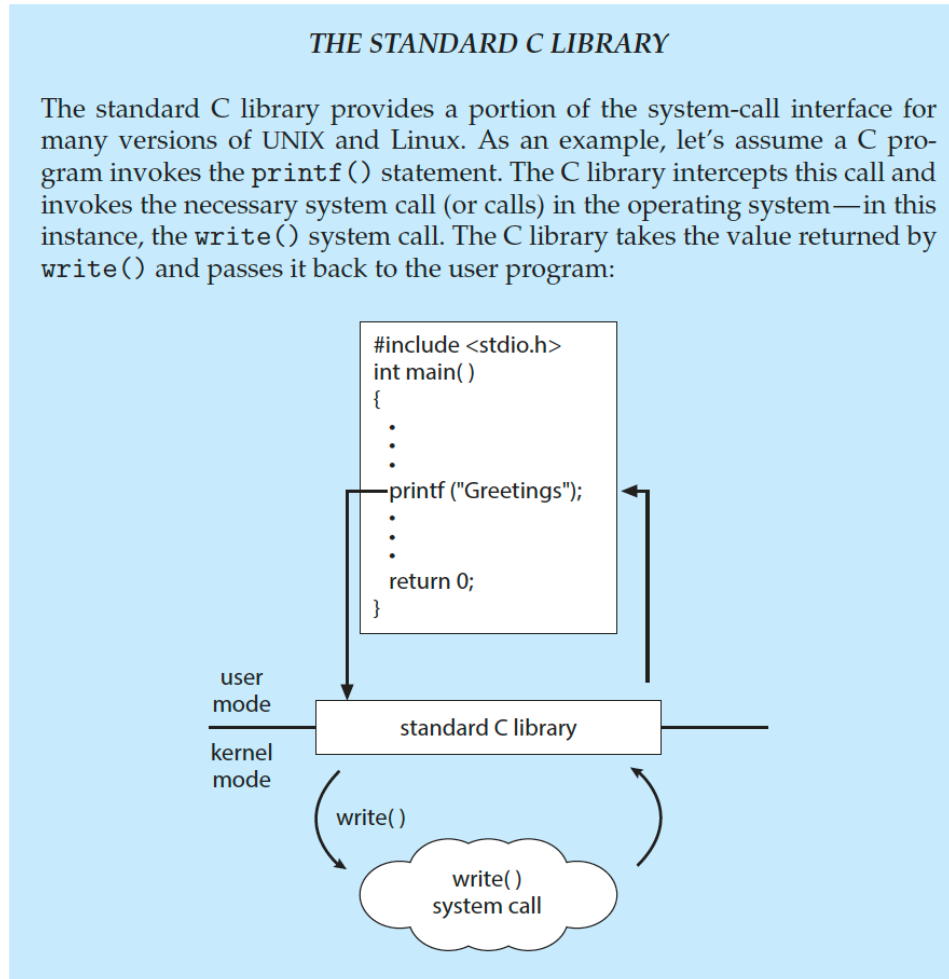
	Windows	Unix
Process control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information maintenance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





Standard C Kütüphanesi Örneği

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





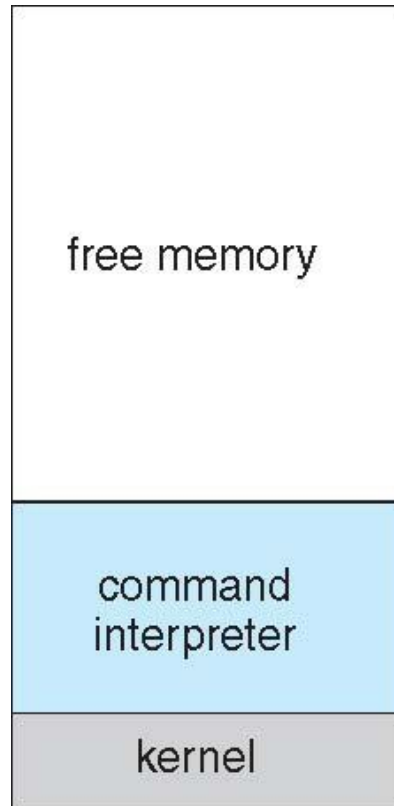
Örnek: MS-DOS

- 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

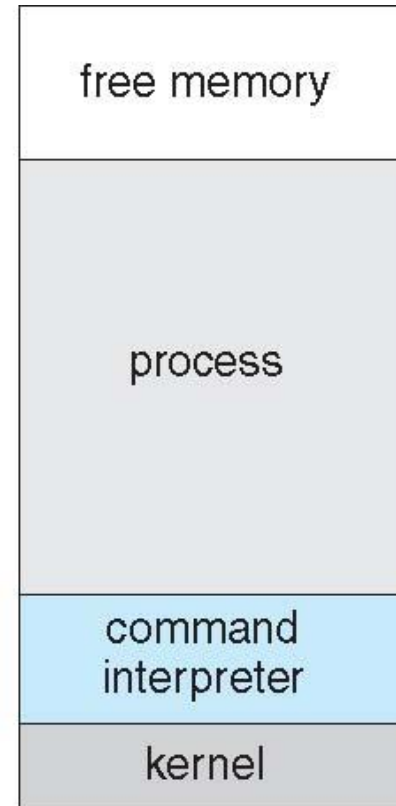




MS-DOS Yürütme



(a)



(b)

(a) Sistem ilk başlatıldığında

(b) Çalışmakta olan program

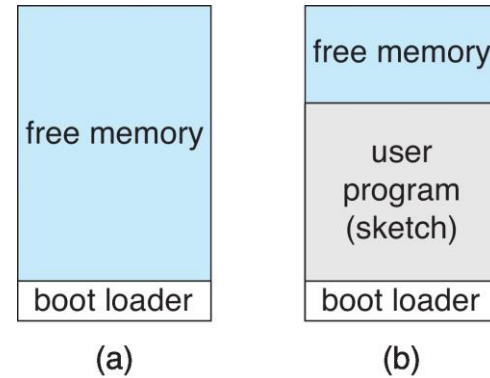




Örnek: Arduino

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



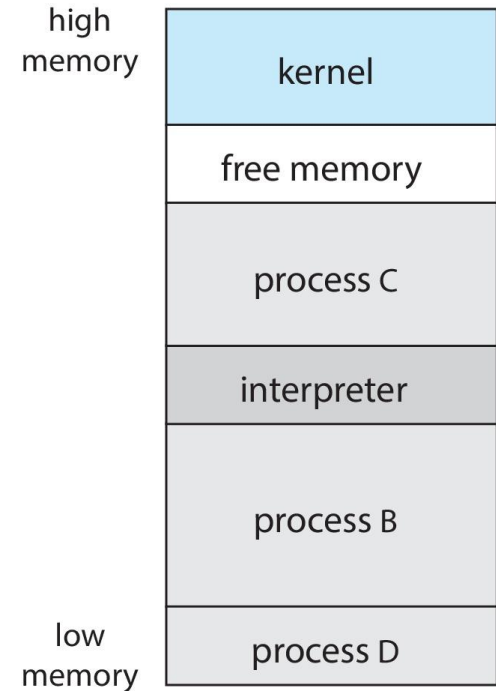
At system startup running a program





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 prosenin kullanıcı komutlarıyla sonlanmasını yada devam etmesinin bekler
- Prosesler
 - Kod = 0 - hatasız
 - Kod > 0 - hata koduİle sonlanır



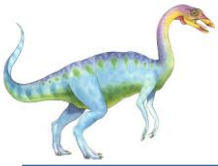


Sistem Servisleri

- Sistem programları program geliştirme ve çalıştırma için ortam sağlar. Sistem programlarını şu başlıklara ayırabiliriz:
 - Dosya yönetimi
 - Durum bilgisi
 - Programlama dili desteği
 - Program yükleme ve çalıştırma
 - İletişim
 - Arkaplan servisleri
 - Uygulama programları

- Kullanıcıların çoğu gerçek sistem çağrılarını 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

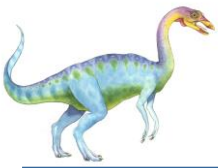




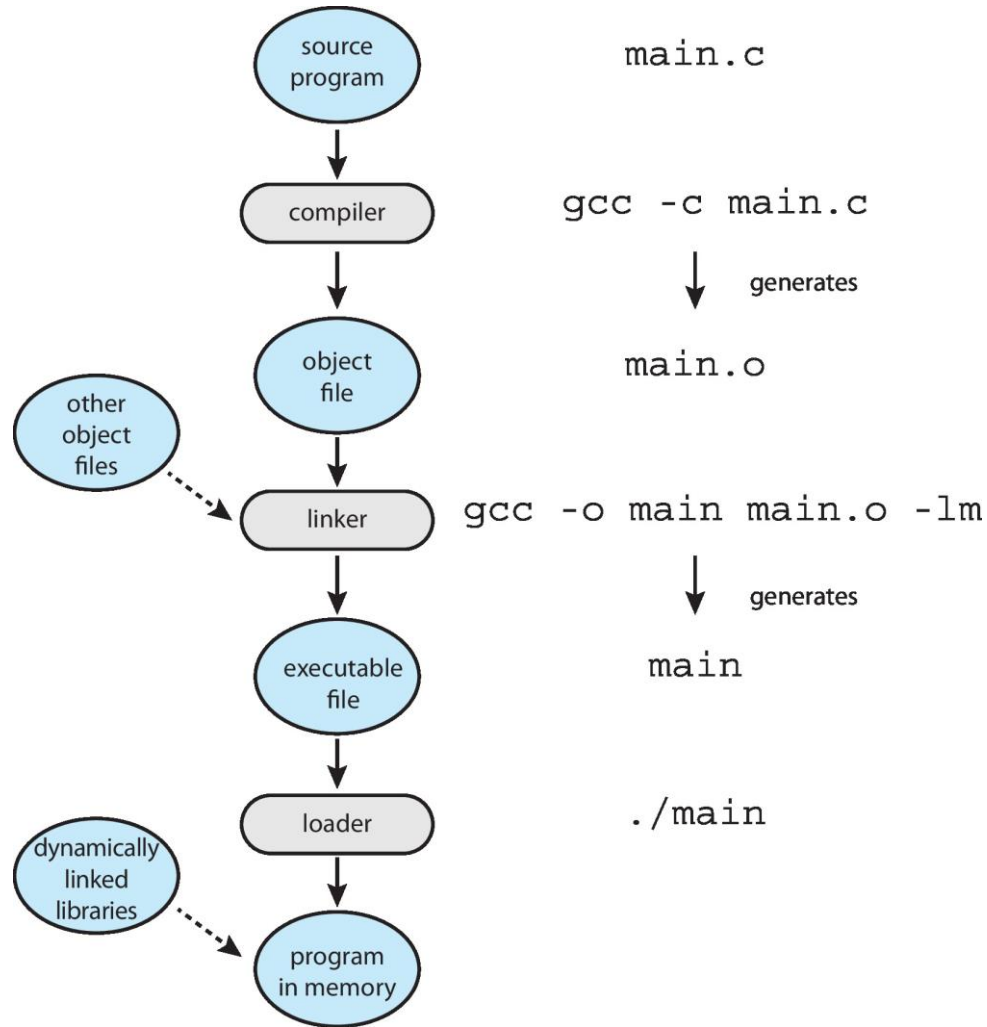
Bağlayıcılar (Linkers) ve Yükleyiciler (Loaders)

- 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 eşleştirecek 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 ve Yükleyicinin Rolü





Uygulamalar Neden İşletim Sistemine Özgü

- Bir sistemde derlenen uygulamalar genellikle diğer işletim sistemlerinde yürütülemez
- Her işletim sistemi kendi benzersiz sistem çağrılarına sahiptir
 - Kendi dosya formatları vb.
- Uygulamaların çoklu işletim sistemi olabilir
 - Python, Ruby gibi yorumlanmış bir dilde yazılmıştır ve birden fazla işletim sisteminde mevcut olan yorumlayıcı
 - Altyapı uygulamasını içeren bir VM desteğine sahip bir dilde yazılmış uygulama (Java gibi)
 - Standart bir dil kullanana (C gibi) uygulama, her işletim sisteminde ayrı ayrı derlenir ve her işletim sisteminde ayrı çalıştırılır
- **Uygulama İkili Arayüzü (ABI)**, API'nin mimari eşdeğeridir,
 - ikili kodun farklı bileşenlerinin belirli bir işletim sistemi için belirli bir mimari, CPU vb. üzerinde nasıl arayüz oluşturabileceğini tanımlar.



Bölüm 2'nin Sonu

