İşletim sistemi, bilgisayar kullanıcısı ile bilgisayar donanımı arasında aracı görevi gören bir programdır. İşletim sisteminin amaçları:

1- Kullanıcı programlarını çalıştırmak ve problemlere çözüm üretmek.
2- Bilgisayar sisteminin kullanımını kolaylaştırmak.
3- Bilgisayar donanımını verimli bir şekilde kullanmak.

İşletim sistemleri ne yapar?

- İşletim sistemleri **kaynak dağıtıcıdır**. (Verimli ve adil CPU kullanımı için çabalar)
- İşletim sistemi **kontrol programıdır**.
  Programların çalışma hatalarını ve yanlış kullanımlarını önlemek için programların yürütülmesini denetler.

Bir işletim sisteminde her zaman çalışan tek program **Kernel'dir.**

İşletim sistemi process management ile ilgili şunları yapmakla mükelleftir:

- Hem sistem hem kullanıcı taraflı process'leri (çalışmakta olan program) oluşturma ve silme.
- Process senkronizasyonu için mekanizmalar sağlamak.
- Kilitlenme (Deadlock) için mekanizamalar sağlamak. **Deadlock,** iki procces'in birbirini beklemesidir.

Bilgisayar sistemi 4 bileşenden meydana gelir.

1- Donanım (CPU (İşlemci) – Hafıza – I/O Aygıtları)
2- Kullanıcılar
3- Uygulama Programları
4- İşletim Sistemi

Bilgisayarın Başlatılması

- Bilgisayar ilk açıldığında ya da yeniden başlatıldığında **önyükleyici program (Bootstrap Program** (ROM veya EPROM'da tutulur)**)** çalıştırılır.
- Aynı HDD'ye birden fazla işletim sistemi kurulabilir. Hangisinin kullanılacağını bootstrap sorar.
- **Önyükleyici program,** işletim sistemini ve çekirdeğini **(Kernel)** çalıştırır. **Kernel'in** görevi sistem programlarını yükleyip çalıştırmaktır.

Bilgisayar Siteminin İşleyişi

- **Buffer,** verilerin geçici olarak depolandığı yerdir.
- **Aygıt yöneticisi,** I/O ile işletim sistemi arasında iletişimi sağlar, verileri taşımaktan **aygıt denetleyicisi** sorumludur bunu **aygıt sürücüsünü** kullanarak gerçekleştirir.

Bilgisayar Sisteminin İşleyişi

1. I/O işlemlerini başlatmak için sürücü, denetleyiciye uygun kayıtları yükler.
2. Aygıt denetleyicisi, önceliği belirlemek için kayıtları okur. Daha sonra aygıttan aygıtın buffer'ına verileri gönderir (mesela klavyeden bir harf girilmesi).
3. Veri aktarımı tamamlandıktan sonra denetleyici, sürücüye işlemin bittiğini bildirmek için bir interrupt atar. Aygıt sürücüsü daha sonra kontrolü OS'e geri verir.

Kesmelerin Ortak Fonksiyonları

- Kesmelerin kontrolü, kesme vektörü aracılığıyla **kesme servis rutinine** gönderilir.
- **Kesme vektöründe**, hangi servis rutininin hangi hafıza adresinde olduğu bilgisi bulunmaktadır.
- **Kesme mimarisi**, kesme talimatın adresini kaydetmelidir.
- İşletim sistemi, kesme eğilimlidir.
- İşletim sistemi, **yazmaçları (register)** ve **program sayaçlarını** hafızaya alarak CPU'nun durumunu muhafaza eder.

Depolama Yapısı

- Depolama yapısı, RAM ve disk (I/O aygıtıdır) olmak üzere ikiye ayrılır.
- Sabit disklerin yüzeyi **track'lere** bölünmüştür. Track'lerde **sector'lere** bölünür.
- **Yükleme talimatı**, bir baytı veya kelimeyi ana bellekten CPU içindeki dahili bir kayıt defterine taşırken, **store (depolama) talimatı** bir kaydın içeriğini ana belleğe taşır.
- Von Neuman mimarisine sahip bir bilgisayar aşağıdaki sıralı adımları gerçekleştirir:
  1- Program counter'ın gösterdiği adresten (bellekten) komutu getirilir.
  FETCH **(İşlemci ile hafıza arasında ya da 2 tane memory arasında gerçekleşir.) (RAM'den kaydedicilere gider.)**
  2- Kod çözülerek anlamlı hale getirilir. DECODE **(Kontrol birimi tarafından gerçekleştirilir.)**
  3- Komut neyse o çalıştırılır. EXECUTE **(Execute, ALU veya çıkış portlarında gerçekleştirilir.)**

Depolama Hiyeraşisi

- Depolama sisteminde: **"Hız – Maliyet – Kalıcılık"** önemlidir.
- **Önbellek (caching)** bilgileri daha hızlı depolama sistemine kopyalamayı sağlar.
  - Hızlı depolama aygıtında yani cache'de
    Veri varsa, bilgi cache'den direk kopyalanır (hızlı)
    Veri yoksa, bilgi cache'e kopyalanır ve oradan kullanılır.

I/O Yapısı

- **Kontrolör,** cihazdan yerel arabelleğe veri aktarımını başlatır. Veri aktarımı tamamlandıktan sonra, aygıt denetleyicisi, çalışmasının tamamlandığını bir **kesme** yoluyla aygıt sürücüsüne bildirir.

- En çok kullanılan veriler **önbellekte** tutulur.
- Chrome'un 11 alt başlığının olmasının sebebi yapılan işi küçük parçacıklara ayırarak daha hızlı çalışmasını sağlamak.
- Görev yöneticisine baktığımızda bir programda bütün CPU'ların seçili olmasının sebebi hangi CPU boşsa onda çalıştırması içindir.
- İşlemci ve RAM'i bütün programlar kullanır ama monitörü bütün programlar kullanmayabilir yani arka planda çalışan monitöre ihtiyaç duymayan bir program olabilir.
- Bir program için 8 haneli bir şifrenin olduğunu varsayalım ve crack yapacağız diyelim. Yine 8 haneli bildiğimiz bir şifreyle değiştirebiliriz. 8'den az ya da çok haneli bir şey yazacak olursak adresler kayacağı için ortaya problemler çıkar

MULTIPROGRAMMING (Çoklu Programlama)	MULTIPROCESSING (Çok İşlemcili Sistemler)
İşletim sisteminde aynı anda birden fazla programın çalıştırılmasıdır. (Hafızaya ne kadar program sığarsa o kadar program çalışır.) - Tek kullanıcı, CPU ve I/O cihazlarını her zaman meşgul edemez. - Multiprogramming, CPU'nun her zaman bir tane işlemi yürütmesini sağlar. - Bir iş seçilir ve iş zamanlayıcısı **(job scheduling)** yoluyla çalıştırılır.	Bir programın alt parçacıklara ayrılarak daha hızlı çalışmasıdır. Avantajları: 1- Artan İş Hacmi (Verim) İşlemci sayısını arttırarak, daha kısa sürede daha fazla iş yapmak hedeftir. 2- Ekonomiktir 3- Artan Güvenlik Multiprocessor'un iki türü vardır: **1- Asimetrik Çok İşlemcili Mimari** Her işlemci özel bir işe atanır. **2- Simetrik Çok İşlemcili Mimari** Her işlemci bütün işleri yapar.

Kesmelerin Ortak Fonksiyonları

Yazılım tarafından üretilen kesmeye **trap (tuzak)** veya **exception (istisna)** denir.

Şu durumlarda yazılımsal kesme yapılır:

1. Yazılım hataları (0'a bölme, mantık hatası yani program çalışır ama yanlış sonuç üretir.)
2. İşletim sistemi tarafından gelen istekler.
3. Sonsuz döngü içeren, işletim sistemini değiştirmeye çalışan programlar.

Çift Çekirdekli Tasarım

- Tek bir çip üzerinde birden fazla core bulunmasına **çok çekirdekli (multicore)** denir

Zamanlayıcı

- Sonsuz döngüyü - kaynakları tek başına tüketmeyi engellemek için **zamanlayıcı (timer)** vardır.

Kümelenmiş Sistemler

- **Kümelenmiş (Clustered) Sistemler,** çok işlemcili sitemlere (multiprocessors) benzer fakat birçok sistem birlikte çalışır.
- Genellikle depolama **storage-area network (SAN)** kullanılarak paylaştırılır.
- Yüksek kullanılabilirliğe **(high-availability)** sahip bir servis sağlar, arızaları atlatır.
  - **Asimetrik kümeleme (Asymmetric clustering)**
    Makinelerden sadece biri çalışır. Diğeri onu dinler. Bir sorun çıkarsa diğeri devreye girer.
  - **Simetrik kümeleme (Symmetric clustering)**
    İki makine de çalışır ve birbirini dinler.
- Bazı kümeleme sistemleri yüksek performans hesaplama **(high performance computing) (HPC)** için kullanılır. Uygulamalar paralelleştirmeyi **(parallelization)** kullanacak şekilde yazılmalıdır.
- Çakışmayı engelleme **(distributed lock manager (DLM))** paralel kümeleme sistemlerinde kullanılır. (**Diğer Bir Tanım:** Bazı sistemlerde çakışan işlemleri önler.) **Switch** adı verilen bir makine yardımıyla da makinelerden birinin sorun çıkarması anında diğerine veriler aktarılır.

İşletim Sistemi Mimarisi

- **Zaman paylaşımı (Timesharing),** kullanıcılar sistemin paylaşılmakta olduğunun farkında olmadan, her programla etkileşime girebilmesidir. 20 msn bir işlem 30 msn'de başka bir işlem çalışıyor yani bir işlemin tamamını yürütmek yerine birçok işi parça parça yapıyor.
  Multitasking (Çoklu Görev), birden fazla process'in aynı anda işletime alınabilmesidir.
  - **İşlemci zamanlaması (CPU scheduling)** CPU'da bir sonraki yürütme için mevcut işlemler arasından uygun işlemi seçer.
  - Eğer işlemler hafızaya sığmazsa, **değiş tokuş işlemi (swapping)** yaparak veriler hafızaya alınır ve çıkartılır.
  - **Sanal hafıza (Virtual memory)** RAM'de yer kalmayınca sabit diskin bir bölümünü RAM olarak kullanmaktır.
  - **Short time job scheduling** procces'ler arasında çok hızlı geçiş yapılmasını sağlar.

İşletim Sistemi İşlemleri

- **Çift modlu (Dual Mode)** işletim sistemini ve diğer sistem bileşenlerini korumayı sağlar.
  - **Mod Bit,** sistemin kullanıcı kodu ya da çekirdek kodunu çalıştırdığını ayırt etmekte kullanılır. Her program iki farklı modda çalışabilir. Bunlar;
    1- **User (Kullanıcı) Modu (mode bit=1)**
    User Mod kullanıcıya ait uygulamaların execute edildiği moddur.
    2- **Kernel (Çekirdek) Modu (mode bit=0)**
    Çekirdeğe ait driver'ların execute edildiği moddur.

I/O Alt Sistemi

- Tampon bellek işlemleri (buffering) – Verilerin geçici olarak depolandığı yerdir.
- Ön bellek işlemleri (caching) – Veriyi geçici olarak daha hızlı depolama birimine aktarmak.
- Kuyruklama (spooling) – Bir işin çıktısını diğer işin girdisi haline getirmek.

- Bellekte tutulan işlemler her zaman çalışan işlemlerdir. Program, hafızada (RAM'de) değilse disktedir.
- Procces'lerin iki tane zamanlayıcısı var:
  - CPU zamanı çalışır.
  - I/O zamanı çalışır.
- **DMA (Direct Memory Access – Çift Yönlü Veri Yolu),** I/O cihazıyla hafıza arasında haberleşme sağlar.
- **Debug** hata bulmak için ve crack yapmak için kullanılır.
- **Statik Link:** Örnek vermek gerekirse A programı her zaman 100 nolu hafızada çalışır.
- **Dinamik Link:** Program her çalıştığında farklı bir adreste çalışabilir.
- **Program:** Sonlu ve sıralı komutlardan oluşan programlama dili kullanarak yazılan yazılımlardır.
- Diskte yer alan her şey dosyadır. Bir dosyanın saklanabileceği diskteki en küçük birime **class** denir.
- **Blocking call,** mevcut sistem çağrısı sonlanana kadar başka işlem yapılmaz.
- **Non-blocking call,** İşlemler aynı anda paralel olarak geçekleştirilebilir.

Hafıza Yönetimi

Hafıza Yönetiminin Görevleri

- Şu anda hangi bellek bölümlerinin kullanıldığını ve kime ait olduğunu takip etme.
- Hangi işlemlerin ve verilerin hafızaya girip çıkacağına karar vermek.
- Bellek alanını gerektiğinde ayırma veya geri iade etme.

➢ **Depolama Birimlerinin Hızları**

❖ **Register > Cache > Main Memory > SSD (Solid State Disk) > Magnetic Disk**

Koruma ve Güvenlik

Koruma (Protection) – İçerden gelen saldırılara karşı korur.
Güvenlik (Security) – Dışarıdan gelen saldırılara karşı korur.

Bilgisayar Ortamları

Portallar dahili sistemlere web erişimi sağlar.
Network ortak iletişim yolu, **TCP/IP** en yaygın protokol

- ➢ Local Area Network (LAN)
- ➢ Wide Area Network (WAN)
- ➢ Metropolitan Area Network (MAN)
- ➢ Personal Area Network (PAN)

Hesaplama-sunucu sistemi istemcilere çeşitli servisler sağlayan bir arayüz sunar (örn. veritabanı)
Dosya sunucu sistemi istemcilere dosyaları kaydetmeyi ve indirmeyi sağlayan bir arayüz sunar.

Hesaplama Ortamları – Uçtan Uca Sistemler

Public Cloud – Ücretini yatıran herkes için **Private Cloud** – Şirket için
Hybrid Cloud – Genel ve özel kullanımlar için
Software as a Service (SaaS) – İnternet üzerinden kullanılabilen bir veya daha fazla uygulama
Platform as a Service (PaaS) – İnternet üzerinden uygulama kullanımı için hazır yazılım (veri tabanı sunucusu)
Infrastructure as a Service (IaaS) – İnternet üzerinden kullanılabilen sunucular veya depolama

Dosyanın 3 Farklı Hakkı Vardır:

1. Kullanıcı 2. Grup 3. Kullanıcı ve Grup Dışındakiler

- ➢ **Defragmentation,** RAM'deki dosyaların kaydırılarak tek bir hale getirilmesidir.
- ➢ **Yazılımsal RAID**; format atınca bilgiler gider.
  Donanımsal RAID; format atılsa bile bilgiler gitmez çünkü karttadır.
- ➢ **Logical Drive**, Windows Gezgini penceresindeki sürücü harfleriyle belirtilen bir depolama aygıtının bir bölümüdür. **Storage** (veriler storage de tutulur) tarafında RAID kullanılarak yapılır.

RAID 0 (Stripe Set)

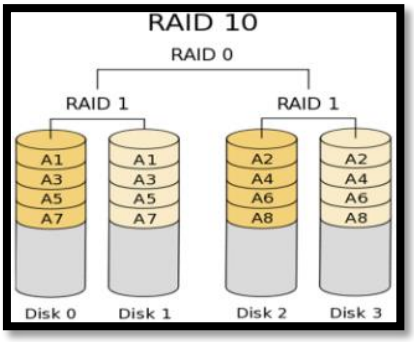
- Minimum 2, maximum 32 diskle kullanılır.
- Disklerden birisi arızalanır ise tüm bilgiler yok olur bu yüzden güvenilir değildir.
- 4 adet 300 GB disk ile toplam 1.2 TB alan tek sürücü altında kullanılabilir.
- **N tane disk varsa "N x Disk Boyutu" kadar alanımız vardır.**

RAID 1 (Mirror)

- Minimum 2 disk gereklidir.
- Veriler bir diske yazılır, kopyası birebir olarak diğer diske de yazılır. Dolayısıyla iki diskten birisi bozulsa dahi yinede verilere ulaşılabilir.
- 2 TB'lık **iki** disk var ise toplam disk alanı 2 TB olacaktır.
- **N tane disk varsa "N/2 x Disk Boyutu" kadar alanımız vardır.**

RAID 10

- Minimum 4, maximum 32 diskle kullanılır.
- 300 GB **dört** disk ile RAID 10 yapıldığında kapasite 600 GB olur.
- **N tane disk varsa "(N/2) x Disk Boyutu" kadar alanımız vardır.**
- Disk 2 – Disk 3 ya da Disk 0 – Disk 1'den herhangi ikisi bozulursa verilere ulaşamayız.

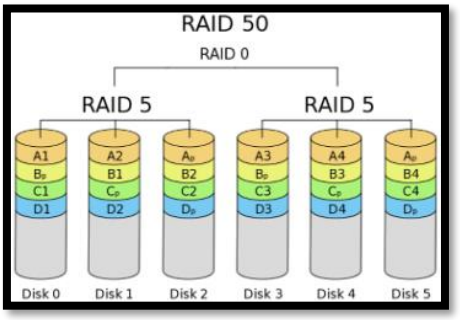


RAID 5

- Minimum 3, maximum 32 diskle kullanılır.
- Herhangi bir diske yazılan verinin yedeği diğer disk ya da disklerde bulunur, böylece veri kaybı olmaz. Bir disk fire (yanma) hakkı vardır.
- 300 GB 3 disk ile RAID 5 yapıldığında kapasite 600 GB olur.
- **N tane disk varsa "(N-1) x Disk Boyutu" kadar alanımız vardır.**

RAID 50

- Minimum 6, maximum 32 diskle kullanılır.



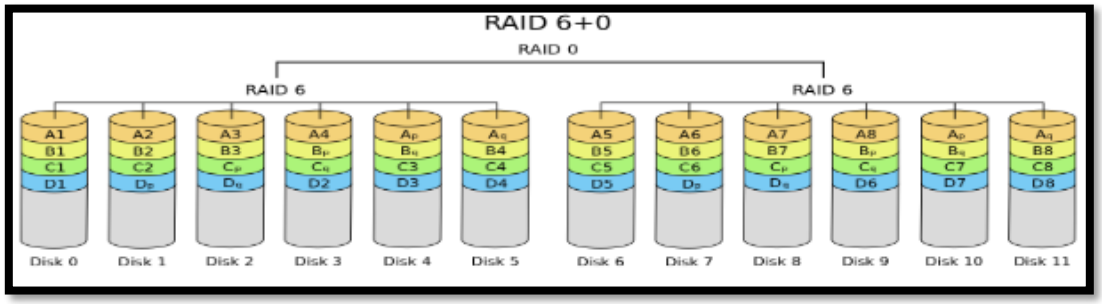
- Her biri 100 GB ise toplamda 400 GB kullanılabilir alan vardır.
- **N tane disk varsa "(N-2) x Disk Boyutu" kadar alanımız vardır.**

RAID 6

- Minimum 4 disk olmalıdır.
- İki disk fire (yanma) hakkı vardır.
- Hot Spare yoktur. **Hot Spare,** hazırda bekleyen ve disklerden birinin arızalanması durumunda bu diskin yerine geçecek diski ifade etmek için kullanılan terimdir.
- **N tane disk varsa "(N-2) x Disk Boyutu" kadar alanımız vardır.**

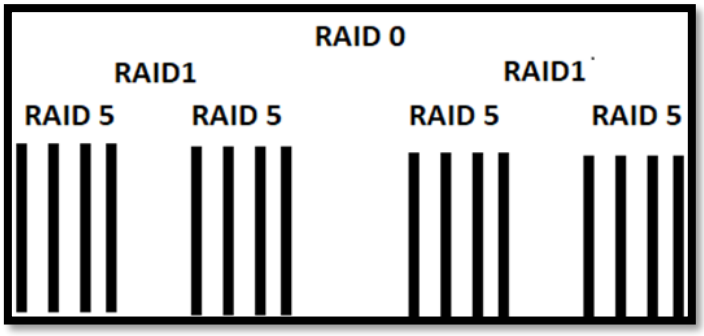
RAID 60

- Minimum 8 disk olmalıdır.
- 2 Gruptan toplamda 4 diske kadar disk arızasını ve kaybını tolere edebilir.

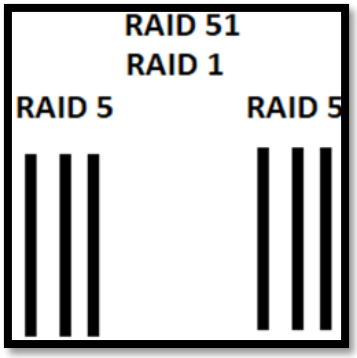


- Her biri 100 GB ise toplamda 800 GB kullanılabilir alan vardır.
- **N tane disk varsa "(N-2) x Disk Boyutu" kadar alanımız vardır.**

ÖRNEKLER:



- Her biri 100 GB ise toplamda 600 GB kullanılabilir alan vardır.



- Her biri 100 GB ise toplamda 400 GB kullanılabilir alan vardır.

NOT:

Hot swap, bozulan diskine yerine yenisinin takılmasıdır.

SAS (en güvenilir) – SATA – SSD (en güvensiz ama hızlı ve pahalı)

İşletim Sisteminin Kullanıcıları Nelerdir?

1. İnsanlar 2. Makineler 3. Diğer bilgisayarlar

Hepsini okuyup yazmayla satır satır okumanın arasındaki fark nedir?

RAM'i verimli kullanmak için satır satır okumak en iyisidir.

Windows'da CMD'den farklı olarak **PowerShell** vardır

Yorumlayıcı (Interpreter)	Derleyici (Compiler)
Herhangi bir hata olana kadar programı çalıştırır. İlk hata gördüğü yerde durur.	Hatayı bütün kodu tamamladıktan sonra gösterir.
Kaynak kodu analiz etmekle zaman harcamaz. Ancak genel yürütme süresi daha yavaştır.	Kaynak kodun analizi için büyük zaman harcar. Ancak genel yürütme süresi daha hızlıdır.
Programı satır satır işler.	Programı bir bütün olarak işler
Python, Java	C, C++

İşletim Sistemi (Hizmetleri) Şunları Sağlar:

1- Kullanıcıya yardımcı olan kullanıcı arabirimleri (UI) vardır.

a. **Komut Satırı (CLI)**
   Kullanıcıdan bir komut alır ve çalıştırır. Dosya oluşturma, silme, listeleme vb.
   Birden çok komut yorumlayıcısı olan sistemlerde komut yorumlayıcı **Shell (kabuk)** olarak bilinir. UNIX ve Linux sistemlerinde; C Shell, Bourne Shell, Korn Shell bunlardan birkaçıdır.

b. **Grafik Kullanıcı Arabirimi (GUI)**
   Grafiksel arayüz sağlar. Kullanıcıya görsel olarak I/O sağlanır. Fare-klavye-monitör kullanılır.

c. **Batch Arabirimi**
   Komutar dosyalara girilir ve bu dosyalar çalıştırılabilir dosyalardır

2- Program Yürütme

3- I/O İşlemleri

Kullanıcı doğrudan I/O'ya erişemediği için OS bunu sağlamalıdır.

4- Dosya Sistemi Değişiklikleri

Dosya oluşturma, güncelleme, silme, bilgi listelemesi, yetkilendirme işlemlerini yapar.

5- İletişim

Process'ler arası ya da başka bilgisayarların process'leriyle iletişim sağlamalıdır.

6- Hata Tespiti

7- Kaynak Paylaşımı (Resource Allocation)

Kullanıcılar ve process'ler arasında kaynakları paylaştırmaktır.

8- Kayıt Tutma (Accounting):

Hangi kullanıcının hangi sistem kaynağını ne kadar kullandığını kaydeder. Bu kayıt tutma, istatistikler için toplanabilir. Kayıt tutma sayesinde sistemi hackleyeni bulabiliriz.

9- Koruma ve Güvenlik

Sistem Çağrıları

- **API,** iki yazılım bileşeninin birbiriyle iletişim kurmasına olanak tanıyan mekanizmadır. Örneğin, meteoroloji müdürlüğünün yazılım sistemi, günlük hava durumu verilerini içerir. Telefonunuzdaki hava durumu uygulaması, API'ler aracılığıyla bu sistemle "konuşur" ve telefonunuzda size günlük hava durumu güncellemelerini gösterir.
- **Sistem çağrıları,** OS tarafından sağlanan hizmetlere bir **arayüz (API)** sağlar.
  En yaygın üç API:
  1) **Windows için WinAPI**
  2) **Unix için POSIX**
  3) **Java için ise JAVA API**
- Programcı, işletim sistemi tarafından sağlanan kütüphaneler aracılığıyla API'ye erişir.
- C dilinde yazılmış, UNIX ve Linux sistemlerinde kullanılan kütüphane dosyalarına **libc** adı verilir.

Sistem Çağrı Arayüzü

➢ **API'deki işlev çağrılarını yakalar**, gerekli sistem çağrılarını çağırır. Sistem çağrısının durumunu, varsa geri dönüş değerini döndürür ve bunları indexlenmiş bir tabloda tutar.

Sistem Çağrısı Parametreleri

Parametreleri işletim sistemine geçirmek için üç yöntem vardır:

1) Parametreleri **register** kullanarak göndermek.
2) Parametreler hafızada bir blokta tutulur ve blokun adresi register ile gönderilir.
3) Parametreler yığına (stack) atılır push ve pop işlemleri ile işletim sistemi tarafından çekilir.

Sistem Çağrı Türleri

1. **İşlem Kontrolü**
   Çalışan bir programın durdurulması normal (end()) veya anormal (abort()) yolla olabilir. Halihazırda çalışan bir program beklenmeyen bir hata ile karşılaşırsa hataya sebep olan **bellek dökümü** alınır **(dump of core/core dump),** hata mesajı alınır ve diske yazılır.
2. **Dosya Yönetimi**
3. **Cihaz yönetimi**
4. **Durum bilgisi**
   Çoğu sistemde saati time() ile tarihi ise date() ile döndürmek için bir sistem çağrısı vardır.
5. **İletişim**
   Süreçler arası iletişimin iki yaygın modeli vardır; **mesaj iletme modeli** ve **paylaşılan bellek modeli**. Mesaj iletme modelinde, işlemler bilgi aktarmak için birbirleriyle mesaj alışverişinde bulunur.
   Alıcı işlemi, genellikle bir kabul bağlantısı **accept connection ()** çağrısı ile iletişimin gerçekleşmesi için izin vermelidir. İşlemler **wait for connection()** çağrısı yürütürler ve bir bağlantı kurulduğunda uyandırılırlar.
6. **Koruma (Protection)**

Sistem Programları

Program geliştirme ve yürütme için ortam sağlar. Aşağıdaki kategorilere ayrılabilirler:

1. **Dosya İşleme**
2. **Durum Bilgileri**
   Bazıları sistemden bilgi ister - tarih, saat, RAM, disk alanı, kullanıcı sayısı vb.
3. **Programlama Dili Desteği**
   Dil compiler'ları ya OS ile sağlanır ya da indirilebilir.
4. **Program Yükleme ve Yürütme**
   Derlenmiş dili RAM'e load eder ve execute eder.
5. **İletişim**
   Başka PC'lere veri aktarır, mail atar, web'de gezinir.
6. **Arka Plan Hizmetleri**
   Belirli sistem programları çalıştırmak için başlangıçta çalışırlar. Ya hemen son bulurlar ya da PC kapanınca sona erer. Sürekli çalışan sistem program süreçleri; hizmetler, alt sistemler veya arka plan programları olarak bilinir.
7. **Uygulama Programları**
   Kullanıcı tarafından çalıştırılan, kullanıcının kullandığı uygulamalardır.

İşletim Sistemi Tasarımı

- Sistem gereksinimleri, **kullanıcı (user)** ve **sistem (system)** hedefleri olmak üzere iki gruba ayrılır:
  Kullanıcı Hedefleri: İşletim sistemi kullanımı kolay, öğrenmesi kolay, güvenli ve hızlı olmalıdır.
  Sistem Hedefleri: İşletim sistemi, esnek, güvenilir, hatasız ve verimli olmasının yanı sıra tasarlanması, uygulanması ve bakımı kolay olmalıdır.
- İşletim sistemi tasarımında **ilkeler (politika)** ve **mekanizmayı** birbirinden ayırmak önemli bir prensiptir.
  Politikalar ne yapılacağına karar verirken, mekanizmalar bir şeyin nasıl yapılacağını belirler.

İşletim Sistemi Yapısı

1. **Basit Yapı – MS-DOS**
   MS-DOS en az alanda en fazla işlevselliği sağlamak için yazılmıştır.
   Aynı anda sadece tek program çalışır. I/O aygıtlarına direk erişir.
   Kötü yazılımlara karşı savunmasız bir yapıdadır.
2. **Daha Karmaşık Yapı – UNIX**
   UNIX, iki bölümden oluşur; *çekirdek* ve *sistem programları*.
3. **Katmanlı Yapı**
   İşletim sisteminin, en alt katmanı donanımdır, en yüksek ise kullanıcı ara yüzüdür.
   Alttaki katman üstteki katmana emir veremez, erişemez ancak üstteki katman alttaki katmana hem erişir hem de emir verebilir.
4. **Microkernel Yapı – Mach**
   Bu yöntem, gerekli olmayan tüm bileşenleri çekirdekten kaldırır.
   Mikroçekirdeğin ana işlevi, istemci programı ile kullanıcı alanında da çalışan çeşitli hizmetler arasında iletişim sağlamaktır.

Modüller
İşletim sistemindeki en güncel metodoloji, **yüklenebilir çekirdek (Kernel) modülleridir.**
Kernel'e program bağlamaktır diyebiliriz. Bu sayede ek hizmetlere sahip olur.

Hibrit Sistemler
Çok az işletim sistemi bir sistem kullanır, bunun yerine OS'nin belli bölümleri belli sistemlerini kullanır.
Örneğin, hem Linux hem de Solaris monolitiktir (tek parça), çünkü işletim sisteminin tek bir adres alanında olması çok verimli performans sağlar.

MacOS Yapısı
MacOS **hibrit** bir yapı kullanır. **Aqua,** arayüz sağlar.
Cocoa, Mac OS uygulamaları yazmak için kullanılan Objective-C için bir API'dır.
Mach; hafıza yönetimi, uzaktan çağrı (RPC) ve process'ler arası iletişim sağlar.
BSD ise komut satırı arabirimi, **Pthreads,** ağ ve dosya sistemleri

IOS
IOS, iPhone , iPad için Apple mobil işletim sistemidir.
Uygulama geliştirmek için Cocoa Touch Objective-C API
Temel (core) hizmetler bulut bilişim, veri tabanları sağlar.

Android
Açık kaynak kodludur.
Linux çekirdeğinden değiştirilmiştir.
Dalvik sanal makinesini içerir.

İşletim Sistemi Hata Ayıklama
Darboğaz, bilgisayar da uyumsuz parçaların yaptığı performans düşüklüğüdür.
Hata Analizi, bir işlem başarısız olursa, kullanıcıları sorunun oluştuğu konusunda uyarmak için hata bilgilerini bir günlük dosyasına yazar.
Çekirdekteki bir arızaya **çökme (crash)** denir.

DTrace
DTrace, işletim sistemi veya uygulamaların davranışlarını veya oluşturdukları hataların sebeplerini anlamak için bir ortam sunan bir framework.

Sistem Boot
Önyükleme programı (bootstrap) makinenin durumunu belirlemek için **diagnostic (sesli uyarı mesajını)** çalıştırmaktır.
Cep telefonları, tabletler vs. **işletim sistemini ROM'da depolar**.
Büyük işletim sistemleri için (Windows, MacOS ve UNIX) veya sık sık değişen sistemler için, önyükleme yükleyicisi **firmwarede** depolanır ve işletim sistemi disktedir.
GRUB, Linux sistemlerindeki ön yükleyici programdır yani bootstrap'dır.
Önyükleme bölümü olan bir diske **önyükleme diski** veya **sistem diski** denir.

Bir program kaç farklı şekilde çalıştırılabilir?
Fareyle tıklayarak, komut satırına yazarak, batch file olarak, Schedule olarak.
4 TEMEL HAFIZA BİRİMİ
Text ve **data**'nın hafızadaki yeri değişmez, **stack** ve **heap**'in hafızadaki yeri değişebilir.

İşlem Kavramı
İşlem, yürütülmekte olan program olarak düşünülebilir.
İşlemlerin birçok parçası vardır:

- Program Kodu (Text Section)
  Kodlar tutulur.
- Program Sayacı (Program Counter)
  Yürütülmekte olan program komutlarının **adres bilgisini** tuttuğu bir **kaydedicidir**.
- Yığın (Stack)
  Geçici verileri tutar. (Fonksiyon adresleri, geri gönüş adresleri, yerel değişkenler)
- Veri Bölümü
  Global değişkenleri içerir.
- Heap çalışma süresi boyunca tahsis edilen dinamik belleği içerir (bağlı liste).

Program diskte depolanan pasif varlıktır (yürütülebilir dosya) iken **işlem** aktif bir varlıktır.
Çalıştırılabilir dosya (program) belleğe yüklendiğinde işlem olur.
Java sanal makinesi (JVM) içinde yürütülebilir bir Java programı yürütülür.

İşlem Durumları
Bir işlem çalıştırılırken durumunu değiştirir ve aşağıdaki durumlardan herhangi birinde olabilir:
Yeni (new): İşlem oluşturuluyor
Çalışıyor (running): İşlem komutları çalıştırılıyor
Bekliyor (waiting): İşlem bir olayın gerçekleşmesini bekliyor
Hazır (ready): İşlem bir işlemciye atanmaya bekliyor
Sonlandırılmış (terminated): İşlem çalışmayı bitirmiş

İşlem Zamanlama
Multiprogramming'in amacı, CPU kullanımını en üst düzeye çıkarmak için bazı işlemlerin her zaman çalışmasını sağlamaktır.
İşlem zamanlayıcısı CPU'da bir sonraki yürütme için mevcut işlemler arasından uygun işlemi seçer.
İşlemler zamanlama kuyrukları (scheduling queues) ile yönetilir

- İş kuyruğu (Job Queue), sistemdeki tüm işlemlerin kümesidir.
- Hazır kuyruğu (Ready Queue), çalışmaya hazır veya çalıştırılmayı bekleyen işlemler.
- Aygıt kuyrukları (Device Queues), bir I/O cihazını kullanmayı bekleyen işlemler kümesidir.

Zamanlayıcılar

İşletim sistemlerinde üç tip zamanlayıcı bulunmaktadır.

- Uzun vadeli zamanlayıcılar (long-term scheduler)
  Uzun vadeli zamanlayıcı hangi işlemlerin iş kuyruğundan alınıp ***hazır kuyruğa*** getirilmesi gerektiğini seçer.
- Orta vadeli zamanlayıcılar (medium-term scheduler)
  Takas (swap) yaparak hafızadan hangi işlemin kaldırılacağını belirler.
- Kısa vadeli zamanlayıcılar (short-term scheduler)
  Kısa süreli zamanlayıcının amacı, yürütmeye hazır "Hazır Kuyruğun" 'dan işlemi seçip ***yürütmek*** için CPU'yu tahsis etmektir.

Ortam Değişikliği

Çalışan bir process'in interrupt edilip anlık durumunun PCB'ye yazılmasına **ortam değişikliği** denir.

Context Switching'in Hızı;

- Bellek hızına
- Kopyalanacak verinin boyutuna
- Özel talimatların varlığına bağlı olarak makineden makineye değişir.

İşlem Oluşturma

Pid, sistemdeki her işlem için benzersiz bir değerdir ve çekirdek içindeki bir işlemin çeşitli özelliklerine erişmek için bir indeks olarak kullanılabilir. Her zaman pid değeri 1 olan bir **init (ata)** process vardır, bütün process'ler bunun child'ıdır.

UNIX'te process'ler **fork()** komutu ile oluşturulur ve her yeni process bir ebeveyn process'inin kopyasıdır. Daha sonra **exec()** komutu ile process'in içi silinir.

İşlem Sonlandırma

Bir program son kodunu işlemeyi bitirdiğinde **exit()** sistem çağrısı kullanılarak process sonlanır.
Bu noktada process bir **wait()** sistem çağrısı ile parent'ını bir durum geriye döndürebilir.
Ebeveyn işlem, **abort()** kullanarak alt süreçleri sonlandırabilir. Bunu yapmasının bazı nedenleri:

- Çocuklar kendisine verilen kaynakları aşmıştır.
- Çocuk işleme atan göreve artık ihtiyaç yoktur yani çocuğa ihtiyaç kalmamıştır.
- Eğer parent yani ana işlem sonlanırsa bazı OS'leri çocuk işlemlerin çalışmasına izin vermez.

pid = wait(&status) Böylece sonlanan işlemin pid'ine sahip olduk.
İşlemini bitirmiş bir process, wait() sistem çağrısı gelene kadar **zombi process'tir.**

Çoklu İşlem Mimarisi - Chrome Tarayıcı

- **Renderer,** sitelerdeki CSS, HTML, JS.. dosyaları render eder.
- **Tarayıcı işlemi,** kullanıcı arayüzünü, diski ve ağı yönetir.
- **Plug-in işlemi,** her türlü plug-in için kullanılır.

İşlemler Arası İletişim

Bir işlem, sistemde yürütülen diğer işlemleri etkileyemiyor veya etkilenemiyorsa **bağımsızdır.**
Bir işlem, sistemde yürütülen diğer işlemleri etkileyebiliyorsa veya onlardan etkilenebiliyorsa **işbirliği yapıyor** demektir.
İşlemler arası iletişimin nedenleri:

- **Bilgi paylaşımı**
- **Hesaplama Hızlandırma:** Belirli bir görevin daha hızlı çalışmasını istiyorsak.
- **Modülerlik**
- **Kolaylık**

İşlemler arası iletişimin iki temel modeli vardır:

- **Shared Memory,** iletişim yapan process'ler arasında ortak bir bellek bölgesi oluşturur. Diğer process'ler bu alana okuma/yazma yapabilir.
- **Message Passing,** iletişim kuran process'ler arasında karşılıklı mesaj gönderilip alınmasıdır.

NOT: Shared memory, message passing'den daha hızlıdır. Message passing daha az miktarda veri geçişi için kullanılır.

Üretici-Tüketici Problem

Üretici-Tüketici problemi kaynakların işlemler tarafından kullanılması ile ilgilidir.
İki çeşit arabellek kullanılabilir:

- **Sınırsız arabellek,** arabelleğin boyutunda bir sınır yoktur.
- **Sınırlı arabellek,** sabit bir arabellek boyutu olduğu varsayılır.

İşlemler Arası İletişim

Doğrudan İletişim

Süreçler birbirleriyle iletişim kurmak için doğrudan veya dolaylı iletişimi kullanabilirler. Doğrudan iletişim altında, iletişim kurmak isteyen her işlemde, iletişimin alıcısı ve göndericisi birbirlerinin kimliklerini bilmelidir.
Mesaj gönderimi iki şekilde gerçekleşir:

1- **receive (Q, message) --> Q işleminden mesaj alır.**
2- **send (P, message) ------> P işlemine mesaj gönderir.**

Dolaylı İletişim

Dolaylı iletişimde, mesajlar posta kutularına veya bağlantı noktalarına gönderilir ve buradan alınır. Bir posta kutusu soyut olarak, işlemler tarafından mesajların yerleştirilebileceği ve mesajların kaldırılabileceği bir nesne olarak görülebilir. Her posta kutusunun benzersiz bir kimliği vardır.

1- **send (A, message) ------> A posta kutusuna mesaj gönderir.**
2- **receive (A, message) ---> A posta kutusundan mesaj alır.**

Dolaylı iletişim bağlantısının özellikleri

- Bağlantı yalnızca işlemler ortak bir posta kutusunu paylaşıyorsa kurulur.
- Bir bağlantı noktası birçok işlemle ilişkilendirilebilir.
- Her işlem çifti birkaç iletişim bağlantısını paylaşabilir.

Senkronizasyon

Bloklama yapılan mesaj **senkrondur:**

- **Send Blocking:** Mesaj gönderdikten sonra, alıcı alana kadar gönderen taraf bloklanır.
- **Receive Blocking:** Gönderici mesaj göndermeden alıcı alamaz.

Bloklanmayan mesaj **asenkrondur:**

- **Nonblocking Send:** Mesaj gönderdikten sonra mesajın alınmasını beklemeden işlemine devam eder.
- **Nonblocking Receive:** Gönderici mesajı göndermeden alıcı, işlemine NULL olarak devam edebilir.

Buffer Kullanımı

İletişim süreçleri tarafından değiştirilen mesajlar geçici bir kuyrukta bulunur. Üç çeşidi vardır:

- **Sıfır Kapasite:** Sırada bekleyen herhangi bir mesaj yoktur. Gönderici, mesaj alınana kadar bloklanmalıdır.
- **Sınırlı Kapasite:** Sırada bekleyen n sayıda mesaj olabilir. Gönderici, bu sınıra ulaştıktan sonra bloklanmalıdır.
- **Sınırsız Kapasite:** Sırada bekleyen sınırsız mesaj olabilir. Gönderici asla bloklanmaz.

İstemci-Sunucu Sistemlerinde İletişim

- Soketler (Sockets)
- Uzaktan Prosedür Çağrıları (Remote Procedure Calls)
- Borular (Pipes)
- Uzaktan Yöntem Çağrısı (Java)

Soketler

- **Soket,** iletişim için bir uç nokta olarak tanımlanır.
- Genel olarak, soketler bir istemci-sunucu mimarisi kullanır. **Sunucu,** belirli bir bağlantı noktasını dinleyerek gelen istemci isteklerini bekler. Bir istek alındığında, sunucu, bağlantıyı tamamlamak için istemci soketinden bir bağlantı kabul eder.

Java'da Üç Tip Socket Vardır

1. Bağlantı Odaklı (TCP)
2. Bağlantısız (UDP)
3. Multicast Sınıfı
   Veriler birden çok alıcıya gönderilebilir.

Uzak Prosedür Çağrıları (Remote Procedure Call)

- Ağa bağlı bilgisayarlar arasında "message passing" temelli iletişim sağlar.
- Bir makinenin, ağdaki diğer makinedeki fonksiyonu uzaktan çağırıp çalıştırması ve çalışan karşıdaki makinede çalışan fonksiyonun sonucunun ayrı bir mesajla kendisine dönmesidir.

Borular

- İki process'in (sürecin) iletişim kurmasına izin veren bir kanal görevi görür.

Sıradan Borular

- Sıradan Borular, standart üretici-tüketici tarzında iletişime izin verir.
- Üretici bir uca yazar, tüketici diğer uçtan okur. Sıradan borular tek yönlüdür.

İsimli Borular

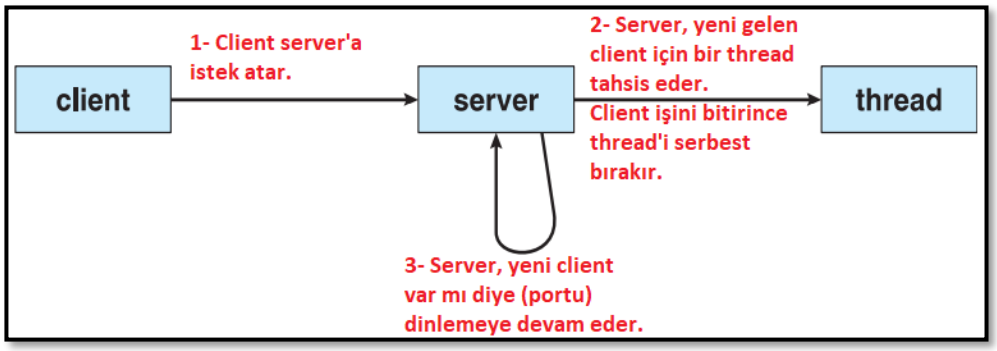
- İsimli Borular sıradan borulardan daha güçlüdür.
- İletişim çift yönlüdür.

1. Buffer **FIFO** tasarımıyla çalışır.
2. **Bir exe'yi neden servis olarak çalıştırırız?**
3. Sürekli çalışması için.
4. **Paket** header ve data olmak üzere 2 kısımdan oluşur.
5. **Üç yollu el sıkışma,** TCP / IP ağında yerel bir ana bilgisayar / istemci ve sunucu arasında bağlantı oluşturmak için kullanılan bir yöntemdir. Gerçek veri iletişimi başlamadan önce hem istemcinin hem de sunucunun SYN ve ACK (alındı) paketlerini değiştirmesini gerektiren üç adımlı bir yöntemdir.

	TCP	**UDP**
Anlam	TCP, verileri iletmeden önce bilgisayarlar arasında bağlantı kurar	UDP, sistemin almaya hazır olup olmadığını kontrol etmeden verileri doğrudan hedef bilgisayara gönderir
Hız	Yavaş	Hızlı
Güvenilirlik	Son derece güvenilir	Güvenilmez
Alındı	Verilerin onayını alır ve kullanıcı isterse tekrar iletme yeteneğine sahiptir.	Ne onay alır ne de kayıp verileri iletmez.

İş Parçacıkları (Threads)

- Bir **process'in (çalışan program)** birden fazla işi aynı anda yapmasını sağlayan yapılara **thread** denir.
- Bir process birden fazla thread'e sahipse aynı anda birden fazla iş gerçekleştirebilir.
- Threadler, ***program kodunu (code), data*** *ve* ***file'ı*** *(dosyaları)* ortak kullanır.
- Birden fazla thread'le programlamaya **multithreading** denir.



Multithreading (Çok İş Parçacıklı Programlamanın) Faydaları

Cevap Verilebilirlik (Responsiveness):
Kaynak Paylaşımı (Resource Sharing)
Maliyet
Ölçeklenebilirlik

- Her thread'in kendine özgü register, yığın, program sayacı ve işlem durumu vardır.
- Tek çekirdeğe sahip bir sistemde aynı anda yalnız bir thread çalıştırılabilir, diğerleri kuyruğa alınır.

Mutlticore Programming

Çok iş parçacıklı uygulamaların zorlukları:

1- **Görevleri Belirleme (Identify Tasks)**
   Farklı thread'lere bölünebilecek parçaları tespit etme.
2- **Denge (Balance)**
3- **Verileri Bölmek (Data Splitting)**
4- **Veri Bağımlılığı (Data Dependency)**
5- **Test ve Hata Ayıklama (Test and Debugging)**

İki Türlü Paralellik Vardır

Birden fazla görevin eş zamanlı yapılmasına ***paralellik*** denir.

1- **Data Parallelism**
   Bir veri kümesinin verilerini birden çok thread'de dağıtmaya denir.
2- **Task Parallelism**
   Verilerin değil, görevlerin birden çok çekirdek arasında dağıtılmasını içerir

Amdahl Yasası

Hem seri hem paralel olan bir uygulamaya ek çekirdekler eklemenin ne kadar hız artışına sebep olacağını hesaplayan yasadır.
Uygulama %75 paralel %25 seri ise 2 çekirdeğe 1,6 kat hızlanma ile sonuçlanır.
Burdaki S değeri %25, N değeri ise çekirdek sayısı yani 2

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

İki Çeşit Thread Vardır

1- **User Threads (Kullanıcı iş parçacıkları)**
   Bu thread'leri kullanıcı yönetir.
2- **Kernel Threads (Kernel (Çekirdek) İş Parçacıkları)**
   Doğrudan işletim sistemi tarafından desteklenir ve yönetilir.
   User thread'e göre daha yavaştır.

Multithread Modeller

1. **Çoktan Bire (Many to One)**
   - Birden fazla user thread, bir tane kernel thread ile eşleştirilir.
   - Birden fazla iş parçacığı paralel olarak çalışamaz.
   - Kernel thread'e bağlanan user thread'lerden biri iptal edilirse o kernel thread'e bağlı bütün thread'lerin process'leri iptal edilir.
   - Solaris Green Threads, GNU Portable Threads kullanır.
2. **Birden Bire (One to One)**
   - Bir user thread'inin bir kernel modeli ile eşleşmesidir.
   - Thread'ler paralele olarak çalışabilir.
   - Windows, Linux kullanır.
3. **Çoktan Çoka (Many to Many)**
   - Birden çok user thread birden çok kernel thread ile eşleşir.
   - Bir user thread engellediğinde diğer user thread'ler diğer kernel thread'ler ile eşlenir.
   - Solaris 9, UNIX kullanır.

İki Seviyeli Model

Bunun, many to many'den farkı ekstra olarak bir user thread'in bir kernel thread'e bağlanmasına izin vermesidir.

Thread Kütüphaneleri

Thread kütüphaneleri, programcıya thread'leri (iş parçacığı) oluşturmak ve yönetmek için API sağlar.

3 temel kütüphane tarafından yönetilir:

- POSIX Pthreads -----> LINUX
- Windows threads
- Java threads ----------> JVM

Thread'ler için iki strateji vardır:

1- **Asenkron (Eş Zamansız)**
Parent thread alt thread'lerin (child) bitmesini beklemez. Paralelel (eş zamanlı) olarak işlemlerini gerçekleştirirler.

2- **Senkron (Eş Zamanlı)**
Parent; child thread'lerin, işlerini bitirmesini bekler.
Burada parent ve child'lar eşzamanlı olarak çalışır ancak child'lar işlerini bitirmeden parent devam edemez.

Implicit Threading (Örtülü Threading)

Thread yönetimini kullanıcıdan kitaplıklara aktarır. Thread'leri oluşturmayı, yönetmeyi kolaylaştırır.

Örtülü iş parçacığı oluşturmak için üç metot vardır:

1. ***Thread Pools***
2. ***OpenMP***
3. ***Grand Central Dispatch***

Thread Havuzları (Thread Pools)

Thread Havuzları, önceden belirlenmiş bir sayıda thread oluşturur ve bunları havuz adı verilen bir sistemde bekletir. Server'e bağlanan her bir client, havuzdaki thread'lerden birini kullanır. İşi bittikten sonra thread'i havuza geri bırakıp çıkar.

Thread Havuzları Kullanmanın Avantajları

1. Yeni bir thread oluşturup daha sonra silmekten daha hızlı ve az maliyetlidir.
2. Bağlanabilecek client sayısı sınırlandırılmıştır.
3. Thread sayısını aşan client'ların istekleri sırada bekletilir.

OPENMP

C, C++, Fortran ile yazılmış programlar için API sağlar.
Kod bloklarının başına kütüphanenin kendi keyword'ünü yazar *(#pragma omp parallel).*

Grand Central Dispatch (GCD)

C diline yönelik uzantıların, bir API'nin ve uygulama geliştiricilerinin paralel olarak çalışacak kod bölümlerini tanımlamasına olanak tanıyan bir çalışma zamanı kütüphanesidir.
C/C++ dilinde süslü parantezlerin başına '^' simgesi koyulunca blok bir thread'e dönüşür.
İki tip kuyruğu vardır:

1- **Seri Kuyruk**
   Programcı her bir thread eklemek istediğinde havuzundan bir blok eksiltir ve onu thread'e tahsis eder. Bloklar, havuzdan FIFO mantığında kaldırılır.
2- **Eşzamanlı**
   Kuyruktan birden fazla blok kaldırılabilir. Böylece birden çok thread paralel olarak çalışabilir.

fork() and exec()'İn Mantığı

Eğer bir thread, process oluşturmak için ***fork()*** çağırırsa, yalnızca process'i çağıran thread çoğaltılır.
exec() tüm iş parçacıkları dahil olmak üzere çalışan işlemi değiştirir.

Sinyal İşleme

Sinyaller, UNIX sistemlerde belli bir olayın gerçekleştiğine dair process'i bilgilendirmek için kullanılır.
Sinyal, kaynağa göre senkron ya da asenkron olabilir.

1. Sinyaller, sinyale neden olan process'e iletilirse bu, ***senkron'dur.***
2. Bir process'e harici bir process tarafından sinyal gönderilirse bu, ***asenkrondur***.

Thread İptali

Thread iptali, bir iş parçacığının tamamlanmadan sonlandırılmasını içerir. Örneğin tarayıcıda arama search barın solundaki X işaretine bastığımızda thread durur. Sayfa yüklenmeyi bırakır.
Pthreads'te, ***cancel()*** fonksiyonu ile iptal gerçekleşir.
İptal edilecek thread'e genelde ***target thread*** denir. İki çeşit target thread durumu (iptal durumu) var:

1. **Asenkron İptal**
   İptal işlemi, target thread'i (hedef iş parçacığını) aniden sonlandırır.
2. **Ertelenmiş İptal (Deferred Cancel)**
   Kendisini sürekli kontrol ederek, olması gereken vakitte kendisini sonlandırır.

İşletim Sistemleri Thread'leri (Kernel threads)

1. **Windows Threads**
   One to One thread birleşmesini uygular.
   Thread; user mod'da çalışırken bir user stack, kernel mod'da çalışırken ise bir Kernel stack içerir.
   Kayıt seti, yığınlar ve özel depolama alanı, iş parçacığının bağlamı (context) olarak bilinir.
   Bir Thread'in Birincil Veri Yapıları Şunları İçerir
   a- **ETHREAD (Execute Thread Block)**
      Thread'in ait olduğu process'in adresini ve thread'in çağırdığı fonksiyonun adresini içerir. Ayrıca KTHREAD için bir pointer içerir.
   b- **KTHREAD (Kernel Thread Block)**
      Thread için zamanlama ve senkronizasyon bilgilerini içerir. Ve ayrıca TEB için bir pointer içerir.
   c- **TEB (Environment Thread Block)**
      TEB, iş parçacığı kullanıcı modunda çalışırken erişilen bir kullanıcı alanı veri yapısıdır.
2. **Linux Thread**
   Lünux'da thread oluşturma, ***fork()*** ve ***clone()*** sistem çağrısı ile yapılır.

İşlem Senkronizasyonu

İş Birliği (Cooperation)

- Bir process'in çalışması başka bir process'i etkiliyorsa bu ***işbirliğidir.***
- İşbirliği yapan işlemler ya **hem kod hem de veri paylaşabilir** ya da **yalnızca dosyalar veya mesajlar aracılığıyla veri paylaşabilir**.
- Process'ler arası veri paylaşımı, ***üretici-tüketici problemi*** ile yapılabiliyordu.
  Ancak problemi hatırlarsak koddaki counter değişkeni ortaktı. Hem üretici hem tüketici aynı anda çalışıp counter'ı değiştirmek isterse ne olacak? (Örneğin, sayaç (counter) değişkeninin değerinin şu anda 5 olduğunu ve üretici ve tüketici süreçlerinin aynı anda "counter++" ve "counter--" ifadelerini yürüttüğünü varsayalım.)
  Bu durum veri tutarsızlıklarına neden olur. Bu yüzden paylaşımlı bir veriyi aynı anda sadece bir process manipüle edebilir.
- Birkaç işlemin aynı anda aynı verilere eriştiği yürütmenin sonucunun erişimin belirli sıraya bağlı olarak gerçekleştiği duruma ***yarış koşulu*** denir.

Kritik Bölüm Problemi (Critical Section)

Kritik bölümde bir işlem yürütülürken, başka bir işlem yürütülemez. Yani, kritik bölümlerinde aynı anda iki işlem yürütülmemelidir.
Kritik bölüm problemi, süreçlerin işbirliği yapmak için kullanabileceği bir protokol tasarlamaktır.

Kritik Bölüm Problemi Çözümü Aşağıdak Üç Gereksinimi Karşılamalıdır:

1. **Karşılıklı Dışlama (Mutual Exclusion)**
   P_i süreci kritik bölümünde yürütülüyorsa, kritik bölümlerinde başka bir işlem yürütülemez.
2. **İlerleme (Progress)**
   Eğer kritik bölümde herhangi bir işlem yoksa ve oraya erişmek isteyenler varsa, biri seçilir ve diğerleri kuyruğa alınır.
3. **Sınırlı Bekleme (Bounded Waiting)**
   Bölüme girmek için bekleyenler sonsuza kadar beklememeli.

İşletim Sisteminde Kritik Bölümü İşleme

Kernel'de işlemlerin çalışması için iki senaryo vardır:

1. **Preemptive**
   İşlemin belli bir süre çalışmasına müsaade edilir. Bu süre bitince de işlem askıya alınır ve başka işlem seçilir o çalışmaya başlar.
   Bir process tamamlanmadan diğer process'in çalıştırmılasıdır.
2. **Non-Preemptive**
   Bir process bitene kadar çalışır.

Peterson Çözümü

- Kritik bölüm problemini çözmeyi sağlayan ve ***karşılıklı dışlama, ilerleme*** ve ***sınırlı bekleme*** gereksinimlerini ele alan bir çözümdür.
- ***turn değişkeni***, kritik bölüme girme sırasının kimin olduğunu gösterir.
- ***flag dizisi***, bir işlemin kritik bölümüne girmek için hazır olup olmadığını belirtmek için kullanılır.
- **Atomik:** Interrupt edilmeyen (kesilmeyen) kod parçacığı. Mesela sistemin saati.
- **NTP:** Zamanı çok dakik olarak tutan bir protokoldür.

Senkronizasyon Donanımı (Synchronization Hardware)

test_and_set() Fonksiyonu

- Paylaşılan kod kısmını kilitler/açar. Atomik olarak çalışır.
- İki, test_and_set() fonksiyonu aynı anda yürütülürse bunlar rastgele bir sırayla yürütülür.
- Girilen parametrenin orjinal değerini döndürür.

compare_and_swap() Fonksiyonu

- İki değişkenin değerini karşılaştırıp doğru olup olmadıklarına bakar sonra yerlerini değiştirir.
- Value değerinin orjinal parametresini döndürür. Atomik olarak yürütülür.
- Takas yalnızca "value" == "expected" ise gerçekleşir.

Mutex Kilitler

Mutex kilidi, kritik bölgeleri korumak ve böylece yarış koşullarını önlemek için kullanılır. Yani koda girerken kilitler *(acquire())* ve çıkarken kilidi açar (release()).
Birisi girmek için istek atarsa *acquire()* tarafından kilit açılana kadar engellenir release() ile kilit kaldırılır fakat burada da ***busy waiting*** oluşur.
Bir lock yani işlem CPU'yu gereksiz yere meşgul ediyorsa buna ***busy waiting*** denir. Gereksiz bir bekleme oluşturduğu için yani sürekli while döngüsüne takılı kaldığı için buna da ***spinlock*** denir.

Semafor (Semaphore)

- Bir processin ya ***wait()*** olup beklemesi ya da ***signal()*** olup işleme alınmasıdır.
  wait() işlemi P() olarak signal işlemi V() olarak adlandırılır.
- İki çeşit semaphore vardır: ***İkili semafor*** ve ***sayma semaforu.***
  İkili semaforlar mutex kilitlerine benzer yani ***başka kimsenin çalışmamasını sağlar.***
  Sayma semaforları, sınırlı sayıdaki bir kaynağa erişimi kontrol etmek için kullanılabilir.
- ***signal(),*** o bölgeyi process'e (işleme) açtığında tetiklenir ve sayı bir artırılır.
  block() fonksiyonu ile, bölgeye erişmek isteyen diğer process'ler kuyruğa alınır.
  wakeup() fonksiyonu ile uyandırılır ve hazır kuyruğuna yerleştirilir. Böylece ***busy waiting*** durumunun önüne geçilir.

Deadlock (Ölümcül Kilit)

İki veya daha fazla process'in aynı kaynaklara erişmek istemesi durumunda birbirini beklemesidir.

Starvation (Açlık)

Örneğin bir öncelik sırası düşünelim herkesin bir öncelik sırası var yüksek öncelikler sıranın önünde düşük öncelikliler sıranın arkasına geçiyor orta öncelikli birinin olduğunu ve sıranın ortasında olduğunu varsayalım. Yüksek öncelikliler bitiyor bu kişiye sıra gelmeden başka yüksek öncelikli birileri geçiyor dolayısıyla bu ortadaki kişiye sıra gelmeme ihtimali var buna ***starvation*** denir.
Özetle ***starvation***, semafor bekleme listesinde bekleyen bir işlemin hiçbir zaman listeden silinmemesi ve process'e hiç bir zaman çalışma sırası gelmemesi diyebiliriz.

Priority Inversiton (Önceliği Terse Çevirme)

Eğer bir thread resource'u kullanıyorsa mutex'i kitliyor, diğer thread kullanmak istediğinde mutex kilitliyse, mutex açılana kadar beklemeye başlıyor. Bu sayede birden çok thread aynı anda aynı resource'a erişemiyor.

CPU Zamanlama

- Tek işlemcili bir sistemde, aynı anda yalnızca bir işlem çalışabilir. Diğerleri, CPU boşalana ve yeniden programlanana kadar beklemelidir.
- ***Çoklu programlamanın amacı***, CPU kullanımını en üst düzeye çıkarmak için bazı işlemlerin her zaman çalışmasını sağlamaktır. Çoklu programlama ***bellekteki işlem sayısı*** ile ölçülür.
- ***CPU Burst:*** Bir process'in işlemcideki çalışma süresi. Örneğin şu process işlemcide 2 saniyede çalışır gibi …
  IO Burst: Bir process'in IO cihazlarında çalışma süresi.

Zamanlayıcı

- ***Zamanlayıcı,*** bellekte yürütülmeye hazır olan işlemlerden bir işlem seçer ve CPU'yu bu işleme tahsis eder.
- CPU boşta kaldığında, işletim sistemi yürütülmek için ***hazır kuyruğundaki*** işlemlerden birini seçmelidir. Seçim süreci, ***kısa süreli zamanlayıcı*** tarafından gerçekleşir.
  Hazır kuyruğunun bir ***ilk giren ilk çıkar (FIFO)*** kuyruğu olmadığını unutmayın.
- CPU zamanlama kararları aşağıdaki durumlardan biri gerçekleşince alır:
  1. Bir işlem running durumundan waiting durumuna geçerken - non-preemptive
  2. Bir işlem running durumundan ready durumuna geçerken - preemptive
  3. Bir işlem waiting durumundan ready durumuna geçerken - preemptive
  4. İşlem sonlanırken - non-preemptive

Dispatcher (Gönderici)

- ***Dispatcher, kısa vadeli programlayıcı*** tarafından seçilen process'e CPU'nun kontrolünü veren modüldür kısaca atama yapar.
  - İçerik değiştirir (switching context)
  - Kullanıcı moduna (user mod) geçiş sağlar
  - Programı yeniden başlatmak için kullanıcı programında uygun konuma atlama yapar.
- Göndericinin bir işlemi durdurup başka bir işlemi başlatması için geçen süre, ***dispatch gecikmesi (dispatch latency)*** olarak bilinir.

Zamanlama Kriterleri

Farklı durumlar için farklı algoritmalar kullanılmaktadır. Bazı kriterler şunlardır:

- **CPU Kullanımı**
- **Verimlilik**
  Tamamlanan işlemlerin sayısıdır.
- **Geri Dönüş Süresi (Turnaround Time)**
  Bir işlemin başlaması ile bitmesi arasıdaki süre.
- **Bekleme Zamanı (Waiting Time)**
  Hazır kuyruğunda bekleme sürelerinin toplamıdır.
- **Yanıt Süresi (Response Time)**
  İşlemcinin, process'e ne kadar süre içerisinde yanıt verdiğidir.

First- Come, First-Served (FCFS) Zamanlama (İlk Gelen İlk Hizmet Alır)

- Hangi işlem önce ready kuyruğuna geldiyse o ilk çalışır.
- Aşağıdaki örnek için (non-preemptive) P1'in önünde process yok o yüzden bekleme süresi 0'dır.
  P1'in Turnaround Time: 24
  P2'nin Turnaround Time: ready+burst time = 24+3=27
  P3'ün Turnaround Time: ready+burst time = 27+3=30
  (30+27+24)/3=27

Process	Burst Time
P_1	24
P_2	3
P_3	3

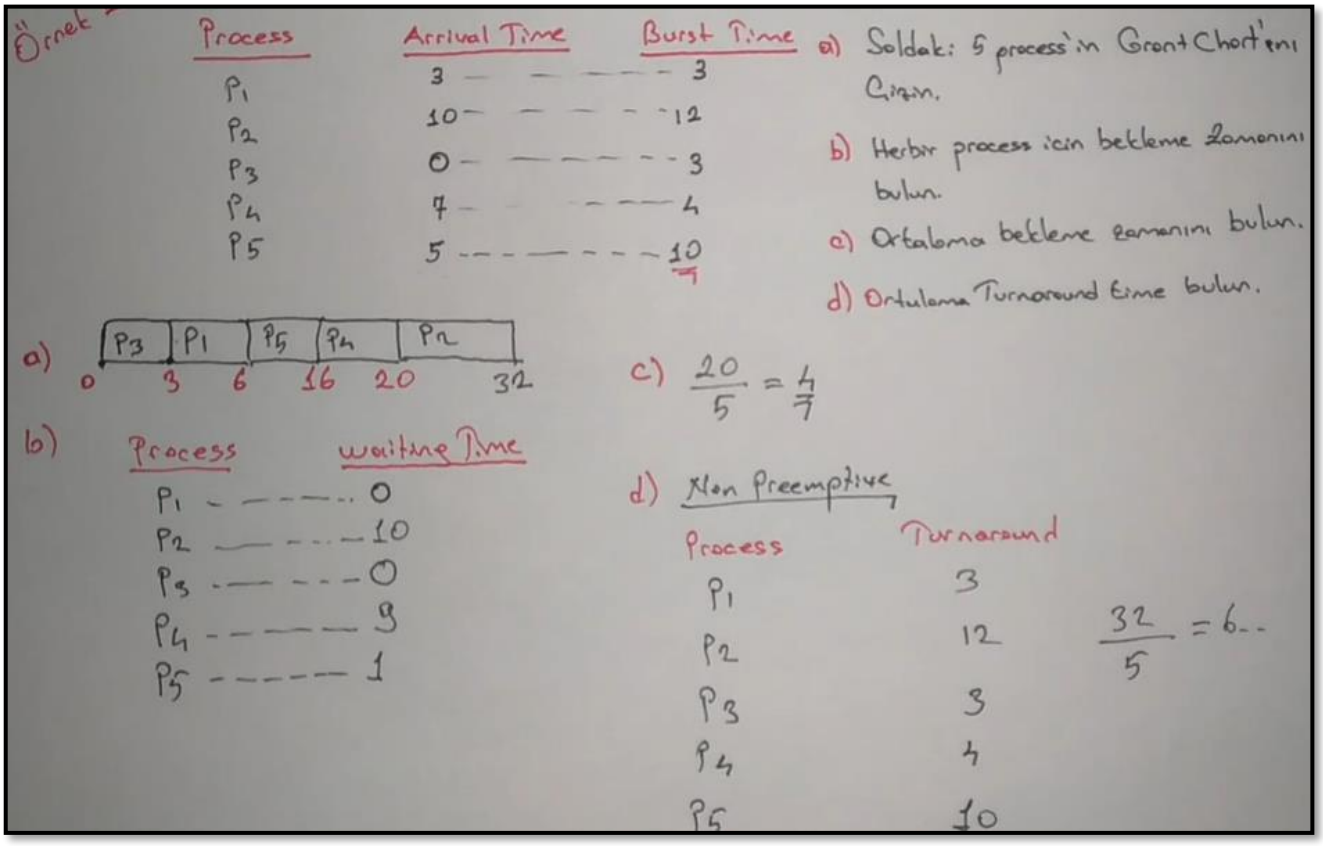
Bekleme zamanları: $P_1 = 0$; $P_2 = 24$; $P_3 = 27$

Ortalama Bekleme Zamanı: $(0 + 24 + 27)/3 = 17$

- P1'in Turnaround Time: 24
  P2'nin Turnaround Time: 3
  P3'ün Turnaround Time: 6
  (30+3+6)=/3=13

Bekleme Zamanları: $P_1 = 6$; $P_2 = 0$; $P_3 = 3$

Ortalama Bekleme Zamanı: $(6 + 0 + 3)/3 = 3$

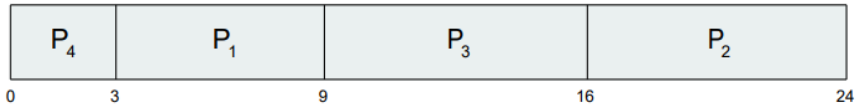


- **Konvoy Efekti,** büyük process'i (büyük zamana ait process'i) bekleme durumu.

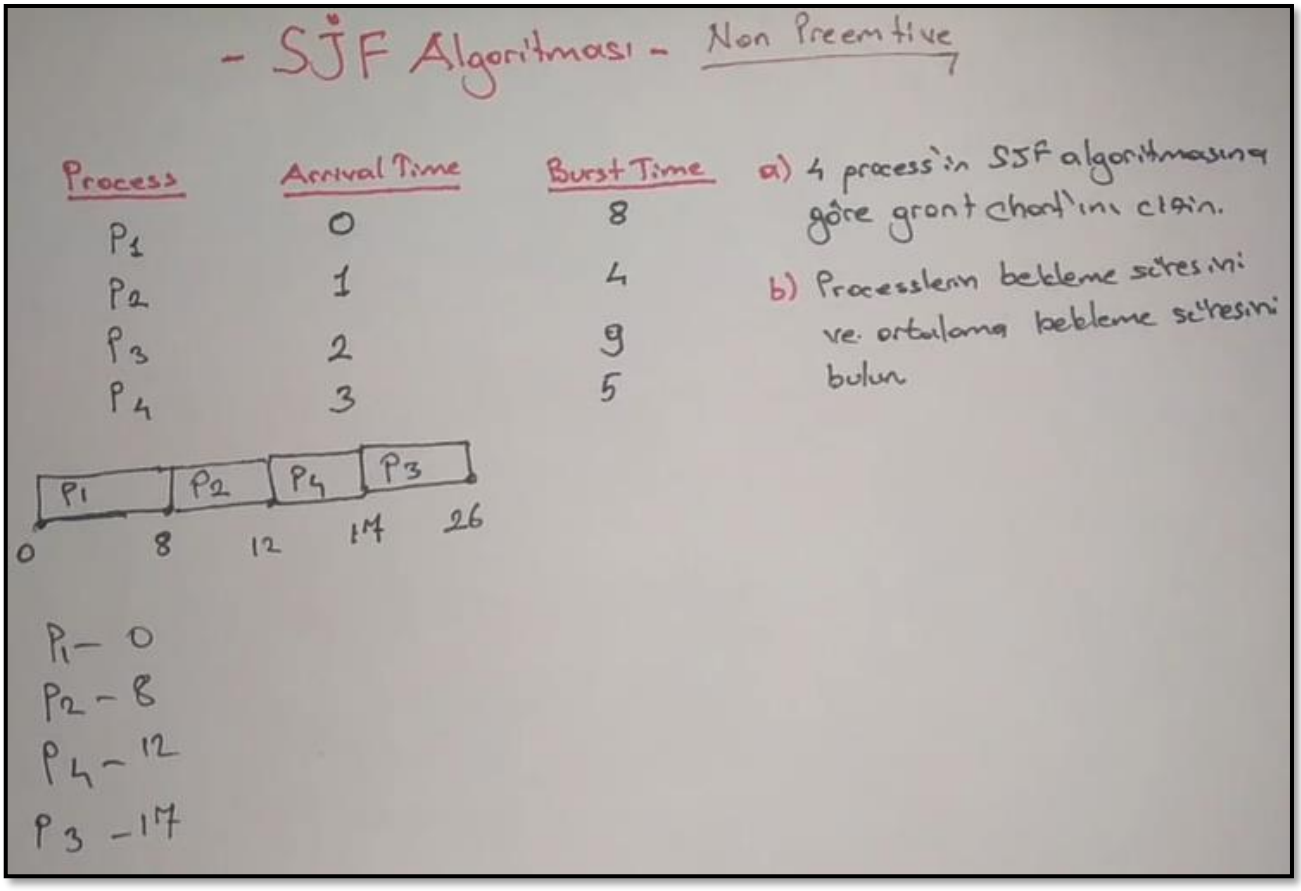
Shortest-Job-First (SJF) Zamanlama (En Kısa İlk İş)

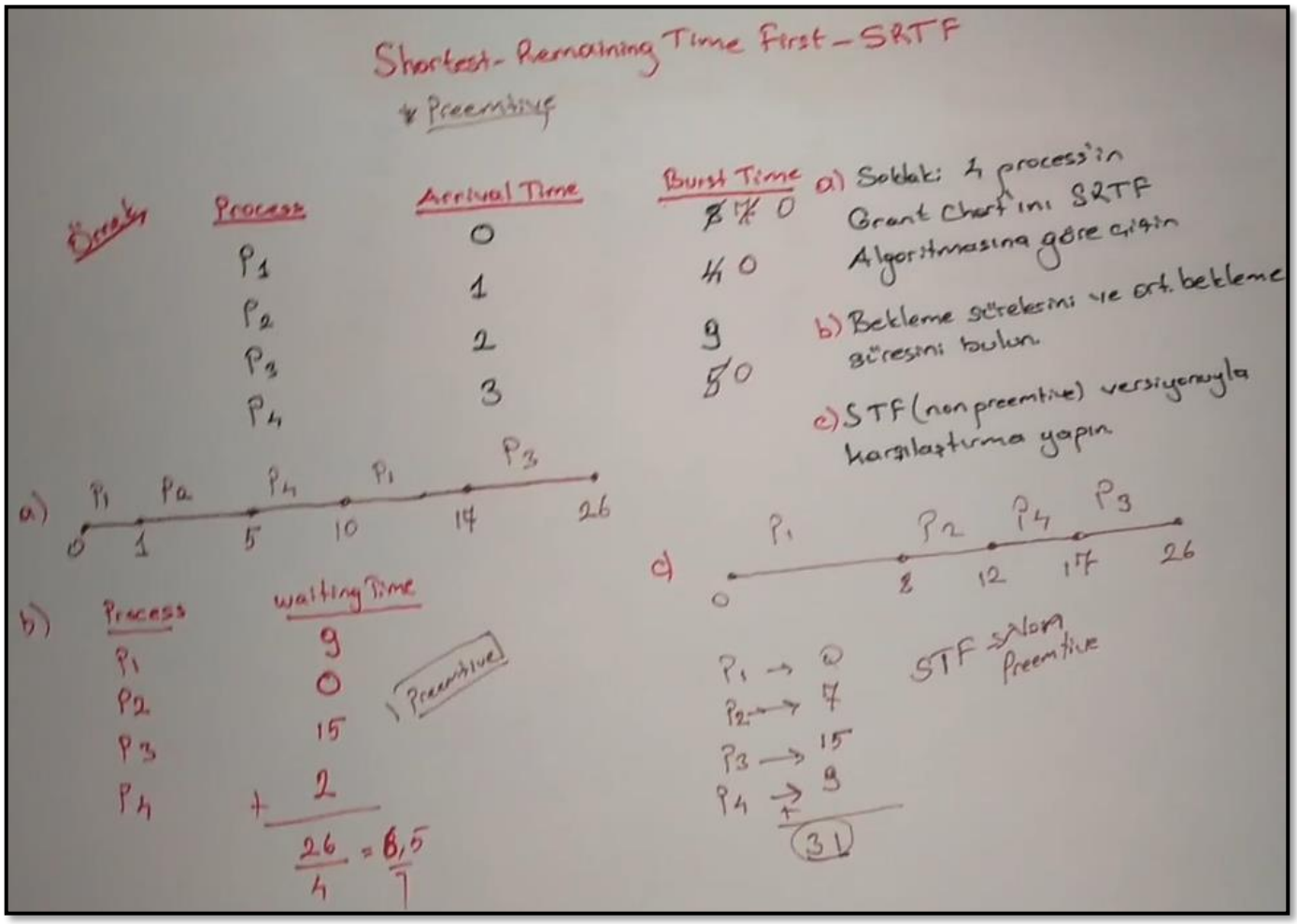
- İki işlemin sonraki CPU patlamaları aynıysa, bağı kırmak için FCFS zamanlaması kullanılır.
- ***Preemptive SJF algoritması,*** halihazırda yürütülmekte olan işlemi önleyecektir.
  Oysa ***nonpreemptive SJF algoritması,*** halihazırda çalışan işlemin CPU patlamasını bitirmesine izin verecektir.
- ***Preemptive SJF algoritması shortest-remaining-time-first*** olarak adlandırılır.
- Aynı süreye ait iki process varsa kuyruğa önce gelen işleme önce alınır.
- Algoritma en kısa zamanı seçer.
  P4>P1>P3>P2
  Turnaround Time P4: 3
  Turnaround Time P1: 9
  Turnaround Time P3: 16
  Turnaround Time P2: 24
  Ortalama Turnaround Time: (3+9+16+24)/4=13

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



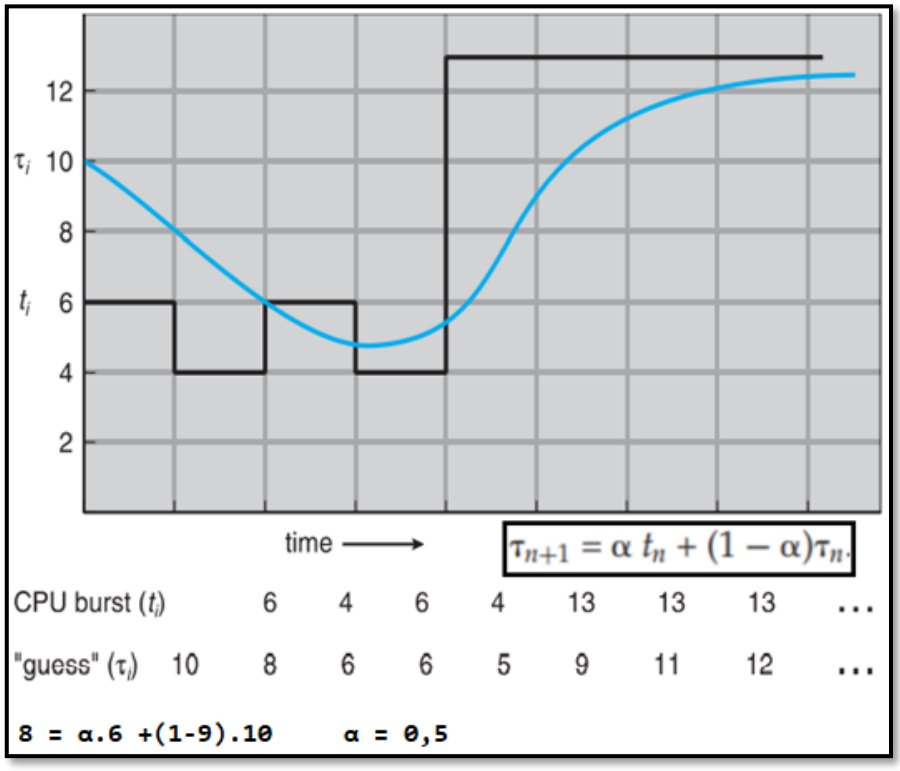
Ortalama bekleme süresi = (0 + 3 + 9 + 16) / 4 = 7





Sonraki CPU Burst Uzunluğunun Belirlenmesi

- Bir sonraki CPU patlamasının uzunluğunu bilemeyebiliriz, ancak değerini tahmin edebiliriz.

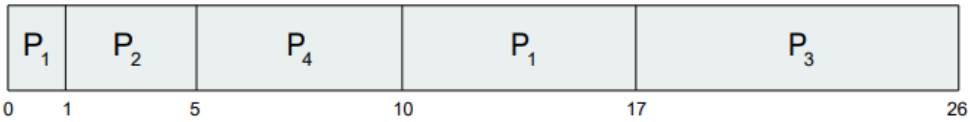


En Kısa Kalan İş Birinci Örneği

- Turnaround Time P1: 9+8=17
  Turnaround Time P2: 0+4=4
  Turnaround Time P3: 15+9=24
  Turnaround Time P4: 2+5=7 --> Buradaki 2 --> 5-3=2'den geliyor.
  17+4+24+7=52 52/4=13

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Preemptive SJF Gantt Grafiği



Ortalama bekleme zamanı= [(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4 = 6.5 msec

Öncelikli (Priority) Zamanlama

- CPU, en yüksek önceliğe sahip işleme tahsis edilir (genellikle en küçük tam sayı = en yüksek öncelik değeridir).
- Yeni gelen işlemin önceliği, o anda çalışmakta olan işlemin önceliğinden daha yüksekse ***preemptive*** yapılır.
  Nonpreemptive'de ise işlem bitirildikten sonra yüksek önceliğe sahip işlem alınır.
- Öncelikli zamanlama algoritmalarıyla iki büyük sorun:
  Starvation (process'e hiç bir zaman çalışma sırası gelmemesi)
  Infinite blocking
- ***Aging (Yaşlanma),*** sistemde uzun süre bekleyen süreçlerin önceliğinin kademeli olarak artırılmasıdır.
  Düşük önceliğe sahip kişi sırada bekledikçe onun öncelik derecesini arttırırız.
- Priority'si en düşük olan önce çalışır.

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Öncelikli planlama Gantt Grafiği

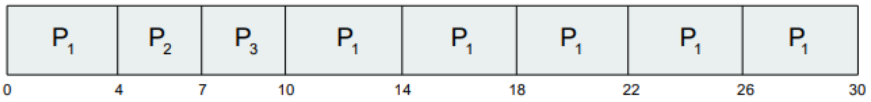
Ortalama bekleme zamanı = 8.2 msec

Round Robin (RR)

- CPU zamanlayıcı hazır kuyruğundan ilk işlemi seçer, 1 kuantum süresinden sonra kesintiye uğramak için bir zamanlayıcı ayarlar ve işlemi CPU'ya gönderir.
- Özetle; bir kuantum zaman belirlenir. Çalışması bu zamandan uzun olan process'ler bu süreye ulaştığında kesilir ve sıradaki process çalışır. Herbiri aynı şekilde uzunsa kesintiye uğrar. Eğer kuantum süresinden kısaysa çalışır ve biter.
- Bir process'in kuantum zamanı ne kadar küçük olursa o kadar kesinti olup sıradaki process çalışacak demektir. Herbir kesinti de context switching demektir.

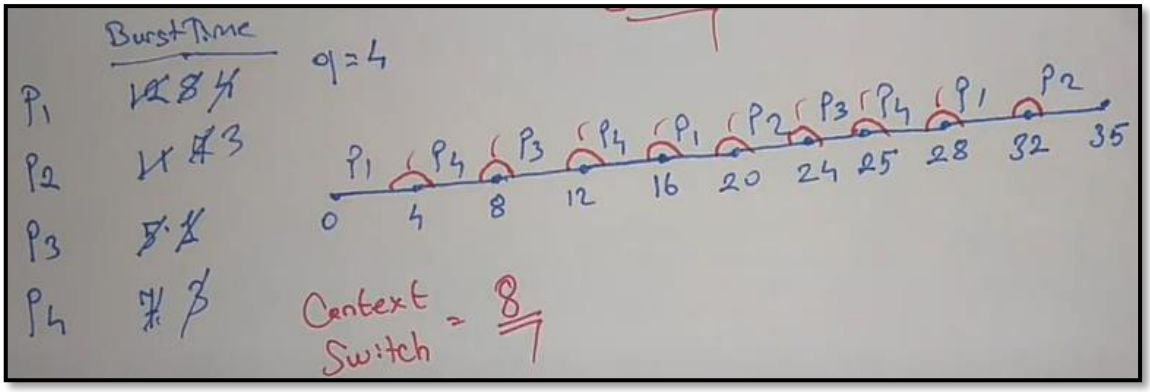
Process	Burst Time
P_1	24
P_2	3
P_3	3

Gantt grafiği:



Kuantum zamanı 4 msec

P_1 6 msec, P_2 4 msec ve P_3 7 msec bekler, ortalama 17/3=5.66 msec



Çok Seviyeli Kuyruk (Multilevel Queue)

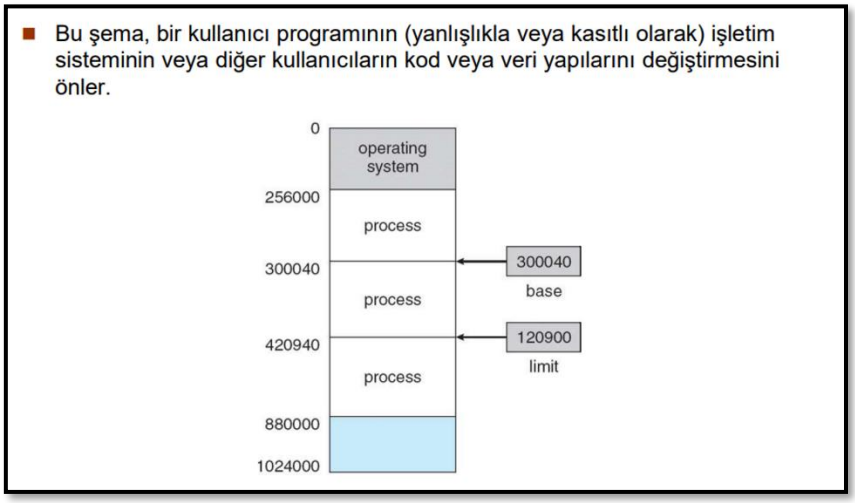
- Bütün process'lerin tek bir kuyrukta durması yerine process'lerin farklı kuyyruklarda durmasıdır.
- Öncelik sıralamalarına göre kuyruk sıralanır. Öncelikli kuyruktaki process'ler bitmeden daha az yetkiye sahip process'lere sıra gelmez.
- İşlemler, işlemin bellek boyutu, işlem önceliği veya işlem türü gibi özelliklerie dayalı olarak olarak kuyruğa atanır.

Çok Seviyeli Geri Besleme Kuyruğu

- Çok seviyeli geri bildirim kuyruğu aşağıdaki parametrelere göre tanımlanır:
  Kuyruk sayısı.
  Bir işlemin ne zaman daha yüksek öncelikli bir kuyruğa yükseltileceğini belirlemek için kullanılan yöntem.
  Bir işlemin ne zaman daha düşük öncelikli bir kuyruğa indirgeneceğini belirlemek için kullanılan yöntem.
  Bir işlemin hizmete ihtiyaç duyduğunda hangi kuyruğa gireceğini belirlemek için kullanılan yöntem.

Hafıza Yönetimi

- Programlar, eriştikleri verilerle birlikte yürütme sırasında en azından kısmen ana bellekte olmalıdırlar.
- Bir işlemin erişebileceği kendi yasal bellek alanı olmalıdır. Bu korumayı bir taban ve bir sınır olmak üzere iki adet kayıt (register) kullanarak sağlayabiliriz:
  Taban registerı en küçük yasal fiziksel bellek adresini tutar; ***limit kaydı***, aralığın boyutunu belirtir.
  Örneğin, taban registerı 300040'a sahipse ve limit registerı 120900 ise, program yasal olarak 300040 ila 420939 (dahil) arasındaki tüm adreslere erişebilir.
- ***Bellek alanının korunması,*** CPU donanımının kullanıcı modunda oluşturulan her adresi kayıtlarla karşılaştırmasını sağlayarak gerçekleştirilir.



- Ayrıcalıklı komutlar (taban ve limit kayıtları) yalnızca çekirdek modunda yürütülebildiğinden, sadece işletim sistemi taban ve sınır yazmaçlarını yükleyebilir.
  Çekirdek modunda yürütülen işletim sistemine, hem işletim sistemi belleğine hem de kullanıcı belleğine sınırsız erişim hakkı verilir.

Adres Bağlama

- Talimatların ve verilerin bellek adreslerine adres bağlaması üç farklı aşamada gerçekleşebilir.
  - **Yükleme Zamanı (Loading Time)**
    Sistem kütüphaneler load-time'da yüklenirken DLL'ler run-time'da yüklenir.
  - **Derleme Zamanı (Compile Time)**
  - **Uygulama Zamanı (Runtime - Execution Time)**
- Bir program yürütülebilmesi için programın belleğe bir işlem olarak yerleştirilmesi gerekir.
- CPU tarafından üretilen bir adrese genellikle ***mantıksal adres*** denir, oysa bellek birimi tarafından görülen bir adrese - yani belleğin bellek adres yazmacına yüklenen adrese - genellikle ***fiziksel adres*** denir.
- Mantıksal adreslerden fiziksel adreslere çalışma zamanı eşlemesi, ***bellek yönetim birimi (Memory Management Unit)*** adı verilen bir donanım aygıtı tarafından yapılır.

Bellek Hafıza Ünitesi

- Örneğin, temel yazmacı 14000'deyse, kullanıcı tarafından konum 0'a yönelik bir girişim, dinamik olarak konum 14000'e konumlandırılır; 346 konumuna bir erişim, 14346 konumuna eşlenir.
- ***Bellek eşleme donanımı***, mantıksal adresleri fiziksel adreslere dönüştürür.

Dinamik Bağlama

- ***Statik bağlama,*** yükleyici (loader) tarafından ikili program görüntüsünde birleştirilen sistem kütüphaneleri ve program kodudur.
- **Stub,** dinamik kütüphanenin adresini gösteren bir işaretçidir.

Swapping

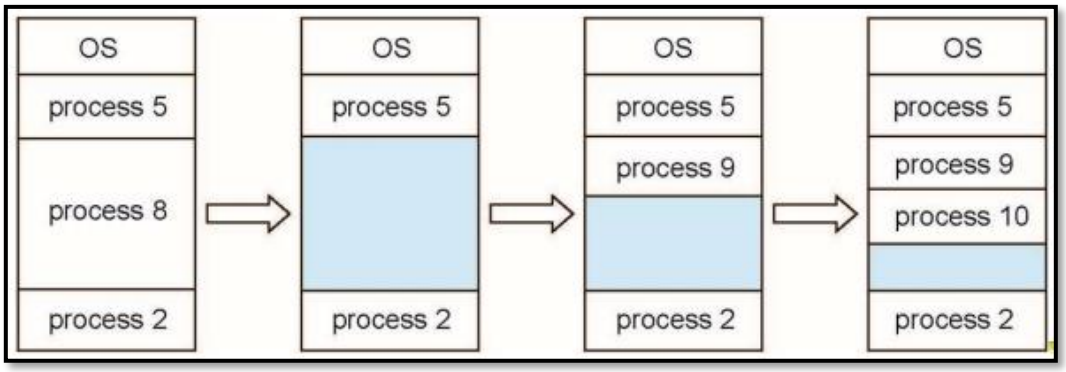
- Standart swap, hafıza ile sabit disk arasında yapılan process değişimidir. Yer kalmazsa process disk'e taşınır. Sırası geldiğinde memory'e geri getilir.
- ***Dağıtıcı,*** sıradaki bir sonraki işlemin bellekte olup olmadığını kontrol eder.
- Değilse ve boş bellek bölgesi yoksa, dağıtıcı o anda bellekte bulunan bir işlemi istenen işlem ile takas eder.
- Kullanıcı işleminin boyutunun 100 MB olduğunu ve yedekleme deposunun saniyede 50 MB aktarım hızına sahip standart bir sabit disk olduğunu varsayalım. 100 MB'lik işlemin ana belleğe aktarımı 2 saniyedir. Karşılıklı takas olacağı için toplam zaman yaklaşık 4 saniye (400ms)'dir.

Bitişik Swap (Contiguous Swap)

- Bellek genelde iki bölüme ayrılır; ***işletim sistemi*** ve ***kullanıcı işlemleri***.
  - ***Low Memory:*** İşletim sisteminin oturduğu hafıza
  - ***High Memeory:*** Kullanıcıya tahsis edilen hafıza
- İşletim sistemini düşük belleğe (low memory) veya yüksek belleğe (high memory) yerleştirebiliriz. Bu kararı etkileyen faktör, kesme vektörünün konumudur. Kesinti vektörü genellikle düşük bellekte olduğundan, programcılar genellikle işletim sistemini de düşük belleğe yerleştirir.
- Kullanıcı işlemlerini birbirinden ve değişen işletim sistemi kodu ve verilerinden korumak için yer değiştirme yazmaçları kullanılır.
  - ***Taban yazmaç,*** en küçük fiziksel adresin değerini içerir.
  - ***Limit yazmaç,*** bir dizi mantıksal adres içerir.
  - ***MMU,*** mantıksal adresi dinamik olarak eşler.
  - Daha sonra çekirdek kodunun geçici olması ve çekirdeğin boyutunun değişmesi gibi eylemlere izin verebilir.

Çoklu Bölüm Yerleştirme

- Bellek ayırmanın en basit yöntemlerinden biri, belleği birkaç sabit boyutlu bölüme bölmektir.
- Bir bölüm boş olduğunda, giriş kuyruğundan bir işlem seçilir ve boş bölüme yüklenir. İşlem sona erdiğinde, bölüm başka bir işlem için kullanılabilir hale gelir.
- ***Delik (Hole),*** kullanılabilir bellek bloğu; çeşitli büyüklükteki delikler bellek boyunca dağılmıştır.
- İşletim sistemi belleğin hangi bölümlerinin kullanılabilir olduğunu (free partitions, hole)ve hangilerinin işgal edildiğini (allocated partitions) gösteren bir tablo tutar.
- Mevcut bellek blokları, bellek boyunca dağılmış çeşitli boyutlarda bir dizi delik içerir. Bir işlem geldiğinde ve belleğe ihtiyaç duyduğunda, sistem bu işlem için yeterince büyük bir delik arar.
- Delik çok büyükse, iki parçaya bölünür. Bir kısım gelen işleme tahsis edilir; diğeri delik kümesine döndürülür. Bir işlem sona erdiğinde, bellek bloğunu serbest bırakır ve bu blok daha sonra delikler grubuna geri yerleştirilir. Yeni delik diğer deliklere bitişikse, bu bitişik delikler daha büyük bir delik oluşturmak üzere birleştirilir.

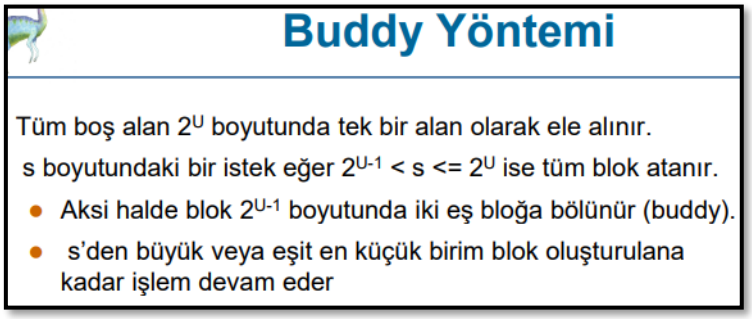


Dinamik Depolamada Yerleştirme Problemi

1- ***İlk Uyan (First Fit),*** yeterince büyük ilk delik tahsis edilir.
2- ***En İyi Uyan (Best Fit),*** yeterince büyük olan en küçük delik tahsis edilir.
3- ***En Kötü Uyan (Worst Fit),*** en büyük dedlik tahsis edilir.

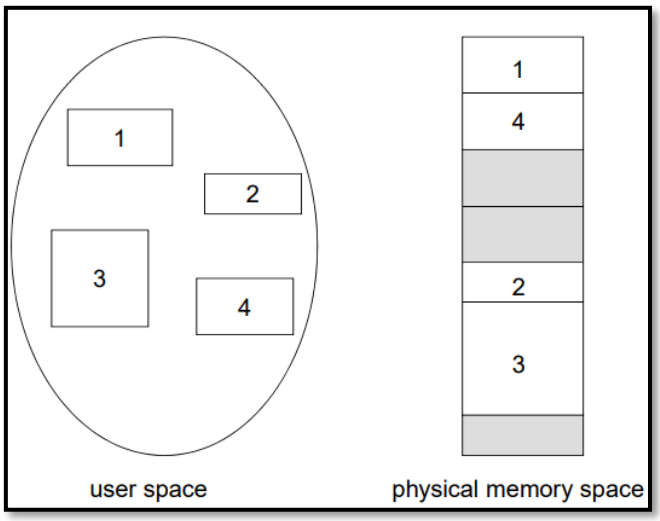
Fragmentation (Parçalanma)

- Bellek tahsisi için hem ilk uyum hem de en uygun stratejileri, harici parçalanma (external fragmentation) oluşturur.
- İşlemler yüklenip bellekten çıkarıldığında, boş bellek alanı küçük parçalara bölünür.
- Bir isteği karşılamak için yeterli toplam bellek alanı olduğunda ancak kullanılabilir alanlar bitişik olmadığında harici parçalanma vardır: depolama alanı çok sayıda küçük deliğe bölünmüştür.
- ***Dış parçalanma (external fragmentation),*** arta kalan bütün boş bellekleri birleştirip büyük bir alan oluşturur.
- **İç parçalanma (internal fragmentation),** programın boyutuna bakılmaksızın eşit büyüklükteki hafıza alanlarına oturular. Kendisine ayrılan bölümde açıkta kalan alana iç parçalanma denir.



Segmentasyon (Segmentation)

- Segment, aşağıdaki birimleri içerir: ana program (main program), prosedür (procedure), fonksiyon (function), metot (method), nesne (object), yerel ve global değişkenler (local and global variables), yığın (stack), sembol tablosu (symbol table) ve diziler (arrays).
- Her bölüm hafızada farklı yerlerde oalbilir. Ardarda sıralı olmak zorunda değillerdir.

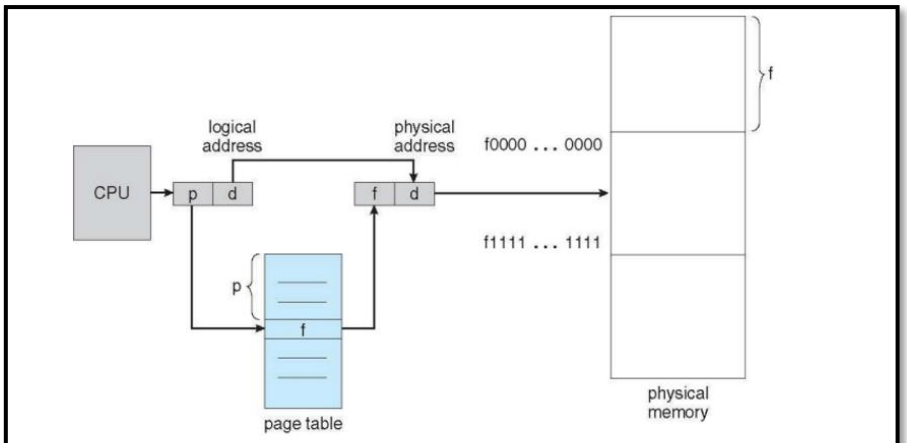


Segmentasyon Mimarisi

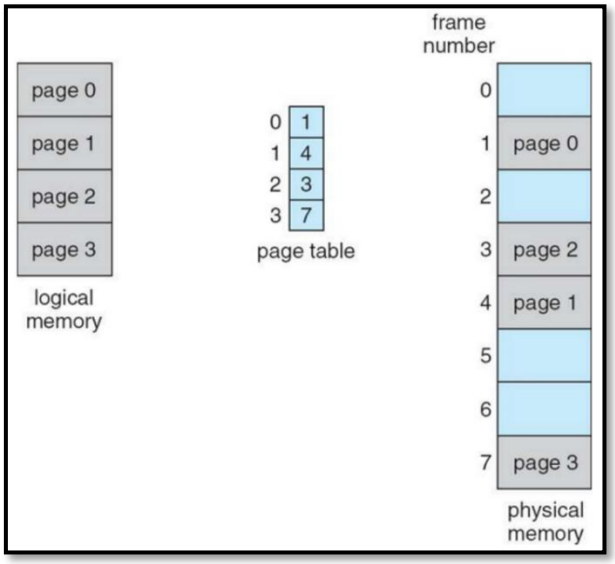
- Her segmentin bir adı ve uzunluğu vardır. Adresler, hem segment adını hem de segment içindeki ofseti belirtir. Uygulama kolaylığı için, segmentler numaralandırılmıştır ve segment adı yerine segment numarası ile anılırlar. Bu nedenle, mantıksal bir adres iki demetten oluşur: <segment-number, offset>
- Normalde, bir program derlendiğinde, derleyici, girdi programını yansıtan segmentleri otomatik olarak oluşturur. Bir C derleyicisi aşağıdakiler için ayrı bölümler oluşturabilir:
  1. Kod (Code)
  2. Global değişkenler (Global variables)
  3. Belleğin ayrıldığı yığın (heap) bellek
  4. Her bir iş parçacığı tarafından kullanılan yığınlar
  5. Derleme süresince bağlanan standart c kütüphaneleri

Paging

- Segmentasyon, bir işlemin fiziksel adres alanının bitişik olmamasına izin verir. Sayfalama, bu avantajı sunan başka bir bellek yönetimi tekniğidir.
- Sayfalama farklı boyutlardaki bellek parçalarının oluşmasını da önler.
- Sayfalama uygulamak için temel yöntem, fiziksel belleği çerçeve (frame) adı verilen sabit boyutlu bloklara ve mantıksal belleği sayfa (page) adı verilen aynı boyuttaki bloklara bölmeyi içerir.



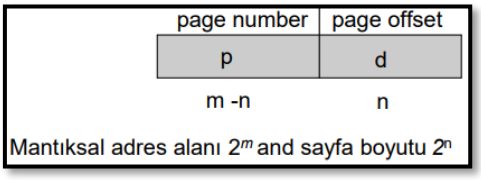
CPU tarafından üretilen her adres iki bölüme ayrılmıştır: bir sayfa numarası (p) ve bir sayfa ofseti (d). Sayfa numarası, sayfa tablosunda bir dizin olarak kullanılır. Sayfa tablosu, fiziksel bellekteki her sayfanın temel adresini içerir. Bu temel adres, bellek birimine gönderilen fiziksel bellek adresini tanımlamak için sayfa ofseti ile birleştirilir.

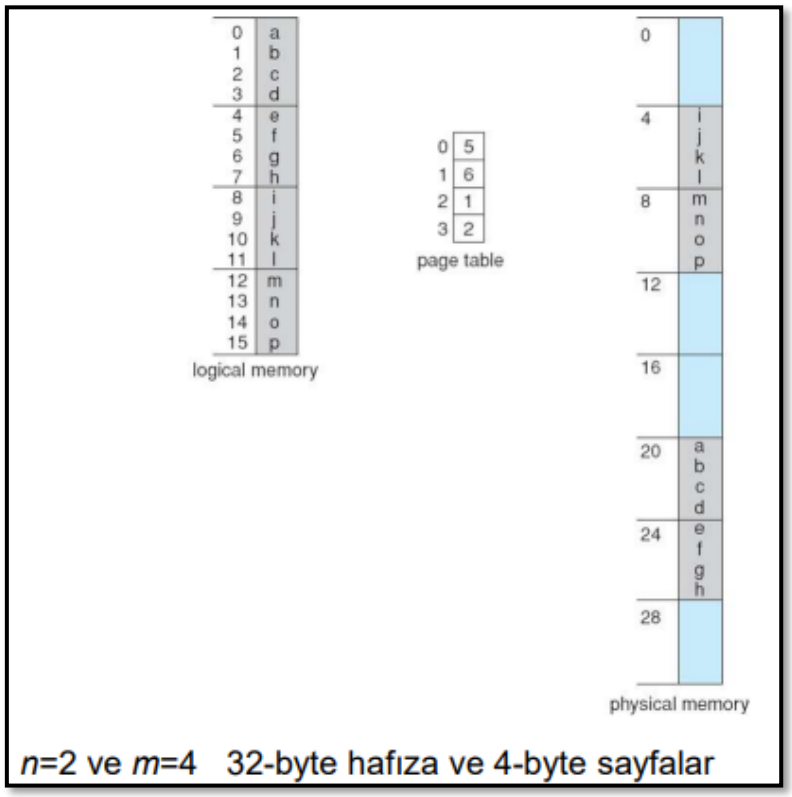


Adres Çevrimi

CPU tarafından oluşturulan adres iki bölüme ayrılmıştır:

1. Sayfa numarası (Page number, p): Fiziksel bellekteki her sayfanın temel adresini içeren bir sayfa tablosunda (page table) dizin olarak kullanılır.
2. Sayfa ofseti (Page offset, d): Bellek birimine fiziksel bellek adresini tanımlamak için temel adresle birlikte gönderilen değer.





Bir sayfalama şeması kullandığımızda, harici parçalanma olmaz: herhangi bir boş çerçeve, ihtiyacı olan bir işleme tahsis edilebilir. Ancak, bazı iç parçalanmalarımız olabilir.İç parçalanmanın hesaplanması:

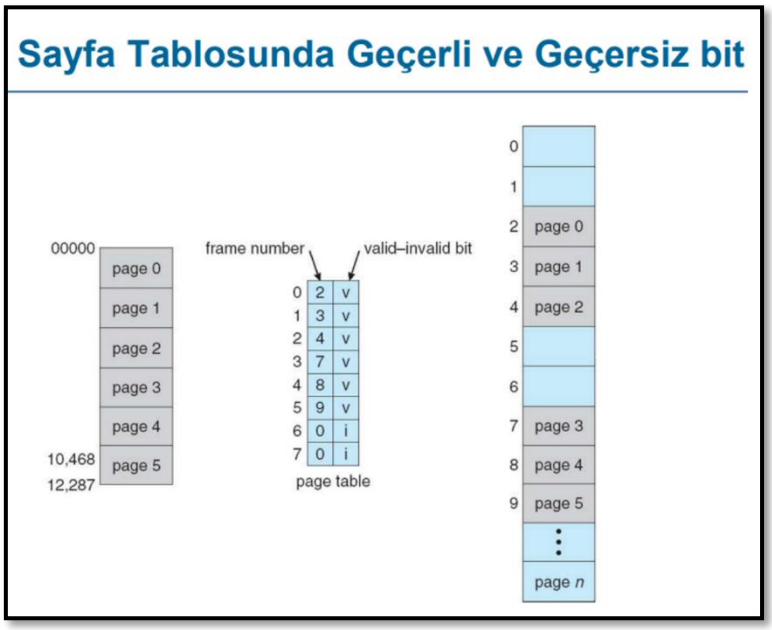
- Sayfa boyutu= 2,048 byte
- İşlem boyutu = 72,766 byte
- 35 sayfa + 1,086 byte
- İç parçalanma 2,048 - 1,086 = 962 byte
- En kötü durum = 1 frame – 1 byte
- Ortalama parçalanma = 1 / 2 çerçeve boyutudur. Bu değer, küçük sayfa

Sayfa Tablosunun Uygulanması

- Sayfa tablosu taban kaydı (Page-table base register, PTBR) , sayfa tablosuna işaret eder.
- Sayfa tablosu uzunluk kaydı (Page-table length register, PTLR), sayfa tablosunun boyutunu belirtir.
- Bu yöntemde, her veri/komut erişimi iki bellek erişimi gerektirir. Biri sayfa tablosu ve diğeri veri / talimat için.

Hafıza Koruma

- Bir bit, bir sayfayı okuma-yazma veya salt okunur olarak tanımlayabilir.
- Belleğe yapılan her başvuru, doğru çerçeve numarasını bulmak için sayfa tablosundan geçer.
- Fiziksel adres hesaplanırken aynı zamanda, salt okunur bir sayfaya yazma yapılmadığını doğrulamak için koruma bitleri kontrol edilebilir.
- Salt okunur bir sayfaya yazma girişimi, işletim sisteminde bir donanım tuzağına (veya bellek koruması ihlaline) neden olur.
- Sayfa tablosundaki her girişe eklenen geçerli-geçersiz (valid-invalid) bit:
  - "geçerli", ilişkili sayfanın işlemin mantıksal adres alanında olduğunu ve dolayısıyla yasal bir sayfa olduğunu belirtir.
  - "geçersiz", sayfanın işlemin mantıksal adres alanında olmadığını belirtir
- Herhangi bir ihlal, kernelda bir tuzakla sonuçlanır.



Kod boyutu 150KB olan bir program farz edelim. Bu programdan 10 tane birden açtığımızı düşünürsek hafızada 150*10 = 1500KB yer kaplaycak ve hepsi aynı kod (eğer runtime de değişmiyorsa) ise hafızada boiuna aynı kod duracak. Bunun yerine kodun tek bir kopyasını bir page'de tutup diğer programarın o sayfayı paylaşmasını sağlamaya paylaşımlı sayaflar denir. Her process'in sadece kendi veri deposu olur. Kodları ortak.

- ***Yeniden giriş kodu (Reentrant code):*** Kendi kendini değiştirmeyen koddur, yürütme sırasında asla değişmez. Böylece iki veya daha fazla işlem aynı kodu aynı anda çalıştırabilir.