7 / 10 / 2015

İşletim sistemi: Bilgisayar kaynaklarının ölçü zaman bakılmaksızın şekilde yöneten ve kullanıcı ile donanımlar arasında anlaşmayı sağlayan yazılımlar topluluğudur

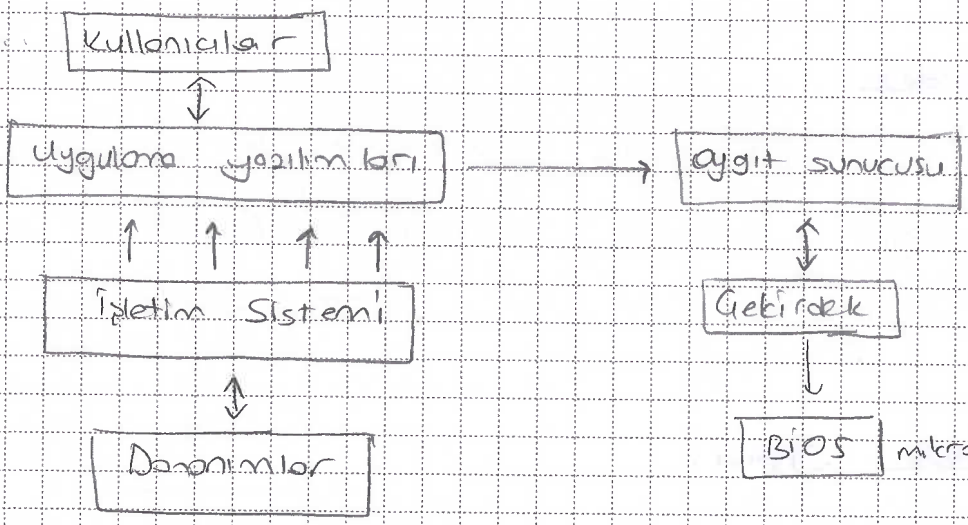
Çekirdek: İşletim sisteminin fiziksel donanımı kontrol eden kısmıdır. Çekirdek bu işlemi üreticinin sağladığı mikro kodlar aracılığıyla yapar. Mikrokodlar donanım üzerinde kalıcı bir bellekte tutulur. Günümüzde bu amaçla bios kullanılmaktadır (Basic Input output System) Bu mikro kodların sağladığı komut setine makine dili denir.

M → Veritabanı

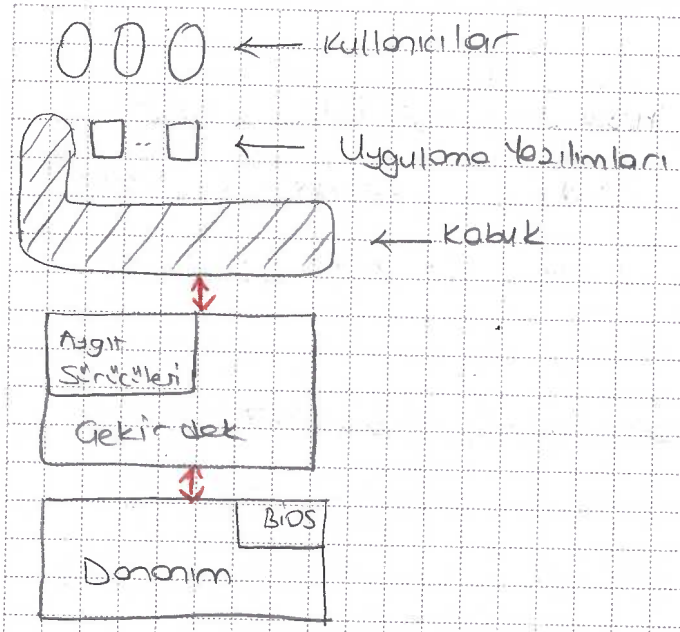
V → template

C → Control

} Entity M.



Kabuk (shell): İşletim sisteminin değişmez bir parçası olmayan yazılımlardır. İşletim sistemiyle kullanıcı arasında arayüz oluşturur. Grafik veya komut tabanlı olabilir.



Yapılarına Göre İşletim Sistemleri

1) Basit Sistemler (Monolitik Çekirdek):

Tek bir katmanda birbirini her an çağırabilen büyük prosedür yığınından oluşur. Prosedürler arasında arayüz parametreleri ve sonuçlar paylaşılır.

2) Mikro Çekirdek:

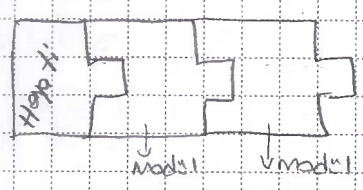
Monolit çekirdeklerin hantallık, yönetilememe ve hata bulma zorluğu gibi dezavantajları nedeniyle geliştirilmiştir. Çekirdekte sadece hayati temel servisler bulunur. Diğer herşey kullanıcı uzayında bulunur.

3) Hibrit Çekirdek Yapısı:

Monolit ve mikro çekirdek yapıları beraber kullanılır. Mikro çekirdekte sistem fonksiyonları (servisler) arttıkça ortaya çıkan performans düşüşünden dolayı hibrit çekirdekler çalışılmıştır.

4) Modüler Sistemler:

Çekirdek modüllerden oluşur. Modüller her işlem için tanımlı prosedürleri içerir. Bu modüller çekirdeğe açılış anında dinamik olarak bağlanırlar. Günümüzde işletim sistemleri bu yapıdadır.

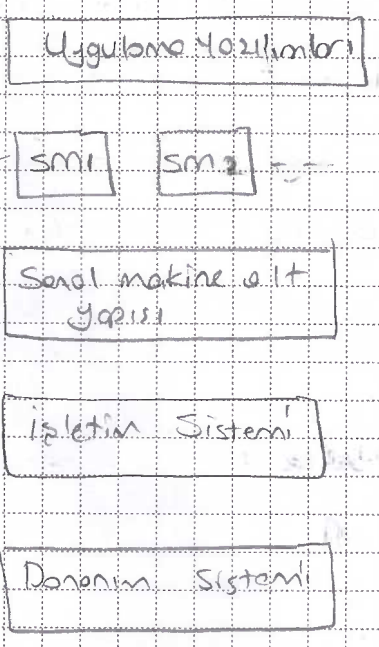


Bu modülleri takıp çıkarabiliriz

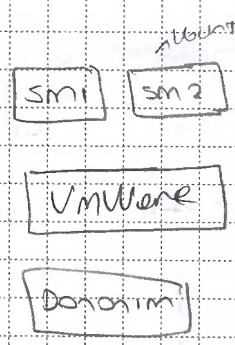
Günümüzde Linux, Unix kullanılmakta

5) Sanal Makineler:

İlk sanal makine IBM tarafından kaynakların programlar arasında paylaştırılırken oluşan problemleri gidermek için oluşturulmuştur. Günümüzde bir çok sanal makine uygulaması mevcuttur. Her sanal makinede farklı işletim sistemi çalıştırılabilir.



" Bazı sanal makinelerde işletim sistemi kullanılmıyor kendi yazılımları yerine onları yükleyerek öyle kullanıyorsun. VMWare, Proxmox gibi. "



6) Dış Çekirdek:

İlk kez MIT'de geliştirilmiştir. İlk kez MIT'de sanal makinelerin kaynaklarını kısıtlamak üzere geliştirilmiştir. Her sanal makineye donanımın bir kopyası sağlanır. Bu yapının en altında ise kaynakları sanal makinelere paylaştıran dış çekirdek bulunur. Dış çekirdek, sanal makinelerin birbirinden haberi olmadan donanımın tamamını kullanıyormuş gibi algılamasını sağlar. (Buradaki sanal makineler gerçek sanal makine oldukları anlamda değil)

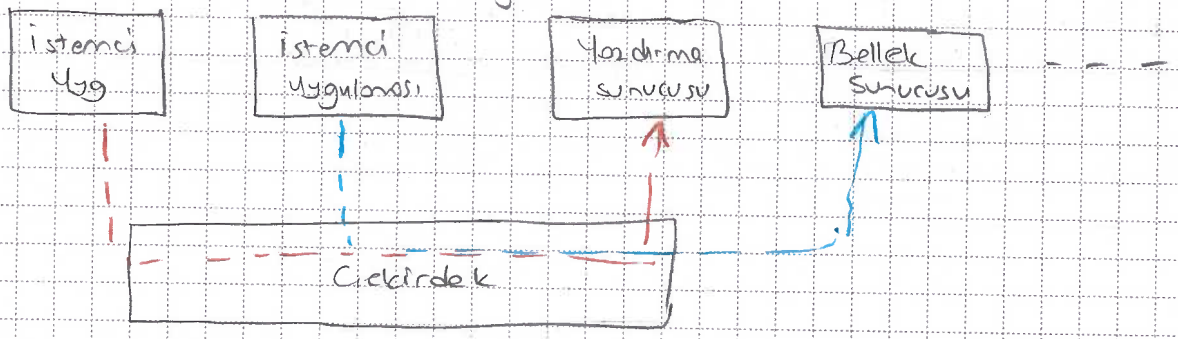
7) Katmanlı Sistemler:

6	Operatör
5	Kullanıcı Programları
4	G/Ç Yönetimi
3	Süreç Yönetimi
2	Bellek Yönetimi
1	CPU tahsisi ve çoklu programlama

Günümüzde kullanılmaktadır. UNIX'in temeli hali olan Multics kullanmıştır.

8) İstemci - Sunucu Yapısı:

Bilgiyi üreten sunucu servisleri ve işi talep eden istemci uygulamaları bulunur. Çekirdeğin görevi istemcilerle sunucular arasındaki iletişimi sağlamaktır.



14 / 10 / 2015

Kullanımlarına Göre İşletim Sistemleri

1) Ana Çatı (Main frame) → çok eski ve çok büyük pc'ler.

2) Sunucu Sistemler Windows Server...

3) Çok işlemcili Sistemler (Simetrik ve Asimetrik) → kullanılmıyor. / Simetrik Multi Processor (SMP) işletim sistemi

4) Kişisel bilgisayar İşletim Sistemleri (Operating Systems) / Ev PC'leri

5) Gerçek zamanlı İşletim Sistemleri RTOS

6) Kümelenmiş Sistemler (Cluster) N tane PC'nin tamamını sanal kümeye dahil edip çalıştırma

7) Dağıtık Sistemler Cluster'ın tersi / işletim sistemleri işlemleri farklı PC'lerde kümelemeden yapar

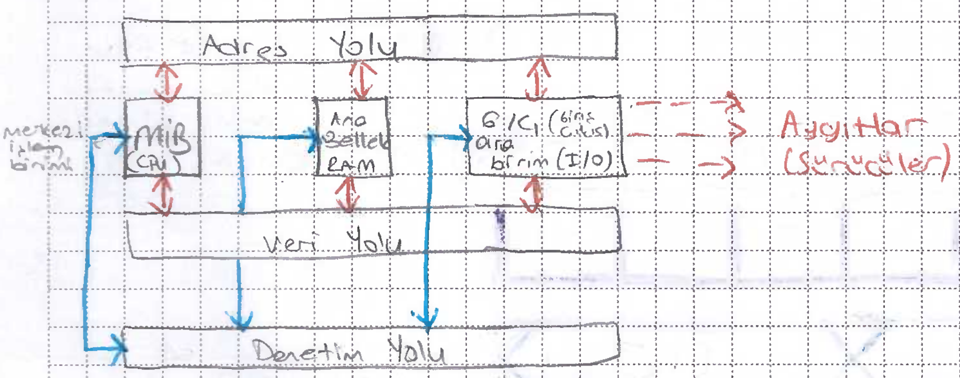
8) Gömülü İşletim Sistemleri

9) Akıllı kart İşletim Sistemleri

Bilgisayar Yapısı

Günümüzde Von Neumann mimarisi kullanılmaktadır.

(Merkezi işlem birimi (CPU) ve bellek olacak demiş)



* Üç yolun işleyişi kısaca anlatılırsa

Bellege Yazma → int i=5; print ("i" &i); → Adresi

Adres yolu = &i;
Veri yolu = 5
Denetim yolu = WR;

MİB Bileşenleri;

1) Aritmetik - Mantık birimi (ALU)
2) Kaydediciler (Yazmaçlar - Registers)
3) Denetim Birimi

} MİB'in içinde bulunan bileşenler

Adres Yolu:

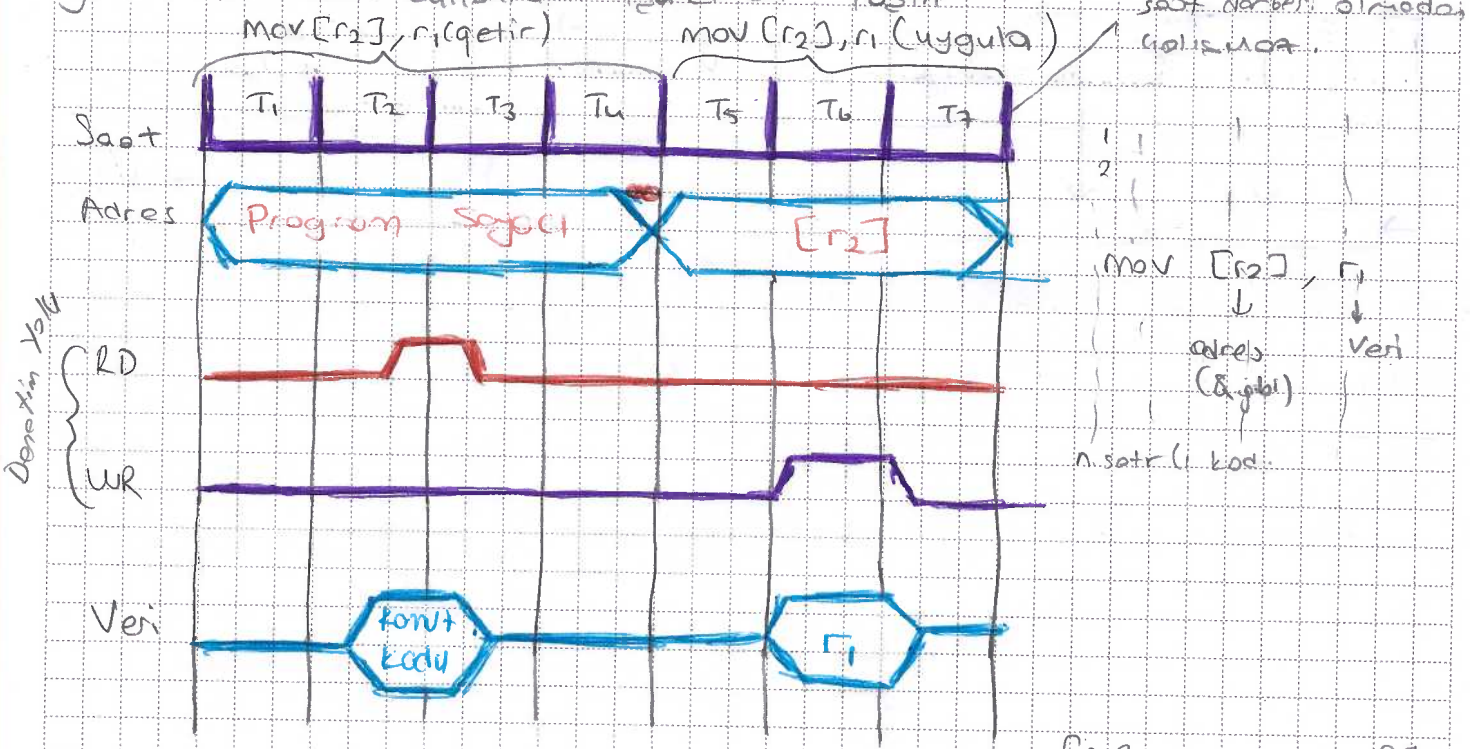
İşlemcinin okuma veya yazma amacı ile erişmek istediği bellek veya G/Ç ara birim adresini belirtmek için kullandığı genelde tek bir yoldur.

Veri Yolu:

Adres yolundaki adrese gönderilen veya alınan verinin taşındığı iki yönlü yoldur. Adres ve veri yolları sadece birlikte kullanıldıklarında anlamlıdır.

Denetim Yolu:

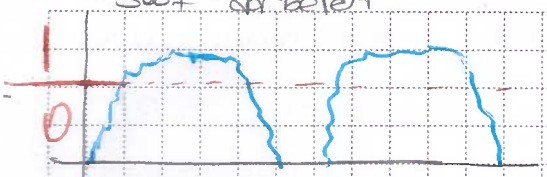
Birimler arasında aktarılan verinin kontrol edilmesi ve zaman uyumlaması için kullanılan işaretleri taşır.



İşlemci hızımız 2GHz ise
3x109 işlemi → 1 saniyede yapılır
1 saniyenin 3 milyonda biri zaman

[r2] → adres belirtir []
r1 → veri
diğer kaydediciler

Saat darbeleri



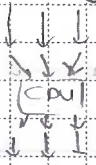
Bir voltaj anında 1 olur anda önce 0'dır

Diğer sayfadaki son çizimi incele grafiği oku yani;

Veri yolunda ve adres yolunda aynı anda veri olmalı ki çalışsın.

Tek İş ve Çok İş Düzeni (Single-tasking / multi-tasking)

Bilgisayar Sisteminde aynı anda birden fazla işlem yapılabilir. İlgini ifade eder. Çok iş düzeni sistem kaynaklarının verimli kullanılmasını sağlar ancak işlemler arasında bağlam değiştirme maliyeti gibi bir dezavantajı vardır.



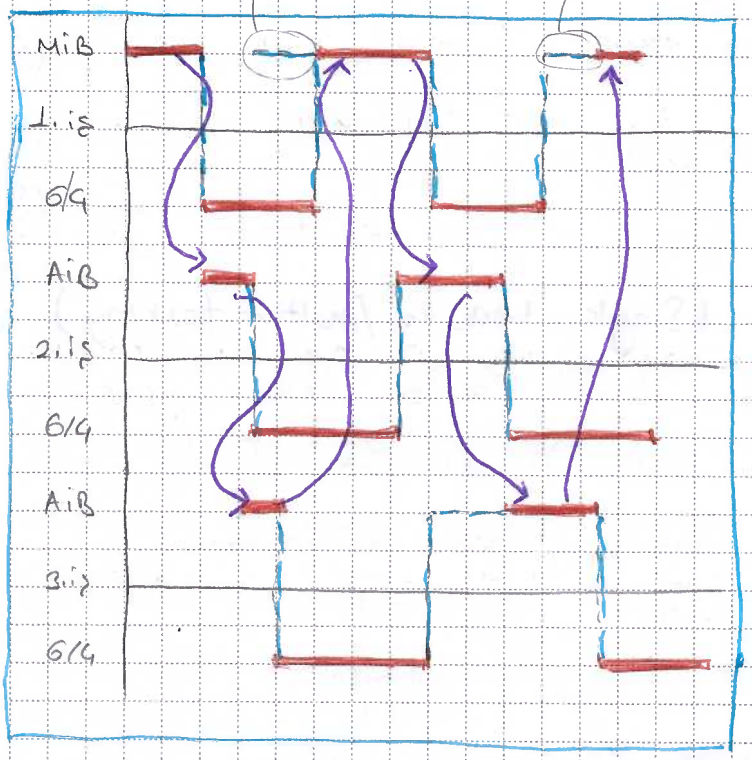
İşin (Job) Tanımı:

Kullanıcıların sistemden bir bütün olarak ele alınmasını istedikleri işlem takımıdır.

Örnek iş: Belgeler klasörünü sıkıştırıp arşivleyen orijinal klasörü silen bir iş.

1) tar -czf belgeler-yedek.tgz Belgeler/ (sıkıştırıp arşivleyen)
2) rm -rf Belgeler/ (silen)

} Bu ikisi "iş".



NOT

AHMET BURAK CAN
HACETTEPE
-Hocam bu pdf'i kullanın

GÖREV(TASK):

Program sözcüğü durgun komut dizisini tanımlarken görev bu komut dizisini işletim boyutuyla ele alan kavramdır.

GÖREV İSKELETİ

İşletimi kesilen programın daha sonra kaldığı yerden devam edilebilmesi ve işletim bütünlüğünün korunabilmesi için önceki konuma ait durum bilgilerini saklayan veri yapılarıdır.

GERÇEK ZAMANLI (REAL-TIME)

İşlemler çalıştırılırken tamamlanması için maksimum süre belirtilir.

GEREKİM İÇİ (online) İŞLEM

İşlenecek verilerin bilgisayar sistemine dolaysız ve aracısız biçimde aktarılmasıdır.

windows 10 performans özelliklerini bok tavsiyedi { Performance tuning Hardening Tips

TOPLU İŞLEM (BATCH PROCESSING)

İşler biriktirilerek veya dönem dönem sisteme sunulur. Sisteme sunulan işler sonlanıncaya kadar kullanıcı müdahalesine kapalı olarak işletilir.

ETKİLEŞİMLİ İŞLEM (REPONSE)

Kullanıcıların işin adımlarını izleye bildiği ve müdahale edebildiği işlemlerdir.

KESİLME / KESME / INTERRUPT

İşlemci görev yürütmesi sırasında acil ve öncelikli olarak yapılması istenen durumlarda kullanılır.

SİSTEM KOMUT YORUMLAYICISI

Kullanıcıların terminaller aracılığıyla girdikleri komutları yorumlayarak bu komutları yerine getirir. Günümüz işletim sistemlerinde grafik arayüzlerde bu kapsamdadır.

BİLGİSAYAR AÇILIŞ SÜRECİ

Bilgisayarda çalışan ilk program BIOS'tur.

1) BIOS Donanımı Kontrol Eder. Bu kontrole POST denir. (Power On Self Test)
2) Önyükleyici (Boot Loader): İşletim sistemini yükleyen kısımdır. Birden fazla işletim sistemi varsa hangisinin açılacağını seçer.
3) Çekirdeğin yüklenmesi: İşletim sistemleri donanımlar ile tanışır.
4) Init (Initialization) süreci: İşletim sisteminde ilk çalıştırılan süreçtir. Diğer tüm süreçleri yönetmekle sorumludur.
5) Başlangıç Hizmet ve Betikleri: Başlangıca koyulmuş prog.
6) Kabuğun yüklenmesi
7) Kullanıcı oturum açma: parola girme, masaüstü simgelerinin gelmesi
8) Kullanıcı Betikleri

4 / 11 / 2015

Sistem Çağrıları (System calls)

Posix(*) uyumlu işletim sistemlerinin programcıya sağladığı arayüzdür. Windows'ta ise API sayesinde sistem servislerine erişilir.

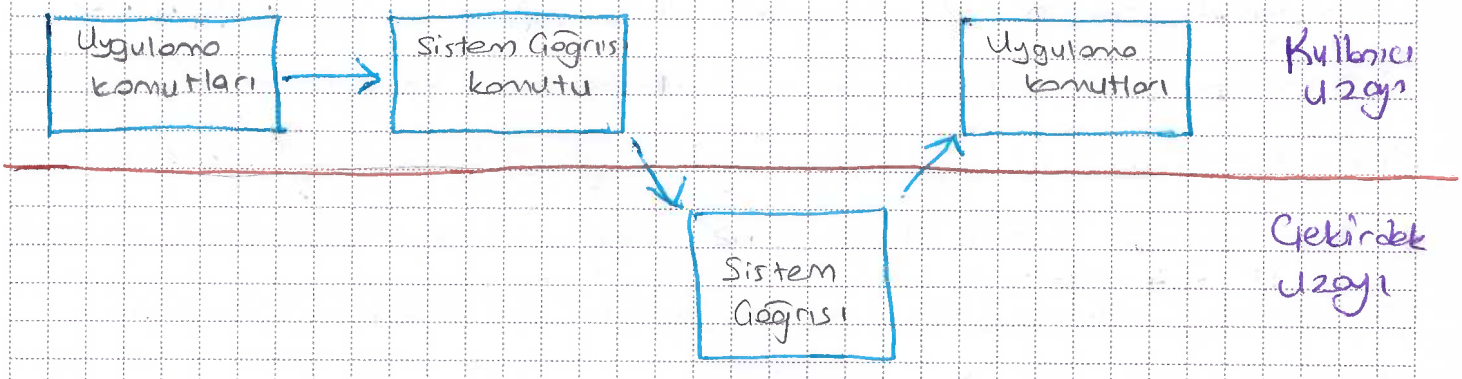
Posix = IEEE tarafından 1988 yılında yayınlanan Portable Operating System Interface (Taşınabilir işletim sistemi) standartlarıdır.

UNIX, Linux (şuan en çok kullanılanlar), BSD, Minix, Solaris, OSX (Apple) gibi işletim sistemleri bunu sağlar. kapsar.

→ BSD = Berkeley Software Distribution lisansı ile çıkartılıyor, yazılan komutların standart olmasını sağlıyor (Posix)

→ free BSD, Open BSD } Bunlara bak. Victure box'ta kur us. gibi birşey dedi.

İşletim sistemlerinde komutların yürütüldüğü kullanıcı ve çekirdek uzayı olmak üzere 2 çekirdek vardır. Sistem çağrıları dışındaki tüm komutlar kullanıcı uzayında yürütülür.



C'de sistem çağrısı yapabilmek için ilgili kütüphaneye initiyor duyulmaktadır. Sistem çağrısı fonksiyon çağırma işlemine benzer. Ancak işletim sisteminin izin verdiği bilgisayarlara çekirdek modda erişim sağlar. En temel sistem çağrıları süreç (process) ve iplik (thread) yönetimi, dosya ve dizin yönetimi, sinyalleşme, güvenlik ve zaman yönetimi çağrılarıdır.

Windows API

LINUX'un sistem çağrıları, windowsta API'ler aracılığı ile çalışan sistem servislerine ulaşır.

C programlarında ilgili API'nin başlık dosyasına yada header (.h) dosyasına yüklenmesi gerekir.

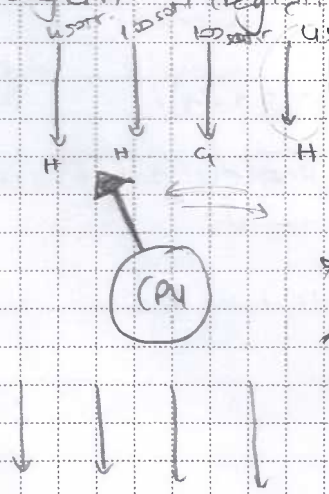
API'lerin hangi başlık to tanımlandığı MSDN dökumanlarında belirtilir.

Süreçler (process)

Yürütülmekte olan her bir program bir processtir.

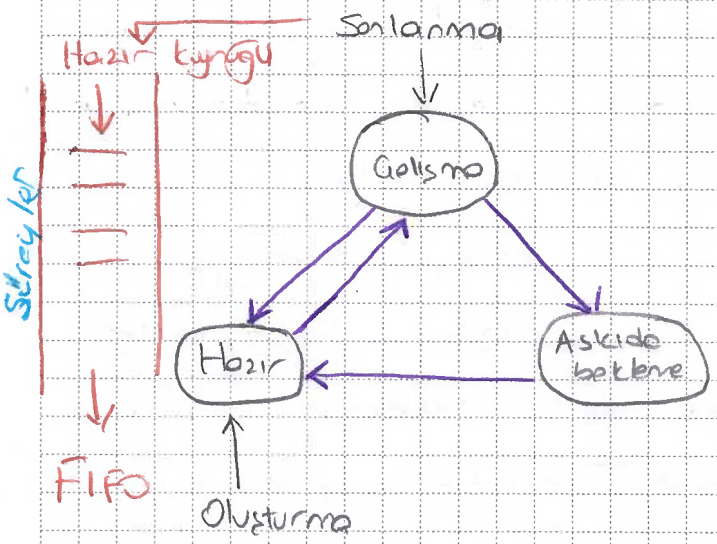
Her processin kendine özel program kodu program sayacı, yığını (stack), veri alanı ve tahsis edilmiş bellek küneşi (HAEP) vardır. Processler hem birbirleri ile hemde dış dünya ile haberleşirler.

Bir işlemcide 1 tane çekirdek varsa 4 tane program sayacı vardır ve gerçek anlamda 4 tane paralel işlem yapabilir. Her çekirdek çoklu işleme tabi tutulduğunda, bu işlemler işlemciyi, program sayacını ve kaydedicileri zaman paylaşımlı olarak kullanılır. Bu processler arası anahtarlı geçişe bağlam değiştirme denir.



* İşlemci tek 1 işlem var gibi de yapıyor.
* İşlemcinin ne zaman hangi işlemi çalıştıracağını işletim sistemi bu nedenle zamanlı işlemler kullanılmalı.

Zaman paylaşımlı çalışmada processlerin CPU, MİB'e ne zaman sahip olacağı kestirilemez. Bu nedenle program kodunda zamanlamaya dayalı işler olmamalıdır. Bu sorunu çözmek için durum modellemesi yapılır. Bir process MİB'e sahipse "çalışıyor" değilse "çalışmıyor" denilir. Processler ömürleri boyunca oluşma, çalışma, bekleme, hazır, sonlanma şeklinde 5 farklı durumda olabilir. Oluşma ve sonlanma durumları geçici hallerdir. Process oluşturulduktan sonra hazır durumuna geçer bu bir kuyruk yapısı olup processlerin çalışma öncesi sıraya alınmasını sağlar.



Processler ile ilgili bilgilerin tutulduğu tablolara process tabloları denir. Süreçlerin gerçekleştirilmesi sırasında işletim sistemi bir kimlik numarası vererek process tablosuna ekler.

Process tablosunda şunlar yer alır;

1) Kimlik:

Süreç numarası (PID) ve anne süreç numarası.

2) Durum:

Hazır, bekliyor ve askıda durumlarından biri yer alır.

3) Program Sayacı:

Sonraki program satırının adresini gösterir.

4) Register (Yazmaç):

İşlemcinin içindeki kaydedilen değerler.

5) Sıralama (İş) Bilgisi:

Süreç önceliklerini tutar.

6) Bellek bilgileri:

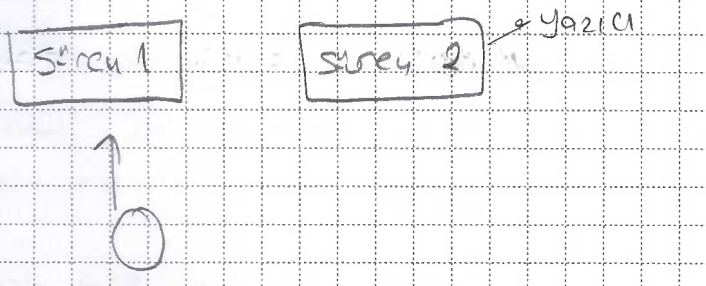
Ayrılmış bellegin taban ve sınır bilgileri, sayfa bilgileri tutulur.

7) İşlemci kullanım bilgisi:

Oluşturulma zamanı, askıya alınma zamanı, işlemci kullanım süresi gibi bilgiler tutulur.

8) Giriş-Çıkış (G/Ç) Bilgileri:

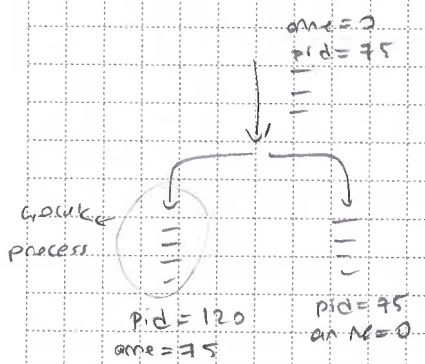
Sürece tahsis edilen G/Ç aygıtları, açılan dosyalar gibi bilgiler.



Process (Süreç) Oluşturma

Bir süreç ancak başka bir süreç tarafından oluşturulabilinir. Bir sürecin birden fazla çocuk süreci olabilir. Ancak her çocuğun tek bir annesi olur. Süreç sonlandırılırken tablo kaydı silinir, Çocukları varsa onlar sonlanana kadar beklenilir yada çocuklar başka süreçe devredilir. Annesi sonlanmış süreçlere yetim(orphan) denir. Process (süreç) eceli ile ölürse "0" değerini döndürür, Aksi durumlarda "-1" döndürür.

Anne sürecin çocuk süreci sistem çağrısı(waitpid) ile beklemediği durumda çocuk process sonlanırsa sonlanma bilgisi kaybolur ve çocuk process artık "zombi process" olur. Zombiler ancak anneleri tarafından gerektiği silinebilirler. Hiç bir zaman waitpid çağrısı yapmayan bir anne process sonlandığında zombi çocuklar, varsa silinir.




```
pid_t sureno;
sureno = fork();
```

çatallaşma (çocuk oluşumu)

```
int sifrecoz (int sirne)
{
  if (anne==0)
    { ---
      waitpid(...) }
  else
    {
      ---
    }
```

* Annenin çocuğunu takip edebilmesi için waitpid'yi beklemesi lazım.
* Çocuk hiç bir şey yapmadan kaybolursa çocuk zombi process oluyor.

Süreçin askıya alınması durumunda tahsis edilen kaynaklarda (bellek, kaydediciler, G/Ç) hasar gelip gelmemesi işletim sistemine bağlıdır. Genelde kısa süreli askıya almalarda kaynaklara el konulmaz. Uzun süreli ise örneğin bellege el konulabilinir.

Durum değişiklikleri mevcut süreçlere işaretlerle üretilir. Windows ve Linux ortamı için bu kütüphaneler sağlanmıştır.

İPLİKLER (thread)

Sürecin askıya alınması durumunda tahsis edilen kaynakların (bellek, kaydediciler, G/Ç) hazır gelip gelmemesi işletim sistemine bağlıdır. Genelde kısa süreli askıya almalarda kaynaklara el konulmaz. Uzun sürecekse örneğin belleğe el konulabilinir.

Durum değişiklikleri mevcut süreçlere işaretlerle iletilir. Windows ve Linux ortamı için bu kütüphaneler sağlanmıştır.

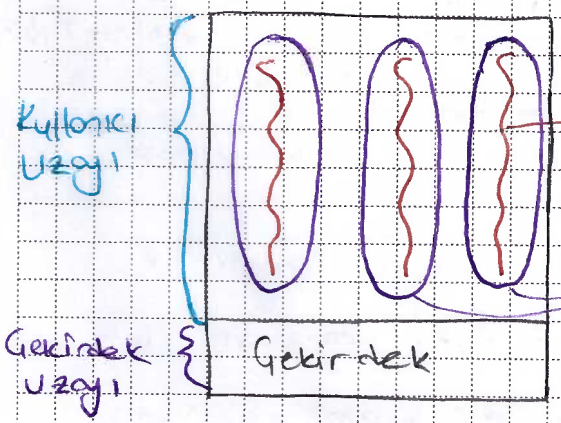
İPLİKLER (thread)

2.12.2015

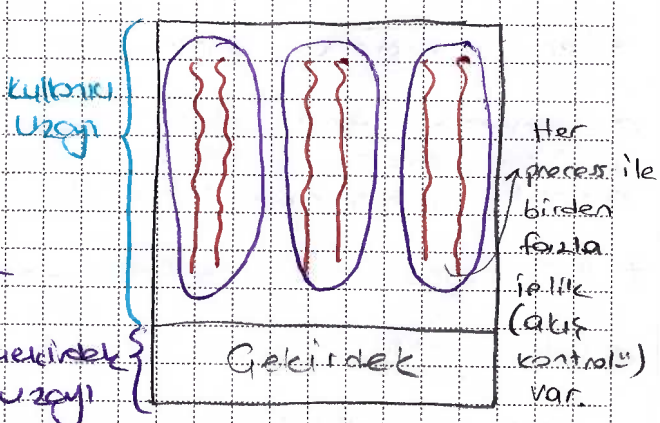
Geleneksel işletim sistemlerinde her sürecin tek akış kontrolü vardır. Süreçler arası paylaşım ve etkileşim ilişkili uygulamalar için çok önemlidir. Çok etkileşimli süreçlerde bu durum zaman kaybı, ölümcül kilitlenme (dead-lock) riski, paylaşım karmaşıklığı gibi problemlere sebep olur.

Bu nedenle, aynı adres uzayında (aynı process ortamında) birden fazla akış kontrolü kullanılmaktadır. İplikler aynı adres uzayında çalışan hafif prosesslerdir. İplikler adres uzayı, bellek, açık dosyalar, global değişkenler gibi kaynakları paylaşırlar. Program sayacı ve kaydedici gibi bilgiler ipliğe özeldir.

Process Modeli



İplik Modeli



"Bir processte meta oluşturduğunda bişey aldığında diğer processlerde etkileniyorsa zaman işletim sistemi process yürütümü yapamıyordur denilir."

Processlerin oluşturulması ve bellek tahsisi maliyetli iken iplikların kaynakları paylaşması nedeniyle oluşturulma ve anahtarlama maliyetleri daha azdır. İplikların arasında koruma yoktur. Yazılım geliştirilirken buna dikkat edilmelidir. Eğer koruma gerekiyorsa iplik yerine process kullanılması daha uygundur. Özetle iplikler işlerin birbirine çok bağlı olduğu birlikte yürütüldüğü ve paylaşıldığı durumda, süreçler ise işlerin birbirinden büyük oranla bağımsız olduğu durumlarda kullanılır. İplik kullanımları çok işlemcili sistemlerde tüm işlemci getirdeklerinden faydalanmayı sağlar

İŞ SIRALAMA

Süreçlerin özelliklerine bağlı olarak kullanılan birbirinden farklı işlemci paylaşım yöntemlerine iş sıralama algoritmaları denir. Bir süreci seçerek işlemciyi ona tahsis (ayırmak) eden fonksiyona kontrolcü (dispatcher) denir. Bir sürecin durdurulup diğerinin başlatılması için geçen süreye kontrolcü geçikmesi denmektedir.

- İş sıralama yöntemi dengeli olmalıdır. (Her süreç eşit zaman ayrılmalı).
- Sonsuz erteleme olmamalıdır.
- Birim zamanda (quantum) en fazla sürece hizmet verilmelidir. (parça denir)
- Etkileşimli kullanıcıların yanıt süresi en aza indirilmelidir.
  - Bekleyen bir iş yaparken kullanıcıya her zaman ekran görüntüsünü vermeli gibi.
- Her istendiğinde hazır olması gereken kaynaklar kontrol edilmelidir.
- Süreçlerin yanıt süreleri göz önünde bulundurulmalıdır.
  - Kimi süreç daha az ise diğer uzun süreliği beklemeleri yerine kısa süreliği hemen yapmalı.
- Öncelik mekanizması olmalıdır.
- Kilit kaynakları kullanan süreçler işini çabuk bitirmelidir.
- Çok yüklü durumda sistem çökmemelidir. Bunu sağlamak için yeni süreç alınmamalıdır veya mevcut süreçlere az hizmet verilmelidir.

4 Eskiden windows'ta bellek dolduğunda mavi ekranı verip kategi yapmış ama artık dolu olsa bile hala çalışıyor.

Sıralama algoritmaları kesintili ve kesintisiz olmak üzere 2 şekildedir.

Kesintisiz algoritmalarda sürecin işi bittiğinde ancak işlemciyi bırakır. Günümüz işletim sistemleri çekirdeğe yönelik süreçlerde kesinti yapmayarak sorunları azaltmaya çalışmaktadır ancak çok süreçli yapılarda verimli olmamaktadır.

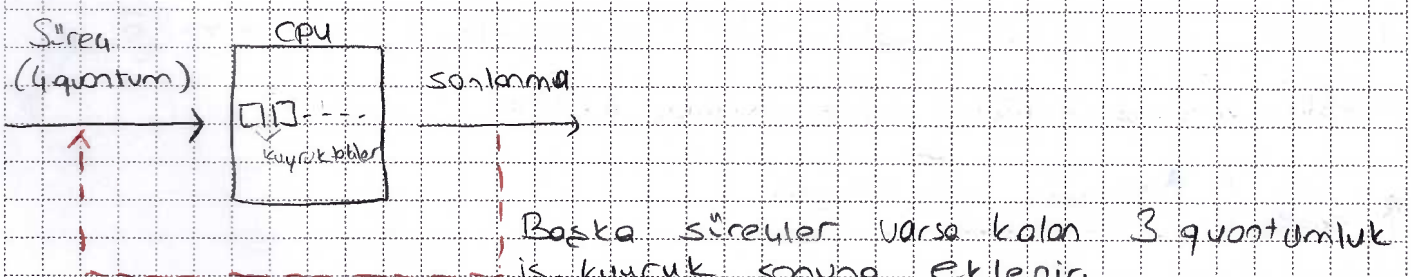
1. FIFO Sıralama : Kuyruk - ilk giren ilk çıkar.

En basit sıralama yöntemidir. Süreçler geliş sırasına göre kesintisiz olarak işletilir. Uzun süreçler yüzünden kısa süreçler gereksiz yere bekler. Etkileşimli süreçler varsa verimli bir yöntem değildir. Genellikle diğer yöntemlerle birlikte kullanılır. Örneğin; aynı öncelikli süreçlerin kendi aralarında sıralanmasında kullanılır.

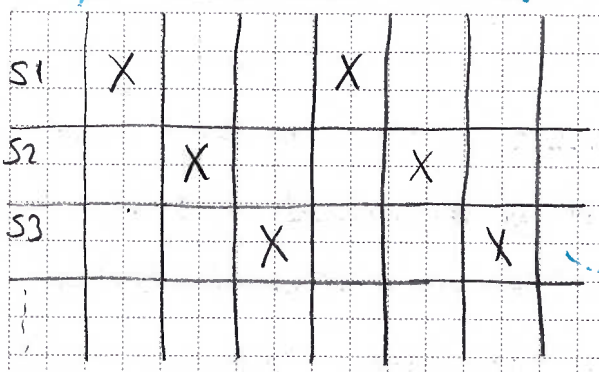
2. Round - Robin Sıralama :

Dönüşümlü sıralama olarakta bilinir. Kesintilidir. İşlemci belirli zaman dilimleri ile süreçlere paylaştırılır. Süreçlerin işlemciyi kullandıkları en küçük zaman dilimine quantum denir.

Aktif süreç quantum süresinde bitmez ise yeniden kuyruğun sonuna aktarılır. Quantum süresi büyük seçilirse işlemcinin yatmasına sebep olur. Çok küçük seçilirse bağlam değiştirme yükü nedeni ile verimsiz olur. Genelde bir quantum milisaniyeler dilimindedir.



CPU quantum zamanı



3.) En kısa iş en önce :

Kesintisiz bir yöntemdir. FIFO'ya göre ortalama bekleme süreleri azalmıştır. Bu yöntemin problemi her sürecin ne kadar süre işlemci kullanacağının bilinmemesidir. Bunun için tahmini süre verilebilir. Tahminden daha uzun sürerse kesilir ve daha sonra çalıştırılır.

Üretim (production) ortamlarında önceden de uygulanmış süreçlerde bu süre tahmin edilebilir. Ancak geliştirme (develope, test) ortamlarında bunu bilmek zordur. (Bir iş yaparken onu bitirmeden başkasına geçmez.)

Not = Burada kullanılan üretim canlı montajında kullanılmıştır. Fabrika gibi değildir.

4.) Kalan süresi en az olan :

3. yöntemin kesintili halidir. Yeni gelen süreçlerde göz önüne alınarak sonlanmasına en az kalan süreç önce çalıştırılır.

Çalışan süreç yeni gelen kısa süreç nedeniyle kesintiye uğrayabilir. Süreçlerin işlemci ihtiyacı bilinmemesi dezavantajıdır.

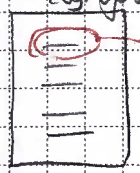
5.) Yanıt Süresine göre :

Önceki yöntemdeki uzun işlerin çok beklemesi problemini düzeltme amaçlı bir yaklaşımdır. Kesintisizdir. Sürecin önceliği hesaplanırken beklediği süre hesaba katılır.

$$\uparrow \text{Öncelik} = \frac{\uparrow \text{Bekleme} + \sum \text{ihtiyaç}}{\sum \text{ihtiyaç}}$$

6) Çok düzeyli kuyruklar:

Tek düzeyli yapılarda aktif süreç örneğin girişi/çıkışı beklerse diğerleride o süreci beklemektedir. Buna çözüm olarak çok düzeyli kuyruklar kullanılmaktadır. Bu yöntemde bellek büyüklüğü, süreç önceliği ve süreç türüne göre ayrı öncelikte kuyruklar bulunmaktadır. Her kuyruğun kendi sıralama algoritması bulunur. Her kuyruğun daha alt düzey kuyruğa göre önceliği vardır. Kuyruklar arasında zaman paylaşımı veya benzer bir yöntem ile işlemci paylaşımı yapılmaktadır.

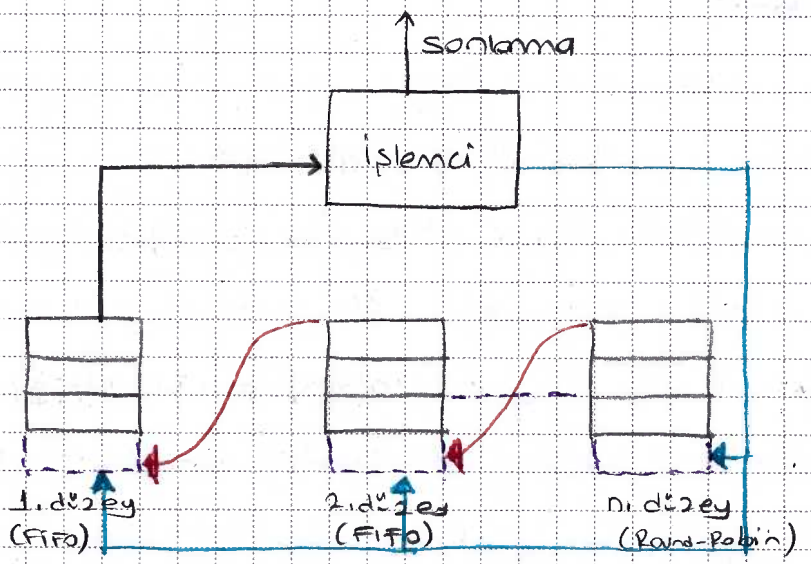


İşlem süresi bunda fakat bu bir indirme vs. gibi bir iş yapıyor yani bekliyor bu bekliyor diye diğer süreçlerde beklemek zorunda kalıyor bunu önlemek için bu sıralama kullanılıyor.

Süreç	Önce-1	Önce-2	Önce-3
k Yaklaşık bellek			
l öncelik			
m Türü			
⋮			

7) Çok düzeyli geri beslemeli kuyruklar:

FIFO ile Round-robin hibrit yapısına benzer. Süreçler işlemci kullanımına göre kuyruklara yerleşirler. ve işlemciyi her kullandığında sonraki kuyruğa aktarılır. Genelde her düzeyde FIFO kullanılır. En alt düzeyde ise Round-Robin kullanılır. Bir kuyruktaki süreç üst kuyruk tekiler bitmeden çalışmaz.



8) Çok İşlemcili Sıralama:

İkiye ayrılır.

a) Asimetrik Çoklu İşlem (AMP)

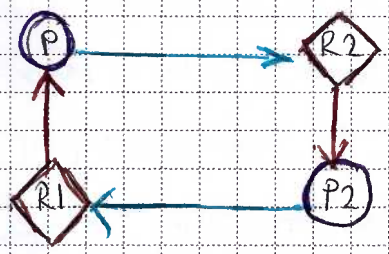
G/Ç işlemleri ve diğer sistem tabanlı işlemler tek bir işlemcide, diğer programlar ise diğer işlemcilerde çalıştırılır.

b) Simetrik Çoklu İşlem (SMP)

Modern işletim sistemleri SMP çalışır. Her işlemcinin kendi sıralama algoritması bulunur. Her işlemci her işi yapabilir. Bütün süreçler tek bir hazır kuyruğunda olabileceği gibi her işlemcinin kendi hazır kuyruğunda olabilir. İşlerin işlemcilere dengeli paylaştırılması işletim sisteminin görevidir.

09 / 12 / 2015

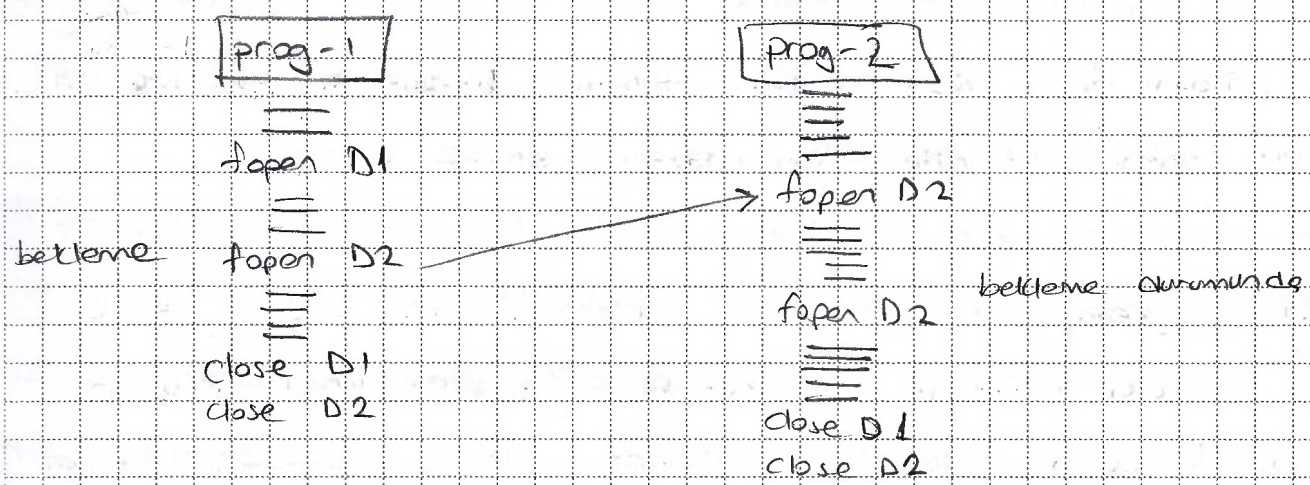
Ölümcül Kilitleme (dead lock)



turuncu ok = tahsisler
mavi ok = bekleme
Mor lar = süreç (process)
turuncular = kaynak (re source)

Çok görevli ortamlarda süreçler sınırlı kaynaklar için (Bellek alanı, işlemci, dosyalar...) yarışırlar. Bir süreç kaynak alamadığında bekleme durumuna geçer. Kaynağı kullanan süreçte bekleme durumuna geçerse süreçler sonsuza kadar bekleme durumunda kalıp kilitlenebilirler.

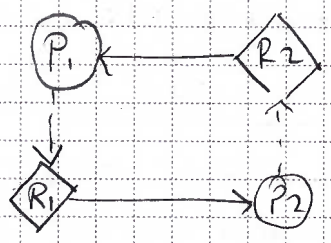
Processler ellerindeki ..... kaynakları ölümcül



Çoklu iplikli uygulamalarda paylaşılan kaynaklar ölümcül kilitlenmeye zemin hazırlar. Modern işletim sistemlerinde geliştirilen yazılımlarda ölümcül kilitlenme denetimi yazılımcıya kalmıştır. Ölümcül kilitlenmeden kaçınılmak için kullanılan algoritmalar aşağıda açıklanmıştır.

1) Kaynak Tahsis Grafı (Çizge)

Her kaynaktan birer adet olduğunda tercih edilir. süreçlerin gelecekte ihtiyaç duyacağı kaynaklara noktalı çizgiyle bağlantı gösterir. Süreçlerin kaynak istegi karşılandığında çevrimsel kuyruk (cyclic quene) olunup olunmadığına bakılır. Tehlike varsa süreç bekletilir. n tane süreç için n^2 işlem yapılır.



2) Banker Algoritması

Her bir kaynaktan birden fazla varsa tercih edilir. bankalar ellerindeki nakit parayı tüm müşterilerinin potansiyel ihtiyaçlarını garanti altına almadan tahsis etmezler. Algoritmada bu şekilde çalışır.

Basdan her süreç için her kaynaktan kaç tanesine ihtiyaç duyacağı kaydedilir. Bu işlem için matrisler kullanılır. Sürecin istediği kaynak sayısı süreci tehlikeye sokacaksa süreç belirtilir. P-süreç sayısını R-kaynak sayısını gösterirse 3 tane PxR büyüklüğünde matris kullanılır. Oldukça maliyetli bir algoritmadır.

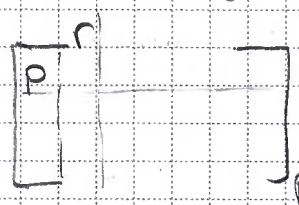
P: süreç sayısı

r: kaynak sayısı

1) maksimum kaynak (pxr)

2) Atanmış kaynak (pxr)

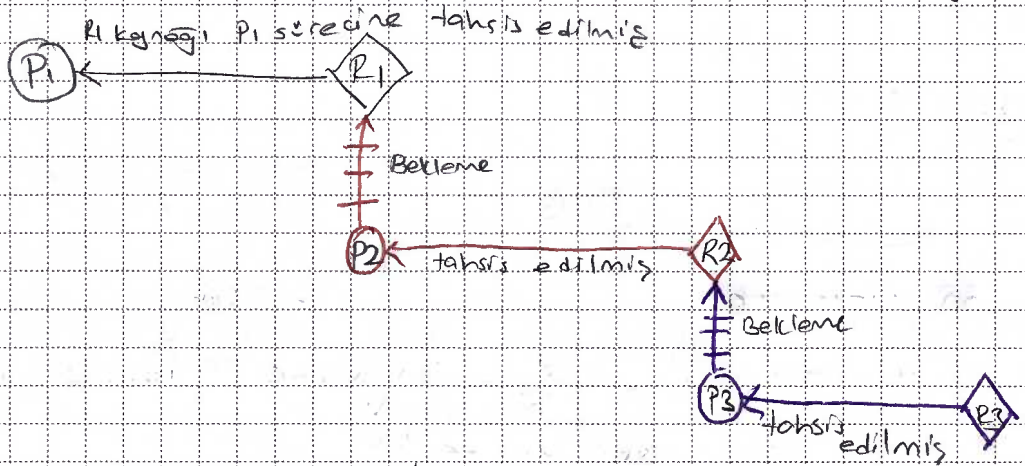
3) Kalan kaynak istegi (pxr)



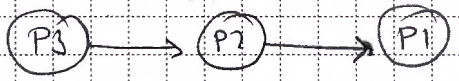
Algoritmanın dezavantajı süreç çalışmadan önce tüm kaynak isteklerini bildirmelidir. Ayrıca süreç ve kaynak sayısı değişmelidir.

3) Bekleme Grafı

Kaynak tahsis graflarında kaynak düğümlerinin atılması ile elde edilir. Her kaynak türünden bir tane bulunan tercih edilir fine çevrimsel kyrık oluşup oluşmadığına bakılır.



Kaynaklar devre dışı bırakıldığında



SOKET PROGRAMLAMA

Soketler hem aynı bilgisayar üzerindeki süreçlerin hem de farklı bilgisayardaki süreçlerin haberleşmesi için kullanılmaktadır. HTTP, FTP, SMTP, Telnet vb uygulamalar soketleri kullanarak haberleşirler. Soket kullanımı işletim sistemine bağlı olmadığından

---

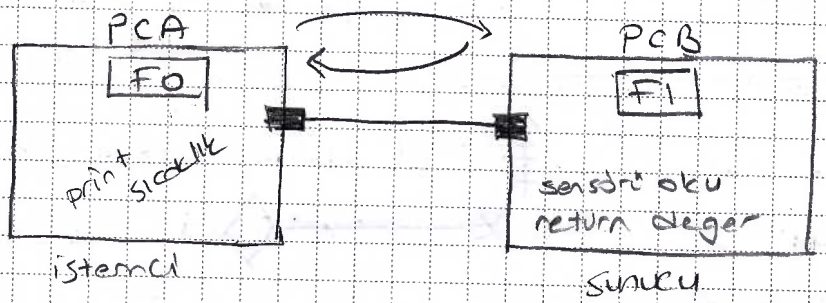
işletim sistemleri birbirleri ile haberleşirler. Soketlerin ip adresi port numaraları bulunur. Genellikle istemci sunucu mimarisinde kullanılır. Sunucu portları dinleyerek yeni bir istemci bağlantısı bekler. Genel uygulamaların port numarası bellidir.

Genel uygulamalar 1024 küçük port kullanılır. Toplamda $2^{16} = 65536$ tane port vardır. 1024'den küçük olanlar ise özel amaçlı kullanılmaktadır. Bütün bağlantılar aynı bir soket bağlantısı ile tanımlıdır.

```
int sicaklik (int sensor_id)
{
  return (deger (sensor_id))
}
main
{
print --- sicaklik (3);
}
```

f1

f0



TCP/IP içinde 3 tane protocol var

- TCP
- UDP
- ICMP

Dinlediği portları

netshat - antulp

Windowsta soket programlama için winsock API

Linuxda da sys(socket.h) soket programlamada kütüphanelerde içerir.

Dosya Sistemi

Bilgisayarda veriler ikincil olarak adlandırılan fiziksel birimler üzerinde depolanır. Bunlar elektrik kesintisinde silinmezler. Birbiri ile ilgili verilerden oluşan koleksiyona dosya denir. Dosyalar işletim sistemleri için saklama birimleridir. Dosya isimleri nokta ile ayrılan iki bölümden oluşur. Bunlar dosyanın adını ve türünü belirtir. Dosyalar kullanıcılar tarafından isimleri ile tanımlanır. İşletim sistemi ise bunun için özel tanımlayıcılar kullanır.

Disklerde veriler kullanılacağı zaman belleğe yüklenir. Bu aktarım bloklar halinde yapılır. depolanmış verilere kolay ve etkin erişebilmek için işletim sistemleri dosya sistemlerini kullanır. Windowsta genellikle NTFS kullanılır. Linuxda genellikle Ext4 kullanılır. FAT dosya sisteminde en eskilerden olduğu için yaygındır. Bu nedenle flash tercih edilir.

RAID
n tane disk dinamik
RAID-1 = Aynalama-yansılama (mirror)
Tüm disklerde aynı veri var.
RAID-5 = 1 tane yedek / (n-1) tane veri
Örnek 6x1 TB disk net = 5 TB olur.

AĞ DOSYA SİSTEMLERİ (Network File System)

Ağ üzerinden başka bir bilgisayardaki disk alanını yerel kaynakmış gibi kullanmaya yarayan sistemlerdir. Bunun için öncelikle uzaktaki kaynak işletim sistemine sanal olarak bağlanmaktadır. (mount) En yaygın kullanılanları:

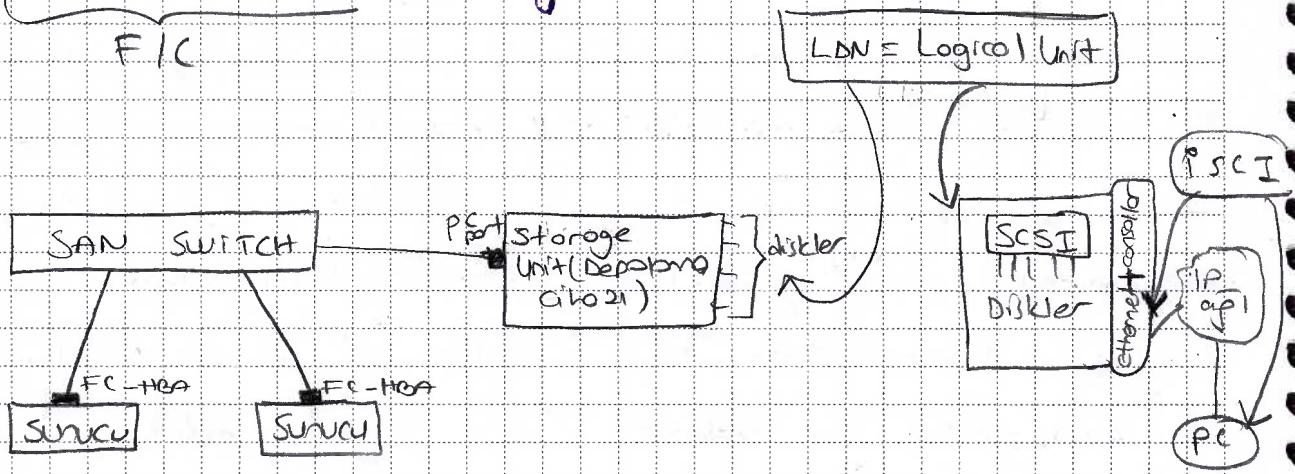
Win : CIFS

Linux Unix : NFS

Bu konuda güncel bir teknolojide issCI bağlantısıdır.

Fibre Channel Storage

FC

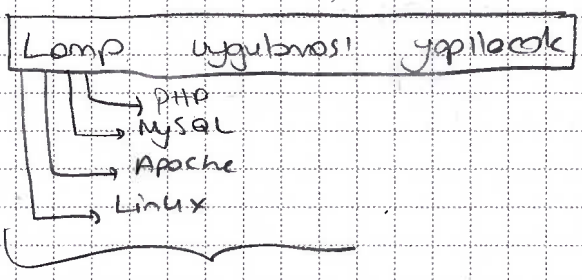


SAN = Storage Area Network

HBA = Host Bus Adapter

Ödev:

Virtual Box Ubuntu Server (son sürüm)



Tüm paketler Ubuntu deposunda kurulacak

Bugün önemli

Zaman Çizelgesi — Dağıtımlar | 16/12/2015

1) Linux Timeline Distro

2) Debian'a en tam Linux Dağıtımı Slackware → Az kaynaklı son kullanıcıya ağırlık verilir.

3) Distro watch → Dağıtımların kontrol edilmesini sağlayan bir sitedir.

Ubuntu:

Linux OS

1992 yılında Linus Torvalds tarafından Linux'un duyurulması öğrenci projesi olarak başladı.

1983 Stallman GNU (ve GPL) yayınlandı.

falsete GNU: GNU is not Unix

GPL: General Public license

Neden Ubuntu ve Ubuntu Nedir?

Ubuntu Debian tabanlıdır. Debian günümüzde halen en kararlı ve güvenli Linux dağıtımlarından birisidir. (Debian'a benzetme olarak 12 bacaklı sandalye denmektedir).

Ubuntu kullanmamızın nedeni ① güncel olması, ② desteğin yaygın olması, ③ hem sunucu hemde son kullanıcı için uygun olmasıdır.

Ubuntu seveneklerı: Kubuntu, Edubuntu, Xubuntu, Ubuntu Mate vb. türevleri vardır.

Desktop Server → Ubuntu Seveneklerı

32/64 bit Seçeneği →

LTS ve Normal (Long Term Supported) →

Yeni sürümler sene de iki defa Nisan ve Ekim aylarından çıkar 15 ve 10 bacaktan gelir.

LTS, serverlerde tercih edilir. Sürekli güncellemeyi önler.

15.10 Linux 2015'in 10. ayında çıkmıştır.

Sanallaştırma Platformları

- Virtual Box
- VmWare
- Hyper-V
- Xen
- Qemu
- OpenVZ → Proxmox (Sunucu Tarafında) kendi grafik arayüzü yok
  Container → Linux üzerinde çok iyi bir şekilde verim almamızı sağlar.

- Hangi portları dinliyorum
  netstat -antlp

Sanal Makine Kurulumu

32/64 bit makine ayarlama

Ön yükleme sırası

VBida Network (NAT/Bridge) Host: Fiziksel (Gerçek) PC/Guest: Sanal PC

NAT avantajı, host için kullanılan bir ip adresini, tüm guestler tarafından kullanılabilir hale getirir. Guest bilgisayarların internete çıkması için, fiziksel dünyada veya izin gerekmez. En kolay yapılandırmadır.

NAT dezavantajı, Guestler kolaylıkla internete çıkar ama fiziksel dünyadan guestlere erişim mümkün değildir.

Bridge (köprü) Avantajı, Guestlerin tamamı, fiziksel dünyada birer host gibi davranır. Host ile aynı ağdan ip adreslerine sahip olur. Fiziksel dünyadan tüm guestlere erişilir.

Bridge dezavantajı, Fiziksel dünyada, guestler için ip gereksinimi olur. yada izin olması gerekmektedir.

NAT kullanıldığında Port yönlendirmesi

TCP 80 → Web sunucusu için

TCP 22 → SSH bağlantısı için

Paket arama apt-cache search ___ aranacak olan paket girilecek.

Sudo apt-get install php5-sybase → Bu paketi kur

update, upgrade, install remove, purge

dpkg_query

↓

debian package query (debian paketi sorgulama)

dpkg-query-l

-L <paket_ismi> (zenity mesela)

netstat_antulp

↳ Çalışır, dinlenen portları görebiliriz.

Service --status-all → tüm çalışan/çalışmayan servisleri gösterir.

Service cups stop // dersek cups bu servisi durdurur.

Onasso başına sudo ekleriz.

↓

Supier user dd

history → yapılan işlemleri gösterir.

* ps aux → görev yöneticisi gibi

↳ processing

Temel komutlar

pwd = hangi dizindeyiz

ls = dosya ve klasörleri listele (-)sah parametresi ile daha güzel çıktı verir)

cd /indirilenler = İçeriğe girer

whoami → ben kimim?

w → who → sistemde kimler var?

uname → sistem bilgisi uname-a // detaylı görünüm.

(nano) pico → metin editörü

Dosya = kütük
dizin = klasör

Ödev Sırasında İşlem Aşamaları

- Blogta yazılanları hallet
- Baştan lamp yapmamışsak apache, mysql, php'yi depodan yükle
  Dökümanı oku!
- İndireceğin kişisel uygulamayı (muhtemelen tar.gz) sunucusunda /var/www (veya /var/www/html) klasörüne gönder.

```
wget http://.../paket.tar.gz
tar xzvf paket.tar.gz
```

- phpmyadmin arayüzüne bağlan, yeni bir kullanıcı ve onaid db oluştur.
- /var/www/... altında ilgili ödev uygulamasını açtığın klasörü web üzerinden çağır.

Örnek http://192.168.1.3/yazılım_klasörü
/phpmadmin

- Geri kalan tüm işlemler Web üzerinden yapılıyor.

cheat sheet
Özet
başvuru sayfası
sayfada küçük küçük bilgiler

Önemli Klasörler (Linux'ta) → sınavda

bin → binary'nin kısaltması dır. Makine diline çevrilmiş dosyalardır. Çalıştırılabilir dosyaların (genellikle komutların) bulunduğu dizin dir - klasördür.

boot → (Açılış) Sistem açılış dosyaları burada durur

dev → (Aygıt) = (device) Burada gerçek dosyalar bulunmaz. İşletim sistemi ile konuşmasını gerçekleştiren donanım bileşenlerinin linki bulunur.

etc → Sistemin ve sunucuların (servislerin) yapılandırma dosyaları burada durur.

home → Standart kullanıcıların kişisel klasörüdür.

media, mnt → Geçici depolama aygıtları dır.

usr, opt → Windows'taki program dosyalarının karşılığıdır.

proc → Gerçek dosyalar değildir. Donanımlar hakkında bilgi almak için kullanılan sanal dosyalardır.

root → En yetkili kullanıcının (root) kişisel ev klasörüdür.

sbin → Sadece root'un çalıştırabileceği binary dosyalar.

var → Sistem kayıtları (log'lar), kuyruklar (e-posta, yazıcı vb), web sayfaları vb. verilerini tutulduğu klasördür.

DOSYA KLASÖR YETKİLERİ → Sınavda

Orjinal:
drwxr-w-xr-w-x

Bölünmüş:
d rwx rwx rwx
(user) (group) (others)

İlk baştaki d klasör olduğunu ifade eder.

r: read yetkisi
w: write yetkisi
x: execute yetkisi (çalıştırma)
d: klasör yetkisi (directory)
l: link yetkisi

3-lü gruplar:

1. Sahibi (user)
2. Grubu (group)
3. Diğerleri (other)

Not: 3lü grup bre ayırıyor. Yetki vermek istediklerine yetkiyi user veriyor.

rwx → bunun yazılımı standarttır, r varsa başta olarak tır. eğer yoksa -wx, yada r-x 3'lü grup için böyle olmalı zorundadır.
2^2 2^1 2^0

Bizim için önemli olan sayı değeridir.

Örn Kullanıcısına tüm yetkileri verin. komutu yazınız.
Çözüm Kullanıcısında rwx olmalıdır. → Sınavda

İkilik karşılığı: 111 olur. Bunu da onluğa çevirirsek

Onluk karşılığı: 7'ye tekabül eder.

Yetkiler ifade edilirken kullanıcı grup ve diğerleri için, onluk değerleri ile ifade edilir.

pişirirsek de böyle

Örn kullanıcısı, grubu ve diğerleri tam yetkili olsun;

Çözüm rwx rwx rwx → 2'lik

7 7 7 → 10'luk

Örn kullanıcı tam yetkili, Grubu sadece okusun. Başka yetki yok

Çözüm rwx r-- --- → 2'lik

7 4 0 → 10'luk

Örn kullanıcı okuma + yazma;

Grubu okuma + yazma;

Diğerleri hiç

Çözüm rw- rw- -- → 2'lik

6 6 0 → 10'luk

Komut ile yetki değiştirme

Yetki değiştirme

chmod [-R] <onluk_yetki_numarası> <dosyalar>

Herşeyi yapması için (chmod)

[-R]: Seçimliktir, yazma zorunlu değildir. <dosyalar>: için yazma zorunludur.

– (rekürsif) alttaki tüm klasörleri de etkileyecektir.

Örn rw rw r olanı değiştirelim

chmod 600 deneme.txt yaparsa

rw------- yapar.

Örn rw- --- --- olanı değiştirelim.

chmod 444 (*) → hepsini

r-- r-- r-- yapar.

Not Klasörlere girilebilmesi için "x" yetkisi mutlaka olmalıdır.

Adres Tarifi

Temel olarak 2 tip adres tarifi vardır. Bağıl ve Mutlak

İşlem yapacağımız klasör nerde sorusunun cevabı adres tarifi ile halledilir.

1. Mutlak Adres Tarifi:

Ana dizinden itibaren yolun tamamı yazılır.

Bulunduğumuz klasöre bağlı olmadan çalışır.

Örn: /var/www/html/index.html

Ana dizin } Mutlaka başta / olacak

2. Bağıl Adres Tarifi:

Bulunduğumuz yere bağlı olarak yapılan tariflerdir.

Örnekler:

- html/index.html
- chmod 660 ../www/*
- sudo cp ~/latest.tar.gz ./ → sudo root etkisi ile çalışıyor cp → kopyalama

Bu klasör altındaki latest. klasörü bulunduğun klasöre kopyala.

- ls -lsa html/
- ls -l ../../../ → 3 üstteki klasör içeriğini listeler.

* 3. de kullanılan ~ işareti ⇒ kullanıcının kendi ev klasörünü ifade eder.

Örnek: ls -l ~/ → ev klasöründeki herşeyi listele.

* ./ → bulunduğumuz klasör demek.

SÜREÇ YÖNETİMİ (PROCESS)

ps → çalışan süreçleri gösterir. (Benim çalıştırdığım)

ps aux → PC'de çalışan tüm süreçleri gösterir.

CTRL + C (^C): Çalışan aktif süreci sonlandırır.

CTRL + Z (^Z): Çalışan aktif süreci bekleme konumuna alır.

bg : Bekleme durumundaki süreci arka planda çalıştırır (background)

fg : Bekleme durumundaki süreci ön planda çalıştırır (foreground)

jobs : Arka plandaki ve bekleme durumundaki süreçleri listeler.

Özel karakterler

& (ve) = Bir komutun en sonuna yazıldığında, doğrudan o komutun arka planda çalışmasını sağlar.

&& = Birden fazla komutun arasına konarak, onları sıra ile çalıştırmayı sağlar.

Örnek: mkdir bilecik && mkdir bilecik/osmaneli

| (boru, pipe) = Bir komutun çıktısını, diğer komuta girdi olarak aktarır.

Örnek: cat /var/www/html/index.html | grep META

- cat → dosyanın içeriğini ekrana döker
- /var/www/html/index.html → Bu dosyanın içeriğini | ile sağdakinde meta olanları bul dedirtiyor.
- grep META → içinde META geçenler ...

> (büyüktür) = Bir komutun çıktısını dosyaya yazar. Dosya yoksa oluşturur. Varsa, silerek yazar.

>> (çift büyüktür) = Komutun çıktısını dosyaya EKLER (append). Dosya yoksa oluşturur.

Ör: ls > dosyalar.txt

dpkg -l > programlar.liste

* ve ? = Dosya ve klasör isimlerinde bilinmeyen karakterler yerine kullanılır.

? → 1 karakter , * → sınırsız karakter.

Örnek:

- pico anta?ya.txt

antalya
antakya

- ls *.html → adı ne olursa olsun listele
- tar -xvzf /test.tar.gz → dosyayı açtık

magenta

sudo cp -prf magenta/* /var/www/html/

- ls -l ~/.*

ls → listele, -l → uzun formatta listele, ~ → ev klasörüm, .* → nokta ile başlayanlar → Nokta ile başlayanlar gizli dosyalardır. ls ile gözükmezler. ls -al ile gözükür.

30.12.2015

CHOWN komutu

Dosya ve klasörlerin sahipliğini değiştirmek için kullanılır.

Kullanımı: (alt klasörler dahil)

chown [-R] <kullanici:grup> <dosya(lar)>

Örnekler

sudo chown -R www-data /var/www/* → kullanıcısı, bunun altındaki herşeyin sahipliğini verdi

chown mehmet ödev.txt → ödev.txt'nin sahibi artık mehmet.

chown mehmet.bm odevler/*.doc → sahibi mehmet, grubu bm olan; ödevler klasörü altında uzantısı doc olanların; } hepsini mehmet.bm sahibi oldu

GREP Komutu

Metin filtreleme işlemi yapar.

Örnek kullanım alanları:

- /var/log/syslog dosyasında içinde error geçen satırları göster.
- /etc/apache2/conf.d klasörü altında wordpress geçen satırları göster.
- Başında # olmayan satırları listele (RegEx)
- ödevler.txt dosyasında kaç tane e-posta adresi var? (RegEx)

Kullanımı:

grep [ix...] <metin> <dosya(lar)>

Parametreler:

-i (ignore case): Büyük - küçük harf duyarlı değil

-r (recursive): alt klasörler dahil

-v : belirtilen metni içermeyenler

Örnekler:

- grep error /var/log/syslog
- grep -ir wordpress /etc/apache2/conf.d
- grep -v "^#" /etc/apache2/security.conf

REGEX (Regular Expression ~ Düzenli ifadeler)

Karakter ve katar eşleştirme işlemleri için kullanılır. Tüm programlama dilleri içerisinde desteklenmektedir. Örnek kullanım alanları:

- e-posta adreslerini bul
- girilen telefon numarası doğru mu?
- Tarih formatı doğru girilmiş mi?
- Metin içerisindeki tüm IP adreslerini keşfet.

Tek dezavantajı yavaş olması.

^ → satırın başında
$ → satırın sonunda
\d → sayıları bulur

cheat sheet regex → google'da aratılır

CRON (Görevli Zamanlayıcı) → sistemde mevcuttur.

Linux'ta herhangi bir komutu (yada betiği, programı) istenilen zaman(lar)da otomatik olarak çalıştırmak için kullanılır.

Kullanımı

crontab [-el]

e : (editör) düzenleme modu

l : (list) görüntüleme modu

Zamanlama yazım biçimi

* * * * *

m h dom mon dow

m : minute

h : hour

dow : day of week

dom : day of month

mon : month

*Zaman belirteci olarak kullanılabilecek ifadeler :

- * "her" anlamındadır.
- /N "N sayısına göre mod alır"
- @hourly, @weekly, @monthly, @daily, @yearly

Örnekler

0 * * * * her saat başı

/5 * * * * 5 dakikada 1

15 23 * * * her gün 23:15'te

0 8 * * 1 her pazartesi 08:00

↓ Herhangi 1 gün

* * * * * → her dakika çalışsın.

* Her 31 Aralıkta 23:59'da

59 23 31 12 *

* Her ayın ilk günü (Sadece saati çalışacak saat tablosu ayarını biz yapacağız.)

0 0 * 1 *

* Her gün

@daily

* Sadece hafta içi her sabah 8'te

0 8 * * 1-5

Sırada bunların tam tersi alacak

1) Bulunduğumuz klasördeki uzantısı pdf olan tüm dosyaları listele

ls *.pdf

2) Kullanıcının kendi ev dizinindeki "bilecik" isminde klasör oluşturan komutu.

mkdir ~/bilecik