İSTANBUL ÜNİVERSİTESİ
AÇIK VE UZAKTAN EĞİTİM FAKÜLTESİ

İŞLETİM SİSTEMLERİ

DR. ÖĞR. ÜYESİ SÜMEYYE KAYNAK
DR. ÖĞR. ÜYESİ BARAN KAYNAK

İŞLETİM SİSTEMLERİ

DR. ÖĞR. ÜYESİ SÜMEYYE KAYNAK
DR. ÖĞR. ÜYESİ BARAN KAYNAK

Genel Yayın Yönetmeni
Prof. Dr. Levent Şahin

Kapak Tasarım
AUZEF

Grafik Tasarımı
AUZEF

Sayfa Tasarımı
Burhan Maden

ISBN: 978-605-07-1171-4

2022

İçindekiler

1. İŞLETİM SİSTEMLERİ-GİRİŞ

Bilgisayar yapmasını istediğinizi yapacaktır, ancak sonuç aklınızda olandan çok farklı olabilir.

Joseph Weizenbaum

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Bilgisayar sisteminin bileşenlerini öğrenebilir.
- İşletim sisteminin yapısını ve çalışma prensiplerini öğrenebilir.
- İşletim sisteminin temel görevlerini ve bu görevleri yerine getirmek için gerçekleştirdiği aktiviteleri öğrenebilir.
- Popüler işletim sistemi türlerini tanıyabilir.

Başlamadan Önce

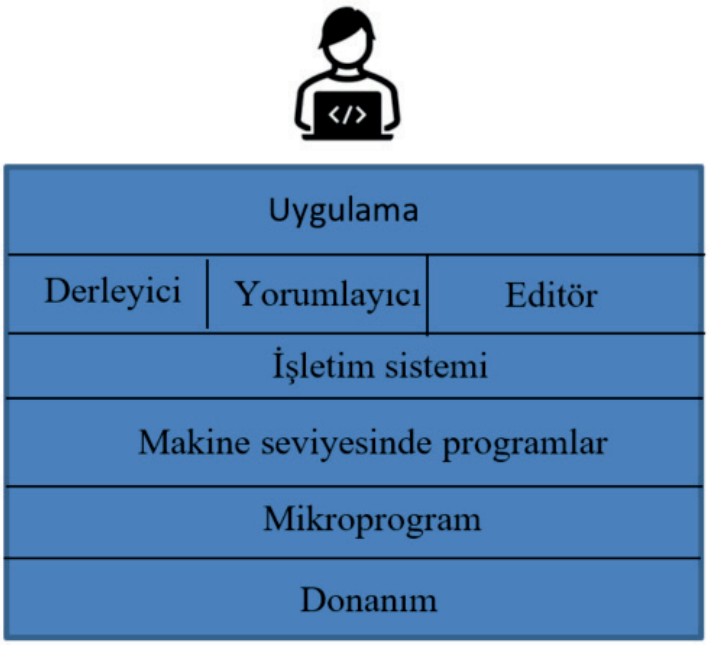
Farklı amaçlar ve mimari yapılar için tasarlanmış birçok işletim sistemi yazılımı geliştirilmiştir. Tüm bu farklılıklara rağmen işletim sistemi yazılımlarının üstlendiği ortak görevler bulunmaktadır. Bu bölümde bilgisayar sistemini oluşturan bileşenler, işletim sisteminin bilgisayar sistemindeki yeri ve önemi anlatılacaktır. Daha sonra, işletim sistemi yazılımlarının üstlendiği görevler incelenecek ve işletim sisteminin bu görevleri yerine getirmek için gerçekleştirdiği aktiviteler anlatılacaktır. Ayrıca yaygın olarak kullanılan işletim sistemleri türleri, bu türlerin kullanım alanları, avantajları ve dezavantajlarına da yer verilecektir.

Birlikte Düşünelim

1. İşletim sistemleri olmasaydı bilgisayar dünyasında neler farklı olurdu?
2. İşletim sistemleri olmasaydı bilgisayar sistemlerinde meydana gelen gelişmeler bu kadar hızlı gerçekleşebilir miydi?
3. İşletim sistemlerinin farklı türlerinin geliştirilmesine neden ihtiyaç duyulmuştur?

1.1. Bilgisayar Sistemleri

Bir bilgisayar temel olarak "yazılım" ve "donanım" bileşenlerinden oluşur. Bilgisayar sistemleri, yazılım ve donanım bileşenlerine ilaveten "kullanıcı" ve "ağ" bileşenini de içerir. Çizim 1'de görüldüğü gibi bu bileşenler kendi içerisinde de farklı çeşitlere ayrılmaktadır. Her birinin farklı bir hizmeti vardır. Bilgisayar sisteminin bileşenlerini tanıyalım:



Çizim 1. Bilgisayar Sisteminin Bileşenleri

Kullanıcılar: Uygulama yazılımlarına doğrudan erişim sağlayan, bilgisayara komutlar veren bilgisayar bileşenlerinden biridir.

Donanım: Merkezi işlem birimi (CPU), bellek, monitör, fare, yazıcı, ekran kartı gibi fiziksel birimler bilgisayarın donanım bileşenini oluştururlar. Bilgisayarla entegre olan, kasanın içinde yer alan donanım birimlerine dahili donanım birimleri adı verilir. CPU, ana bellek, ekran kartı, ses kartı gibi birimler dahili donanım birimlerine örnektir. Kasanın dışında yer alan donanım birimlerine de harici donanım birimleri adı verilir. Fare, klavye, ekran, yazıcı, tarayıcı gibi birimler harici donanım birimlerine örnektir.

Yazılım: Donanım birimlerinin mekanik işlevlerinin bir anlam kazanabilmesi ve donanım birimlerinin düzgün çalışabilmesi için yazılımlara ihtiyaç duyulur. Bilgisayarda kullandığımız her türlü programa yazılım denmektedir. Yazılımlar farklı amaçlara hizmet etmek amacıyla tasarlanıp geliştirilmiştir. Yazılım klasik yaklaşımla 2 ana başlıkta sınıflandırılabilir.

- Uygulama yazılımları
- Sistem yazılımları

Uygulama yazılımları: Sistem yazılımları ile uyumlu çalışan, programcının bir problemini çözmek amacıyla geliştirilen özel yazılımlardır. Uygulama yazılımları sistem yazılımları ile entegreli bir şekilde çalışmaktadırlar. Bu nedenle bir işletim sisteminde çalışabilecek şekilde geliştirilen uygulama yazılımı farklı bir işletim sisteminde çalışmayacaktır. Örneğin; Windows işletim sistemi kullanan bir kişisel bilgisayar için geliştirilen bir uygulama yazılımı Macintosh işletim sistemine sahip bir bilgisayarda çalışmayacaktır. Bir metin yazılımı olarak Word, resim uygulamaları için hazırlanmış Paint, Adobe Photoshop; web tarayıcı görevi için Google Chrome, Opera, Firefox; sıkıştırma programı olarak WinRAR uygulamaları birer uygulama yazılımlarıdır.

Sistem yazılımları: Uygulama yazılımları ile bilgisayar donanımı arasındaki bağlantı sistem yazılımları aracılığıyla gerçekleştirilir. Sistem yazılımları temelde bilgisayarı çalıştıran, donanım birimlerinin denetiminden ve yönetiminden sorumlu olan, kullanıcı ile bilgisayar arasındaki iletişimi sağlayan yazılımlardır. Sistem yazılımlarını derleyiciler, yorumlayıcılar, sürücüler ve işletim sistemi yazılımları oluşturur.

Derleyiciler: Bir programlama dilinde yazılmış bir kaynak kodun, başka bir hedef dile veya bilgisayarın anlayabileceği makine diline çeviren aracı yazılımlardır. Derleyiciler; koddaki hataları yakalama ve iyileştirme işlemlerini kaynak kodu hedef koda çevirirken gerçekleştirir. Derleyici; kaynak kodu hedef koda çevrildikten sonra çalıştırır. Derleyici yazılımları sayesinde yüksek seviyeli diller ile uygulamalar geliştirilebilmektedir.

Yorumlayıcılar: Derleyicilerle aynı görevi üstlenirler. Bu görevi farklı bir şekilde icra ederler. Yorumlayıcılar, kaynak kodu satır satır veya bloklar halinde yorumlar, makinenin komut setine çevirir ve kodu çalıştırır. Herhangi bir komut satırının çevrilmesinde ya da çalıştırılmasında bir hata ile karşılaşılırsa yorumlayıcı çalışmasını durdurur ve hatalı satırı programcıya bildirir. Sırası gelmeyen komut satırları çalıştırılmaz ve bu satırlardaki hatalar görülmez. Kodun bütünselliği hakkında bir fikre sahip değildir. Bu nedenle de kodun bütününe dair iyileştirmeler gerçekleştiremez.

Derleyiciler, yorumlayıcılara göre daha hızlıdırlar. Tüm programı girdi olarak alırlar ve programı .obj koduna dönüştürürler. Bu kodu saklarlar. Bu nedenle derleyiciler yorumlayıcılara nazaran daha fazla hafızaya ihtiyaç duyarlar. Yorumlayıcılar karşılaştıkları ilk hatayı rapor ederler ve bir sonraki hataya kadar ilerlerler. Derleyici kaynak kodundaki bütün hataları bulur. Bu nedenle yorumlayıcının hata ayıklama işlemi daha kolaydır (Çobanoğlu, 2018). Pascal, C, C++, C#, Visual Basic dilleri derleyici kullanan programlama dilleridir. HTML, XML, PHP dilleri yorumlayıcı kullanan programlama dilleridir. Hem derleyici hem de yorumlayıcı kullanan programlama dilleri de vardır. Java dili örnek olarak verilebilir. Java dilinde kod önce derlenerek "byte code" adı verilen ve sadece Java sanal makinelerinde çalıştırılabilen bir kod üretmektedir. Bu üretilen ara kod, Java sanal makinesinde bir yorumlayıcı ile çalıştırılmaktadır.

Sürücüler: Bilgisayar donanımlarının işletim sistemine tanıtılmasını sağlayan yazılımlardır. İşletim sistemi tanımadığı bir donanımı yönetemez ve çalıştıramaz. Bir bilgisayara yeni bir işletim sistemi yüklendiğinde veya yeni bir donanım takıldığında donanıma ait sürücü yüklenmezse o donanım bilgisayar tarafından tanınmaz ve çalıştırılamaz. Bazen sürücüler yüklü olduğu halde donanımlarda hatalı çalışmalarla karşılaşabiliriz. Bu problemin çözümü sürücünün güncellenmesi olabilir. Windows işletim sistemi kullanılıyorsa "Aygıt Yönetici" arayüzünden donanımın şu an ki versiyonunu görebilir ve sürücü güncelleme işlemini gerçekleştirebiliriz.

İşletim sistemleri: Bilgisayar sisteminin donanım kaynaklarını yöneten, donanım ile kullanıcı arasındaki etkileşimi sağlayan ve donanım karmaşıklığından kullanıcıyı soyutlayan, uygulamaların geliştirilmesine zemin hazırlayan sistem yazılımıdır. İşletim sistemleri bilgisayarların ROM denilen hafıza biriminde saklanır ve bilgisayar açıldığında işletim sistemi yazılımının gerekli olan kısımları RAM olarak isimlendirilen hafıza birimine getirilir ve çalıştırılır.

Okuma Önerisi

Avcı, M., Özyıldırım, B. M. ve Ülgen, O. (2016). İşletim Sistemleri ve Sistem Programlama (3. bs.). Adana: Karahan Kitabevi.

Yıllar içerisinde bilgisayar donanımları geliştiği gibi yazılım teknolojileri de gelişmiştir. Gelişim o anki ihtiyaca ve teknolojiye bağlı olarak oluşmuştur. İşletim sistemleri de zamanla gelişen donanım ve ihtiyaca bağlı olarak bir gelişim ve değişim yaşamıştır. İlk kuşak bilgisayar sistemlerinde işletim sisteminin görevlerini bilgisayar uzmanları gerçekleştiriyordu. Transistörlerin gelişimi ile ikinci kuşak bilgisayar dönemine geçiş yapıldı ve delikli kartların üzerindeki komutları manyetik ortama aktarıp makine dilini kullanan işletim sistemleri gelişti (Yıldırım). Üçüncü kuşak bilgisayar sistemlerinde tümleşik devrelerin üretimi gelişmiştir. Bu gelişim bilgisayar sistemlerinin artık daha hızlı olmasını ve aynı anda birden fazla görevi icra edebilmesini sağlamıştır. Bu döneme ait işletim sistemleri zaman paylaşımlı çalışabilmekteydi. Mikroişlemcilerin gelişimi ile dördüncü kuşak bilgisayar sistemlerine geçiş sağlandı. Mikroişlemciler bilgisayarların daha küçük boyutlara erişmesini sağlamış bunun yanında kapasite, hız ve kullanılabilirlik işlevleri artmıştır. Bilgisayarlar bu özellikleri ile telefon, TV, tablet, beyaz eşya gibi hayatımızın her alanına girmiştir. Bu dönemde amacına uygun olarak işletim sistemleri geliştirilmiştir.

Anahtar Kavram

Transistör: Germanyum veya silisyumun elektrik geçirme özelliğinden yararlanılarak üretilen, elektronik tüplerin elektrik titreşimlerini genişletmekte kullanılan çok küçük, uzun ömürlü elektronik alet.

İşletim sistemleri farklı dönemlerde farklı görevler üstlenmiştir. Donanım ve yazılım teknolojilerinin gelişimi ile farklı amaçlara hizmet eden bilgisayar sistemleri geliştirilmiştir. Telefon, cebimizdeki bilgisayar sistemidir. Telefon, Televizyon, sunucu, masaüstü bilgisayar sistemlerinde bulunan işletim sistemleri birbirlerinden farklı, amacına uygun olarak geliştirilmiştir.

İşletim sistemleri birbirlerinden farklılık gösterse de tüm işletim sistemi yazılımlarının sorumluluğunda olan temel görevler vardır. Bunlar (Taşçı, 2017) (Yıldırım):

Şekil 1. İşletim Sistemi Yazılımlarının Sorumluluğunda Olan Temel Görevler

Proses yönetimi:

Bir bilgisayar programı, bilgisayar tarafından çalıştırıldığında özel bir görevi icra eden bir dizi komutlardan oluşur. Bilgisayar programları algoritmalar olarak bilinen iyi tanımlanmış görevleri barındırır. Bir bilgisayar programı programlama dili kullanılarak yazılırlar. Programların bilgisayar tarafından icra edilebilmesi için ana belleğe getirilmesi gerekir. Programlar işletilmediği sürece hiçbir işlem yapmazlar. Çalıştırılmak üzere ana belleğe getirilen kod parçaları CPU tarafından işletilir. CPU'da bulunan kod parçaları yürütülen kod parçalarıdır ve yürütülmekte olan kod parçaları proses olarak anılır.

İşletim sistemleri, üzerinde bulunduğu sistemin özelliklerine göre (tek çekirdekli, tek işlemcili, çok çekirdekli) proses yönetimini gerçekleştirir. Tek işlemcili veya tek çekirdekli bir sistemde birden fazla proses aynı anda çalıştırılamaz. Modern işletim sistemleri çok çekirdekli donanım üzerinde aynı anda birçok prosesin yönetimini gerçekleştirebilir. Proses yönetimini gerçekleştirirken işletim sisteminin gerçekleştirdiği bazı aktiviteler vardır. Bunlar (Taşcı, 2017);

İpucu
Proses Yönetimi konusuna 2. bölümde yer verilecektir.

- Yeni proses oluşturma ve sonlandırma
- Proseslerin oluşturulması, yürütülen proseslerin bekletilmesi, yeniden çalıştırılması,
- Prosesler arasındaki senkronizasyonu ve proses iletişimini sağlama.
- Kilitlenme probleminin üstesinden gelebilecek bir mekanizma sağlama.

Prosesler ve işletim sisteminin proses yönetimini gerçekleştirme teknikleri ayrıntılı olarak "Bölüm 2"de anlatılmaktadır.

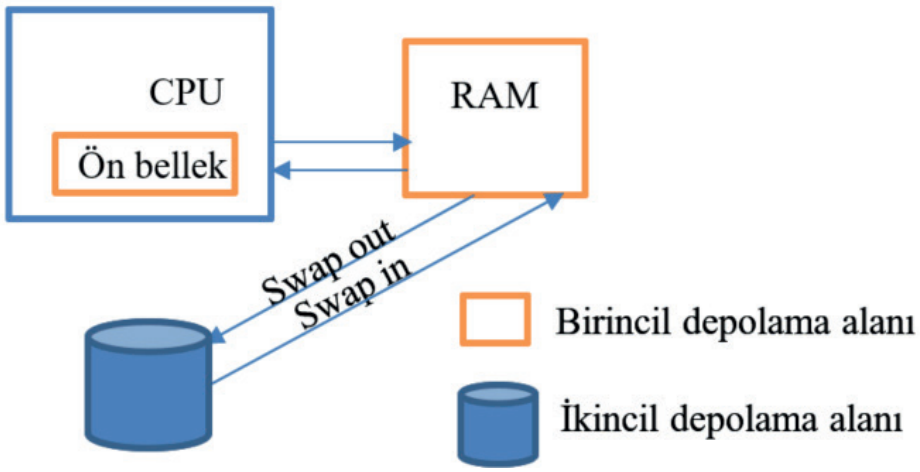
Bellek yönetimi:

İpucu
Bellek Yönetimi konusuna 3. bölümde yer verilecektir.

Bilgisayarda geçici olarak verilerin saklandığı merkeze ana bellek (RAM) denir. Ana bellek kontrollü paylaşıma izin verir. Ana bellek işletim sistemini ve diğer program parçalarını saklar. Bir program çalıştırılacağı zaman program aynı boyutta veya farklı boyutlarda (kullanılan yönteme göre) parçalara bölünür. Ana bellekte aynı boyutlarda veya farklı boyutlarda sanal olarak bölümlendirilir. Program parçaları ana belleğin farklı bölmelerine veya art arda gelen bölümlerine yerleştirilir. Ana belleğin bölmelerine yerleştirilen program parçalarının ana bellekteki konumu proses kontrol bloğu kaydedicisi tarafından saklanır.

İşletim sistemi ana belleği prosesler arasında en etkin bir şekilde paylaştırılmasını hedefler. Bu paylaşımın kontrollü bir şekilde gerçekleştirilmesi gerekir. İşletim sisteminin bu görevi bellek yönetimidir. Belleğin hangi bölümlerinin kullanımda olduğu hangi bölümlerinin kullanılabilir olduğu bilgisini kullanarak program parçalarına bellek tahsis eder. Farklı programlar ve program parçalarının ana bellekte en uygun yere yerleştirme işlemi bellek yönetimi kapsamında belirli algoritmalarla gerçekleştirir. Gerektiğinde tahsis edilen bellek alanını geri alma görevi de bellek yönetimine aittir.

Ana bellek kısıtlı boyuta sahiptir. Bu nedenle tüm programların ana bellekte saklanması mümkün değildir. Diskte bulunan hangi program parçasının ana belleğe yükleneceğinin kararı bellek yönetiminde bulunan algoritmalar tarafından belirlenir. Bellek yönetimi kapsamında karar verilen ana bellekte yer alan program parçaları diske, diskte yer alan program veya program parçaları ana belleğe taşınır. Disk ile ana bellek iletişim halindedir. Çalışma zamanındaki prosesler bellek ile disk arasında sürekli yer değiştirir. Diskten ana belleğin boş bir alanına program veya program parçasının taşınması işlemine belleğe taşıma (swap in), bellekten diske taşınması olayına diske taşıma (swap out) denir (Bk. Çizim 2).



Çizim 2.Görevin Ana Bellek-Disk Arasındaki Yer Değişimi

Ana bellekte açılan boş bellek alanları yeni program parçalarının tanımlanabilmesine ve öncelikli işlerin çalıştırılabilmesine olanak sağlar (Doğu Akdeniz Üniversitesi B. B., 2021). Ana bellekten hangi görevin diske taşınacağı, hangi bellek alanının boşaltılacağı işletim sistemi tarafından karar verilir. Bu kararı çeşitli kriterleri dikkate alarak gerçekleştirir. Bunlardan bazıları;

- Öncelik değeri
- En çok ana belleği kullanma
- Ana bellekte kapladığı boyut
- Kaynak bekleme süresi

İşletim sistemi öncelik değeri düşük, giriş-çıkış işlemleri fazla olan, kaynak bekleme süresi uzun olan, ana bellekte kapladığı alanın büyük olduğu, çok fazla ana belleği kullanan görevleri işletim sistemi ana bellekten diske taşımayı tercih eder.

Bellek yönetiminin görevlerinden biri de bellek alanlarının korunmasıdır. Bellek alanları, yerleştirilen proseslerin izinlerine göre işleme tabi olur. Örneğin; işletim sisteminin bulunduğu bellek alanı, kullanıcılardan korumak adına okunabilir ve çalıştırılabilir iznine sahiptir. Bu bellek alanına kullanıcı tarafından müdahale edilemez. Program parçaları izinsiz bir başka program parçasının bellek alanına erişmemelidir.

Bellek yönetimi ile ilgili ayrıntılı bilgi “Bölüm 2”de verilmektedir.

Giriş-çıkış birimlerinin yönetimi:

Giriş-çıkış birimleri hem dış ortamdan verinin alınması hem de dış ortama verinin gönderilmesini sağlayan bilgisayar donanım birimleridir. Klavye, fare, mikrofon giriş birimlerine; monitör, kulaklık, hoparlör çıkış birimlerine örnek

olarak verilebilir. Hem giriş hem de çıkış birimi olarak görev yapan bilgisayar donanımları da mevcuttur. Sabit diskler, hafıza kartları bu duruma örnek olarak verilebilir.

İşletim sistemi giriş-çıkış birimlerinin yönetiminden sorumludur. Bir giriş-çıkış biriminin işletim sistemi tarafından tanımlanması için o birime ait sürücü yazılımının bilgisayara yüklenmesi gerekir. Sürücü yazılım işletim sistemi ile ilişki kurar ve kullanıcıyı donanım ayrıntılarından soyutlar.

İşletim sistemi, giriş-çıkış birimlerinin yönetimini bazı aktivitelerle gerçekleştirilir. Bunlar;

- Giriş-çıkış birimlerinin ön bellek sistemini oluşturur.
- Ön belleğe okuma-yazma işlemi gerçekleştirilir.
- Spooling işlemlerinin gerçekleştirilmesi
- Sürücü yönetiminin gerçekleştirilmesi
- Aygıt yöneticisi arabirimini sağlamak

Disk yönetimi:

Tüm verilerin saklanması için ana belleğinin boyutunun küçük olması nedeniyle bilgisayar sistemleri ikincil belleklere ihtiyaç duyarlar. Oluşturulan dosya ve klasörler diskler üzerinde tutulurlar. Diskler üzerinde tutulan veriler güç kaynağı kesildiğinde kaybolmazlar. Manyetik disk, manyetik teyp, disk, disket, SSD, CD, DVD gibi ikincil belleklerde tutulan veriler ihtiyaç duyulduğunda ana belleğe yüklenirler. İşletim sistemi disk yönetimini bazı aktivitelerle gerçekleştirir. Bunlar;

- Disk üzerindeki boş alanın yönetimi
- Yeni dosyaların yazılması için depolama alanı tahsisi
- Bellek erişim isteklerinin zamanlanması
- Disk planlanması

Koruma ve güvenlik:

Bir bilgisayar sistemi, çok kullanıcılı ise ve birçok prosesin eş zamanlı çalışmasına izin veriyorsa kullanıcıların ve proseslerin birbirlerinin girişiminden korumalıdır (Türkoğlu). İşletim sistemi; dosya yönetimini, bellek yönetimini, giriş-çıkış birimlerinin yönetimini, proses yönetimini, disk yönetimini, ağ yönetimini bu ilkeyi temel alarak gerçekleştirir. Bu amaçla kullanıcı ve proses erişim kontrolünü, bilgi erişim kontrolünü gerçekleştirir. İşletim sistemi kullanıcıların birbirlerinden izole olmasını ve bilgisayar kaynaklarının kontrollü paylaşımını sağlar. İzinli ve izinsiz kullanıcıları ayırt eder ve kullanıcıya tanımlanan yetki ölçüsünde bilgisayar kaynaklarının kullanımına izin verir. Sistemde oluşan veri akışı denetlenir. Virüs taraması, izlemesi gerçekleştirilir. İşletim sistemi kullanıcı ve kaynakların güvenliği sağlar.

Ağ yönetimi:

Giriş-çıkış birimleri, saati, belleği, CPU'su ortak olmayan dağıtık yapıdaki bilgisayarlar birbirleriyle ağ aracılığıyla iletişim kurarlar. Her bilgisayarın kendine ait işlemcisi (CPU), belleği, giriş-çıkış birimleri vardır. Bir ağ yapısının oluşturulabilmesi için temel olarak yönlendirme ve bağlantı stratejisine karar verilmelidir. Bilgisayarlar arasındaki iletişim ağ protokolleri temel alınarak gerçekleştirilir. Ağ protokolleri, iletişim için gerekli olan kuralları, adımları belirler.

İşletim sisteminin ağ yönetimindeki en önemli görevi kullanıcıların güvenli bir şekilde ortak kaynaklara erişimini sağlamaktır.

Dosya yönetimi:

Veriler ve programlar bilgisayarda dosyalar halinde saklanır. Dosyalar, birbirleriyle ilişkili veri ve programları barındırır. Windows işletim sistemine sahip masaüstü bir bilgisayarı açtığınızda resimler, müzikler, programlar için ayrı ayrı dosyaların bulunduğunu görürüz. Kullanıcılar da ilişki veri veya programlarını bir arada tutmak için dosyalar oluşturabilirler. Dosya yapısı sanal bir yapıdır. Fiziksel olarak bilgisayarınızın belleğinde bir dosya oluşumu gerçekleşmez. Dosya yapısı içerisinde bulunan veri ve programlar belleğin farklı bölümlerinde saklanabilirler. Fakat kullanıcılar bu farklılığı görmezler. İşletim sistemi sayesinde farklı yerlerdeki ilişkili veri veya programların aynı yerde bulunuyor görüntüsü oluşturulur. Benzer dosyalar bir araya gelerek klasörleri (dizin) oluşturur.

İşletim sistemi dosya yönetimini bazı aktivitelerle gerçekleştirir. Bunlar;

- Dosyaların oluşturulması, silinmesi
- Klasörlerin oluşturulması, silinmesi
- Dosyaların, klasörlerin yönetimi
- Dosya ve klasörlerin ana belleğe taşınması
- Dosya ve klasörlerin disklere taşınması
- Dosyaların yedeğinin alınması

İşletim sistemi yazılımı temel görevlerini yerine getirirken işletim sisteminden beklenen bazı kriterler dikkate alır. Bunlar;

- Etkin bir şekilde işleri gerçekleştirmek.
- İşler arasındaki bekleme süresi en az düzeyde olmalıdır.
- CPU'nun boşta kaldığı süre az olmalıdır.
- Sistemin cevap verme süresi kısa olmalıdır.
- Az zamanda çok iş yapabilmelidir.
- Hatalardan arındırılmalıdır ve güvenirlik sağlanmalıdır.
- İşletim sistemi temel görevlerinden taviz vermeden daha az boyutlarda bellekte yer kaplamalıdır.
- Arayüz kullanıcı dostu olmalıdır.

İşletim sistemleri bilgisayar, tablet, video oyun konsollarında, cep telefonlarında, sunucularda, arabalarda, beyaz eşyada, kol saatlerinde hemen hemen her eşyada bulunmaktadır. Farklı amaçlarla üretilen bu eşyalar için farklı işletim sistemleri geliştirilmiştir. Yaygın olarak kullanılan işletim sistemi türleri aşağıda verilmektedir.

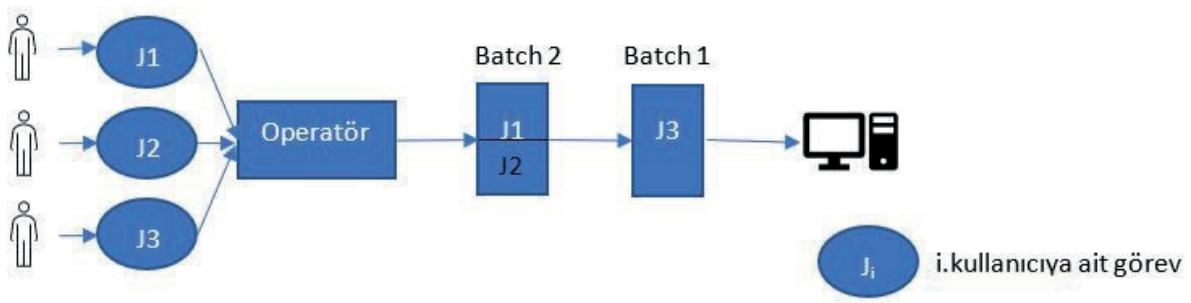
- Toplu işletim sistemleri (Batch Operating System)
- Zaman paylaşımlı işletim sistemleri (Time-sharing operating system)
- Dağıtılmış işletim sistemleri
- Ağ işletim sistemleri
- Gerçek zamanlı işletim sistemleri
- Mobil işletim sistemleri
- Gömülü işletim sistemleri

Toplu işletim sistemleri

Toplu işlem çok görevli bilgisayar sistemlerinde kullanılan görevlerin aynı anda işleme alınma biçimlerinden biridir. Toplu işlemde, görevler kuyrukta biriktirilir. Aynı gereksinimlere sahip olan benzer görevler gruplanır. Gruplama işlemini operatör yapar. Grupların her birine "batch" denir. Gruplanan işler sırası ile işlenir. Görevler sisteme sunulduktan sonra görev tamamlanana kadar kullanıcı bilgisayarla etkileşim kuramaz. Kullanıcı görevin işlem akışını izleyemez, denetleyemez. Toplu işletim sisteminin iş akışı Çizim 3'te gösterilmektedir.

Toplu işletim sistemleri 1960 yıllarda kullanılmaya başlanmıştır. Kullanıcılar, programlarını ve verilerini delikli kartlara tanımlardı. Delikli kart desteleri bekleme kuyruklarında biriktirilip bilgisayar operatörüne sunulurdu. Operatör uygun olan grubu işlemciye gönderir ve grup halinde işlem adımlarının tamamlanması beklenirdi.

Toplu işletim sistemleri çok kullanıcılı sistemdir. Herhangi bir işin bekleme süresinin tahmin edilmesi ve hata ayıklama işlemi bu tür işletim sisteminde oldukça zordur. Karmaşık, birden çok alt adımdan oluşan ve adım akış sırası değişmeyen, uzun süren işlerde toplu işlem türü başarılıdır (Saatçi, 2002).

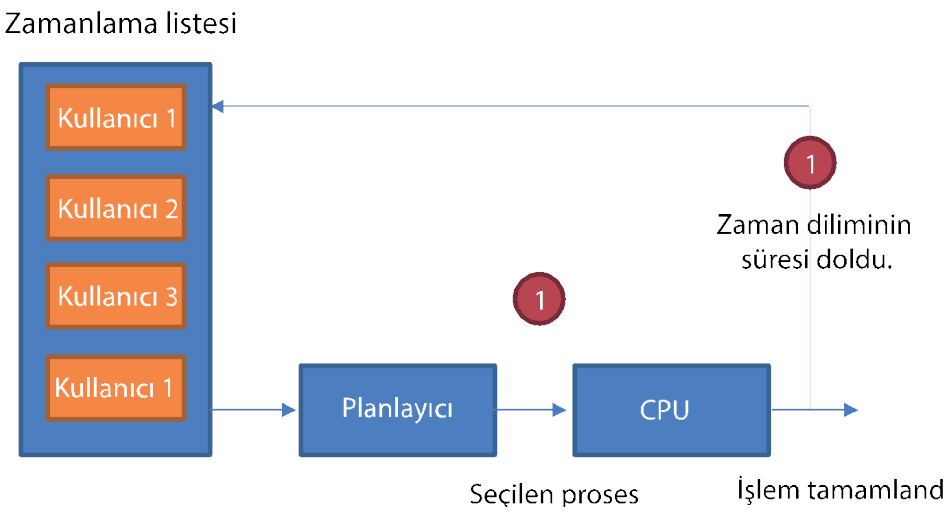


Çizim 3. Toplu İşletim Sistemlerinde İş Akışı.

Zaman Paylaşımlı İşletim Sistemleri

Zaman paylaşımlı sistemlerde, çok kısa zaman aralıklarıyla kullanıcılara paylaştırılır. Kullanıcı, diğer kullanıcılarla sistemi paylaştığını fark etmez.

Zaman paylaşımlı sistemlerde görev işleme etkileşimli bir şekilde gerçekleştirilir. Toplu işlem türünden farklı olarak bu işletim türünde iş adımları adım adım işletilir, işlem akışı izlenebilir ve hatta işletim akışına müdahale edilebilir. Kullanıcılar, kullanıcı arayüzü aracılığıyla sistem ile etkileşim kurabilir.



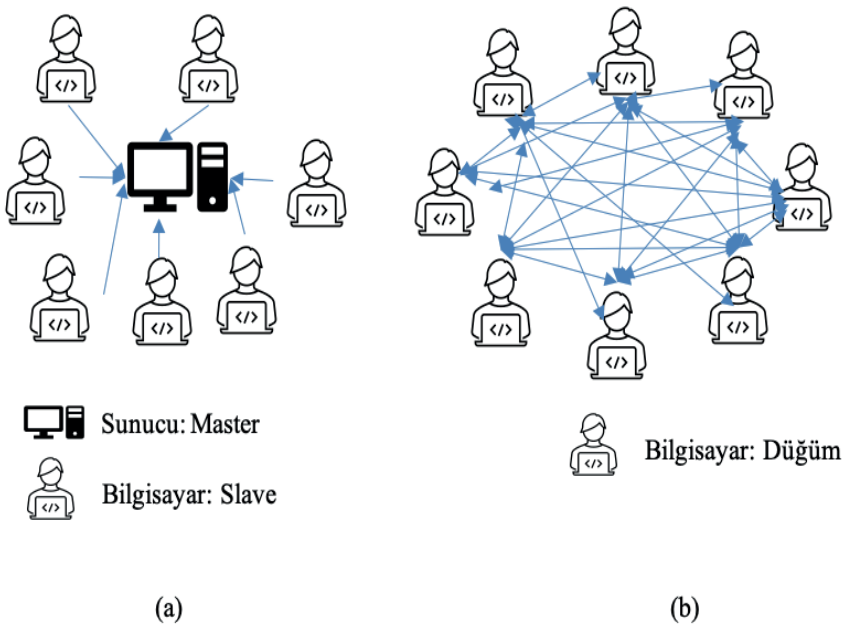
Çizim 4. Zaman Paylaşımlı Sistemin Mimari Yapısı

Çizim 4'te görüldüğü gibi zaman paylaşımlı sistemde kullanıcıların işlerinin sırasıyla tutulduğu bir kuyruk listesi bulunmaktadır. Planlayıcı bu kuyruk yapısından planlama algoritmasını temel alarak bir proses seçer ve CPU'ya gönderilir. CPU, proses için ayrılan zaman dilimi süresi boyunca prosesi çalıştırır. Proses bu zaman dilimi içerisinde tamamlanabilir veya tamamlanamaz ve bekleme kuyruğuna gönderilir.

Her kullanıcıya eşit bir zaman dilimi ayrılır. Bu zaman dilimi çok kısa olduğu için kullanıcılar zaman paylaşımlı olarak bilgisayar kaynaklarını paylaştığının farkına varmaz. Zaman paylaşımlı işletim sistemi bahsedilen gereksinimleri karşılamak üzere geliştirilmiştir.

Dağıtılmış işletim sistemleri

Dağıtılmış sistemlerin mimari yapısını daha iyi anlayabilmek adına öncelikle merkezi ve merkezi olmayan sistemlerin mimari yapılarını, avantajlarını ve dezavantajlarını ve neden dağıtık sistemlere ihtiyaç duyulduğunu inceleyelim.



Çizim 5. Merkezi Sistem Mimarisi (a)-Merkezi Olmayan Sistem Mimarisi (b).

Çizim 5. (a)'da görüldüğü gibi merkezi bir sistem mimarisi istemci-sunucu mimari yapısını kullanır. Bu mimari yapıda bir veya birden fazla istemci bilgisayar sunucu bilgisayarla doğrudan iletişim halindedir. Yaygın olarak kullanılan mimari yapıdır. İstemci bilgisayarlar sunucu bilgisayara istekte bulunur ve sunucu bilgisayar bu isteğe karşılık bir cevap gönderir.

Merkezi mimari yapılarında;

- İstemci bilgisayarların sayısı arttırılarak bir dikey ölçeklenebilirlik sağlanabilir. Fakat bu ölçeklenebilirlik kısıtlıdır.
- Çok fazla kullanıcının ağa bağlanıp sunucu (master) düğüme istekte bulunması darboğazlara neden olabilir.
- Merkezi mimari yapılarının performansını ağın ve sunucu bilgisayarın performansı belirler. Tek bir merkez noktası bulunduğundan bir istemci düğümün bağlantısının kopması sistemin başarısız olmasına neden olur.
- Veri yedekleme konusu zayıftır. Sunucu bilgisayar bozulduğunda ve herhangi bir yedek alınmadıysa verilerin kaybı söz konudur.
- Tek bir sunucu olduğu için merkezi sistemlerinin bakımının kolay olduğu düşünülebilir. Fakat tek bir sunucunun olması sebebiyle bakım gerekçesiyle sunucunun kapatılması verimsiz ve profesyonel olmayan bir davranıştır.

Yukarıda bahsedilen dezavantajlarının yanında merkezi sistemlerde

- İstemcilere güvenli bir şekilde hizmet vermek oldukça kolaydır.
- Kurulumu diğer sistemlere nazaran kolaydır.
- Güncellemeler daha kolay gerçekleştirilir.

Merkezi sistemlerde günümüzde yaygın olarak kullanılmaktadır. Wikipedia uygulaması bu mimari yapıya örnek verilebilir. Yüksek performanslı bir sunucu kullanıcılardan gelen isteklere cevap döndürür.

Çizim 5. (b)'de merkezi olmayan bir sistemin yapısı görülmektedir. Bitcoin uygulaması merkezi olmayan sistemlerin popülerliği ve uygulama alanı arttırmıştır. Merkezi olmayan sistemde ağda bulunan her düğüm birbirlerinden bağımsızdır ve birbirlerine doğrudan bağlanır. Her düğüm hem istemci hem de sunucu rolü üstlenir.

Merkezi olmayan bir yapıya sahip sistemlerde;

- Her düğüm kendi performansını arttırarak bütün sistemin performansının arttırılmasına katkı sağlayabilir. Dikey ölçeklenebilirlik mümkündür fakat kısıtlıdır.
- Küçük sistemler için uygun değildir.
- Bir düğüme müdahale etmenin bir yolu yoktur.
- Uygulamanın geliştirilmesi merkezi sistemlere nazaran daha zordur.

Yukarıda bahsedilen dezavantajlarının yanında merkezi olmayan sistemlerde;

- Darboğaz problemi ya hiç yaşanmaz ya da minimum seviyede yaşanır.
- Tüm sistemin tamamen çalışamaz olması oldukça zordur. Tek bir düğümün bozulması sistemin tamamını etkilemez.

Dağıtılmış sistemler farklı konumlarda bulunan bilgisayarların bir araya getirilerek tek ve bütünleşik bir makine gibi davranmasını sağlayacak şekilde tasarlanmıştır. Dağıtık sistemler sayesinde herhangi bir kullanıcı kendi bilgisayarın kaynaklarının yanı sıra ağ ile birbirlerine bağlı olan diğer bilgisayarların da kaynaklarını kullanabilme imkânı sağlanmıştır (bk. Çizim 4). Kullanıcı tek bir bilgisayar ile etkileşim kurduğu izlenimi yaşarlar. Ortamda birden fazla bilgisayarın bulunduğunun farkında değildirler. Bu özelliği merkezi olmayan sistemlerden ayıran özelliklerinden biridir. Örneğin dağıtık sistemde birden fazla bellek ve CPU olmasına rağmen kullanıcı, tek bellek ve CPU olarak görür. Çalıştırdığı programların hangi CPU'da işlendiği ve hangi belleği kullandığını kullanıcılar bilmez.

Dağıtık sistemlerde farklı işletim sistemlerine sahip bilgisayarlar ağ aracılığıyla birbirlerine bağlanırlar. Dağıtık sistemlerin etkili, hızlı, tutarlı ve güvenilir bir şekilde çalışabilmesi için bazı zorlukların aşılması gerekir. Bunlar;

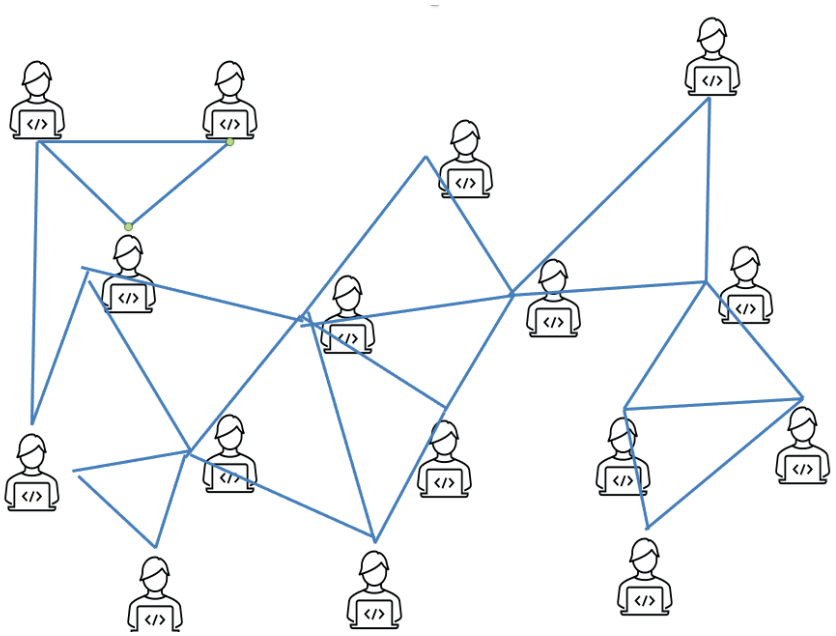
- Farklı sistemler birbirleriyle uyumlu ve tutarlı çalışmalıdır.
- Ağdaki bir bileşenin bozulması dağıtık sistemin bozulmamasını sağlamalıdır.
- Ölçeklenebilirlik ihtiyaç dahilinde arttırılabilmelidir.
- Tek bir bilgisayar varmış görüntüsünü vererek kullanıcı dağınıklıktan soyutlanmalıdır.

Dağıtık işletim sistemleri, bu problemlerin üstesinden gelebilme ve ağ üzerinde bulunan tüm bilgisayar kaynaklarının etkili, hızlı, güvenilir bir şekilde kullanılabilme imkânı sağlar. Dağıtık işletim sistemi bu görevi; farklı noktalarda icra edilen proseslerin iletişimini, senkronizasyonunu sağlayarak, tutarlılık, hata toleransı ve güvenlik ilkelerini temel alarak gerçekleştirir.

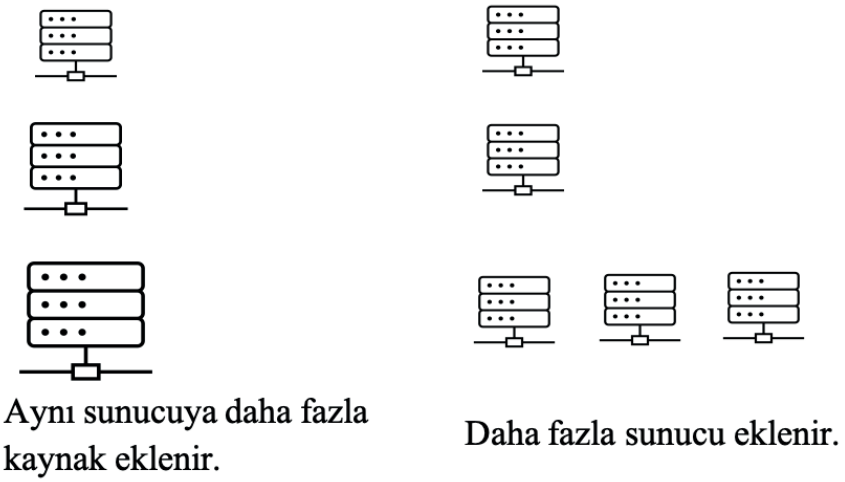
Ölçeklenebilirliğin yüksek olması büyük boyutlardaki verileri işlemek gibi yüksek performanslı işlemlerde dağıtık sistemler başarılı performans gösterir. Eşit ve birbirlerine bağlı düğümlere işlemler ağ üzerinden eşit olarak paylaştırılır.

Dağıtık bir sistemde;

- Yatay ölçeklendirme gerçekleştirilebilir.
- Hata toleransı çok daha yüksektir.
- Başlangıç maliyeti yüksek olmasına rağmen yatay ölçeklenebilirliği sayesinde hızla daha uygun maliyetli hale gelirler. Yatay ölçeklenebilir ve dikey ölçeklenebilirlik Çizim 5'te gösterilmektedir.
- Büyük boyutlardaki veri daha küçük parçalara bölünerek ağ üzerindeki düğümlere iş yükünü paylaştırır.
- Paralellik desteklenir.



Çizim 6. Dağıtık Sistem

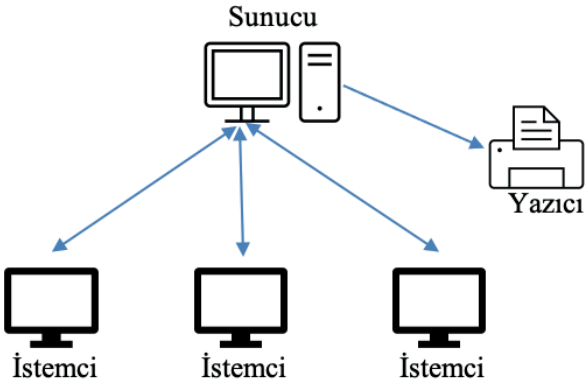


Çizim 7. Dikey ve Yatay Ölçekleme.

Ağ İşletim Sistemi

Ağ ile birbirlerine bağlı bilgisayarların istemci-sunucu yaklaşımını (bk. Çizim 8) temel alarak birbirlerinin kaynaklarından faydalanabildiği sistemler için geliştirilen bir işletim sistemidir. İstemci-sunucu yaklaşımında sunucu olarak ifade edilen bir bilgisayar sistemi, istemci olarak ifade edilen bilgisayar sistemlerine hizmet sunar. İstemci ve sunucu bilgisayarlar arasındaki iletişim ağ protokolleri çerçevesinde ağ aracılığıyla gerçekleştirilir. İstemci bilgisayarlar sunucu olarak nitelendirilen bilgisayar sistemlerinin kaynaklarından faydalanılır.

Ağ işletim sisteminde kullanıcı ağ üzerinde birden fazla bilgisayarın varlığından haberdardır. Her kullanıcının oturum açabildiği bir işletim sistemi mevcuttur.



Çizim 8. İstemci-sunucu mimarisi.

Gerçek Zamanlı İşletim Sistemleri

Gerçek zamanlı işletim sistemlerinde proseslerle ilişkili bir son tarih vardır ve işletim sistemi bu tarihe kadar prosesi tamamlaması gerekir. Bu amaçla gerçek zamanlı işletim sistemlerinde görevler hassas bir zamanlama ile icra edilirler. Gerçek zamanlı işletim sistemlerinde en ufak gecikme kabul edilmez. Tepki ve işlem süresi oldukça kısadır. Bu özellikleri ile kritik görevler için tercih edilen bir işletim sistemidir.

Gerçek zamanlı işletim sisteminin bir diğer önemli özelliği ise proseslerin BigO karmaşıklığını kestirebilmesidir. Mevcut donanımda prosesin aşağı yukarı ne kadar sürede tamamlanabileceği tahmin edilebilmektedir. Örneğin: bir prosesin tamamlanma süresi olarak 180 ms olduğu tahmin edilmiş olsun. Donanıma göre bu prosesin tamamlanma süresi değişkenlik gösterebilir. İşletim sisteminin anladığı şey bu prosesin 180 ms'de bitmiş olması gerektiğidir. İster prosesin çalışması tamamlansın isterse proses kitlensin, sonsuz döngüye girsin veya hata versin, proses 180 ms sonra CPU'dan ayrılır. Prosesin tamamlanması 180 ms'den önce de olabilir. İşletim sistemi bunu önemsemez. 180 ms CPU bekler. Bu performans kayıplarına neden olur.

Prosesin ihtiyaç duyacağı tüm kaynaklarda prosese ayrılan süre boyunca kitlenir ve diğer proseslerin bu kaynaklara erişilmesine izin verilmez. Bu özelliği kilitlenme durumu önler. Bu durum aynı zamanda aynı kaynağa erişmek isteyen diğer prosesin bu kaynağa ne zaman erişebileceğinin bilinmesini de sağlar. Gerçek zamanlı işletim sistemleri ile genel amaçlı işletim sistemlerinin temel özellikleri Tablo 1'de verilmektedir.

Tablo 1. RTOS & GPOS

Deterministiktir: Çalışma zamanı örüntüsünde rastgelelik yoktur.	Çalışma zamanı örüntüsü rastgelelik içerir.
Yanıt/tepki süresi tahmin edilebilirdir.	Dinamik bir yapı vardır.
Zaman kısıtlıdır.	Cevap verme süresinin garantisi verilmez.

QNX işletim sistemi en çok tercih edilen, kullanıcı arayüzleri olan, kapalı kaynak kodlu gerçek zamanlı işletim sistemlerinden biridir. FreeRTOS, Windows CE, Zephyr projesi gerçek zamanlı işletim sistemleridir.

Mobil İşletim Sistemi

Günlük hayatımızda kullandığımız donanımlar üzerinde donanımları kontrol eden yazılım uygulamaları mevcuttur. Donanımlar geliştikçe, yetenekleri arttıkça bu donanımlar üzerindeki uygulamalarında yetenekleri artmakta gelişmektedirler. Mobil işletim sistemleri de bu amaçla ortaya çıkmıştır. Tablet, PDA, telefon, saat gibi akıllı cihazlar geliştikçe akıllı cihazları denetleyen, yöneten uygulamalarda

gelişmektedir. Mobil işletim sistemleri taşınabilir akıllı cihazlar için geliştirilmiş işletim sistemidir.

Linux tabanlı olan Android, Apple şirketinin mobil cihazları için geliştirdiği iOS ve Microsoft tarafından geliştirilen Windows Phone, Symbian yaygın olarak kullanılan mobil işletim sistemleridir.

Android işletim sistemi, Linux çekirdeği üzerine inşa edilmiş mobil cihazlar için geliştirilmekte olan açık kaynak yazılımlı ve ücretsiz bir işletim sistemidir. Android işletim sisteminin ücretsiz olması ve açık kaynak kodlu olması sistemin daha hızlı gelişmesini sağlar.

Yazılım geliştiriciler gelişmiş API (Application Programming Interface) ile Android uygulamaları kolaylıkla geliştirebilirler. Uygulama geliştirme dili Java programlama dilidir. Android uygulamaları Google Play markette kullanıcıların hizmetine sunulur. Uygulamalar .apk uzantısını desteklemektedir.

iOS (IPhone Operating System) işletim sistemi Apple tarafından iPhone telefonları için geliştirdiği Unix tabanlı kapalı kaynak yazılımlı ve ücretli bir işletim sistemidir. Kapalı bir sistem olması iOS işletim sistemini saldırılara karşı daha korunaklı hale getirir. Ücretli olması kullanıcı sayısını düşürmektedir.

iOS işletim sistemi iPhone'dan sonra Apple tarafından geliştirilen iPod Touch, iPad, Apple TV, MacBook gibi cihazlarında da kullanılmıştır. Apple'in cihazlarından başka hiçbir sistemde iOS çalışmamaktadır. iOS işletim sisteminde 2007 yılına kadar kullanıcıların uygulama geliştirilmesine izin verilmemiştir. 2007 yılında Apple SDK'nin geliştirildiği duyurulmuş ve iOS Apps mağazası açılmıştır. Uygulama geliştirmek için Swift programlama dili kullanılmaktadır. Swift programlama dilinin kullanılabilmesi için Mac bir bilgisayar kullanılması gerekir. XCode geliştirme aracı kullanılarak Swift programlama dili ile iOs uygulamaları kolaylıkla geliştirilebilmektedir.

iOS işletim sistemi sade ve kullanışlı bir kullanıcı arayüzüne sahiptir. Güvenliğe azami önem verir. Kullanıcılar tarafından geliştirilen uygulamalar birbirlerinden uzak tutulurlar. Bir uygulama sebebiyle mobil cihaza virüs bulaşması durumunda iOS işletim sisteminin diğer yazılımları zarar görmez.

Windows Phone Windows CE çekirdeğine sahip Nokia telefonlarla beraber tanıştığımız Windows Mobile serisinin devamı niteliğinde olan kapalı kaynak kodlu işletim sistemidir. Windows Phone işletim sistemi Windows tabanlı bir işletim sistemidir. Fakat masaüstü Windows uygulamalarını çalıştıramaz. Geliştirilen uygulamalar Windows Phone mağazalarında hizmete sunulur.

Windows Phone işletim sistemleri iOS ve Android işletim sistemleri ile rekabet edememiş ve başarısız olmuştur.

Symbian, Symbian şirketi tarafından geliştirilmiş Psion şirketinin geliştirdiği EPOC işletim sistemini temel alan mobil cihazlar için geliştirilmiş işletim sistemidir (Kaya, 2009). Uygulama geliştirme için C++ programlama dili kullanılmaktadır.

Anahtar Kavram

Virüs: Bilgisayarlara veri girişi yoluyla yüklenen ve sistemin veya programların bozulması, beklenildiği gibi çalışmaması, veri kaybı gibi olumsuzluklara sebebiyet veren yazılım.

Gömülü İşletim Sistemi

Gömülü sistemler, işlevselliği ve güvenirliği arttırmak için özel bir amaç için tasarlanan bilgisayar sistemleridir. Tıbbi cihazlar, hesap makineleri, beyaz eşyalar, akıllı kol saatleri, otomobiller, kameralar, küçük ev aletleri, oyuncaklar gibi birçok yerde gömülü sistem kullanılmaktadır. Gömülü sistemlere ait uygulama geliştirmek, daha küçük bir hedefe sahip olunduğu için daha basittir.

Şekil 2. Gömülü Sistemler (Alkar, 2015)

Gömülü işletim sistemleri gömülü bilgisayar sistemleri için geliştirilmiştir. Standart masaüstü işletim sistemlerinin sağladığı ve özel uygulamalar tarafından kullanılmayan birçok fonksiyonu bir kenara bırakarak güvenilir ve verimli kaynak kullanımı için tasarlanmıştır. Embedded Java, Windows IoT, FreeRTOS, Gömülü Linux, Windows CE, Windows XP Embedded işletim sistemleri gömülü sistemler için geliştirilmiş gömülü işletim sistemlerine örnektir. Gömülü sistemler için geliştirilen işletim sistemlerinin her birinin özel bir fonksiyonu bulunur.

Bu Bölümde Ne Öğrendik?

* Bilgisayar temel olarak yazılım ve donanım bileşenlerinden oluşur.
    * Yazılım barındırmayan bir bilgisayar sadece mekanik işlevlerden ibarettir. Yazılım bileşeni kullanıcıların özel ihtiyaçlarını geliştiren uygulama yazılımları ve sistemin güvenli bir şekilde çalışması, korunması, sürekliliğin sağlanması için geliştirilen sistem yazılımları bileşenlerinden oluşur.
* İşletim sistemi bilgisayar ile kullanıcı arasında iletişimi sağlayan, bilgisayar donanımlarının birbirleri ile senkronize çalışmasını sağlayan sistem yazılımıdır. Gerçek zamanlı işletim sistemleri, mobil işletim sistemi, ağ işletim sistemi, dağıtılmış işletim sistemleri gibi farklı amaçlar için tasarlanmış farklı türlerde işletim sistemi yazılımları geliştirilmiştir.
    * Windows işletim sistemleri, Symbian, Android, iOS, QNX, FreeRTOS, Linux işletim sistemleri günümüzde yaygın olarak kullanılan işletim sistemi yazılımlarıdır.
* İşletim sistemleri farklı amaçlar için geliştirilmiş olsalar da ortak bazı sorumlulukları vardır. Proses yönetimi, bellek yönetimi, giriş-çıkış birimlerinin yönetimi, disk yönetimi, koruma ve güvenlik, ağ yönetimi, dosya yönetimi işlemleri işletim sistemi yazılımlarının temel görevleridir.
    * Bu görevleri yerine getirmek için işletim sistemleri birçok aktivite gerçekleştirirler.
* İşletim sistemi yazılımı temel görevlerini yerine getirirken işletim sisteminden beklenen bazı kriterleri dikkate alır. Bu kriterler; bekleme süresi ve cevap verme süresinin kısa olması; hatasız, güvenilir ve hızlı bir şekilde aktivitelerini gerçekleştirebilme; yeteneklerinden taviz vermeyecek şekilde daha az boyutlarda bellekte yer kaplama ve arayüzünün kullanıcı dostu olması şeklinde sayılabilmektedir. Sadece işletim sistemi yazılımları için değil ideal bir yazılım uygulaması için de bu kriterler dikkate alınarak geliştirilmelidir.
    * Farklı amaçlarla üretilen eşyalar için farklı işletim sistemleri geliştirilmiştir. Popüler olan işletim sistemi türleri; toplu işletim sistemleri, zaman paylaşımlı işletim sistemleri, dağıtılmış işletim sistemleri, ağ işletim sistemleri, gerçek zamanlı işletim sistemleri, mobil işletim sistemleri, gömülü işletim sistemleri şeklindedir.

Kaynakça

(tarih yok).

(2021). Debian Manpages: https://manpages.debian.org/ adresinden alındı

Aksan, C. (2021). *Paket Yöneticisi Nedir? Neden İhtiyaç Duyulur?* ceaksan.com: https://ceaksan.com/tr/paket-yoneticisi adresinden alındı

Aladağ, M. (2015). *Windows 10 Yenilikleri – PowerShell 5 ile Paket Yönetimi OneGet*. cozumpark.com: https://www.cozumpark.com/windows-10-yenilikleri-powershell-5-ile-paket-yonetimi-oneget/ adresinden alındı

Alkar, A. (2015). Embedded system basics and application. Ankara.

Çobanoğlu, B. (2018). *Herkes için Python*. Pusula.

Doğu Akdeniz Üniversitesi, B. B. (2021). İşletim Sistemleri-Bellek Yönetimi.

Doğu Akdeniz Üniversitesi, B. V. (2021). İşletim sistemleri-Kilitlenmeler. Siirt.

Gülbağ, A. (2017). İşletim Sistemlerine Giriş.

javaTpoint. (2021). *OS*. javaTpoint: https://www.javatpoint.com/os-attributes-of-a-process adresinden alındı

Kaya, A. (2009). Symbian İşletim Sistemi. *Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri*. Şanlıurfa.

Özdemir, H. (2018). *Linux Paket Yöneticileri*. pythontr.com: https://www.pythontr.com/makale/linux-paket-yoneticileri-632 adresinden alındı

Saatçi, A. (2002). Bilgisayar işletim sistemleri. Ankara: Hacettepe Üniversitesi. http://hilmi.trakya.edu.tr/ders_notlari/os/isleti_sistemleri.pdf adresinden alındı

Samet, R. (2018). Bilgisayar Sistemleri. (A. Ü. Enstitüsü, Dü.)

Shotts, W. (2013). *The Linux Command Line- Second Internet Edition*. No Starch Press.

Silberschatz, A., Gagne, G., & Galvin, P. (tarih yok). *Operating System Concepts*. içinde John Wiley & Sons, Inc. 2021 tarihinde alındı

Taşcı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Taşçı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Türkoğlu, İ. (tarih yok). İşletim Sistemleri (Ders Notları). *Fırat Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Bölümü*. Türkiye. 2021 tarihinde alındı

Wikipedia. (2021). *Paket yönetim sistemi*. Wikipedia.org: https://tr.wikipedia.org/wiki/Paket_y%C3%B6netim_sistemi adresinden alındı

Yıldırım, S. (tarih yok). Bilgi Teknolojilerine Giriş3. Atatürk Üniversitesi. 2021 tarihinde alındı

2. PROSES YÖNETİMİ

Fizik, evrenin işletim sistemidir.

Steven R. Garman

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Proses ve program parçası arasındaki farkı öğrenebilir.
- CPU planlama algoritmalarını tanıyabilir ve parametrelerini hesaplayabilir.
- Prosesler arasındaki iletişimin neden olabileceği problemleri öğrenebilir ve çözüm üretebilir.
- Prosesler arasındaki iletişimi sağlayan mekanizmaları öğrenebilir ve çözüm üretebilir.
- Kilitlenme problemlerine çözüm üretebilir.

Başlamadan Önce

Bir önceki bölümde, işletim sisteminin bilgisayar sistemindeki yeri ve önemi anlatılmış; işletim sistemi yazılımlarının üstlendiği görevler incelenerek, yaygın olarak kullanılan işletim sistemleri türlerine yer verilmişti. Bu bölümde prosesler, işletim sisteminin proses yönetimi için gerçekleştirdiği aktiviteler daha ayrıntılı bir şekilde anlatılacaktır. Proseslerin birbirleriyle uyumlu bir şekilde çalışabilmesi için gerekli olan koşullar, proseslerin birbirleriyle iletişim kurmasını sağlayan mekanizmalar, proseslerin oluşturabileceği sorunlar ve bu sorunları önlemek için kullanılan yaklaşımlar ele alınacaktır.

Birlikte Düşünelim

1. Prosesler arasındaki iletişim için hangi mekanizmalar kullanılmaktadır?
2. Kilitlenme durumundan kurtulmanın yolları neler olabilir?
3. Çok çekirdekli sistemlerde gerçek anlamda paralellik sağlanmakta mıdır?

2.1. Prosesler

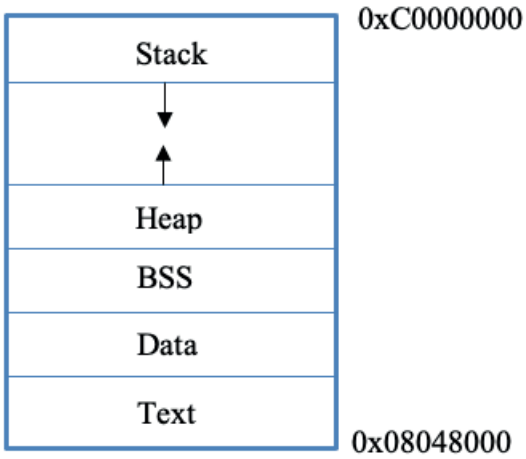
Proses, daha sonra tüm hesaplamaların temelini oluşturan yürütülmekte olan program parçasıdır. Program çalıştırılmadığı sürece pasif bir varlıktır, proses ise programın aksine aktif bir varlıktır. Toplu (Batch) sistemlerde proses yerine iş (job) terimi kullanılmaktadır (Silberschatz, Gagne, & Galvin). Birçok işletim sisteminde proses terimi yerine job terimi kullanılmaktadır. Bu iki terim birbirlerinin yerine kullanılmaktadır.

Şekil 3. Proses Özellikleri (javaTpoint, 2021)

Prosesin özellikleri proses kontrol bloğunda tutulur. Proses kontrol bloğu, her proses için işletim sistemi tarafından tutulan bir veri yapısıdır. Proses kontrol bloğu, proses ID (PID) numarası ile tanımlanırlar. Bu numara sayesinde hangi prosesin bilgilerini tuttuğu anlaşılır. Her proses, proses kontrol bloğu tarafından işletim sisteminde temsil edilir. Proses kontrol bloğu prosesin ihtiyaç duyabileceği tüm bilgileri saklamamızı sağlar. Kaydedicilerin içeriğini depolamaktan sorumludur.

Bir prosesin durum bilgisi değiştiğinde işletim sistemi tarafından proses kontrol bloğundaki proses durumu alanına ait bilgi güncellenir.

Bir proses ana belleğe "Çizim 9"da gösterilen bir düzen içerisinde yerleştirilir. Stack ve Heap kavramlarının dilimizdeki karşılığı "yığın" dır. Aynı anlama gelirler fakat prosesin farklı bölümlerini saklarlar. Bir program tarafından çağrılan bir fonksiyonun ihtiyaç duyabileceği bütün veriler (yerel değişkenleri, yerel fonksiyonları ve dönüş adresleri) bu alanda saklanır.



Çizim 9. Prosesin Ana Bellek İçerisindeki Düzeni

```
int smile (int a)
{return (a+2);}
```

Kod 1. Stack bölümü örnek kod parçası.

Kod 1'de tanımlanan kod parçası yerel fonksiyon olması, yerel değişkenler içermesi ve dönüş adresiyle beraber ana belleğin stack bölümünde saklanır. Stack bölümü "en son giren en önce çıkar" (last input first output-LIFO) mantığı ile çalışır. Bu bölüm genellikle işletim sistemi çekirdek alanının hemen altındaki daha yüksek bellek adreslerinde yer alır. Standart X86 mimari yapılarında 9'da görüldüğü gibi stack alanı aşağıya doğru büyür. Bazı mimari yapılarında ise zıt yönlü bir büyüme olur. Stack işaretçisi (stack pointer) yığının en üst noktasını gösterir. Stack pointer ile heap pointer karşılaşırsa veya stack alanının sınırına ulaşılırsa kullanılabilir boş belleğin tükendiği anlamına gelir.

İpucu
Bellek konusuna 3. bölümde yer verilecektir.

Bir program çalışma zamanı sırasında belleğe ihtiyaç duyduğu zaman dinamik bellek sağlamak için heap bölümünü kullanılır. Örneğin; boyutu yalnızca çalışma

zamanında bilinebilen ve program yürütülmeden önce derleyici tarafından statik olarak belirlenemeyen değişkenler için programcı tarafından talep edilen miktarda heap bölümünde yer ayrılır. Heap bölümü C dilinde "malloc/free" C++ dilinde ise free/delete" komutları ile yönetilebilir.

BSS segmentinin açılımı "Block Started by Symbol"'dür. Başlatılmamış veri segmenti anlamına gelir. BSS segmenti o aritmetik değer ile başlatılmış veya kaynak kodun içerisinde açık bir şekilde herhangi bir değer ataması olmamış tüm genel ve statik değişkenleri içerir. Örneğin *static int a;* olarak tanımlanmış bir değişken BSS segmentine tahsis edilir.

Data segmenti, BSS segmentinin aksine başlangıç değeri atanan global ve statik veri değişkenlerini içerir. (Bk. Kod 2)

```
int global_var; // Data alanında saklanır.
int main()
{
static int a; // Data alanında saklanır.
//Diğer kod ifadeleri
return 0;
}
```

Kod 2. Data Alanı Örnek Kod Parçası

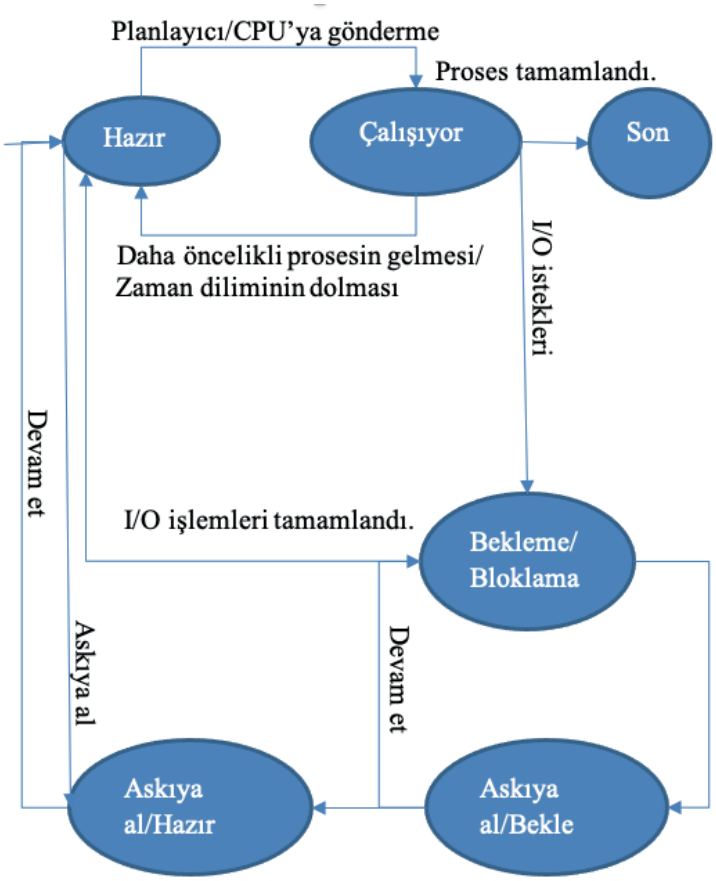
Text bölümü, code bölümü olarak da bilinir. Programın çalıştırılabilir komutlarının bulunduğu bölümdür. Bu bölümde yer alan komutların sadece okunabilmesine ve çalıştırılmasına izin verilir.

CPU planlama algoritmalarına göre ana bellekten kod parçaları seçilir. Her proses belirli bir süre işlemci tarafından yürütülür. Prosesler, kendilerine ayrılan sürenin dolması ile CPU'dan ayrılırlar. Bu süre zarfında prosesler tamamlanamadıysa bekleme kuyruğuna alınırlar. Bazı prosesler giriş-çıkış işlemleri gereksiniminden veya öncelikli prosesin gelmesi durumları gibi sebeplerden dolayı zamanı dolmadan işlemciyi terk edebilir ve bekleme kuyruğuna gelebilir. İş planlayıcısı tarafından bekleme kuyruklarından ilgili proses seçilerek CPU'ya gönderilir. Tüm bu süreçlerde proses farklı durumlara geçer. Prosesin durumları ve bu durumlara geçiş akışı Çizim 10'da verilmektedir.

Çizim 10'da görüldüğü gibi ana bellekten planlayıcı tarafından seçilip getirilen program prosesin "yeni" durumunu temsil eder. Proses oluşturulduktan sonra "hazır" durumuna geçer. Proses çalıştırılmak için hazırdır. İşletim sistemi çalıştırılacak diğer işlemleri diskten alır ve ana belleğe getirir. CPU'ya atanan ve çalıştırılan proses "çalışıyor" durumuna geçer. Eğer bir proses giriş-çıkış işlemlerine ihtiyaç duyarsa "Bekleme/Bloklama" durumuna geçer. Prosesin giriş-

çıkış işlemleri bittiğinde çalıştırılmak üzere "hazır" durumuna geçer. Bekleme kuyruğu tamamen dolu olduğunda "Askıya al-bekle" durumuna geçiş yapar. Hazır kuyruğu tamamen dolu olduğunda "Askıya al-hazır" durumuna geçer. Kuyruklar boşaldığında ilgili proses hazır durumuna geçer. Prosesin tüm işlemleri tamamlandığında "son" durumuna sahip olur.

İşletim sistemi proses oluşturabilir, planlama işlemi gerçekleştirir, çalıştırır, silebilir veya öldürebilir. Proseslerin planlanması çeşitli planlayıcılar tarafından yapılır. Uzun dönemli planlayıcılar iş planlayıcı olarak da bilinirler. Diskten prosesleri seçerler ve ana bellekte yer alan hazır kuyruğunda prosesleri tutarlar. Temel amacı giriş/çıkış işlemlerine bağlı veya CPU'ya bağlı prosesler arasından en iyi seçimleri yapmaktır. Bu görev oldukça önem arz eder. İş planlayıcısı görevini en uygun şekilde gerçekleştirmezse eğer giriş-çıkış işlemleri uzun süren prosesler, diğer proseslerin çok daha fazla beklemesine, CPU'nun çok daha fazla atıl durumda kalmasına neden olabilir. Bir diğer planlama işlemi CPU tarafında gerçekleştirilir. CPU planlayıcı olarak da anılırlar. CPU planlayıcısı, hazır kuyruğunda yer alan proseslerin seçimini ve planlamasını gerçekleştirir. Hazır kuyruğunda bekleyen CPU planlayıcı tarafından seçilen prosesler, CPU'ya çalıştırılmak üzere gönderilir. Hazır kuyruğundan hangi prosesin seçileceğini belirlemek algoritmalar ile yapılır. CPU planlayıcısı görevi en uygun şekilde yapmazsa hazır kuyruğunda bekleme süresi artabilir. Bu durum istenmeyen bir durumdur. Prosesler farklı durumlara geçiş yaparlar (yukarıda bu durumdan bahsedilmiştir). Çalışan bir proses, bazı zamanlarda giriş/çıkış işlemlerine ihtiyaç duyabilir ve o zaman bu prosesin durum bilgisi "bekleme/bloklama" olur. Orta vadeli planlayıcı işte tam bu durumlarda devreye girer ve giriş/çıkış işlemlerine ihtiyaç duyan prosesi alır ve diğer proseslerin CPU tarafından işletilmesine imkân sağlar. Bu takas durumudur. Orta vadeli planlayıcı, en uygun şekilde prosesleri askıya almak ve gerektiği vakitte prosesleri tekrar sürdürmekten sorumludur.



Çizim 10. Proses Durum Diyagramı.

İpucu
Disk konusuna 4. bölümde yer verilecektir.

Program parçaları çalıştırılmadan önce diskte saklanırlar. Diskten iş planlayıcı tarafından ana belleğe getirildiği zaman prosesin varış zamanını belirtir. Proses ana belleğe getirildikten sonra CPU'ya çalıştırılmak üzere anahtarlanır. Prosesin CPU'da kaldığı toplam süre patlama süresi (burst time) olarak bilinir. Patlama süresi prosesin sadece CPU'da kaldığı süreyi temsil ettiği için prosesin bekleme süresi bu süreye dahil değildir. Bir prosesin çalışma süresini proses çalıştırılmadan tam anlamıyla tahmin etmek oldukça zordur. Prosesin tamamlanması için gerekli olan zaman tamamlanma süresi "completion time" denir. Prosesin ana belleğe gelmesinden tamamlanmasına kadar geçen süre dönüş süresi "turnaround time" denir. Bekleme süreleri dönüş süresine dahildir. Prosesin CPU'ya atanmak için beklediği toplam süre bekleme süresi "waiting time" denir. Prosesin ana belleğe geldikten sonra CPU'ya ilk anahtarlanmasına kadar geçen süre ise tepki süresi (response time)'dir.

2.2. CPU Planlama

Tek programlamada CPU tek bir prosese anahtarlanır ve o proses tamamlanana kadar sadece ona hizmet ederdi. Prosesin bekleme durumlarında CPU atıl

durumda kalır ve verimsizlik meydana gelirdi. Daha çok zamanda daha az görev tamamlanırdı.

Çoklu programlamada CPU tek bir prosese atanmaz. Bekleme sürecinde CPU'ya başka prosesler de atanır ve bekleme süresi başka bir prosesin icra edilmesi şeklinde değerlendirilir. Böylece CPU atıl durumda kalmaz ve daha az zamanda daha fazla görev yerine getirilebilir.

CPU planlayıcısı, çoklu programlamada CPU'dan maksimum derecede yararlanabileceği şekilde proses seçimi gerçekleştirir ve planlamasını yapar. CPU planlayıcısı işletim sisteminin bir parçasıdır. İşletim sistemi bu görevi yerine getirirken çeşitli algoritmalar kullanır.

Bir planlama algoritmasının temel amaçları aşağıda özetlenmektedir (javaTpoint, 2021);

- Maksimum seviyede CPU'dan faydalanmak.
- Bekleme, tepki ve dönüş sürelerini en aza indirmek.
- Maksimum verimlilik elde etmek.

CPU planlayıcısı bu amaçları yerine getirmek için çeşitli algoritmaları kullanır. Yaygın olarak kullanılan algoritmalardan bazıları;

- İlk gelen ilk hizmeti alır. (FCFS)
- Round Robin
- Önce en kısa iş (Shortest Job First- SJF)
- Öncelik temelli planlama (Priority based scheduling-PbS)

Algoritmaların ayrıntılarını sırasıyla inceleyelim:

2.2.1. FCFS Planlama Algoritması

FCFS algoritmasının temel mantığı oldukça basittir. İsminden de kolayca anlaşılacağı gibi bu algoritmayı kullanan CPU planlayıcısı, proseslerin varış sürelerini dikkate alarak prosesleri CPU'ya anahtarlar. İlk gelen proses önceliğe sahiptir. İlk gelen proses, uzun bir CPU işlemine (burst time) ihtiyaç duyuyorsa, diğer prosesler uzun bir süre bu prosesin tamamlanmasını bekleyeceklerdir. Bazı prosesler ise CPU'ya hiç anahtarlanamayacaktır. Bu durum starvation dilimizdeki karşılığı açlık durumunun oluşması demektir.

FCFS algoritmasının uygulanmasının ve anlaşılmasının kolay olması FCFS algoritmasının kullanılma nedenleridir. Fakat FCFS algoritmasının performansı zayıftır. FCFS algoritmasında, proses sonsuz döngüye girerse veya başka herhangi bir sıkıntı meydana geldiğinde kilitlenme durumu yaşanabilir ve açlık probleminin yaşanma ihtimali doğası gereği yüksektir.

FCFS algoritmasının bekleme süresi hesabını örnek senaryo üzerinden gerçekleştirelim:

Anahtar Kavram

Algoritma: Herhangi bir kural ve işlemin sistematik biçimde adım adım uygulanması yoluyla bir sorunun bertaraf edilmesi veya sonuca en hızlı şekilde ulaşılması işlemi.

Örnek: P0, P1, P2, P3 isminde 4 farklı prosesin bekleme kuyruğunda olduğunu düşünelim (Bk. Tablo 2).

Tablo 2. FCFS Proses Bilgileri

Proses	P0	P1	P2	P3
Varış zamanı (AT)	0	1	10	12
Burst time (BT)	3	6	4	1
Tamamlanma süresi (CT)	3	9	14	15

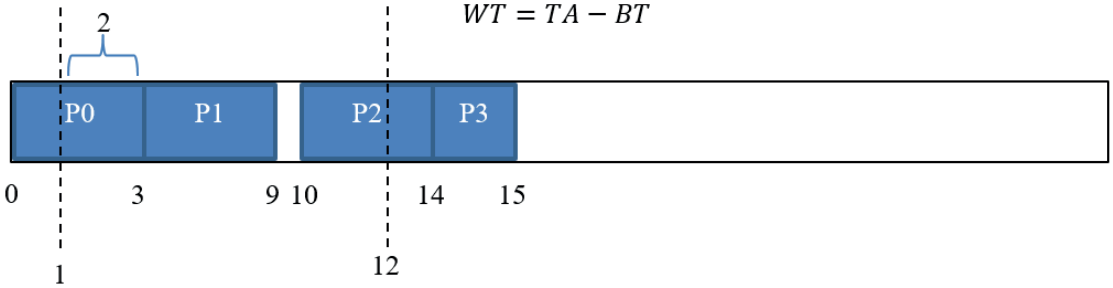
Bu bilgiler ışığında proseslerin dönüş süresi (turnaround) (TA) ve bekleme sürelerini (WT) hesaplayalım. Prosesleri GANTT çizelgesi üzerinde yerleştirelim.

Çözüm:

Dönüş süresi (TA): Prosesin ana belleğe gelmesinden tamamlanmasına kadar geçen süreyi temsil eder.

$$TA = CT - AT$$

$$WT = TA - BT$$



Yukarıda verilen GANTT şemasında görüldüğü gibi 0. zaman noktasında P0 prosesi gelir ve CPU'da işlenme süresi 3 birimdir. Herhangi bir bekleme gerçekleşmediği için tamamlanma süresi de 3 birimdir. TA= 3-0=3 birimdir. P0, CPU'ya anahtarlandıktan 3 birim sonra tamamlanır. WT=0'dır.

P1 prosesi 1. Zaman noktasında gelir. Fakat P0 o sırada işlem görmektedir. Bu nedenle 2 birimlik bir bekleme süresi geçirir. (WT=2). Ancak 3. zaman noktasında CPU'ya anahtarlanabilir ve CPU'da geçireceği zaman birimi 6'dır. P1 prosesinin tamamlanma süresi bu durumda 9 olur. TA=9-1=8 birimdir. P2 prosesi 10. zaman noktasında gelir. O zaman noktasında CPU boştadır ve hemen işleme alınır. WT=0'dır. TA=14-10=4 birimdir. P3 prosesi 12. zaman noktasında ana belleğe gelir. O sırada CPU, P2'yi çalıştırmaktadır. P2, 2 birimlik bir bekleme süresi geçirir. WT=2 birim TA= 15-12=3 birimdir.

2.2.2. Round Robin (RR) Planlama Algoritması

Round robin planlama algoritması işletim sisteminde en çok kullanılan planlama algoritmalarından biridir. Round robin planlama algoritması FCFS algoritmasının yetersiz kaldığı noktayı zaman paylaşımı ile aşar. RR algoritmasında her prosese kuantum adı verilen belirli bir zaman dilimi verilir. Hazır kuyruğunda bulunan her proses kendisine ait olan kuantum süresi kadar CPU'da kalır. Bu zaman dilimi içerisinde proses işlemini tamamlayabilir veya tamamlanamadan tekrar hazır kuyruğuna gönderilir. Tekrar sırasının gelmesini bekler. Bu şekilde sonsuz döngülerin sebep olduğu kilitlenmelerin önüne geçilmiş olur ve açlık problemi oluşmaz.

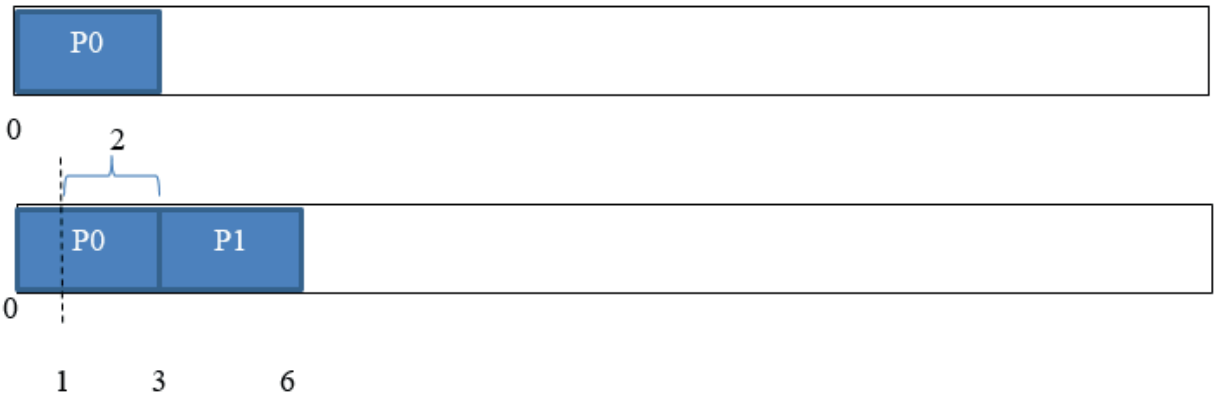
Örnek: Po, P1, P2 isminde 3 farklı prosesin bekleme kuyruğunda olduğunu düşünelim (Bk. Tablo 3). Sistemin kuantum süresinin 3 olduğu varsayalım. RR algoritması kullanılarak GANTT şemasını çizelim.

Tablo 3. RR Örnek Proses Bilgileri.

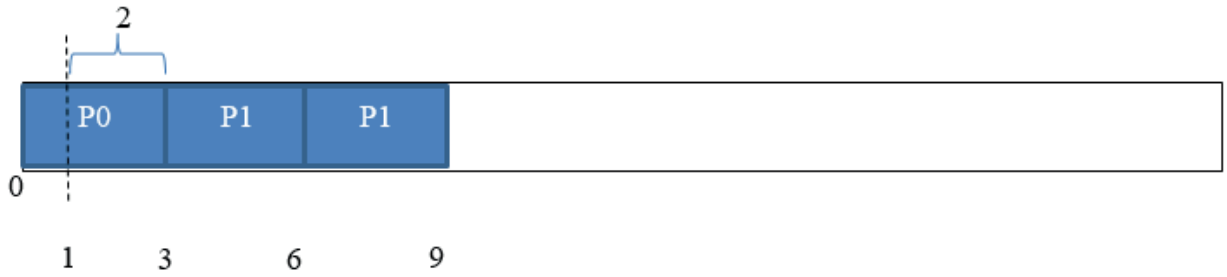
Proses	P0	P1	P2
Varış zamanı (AT)	0	1	10
Burst time (BT)	3	6	4

Bu bilgiler kullanılarak GANTT şemasını çizelim.

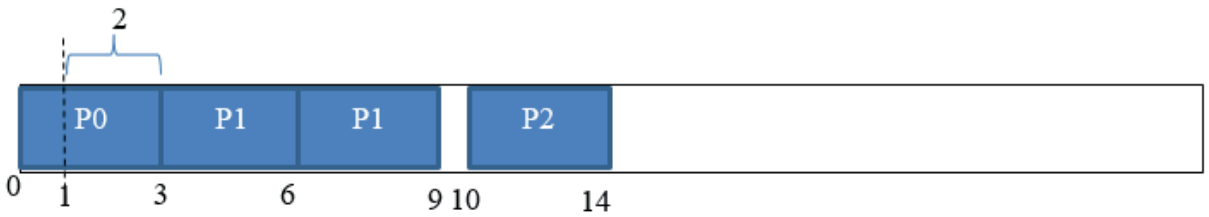
Çözüm:



P1 prosesi tamamlanmadan CPU'dan ayrılmıştır. P1 prosesinin 3 birimlik daha görevi vardır. P2, 10. Zaman noktasında gelecektir. İşlem sırası Po'dadır. Po, 3 birimlik kuantum süresi boyunca tamamlanmıştır. Bu nedenle işlem sırası tekrar P1'dedir. P1, 3 birim daha CPU'da işlenir ve tamamlanır.



10. zaman noktasında P2, CPU'da işleme alınır ve başka bir proses bekleme kuyruğunda olmadığı için P2 tamamlanır.



2.2.3. SJF Planlama Algoritması

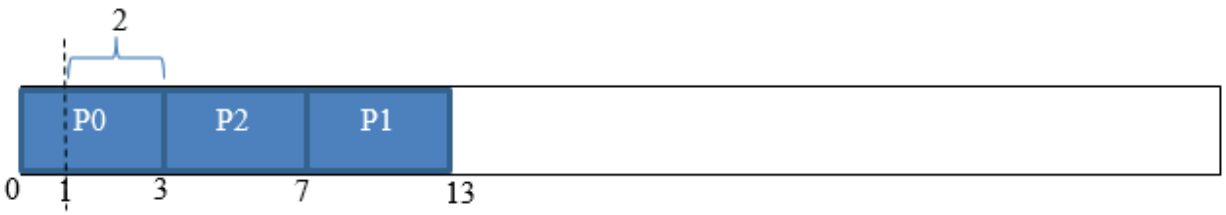
SJF algoritması, patlama süresini (burst time-BT) dikkate alarak planlama işlemini gerçekleştirir. En az iş yüküne sahip olan proses en önce işleme alınır. Böylece ortalama bekleme süresi minimize edilmeye çalışılır. SJF algoritmasında uzun bir BT süresine sahip bir proses çok beklemek zorunda kalabilir.

Örnek: P0, P1, P2 isminde 3 farklı prosesin bekleme kuyruğunda olduğunu düşünelim (Bk. Tablo 4). SJF algoritması kullanılarak GANTT şemasını çizelim.

Tablo 4. SJF Örnek Proses Bilgileri

Proses	P0	P1	P2
Varış zamanı (AT)	0	1	2
Burst time (BT)	3	6	4

Çözüm:



2.2.4. Öncelik Temelli Planlama Algoritması

Öncelik temelli planlama algoritmasında her prosese bir öncelik değeri atanır. Bu değer bir sayısal ifadedir. Bazı sistemlerde bu sayı değerinin yüksek olması prosesin daha öncelikli olduğunu gösterirken bazı sistemlerde de prosesin öncelikli olabilmesi için prosesin öncelik değerinin düşük olması gerekir. Önceliği yüksek olan proses ilk önce CPU'ya atanır.

Öncelik değeri, proseslerde değişken gösterebilir veya sabit kalabilir. Proses süreci boyunca öncelik değeri değişmezse sabit öncelik, kendini değiştirmeye belirli aralıklarla değiştirmeye devam ederse dinamik öncelik olarak adlandırılır (javaTpoint, 2021).

Öncelik temelli planlama algoritmalarında 2 farklı yaklaşım kullanılmaktadır. İlk yaklaşımda başkasına engel olmayan bir öncelikli planlamadır. Bu planlama türünde önceliği yüksek olan bir proses çalışır durumdayken kendisinden daha öncelikli bir prosesin o süre içerisinde gelmesi, prosesi etkilemez çalışması tamamlanana kadar CPU tarafından çalıştırılmaya devam edilir. Bu şekilde olan öncelikli planlama algoritmasında kesintisiz (non-preemptive priority) planlama denir. Eğer ki önceliği yüksek olan bir proses çalışır durumdayken kendisinden daha öncelikli bir prosesin gelmesi, prosesi etkilerse ve CPU yeni gelen önceliği daha yüksek olan prosese anahtarlanırsa öncelikli planlama algoritması kesintili (preemptive priority) olarak anılır.

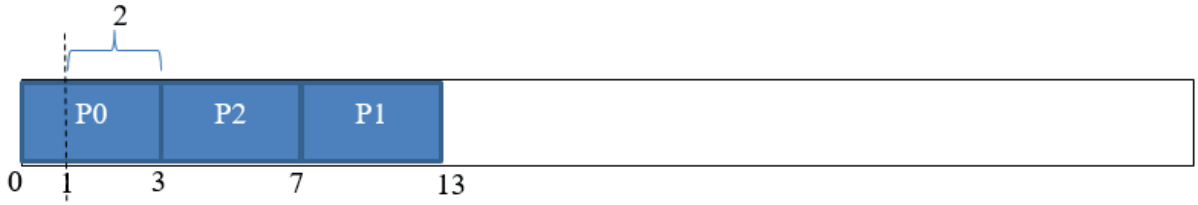
Kesintisiz ve kesintili algoritmaları arasındaki farkı daha iyi anlayabilmek için birer örnek çözelim.

1. ***Kesintisiz öncelikli planlama algoritması:***

Örnek: P0, P1, P2 isminde 3 farklı prosesin bekleme kuyruğunda olduğunu düşünelim (Bk. Tablo 5). Kesintisiz (Non-preemptive) öncelikli planlama algoritması kullanılarak GANTT şemasını çizelim.

Tablo 5. Non-Preemptive Öncelikli Planlama Algoritması Örnek Proses Bilgileri

Proses	P0	P1	P2
Varış zamanı (AT)	0	1	2
Burst time (BT)	3	6	4
Öncelik değeri	2 (Düşük)	3	4



Çözüm:

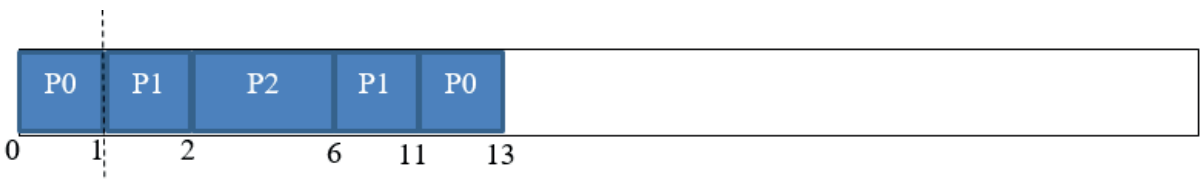
2. Preemptive öncelikli planlama algoritması:

Örnek: Po, P1, P2 isminde 3 farklı prosesin bekleme kuyruğunda olduğunu düşünelim (Bk. Tablo 6). Kesintili (Preemptive) öncelikli planlama algoritması kullanılarak GANTT şemasını çizelim.

Tablo 6. Preemptive Öncelikli Planlama Algoritması Örnek Proses Bilgileri

Proses	P0	P1	P2
Varış zamanı (AT)	0	1	2
Burst time (BT)	3	6	4
Öncelik değeri	2 (Düşük)	3	4

Çözüm:



2.3. Proses Senkronizasyonu

Bazı prosesler birbirleriyle ilişkilidir. Bir proses; ebeveyn(parent) proses veya çocuk (child) proses olabilir. Çocuk proses, ebeveyn prosesinden doğar. Ebeveyn prosesler, çocuk prosesi ile eş zamanlı çalışabilir. Çocuk proseslerin kendi aralarında ortak kullandıkları bir kaynak olabilir. Çocuk prosesler ebeveyn prosesi ile ortak bir kaynak paylaşabilir. Ebeveyn prosesler başka ebeveyn proseslerle iş birliği yapabilir. Bu şekilde birden fazla proses birbirleriyle iş birliği yaptığında birbirlerinin çalışmasını engellemeyecek şekilde proses seçimlerinin yapılması gerekir. Aksi takdirde çıktılarda hata meydana gelebilir. Prosesler arasındaki bu dengenin korunması ve prosesler arasındaki iletişimin sağlıklı bir şekilde devam edebilmesi için proseslerin senkronize edilmesi gerekir. Çeşitli senkronizasyon mekanizmaları vardır.

Prosesler arasında meydana gelen 2 temel olay vardır:

1. Yarış durumu: Birden fazla proses aynı anda belleğin aynı alanını okumaya, yazmaya veya aynı alana erişmeye çalıştıklarında oluşur. Bir proses paylaşılan bir kaynağa erişmek istediğinde bu kaynağı kullanan diğer proseslerin bu kaynak üzerindeki çalışmasının bitmiş olması gerekir.

2. Kritik bölge: Aynı kaynağa erişmek isteyen birden fazla proses yarış durumuna geçer. Bir prosesin paylaşılan kaynaklar üzerinde işlem yaptığı kod parçası kritik bölge olarak isimlendirilir. Bir proses kritik bölgedeyse başka bir proses kritik bölgede olmamalıdır.

Kritik bölge yönetimi işletim sistemi tarafından yapılmaktadır. İşletim sistemi proseslerin yarış durumunun ortaya çıkmaması ve kritik bölge problemini çözmek için senkronizasyon mekanizmalarına başvurur. Mekanizmalar aşağıda belirtilen koşulları sağlamalıdır:

1. Karşılıklı dışlama: Bir proses kritik bölgedeyse başka bir proses kritik bölgeye giremez. Karşılıklı dışlama prosesler arasındaki yarış durumunu engeller.

2. İlerleme (Progress): Kritik bölgede herhangi bir proses bulunmuyorsa kritik bölgeye girmek isteyen başka proseslerden birine izin verilmelidir.

3. Sınırlı bekleme (Bounded Waiting): Proses kritik bölgeye girmek için sonsuz bir bekleme yapmamalıdır.

Kritik bölge problemine yazılımsal, donanıma dayalı ve hem yazılıma hem de donanıma dayalı çözümler geliştirilmiştir.

Anahtar Kavram

Algoritma: Herhangi bir kural ve işlemin sistematik biçimde adım adım uygulanması yoluyla bir sorunun bertaraf edilmesi veya sonuca en hızlı şekilde ulaşılması işlemi.

2.3.1. Yazılıma Dayalı Çözümler

1. Ortak Değişken Kullanımı: Bu mekanizmada global bir kilit değişkeni ("lock" isminde) kullanılır. "Lock" ismindeki değişken 1/0 değerlerini alır. Değerin 0 olması kritik bölgede herhangi bir prosesin bulunmadığını 1 değerinin olması kritik bölgede bir prosesin çalışır durumda olduğunu gösterir. Diğer proseslerin bu değişkenin değerine bakılarak kritik bölgeye girmesi engellenir.

```
1    While (lock!=0);
2    lock=1;
3    //Kritik bölge
4    Exit Section
5    lock=0;
```

Kod 3. Lock Değişkeni Kullanımı

"lock" değişkeninin değeri 1 olduğunda kritik bölgede bir prosesin bulunduğu anlamına geldiğinden while döngüsü içerisinde bu prosesin kritik bölgeden çıkması beklenir. "lock" değişkeninin değerinin 0 olması ile artık kritik bölge serbesttir ve diğer kuyrukta bekleyen proseslerden biri kritik bölgeye girebilirler.

Bu çözüm başlangıçta kritik bölge problemini çözdüğü düşünülebilir. Fakat bazı durumlarda kritik bölge probleminin çözümü için gerekli olan karşılıklı dışlama özelliğini sağlayamamaktadır. Senaryoyu inceleyelim ve 2 farklı prosesin aynı anda kritik bölgeye nasıl girdiğini görelim.

P1 ve P2 isminde 2 prosesimiz mevcut olsun. P1 prosesi kritik bölgeye girmek istiyor ve "lock" değişkeninin değeri 0. P1 prosesi Kod 3' de bulunan kod parçasının 1. Satırı CPU tarafından işlenir. CPU tam bu esnada P2 prosesine anahtarlanabilir. (P1'e ayrılan zaman diliminin süresinin dolması veya öncelik değerinin daha yüksek olması gibi sebeplerden) P1 prosesi, "lock" değişkeninin değerini daha değiştiremeden P2 prosesi Kod 3'te bulunan kod parçasının 1. Satırına gelir ve "lock" değişkeninin değerinin 0 olduğunu görür ve kritik bölgeye girer. Bu durumda hem P1 hem de P2 prosesi kritik bölgededir. Bu çözüm yeterli değildir.

2. Global Paylaşılan "turn" Değişkeni:

Bu yaklaşım yalnızca 2 proses arasındaki senkronizasyonu sağlamak için kullanılabilir. "turn" isminde bir kilitleme değişkeni vardır. Bu değişken 2 proses arasında paylaştırılır. Yaklaşımın sözde kodu Kod 4 ve Kod 5 verilmektedir.

```
1    While (turn!=i);
2    //Kritik bölge
3    turn =j;
```

Kod 4. Proses_i İçin Kritik Bölgenin Sözde Kodu

```
1    While (turn!=j);
2    //Kritik bölge
3    turn =i;
```

Kod 5. Proses_j İçin Kritik Bölgenin Sözde Kodu

Bu yaklaşımda, "lock" değişkeni yaklaşımında karşılıklı dışlamanın garanti edilememesi problemi çözümlenmiştir. "lock" değişkeni yaklaşımı 2'den fazla proses için uygulanabiliyordu. "turn" değişkeninde ise sadece 2 proses kullanılır. "turn" değişkeni i veya j değerlerinden birine sahip olabilir. Sözde kodları inceleyecek olursak eğer, "turn" değişkeninin değeri i olduğunda kritik bölgeye girme hakkının proses_i'ye aittir. "turn" değişkeninin değeri j olduğunda kritik bölgeye girme hakkı proses_j'ye aittir. Kod 5'te proses_i, "turn" değişkeni j değerine sahip olduğu sürece proses_j'yi bekler. Proses_j'nin kritik bölgede olduğunu anlar. Ne zaman ki proses_j kritik bölgede çalışmasını tamamlar ve

"turn" değişkenini i değeri yapar; o zaman proses_i, while döngüsünden çıkar ve kritik bölgeye girer. Aynı durum proses_j için de geçerlidir. Böylece 2 proses arasındaki karşılıklı dışlama sağlanmış olur.

Bu yaklaşımda, ilerleme (progress) koşulu garanti edilemez. Kritik bölgeye girmeye hak kazanan prosesin (örneğin proses_1) kritik bölgede sonsuz döngüye girmesi sonsuz süre için kilitlenmeye neden olacaktır. Bu durumlarla karşılaşılması diğer prosesin (örneğin proses_2) çok uzun süre beklemesine hatta hiç sıra gelmemesine neden olur. Aynı zamanda bu yaklaşımda proseslerin sıra ile çalışma zorunluluğu vardır.

3. İlgili Değişken Mekanizması (Interested Variable Mechanism):

Senkronizasyon mekanizmasında ilerleme koşulunun sağlandığından emin olunmalıdır. "turn" değişkeni yaklaşımında bu koşul garanti edilmemektedir. Aynı zamanda sıra ile çalışma zorunluluğu ortadan kaldırılmalıdır. Ekstra değişken tanımlamasıyla bu durumlar aşılabilir. Yaklaşımın sözde kodu Kod 6 ve Kod 7'de verilmiştir.

```
1 flag[i] =true;
2 While (flag[j]== true);
3 //kritik bölge
4 flag[i]=false;
```

Kod 7. Proses_i için kritik bölgenin sözde kodu.

Bu yaklaşımda ilgilenilen ekstra değişken mantıksal bir değer "true, false" alır. Kod 6 ve Kod 7 verilen sözde kod parçasının 1 ve 5 numaralı satırlar proseslerin kritik bölgeye girmek isteyip istemedikleri belirtir. Proses_j kritik bölgeye girmek isterse eğer flag değişkeninin i. indisinin değerini true yapar. Prosesler kritik bölgeye girmeden önce diğer prosesin kritik bölgeye girmek isteyip istemediğini 2 ve 6. satırlarda kontrol eder ve diğer prosesin kritik bölgeye girme isteği varsa proses onu bekler. Kritik bölgede çalışmasını tamamlayan proses flag değişkeninin değerini false yaparak diğer prosese kritik bölgeye girmek istemediğini beyan eder.

```
5 flag[j] =true;
6 While (flag[i]== true);
7 //kritik bölge
8 flag[j]=false;
```

Kod 6. Proses_j için kritik bölgenin sözde kodu.

Sözde kodu daha derinden inceleyelim. CPU, proses_i'nin 1. satırı işlendikten sonra proses_j'ye anahtarlandığını düşünelim. Proses_j'nin de 1. satırı çalıştırılır. Bu durumda her iki prosesin flag[i] ve flag[j] değerleri "true" değerine sahip olur. Bu durumda kilitlenme meydana gelir. Her iki proseste birbirlerinin "flag" değişkenlerinin değerini "false" yapmasını bekler.

```
flag[i] =true;
While (flag[j]){
flag[i]=false;
//delay
flag[i]=true;}
//kritik bölge
flag[i]=false;
```

Kod 8. Proses_i için kritik bölgenin sözde kodu.

```
flag[j] =true;
While (flag[i]){
flag[j]=false;
//delay
flag[j]=true;}
//kritik bölge
flag[j]=false;
```

Kod 9. Proses_j için kritik bölgenin sözde kodu.

Kod 8 ve Kod 9'da verilen sözde kodlar incelendiğinde bir proses while döngüsü içerisindeyken diğer prosesin kritik kesime girmesine izin verilmektedir. Karşılıklı dışlama koşulu sağlanmaktadır. Bu sözde kod parçası livelock durumuna neden olabilir.

Livelock (Canlı kilitlenme): İki veya daha fazla proses yararlı bir iş yapmadan diğer proseslerin değişikliklerine karşılık durumlarını sürekli olarak değiştirdikleri bir durumdur. Bu durumu Kod 8 ve Kod 9 'da verilen sözde kod üzerinde inceleyelim.

- Pi, [i] işaretini true olarak ayarlar.
- Pj, [j] işaretini true olarak ayarlar.
- Pi bayrağı [j] kontrol eder.
- Pj [i] bayrağını kontrol eder.
- Pi, [i] işaretini false olarak ayarlar.

- Pj, [j] işaretini false olarak ayarlar.
- Pi, [i] işaretini true olarak ayarlar.
- Pj [j] işaretini true olarak ayarlar.

Bu akış süresiz bir şekilde uzatılabilir.

4. Dekker Algoritması Çözümü

Dekker algoritması senkronizasyon için gerekli olan tüm koşulları sağlar. Dekker algoritması Kod 10 ve Kod 11'de verilmektedir.

```
flag [i] =True;
While (flag[j])
If(turn==j)
{flag[i]=False;
While(turn==j);
flag[i]=true;}
//kritik kesim
flag[i]=False;
turn=True;
```

Kod 10. Proses_i için dekker algoritması.

```
flag [j] =True;
While (flag [i])
if (turn = = i)
{flag [j] =False;
While (turn = = i);
flag [j] =True;
}
//kritik kesim;
flag [j] =False;
```

Kod 11. Proses_j için dekker algoritması.

5. Peterson Algoritması Çözümü

Peterson çözümü senkronizasyon için gerekli olan tüm koşulları sağlamaktadır. Peterson algoritması ikiden fazla proses için genelleştirilebilir (Gülbağ, 2017).

```
flag [i] =True;
turn =j
While (flag [j]&& turn==j);
//kritik kesim;
flag [i] =False;
```

Kod 12. Proses_i İçin Peterson Algoritması

```
flag [j] =True;
turn =i
While (flag [i]&& turn==i);
//kritik kesim;
flag [j] =False;
```

Kod 13. Proses_j İçin Peterson Algoritması

2.3.2. Donanıma Dayalı Çözümler

Yazılıma dayalı çözümlerde proseslerin sürekli kaynağın serbest olup olmadığını kontrol etmesi CPU'yu gereksiz yere yorar. Dekker ve Peterson algoritmaları karmaşıktır ve karşılıklı dışlama mekanizmalarında kullanımı donanıma dayalı çözümlere göre daha zordur.

1. Kesmeleri aktif / pasif yapmak: Proses kritik bölgeye girmeden önce kesmeler pasif yapılarak başka bir prosesin CPU'ya anahtarlanması engellenir. Bu çözüm yolu çok işlemcili sistemlerde zaman paylaşımlı olarak CPU'dan yararlanma yaklaşımına aykırıdır.
2. Özel ayrıcalıklı makine komutlarını kullanmak: Proses kritik bölgeye girmeden önce özel atomik donanım komutları ile başka bir prosesin kritik bölgeye girmesi engellenir. Bu komutlar tek bir komut çevriminde gerçekleşen ve kesintiye uğramayan komutlardır. Bu komutlara örnek olarak testandset komutları, compare, swap, sleep-wake komutları verilebilir.

Donanıma dayalı çözümler karşılıklı dışlama ve ilerleme koşullarını sağlamaktadırlar fakat sınırlı bekleme koşulunu sağlayabilmesi için ek mekanizmalarla birlikte kullanılmaları gerekir.

2.3.3. Donanım ve Yazılıma Dayalı Çözümler

1. Semaforlar:

1965 yılında Edsger Dijkstra'nın prosesler arasındaki senkronizasyonu sağlamak için semafor yapısını geliştirmiştir. Semafor yapısında P() ve V() işlemleri tanımlanır. V() işleminde s değişkeni arttırılır ve P() işleminde ise test

edilir ve s değişkeni azaltılır. Proses senkronizasyonuda P() ve V() işlemleri kullanılır (Gülbağ, 2017).

$$V(s) = s = s + 1$$

$$P(s) = while(s \leq 0)\,\{bekle\}; s = s - 1$$

Semaphore s=1;

```
P(s);
//kritik kesim
V(s);
```

Kod 14. Proses0-semaphore

```
P(s);
//kritik kesim
V(s);
```

Kod 15. Proses1-semaphore

Kod 14 ve Kod 15'te verilen kodlarda ikili semaphore yapısından faydalanılmıştır. Semaphore yapısı kullanılarak üretici-tüketici problemi gibi problemlerin çözümünde sayan semaforlardan faydalanılır.

2. Monitörler:

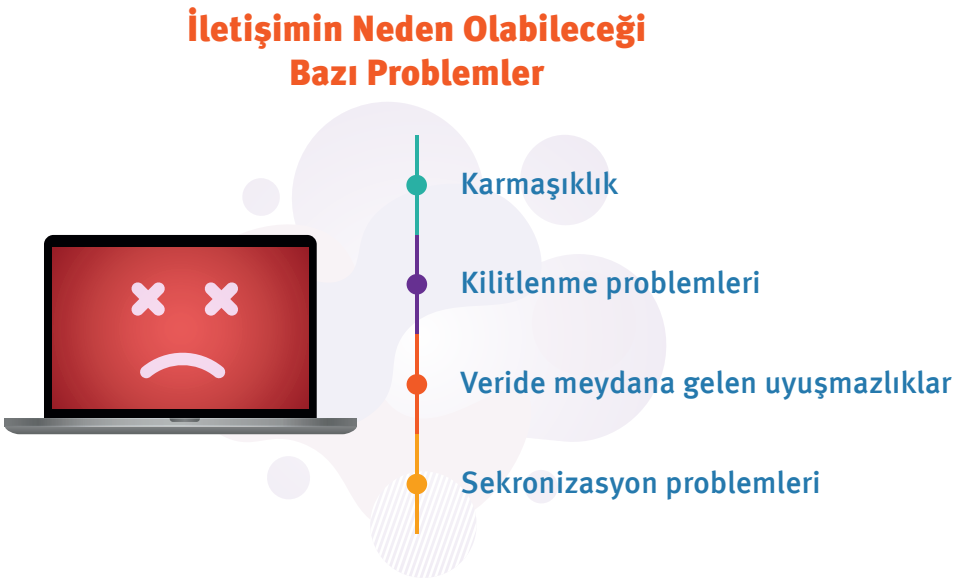
Prosesler arasındaki senkronizasyonu sağlamak için geliştirilen programlama dili desteği gerektiren üst seviyeli bir yapıdır. Monitörler içerisinde proseslerin kritik bölgeleri tutulur. Senkronizasyonu sağlamak adına monitörlerin sunduğu en önemli özellik, bir anda sadece bir işlemin monitörlerde aktif olmasıdır (Gülbağ, 2017).

2.4. Prosesler Arası İletişim

Prosesler arasındaki senkronizasyonu sağlamak için prosesler arasında bilgi akışının sağlanması gerekir. Aynı adres uzayında paylaşım yapan prosesler arasındaki iletişim için paylaşılabilen bellek veya paylaşılan adres yöntemleri kullanılmaktadır. Farklı adres uzaylarında paylaşım yapan prosesler arasındaki iletişim için özel çekirdek seviyesinde veri yapıları, mesaj geçişleri gibi yöntemler kullanılmaktadır. Prosesler arasındaki iletişim için kullanılan mekanizmalardan bazıları;

Şekil 4. İletişim İçin Kullanılan Mekanizmalar

Prosesler arasındaki iletişimin neden olabileceği bazı problemler vardır:



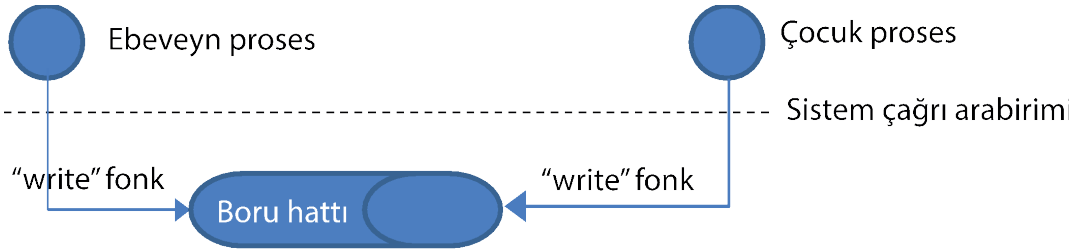
Şekil 5. İletişimin Neden Olabileceği Bazı Problemler

2.4.1. Boru Hattı (Pipe) Modeli

Boru hattı modeli ilk gelen ilk çıkar çalışma mantığını temel alan bir veri yapısıdır. Pipe yapısı farklı adres uzayına sahip iki prosesin birbirleriyle iletişim kurmasını sağlar. Prosesler "write" ve "read" komutlarını kullanarak aralarında bulunan boru hattı veri yapısına bilgiyi yazar ve boru hattı veri yapısından bilgiyi okur. Boru hattı veri yapısının kapasitesi sınırlıdır. Bir proses boş bir boru hattından veri okumaya çalışırsa proses bloke olur. Eğer kapasitesi tamamen dolmuş olan boru hattına bir proses veri yazmaya çalışırsa proses bloke olur.

Boru hattı modeli 2 farklı yapıya sahiptir: Sıradan boru hatları ve isimlendirilmiş boru hatları.

1. Sıradan boru hatları (Ordinary pipe model): Bu veri modelinde üretici-tüketici tipi bir veri iletişimi mevcuttur. Veri iletişimi tek yönlüdür ve üretici borunun yazma ucundan bilgiyi yazar. Tüketici, borunun okuma ucundan yazılan bilgiyi ilk gelen ilk çıkar çalışma mantığını temel alarak okur (Çizim 11). Sıradan boru hatları ile iletişim kuran prosesler ebeveyn-çocuk ilişkisi içerisindedir. Windows işletim sistemlerinde sıradan boru hatları anonymous pipe olarak anılır.



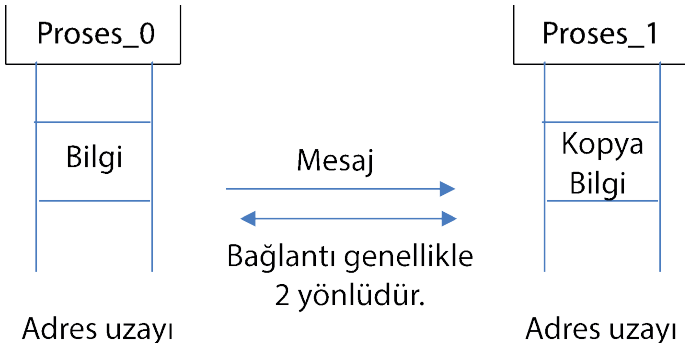
Çizim 11. Sıradan Boru Hattı

2. İsimlendirilmiş boru hatları (Named pipes): İsimlendirilmiş boru hatları, çift yönlü ve yarı çift yönlü olarak kullanılabilmektedir. İsimlendirilmiş boru hatlarında, sıradan boru hatlarında olduğu gibi "ilk giren ilk çıkar" çalışma prensibi temel alınır. Bu boru hattı veri yapısını kullanan proseslerin ebeveyn-çocuk ilişkisi içerisinde olmasına ihtiyaç yoktur. Birden fazla proses boru hattına bilgi yazabilir, tek bir proses bu bilgileri okur. Okuma ve yazma işlemleri mesaj geçiş yoluyla gerçekleştirilir. Unix, Windows işletim sistemlerinde kullanılır.

2.4.2. Mesaj Geçişi Yöntemi

Mesaj geçişi yöntemi, farklı adres uzayına sahip proseslerin haberleşmelerini ve bilgi paylaşmalarını sağlar. Dağıtık bir mimariye sahip bilgisayar sistemlerinde farklı bilgisayarlardaki proseslerin birbirleriyle haberleşmesini sağlar. Mesaj geçişi yönteminde mesajın gönderimi ve alımı için atomik yapıda komutlar kullanılır. Bilgi alışverişi, bilginin doğrudan bilgisayara gönderilerek yapılabileceği gibi port veya posta kutularına gönderilerek de yapılabilir. Komutlarla bilgisayarlar arasında bilginin doğrudan alışverişinin yapılmasına doğrudan haberleşme denir. Port veya posta kutuları kullanılarak bilgi alışverişinin yapılmasına dolaylı haberleşme denir. Doğrudan ve dolaylı haberleşmenin her ikisinde de komutlar kullanılır.

Doğrudan haberleşmede mesajlar göndericinin ve alıcının adresini içermek zorundadır. Send (A, mesaj) fonksiyonuyla A prosesinin adres uzayına mesaj gönderilir. Receive (B, mesaj) fonksiyonu ise B prosesinin adres uzayından mesajı alır. Bu haberleşme türünde 2 proses arasında doğrudan bir bağlantı kurulur.



Çizim 12. Prosesler Arasındaki Doğrudan İletişim

Dolaylı haberleşmede mesajlar port veya posta kutularına bırakılır ve tekrar port veya posta kutularından okunur. Posta kutusu paylaşılabilirdir. Posta kutusu proseslerin adres uzayında oluşturulabilir veya kullanıcı adres uzayında oluşturulabilir. Haberleşme için yeni bir posta kutusu oluşturulur ve prosesler posta kutusu aracılığı ile haberleşir. Haberleşme işlemi bittikten sonra posta kutusu kaldırılır. Haberleşme send() ve receive() komutları ile gerçekleştirilir.

Gönderici göndereceği bilgiye karar verir ve özel bir hedef belirler. Hedef proses bilgiyi almaya karar verdiğinde prosesler arasındaki iletişim ağı kurulur. İletişim ağı, ikiden fazla proses arasında oluşturulabilir. Birbirlerine bağlı birden fazla proses tarafından mesaj gönderim işlemi gerçekleştirilebilir fakat mesaj alma işlemi için tek bir prosese izin verilir.

Send() komutu asenkron ve senkron yapıda olabilirler. Asenkron send() komutu bilgiyi alıcının posta kutusuna gönderir ve alıcının bilgiyi almasını beklemez. Senkron send() komutu ise alıcı ile göndericinin uyum içerisinde bilgi alışverişini gerçekleştirmesini sağlar. Gönderici bilgiyi alıcının posta kutusuna gönderir ve alıcının bu bilgiyi posta kutusundan alıncaya kadar alıcıyı bekler.

Receive() komutu bloklanan ve bloklanmayan yapıda olabilirler. Bloklanan receive() komutunda alıcı mesajı alana kadar bekler. Bloklanmayan receive() komutunda alıcı posta kutusunda mesaj bulamasa da, mesajı alamasa da işlemine devam eder.

Farklı yapılara sahip Send() ve receive() komutları bir arada kullanıldığında yapılarına özgü davranışlar da bir araya gelir ve yapıların birlikteliklerinden doğan davranışı sergilerler. Örneğin; Asenkron send()-bloklanmayan receive() bir arada kullanıldığında gönderici mesajı alıcıya gönderir ve alıcının mesajı alıp almadığıyla ilgilenmez ve işine devam eder. Bloklanan receive() ise posta kutusundan mesajı alana kadar bloklanır.

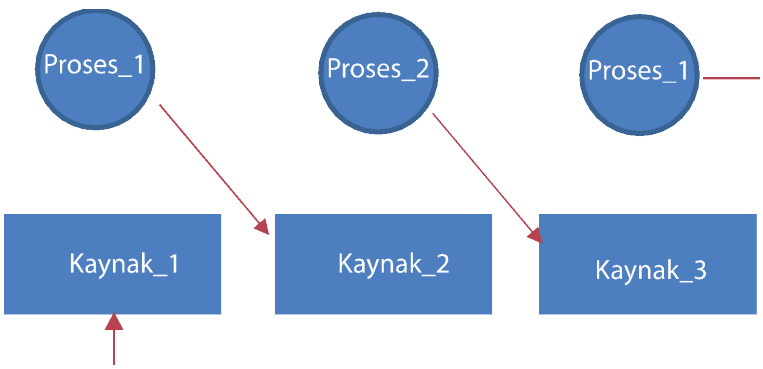
2.5. Kilitlenme (Deadlock)

Prosesler, çalışabilmeleri için kaynaklara ihtiyaç duyarlar. Prosesler kaynakları sıra ile alırlar. İşletim sistemi prosesin istediği kaynak boşta ise

kaynağı prosese verir. Başka bir proses tarafından kullanımda ise proses, kaynak alımı için bekletilir. Proses ihtiyaç duyduğu kaynağı kullandıktan sonra kaynağı serbest bırakır.

Birbirleriyle kaynak paylaşımı olan proseslerin her biri bazı durumlarda döngüsel olarak bir başka prosesin elinde olan bir kaynağı talep edebilir. Bu durumda, proseslerin hiçbiri çalışamayacaktır. Prosesin ihtiyaç duyduğu kaynak ne zaman ki serbest bırakılır o zaman proses çalışmasına devam eder.

Çizim 13'te Proses_1, Kaynak_1'i elinde tutmaktadır ve Proses_2'nin elinde tuttuğu Kaynak_2'yi talep etmektedir. Proses_2, Kaynak_2'yi elinde tutmaktadır ve Proses_3'ün kullandığı Kaynak_3'ü talep etmektedir. Proses_3, Kaynak_3'ü tutmakta ve Proses_1'de bulunan Kaynak_1'i talep etmektedir. Proses_1, Proses_2, Proses_3 çalışamaz durumdadır. Çalışabilmeleri için gerekli olan kaynaklar bir başka prosesler tarafından tutulmaktadır. Proseslerin hiçbirinin çalışamaz olduğu bu duruma kilitlenme denir.



Çizim 13. Kilitlenme

Kilitlenme durumunda prosesler ihtiyaç duydukları kaynakları bir türlü alamazlar. Proseslerin kaynakları uzun bir süre alamaması açlık durumunun oluşmasına sebep olur. Her açlık durumu kilitlenme oluşumunun sebebi değildir fakat her kilitlenme olayında açlık durumu oluşur. Kilitlenme durumunda prosesler kaynakları sonsuza kadar beklerler. Açlık durumunda kaynaklar sonsuza kadar beklenmez, uzun süre beklenir. Açlık durumunun oluşması yüksek önceliğe sahip proseslerin düşük öncelikli prosesler üzerindeki etkilerinden biridir.

Kilitlenmenin oluşabilmesi için 4 adet gerekli koşul bulunmaktadır:

1. Karşılıklı dışlama (Mutual exclusion): Bir kaynak aynı anda birden fazla proses tarafından kullanılamaz. Bir kaynak bir proses tarafından kullanılıyorsa bu kaynağın diğer prosesler tarafından aynı anda kullanımı engellenir.
2. Tut ve bekle (Hold and wait): Bir proses bazı kaynakları tutuyorken aynı zamanda bazı kaynakları beklemesi durumudur.

3. Kesme durumu yok (No preemption): Bir proses tamamlanıncaya kadar yürütülmeye devam ettirilir. Kesme yoktur. Zamanlayıcı tarafından başka bir proses planlanamaz.
4. Döngüsel bekleme: Bütün prosesler döngüsel bir şekilde bir başka prosesin elinde tuttuğu kaynağı talep eder ve bekler.

Kilitlenmenin oluşabilmesi için 4 koşulun aynı anda gerçekleşmesi gerekir.

Kilitlenme durumunun engellenmesi ve kilitlenme durumunda kullanılan yaklaşımlar;

1. Kilitlenmeyi önlemek:

Kilitlenmeyi önlemek için 4 koşuldan biri kaldırılır veya döngüsel bekleme koşulu engellenir. Kilitlenmeyi önlemek için uygulanan örnek uygulamalar;

- Bir proses bir kaynak talep ediyorsa bu prosese bu kaynağı tahsis etmeden önce prosesin elinde tuttuğu tüm kaynakları bırakması istenir.
- Bir prosesin ihtiyaç duyduğu tüm kaynaklar proses çalıştırılmadan önce tahsis edilir (Pratikte uygulaması oldukça zordur. Çünkü bir prosesin başlangıçta ihtiyaç duyduğu tüm kaynakları tespit edemez).
- Kesme işlemi aktif edilir.
- Kilitlenmeye neden olan prosesin tuttuğu kaynağın elinden alınması (İyi bir yaklaşım değildir. Çünkü prosesin yaptığı tüm işler çöpe gider. Tutarsızlık meydana gelebilir).
- Kaynağı bırakma ve tahsis etme işlemleri uyumlu bir şekilde gerçekleştirilerek döngüsel beklemenin önlenmesi

2. Kilitlenmeden kaçınmak:

Kilitlenmeden kaçınmada her adımda sistemin güvenli durumda olup olmadığı kontrol edilir. İşletim sistemi kaynak atamalarının her birini gözden geçirir, güvensiz bir durumun oluşması engellenir. Güvensiz bir durum oluşturan prosesin çalışması durdurulur, kaynak ataması engellenir.

Banker algoritması Dijistra tarafından geliştirilmiştir. Kilitlenmeden kaçınmak için kullanılan bir algoritmadır. Her proses ne kadar kaynağa ihtiyaç duymaktadır? Proses şu anda ne kadar kaynağı elinde tutmaktadır? Ne kadar kaynak ulaşılabilir durumdadır? Bilgileri ile kaynak atama planlamasının gelecekte kilitlenmeye neden olup olmayacağı belirlenir. Kilitlenmeye neden olmayacak bir proses çalışma sırası keşfedilirse güvenli olarak kabul edilir ve proseslerin çalışmasına izin verilir. Eğer ki planlama bir kilitlenmeye neden olacaksa;

- Kilitlenmeye neden olacak proses başlatılmaz.
- Kaynak talebi kilitlenmeye neden olacaksa kaynak tahsisi yapılmaz.

3. Kilitlenme durumunu tespit etmek ve bu durumdan kurtulmak:

Kilitlenme durumu oluştuğunda kilitlenmenin meydana geldiğini bilmek ve bu durumdan kurtulmak için yapılan eylemlerdir. Kilitlenme durumunu tespit edebilmek için kaynak atama grafiğinden faydalanılır. Kaynak tahsis grafiğinde sistemin durumu görülür. Bu grafik kaynakları tutan ve kaynak talebinde bulunan tüm prosesler hakkındaki bilgileri barındırır. Kaynak tutma matrisi ve kaynak talep etme matrisleri de kilitlenme durumunun tespitinde yardımcıdır.

Kilitlenme meydana geldiğinde bu durumdan kurtulabilmek için;

- Kilitlenmeye neden olan prosesin çalışması yarıda kesilebilir. Bu prosese tahsis edilen kaynaklar elinden alınabilir.
- Kilitlenmenin yaşandığı tüm proseslerin çalışması durdurulabilir.
- Kilitlenmenin yaşandığı proseslerin çalışması birer birer durdurulabilir.
- Kilitlenmenin yaşandığı prosesler belirli bir noktaya kadar geri çekilip tekrardan proseslerin çalışması başlatılabilir (Doğu Akdeniz Üniversitesi B. V., 2021).

Bu Bölümde Ne Öğrendik?

* Tüm hesaplamaların temelini oluşturan yürütülmekte olan program parçasına proses denmektedir. Program, çalıştırılmadığı sürece pasif bir varlık iken; proses programın tersine aktif bir varlıktır.

    * İhtiyaç duyduğumuz tüm programlar, ana belleğin kısıtlı bir kapasiteye sahip olması nedeniyle diskten ana belleğe getirilememektedir. CPU'da çalıştırılacak olan program parçaları ana belleğe getirilmektedir. Ana bellekte yer alan farklı uygulamalara ait prosesler CPU planlama algoritmasına göre seçilmekte ve CPU tarafından işletilmektedir.

* CPU planlayıcısının amaçlarını yerine getirmek için yaygın olarak kullandığı algoritmalar; "ilk gelen ilk hizmeti alır (FCFS)", "Round Robin", "önce en kısa iş (Shortest Job First- SJF)" ve "öncelik temelli planlama (Priority based scheduling-PbS)" şeklinde sıralanabilir.

    * Bazı prosesler birbirleriyle ilişki içerisindedir. Prosesler birbirlerinin çalışmasını engellemeyecek şekilde uygun bir sırada seçilmeli ve çalıştırılmalıdır. Prosesler arasındaki bu dengenin korunması ve prosesler arasındaki iletişimin sağlıklı bir şekilde devam edebilmesi için proses senkronizasyonunun sağlanması gerekmektedir. Tüm bu ihtiyaçlar için yazılımsal, donanımsal çözümler geliştirilmiştir. Geliştirilen çözümler; hatalı sonuçların oluşmamasını, kilitlenmeyi önlemeyi, proseslerin kilitlenmeden kaçınabilmesini, kilitlenme meydana geldiğinde kilitlenmeyi tespit edip kilitlenmeden kurtulabilmeyi, prosesler arasındaki iletişimin hatasız ve kesintisiz olabilmesini hedeflemektedir.

* Prosesler arasındaki iletişim için kullanılan mekanizmalardan bazıları; sinyaller, soketler, boru hatları (pipe yapısı), paylaşımlı bellek, paylaşımlı dosyalar, mesaj geçişleri, uzaktan yöntem çağırma (RPC) olarak sıralanabilir.

    * Prosesler arasındaki iletişimin neden olabileceği bazı problemler vardır. Bu problemler; karmaşıklık, kilitlenme problemleri, veride meydana gelen uyuşmazlıklar, senkronizasyon problemlerinden oluşmaktadır.

* Prosesler, çalışabilmek için kaynaklara gereksinim duymaktadırlar. Proseslerin hiçbirinin çalışamaz duruma geldiğinde kilitlenme meydana gelmiş demektir. Kilitlenmenin oluşabilmesi için dört adet gerekli koşul bulunmakta ve bu koşulların aynı anda gerçekleşmesi gerekmektedir. Bu koşullar; "karşılıklı dışlama", "tut ve bekle", "kesme durumu yok" ve "döngüsel bekleme" şeklindedir.

    * Kilitlenme durumunun engellenmesi ve kilitlenme durumunda kullanılan bazı yaklaşımlar mevcuttur. Kilitlenmeyi önlemek, kilitlenmeden kaçınmak, kilitlenme durumunu tespit etmek ve bu durumdan kurtulmak bu yaklaşımlardandır.

* Kilitlenme meydana geldiğinde bu durumdan kurtulabilmek için; kilitlenmeye neden olan prosesin çalışması yarıda kesilebilir ve bu prosese tahsis edilen kaynaklar elinden alınabilir; kilitlenmenin yaşandığı tüm proseslerin çalışması durdurulabilir, kilitlenmenin yaşandığı proseslerin çalışması teker teker durdurulabilir veya kilitlenmenin yaşandığı prosesler belirli bir noktaya kadar geri çekilip tekrardan proseslerin çalışması başlatılabilir.

Kaynakça

(n.d.).

(2021). Retrieved from Debian Manpages: https://manpages.debian.org/

Aksan, C. (2021). *Paket Yöneticisi Nedir? Neden İhtiyaç Duyulur?* Retrieved from ceaksan.com: https://ceaksan.com/tr/paket-yoneticisi

Aladağ, M. (2015). *Windows 10 Yenilikleri – PowerShell 5 ile Paket Yönetimi OneGet*. Retrieved from cozumpark.com: https://www.cozumpark.com/windows-10-yenilikleri-powershell-5-ile-paket-yonetimi-oneget/

Alkar, A. (2015). Embedded system basics and application. Ankara.

Çobanoğlu, B. (2018). *Herkes için Python*. Pusula.

Doğu Akdeniz Üniversitesi, B. B. (2021). İşletim Sistemleri-Bellek Yönetimi.

Doğu Akdeniz Üniversitesi, B. V. (2021). İşletim sistemleri-Kilitlenmeler. Siirt.

Gülbağ, A. (2017). İşletim Sistemlerine Giriş.

javaTpoint. (2021). *OS*. Retrieved from javaTpoint: https://www.javatpoint.com/os-attributes-of-a-process

Kaya, A. (2009). Symbian İşletim Sistemi. *Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri*. Şanlıurfa.

Özdemir, H. (2018). *Linux Paket Yöneticileri*. Retrieved from pythontr.com: https://www.pythontr.com/makale/linux-paket-yoneticileri-632

Saatçi, A. (2002). Bilgisayar işletim sistemleri. Ankara: Hacettepe Üniversitesi. Retrieved from http://hilmi.trakya.edu.tr/ders_notlari/os/isleti_sistemleri.pdf

Samet, R. (2018). Bilgisayar Sistemleri. (A. Ü. Enstitüsü, Ed.)

Shotts, W. (2013). *The Linux Command Line- Second Internet Edition*. No Starch Press.

Silberschatz, A., Gagne, G., & Galvin, P. (n.d.). In *Operating System Concepts*. John Wiley & Sons, Inc. Retrieved 2021

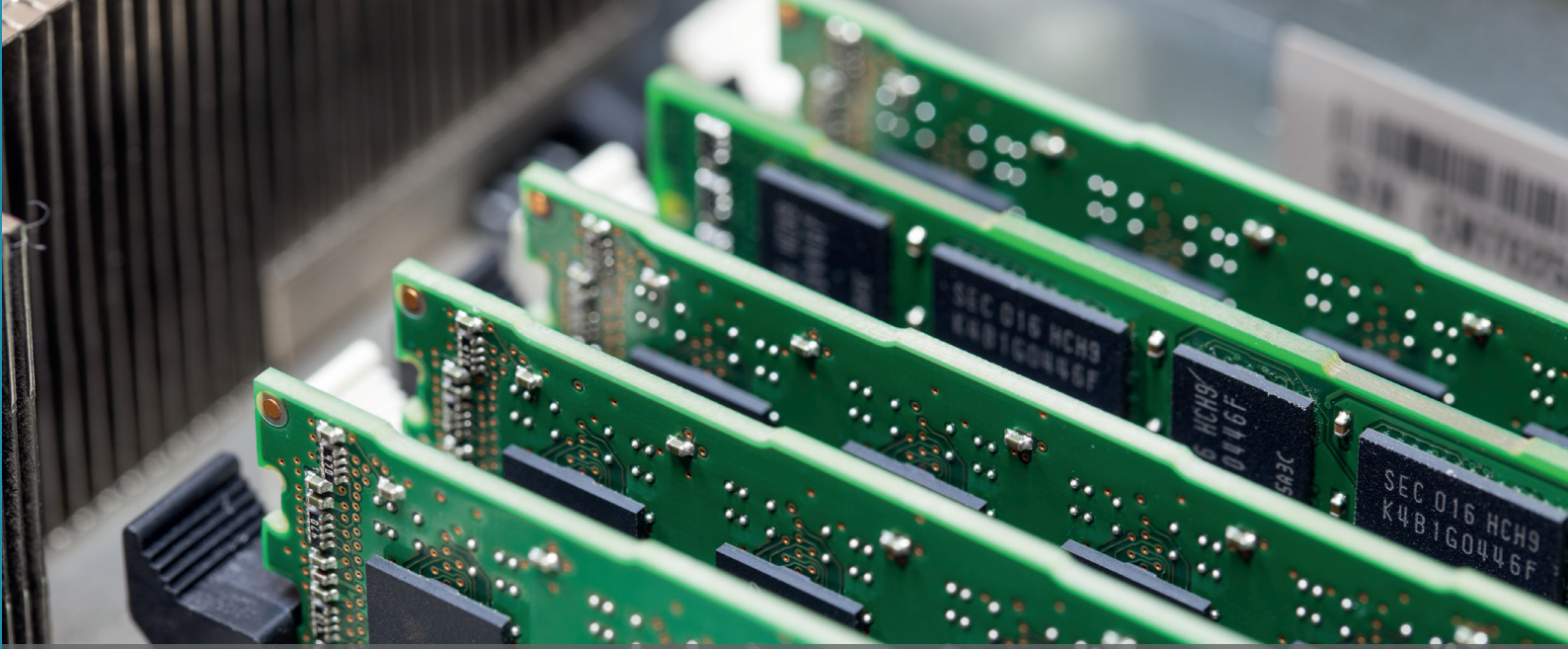
Taşcı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Taşçı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Türkoğlu, İ. (n.d.). İşletim Sistemleri (Ders Notları). *Fırat Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Bölümü*. Türkiye. Retrieved 2021

Wikipedia. (2021). *Paket yönetim sistemi*. Retrieved from Wikipedia.org: https://tr.wikipedia.org/wiki/Paket_y%C3%B6netim_sistemi

Yıldırım, S. (n.d.). Bilgi Teknolojilerine Giriş3. Atatürk Üniversitesi. Retrieved 2021



3. BELLEK YÖNETİMİ

Veri tabanı: Hafızanız çöktüğünde kaybettiğiniz bilgiler.

Dave Barry

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Bellek organizasyonu yöntemlerini öğrenebilir.
- İşletim sisteminin bellek yönetimi için gerçekleştirdiği aktiviteleri öğrenebilir.
- Bellek yönetim tekniklerini öğrenebilir.
- Bellek yönetiminde kullanılan algoritmaları tanıyabilir.
- Bellek problemlerini tanıyabilir ve çözüm üretebilir.

Başlamadan Önce

Bir önceki bölümde prosesler ve işletim sisteminin proses yönetimi için gerçekleştirdiği aktiviteler, proseslerin birbirleriyle uyumlu bir şekilde çalışabilmesi için gerekli koşullar, proseslerin birbirleriyle iletişim kurmasını sağlayan mekanizmalar, proseslerin oluşturabileceği sorunlar ve bu sorunları önlemek için kullanılan yaklaşımlar detaylıca açıklanmıştı. Bu bölümde bellek birimlerinin prosesler için nasıl düzenlendiği, işletim sisteminin güvenilir bir bellek yönetimi için gerçekleştirdiği aktiviteler anlatılmaktadır.

Birlikte Düşünelim

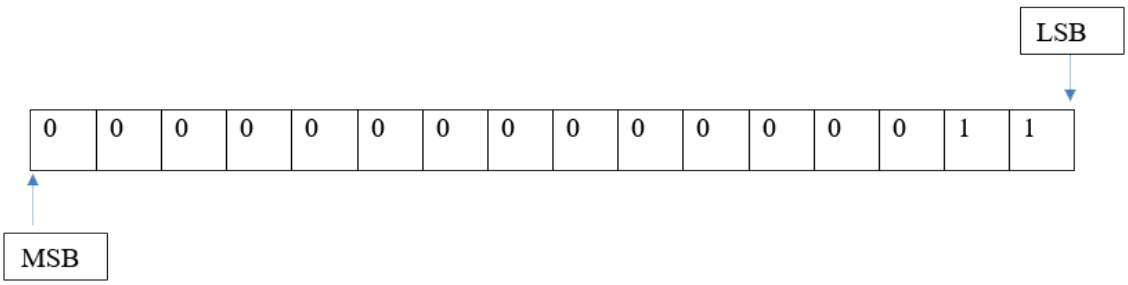
1. İşletim sistemi bellek yönetimi için hangi aktiviteleri gerçekleştirmektedir?
2. Bellek kapasitesinden daha büyük boyutlardaki prosesler çalıştırılabilir mi? Nasıl?
3. Bellek fiziksel ve mantıksal nasıl organize edilmektedir?

3.1. Bellek

Bellek, bilgisayar verilerinin saklandığı birimdir. Birincil ve ikincil bellek olarak 2 türü mevcuttur. Birincil bellek, ana bellek olarak isimlendirilir ve geçici olarak verilerin tutulmasını sağlar. İkincil bellek, disk olarak isimlendirilir ve elektrik kesintisi ile veriler kaybolmaz.

Bilgisayar 1 ve o'ların oluşturduğu ikili sayı sistemini bilir. Veriler öncelikle ikili sayı sistemine çevrilir sonra belleğe yerleştirilir. Örneğin; bilgisayarda int x=3; yazdığımızda bilgisayarda öncelikle bu yazdığımız veri ikili sayı sistemine çevrilir, bellek bloklarına ikili sayı sistemindeki şekliyle yazılır.

32 bit sistemleri düşünelim. "Int" veri tipinin boyutu 2 byte yani 16 bittir. 1 bellek bloğu 1 bit saklar. int x=3; bellekte aşağıdaki gibi saklanır (LSB: düşük öncelikli bit, MSB: yüksek öncelikli bit).



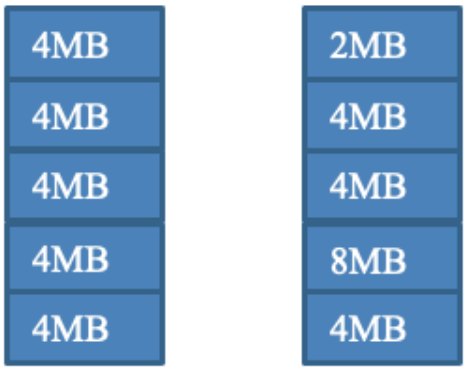
En soldaki bit işaret bitidir. İşaret biti o ise pozitif sayıyı 1 ise negatif sayıyı temsil eder.

3.2. Proseslerin Ana Belleğe Yerleştirilmesi

Prosesleri uygun bir şekilde ana belleğe yerleştirmek, bellek yöneticisinin temel işlevlerinden biridir. Birden fazla prosesin ana belleğe yerleştirilmesi için kullanılan teknikler vardır.

3.2.1. Sabit Bölümleme

Sabit bölümleme tekniğinde bellek eşit veya farklı büyüklerde sabit bölümlere ayrılırlar. Bu bölümler sabittir. Prosesin büyüklüğüne göre dinamik olarak büyüyüp küçülemez. İşletim sistemi her zaman ilk bölümdedir. Diğer bölümler kullanıcıların işlemleri için kullanılır.



Çizim 14. Sabit Bölümleme

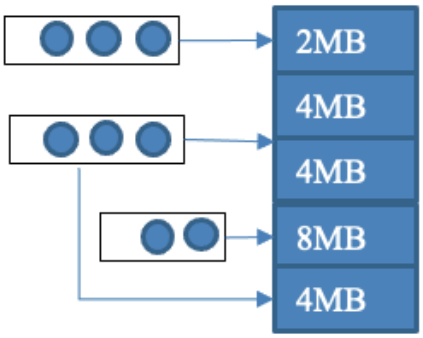
Sabit bölümlemede tekniği kullanıldığında oluşabilecek bazı problemler mevcuttur.

- Program bölümlenmiş alana sığmayabilir.
- Bellek kullanımı verimsizdir. İç parçalanma ve dış parçalanma durumları sebep olur.
- Proseslerin boyutu kısıtlanır. Proses boyutu, en yüksek boyuta sahip bölümün boyutundan büyük olamaz. Büyük olduğu takdirde proses belleğe yüklenemez.
- Aynı anda belleğe yüklenecek proses sayısı azdır.

İç parçalanma: Prosesin boyutu, bölümlenmiş alandan çok küçük olabilir. Proses küçük bir alan kaplasa dahi tahsis edilen bölümün tamamını elde tutar. Bu durum iç parçalanma (internal fragmentation) olarak bilinir. Çizim 14'te 4 MB'lık bölümlenmiş alana 3 MB boyuta sahip bir prosesin yerleştirildiğini düşünelim. Bu durumda 1 MB'lık bölge kullanılamaz durumda olur ve 1 MB boşa gider.

Dış parçalanma: Çeşitli bölümlerin kullanılmayan alanların toplamı, prosesi yüklemek için yeterli alana sahip olsa bile prosesin yüklenmesi için kullanılamaz. Çizim 14'te eşit bölümlendirme yapılmış ana belleğin her bir bölmesine 2 MB'lık bir proses yerleştirildiğini düşünelim. Bu durumda 2MB'dan toplam 10 MB'lık toplam kullanılmayan alan vardır.10 MB'lık kullanılmayan toplam alan olsa bile bu alanların hiçbirini kullanamayız.

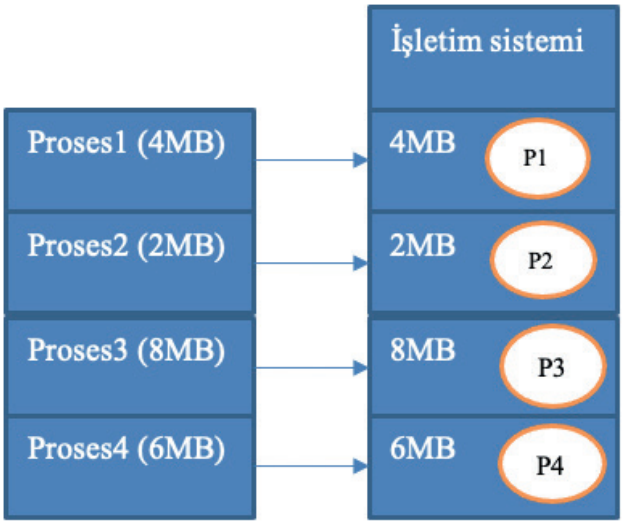
Eşit büyüklükte bölümlemede prosesler belleğe rastgele yerleştirilebilir. Farklı büyüklükteki bölümleme de prosesler belleğe rastgele yerleştirilemezler. Prosesler bölüm boyutlarına göre gruplandırılır ve her boyuta ait kuyruklarda belleğe yerleştirilmek üzere beklerler. Farklı bölümlendirme de kullanılan farklı yöntemler bulunur. En basit yöntem ise sığabileceği en küçük bölüme prosesi yerleştirmektir.



Çizim 15. Farklı Bölümlemede Proseslere Bellek Tahsisi

3.2.2. Dinamik Bölümleme

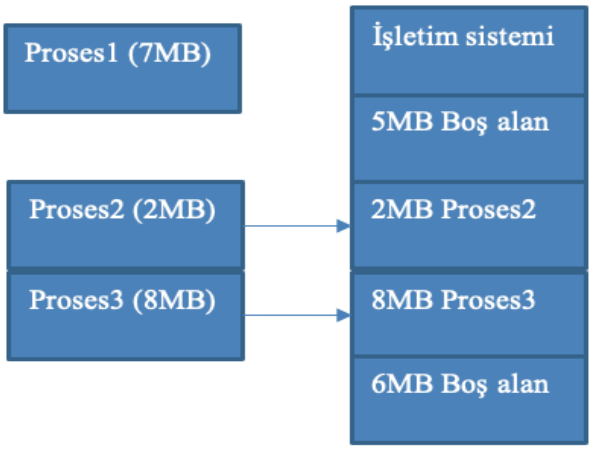
Dinamik bölümlemede proseslerin gereksinim duyduğu alan kadar prosese bellekte yer ayrılır. Dinamik bölümlemede bölümler farklı boyuttadırlar ve farklı boyutlarda çok sayıda bölme vardır. Dinamik bölümleme ile sabit bölümlemede yaşanan problemlerin birçoğunun üstesinden gelinmektedir.



Çizim 16. Dinamik Bölümleme

Dinamik bölümlemede, iç parçalanma problemi oluşmaz. Proseslerin boyutuna göre dinamik olarak bellek bölümlendirilir. Proseslerin boyutu ile ilgili bir kısıtlama söz konusu değildir. Çünkü ana bellek başlangıçta belirli boyutlarda bölümlendirme gerçekleştirmemiştir. Aynı anda belleğe yüklenecek proses sayısı dinamiktir. Statik bölümlendirmeye göre daha fazladır.

Dinamik bölümlemede dış parçalanma problemi statik bölümleme de olduğu kadar kesin ve çok olmasa da bu problemin tamamen çözüme kavuştuğu söylenemez.



Çizim 17. Dinamik Bölümlemede Dış Parçalanma Problemi

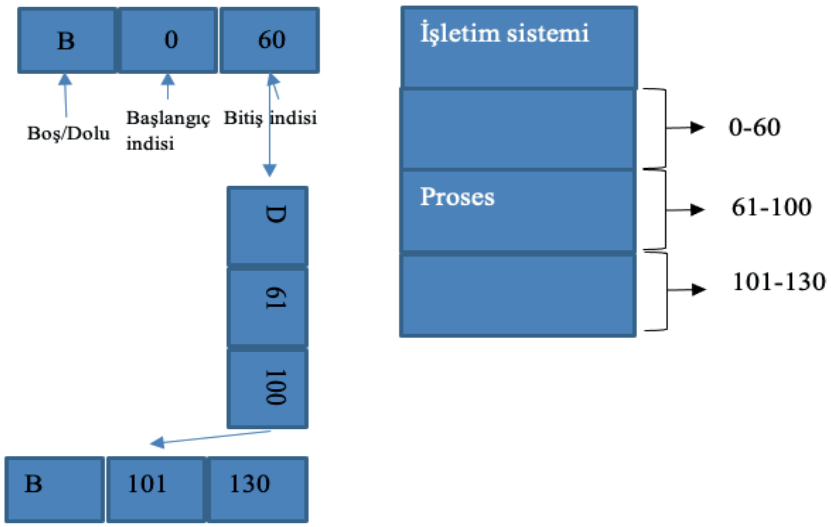
Çizim 17'de görüldüğü gibi Proses2 ve Proses3'e ana bellekte yer tahsis edilmiştir. Ana bellekte toplam 11 MB'lık boş bir alan bulunmaktadır. Proses1'in boyutu 7 MB'dır. 5 MB ve 6 MB'lık bölümlere yerleştirilemez. Bellekte toplam 11 MB alan olmasına rağmen proses1'e bellekten yer tahsis edilemez. 5 MB'lık ve 6 MB'lık boş alanlar arda ardına olsaydı tahsis edilebilirdi. Harici bölümleme problemini çözebilmek için boş alanların arda ardına gelme kuralının değiştirilmesi gerekmektedir.

Belleğin farklı bölümleri prosese tahsis edilerek dış parçalanma problemi çözümlenebilir. Bir başka çözüm yöntemi olarak sıkıştırma işlemi uygulanabilir. Prosesler ana belleğe arda ardına yüklenerek, boş alanların bitişik olması sağlanabilir. Buna **sıkıştırma** denir. Sıkıştırma tekniği ile daha büyük boyutlardaki prosesler ana belleğe yüklenebilir. Sıkıştırma tekniği dış parçalanma problemine çözüm getirir fakat verimsizliğe neden olur. Birleştirme işlemi esnasında CPU boşta kalır.

Dinamik bölümlemede tahsis edilen bölümler ve boş alanların kayıtlarının tutulması gerekir. İşletim sistemi bu görev için bit haritası ve bağlantılı liste isimlerinde 2 veri yapısı kullanır. Bit haritası çok fazla tercih edilen bir veri yapısı değildir. Bağlantılı liste boş ve dolu olan bölümleri izlemek için kullanılan popüler bir veri yapısıdır.

Her bir alan için (dolu veya boş olabilir.) düğümün tanımlandığı bir bağlantılı liste vardır (Bk. Çizim 18). Her düğümün 3 alanı vardır:

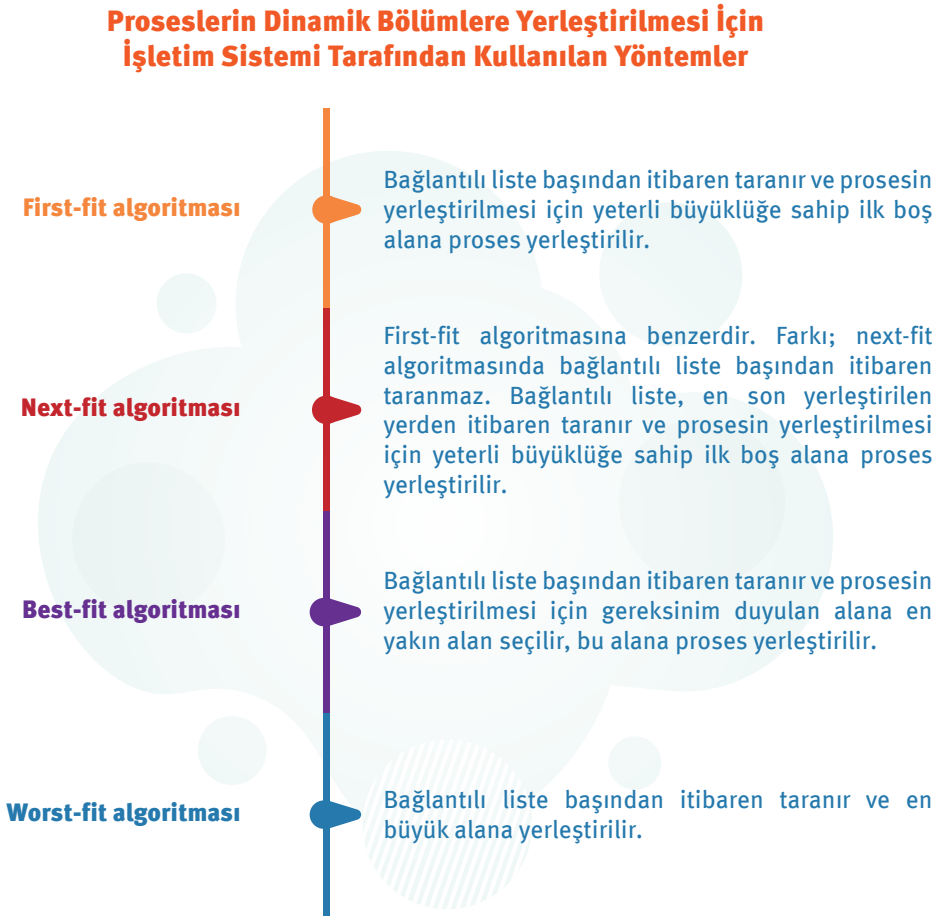
- Düğümün ilk alanı bir bayrak bitidir. Bayrak biti düğümün temsil ettiği bölümün dolu mu yoksa boş mu olduğunu gösterir.
- Düğümün ikinci alanı bölümün başlangıç indisini gösterir.
- Düğümün son alanı bölümün son indisini gösterir.



Çizim 18. Bağlantılı Liste

3.2.3. Bölümleme Algoritmaları

Proseslerin dinamik bölümlere yerleştirilmesi için işletim sistemi tarafından kullanılan birçok yöntem vardır.



Şekil 6. Proseslerin Dinamik Bölümlere Yerleştirilmesi İçin İşletim Sistemi Tarafından Kullanılan Yöntemler

Örnek: Bellekte boş bölgelerin büyükleri sırasıyla 10K, 8K, 4K, 15K, 12K, 20K, 18K'dır. Büyüklükleri 12K, 10K ve 9K olan 3 proses belleğe yüklenmek isteniyor. Aşağıda verilen algoritmaları kullanarak 3 prosesi ana belleğe yerleştirelim.

a. First-fit algoritması
b. Next-fit algoritması
c. Best-fit algoritması
d. Worst-fit algoritması

First-fit algoritması:

12K prosesi 4. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 3K, 12K, 20K, 18K

10K prosesi 1. Boşluğa yerleştirilir. Boş bölmelerin listesi: 8K, 4K, 3K, 12K, 20K, 18K

9K prosesi 4. Boşluğa yerleştirilir. Boş bölmelerin listesi: 8K, 4K, 3K, 3K, 20K, 18K

Next-fit algoritması:

12K prosesi 4. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 3K, 12K, 20K, 18K

10K prosesi 5. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 3K, 2K, 20K, 18K

9K prosesi 6. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 3K, 2K, 11K, 18K

Best-fit algoritması:

12K prosesi 5. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 15K, 20K, 18K

10K prosesi 1. Boşluğa yerleştirilir. Boş bölmelerin listesi: 8K, 4K, 15K, 20K, 18K

9K prosesi 5. Boşluğa yerleştirilir. Boş bölmelerin listesi: 8K, 4K, 6K, 20K, 18K

Worst-fit algoritması:

12K prosesi 5. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 15K, 12K, 8K, 18K

10K prosesi 6. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 15K, 12K, 8K, 8K

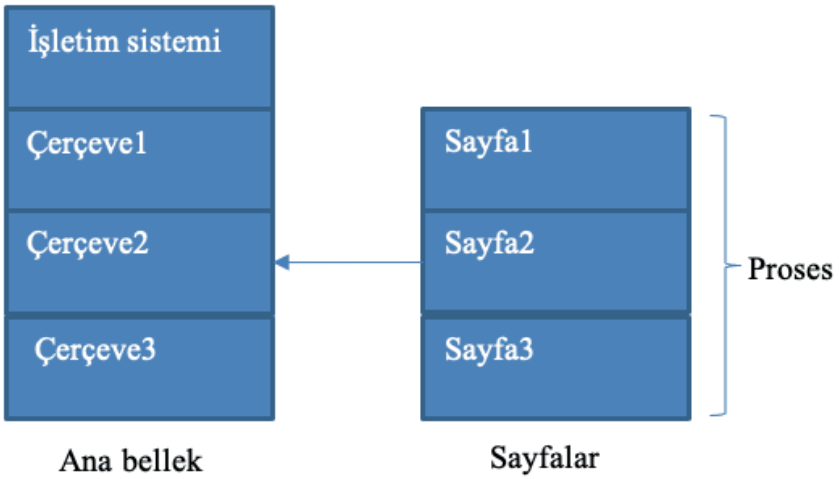
9K prosesi 1. Boşluğa yerleştirilir. Boş bölmelerin listesi: 10K, 8K, 4K, 6K, 12K, 8K, 8K

3.2.4. Sayfalama

Belleğin daha verimli kullanılabilmesi ve bellek yönetiminin daha etkin bir şekilde yapılabilmesi için sayfalama tekniği kullanılmaktadır. Sayfalama tekniği ana bellekte yer alan arda ardına bulunmayan boş bellek alanlarından faydalanmayı hedeflenir. Proses sayfa (page) şeklinde küçük ve eşit boyutlarda bölümlendirilir. Ana bellek, sayfa ile aynı boyutta çerçeve (frame) ismini verdiğimiz bölümlere ayrılır (bk. Çizim 19. Sayfalama tekniği.). Prosesin bir sayfası ana bellekteki bir çerçeve içerisinde depolanır. Aynı prosese ait sayfalar ana belleğin farklı yerlerinde bulunan çerçevelere kaydedilebilirler. Sayfa ve çerçeve boyutu işletim sistemi tarafından belirlenir.

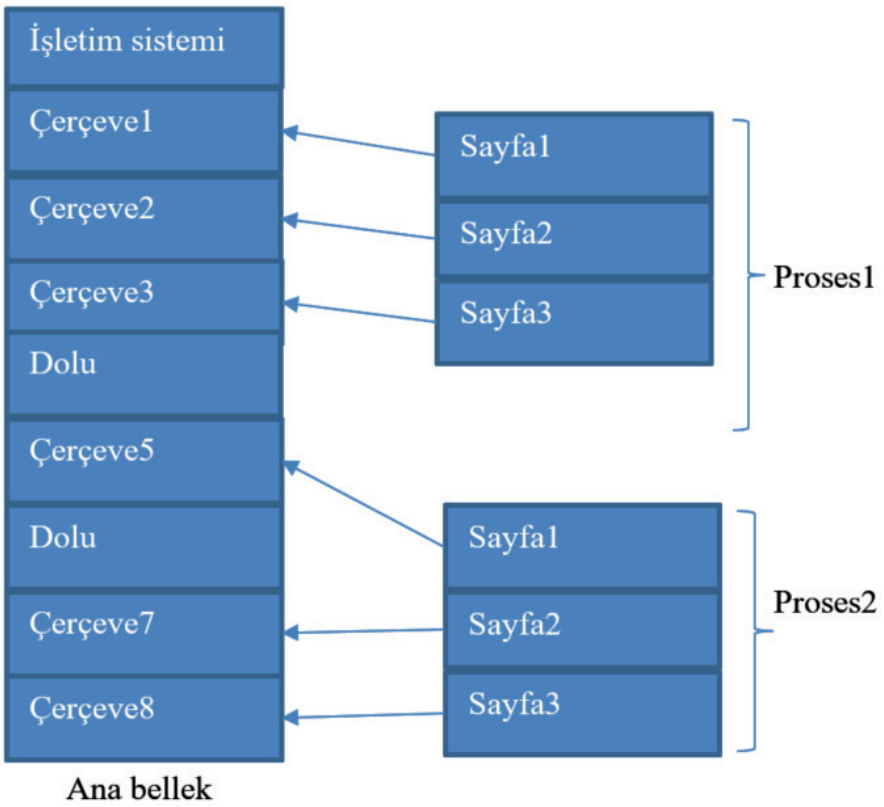
Okuma Önerisi

Bellek Yönetimi konulu makaleyi çevrim içi olarak aşağıdaki bağlantıdan okuyabilirsiniz



Çizim 19. Sayfalama Tekniği

Örnek: Ana belleğin boyutu: 8 KB ve çerçeve boyutu 1 KB olsun. Çerçeve sayısı: 8 olur. 3 KB boyuta sahip 2 prosesin her biri 1 KB'lık 3 sayfa içerir. Çizim 20'de gösterilen bir bellek düzeni içerisinde Proses1 ve Proses2 ana belleğe yerleştirilir.



Çizim 20. Sayfalama Tekniği Uygulaması

CPU'da her bir sayfa için mantıksal bir adres üretilir. Her bir sayfanın bulunduğu çerçevenin gerçek adresi fiziksel adrestir. Mantıksal adresi kullanarak CPU bir sayfaya eriştiğinde işletim sistemi bu sayfaya fiziksel olarak erişebilmek için fiziksel adresi kullanır. Her prosese ait sayfa tablosu bulunur. Sayfa tabloları ana bellekte saklanır ve ana bellekte saklandığı adres sayfa tablosu temel kaydedicisindedir. Sayfa tablolarında sayfa numarası ve bu sayfaların yerleştirildiği çerçeveler ile ilgili bilgiler bulunur.

Mantıksal adresin 2 bölümü vardır: Sayfa numarası ve offset.

Örnek: Mantıksal adresin 12 bit ve sayfa boyutunun 1 KB olduğunu düşünelim. Program başına göre bağıl adresi 1453 olan bellek erişiminin mantıksal adres eş değerini bulalım.

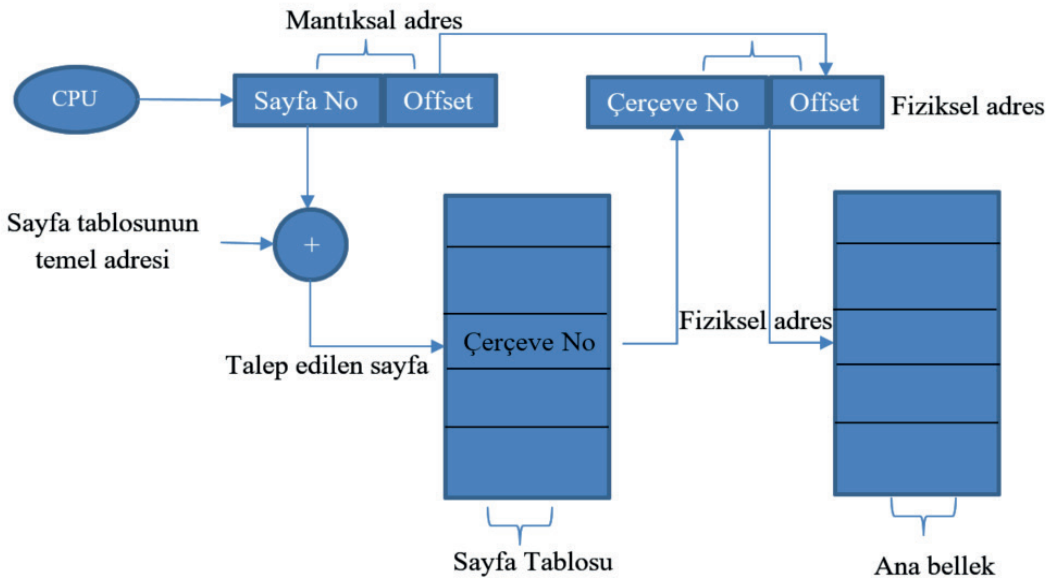
12 bit ile 4 KB'lık bir alan adreslenebilir. 1 KB sayfa boyutundan ana bellekte 4 çerçevede vardır. 4 sayfa 2 bit ile temsil edilir. 12-2=10 bit offset değeridir.

$(1453)_{10} = (10110101101)_2$ 'dir.

Mantıksal adres: (010110101101). En anlamlı 2 bit sayfa numarasıdır. Sayfa no=2; 2. Sayfada ve offset değeri: 429 olan bellek erişimini temsil eder.

3.2.5. Adres Dönüşümü

Prosese ait sayfa tablosu kullanılarak mantıksal adres fiziksel adrese dönüştürülmelidir. Dönüştürülme adımları Çizim 21'de gösterilmektedir:



Çizim 21. Adres Dönüşümü

3.2.6. Sanal Bellek

Sanal bellek ana belleğin boyutunun daha büyük olduğu izlenimi veren aslında bir depolama şeklidir. Sanal bellekte disklerden faydalanılır. Kullanıcı uygulamaları diske yüklenir. İşletim sistemi çalıştırılması gereken uygulamaların çalışması gereken bölümlerini diskten ana belleğe getirir. Ana bellek içerisinde çalıştırılması gereken farklı uygulamaların çalıştırılması gereken bölümleri yer alır. Bilgisayar sistemi birden çok görevi yapabilir hale gelir. Ana belleğin kapasitesinin çok üstünde bir depolama işlevi yapılır. Sanal bellek tekniği ile CPU kullanımı artar.

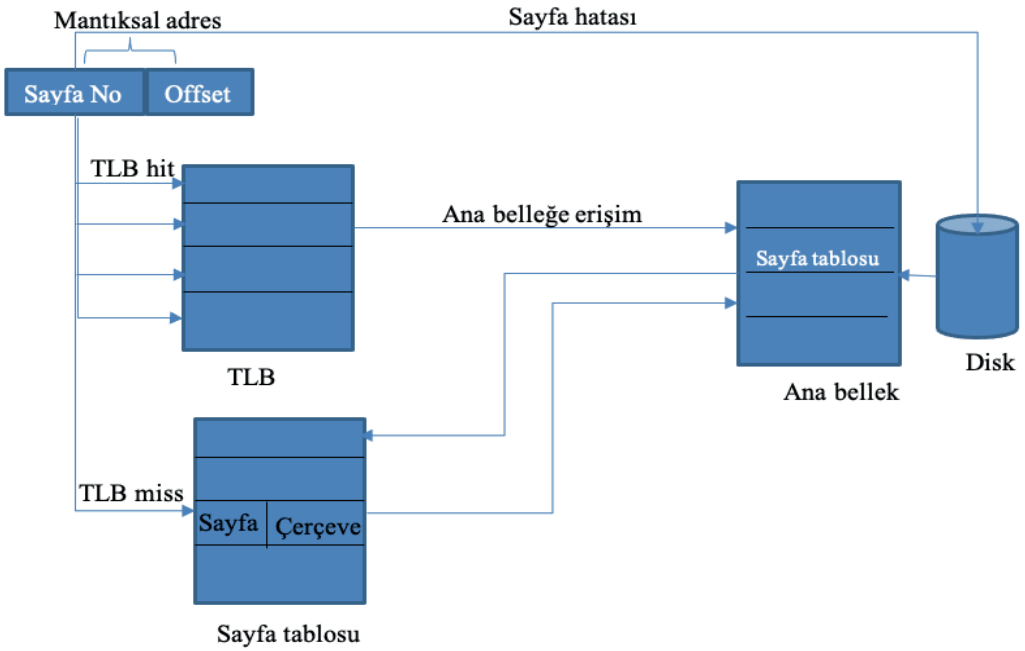
Sanal bellek tekniğinde; proseslere ait olan sayfalar, ana belleğin boş alanlarından (çerçeve sayısından) daha fazla ise ana bellekte yer alan en az kullanılan veri ana bellekten diske taşınır ve prosese ait sayfalar ana belleğe yerleştirilir. Uygulamanın tüm sayfalarının bir anda ana belleğe yüklenmesine gerek yoktur. Gerekli olan sayfalar gerektiği sürece ana belleğe getirilir. Taşıma işlemi sırasında CPU başka bir uygulamanın görevini icra etmekle meşgul olur. Böylece CPU atıl durumda kalmaz.

Sanal bellek tekniği ile;

- Çoklu programlama derecesi artar.
- Kullanıcılar daha fazla uygulama ve daha büyük boyutlardaki uygulamaları çalıştırabilirler.
- Ana bellek ihtiyacı diskle karşılandığından daha fazla ana bellek satın alınmasına ihtiyaç yoktur.

3.2.7. Translation Look Aside Buffer (TLB) Kullanımı

Sayfa tablosu ana bellekte saklanmaktadır. Fiziksel adresin hesaplanması için sayfa tablosuna tekrar tekrar erişim sağlanır. TLB bir önbellektir. CPU'nun TLB'ye erişmesi için geçen süre, ana belleğe erişmek için geçen süreden daha kısadır. TLB, ana bellekten daha küçük boyutta, kaydedicilerden daha büyük boyuttadır. TLB, sayfa tablosunun en çok kullanılan sayfalarıyla ile ilgili bilgileri tutar. TLB kullanılarak sanal adresin fiziksel adrese dönüşümü Çizim 22'de gösterilmektedir.



Çizim 22. Sayfalamada TLB Kullanımı

CPU tarafından üretilen sanal adresin fiziksel adrese dönüştürülmesi gerekir. Sanal adres sayfa numarası ve offset değerlerine sahiptir. Öncelikle sayfa numarası TLB üzerinden aranır. Eğer TLB üzerinde bulunursa TLB hit olur ve çerçeve numarasının yanına offset konularak fiziksel adres elde edilir. Eğer TLB üzerinde bulunamadıysa TLB miss olur ve sayfa tablosunda arama yapılır. Sayfa tablosunda bulunursa sayfa tablosundaki çerçeve bilgisinin yanına offset bilgisini yazılarak fiziksel adres bulunur. TLB yeni bilgi ile güncellenir. Şayet sayfa tablosunda da bulunamaz ise o zaman sayfa hatası verir. Sayfa hatası ana bellek üzerinde aradığımız bilginin olmadığını gösterir. Bu bilgi diskte aranır ve ana belleğe getirilir. Sayfa tablosunda ve TLB'de güncellemeler gerçekleştirilir.

3.2.8. Sayfa Değiştirme Algoritmaları

İşletim sisteminin görevlerinden biri bellek yönetimidir. Belleğin güvenli ve verimli bir şekilde kullanılması hedeflenmektedir. Bellek yönetiminde karşılan iç parçalanma, dış parçalanma, büyük boyuttaki uygulamaların çalıştırılamaması gibi problemler sayfalama, sanal bellek teknikleri ile büyük ölçüde çözüme kavuşturulmuştur. Sayfalama tekniğinin kullanımı dış parçalanma problemini çözüme kavuşturur, iç parçalanma problemini de büyük ölçüde kontrol altına alır. Bellek yönetiminde karşılaşılan bir diğer problem ise bellek boyutunun yetersiz kalmasıdır. Bu problem sanal bellek ile çözümlenmiştir. Ana belleğe sığmayan bilgiler diske yazılır ve diskte yer alan bilgiler gerektiğinde de ana belleğe yüklenir. Hangi bilgilerin ana bellekte, hangi bilgilerin diskte olacağına ve ne

zaman disk ile ana bellek arasında bilgi değişiminin olacağına **sayfa değiştirme algoritmaları** belirler.

Çeşitli sayfa değiştirme algoritmaları vardır.

1. Optimal sayfa değiştirme algoritması: Optimal sayfa değiştirme algoritmasında ana bellekte bulunan ve uzun süre ihtiyaç duyulmayacak olan sayfa diske taşır. Pratikte uygulanabilir değildir fakat diğer algoritmaların karşılaştırılmasında kullanılabilir.
2. En son kullanılan sayfa (LRU) değiştirme algoritması: En uzun süredir kullanılmamış sayfa diske taşınır. Optimal sayfa değiştirme algoritmasının tam tersi geleceğe değil geçmişe bakar. Sayfalara son erişim zamanı bilgisi tutulur. Bu bilgi sisteme ek yük getirir (Gülbağ, 2017).
3. İlk gelen ilk çıkar (FIFO) değiştirme algoritması: Sayfalar arasında en erken kullanılmış, ilk gelen sayfa diske taşınır. Uygulanması kolaydır fakat problemlere neden olabilir. Sürekli kullanımda olan sayfalar olabilir (Gülbağ, 2017).

Anahtar Kavram
Optimal: En uygun.

Örnek:

İşletim sisteminden sırasıyla "3,4,3,5,2,3,1,2,5,1" şeklinde çerçeveler talep edildiğini varsayalım. Ana belleğe sadece 3 çerçeve anlık olarak sığabiliyor. Optimal, FIFO ve LRU sayfa değiştirme algoritmalarını kullanarak sayfa yer değiştirme dizinlerini oluşturalım.

Optimal:

İstekler	3	4								
		4								
Miss/Hit	Miss	Miss	Hit	Miss	Miss	Hit	Miss	Hit	Hit	Hit

Optimal algoritması kullanıldığında gerçekleşen sayfa hatası sayısı: 5

FIFO:

									5	
									3	
							Miss	Hit	Miss	Hit

FIFO algoritması kullanıldığında gerçekleşen sayfa hatası sayısı: 7

LRU:

İstekler	3	4	3	5	2	3	1	2	5	1
Çerçeve3				5	5	5	1	1	1	1
Çerçeve2		4	4	4	2	2	2	2	2	2
Çerçeve1	3	3	3	3	3	3	3	3	5	5
Miss/Hit	Miss	Miss	Hit	Miss	Miss	Hit	Miss	Hit	Miss	Hit

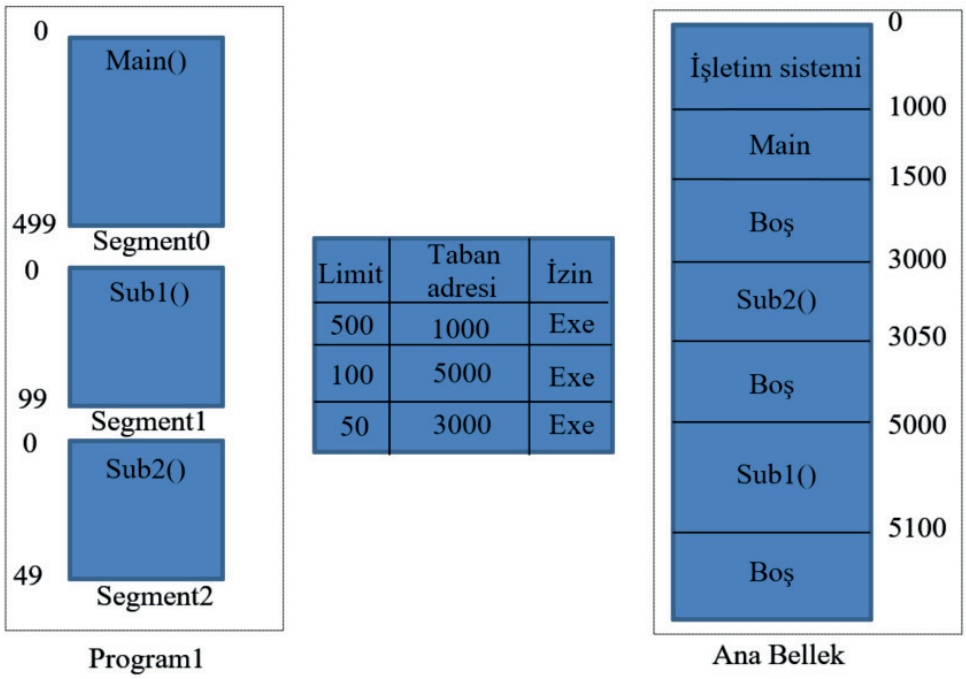
LRU algoritması kullanıldığında gerçekleşen sayfa hatası sayısı: 6

LRU ve optimal sayfa değiştirme algoritmalarında çerçeve sayısının arttırılmasının sayfa hatası sayısını azaltacağı görülmüştür. Balady, FIFO sayfa değiştirme algoritmasında ise çerçeve sayısının artmasının sayfa hatası sayısını arttıracağını bulmuştur. FIFO algoritmasının gösterdiği tuhaf bir davranıştır. Bu anomali Belady's Anomaly olarak isimlendirilmiştir (javaTpoint, 2021).

3.2.9. Segmentasyon

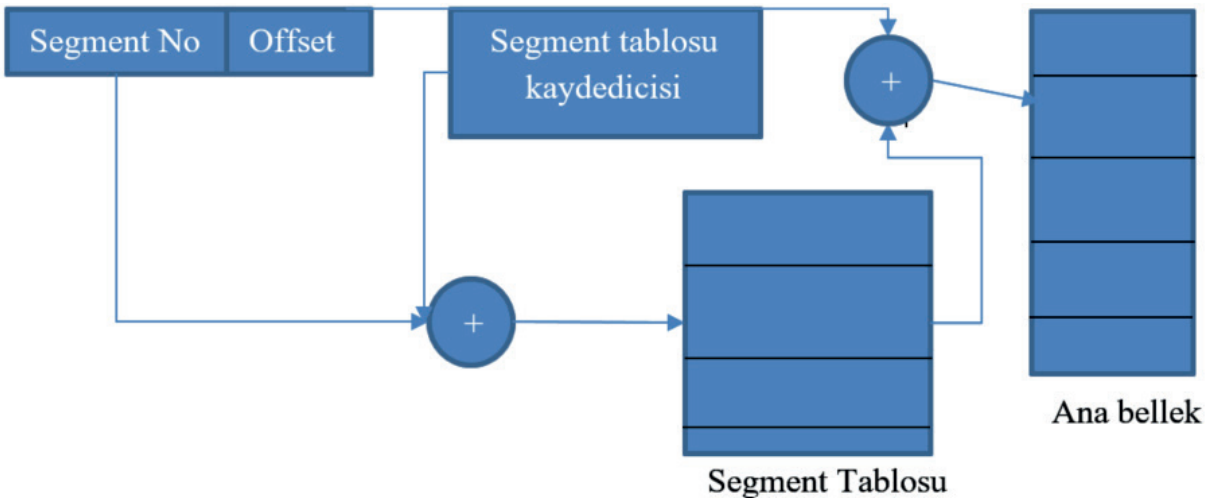
Segmentasyon, ana belleğin farklı boyutlarda bölümlere ayrıldığı bir bellek yönetim tekniğidir. Ana bellekteki her bir bölüm segment olarak isimlendirilir ve proseslere segmentler tahsis edilirler.

Sayfalama işleminde prosesler eşit boyutlarda bölümlere ayrılmaktadır. Sayfalama tekniği aynı sayfaya yüklenmesi gereken proses bölümlerinin var olabileceği gerçeğini göz ardı etmektedir. Aynı görevi farklı sayfalara bölebilir ve bu sayfalar ana belleğe yüklenebilir veya yüklenemeyebilir. Verimin düşmesine neden olunur. Örneğin main metodu tek bir sayfa içerisinde olmayabilir. Segmentasyon ile bu problemin üstesinden gelinmektedir. Segmentasyonda her bir segment aynı türdeki fonksiyonları içerir (bk. Çizim 23). Örneğin main metodunu bir segment içerirken kütüphane fonksiyonlarını bir başka segment içerir.



Çizim 23. Segmentasyon Tekniği

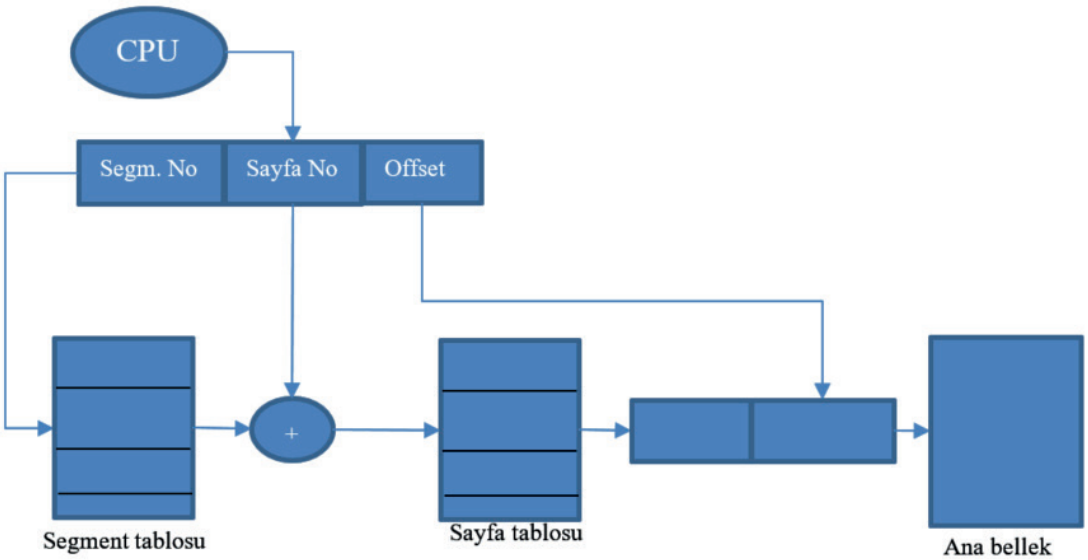
Segmentasyonda mantıksal adres 2 içeriğe sahiptir: Segment numarası ve offset bilgisi. CPU tarafından üretilen mantıksal adresin segmentasyon tekniğinde fiziksel adrese dönüşüm adımları Çizim 24'te gösterilmektedir. Segment tablosu kaydedicisi segment tablosunun başlangıç adresini saklar. Segment numarası ve segment tablosunun başlangıç adresi toplanarak ilgili segmentin ana bellekte bulunduğu adres bulunur. İlgili segmentin başlangıç adresi tablodan alınır. Başlangıç adresine offset değeri eklenerek fiziksel adres elde edilmiş olur.



Çizim 24. Segmentasyonda Adres Dönüşümü

İşletim sistemleri segmentasyonu veya sayfalamayı tek başına kullanmayı genellikle tercih etmezler. Segmentasyon ile sayfalama beraber kullanıldığında her iki yöntemin avantajlarından faydalanılır. Segment ile sayfalamanın birlikte kullanıldığı yöntemde ana bellek değişken boyutlarda segmentlere ayrılır, segmentler aynı boyutlarda sayfalara bölünür. Sayfaların boyutu segmentlerin boyutundan küçüktür. Bu yöntemi kullanan işletim sistemlerinde CPU'nun ürettiği sanal adres segment numarası, sayfa numarası ve offset bilgilerinden oluşur. Segment tablosu sayfa tablolarının adreslerini tutar. Her bir sayfa tablosu da segmentin her bir sayfasını gösterir.

Segmentasyon ile sayfalama tekniklerinin bir arada kullanıldığında sanal adresin fiziksel adrese dönüşümü Çizim 25'te gösterilmektedir:



Çizim 25. Segmentasyon ve Sayfalama Tekniklerinin Bir Arada Kullanımı

Bu Bölümde Ne Öğrendik?

* Bilgisayar sistemlerinde süreçlerin yürütülebilmesi için bilgisayar verilerinin saklandığı bellek birimlerine ihtiyaç vardır. Geçici verilerin tutulduğu bellek birimi birincil bellek (ana bellek) olarak isimlendirilmektedir. Ana bellekte tutulan veriler elektrik kesintisinde kaybolur. Elektrik kesintisinde verilerin kaybolmadığı, ana belleğe göre erişimin daha uzun sürdüğü bellek birimi ikincil bellek (disk) olarak anılır.
    * İşletim sistemi bellek birimlerini kullanarak proseslerin güvenli bir şekilde işletilmesini sağlamaktadır. Proseslere ana bellekte uygun yerler tahsis edilmesi, ana bellek kapasitesi yeterli olmadığı durumlarda en uygun prosesin diske taşınması, proseslerin güvenliğini sağlamak adına bellek birimlerinin korunması, bellek birimlerinin güvenli bir şekilde paylaştırılması, bellek birimlerinin mantıksal ve fiziksel organizasyonunu gerçekleştirmek bilgisayar sisteminin güvenli bir işleyişi için işletim sisteminin gerçekleştirdiği bellek aktiviteleridir.
* Birden fazla prosesi ana belleğe yerleştirmek için sabit bölümleme, dinamik bölümleme, sayfalama, sanal bellek, segmentasyon teknikleri kullanılmaktadır.
    * Prosesler dinamik bölümleme yöntemi ile bölümlenmiş ana bellek bölümlerine FCFS, Next-fit, best-fit, worst-fit gibi algoritmalar kullanılarak yerleştirilir.
* Dinamik bölümlemede boş kalan bellek bölümlerin arda ardına gelmemesi durumunda bellek bölümlerinin çöpe gitmesi problemini çözümlemek için sıkıştırma yöntemi kullanılmaktadır. Sıkıştırma yöntemi boş alanları bir araya toplamaya çalışmakta ve bu verimsizliğe neden olmaktadır. Sayfalama tekniği kullanılarak bellekte farklı bölgelerde bulunan boş bellek bölümlerine proses sayfalarının yerleştirilmesi sağlanmaktadır. Proses sayfaları eşit boyuttadır. Aynı sayfada bulunması gereken proses bölümlerinin farklı sayfalara ayrılması muhtemeldir.
    * Segmentasyon yöntemi ile bellek bölümleri farklı boyutlarda dinamik olarak bölümlenebilmektedir. Segmentasyon yönteminde her bir segment ilgili fonksiyonları barındırmaktadır.
* Segmentasyon ve sayfalama tekniklerinin bir arada kullanımı her iki yöntemin avantajlarını bir araya getirmektedir.
    * Sanal bellek ana belleğin boyutunu daha büyük olduğu izlenimi veren bir yöntemdir.
* Sanal bellekte; ana bellek ve disk uyumlu bir şekilde beraber kullanılmaktadır. Ana belleğe sığmayacak prosesler disklere konmakta ve ihtiyaç halinde ana belleğe getirilmektedir. Ana bellek kapasitesi dolduğunda ise prosesler diske taşınmaktadır. İşletim sistemi gerçekleştirdiği aktivitelerle güvenli bir bellek yönetimi hedeflemektedir.

Kaynakça

(tarih yok).

(2021). Debian Manpages: https://manpages.debian.org/ adresinden alındı

Aksan, C. (2021). *Paket Yöneticisi Nedir? Neden İhtiyaç Duyulur?* ceaksan.com: https://ceaksan.com/tr/paket-yoneticisi adresinden alındı

Aladağ, M. (2015). *Windows 10 Yenilikleri – PowerShell 5 ile Paket Yönetimi OneGet*. cozumpark.com: https://www.cozumpark.com/windows-10-yenilikleri-powershell-5-ile-paket-yonetimi-oneget/ adresinden alındı

Alkar, A. (2015). Embedded system basics and application. Ankara.

Çobanoğlu, B. (2018). *Herkes için Python*. Pusula.

Doğu Akdeniz Üniversitesi, B. B. (2021). İşletim Sistemleri-Bellek Yönetimi.

Doğu Akdeniz Üniversitesi, B. V. (2021). İşletim sistemleri-Kilitlenmeler. Siirt.

Gülbağ, A. (2017). İşletim Sistemlerine Giriş.

javaTpoint. (2021). *OS*. javaTpoint: https://www.javatpoint.com/os-attributes-of-a-process adresinden alındı

Kaya, A. (2009). Symbian İşletim Sistemi. *Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri*. Şanlıurfa.

Özdemir, H. (2018). *Linux Paket Yöneticileri*. pythontr.com: https://www.pythontr.com/makale/linux-paket-yoneticileri-632 adresinden alındı

Saatçi, A. (2002). Bilgisayar işletim sistemleri. Ankara: Hacettepe Üniversitesi. http://hilmi.trakya.edu.tr/ders_notlari/os/isleti_sistemleri.pdf adresinden alındı

Samet, R. (2018). Bilgisayar Sistemleri. (A. Ü. Enstitüsü, Dü.)

Shotts, W. (2013). *The Linux Command Line- Second Internet Edition*. No Starch Press.

Silberschatz, A., Gagne, G., & Galvin, P. (tarih yok). *Operating System Concepts*. içinde John Wiley & Sons, Inc. 2021 tarihinde alındı

Taşcı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Taşçı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Türkoğlu, İ. (tarih yok). İşletim Sistemleri (Ders Notları). *Fırat Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Bölümü*. Türkiye. 2021 tarihinde alındı

Wikipedia. (2021). *Paket yönetim sistemi*. Wikipedia.org: https://tr.wikipedia.org/wiki/Paket_y%C3%B6netim_sistemi adresinden alındı

Yıldırım, S. (tarih yok). Bilgi Teknolojilerine Giriş3. Atatürk Üniversitesi. 2021 tarihinde alındı

4. DOSYA SİSTEMLERİ

Üç kesin gerçek vardır.
Ölüm, vergiler ve veri kaybı.

David Dixon

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Disk yönetim yöntemlerini öğrenebilir.
- Dosya ve Dizin yapıları üzerinde işlemler gerçekleştirebilmeyi öğrenebilir.
- İşletim sistemlerinin kullandığı farklı dizin yapıları hakkında fikir sahibi olabilir.
- Disk problemlerini tanıyabilir ve çözüm üretebilir.
- Disk planlama algoritmalarını tanıyabilir ve parametrelerini hesaplayabilir.

Başlamadan Önce

Bu bölümde işletim sisteminin dosya yönetimi için gerçekleştirdiği aktiviteler; dosya yönetimi, dizin yönetimi ve disk yönetimi başlıkları altında anlatılacaktır. Dosya ve dizin kavramlarına detaylıca yer verilecek ve işletim sistemlerinde yaygın olarak kullanılan farklı dosya erişim yöntemleri, alan tahsis etme yöntemleri, boş alan yönetimi için başvurulan yöntemler, disk planlaması için kullanılan algoritmalar örneklerle birlikte anlatılacaktır.

Birlikte Düşünelim

1. Dosyalara disk üzerinde alan tahsis etme işleminde yapay zeka tekniklerinden faydalanılabilir mi?
2. Disk planlamasında hangi parametreler dikkate alınmalıdır?
3. Disk planlamasında kullanılan parametreler bilgisayar performansına bağlı mıdır?

4.1. Dosya

Dosya benzer özellikler gösteren bir dizi kaydın tutulabilmesini sağlayan bir veri yapısıdır. Fotoğrafların bir arada olduğu fotoğraf dosyası, ses verilerinin olduğu ses dosyaları örnek olarak verilebilir. Benzer özellikler gösteren bir dizi dosya dizin olarak bilinir. Ses dosyaları bir araya gelerek ses dizinini oluştururlar. Farklı seviyelerde dizinler dosya sistemini oluşturur.

Şekil 7. Dosyaların Özellikleri

Dosyalar üzerinde birçok işlem gerçekleştirilebilir.

Şekil 8. Dosyalar Üzerinde Gerçekleştirilen İşlemler

4.2. Dosya Erişim Yöntemleri

Disk üzerinde tutulan dosyalara erişmek için geliştirilen yöntemlerden bazıları aşağıda verilmektedir.

Sıralı erişim: Bu erişim türünde işletim sistemi dosyayı kelime kelime okur. Dosya işaretçisi, dosyanın ilk kelimesinin adresini tutar. Kullanıcı, dosyayı okumak istediğinde dosya işaretçisi kelimeyi verir ve değerini 1 kelimelik arttırır. Bu şekilde kelime kelime dosyanın sonuna kadar işlemler devam eder.

İlk işletim sistemleri sadece sıralı erişime izin vermekteydi. Modern işletim sistemleri sıralı erişimi kullanmakla birlikte doğrudan erişim ve indekslenmiş erişim yöntemlerini de kullanmaktadır.

Doğrudan erişim: Doğrudan erişim yönteminde ulaşılmak istenen veriye dosyanın başından itibaren sırayla gelinmez, doğrudan erişim gerçekleştirilir.

Anahtar Kavram

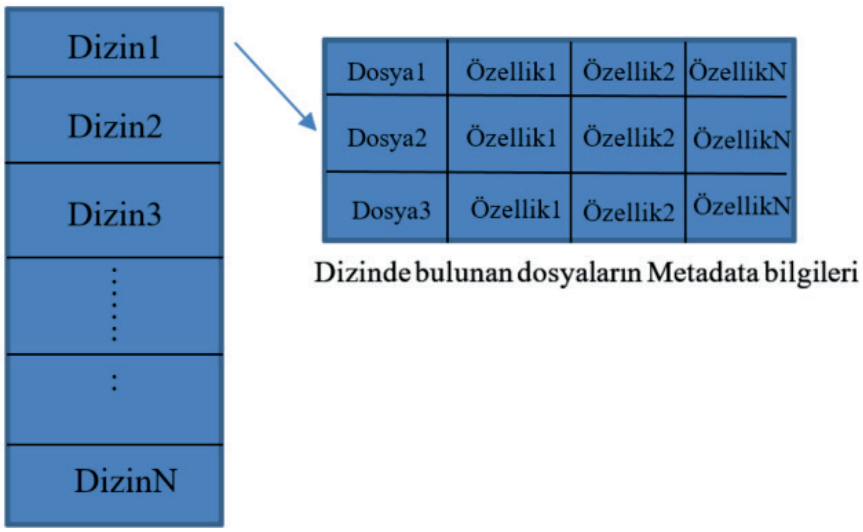
Veri tabanı: Bilgisayar kullanımında çözüme kavuşmak için işlenebilir hale getirilmiş olan bilgi ortamıdır.

Genellikle veri tabanı sistemlerinde gerekli olan ve kullanılan bir dosya erişim yöntemidir.

İndekslenmiş erişim: Bir dosya, dosyalananlardan herhangi birine göre sıralanabiliyorsa, dosyaya indeks atanabilir. İndeks, dosyadaki kaydın adresidir. Çok büyük veri tabanlarında arama yapmak bu yöntemle oldukça kolaydır.

4.3. Dizin

Birbirlerine benzer ve ilgili dosyalar bir araya gelerek dizinleri oluştururlar. Dizinler disklerde saklanırlar. Dizinler sahip oldukları her dosya için dosyanın özelliklerini içeren bir kayda sahiptir (Bk. Çizim 26).



Çizim 26. Dizinlerde Bulunan Dosya Metadata

Şekil 9. Dizin

4.3.1. Dizin Hiyerarşisi

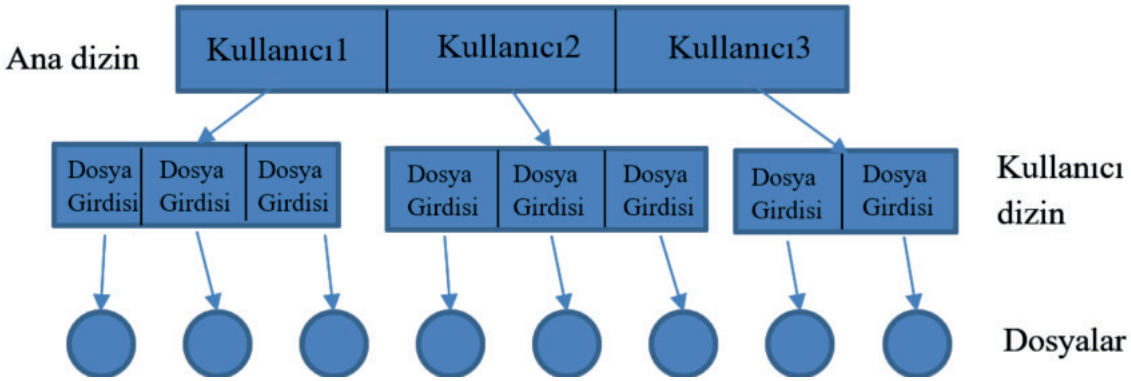
Şekil 10. Dizinlerin Saklanma Şekilleri

Tek Seviyeli Dizinler:

Tek seviyeli dizinde bütün sistemde yalnızca bir dizin vardır. Bu dizin dosya sistemlerinde bulunan bütün dosyaları içerir. Tek seviyeli dizin yapıları basit sistemlerde kullanılmaktadır. En basit dizin yapısıdır. Uygulaması oldukça basittir. Tek bir dizin olduğu için dosyalar aynı isme sahip olamaz. Dosya sistemlerinde çok sayıda dosya varsa eğer tek bir dizin içerisinde bir dosyayı aramak uzun zaman alabilir. Koruma seviyesi oldukça düşüktür. Farklı kullanıcıların dosyaları aynı dizin altında tutulmaktadır. Aynı türdeki dosyaları gruplandırmak mümkün değildir. Farklı türlerdeki dosyalar aynı dizin içerisinde yer alır. İşletim sistemleri dosyaları isimlendirme de bazı kurallar koyar. Karakter sayısı, bazı karakterlerin kullanımında kısıtlamalar mevcuttur. Bu nedenle her dosya için eşsiz bir isimlendirme yapmak saklanabilecek dosya sayısını da etkiler.

İki Seviyeli Dizinler:

İki seviyeli dizin yapısında, her kullanıcı için bir dizin oluşturulur. Ana bir dizin vardır ve her bir kullanıcı için oluşturulan dizinler ana dizine bağlıdır (bk. Çizim 27). Kullanıcılar için oluşturulan dizinlerin ana dizine bağlanması iki seviyeli dizin yapısını oluşturur. Tek seviyeli dizin yapısına göre koruma seviyesi yüksektir. İşletim sistemi bir kullanıcının bir başka kullanıcının dizinine izinsiz girmesine izin vermez.



Çizim 27. İki Seviyeli Dizin Yapısı

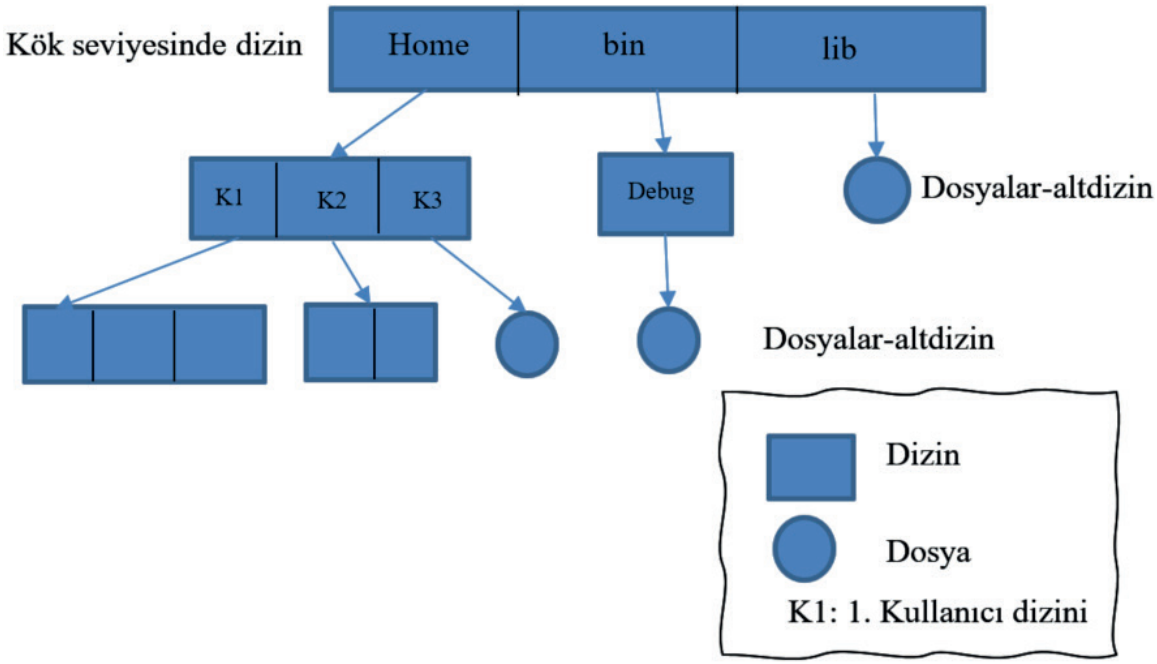
Her kullanıcının dosyaları farklı dizinlerde bulunur. Farklı kullanıcılar aynı isme sahip dosyalar oluşturabilirler. Her dosya ismi "/kullanıcının ismi/ dizin ismi/" şeklinde bir yapı ile başlar. Arama işlemi tek seviyeli dizin yapısına göre daha kolaydır. Hangi kullanıcı için dosya arama gerçekleştiriliyorsa kullanıcının dizininde arama yapılır. Aynı tür dosyalar, bir kullanıcının dizininde gruplandırılamaz.

Ağaç Yapısı Şeklinde Olan Dizinler:

Ağaç yapılı dizin sisteminde, kök seviyesinde dizin vardır. Kök seviyesindeki dizinlere dosya veya başka dizinler bağlanabilir. Bu dizinlere başka dizinler veya dosyalar bağlanabilir. Bu şekilde bir ağaç yapısında dizin yapısı oluşturulur. Ağaç yapısı dizinler ile iki seviyeli dizin yapısında oluşan problemlerin üstesinden gelinebilir. Örneğin: Benzer türdeki dosyalar bir dizin içerisinde gruplandırılabilir.

Her kullanıcıya ait bir dizin yapısı vardır. Bir kullanıcı başka bir kullanıcının dizinine erişemez. Kullanıcıların kök seviyedeki dizinlerde bulunan verileri okuma izni vardır fakat herhangi bir değişiklik gerçekleştiremezler. Sadece sistem yöneticisinin kök seviyesindeki dizinlere tam erişim hakkı vardır. Dizinde dosya arama işlemi daha hızlıdır.

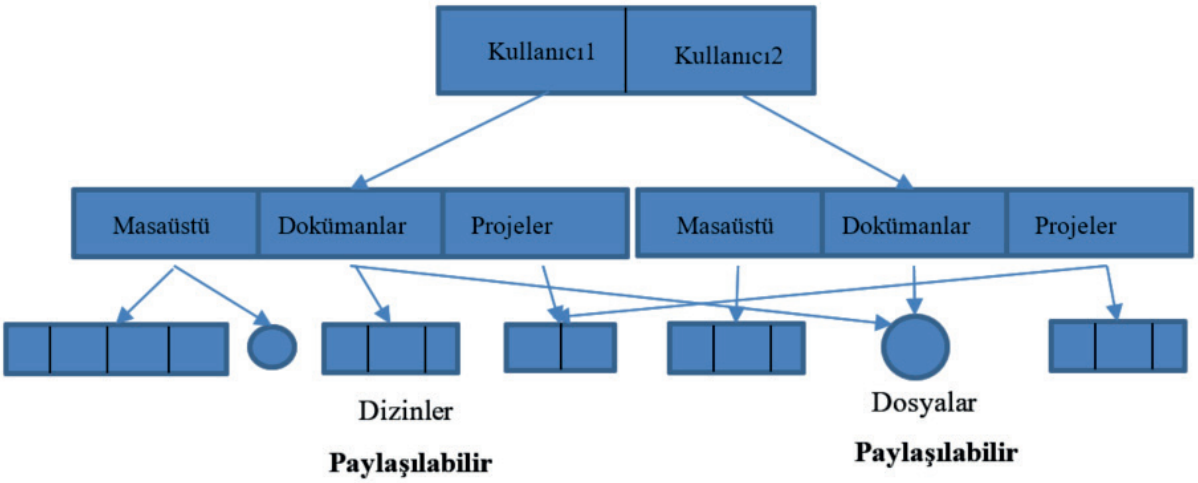
Modern işletim sistemlerinde tercih edilen dizin yapısıdır. Bir dosyaya doğrudan erişim sağlanabileceği gibi dolaylı yoldan bir erişim de sağlanabilir. Doğrudan erişim de sistemin kök dizini referans alınır. Dolaylı erişim yönteminde ise sistemin şu an çalışan dizini referans alınır.



Çizim 28. Ağaç Yapılı Dizin

Döngüsel Olmayan Graf Yapısındaki Dizin:

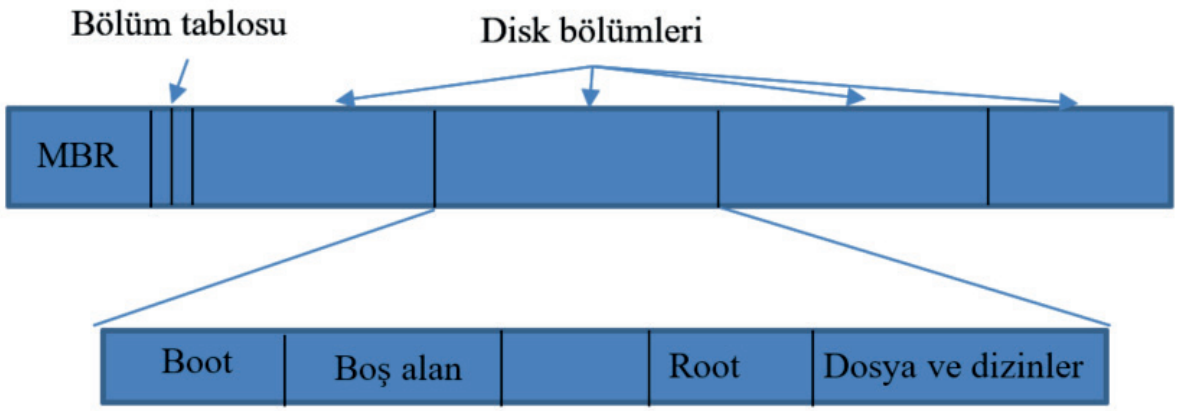
Döngüsel olmayan graf dizinlerinde her kullanıcı için bir dizin tanımlanır. Dizinler, dizin veya dosyalar barındırabilir. Bir dizin içerisindeki dizin bir başka kullanıcı tarafından kullanılabilir. Farklı dizinler aynı dosya veya alt dizine işaret edebilirler (bk. Çizim 29).



Çizim 29. Döngüsel Olmayan Graf Yapısında Dizin

4.4. Dosya Sistemleri

Dosya yönetiminden işletim sisteminin dosya sistemleri bölümü sorumludur. Dosya sistemleri program ve verileri saklamak, bulmak ve erişebilmek için kullanılan mekanizmadır. Dosya sistemleri katmanlı bir yapıda diskler üzerinde saklanırlar. Çizim 30 örnek bir dosya sistemi düzenini göstermektedir:



Çizim 30. Örnek Dosya Sistemi Düzeni (Samet, 2018) (Gülbağ, 2017).

Diskler birden çok bölüme ayrılırlar. Diskin 0. Sektörü MBR (Master boot record) olarak bilinir. MBR, işletim sisteminin hard-diskte nerede bulunduğu ile ilgili bilgileri barındırır. Böylece; işletim sistemi RAM'de başlatılır. MBR'ye, sabit diskteki her bölümü bulan bir bölüm tablosu içerdiğinden ana bölüm tablosu da denir.

Ana bellekte veriler uçucudur. Bu nedenle bilgisayarı açtığımızda CPU ana belleğe doğrudan erişemez. CPU'nun ilk eriştiği yer ROM içerisindeki BIOS olarak isimlendirilen özel bir programdır. BIOS, CPU'nun ilk eriştiği MBR'yi okur ve çalıştırır. MBR programı diskin aktif bölümünü tespit eder ve bölümün ilk bloğunu okur. Bölümün ilk bloğu boot'tur. Boot bloğundaki program işletim sistemini ana belleğe yükler.

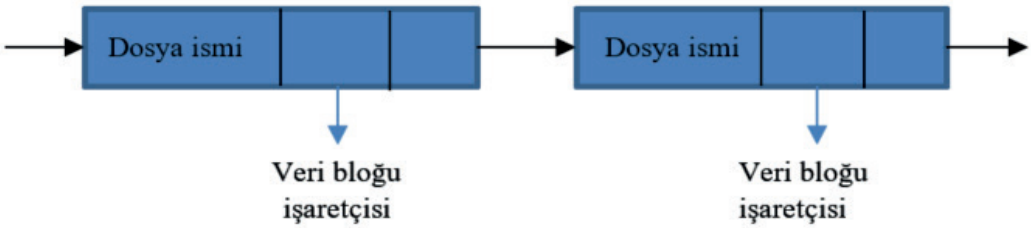
4.5. Dizin Uygulaması

Dizin yapılarının uygulanmasında algoritmalardan faydalanılmaktadır. Algoritma seçimi sistemin performansını etkiler. Yaygın olarak kullanılan 2 algoritma vardır: Doğrusal liste algoritması, Hash tablosu algoritması.

Doğrusal Liste Algoritması: Bu algoritmada bir dizindeki tüm dosyalar bağlı liste olarak tutulurlar. Bağlı liste veri yapısının davranışlarını sergiler. Her bir dosya, veri bloğunun adresini ve dizindeki bir sonraki dosyayı gösteren işaretçiler (pointer) içerir (Bk. Çizim 31) .

Yeni bir dosya oluşturulduğunda öncelikle tüm liste gezinir ve yeni oluşturulan dosya ile aynı isme sahip bir dosya var mı diye kontrol edilir. Eğer yok ise o zaman dosya oluşturulur. Bağlı liste olduğu için listenin başına veya

sonuna eklenir. Dosya üzerinde gerçekleştirilecek silme, oluşturma, güncelleme gibi işlemlerin hepsi için liste baştan sona tek tek gezinilmek zorundadır. Bu da sistemin verimliliğini düşürür.



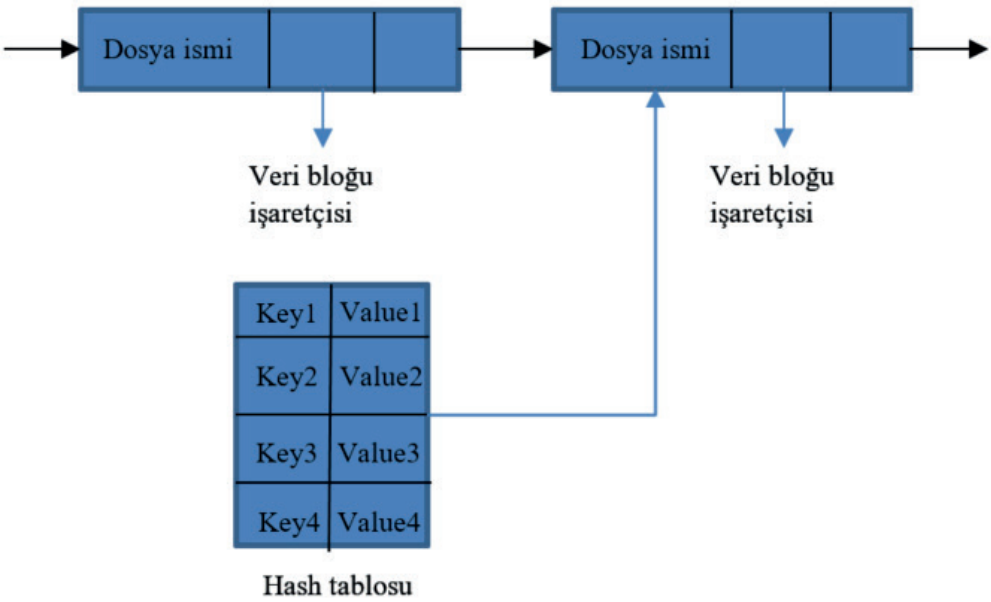
Çizim 31. Dosya Sistemlerinde Bağlı Liste Yapısı ((javaTpoint, 2021).

Hash Tablo Algoritması: Doğrusal bağlı liste algoritmasının eksiklerinin üstesinden gelebilmek için bağlı listenin yanında hash tablosunu da kullanan yaklaşımdır. Bu yaklaşımda her bir dosya için bir anahtar-değer çifti oluşturulur ve bu değerler hash tablosunda saklanır (Bk. Çizim 32). Anahtar değer, dosya adına hash fonksiyonunun uygulanmasıyla elde edilir. Değer ise dizinde bulunan ilgili dosyanın adresini gösterir.

Hash tablosunun kullanımı ile dosya arama işlemi daha hızlı olur. Bütün liste tek tek gezinmez. Hash tablo kontrol edilir. Aranan kayıt hash tablosunda varsa ilgili dosyanın adresi alınır ve doğrudan ilgili dosyaya erişim gerçekleştirilir.

$$Hash_{fonk}(dosya_{ismi}) = key$$

$$Value = Dosya\ adresi\ işaretçisi$$

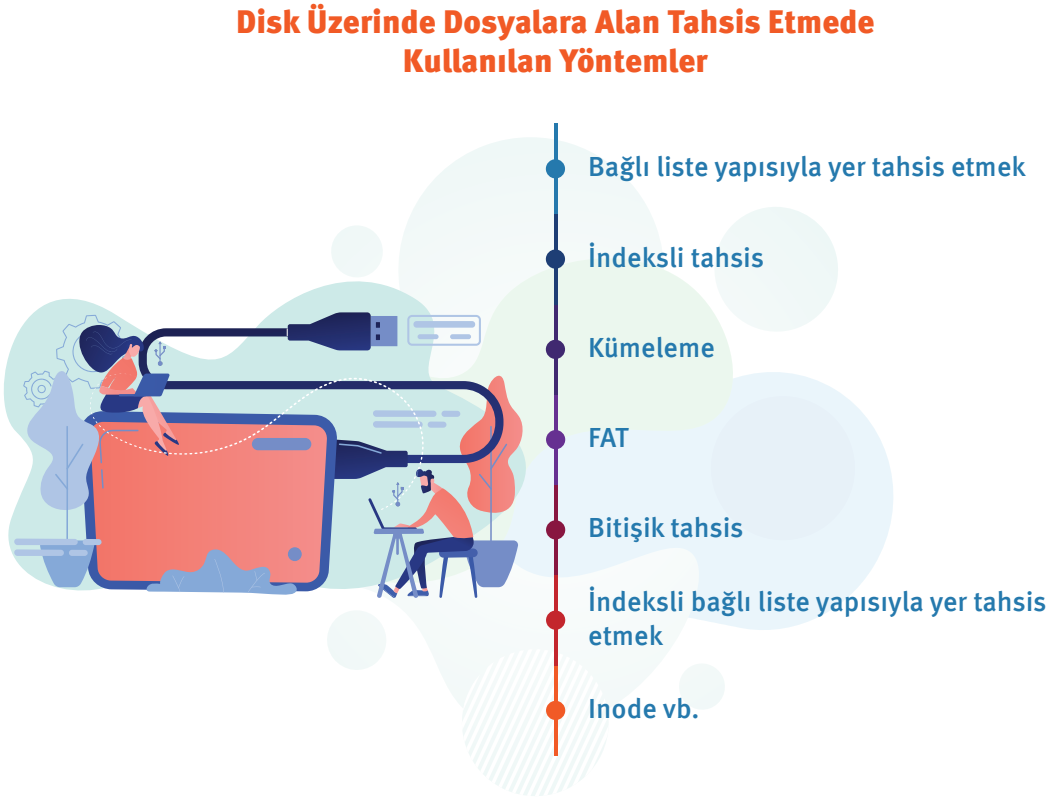


Çizim 32. Hash Tablosu

4.6. Tahsis Etme Yöntemleri

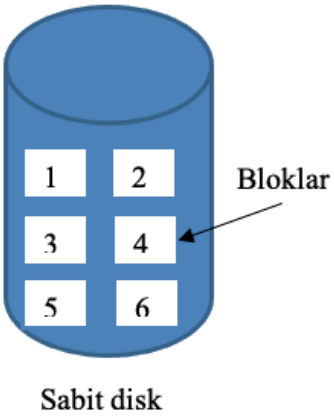
Dosyalar diskler üzerinde saklanır. Disk üzerinde dosyalara alan tahsis etmede farklı yöntemler kullanılmaktadır. Bu yöntemlerden bazıları;

Kullanılan yöntemlerden birkaçını ayrıntılı olarak görelim.



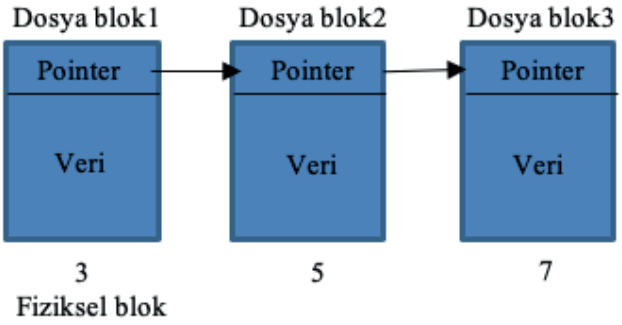
Şekil 11. Disk Üzerinde Dosyalara Alan Tahsis Etmede Kullanılan Yöntemler

Bitişik Tahsis: Dosyalar mantıksal bloklara bölünür ve bu bloklara aynı sırada sabit disk üzerinde arda ardına fiziksel bloklar tahsis edilir (bk. Çizim 33).



Çizim 33. Bitişik Tahsis

Bağlı Liste Yapısıyla Yer Tahsis Etmek: Bu yöntemde bağlı liste veri yapısı kullanılmaktadır. Dosya bloklara ayrılır. Her blok bir diğer bloğu göstermek için bir gösterici (pointer) ve bloğun verisini içerir (bk. Çizim 34). Bloklar sabit diskte sırayla kaydedilmek zorunda değildir. Diskte boş ve uygun yere konabilir. Dosyanın ilk adresinin bilinmesi dosyaya ait tüm bloklara erişmek için yeterlidir. Bağlı listenin en büyük eksikliği belirli bir bloğa rastgele bir erişim gerçekleştirilememesidir.



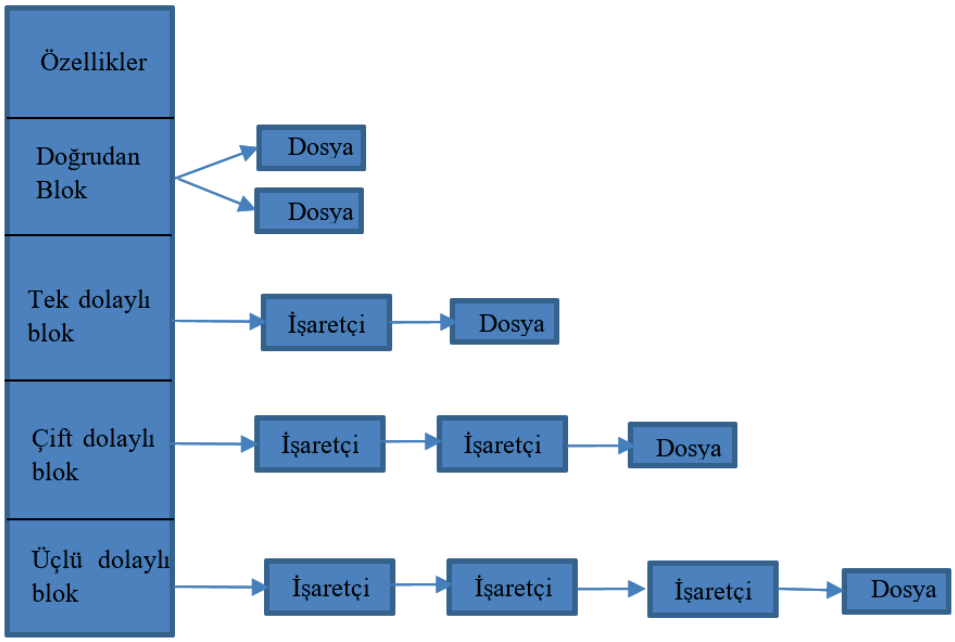
Çizim 34. Bağlı Liste Yapısıyla Yer Tahsisi.

FAT-Dosya Tahsis Tablosunun Kullanımı: Dosya tahsis tablosu disk bloğu bağlantılarının hepsini toplar. Her bir disk bloğu için bir girdi mevcuttur. Girdi, blok sayısı ile indekslenir. FAT kullanılarak bir varlığa erişmek kolaydır. Bütün listeyi dolaşmaya gerek kalmaz. FAT'tan istenilen blok okunur ve bloğa erişilir. MS-DOS ve NT Windows önceki versiyonlarında kullanılmaktadır. FAT dosya sisteminin girdilerin bit sayısının farklılıklarına göre FAT12, FAT16, FAT32 türleri geliştirilmiştir.

İpucu
Windows konusuna 5. bölümde yer verilecektir.

Inode: Inode; indeks node'un kısaltılmış halidir. Unix temelli işletim sistemlerinde kullanılır. Inode, dosya sistemleri için geliştirilmiş özel bir disk

bloğudur. Inode tarafından her dosya indekslenir (Çizim 35. INode tablosu.). Inode sayısı kadar bir dosya sisteminde dosya ve dizin oluşturulabilir. Her dosyanın bir Inode tablosu vardır. Inode; disk bloklarının hangi dosyayla ilişkili olduğu bilgisini, dosya özelliklerini (dosya boyutu, izinleri gibi) içerir.



Çizim 35. INode Tablosu.

4.7. Boş Alan Yönetimi

Dosya yönetimi diskte bulunan boş alanların yönetiminden de sorumludur. Diskte bulunan boş alanları takip etmelidir. Bunun için kullanılan 2 yaygın yöntem vardır: Bit vektör ve bağlı liste.

Bit vektörde her blok bir bit ile temsil edilir. Bit değeri 1 ise blok boş, bit değeri o ise blok doludur. Bitler, bit haritası vektörünü oluşturur. Başlangıçta tüm bitler boş olduğu için bit haritası vektörü tamamen 1 değerlerinden oluşur.

Bağlı liste yönteminde bütün boş bloklar bağlı liste ile birbirlerine bağlıdır. Kullanılan işaretçi boş blok listesinin ilk bloğunun adresini tutar.

4.8. Disk Planlaması ve Parametreleri

İşletim sisteminin sorumluluklarından biri de disk yönetimini gerçekleştirmektir. Diske gelen her isteğe karşı işletim sistemi adil olmalıdır. Maksimum sistem verimliliği elde edecek şekilde diskte planlamalar gerçekleştirmelidir.

Diske bir istek geldiğinde işletim sistemi öncelikle isteği inceler. Diske veri eklenmesi işlemi ise öncelikle diskte boş blok aranır ve en uygun bloğa verinin eklenmesi sağlanır. Diskten veri okunması veya diskteki verinin güncellenmesi işlemleri için öncelikle ilgili veri disk bloklarında aranır ve veri istekte bulunan prosese verilir veya güncellenir.

Şekil 12. Disk Planlamasında Dikkate Alınan Parametreler

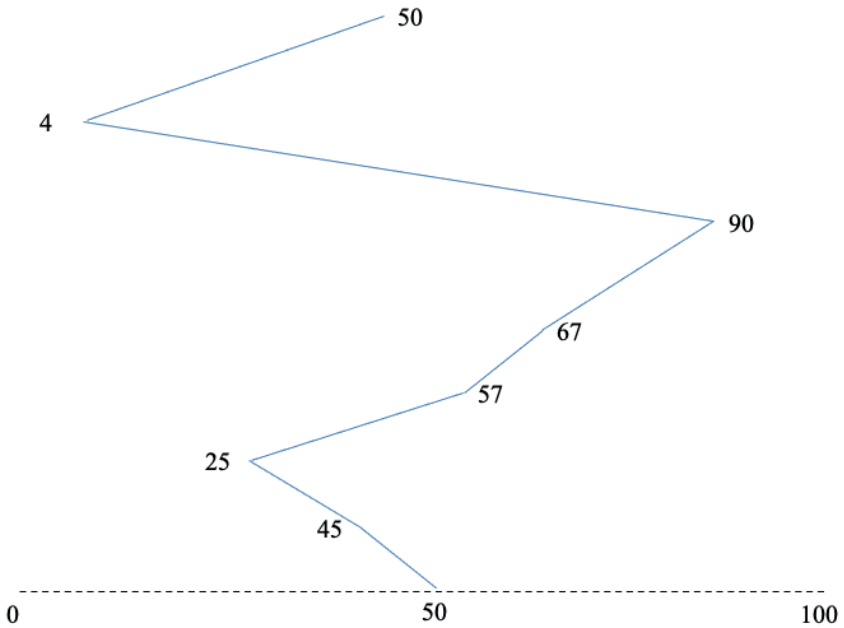
4.9. Disk Planlama Algoritmaları

Şekil 13. Disk Planlama Algoritmaları

FCFS planlama algoritması: En basit planlama algoritmasıdır. Varış sırası dikkate alınır. İlk gelen istek ilk hizmeti alır.

Örnek: 100 silindir içeren bir disk düşünelim. Disk kafası başlangıçta 50 nolu silindirde bulunmaktadır. Sırasıyla 45, 25, 57, 67, 90, 4, 50 nolu silindirlere istek vardır. Silindirin toplam kafa hareketini FCFS planlama algoritmasını kullanarak bulunuz.

Çözüm:

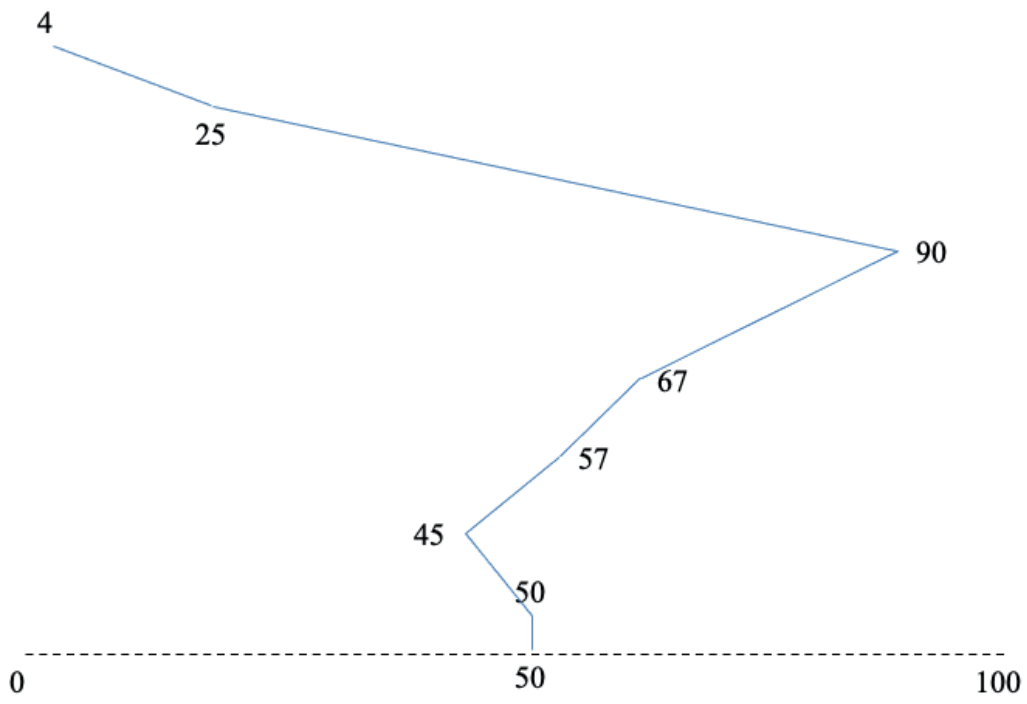


Toplam Kafa Hareketi = (50-45)+(45-25)+(57-25)+(67-57)+(90-67)+(90-4)+(50-4) = 212

SSTF planlama algoritması: En kısa arama süresini dikkate alır. Kafanın mevcut pozisyonuna en yakın silindirlerdeki isteklere öncelik verir.

Örnek: 100 silindir içeren bir disk düşünelim. Disk kafası başlangıçta 50 nolu silindirde bulunmaktadır. Sırasıyla 45, 25, 57, 67, 90, 4, 50 nolu silindirlere istek vardır. Silindirin toplam kafa hareketini SSTF planlama algoritmasını kullanarak bulunuz.

Çözüm:



Toplam Kafa Hareketi = (50-50)+(50-45)+(57-45)+(67-57)+(90-67)+(90-25)+(25-4) = 136

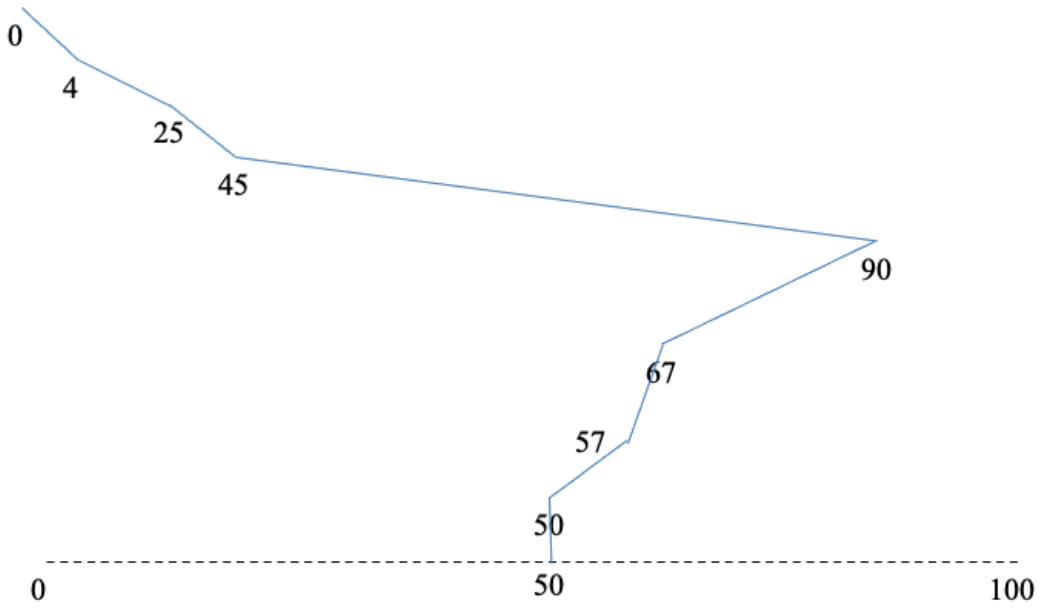
Scan planlama:

Bu planlama çeşidinde disk kafası sonuna kadar belirli bir yöne doğru hareket eder. Yol boyunca gelen tüm isteklere cevap verir. Gittiği yöndeki en son isteğe cevap döndürdükten sonra ters döner. Ters yöndeki isteklere cevap verir. Diskin en başına kadar gider.

Örnek: 100 silindir içeren bir disk düşünelim. Disk kafası başlangıçta 50 nolu silindirde bulunmaktadır. Sırasıyla 45, 25, 57, 67, 90, 4, 50 nolu silindirlere istek vardır. Silindirin toplam kafa hareketini Scan planlama algoritmasını kullanarak bulunuz.

Çözüm:

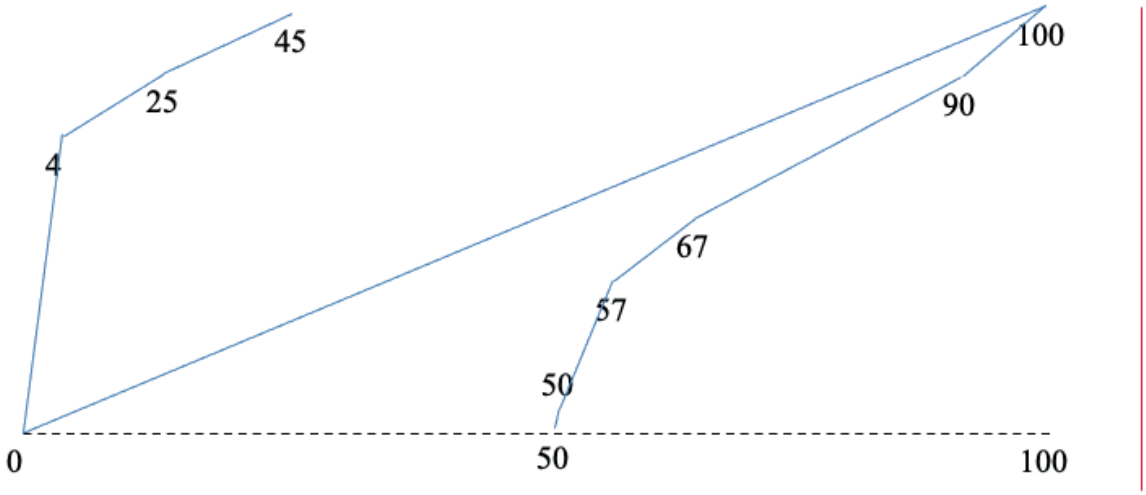
Toplam kafa hareketi=(50-50)+(57-50)+(67-57)+(90-67)+(90-45)+(45-25)+(25-4)+(4-0)= 150



C-Scan algoritması: Scan algoritmasının bir türüdür. Scan algoritmasından farkı disk kafası belirli bir yönde ilerler, yol üzerindeki silindirlere cevap döndürür. Disk kafası yol üzerindeki son silindire geldikten sonra silindirin en başına konumlanır. Silindirin başından başlar ve geri kalan silindirlere cevap döndürür.

Örnek: 100 silindir içeren bir disk düşünelim. Disk kafası başlangıçta 50 nolu silindirde bulunmaktadır. Sırasıyla 45, 25, 57, 67, 90, 4, 50 nolu silindirlere istek vardır. Silindirin toplam kafa hareketini C-Scan planlama algoritmasını kullanarak bulunuz.

Çözüm:



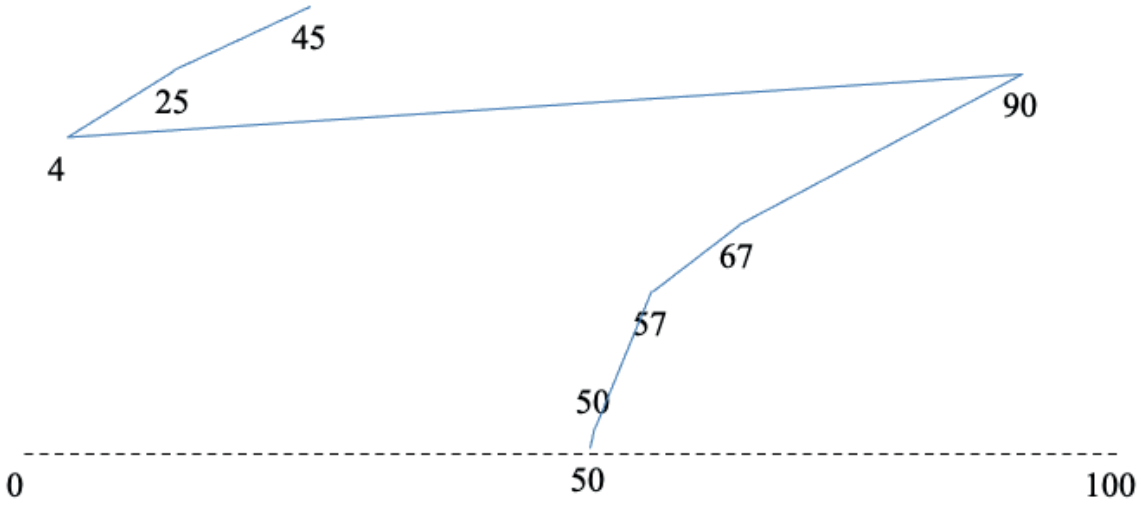
Toplam kafa hareketi=(50-50)+(57-50)+(67-57)+(90-67)+(100-90)+(100-0)+(4-0)+ (45-25)+(25-4)=195

Look planlaması:

Scan ve C-Scan algoritmalarında disk başı diskin sonuna kadar gider. Look planlamasında diskin sonuna kadar gidilmez. Gittiği yönde, gelen en son isteğin bulunduğu diske gider ve yönünü değiştirir. Disk başı yönünü değiştirerek bu yöndeki ilk isteğin bulunduğu silindire gider.

Örnek: 100 silindir içeren bir disk düşünelim. Disk kafası başlangıçta 50 nolu silindirde bulunmaktadır. Sırasıyla 45, 25, 57, 67, 90, 4, 50 nolu silindirlere istek vardır. Silindirin toplam kafa hareketini Look planlama algoritmasını kullanarak bulunuz.

Çözüm:



Toplam kafa hareketi=(50-50)+(57-50)+(67-57)+(90-67)+(90-4)+(45-25)+(25-4)=167

Bu Bölümde Ne Öğrendik?

* Dosya, benzer özellikler gösteren bir dizi kaydı saklayabilmek için kullanılan veri yapısıdır. Benzer özellikler gösteren dosyalar bir araya gelerek dizin yapılarını oluşturmaktadırlar. Farklı seviyelerde oluşturulan dizinler dosya sistemlerini oluşturmaktadırlar.
    * Dosya ve dizinlerin kendilerine özgü özellikleri, dosya ve dizinler üzerinde yapılacak işlemler vardır. Dosya ve dizinler disk birimlerinde saklanırlar.
* Disk üzerindeki dosyaya ulaşmak için doğrudan erişim, indeksli erişim, sıralı erişim yöntemleri kullanılmaktadır.
    * Dosyalar dizin yapıları içerisinde bağlı liste veri yapısı kullanarak tutulmaktadır. Bağlı liste kullanımında dosya üzerinde gerçekleştirilecek silme, oluşturma, güncelleme gibi işlemlerin hepsi için liste baştan sona tek tek gezinilmek zorundadır. Bu da sistemin verimliliğini düşürmektedir. Bu eksikliklerin üstesinden gelebilmek için bağlı listenin yanında hash tabloları da kullanılmaktadır. Her bir dosya için bir anahtar-değer çifti oluşturulmakta ve bu değerler hash tablosunda saklanmaktadır. Hash tablosunun kullanımı ile dosya arama işlemi daha hızlı olur.
* Dosyalar diskler üzerinde saklanırlar. Disk üzerindeki boş alanlar, tahsis etme yöntemleri kullanılarak dosyalara verilmektedir. Boş alanların yönetimi, disk üzerinde en uygun alanın tahsis edilmesi, gelen isteklere en uygun sırada hizmet sunmak, disk planlaması işletim sisteminin bellek yönetimi kapsamında gerçekleştirdiği aktivitelerdir. İşletim sistemi bu aktiviteleri gerçekleştirmek için veri yapılarından ve algoritmalardan faydalanmaktadır.
    * Disk yönetimi işletim sisteminin sorumluluklarından biridir. Diske gelen her isteğe karşı işletim sistemi adil olmalı, maksimum sistem verimliliği elde edecek şekilde diskte planlamalar gerçekleştirmelidir. Arama zamanı, dönme gecikmesi, transfer süresi, disk erişim süresi, disk cevaplama süresi ve bant genişliği disk planlamasında dikkate alınan parametreleri oluşturmaktadır.
* FCFS planlama algoritması, SSTF planlama algoritması, SCAN planlama, C-SCAN planlama, LOOK planlama, C-LOOK planlama yaygın olarak kullanılan disk planlama algoritmalarıdır.

Kaynakça

(tarih yok).

(2021). Debian Manpages: https://manpages.debian.org/ adresinden alındı

Aksan, C. (2021). *Paket Yöneticisi Nedir? Neden İhtiyaç Duyulur?* ceaksan.com: https://ceaksan.com/tr/paket-yoneticisi adresinden alındı

Aladağ, M. (2015). *Windows 10 Yenilikleri – PowerShell 5 ile Paket Yönetimi OneGet*. cozumpark.com: https://www.cozumpark.com/windows-10-yenilikleri-powershell-5-ile-paket-yonetimi-oneget/ adresinden alındı

Alkar, A. (2015). Embedded system basics and application. Ankara.

Çobanoğlu, B. (2018). *Herkes için Python*. Pusula.

Doğu Akdeniz Üniversitesi, B. B. (2021). İşletim Sistemleri-Bellek Yönetimi.

Doğu Akdeniz Üniversitesi, B. V. (2021). İşletim sistemleri-Kilitlenmeler. Siirt.

Gülbağ, A. (2017). İşletim Sistemlerine Giriş.

javaTpoint. (2021). *OS*. javaTpoint: https://www.javatpoint.com/os-attributes-of-a-process adresinden alındı

Kaya, A. (2009). Symbian İşletim Sistemi. *Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri*. Şanlıurfa.

Özdemir, H. (2018). *Linux Paket Yöneticileri*. pythontr.com: https://www.pythontr.com/makale/linux-paket-yoneticileri-632 adresinden alındı

Saatçi, A. (2002). Bilgisayar işletim sistemleri. Ankara: Hacettepe Üniversitesi. http://hilmi.trakya.edu.tr/ders_notlari/os/isleti_sistemleri.pdf adresinden alındı

Samet, R. (2018). Bilgisayar Sistemleri. (A. Ü. Enstitüsü, Dü.)

Shotts, W. (2013). *The Linux Command Line- Second Internet Edition*. No Starch Press.

Silberschatz, A., Gagne, G., & Galvin, P. (tarih yok). *Operating System Concepts*. içinde John Wiley & Sons, Inc. 2021 tarihinde alındı

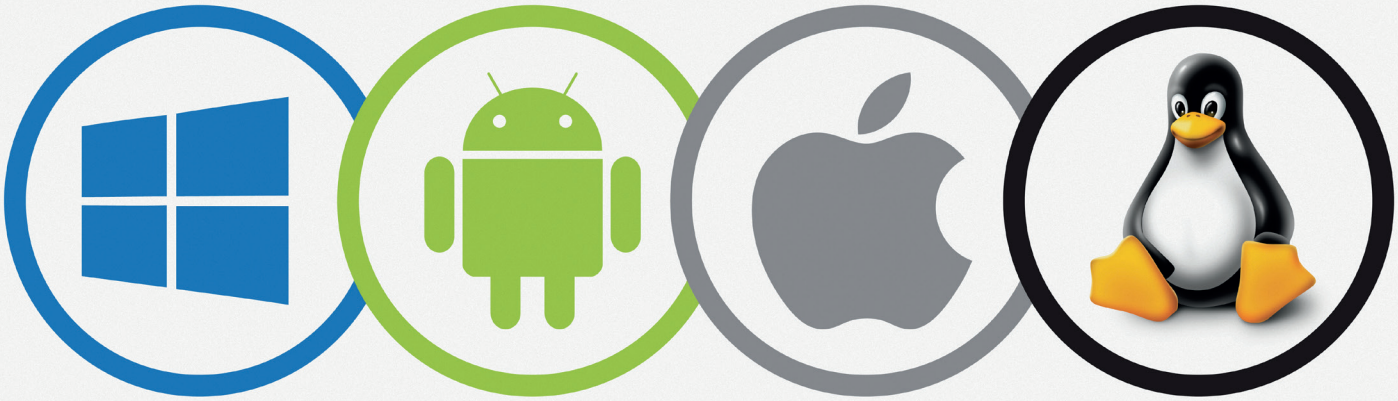
Taşcı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Taşçı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Türkoğlu, İ. (tarih yok). İşletim Sistemleri (Ders Notları). *Fırat Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Bölümü*. Türkiye. 2021 tarihinde alındı

Wikipedia. (2021). *Paket yönetim sistemi*. Wikipedia.org: https://tr.wikipedia.org/wiki/Paket_y%C3%B6netim_sistemi adresinden alındı

Yıldırım, S. (tarih yok). Bilgi Teknolojilerine Giriş3. Atatürk Üniversitesi. 2021 tarihinde alındı



5. LİNUX VE WİNDOWS İŞLETİM SİSTEMLERİ

Tüm zamanların en yaygın bilgisayar virüsleri:
UNIX ve C

Richard P. Gabriel

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Linux işletim sistemini öğrenebilir.
- Linux dağıtımları hakkında bilgi sahibi olabilir.
- Ubuntu kurulumunu öğrenebilir.
- Windows işletim sistemini öğrenebilir.
- Windows lisanslama tiplerini öğrenebilir.

Başlamadan Önce

Bu bölümde yaygın olarak kullanılan Linux işletim sistemleri, Linux dağıtımları ve Windows işletim sistemi ve açıklamalarıyla birlikte sürümleri hakkında bilgiye yer verilecektir. Bununla birlikte, açık kaynak kodlu Linux işletim sistemlerinin kurulumu ve Windows lisanslama türleri anlatılacaktır.

Birlikte Düşünelim

1. Farklı işletim sistemi türlerine neden ihtiyaç duyulmuştur?
2. Açık kaynak kodlu işletim sistemleri güvenilir midir?
3. İşletim sistemi yazılımlarında kaynak kod paylaşımına neden ihtiyaç duyulmuştur?

5.1. Linux İşletim Sistemleri

Linux, açık kaynaklı unix türevi açık kaynak kodlu bir işletim sistemidir. Yüzlerce farklı linux dağıtımı vardır. Bu dağıtımlar, içinde linux çekirdeğini kullanarak farklı yazılımları ve kütüphaneleri kullanım amacına göre paketlerler. Linux çekirdeği Linus Torvalds tarafından yazılmıştır ve ilk sürümü 17 Eylül 1991 tarihinde çıkmıştır. Linux ismi aslında işletim sisteminin çekirdeğine verilen bir isimdir. Linux çekirdeği, GNU Projesinin katkıları ile geliştirilen çeşitli açık kaynak yazılımlar ile zenginleştirilmiştir. Bu nedenle, Linux yerine GNU/Linux isminin kullanımı daha doğru olacaktır. Açık kaynaklı linux kernel kodlarına https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git adresinden erişim sağlayabilirsiniz.

Linus Torvalds (1969-)
Linus, Linux işletim sisteminin geliştiricisi olan bilgisayar bilimcidir. 1991 yılında Helsinki Üniversitesinde bilgisayar bilimi öğrencisi iken, ilk kişisel bilgisayarını satın almış ancak bilgisayarın işletim sisteminden memnun olmadığı için Torvalds Üniversitesinin bilgisayarlarında kullandığı Unix işletim sistemini tercih etmiştir. Bir müddet sonra Unix'in kendi PC tabanlı sürümünü oluşturmaya karar vermiş ve aylarca süren çalışması Linux işletim sisteminin oluşmasını sağlamıştır.

5.1.1. Linux Dağıtımları

Yüzlerce farklı linux dağıtımı vardır. Bu dağıtımlar kullanım amaçlarına göre farklı şekillerde tasarlanmışlardır. Sunucular, kişisel bilgisayarlar, gömülü cihazlar gibi farklı amaçlar için özelleştirilmişlerdir. Bunun yanında ticari veya ticari olmayan amaçlar için ayrılabilir. Farklı donanımlar için çeşitlendirilebilir. Bilimsel amaçlı, otomasyon amaçlı gibi çalıştırılacak iş için de özelleştirilmiş dağıtımlar vardır. Dağıtımların listesine https://distrowatch.com adresinden erişilebilir.

Anahtar Kavram
Sunucu: Diğer kullanıcılar tarafından erişilen kaynakları bir ağda barındıran bilgisayar.

Wikimedia Traffik Analiz Raporuna göre (https://stats.wikimedia.org/archive/squid_reports/2015-01-new/SquidReportOperatingSystems.htm) GNU-tabanlı en çok kullanılan linux dağıtımları şunlardır;

- Debian
  - Knoppix
  - Linux Mint Debian dition
  - Ubuntu
- Fedora
  - Red Hat Enterprise Linux
  - CentOS
  - Oracle Linux
- Mandriva Linux
  - Mageia
  - PCLinuxOS
- openSuse
  - SUSE Linux Enterprise
- Arch Linux
  - Manjaro Linux
- Gentoo
- Slackware

Android, linux çekirdeği kullanan bir işletim sistemidir. Google tarafından ticari amaçlı geliştirilmektedir. Akıllı telefonlarda, televizyonlarda ve birçok farklı gömülü cihazlarda kullanılmaktadır.

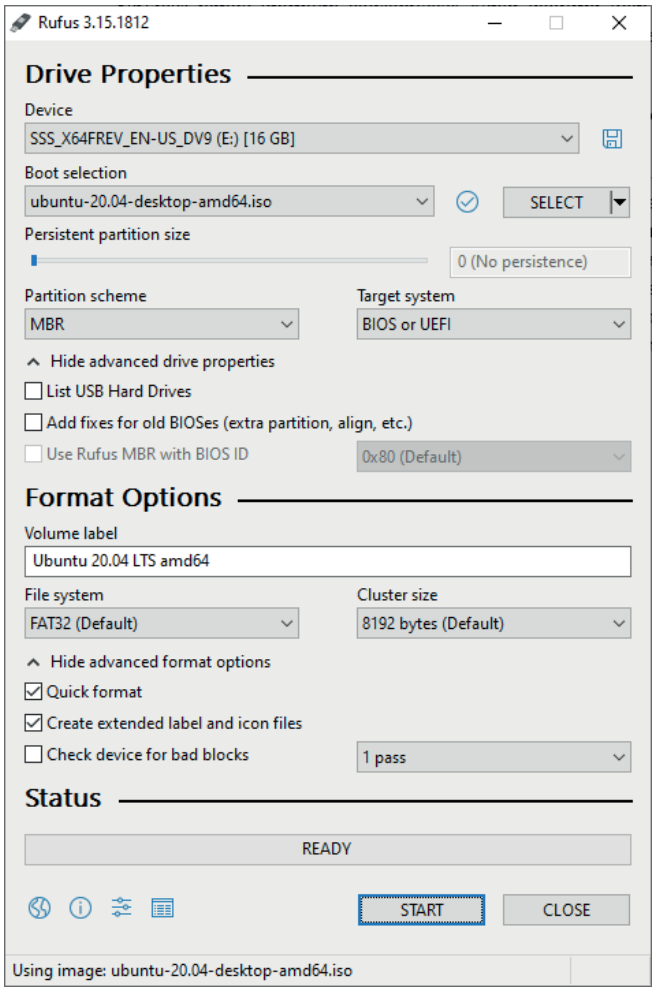
Okuma Önerisi
Çetin, G. (2003). Linux İşletim Sistemi. Ankara: Seçkin Yayıncılık.

Routerlar için özel tasarlanmış linux dağıtımları vardır. Evlerde veya iş yerlerinde kullanılan modemlerde hafif linux dağıtımları kullanılmaktadır. Ağ işlemleri için özelleştirilmiştir. Sadece gerekli olan bileşenleri içerirler. Router, modem gibi gömülü diğer cihazlar için de farklı linux dağıtımları vardır.

Bulut çözümler için de tasarlanan dağıtımlar mevcuttur. Amazon, Amazon Web Services içinde optimum çalışabilecek kendi dağıtımı olan Amazon Linux'ü önermektedir. Kubernetes kullanımı için RancherOS, sanallaştırma için OpenVZ, Xen, KVM gibi dağıtımlar kullanılabilir.

5.1.2. Ubuntu Kurulumu

Kişisel veya iş bilgisayarlarında masaüstü kullanımı için en çok bilinen Debian tabanlı Ubuntu dağıtımının masaüstü sürümünün kurulumunu ve ayarlarını inceleyelim. Güncel sürümü https://ubuntu.com/download/desktop adresinden indirebilirsiniz. Sürümler arasında LTS ifadesi "long-term support (uzun süreli destek)" anlamına gelmektedir. LTS sürümleri için 5 yıl boyunca ücretsiz güvenlik ve bakım güncellemeleri garanti edilir. Ubuntu iso dosyasını bilgisayarınıza indirdikten sonra kullanabilmek için bu dosyayı bir medyaya yazmalısınız. Bu medya bir CD/DVD veya bir usb bellek olabilir. Medyaya iso imajının yazılabilmesi için bir yazılım kullanabilirsiniz. Aksi takdirde iso imajının doğrudan medya içine kopyalanması durumunda bilgisayarınız bu medyadan boot edilemeyecektir. USB bellek içine iso yazma işlemini Resim 1.de gösterildiği gibi Rufus yazılımı ile kolayca yapabilirsiniz:



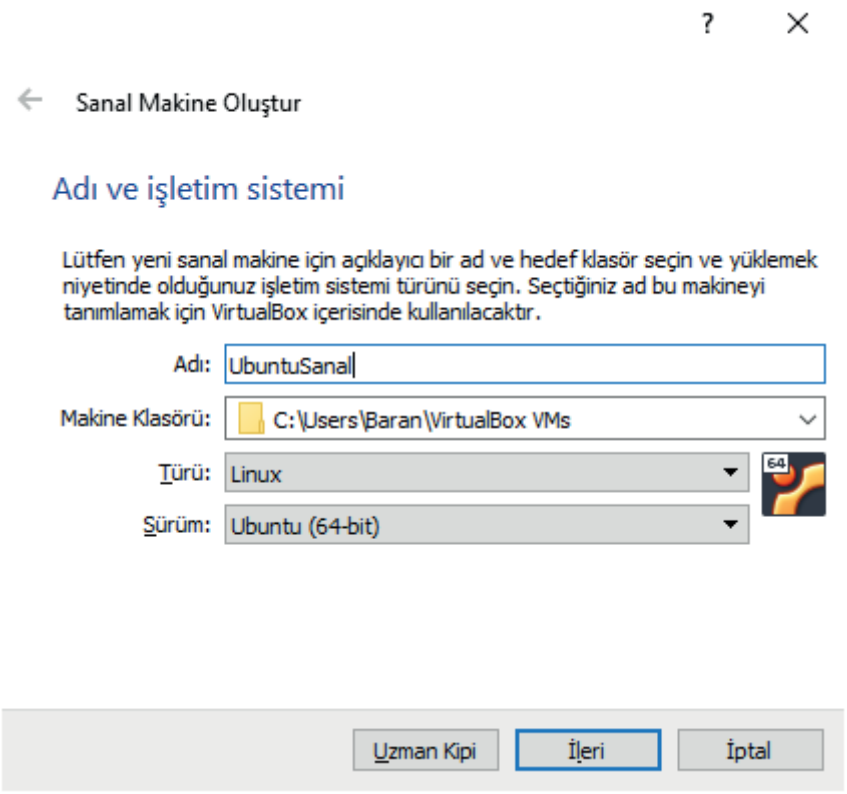
Resim 1. USB Belleğe İSO Yazma

USB medyanıza Ubuntu iso imajını yazdıktan sonra bilgisayarınızı yeniden başlatmalı ve BIOS boot ayarlarınızdan usb medyayı öncelikli olarak seçmelisiniz. Bilgisayarınız USB depolama cihazınızdan boot edildiği zaman karşınıza Ubuntu ekranı gelecektir.

Dilerseniz kurulumu bir sanal bilgisayar ile de deneyimleyebilirsiniz. Bunun için çeşitli sanallaştırma yazılımlarını bilgisayarınıza kurabilirsiniz. Bu bölümde VirtualBox yazılımı ile anlatımı yapacağız. VirtualBox yazılımı ücretsiz bir sanallaştırma yazılımıdır. Kullanmakta olduğunuz işletim sisteminizin içinde kolayca yeni bir sanal bilgisayar oluşturup kullanabilirsiniz. VirtualBox'da yeni bir sanal bilgisayar oluşturmak için Resim 2.de gösterildiği gibi yazılımda bulunan yeni tuşuna basınız:

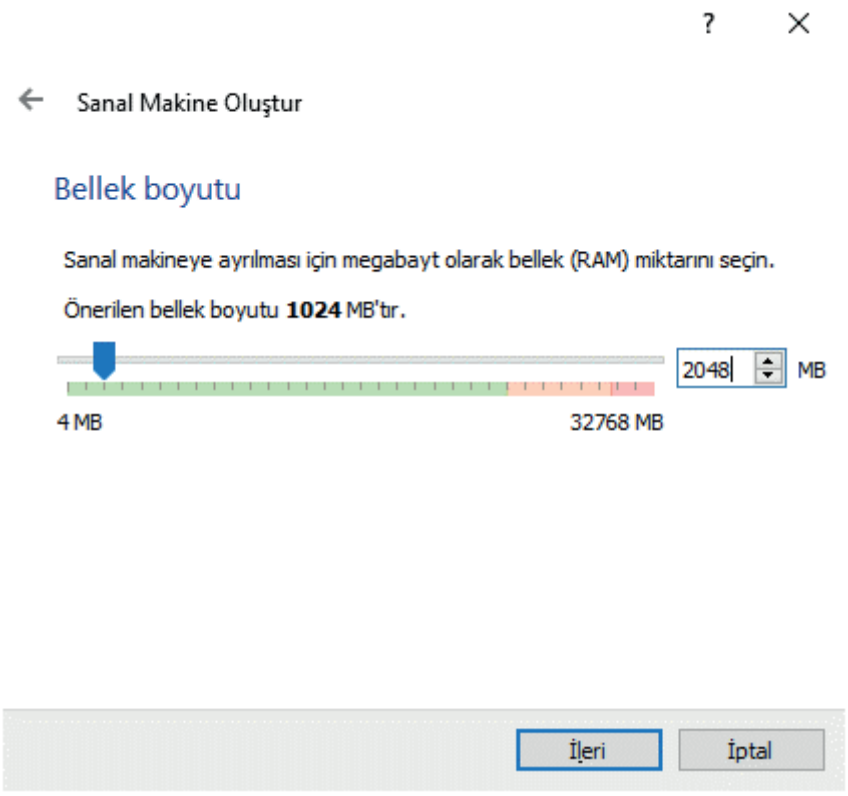
Resim 2. Yeni Sanal Bilgisayar Oluşturma

Sanal bilgisayarınıza bir isim verip kaydetmek istediğiniz klasörü belirtiniz. İlgili işletim sisteminin türünü ve sürümünü seçip ileri tuşu ile bellek boyutu ayarlama ekranına geçebilirsiniz (Resim 3.).



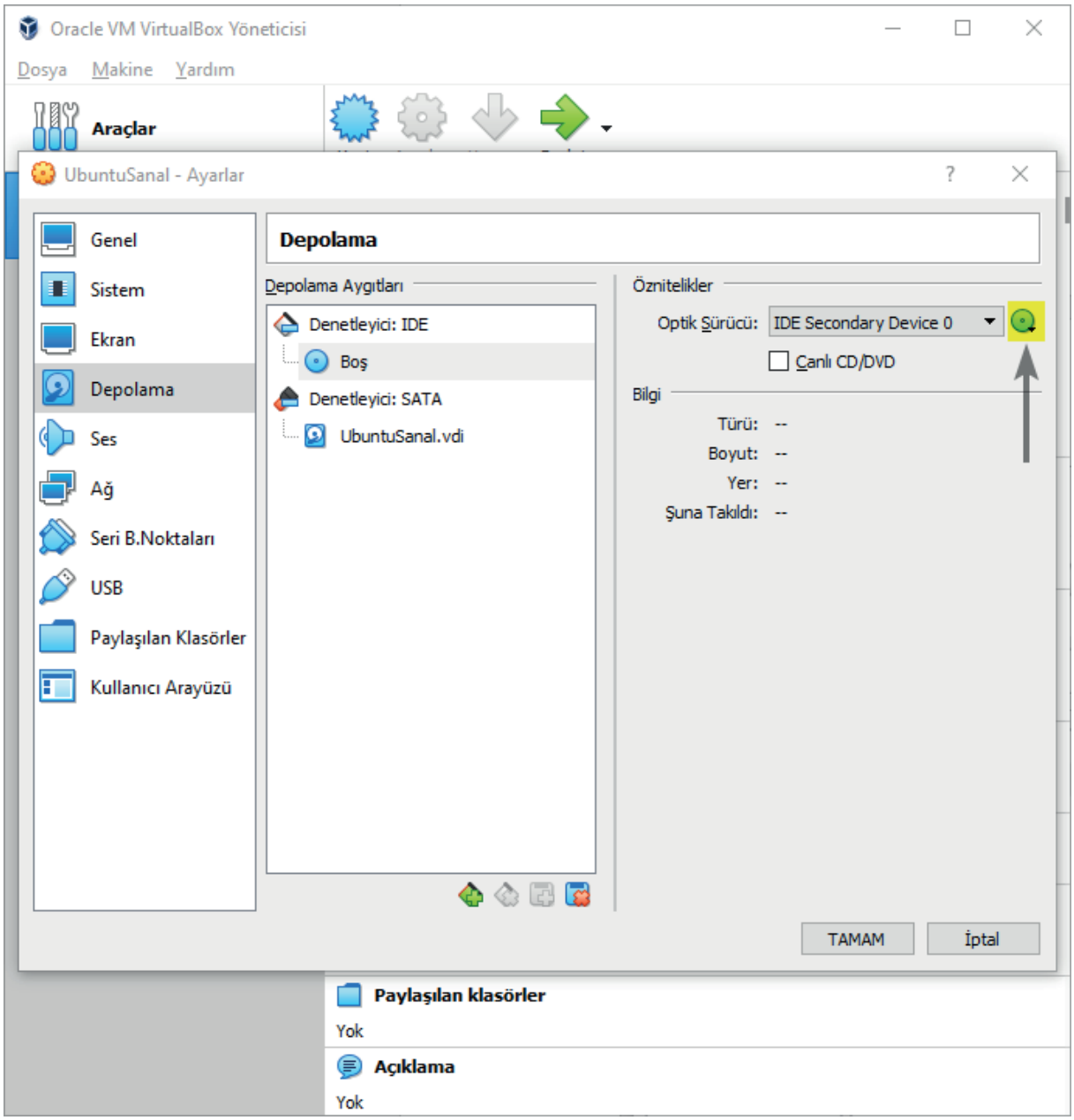
Resim 3. Sanal Makine Adı ve İşletim Sistemi Türü

Sanal bilgisayarınızın bellek boyutunu ayarlarken sisteminizde kullanmakta olduğunuz bellek miktarını dikkate almalısınız. Kapasiteden fazla bellek kullanılması durumunda paging dosyası kullanılacaktır. Paging dosyası, disktedir ve diskte işlem yapmak sisteminiz aşırı derecede yavaşlayacaktır. Sanal bilgisayara çok düşük bellek ayrılması durumunda ise sanal bilgisayarınız yetersiz bellek sebebiyle istenen performansı gösteremeyecektir. Sanal bilgisayarda yapacağınız işleme göre bellek miktarını kendiniz uygun değeri ayarlayabilirsiniz (Resim 4.).



Resim 4. Sanal Makinede Bellek Boyutu Ayarlama

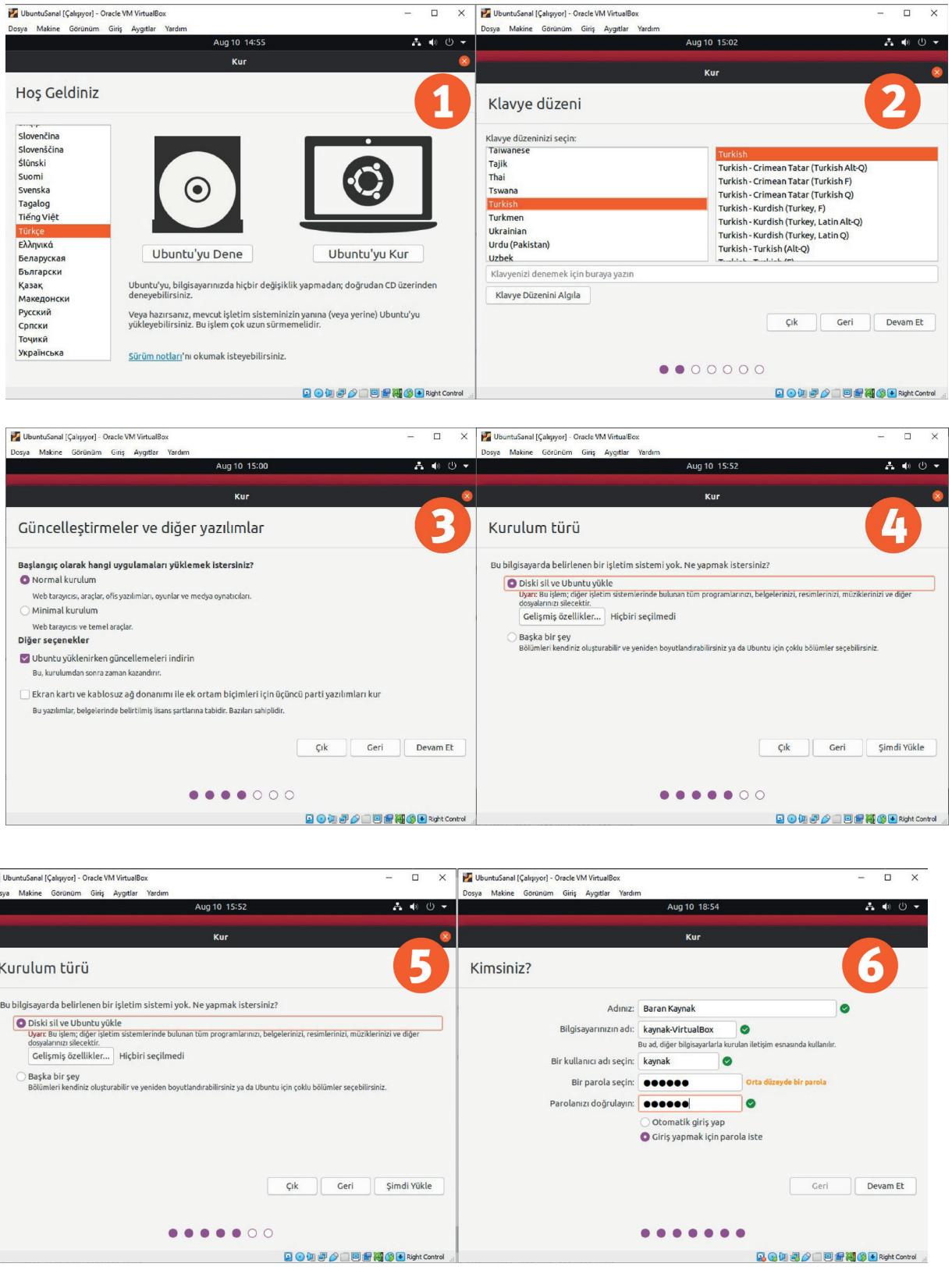
Sanal makinenin iso dosyasından başlatılabilmesi için Ayarlar > Depolama > Optik Sürücü menüsünden ilgili dosyayı seçmelisiniz (Resim 5.).



Resim 5. ISO Dosyası Bağlama

Başlat tuşu ile sanal makineyi başlatabilirsiniz. Sanal makine seçtiğiniz imaj dosyasından boot edilecektir. Ubuntuyu "Ubuntu'yu Dene" tuşu ile kurmadan imaj üzerinden kullanabilirsiniz. Ancak yaptığınız değişiklikler kalıcı olmaz. "Ubuntu'yu Kur" tuşu ile kurulum başlatılır. Kurulum ekranı sizi gerekli ayarları yapmanız için yönlendirecektir. Kurulum esnasında aşağıdaki adımlar karşınıza çıkacaktır. Bu adımlar Resim 6.da gösterilmektedir.

1. Dil seçimi: Kurulum ve sistem dilini seçiniz.
2. Klavye düzeni ayarları: Kullanmakta olduğunuz klavye düzenini seçiniz.
3. Güncelleştirmeler ve diğer yazılımlar: Kurulum anında birtakım yazılımları seçerek kullanıcı yazılımlarını özelleştirebilirsiniz. Dilerseniz kurulum anında güncelleştirmelerin de internetten otomatik yüklenmesini sağlayabilirsiniz.
4. Disk bölümleme: İşletim sisteminin kurulacağı diski ve diskin bölümlerini ayarlayabilirsiniz.
5. Zaman dilimi ayarı: Sistemin zaman dilimini ayarlamalısınız. Türkiye için Istanbul seçilmelidir. Bu durumda sistemde zaman dilimi UTC+3 olarak ayarlanacaktır.
6. Kullanıcı bilgileri: Güvenlik sebebiyle root kullanıcısını doğrudan Ubuntu'da kullanamazsınız. Bunun yerine bir kullanıcı oluşturup bu kullanıcı ile sistemi kullanmanız önerilir. Bu ekranda kullanıcı bilgilerinizi ve parolanızı girerek kullanıcınızı oluşturabilirsiniz.



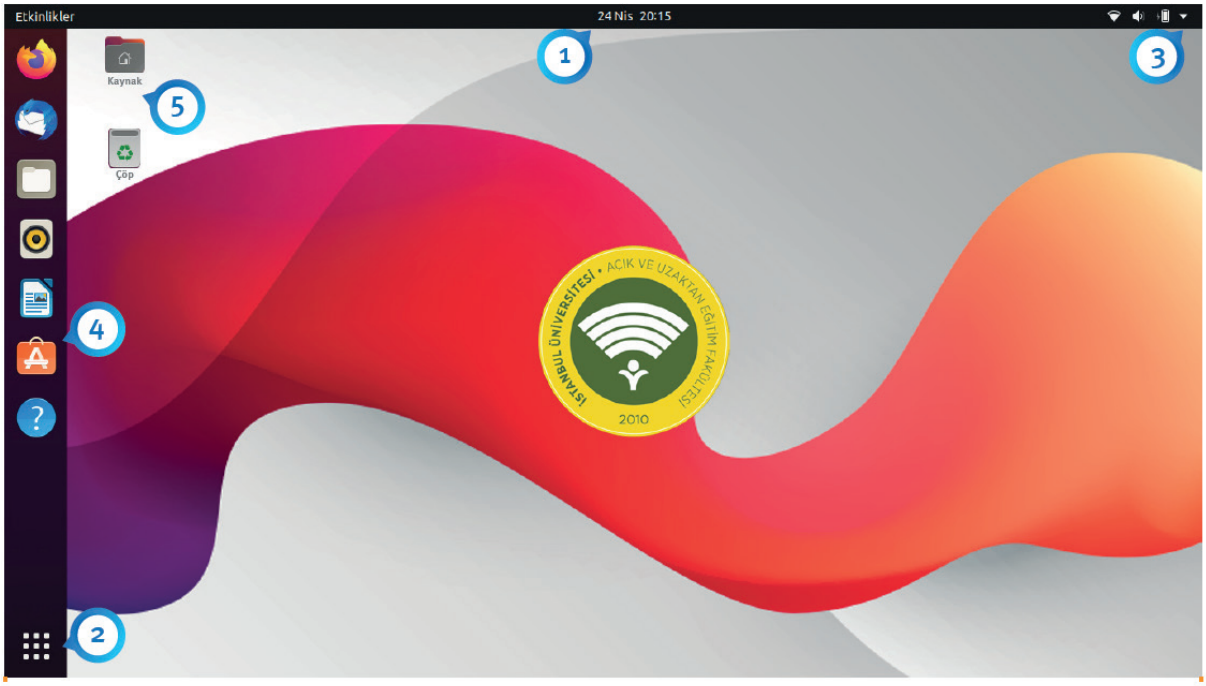
Resim 6. Ubuntu Kurulum Adımları

Ubuntu 20.04 sürümünde, Gnome arayüzü yüklü gelmektedir. Gnome kısaltmasının açılımı GNU Netwok Object Model Environment'tir. Gnome çok popüler bir arayüz yazılımıdır. Masaüstünde kullandığınız tüm arayüzler ve bunlara bağlı fonksiyonlar Gnome ile size sağlanır. Sunucu sistemleri veya gömülü sistemler gibi arayüze ihtiyacı olmayan bilgisayarlarda bir masaüstü yazılımına ihtiyaç yoktur. Bu sistemler için Ubuntu Server gibi içinde herhangi bir kullanıcı arayüzü yazılımı olmayan dağıtımlar tercih edilir. Bu sistemlerde işletim sistemini terminal üzerinden komutlarla yönetmeniz gerekecektir. Gnome dışında farklı kullanıcı arayüzleri de elbette vardır. Bunlar KDE, MATE, Cinnamon, Xfce gibi yazılımlardır. Örnek Kubuntu, içinde KDE arayüzü ile, Lubuntu LXQt arayüzü ile varsayılan olarak gelmektedir.

Sanal makineye kurduğumuz Ubuntu işletim sisteminin temel özelliklerini tanıyalım. Kullanıcınız ile işletim sistemine giriş yaptıktan sonra karşınıza Resim 7.deki gibi bir masaüstü ekranı gelecektir. Masaüstü ekranında aşağıdaki bileşenler vardır:

Anahtar Kavram

Arayüz: Bilgisayar yazılımlarının kullanıcı tarafından ön sayfadır. Üzerinde çeşitli resimler, grafikler, yazılar barındırır.



1. Bildirim menüsü - 2. Uygulamalar - 3. Sistem menüsü - 4. Sık kullanılanlar - 5. Masaüstü öğeleri

Resim 7. Ubuntu Masaüstü

5.2. Windows İşletim Sistemi

Windows işletim sistemi, Microsoft şirketi tarafından geliştirilen ticari bir işletim sistemidir. Kaynak kodları linuxde olduğu gibi açık değildir, kapalı kaynak kodludur. 1985 yılında ilk sürümü piyasaya çıkmıştır. Kişisel bilgisayarlarda en çok kullanılan işletim sistemi olarak piyasayı domine etmiştir. Kurumsal sistemler ve sunucular için Windows NT ailesi geliştirilmiştir. Teknolojinin gelişmesi ile taşınabilir cihazlar, iot cihazları gibi yeni cihazlar ortaya çıkmıştır. Windows işletim sisteminin de bu tip yeni cihazları destekleyen farklı sürümleri çıkarılmıştır.

5.2.1. Windows İşletim Sistemi Sürümleri

Windows işletim sistemi, ilk sürümü olan Windows 1.01 ile 20 Kasım 1985 tarihinde piyasaya çıkmıştır. Windows 2.0, Windows 3.0, Windows 3.1 sürümleri ile zamanla gelişmiştir. Windows'un ilk sürümleri 16 bitlik DOS tabanlı sürümlerdir. 32 bitlik DOS tabanlı sürümlere, bir başka deyişle Windows 9x ailesine, Windows 95 ile geçilmiştir. Windows 95, 15 Ağustos 1995 tarihinde piyasaya sürülmüştür. Windows 98 ve Windows 98 SE (Second Edition) sürümleri 1998 ve 1999 tarihlerinde çıkmıştır. Windows Me (Millennium Edition), 2000 yılında son DOS tabanlı olan Windows işletim sistemi olarak piyasaya sürülmüştür.

Microsoft, Windows NT (New Technology) sürümlerini iş amaçlı tasarlamıştır. Özellikle kurumsal bilgisayarlar için tasarlanan bu işletim sistemi çekirdeği, Windows XP'de de kullanılmıştır. İlk sürümü Windows NT 3.1'dir. 1993 yılında çıkmıştır. Windows NT 3.5, Windows NT 4.0 sürümleri sırasıyla 1995 ve 1996 tarihlerinde piyasaya çıkmıştır. Windows NT 5x ailesini, 4 adet farklı işletim sistemini kapsamaktadır. Bunlar;

- Windows 2000: 17 Şubat 2000 tarihinde piyasaya sürülmüştür. Kullanıcı ve kernel katmanlarından oluşmaktadır. Donanım faaliyetlerinden kernel katmanı sorumludur. Kullanıcı grafik arayüzü işlemlerinden ise kullanıcı katmanı sorumludur.
- Windows XP: 25 Ekim 2001 tarihinde piyasaya çıkmıştır.
- Windows Fundamentals for Legacy PCs (FLP): Windows XP'nin çalışması için gerekli olan donanım yeterliliklerine sahip olmayan eski bilgisayarlar için geliştirilmiş olan, temel bileşenleri içeren bir Windows sürümüdür.
- Windows Server 2003: Kurumsal amaçlı çok geniş gereksinimleri karşılamak amacı ile üretilmiştir. 25 Nisan 2003 tarihinde piyasaya çıkmıştır. .NET framework yapısını içeren ilk sürümüdür. Sunucu sistemlerinde kullanılmak amacı ile üretilmiştir. İçinde birçok sunucu yazılımları bir bileşen olarak hazır gelir. Bu yazılımlar sürümlerine göre farklılık gösterir.

Windows NT 6x ailesi ile Windows Vista, Windows 7, Windows 8, Windows 8.1 ve Windows 10 sürümleri çıkarılmıştır.

- Windows Vista: 8 Kasım 2006 tarihinde ilk sürümü yayımlanmıştır. Grafik arayüzlerinde eski sürümlere göre ciddi yenilikler sunmuştur.
- Windows 7: 22 Ekim 2009 tarihinde piyasaya sürülmüştür. Windows Vista'ya göre daha hızlı ve stabildir.
- Windows 8: 26 Ekim 2012 tarihinde satışa sunulmuştur. Metro tasarım dili ile kullanıcı arayüzünde değişiklikler yapmıştır. Dokunmatik ekranlar için daha kolay kullanılabilir olması hedeflenmiştir. Windows mağazası ile kullanıcılar uygulamaları online indirebilmekte ve yönetebilmektedirler. UEFI güvenli boot özelliği desteklenmiştir. Sosyal hesaplarla eşitleme özellikleri gelmiştir.
- Windows 8.1: Windows 8'de eleştirilen eksiklikler bu sürümde giderilmiştir. Haziran 2013 tarihinde Windows 8'e bir güncelleştirme olarak gelmiştir.
- Windows 10: 29 Temmuz 2015 tarihinde piyasaya çıkmıştır. Kullanıcıların farklı cihazlarda işletim sistemini daha tutarlı kullanabilmesi için geliştirmeler içerir. 32 bit ve 64 bit işlemci desteğinin dışında ARM işlemcilerinin de destekleneceği belirtilmiştir.
- Windows 11: 24 Haziran 2021 tarihinde tanıtılmıştır. Donanımsal güvenlik gereksinimi gerektirmektedir. Kurulum için TPM 2.0 (Trusted Platform Module) donanımına sahip bir anakart gerekmektedir. Önyükleme (boot) için eski tip BIOS yerine UEFI gerekmektedir.

Windows NT 5x ve NT 6x ailesi hibrit çekirdek içermektedir. Windows NT, Windows Server ailesini de kapsamaktadır.

Windows CE, Windows Embedded, Windows Mobile ve Windows Phone isimli farklı ailelerde Microsoft çeşitli işletim sistemleri geliştirmiştir. Bu aileler, çeşitli platformlar için özel çıkarılmış sürümlerdir.

5.2.2. Windows Lisanslama Tipleri

Windows, Microsoft şirketi tarafından pazarlanan ticari bir üründür. Linux gibi açık kaynaklı bir yazılım değildir ve kar amacı gütmektedir. Bu sebeple Windows, kullanıcılara lisanslanarak satılmaktadır. Windows lisanslama üç farklı türde yapılabilmektedir:

1. OEM Lisansı
2. Parekende (Retail) Lisansı
3. Toplu (Volume) Lisans

OEM (Original Equipment Manufacturer) lisanslama modelinde, işletim sistemi kullanıcıya donanım ile verilir. Lisans donanıma bağlanmıştır. Lisansı farklı bir donanıma taşımanız mümkün değildir. Büyük bir donanım değişikliğinde yeni bir lisans alınması zorunludur.

Perakende lisanslama modeli FPP (Full Packaged Product) olarak da bilinir. Lisans donanımdan bağımsızdır. Kullanıcı istediği bilgisayara lisansını taşıyabilir. Bir kez alması yeterlidir. Ancak en pahalı lisanslama modelidir.

Toplu lisanslama modeli, çok fazla bilgisayarı olan şirketlerde kullanılır. Bu tip yerlerde lisansların çok hızlı ve sık aktive edilmesi gerekebilir. Bunun için her biri için tekrar satın almakla uğraşmamak adına toplu bir şekilde hacimli miktarda lisans satışı yapılır. Microsoft, bu kurumlara KMS (Key Management Service) sunucusu kurar. Kuruma atanan VLK (Volume License Keys) anahtarları KMS sunucusu ile yerel olarak etkinleştirilir ve kurum içinde kullanılır. Akademik Toplu Lisanslama da bu yöntemin okullar için olan türüdür. Academic Volume Licensing olarak bilinir.

Bu Bölümde Ne Öğrendik?

* Günümüzde en çok kullanılan işletim sistemleri Linux tabanlı işletim sistemleri ve Windows işletim sistemleridir.
    * Linux tabanlı işletim sistemleri açık kaynak kodlu işletim sistemleri örneğidir. Windows tabanlı işletim sistemlerinin kaynak kodları kapalıdır ve ticari bir işletim sistemidir.
* Kullanım amacına göre farklı şekillerde tasarlanmış Linux çekirdeğini kullanan işletim sistemi dağıtımları vardır. Sunucular, kişisel bilgisayarlar, gömülü cihazlar için, hatta özel bir iş için özelleştirilmiş Linux dağıtımları vardır. Bu dağıtımların hepsinin ortak noktası Linux çekirdeğini kullanmalarıdır.
    * Windows işletim sistemleri Microsoft şirketi tarafından geliştirilen ticari bir işletim sistemidir. Grafik arayüzü gelişmiştir. Kullanıcı dostudur. Kişisel bilgisayarlarda en çok kullanılan işletim sistemidir.
* Sunucularda, mobil cihazlarda, kişisel bilgisayarlarda kurumsal şirketlerde, IoT cihazlarında kullanılabilen farklı Windows işletim sistemi sürümleri geliştirilmiştir.
    * Windows, Microsoft şirketi tarafından pazarlanan, kâr amacı taşıyan bir üründür. Kaynak kodları kapalıdır. Bu nedenle Microsoft, Windows işletim sistemi kullanıcılara lisanslanarak satılmaktadır.
* İşletim sisteminin donanımla beraber verildiği lisanslama modeli OEM olarak isimlendirilir. OEM modeli donanıma bağlıdır.
    * Donanıma bağlı kalmadan kullanıcının lisansını istediği bilgisayara taşıyabildiği perakende lisanslama modeli FPP olarak isimlendirilir. Genelde kişisel bilgisayarlarda FPP lisanslama modeli kullanılmaktadır.
* Lisansların çok hızlı ve sık aktive edilmesi gereken çok sayıda bilgisayarın olduğu şirketlerde toplu lisanslama modeli tercih edilmektedir. Microsoft, bu şirketlere KMS (Key Management Service) sunucusu kurar. Kuruma atanan VLK (Volume License Keys) anahtarları KMS sunucusu ile yerel olarak etkinleştirilir ve kurum içinde kullanılır. Akademik Toplu Lisanslama da bu yöntemin okullar için olan türüdür. Academic Volume Licensing olarak bilinmektedir.

6. KABUK KOMUTLARI VE BETİK PROGRAMLAMA

Herhangi bir program sadece yararı kadar iyidir.

Linus Torvalds

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- İşletim sistemlerinde kabuğu öğrenebilir.
- Linuxde temel kabuk komutlarını öğrenebilir.
- Linuxde prosesleri yönetebilir.
- Bash ile betik programlamayı öğrenebilir.

Başlamadan Önce

Bu bölümde Linux işletim sistemlerinin kabuklarından biri olan bash kabuk komutları ve bash ile betik programlamaya yer verilecektir. Linux işletim sistemi, diğer işletim sistemlerine göre terminalden çok daha kolay kontrol edilebilen bir işletim sistemidir. Diğer işletim sistemlerinin de kendi kabuk komutları ve programlama yöntemleri vardır. Linux kabuk komutlarının çok geniş bir kullanıcı kitlesi olması sebebiyle bu bölümde Linux üzerinden örnekler verilecektir.

Birlikte Düşünelim

1. Linux işletim sistemlerinde proseslerin listelenmesini sağlayan komutlar nelerdir?
2. Kullanıcılar işletim sistemlerini nasıl kullanırlar?
3. İşletim sisteminde sunulan grafik arayüzü ile yapamadığınız işlemleri nasıl gerçekleştirebilirsiniz ve bunları otomatize edebilir misiniz?

6.1. Kabuk Nedir?

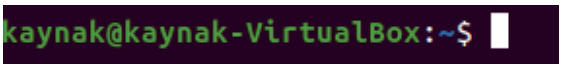
İşletim sistemine bir kullanıcı olarak komut göndermek için bir arayüze ihtiyacımız vardır. Grafiksel arayüzlerde bu komutları kullanıcı ekrandaki öğeler aracılığıyla kolaylıkla yapabilir. Ancak grafik arayüzleri, sizi kısıtlar ve istediğiniz tüm komutları çalıştırmanıza veya komutların parametrelerini özelleştirmenize çoğunlukla yardımcı olamaz. Kabuk (shell), bir komut satırı olarak bilinir. Kabuk, komutlarınızı klavyeden alıp işletim sistemine ileten bir programdır. Linux dağıtımlarında çoğunlukla bash isimli bir programdır. Bash, GNU Project tarafından geliştirilmiştir.

Ubuntu 20.04 dağıtımında, "Uçbirim" (Terminal) yazılımını kullanarak komutlarınızı işletebilirsiniz. Uçbirim yazılımını açmak için Ctrl+Alt+T kısayolunu kullanabilirsiniz. Ubuntu masaüstü yazılımı olan Gnome'u kullanırken, erişmekte olduğunuz "Uçbirim" yazılımı, sizin komutlarınızı yazabildiğiniz bir editördür. Uçbirim, komutlarınızı aslında kabuk ile işletir ve sonuçlarını ekranda gösterir. Gnome ile sunulan bu arayüzü kullanmadan, kabuğa daha ilkel bir şekilde erişmek mümkündür. Gnome ile çalışırken, Ctrl+Alt+F3 tuşlarına basarak doğrudan Resim 8.de gösterilen sistem konsoluna (tty) erişebilirsiniz. Tekrar Gnome arayüzüne dönmek için Ctrl+Alt+F1 tuşuna basmanız gerekir.

```
Ubuntu 20.04 LTS kaynak-VirtualBox tty3

kaynak-VirtualBox login: kaynak
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.11.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


337 güncelleme hemen kurulabilir.
Bu güncellemelerin 0 tanesi güvenlik güncellemesidir.
Bu ek güncellemeleri görmek için şu komutu çalıştırın: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** Sistemin yeniden başlatılması gerekiyor ***
Last login: Sat Aug 14 22:46:41 +03 2021 on tty3
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kaynak@kaynak-VirtualBox:~$ _
```

Resim 8. Ubuntu TTY Konsol Erişimi

Terminalde "kullanıcı@bilgisayaradı:~$" şeklinde Resim 9.'daki gibi bir satır ile karşılaşırsınız. Burada hangi kullanıcı ile hangi bilgisayarda ve hangi dizinde olduğumuzu görebiliriz. $ işareti yerine # işaretini görüntülediğiniz zaman, çalıştıracağınız komutlar superuser ayrıcalığı ile çalıştırılacaktır. Bir başka deyişle sistemde yetkili bir kullanıcı olarak komutlarınız çalışacaktır.



Resim 9. Terminal Komut Satırı

6.2. Linux Kabuk Komutları

Temel Linux komutları ile GUI olmadan herhangi bir komut satırı ekranı (CLI) ile Linux işletim sisteminizi yönetebilirsiniz. Bu bölümde anlatılan komutlar temel komutlardır. Linux'de komutların büyük küçük harfe duyarlı olduğunu unutmayalım.

6.2.1. Dosya ve Dizin İşlemleri

Linux'de her şey bir dosya olarak temsil edilir. Bu sebeple dizinlerin yapısını anlamak ve hızlıca dolaşabilmek için dizin komutlarını bilmek gerekir.

- pwd: Bulunduğunuz dizini gösterir. (Print Working Directory)
- cd: Dizinler arasında geçiş yapmayı sağlar. (Change Directory)
- ls: Dizinin altındaki dosya dizinleri listeler. (List Directory)

Anahtar Kavram

Terminal: Veri iletişim ortamında veri giriş ve çıkışına imkan veren donanım birimi.

Bulunduğunuz dizinin neresi olduğunu öğrenmek için pwd komutu çalıştırılır. Terminalde oturum açtığınız zaman varsayılan olarak oturum açtığınız kullanıcının home dizininde işlem yaparsınız. Terminalde ~ işareti ile bulunduğunuz kullanıcının home dizini belirtilir. Tam açılımını görebilmek için pwd komutunu çalıştırdığımızda "/home/kaynak" şeklinde bir çıktı alırız.

Bir başka dizin içinde işlemlerimize devam etmek isteyebiliriz. Bu durumda cd komutu kullanılmalıdır. cd komutu parametre olarak gitmek istediğiniz dizinin yolunu ister. Bu yolu tam yol olarak veya bulunduğunuz dizine bağlı olarak verebilirsiniz. /home/kaynak dizininin içinde Resimlerim isminde bir klasör var ise /home/kaynak dizinde iken "cd Resimlerim" yazarak ilgili klasöre girebilirsiniz. Ayrıca "cd /home/kaynak/Resimlerim" şeklinde tam yol belirterek de ilgili dizine girmek mümkündür. Bir üst dizine temsil etmek için iki (..), mevcut dizini temsil etmek için ise tek nokta (.) kullanabilirsiniz. Resim 10.da bu komuta ait örnekler gösterilmektedir:

```
kaynak@kaynak-VirtualBox:~$ cd Resimler/
kaynak@kaynak-VirtualBox:~/Resimler$ pwd
/home/kaynak/Resimler
kaynak@kaynak-VirtualBox:~/Resimler$ cd ..
kaynak@kaynak-VirtualBox:~$ cd /home/kaynak/Resimler/
kaynak@kaynak-VirtualBox:~/Resimler$ cd ../Müzik/
kaynak@kaynak-VirtualBox:~/Müzik$ pwd
/home/kaynak/Müzik
```

Resim 10. Dizinler Arası Dolaşma

ls komutu ile bulunduğunuz dizindeki dosyaları ve dizinleri görüntüleyebilirsiniz. Bakınız (Resim 11.).

```
kaynak@kaynak-VirtualBox:~$ ls
Belgeler  Genel  İndirilenler  Masaüstü  Müzik  Resimler  Şablonlar  Videolar
```

Resim 11. ls komutu.

ls komutu, ls [seçenekler] [dosya veya dizin adı] şeklinde kullanılır. "man ls" komutu ile bu komuta ait tüm detayları inceleyebilirsiniz. Aşağıda bu seçeneklerden en çok kullanılanlara yer verilmiştir. Bu seçenekleri birlikte kullanabilirsiniz:

- -a Nokta (.) ile başlayan gizli dosyaları da listeye dahil eder.
- -l Dizini liste biçiminde gösterir. Resim 12.de gösterildiği gibi ilk sütunda dosya veya dizine ait erişim izinleri, ikinci sütunda hard link sayısı, üçüncü sütunda dosya veya dizinin sahibini, dördüncü sütunda grubunu, beşinci sütunda boyutunu ve son değiştirme tarihini liste şeklinde gösterir.

```
kaynak@kaynak-VirtualBox:~$ ls -l
toplam 32
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Belgeler
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Genel
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 İndirilenler
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Masaüstü
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Müzik
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Resimler
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Şablonlar
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Videolar
```

Resim 12. ls Komutu Liste Biçimi.

- -h Dosya boyutlarını byte cinsinden değil, kilobayt, megabayt, gigabayt gibi daha kolay okunabilir şekilde gösterilmesini sağlar.

- -r Listeyi tersine sıralar.
- -t Listedeki öğeleri değişiklik tarihine göre sıralar.

Komut satırından dosyaların türünü öğrenmek için "file" komutu kullanılabilir. Resim 13'te "file kalemler.jpeg" dosyasının detayları gösterilmektedir:

```
kaynak@kaynak-VirtualBox:~/Resimler$ file kalemler.jpeg
kalemler.jpeg: JPEG image data, baseline, precision 8, 1200x900, components 3
```

Resim 13. file komutu.

Text içerikli dosyaların içeriğini görüntülemek için less komutu kullanılabilir. less ile sayfalama şeklinde page up ve page down tuşlarıyla dosyada gezinebilirsiniz. Çıkmak için q tuşunu kullanabilirsiniz.

Dizin oluşturma, taşıma, dosya kopyalama, taşıma, silme gibi dizin ve dosya değişiklikleri için aşağıdaki komutlar kullanılır:

- cp: Dosya ve dizinleri kopyalamak için kullanılır.
- mv: Dosya ve dizinleri taşımak için kullanılır. Yeniden adlandırma için de kullanılabilir.
- mkdir: Yeni bir dizin oluşturmak için kullanılır.
- rm: Dosya ve dizinleri silmek için kullanılır.
- ln: Hard veya soft link oluşturmak için kullanılır.

cp komutu "cp [seçenekler] kaynak hedef" şeklinde kullanılır. "cp /home/kaynak/Resimler/kalemler.jpg /home/kaynak/yeni_kalemler.jpg" komutu ile kopyalama yapılabilir. Kopyalama işlemleri bir grafik arayüzü ve fare ile daha kolay yapılabilir. Ancak daha ileri düzeyde bir ihtiyaç olduğunda grafik arayüzü ile işlemler daha uzun sürebilir. Örnek olarak bir dizinde bulunan tüm txt dosyalarını bir başka dizine kopyalamak isteyelim. Bu işlemi yaparken hedef dizinde bu dosyaların bazılarının da olduğunu varsayalım. Hedef dizinde bulunan dosya eğer kaynak dizinde bulunan dosyadan eski ise güncellenmesini isteyelim. Bu durumda "cp -u *.txt /home/kaynak/yeni" komutu bulunduğumuz dizindeki tüm txt uzantılı dosyaları alıp yeni klasörüne kopyalayacaktır. -u parametresi ise eğer kopyalanmak istenen dosya yeni ise üzerine yazacaktır. -r (--recursive) parametresi ise dizinin içindeki tüm alt dizinleri kopyalamak için kullanılır.

mv komutu ile bir dosyayı veya dizini kaynak dizinden hedef dizine taşıyabilirsiniz. Bunun dışında taşırken yeni bir isim verip dosyanın veya dizinin adını da değiştirebilirsiniz. Kullanımı cp komutuna benzerdir.

mkdir komutu ile yeni bir dizin oluşturabilirsiniz. -p parametresi ile kullanıldığında belirtilen yol üzerinde olmayan ana dizinler de oluşturulur. "mkdir -p /home/kaynak/yenidizin/altdizin/" komutu çalıştırıldığında "yenidizin" eğer

mevcut değilse önce yeni dizini oluşturur sonra da “altdizin”i oluşturur. Bu şekilde alt alta dizinleri otomatik oluşturabilirsiniz.

rm komutu dosya veya dizinleri silmek için kullanılır. Komut bilinçsiz kullanıldığı zaman çok tehlikeli olabilir. -i (--interactive) parametresini kullanarak silme işleminden önce kullanıcıya onay için soru sorulmasını isteyebilirsiniz. Aksi takdirde kullanıcıya soru sormadan belirtilen tüm dosya/dizinleri siler. -r (--recursive) parametresi bu komutu tehlikeli yapan parametrelerden biridir. Rekürsif olarak tüm alt öğeler tek bir komutla silinir. -f (--force) parametresi, kullanıcıya sormadan ve olmayan öğeleri göz ardı ederek silinmesini sağlar. “rm -rf /home/kaynak” komutu /home/kaynak dizininde bulunan tüm öğeleri sormadan siler.

ln komutu hard veya sembolik link oluşturmak için kullanılır. Hard link, bir dosyayı gösteren bir adrestir. Orijinal dosyanın tüm özelliklerine ve içeriğine erişirsiniz. Hard linki farklı disk bölümlerinde yapamazsınız, aynı disk bölümünde olmalıdır. Soft link ise daha esnektir. Farklı dosya sistemleri ile bölümlenmiş disklerdeki öğeleri linkleyebilirsiniz. Soft link hem dizin hem de dosyalar için oluşturulabilir. Orijinal dosyanın konumunu gösterir. Soft linkte orijinal dosyanın yetkileri değiştirilse bile linkte bulunan yetkiler değişmez. “ln dosya1.txt dosya2.txt” komutu ile hard link oluşturabilirsiniz. Bu durumda her iki dosya da aslında fiziksel olarak aynı öğeye işaret ediyor olacaktır. Herhangi birinde yaptığınız içerik değişikliği diğerinde de gerçekleşecektir. “ln -s dosya1.txt softlink.txt” komutu ile soft link oluşturabilirsiniz. Soft link komutu ile dizinleri linkleyebilirsiniz, hard link ile dizinler linklenemez. Soft link öğelere farklı yetkiler atanmasını sağlar.

6.2.2. Komut Bilgileri

Shell ile çeşitli komutlar çalıştırıp, çıktıları ekranda görüntülenir. Bu komutlar hakkında daha detaylı bilgiye ihtiyacımız olduğu zaman komutlar hakkında çeşitli bilgileri nasıl ediniriz? Bir komutun hangi dizinde olduğu, ne gibi parametreleri aldığı veya kısaca ne işe yaradığı gibi komutlar hakkında bilgi edinebilmemiz için de çeşitli komutlar vardır. Bununla birlikte birkaç komutu birleştirerek kendi özel komutunuzu sisteme tanıtabilirsiniz. Bu bölümde aşağıdaki komutlar hakkında bilgi verilmektedir:

Şekil 14. Komut Bilgileri

Kabuk ile çalıştırılan komutlar; bir program, kabuk içinde bulunan bir yerleşik komut, bir kabuk betiği (shell script) veya bir kısaltma (alias) olabilir. Programlar, C, C++, GoLang gibi çeşitli dillerde yazılmış kodların derlenmiş hali olabileceği gibi shell, perl, python gibi script dilleri ile de yazılmış olabilirler. Programlar, çoğunlukla /usr/bin klasörü altında saklanır.

Bir komutun hangi tip komut olduğunu anlamak için type komutu kullanılır. cd komutu bir kabuk komutudur. Resim 14.te gösterildiği gibi "type cd" komutunu çalıştırdığımızda, cd komutunun bir kabuk yerleşik komutu olduğunu gösterir. Resim 15.te gösterildiği gibi "type rm" ise rm komutunun bir program olduğunu ve /usr/bin/rm dizininde bulunduğunu gösterir. Resim 16.da ise ls komutunun aslında bir alias olduğunu ve ls komutunun "ls –color=auto" komutunun kısaltması olduğu anlaşılır:

```
kaynak@kaynak-VirtualBox:~$ type cd
cd bir kabuk yerleşiğidir
```

Resim 14. type cd.

```
kaynak@kaynak-VirtualBox:~$ type rm
rm /usr/bin/rm'dir
```

Resim 15. type rm.

```
kaynak@kaynak-VirtualBox:~$ type ls
ls `ls --color=auto' için takma addır
```

Resim 16. type ls.

which komutu ile çalıştırılabilir komut dosyalarının nerede olduğunu öğrenebiliriz. Bir sistemde bazen bir programın birden fazla versiyonu olabilir. Bu versiyonlar aynı komut ile çalıştırılabilir. Bu sebeple belirtilen komutun hangi dizinlerde olduğunu görüntüleyebilirsiniz. Resim 17.de which komutu ile man komutunun binary dosyasının yolunu görüntülenmektedir:

```
kaynak@kaynak-VirtualBox:~$ which man
/usr/bin/man
```

Resim 17. which man.

Kabukta yerleşik olan komutlar için yardım almak isterseniz help komutu yardımcı olacaktır. help komutunu, help komutunun nasıl kullanıldığını öğrenmek için kullandığımızda Resim 18.deki gibi bir çıktı alırız. Bu çıktıda ilgili komutun parametreleri hakkında bilgi verilmektedir. Köşeli parantez içinde verilen parametreler zorunlu değildir:

```
kaynak@kaynak-VirtualBox:~$ help help
help: help [-dms] [desen ...]
    Display information about builtin commands.

    Displays brief summaries of builtin commands.  If PATTERN is
    specified, gives detailed help on all commands matching PATTERN,
    otherwise the list of help topics is printed.

    Options:
      -d        output short description for each topic
      -m        display usage in pseudo-manpage format
      -s        output only a short usage synopsis for each topic matching
                PATTERN

    Arguments:
      PATTERN   Pattern specifying a help topic

    Exit Status:
    Returns success unless PATTERN is not found or an invalid option is given.
```

Resim 18. Help Komutu.

Programların dokümanlarını görüntülemek için man komutu kullanılır. help komutu kabuk yerleşik komutları için yardım bilgisi sunarken man, programların kullanma kılavuzlarını less ile gösterir. Programlar için man sayfaları, bölümler halinde tanımlanır. Bu bölümler Tablo 7.de verilmiştir. Sadece ilgili bölümü görüntülemek için "man bolum_numarası programadi" şeklinde komut çalıştırılmalıdır. Program kılavuzlarında belirli bölümler girilmiş olabilir. Bu durumda sadece kılavuzda yer alan bölümleri görüntüleyebilirsiniz.:

Tablo 7. man Dosyaları Bölümleri.

Bölüm No	Bölüm İçeriği
1	Kullanıcı komutları
2	Sistem çağrıları
3	Kütüphane çağrıları
4	Özel dosyalar
5	Dosya biçimleri ve dosya dönüşümleri
6	Oyunlar
7	Çeşitli bilgiler
8	Sistem yönetim komutları
9	Çekirdek belgeleri

info komutu ile man komutu aynıdır. Ancak info komutu named programı ile bilgileri gösterir. Info dosyaları birbirlerine linklenmiştir. Bu linkler arasında dolaşabilirsiniz.

whatis komutu, programın adını ve bir satırlık açıklamasını gösterir. Resim 19.da "whatis pwd" komutu ile pwd komutunun kısa açıklaması gösterilmektedir:

```
kaynak@kaynak-VirtualBox:~$ whatis pwd
pwd (1)              - print name of current/working directory
```

Resim 19. whatis Komutu.

apropos komutu, komutların man sayfalarının içinde arama yaparak yazdığınız ifadeyi içeren komutları kısa bilgileri ile listeler. Listede ilk sütun ilgili komutun adını, ikinci sütun man sayfasındaki bölüm numarasını son sütun ise açıklamayı gösterir. Resim 20.de "apropos pwd" komutunun örneği verilmiştir. man komutu -k parametresi ile kullanılırsa aynı görevi görür:

```
kaynak@kaynak-VirtualBox:~$ apropos pwd
pwd (1)              - print name of current/working directory
pwdx (1)             - report current working directory of a process
unix_chkpwd (8)      - Helper binary that verifies the password of the curre...
```

Resim 20. apropos Komutu.

Kabukta birden fazla komutu tek bir satırda işletmeniz mümkündür. Komutlar arasına noktalı virgül koyarak tek bir satırda yazabilirsiniz. Bazen birden fazla komutun tek bir komut ile çağrılması istenebilir. Bu şekilde zamandan tasarruf edilir ve hata yapma olasılığı da azalmış olur. Örneğimizde kullanıcının özel verilerini silebileceği bir acil durum kodu oluşturalım. Kullanıcımız /home/kaynak/Resimler klasöründe özel resimlerini saklıyor olsun ve /home/kaynak/Wallets klasöründe de dijital cüzdanlarının özel anahtarlarını saklıyor olsun. Tek bir komut ile tüm bu dizinlerdeki kritik verilerini silmek için "rm -rf /home/kaynak/Resimler; rm -rf /home/kaynak/Wallets" komutunu çalıştırmalı. Ekrana da bir mesaj basmak için sonuna "echo 'İşlem tamam!'" komutu yazılmalıdır. Bu komutları "acildurum" komutuna bağlamak istersek alias komutunu kullanabiliriz. "alias acildurum=' rm -rf /home/kaynak/Resimler; rm -rf /home/kaynak/Wallets;echo 'İşlem tamam!'"" komutu ile tanımlama kolayca yapılabilir. "acildurum" komutu çalıştırıldığı anda yazılan tüm komutlar çalıştırılacaktır. Bu kısayolu silmek için ise "unalias acildurum" komutunu yazmak gerekir.

Anahtar Kavram
İşlemci: Bilgisayarda verilen komutları yorumlayan ve işlemleri yürüten birim.

6.2.3. Proses Yönetimi

Uygulamalar işletim sisteminde çalıştırılmak istendiğinde prosesler ile sistemde temsil edilirler ve işlemcide sırayla çalıştırılırlar. Linux çekirdeği, bu

proseslerin yönetiminden sorumludur. İşletim sistemi prosesleri oluştururken bir id verir ve bu id numarası ile takip eder. Linuxte proses yönetimi için aşağıdaki komutları kullanabilirsiniz:

Şekil 15. Linuxte Proses Yönetimi İçin Kullanılan Komutlar

ps komutu, o anda çalışmakta olan proseslerin durumunu ekrana getirir. Terminale ps komutu yazıldığı zaman Resim 21.de görüldüğü gibi iki adet prosesin çalışmaktadır. bash, komutu çalıştıran kabuk uygulamasıdır. ps ise şu anda çalıştırdığımız ve proseslerin durumunu bize raporlamakta olan uygulamadır. Bu proseslerin birer PID değerlerinin olduğu görülmektedir. Bu proses idler ile proseslere müdahale edilebilir. Terminalden ps komutu çalıştırıldığı için sadece mevcut terminalin çalıştırmakta olduğu prosesler listelenir. Tüm sistemin prosesleri hakkında bilgi almak için "ps x" komutu kullanılabilir. "ps aux" komutu ile de daha detaylı bilgiler alınabilir.

```
kaynak@kaynak-VirtualBox:~$ ps
    PID TTY          TIME CMD
 157813 pts/0    00:00:00 bash
 188388 pts/0    00:00:00 ps
```

Resim 21. ps Komutu

top komutu ile proseslerin durumu canlı bir şekilde izlenebilir. ps komutunda bir anın durumu gösterilirken top komutunda ekran sürekli yenilenir. Bu şekilde anlık kaynak tüketimi kolaylıkla izlenebilir. Resim 22.de top komutunun bir ekran görüntüsü gösterilmektedir:

```
top - 22:51:10 up 8 days, 22:29,  2 users,  load average: 0,00, 0,04, 0,01
Tasks: 192 total,   1 running, 190 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   1985,3 total,    674,6 free,    477,7 used,    833,0 buff/cache
MiB Swap:    448,5 total,    129,8 free,    318,7 used.   1347,4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  12444 kaynak    20   0 3672580 219348  66540 S   0,7  10,8   4:50.66 gnome-+
 188472 kaynak    20   0   14604   3784   3268 R   0,3   0,2   0:00.01 top
      1 root      20   0  169176   8612   5660 S   0,0   0,4   0:19.73 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.03 kthrea+
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_pa+
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworke+
      8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_per+
      9 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_ta+
     10 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_ta+
     11 root      20   0       0      0      0 S   0,0   0,0   0:17.39 ksofti+
     12 root      20   0       0      0      0 I   0,0   0,0   0:10.84 rcu_sc+
     13 root      rt   0       0      0      0 S   0,0   0,0   0:09.87 migrat+
     14 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_i+
     16 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
     17 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kdevtm+
     18 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
     19 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 inet_f+
     20 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kauditd
     21 root      20   0       0      0      0 S   0,0   0,0   0:00.49 khungt+
     22 root      20   0       0      0      0 S   0,0   0,0   0:00.00 oom_re+
     23 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 writeb+
```

Resim 22. top Komutu

Terminalden gedit komutunu çalıştırdığınızda ekranda yeni bir pencerede gedit programının açıldığını görürsünüz. Bu durumda uygulama yeni bir proses ile çalışmaktadır ve terminal ekranında komutunuza bir dönüş olmaz. Terminal, gedit programının kapanmasını beklemektedir. gedit programının penceresini kapatırsanız terminal tekrar eski haline dönecek ve komut girişi yapabileceksiniz. Terminalde çalışmakta olan programları dilerseniz Ctrl+C tuşlarına basarak

da sonlandırabilirsiniz. Programı sonlandırmadan terminalde nasıl işlem yapabiliriz? Bunun için programı arka planda başlatmalıyız. Bir programı arka planda başlatabilmek için ilgili komutun sonuna & işareti eklemeliyiz. Resim 23.te gedit & işareti ile arka planda çalıştırılmış ve ekranda [2] ve 188646 ifadeleri basılmıştır. [2], ilgili işin kaç numaralı iş olduğunu göstermektedir. Kabuk, işleri bizim için arka planda yönetmektedir ve bunları numaralandırmaktadır. Şu anda çalıştığımız terminalde yapılan her iş için bir iş numarası atanır. 188646 ise işletim sisteminin bu prosesi başlatırken atadığı proses idsidir. Resim 24.te görüldüğü gibi, ps komutu ile bakıldığında ilgili prosesin hala çalışmakta olduğunu görebiliriz. jobs komutu ile mevcut terminalden çalıştırılan işlerin listesini görüntüleyebiliriz. Resim 25.te jobs komutu ile terminalde 1 numaralı işin durduğunu ve 2 numaralı işin hala çalışmakta olduğunu görüntüleyebilmekteyiz:

```
kaynak@kaynak-VirtualBox:~$ gedit&
[2] 188646
kaynak@kaynak-VirtualBox:~$
```

Resim 23. Programı Arka Planda Çalıştırma

```
kaynak@kaynak-VirtualBox:~$ ps
    PID TTY          TIME CMD
 157813 pts/0    00:00:00 bash
 188461 pts/0    00:00:00 top
 188646 pts/0    00:00:00 gedit
 188720 pts/0    00:00:00 ps
```

Resim 24. Çalışmakta Olan Proseslerin Durumu

```
kaynak@kaynak-VirtualBox:~$ jobs
[1]+  Durdu                 top
[2]-  Çalışıyor             gedit &
```

Resim 25. Terminaldeki İşlerin Durumu

Terminalde arka planda çalışmakta olan bir işe geri dönebilmek için fg (foreground) komutu kullanılır. fg komutu parametre olarak % işareti ile işin numarasını almaktadır. Çalışmakta olan 2 numaralı bir işi tekrar terminalde ön plana almak için "fg %2" komutu kullanılmalıdır. Bazen ön planda çalışan bir uygulamayı sonlandırmadan durdurmak (bekletmek) isteyebiliriz. Bu durumda klavyeden Ctrl+Z tuşlarına basarak ilgili prosesi beklemeye alırız. Beklemede olan bir prosesi ise fg komutu ile ön planda çalıştırmaya devam ettirebiliriz. Dilerseniz bg komutu ile de proses arka planda çalışmaya devam edebilir. Hatırlarsanız arka planda bir uygulamayı çalıştırmak için komutun sonuna & işareti koyuyorduk. bg komutu da bu şekilde aynı işlevi gerçekleştirir.

Proseslere bir sinyal yollamak için kill komutu kullanılır. kill komutu bir parametre olarak sinyal adı veya numarası bekler ve opsiyoneldir. Parametreden sonra proses idsi verilmesi gerekir. Sinyal parametresi verilmediği zaman varsayılan olarak TERM sinyali gönderilir. TERM sinyali ilgili prosesi sonlandırır. Bazı sinyal türleri Tablo 8.de görülmektedir.

Tablo 8. Sinyal Türleri.

NUMARA	AD	AÇIKLAMA
2	INT	***Interrupt.*** Prosese sonlandırma sinyali gönderir. Ctrl+C ile aynı görevi görür.
9	KILL	Kill. Prosese hiçbir sinyal gönderilmeden doğrudan çekirdek tarafından proses sonlandırılır. Program cevap vermiyorsa kullanılmalıdır.
15	TERM	Terminate. Prosese bir sonlandırma sinyali gönderir ve bu şekilde sonlandırır. Çekirdek sonlandırma işleminden önce prosese bilgi verir.
18	CONT	Continue. STOP sinyali ile durdurulan bir prosesin devam etmesini sağlar.
19	STOP	Stop. Prosesin bekletilmesini sağlar. Proses bundan haberdar olmaz.

6.3. Kabuk Programlama

Kabuk, yazılan komutları sırası ile işletir. Bash programı bu kabuklardan birisidir. Bash üzerinde satırlar halinde işletilen komutları bir dosya içinde düzenli bir şekilde yazarak betikler oluşturmak mümkündür. Bu başlık altında bash isimli kabukta çalıştırmak üzere betik programlamayı öğreneceğiz. Değişkenler, operatörler, koşullu ifadeler, döngüler gibi temel kavramları öğreneceğiz. Linux dışındaki diğer işletim sistemlerinde de farklı kabuklarda çalıştırılmak üzere programlar yazabilirsiniz. Bu kabukların kendi yazım standartlarını öğrenmeniz gerekir.

Herhangi bir text editör ile script (betik) dosyası oluşturabilirsiniz. Linuxte script dosyasının uzantısı sh olmalıdır. Dosya oluşturulduktan sonra chmod +x komutu ile çalıştırılabilir olarak izin verilmelidir. Aşağıda gösterilen örnek kod parçasını merhaba.sh isimli bir dosyaya kaydederek ilk script dosyamızı oluşturabiliriz. Çalıştırmak için komut satırına ./merhaba.sh yazmamız gerekir. Bu script "#!/bin/bash" ile başlatılmıştır. Bu satır, scriptin hangi kabuk ile yorumlanacağını gösterir.

```
#!/bin/bash
echo "Merhaba!"
```

6.3.1. Değişkenler

Bash ile değişken tanımlama oldukça kolaydır. Tip belirtmenize gerek yoktur.

ders="İşletim Sistemleri" şeklinde bir ders isimli değişkeni tanımlayabilirsiniz. Bu değişkeni kullanabilmek için $ işareti kullanılmalıdır. Ekrana bu değişken içindeki değeri yazdırabilmek için echo $ders komutu kullanılmalıdır. Aşağıdaki script ile örneklendirilmiştir.

```
#!/bin/bash
ders='İşletim Sistemleri'
echo $ders
```

Değişkenler tanımlandıktan sonra sadece okunabilir olarak işaretlenebilir. Bunun için readonly komutunu kullanmalısınız. readonly ders komutundan sonra bu değişkene tekrar bir değer ataması yapamazsınız. Değişkenleri bellekten silmek için ise unset komutu kullanılmalıdır. unset ders komutu ile ders değişkenini bellekten silebilirsiniz.

6.3.2. Operatörler

Operatörler aritmetik, ilişkisel, mantıksal, sting ve dosya test olarak ayrılabilir. Bu operatörlere ve anlamlarına kısaca bakalım. Örneklerde kullanılan değişkenler x = 5 ve y = 10 olarak tanımlanmıştır.

Tablo 9. Aritmetik Operatörler

Operatör	Tanım	Örnek
+ (Toplama)	Her iki tarafında bulunan değerler için toplama işlemi yapar.	'expr $x + $y' 15 değerini döndürür.
- (Çıkarma)	Her iki tarafında bulunan değerler için çıkarma işlemi yapar.	'expr $x - $y' -5 değerini döndürür.
* (Çarpma)	Her iki tarafında bulunan değerler için çarpma işlemi yapar.	'expr $x * $y' 50 değerini döndürür.
/ (Bölme)	Sol tarafında bulunan değeri sağ tarafındaki değerle böler.	'expr $y / $x' 2 değerini döndürür.
% (Mod)	Sol tarafındaki değeri sağ tarafındaki değer ile böler ve kalanı geri döndürür.	'expr $y % $x' 0 değerini döndürür.
= (Atama)	Sağ tarafta bulunan değeri sol tarafındaki değişkene atar.	x=$y y'nin değerini x değişkenine atar.
== (Eşitlik)	Her iki tarafta bulunan değerleri karşılaştırır. Her iki taraf aynı ise true, değil ise false çıktısını verir.	[$x == $y] false değerini döndürür.
!= (Eşit değil)	Her iki taraftaki değerleri karşılaştırır. Ancak değerler farklı ise true değerini döndürür.	[$x != $y] true değerini döndürür.

Tablo 10. İlişkisel Operatörler

Operatör	Tanım	Örnek
-eq	İki operantın eşitliğini kontrol eder. Eşit ise true değerini, değil ise false değerini döndürür.	[$x -eq $y] false değerini döndürür.
-ne	İki operantın eşitliğini kontrol eder. Eşit değil ise true değerini döndürür.	[$x -ne $y] true değerini döndürür.
-gt	Sol operantın değerinin sağ operantın değerinden büyük olup olmadığını kontrol eder. Büyük ise true değerini döndürür.	[$x -gt $y] false değerini döndürür.
-lt	Sol operantın değerinin sağ operantın değerinden küçük olup olmadığını kontrol eder. Küçük ise true değerini döndürür.	[$x -lt $y] true değerini döndürür.
-ge	Sol operantın değerinin sağ operantın değerinden büyük ya da eşit olup olmadığını kontrol eder. Büyük ise true değerini döndürür.	[$x -ge $y] false değerini döndürür.
-le	Sol operantın değerinin sağ operantın değerinden küçük ya da eşit olup olmadığını kontrol eder. Küçük ise true değerini döndürür.	[$x -le $y] true değerini döndürür.

Tablo 11. Mantıksal Operatörler

Operatör	Tanım	Örnek
!	Mantıksal ifadenin tersini alır.	[! false] true değerini döndürür.
-o	Mantıksal OR (ya da) ifadesidir. Operantlardan birisi true ise true değerini döndürür.	[$x -lt 10 -o $y -gt 50] true değerini döndürür.
-a	Mantıksal AND (ve) ifadesidir. Operantlardan her ikisi de true ise true değerini döndürür.	[$x -lt 10 -a $y -gt 50] false değerini döndürür.

6.3.3. Koşullu İfadeler

Bir şarta bağlı olarak program işletmek için koşullu ifadelere ihtiyaç duyarız. Kod içinde kontrolleri yaparak ilgili kod parçasının çalışmasını veya çalışmamasını sağlayabiliriz. Bash scriptlerde diğer programlama dillerinde olduğu gibi yine if/else ifadesi vardır. Bash ile if ifadesinden sonra test etmek istediğiniz koşulu köşeli parantez içinde vermeniz gerekir. if deyiminin içine aldığı kod parçasının bitiminde fi ile if bloğunu sonlandırabilirsiniz. Diğer durumları ifade eden else if deyimini burada elif ile belirtebilirsiniz. Aşağıdaki örnek kod parçasında örnekler görülebilir. Bu script çalıştırıldığı zaman ekrana "18 yaşında!" ifadesi yazılacaktır.

```
#!/bin/bash
yas=18

if [ $yas -gt 18 ]
then
        echo "18 den büyük!"
elif [ $yas -lt 18 ]
then
```

```
        echo “18 den küçük!”
else
        echo “18 yaşında!”
fi
```

6.3.4. Döngüler

Komutların tekrarlı olarak çalıştırılmasını döngüler sayesinde gerçekleştirebiliriz. Bu döngülerden birkaçı olan for, while, until döngülerini inceleyelim. Bu döngülerin iç içe kullanılabileceğini unutmayalım.

For döngüsü ile bir liste içindeki elemanların her birisi için komutlarınız tekrar edilir. Her iterasyonda belirtilen değişkene sıradaki değer atanır ve kod içinde kullanılabilir. Aşağıdaki for döngüsünde 1’den 5’e kadar sayılar ekrana yazılır:

```
#!/bin/bash
for i in 1 2 3 4 5
do
  echo “Sıradaki: $i”
done
```

Aynı kod parçasını aşağıdaki şekilde de yazabiliriz. Tek tek sayıları yazmak yerine aralığı belirterek de sayı listesini oluşturabiliriz:

```
#!/bin/bash
for i in {1..5}
do
  echo “Sıradaki: $i”
done
```

Aşağıdaki kod parçasında ise i değerinin birer birer arttırılarak döngünün ilerletildiğini görebiliriz:

```
#!/bin/bash
for (( i=1; i<=5; i++ ))
do
  echo “Sıradaki: $i”
done
```

While döngüsü ile belirtilen şart sağlandığı sürece döngü devam eder. Aşağıdaki kod parçasında 1’den 5’e kadar sayılar ekrana yazılmıştır. Döngünün her iterasyonunda şart tekrar tekrar kontrol edilir:

```
#!/bin/bash
sayi=1
while [ $myvar -ne 5 ]
do
   echo $sayi
```

```
    sayi=$(( $sayi + 1 ))
done
```

Until ise while döngüsünün tam tersi olarak düşünülebilir. Şart sağlandığı takdirde döngü duracaktır. Burada sayi değişkeni 5'e eşit olana dek döngü devam eder:

```
#!/bin/bash
sayi=1
until [ $myvar -eq 5 ]
do
    echo $sayi
    sayi=$(( $sayi + 1 ))
done
```

Bu Bölümde Ne Öğrendik?

* İşletim sistemlerinde kullanıcı çekirdeğe doğrudan komut gönderememektedir. Çekirdeğe ancak bir arayüz kullanarak veya bir kabuk programı kullanarak komut gönderilebilir.

    * İşletim sistemlerinden Linux için geliştirilen Bash kabuk programını kullanarak işletim sistemi arayüz olmadan yönetilebilmektedir. Ubuntu işletim sisteminde Bash terminaline ulaşmak için Ctrl+Alt+T kısayolu kullanılabilir.

* Linux'de her şey bir dosya olarak temsil edildiğinden ötürü dizinlerin yapısını anlamak ve hızlıca dolaşabilmek gerekli olan temel konutları bilmek gerekmektedir. Bunlar; bilgi almayı sağlayan (pwd), dizinler arasında dolaşmayı sağlayan (cd), listeleme yapan (ls) komutlarından oluşmaktadır.

    * Dizin oluşturma, taşıma, dosya kopyalama, taşıma, silme gibi dizin ve dosya değişiklikleri için de bazı komutları bilmek gerekmektedir. Dosya ve dizinlerin kopyalanması (cp), taşınması (mv), yeni dizin oluşturulması (mkdir), silinmesi (rm), link oluşturulması (ln) için gerekli olan komutlardır.

* Kabuk komutları hakkında bilgi alınabilecek çeşitli yardımcı komutlar bulunmaktadır. Bunlar; komutun nasıl yorumlandığı hakkında bilgi veren (type), program dosyasının tam konumunu veren (which), kabuk komutları kullanımı için yardım bilgisi gösteren (help), bir komutun kullanımı hakkında detaylı bilgi veren (man), bir komut hakkında yazılmış kullanım bilgisini gösteren (info), komut hakkında çok kısa bilgi veren (whatis), belirtilen komuta benzer komutları listeleyen (apropos) ve bir komut için kısaltma tanımlamak için kullanılan (alias) komutlarından oluşmaktadır.

    * Linux işletim sistemlerinde proseslerin listelenmesini sağlayan komutlar bulunmaktadır. Çalışmakta olan proseslerin anlık durumunu raporlayan (ps) komutu ve çalışmakta olan proseslerin durumunu izlemek için kullanılan (top) komutu önemli komutları oluşturmaktadır.

* Jobs komutu terminalde çalışan işlerin listelenmesi için çalıştırılabilecek komut iken; bg komutu arka planda çalışan işleri listelemek için kullanılmakta fvg komutu ise ön planda çalışan işleri listelemekte kullanılmaktadır.

    * Proseslere çeşitli sinyaller göndermek mümkündür. Kill komutu prosese idsi ile sinyal göndermekte; killall komutu ise proses ağacına ismi ile sinyal göndermektedir.

* Kabuk, yazılan komutları sırası ile işletir. Bash programı bu kabuklardan birisidir. Bash üzerinde satırlar halinde işletilen komutları bir dosya içinde düzenli bir şekilde yazarak betikler oluşturmak mümkündür.

    * Bash programı üzerinde satırlar halinde işletilen komutları bir dosya içinde düzenli bir şekilde yazarak betikler oluşturulabilmektedir. İşletim sistemlerinde birtakım işleri betikler ile tek bir dosya içinde hazırlamak mümkündür.

7. SUNUCU YAZILIMLARI VE PAKET YÖNETİMİ

Eski günlerde insanlar posta arabalarını soydular ve zırhlı kamyonları devirdiler. Şimdi sunucuları yıkıyorlar.

Richard Power

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Linux ve Windows sunucular hakkında bilgi sahibi olabilir.
- Paket yönetimi kavramını öğrenebilir.
- Paket yönetimi kapsamında gerçekleştirilen ortak işlevleri öğrenebilir.
- Farklı Linux dağıtımları üzerinde temel paket yönetimi işlemlerini gerçekleştirebilir.
- Windows işletim sistemlerinde paket yönetiminin nasıl gerçekleştirildiğini öğrenebilir.

Başlamadan Önce

Bu bölümde Linux ve Windows sunucu işletim sistemleri hakkında kısa bilgiler verildikten sonra, en çok kullanılan iki sunucu işletim sistemlerinde temel sunucu yönetiminin nasıl yapıldığı açıklanacaktır. Linux sunucuların çok yaygın olması ve kullanıcı arayüzüne sahip olmaması dolayısıyla Linux'de farklı tiplerde sunucu yazılımlarının kurulum örnekleri verilecektir. Paket yönetimi kavramı Linux ve Windows işletim sistemleri temel alınarak anlatılacak; paket yönetimini gerçekleştirebilmek için yaygın olarak kullanılan paket yöneticilerinin komutları verilecektir.

Birlikte Düşünelim

1. Sunucular ile diğer bilgisayarların farkları nelerdir?
2. Sunucularda yazılımlar otomatik nasıl başlatılırlar ve hata olduğunda ne yapılır?
3. Linux ve Windows sunucu işletim sistemlerinin farkları nelerdir?
4. Paket yönetim sistemlerinin kullanıcıya sunduğu avantajlar nelerdir?

7.1. Linux Sunucuları

Okuma Önerisi
Linux İşletim Sistemine Karşı Tutum konulu makaleyi çevrim içi olarak aşağıdaki bağlantıdan okuyabilirsiniz.

Sunucu bilgisayarlarda kullanılan işletim sistemleri çoğunlukla Unix ve türevi işletim sistemleridir. 25 Ağustos 2021 tarihli W3Techs (https://w3techs.com/technologies/history_overview/operating_system/ms/y) verisine göre web sayfalarını sunan bilgisayarların %77.2'si Unix ve türevi, %22.9'u ise Windows işletim sistemine sahiptir. Linux işletim sistemleri her ne kadar kişisel bilgisayarlarda tercih edilmiyor olsa da sunucularda çok yüksek oranda tercih edilmektedir. Bunun sebebi, Linux işletim sistemlerinin stabil olması, daha az kaynak tüketiyor olması, açık kaynak olması, çoğunlukla lisans ücreti gerektirmiyor olması, güvenlik açısından diğer işletim sistemlerine göre kontrolünün daha kolay olması olarak sıralanabilir.

Sunucular için geliştirilen Linux dağıtımları aslında yine aynı Linux çekirdeğini kullanırlar. Bu dağıtımların farklı olan tarafı, sunucu için özelleşmiş paketleri sunması ve mümkün olduğunca gereksiz paketlerden arınmış olmasıdır. Sunucular çoğunlukla ekransız (headless) çalışırlar. Sunucu yönetimi, uzak bağlantılar ile gerçekleştirilir. Bu sebeple bir masaüstü arayüz yazılımına ihtiyaç yoktur. Sunucuda gereksiz olan tüm yazılımların arındırılmış olması güvenlik ve kaynak yönetimi açısından çok önemlidir.

Sunucular, web, mail, veri tabanı, dns gibi çeşitli amaçlarda hizmet verebilirler. Sunucuların diğer bilgisayarlardan farkı, sürekli olarak çalışması ve her an için hizmete hazır olmasıdır. Çok büyük bir kitleye hizmeti kesintisiz bir şekilde vermesi istenir. Bu başlık altında systemd ve sunucu yazılımları anlatılmaktadır. Debian tabanlı Ubuntu Server 20.04 LTS versiyonu kullanılarak örnekler verilmiştir.

7.1.1. Linux Sistemleri

Systemd, işletim sistemi başladığında başlatılan ilk işlemdir. PID'si 1'dir. Bilgisayar kapanıncaya kadar bu işlem açık kalır. İşletim sistemi başlatıldığında çalışması gereken servisler systemd tarafından otomatik başlatılır. Gerektiğinde durdurulur veya servisler yeniden başlatılır. Systemd bu servislerin loglarını toplar. Özet olarak sistemin ve servisleri organize eden bir programdır. Systemd, ps komutu ile doğrudan görüntülenemez. Ancak "ps -ef" komutu ile sistemde çalışan tüm prosesler listelenebilir. Resim 26.'da systemd'nin 1 numaralı PID'sine sahip olduğu görüntülenebilir. Resim 27.de pstree komutu ile kök prosesin systemd olduğunu ve diğer tüm proseslerin systemd tarafından başlatıldığını görebiliyoruz.

Ubuntu ve birçok dağıtım, eskiden init kullanırken artık init yerine systemd kullanmayı tercih etmektedir. Bu geçişin birçok sebebi vardır. Init, servisleri sırayla başlatır ancak systemd servisleri paralel başlatabilir. Paralel yürütme kabiliyeti, çok çekirdekli işlemcilerde oldukça önemlidir. Systemd, donanımda gerçekleşen değişiklikleri anlık olarak algılayabilir. Systemd, ön yükleme esnasında servisleri hızlı başlatılması için servisleri bağımlılıklarına göre sıraya koyabilir. Systemd ile

servisler otomatik olarak yeniden başlatılabilir. Bir servis herhangi bir nedenden dolayı sonlanırsa systemd sonlanan servisi yeniden başlatabilir. Bu sebeplerden dolayı systemd init'e göre tercih edilmektedir. Systemd kullanırken servisleri systemctl komutu ile servisler yönetilirken init ile service komutu ile yönetilir.

```
kaynak@kaynak-VirtualBox:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Ağu11 ?        00:00:24 /lib/systemd/systemd --syst
root           2       0  0 Ağu11 ?        00:00:00 [kthreadd]
root           3       2  0 Ağu11 ?        00:00:00 [rcu_gp]
root           4       2  0 Ağu11 ?        00:00:00 [rcu_par_gp]
root           6       2  0 Ağu11 ?        00:00:00 [kworker/0:0H-events_highpr
root           8       2  0 Ağu11 ?        00:00:00 [mm_percpu_wq]
root           9       2  0 Ağu11 ?        00:00:00 [rcu_tasks_rude_]
root          10       2  0 Ağu11 ?        00:00:00 [rcu_tasks_trace]
root          11       2  0 Ağu11 ?        00:00:24 [ksoftirqd/0]
root          12       2  0 Ağu11 ?        00:00:16 [rcu_sched]
root          13       2  0 Ağu11 ?        00:00:16 [migration/0]
root          14       2  0 Ağu11 ?        00:00:00 [idle_inject/0]
root          16       2  0 Ağu11 ?        00:00:00 [cpuhp/0]
```

Resim 26. Systemd PID

```
kaynak@kaynak-VirtualBox:~$ pstree -n
systemd─┬─systemd-udevd
        ├─acpid
        ├─cron
        ├─dbus-daemon
        ├─NetworkManager───2*[{NetworkManager}]
        ├─networkd-dispat
        ├─rsyslogd───3*[{rsyslogd}]
        ├─switcheroo-cont───2*[{switcheroo-cont}]
        ├─systemd-logind
        ├─udisksd───4*[{udisksd}]
        ├─gdm3─┬─2*[{gdm3}]
        │      └─gdm-session-wor─┬─2*[{gdm-session-wor}]
        │                        ├─gdm-x-session─┬─2*[{gdm-x-session}]
        │                        │               ├─Xorg───{Xorg}
        │                        │               └─gnome-session-b─┬─ssh-agent
        │                        │                                 └─2*[{gnome+
        │                        └─2*[gdm-session-wor───2*[{gdm-session-wor}]]
        ├─unattended-upgr───{unattended-upgr}
        ├─2*[kerneloops]
        ├─rtkit-daemon───2*[{rtkit-daemon}]
        ├─upowerd───2*[{upowerd}]
        ├─colord───2*[{colord}]
        ├─systemd─┬─(sd-pam)
        │         ├─pulseaudio───3*[{pulseaudio}]
        │         ├─tracker-miner-f───4*[{tracker-miner-f}]
        │         ├─dbus-daemon
        │         ├─gvfsd─┬─2*[{gvfsd}]
```

Resim 27. Systemd pstree

7.1.1.1. Linux Servis Yazımı

Linux systemd ile bir servis tanımlayalım ve bu servisi yönetelim. Bu işlemi gerçekleştirebilmek için yetki seviyenizi superuser konumuna getirmelisiniz. Komutlarınızı sudo ile yazdığınızda bu yetkiye sahip olacaksınız. /etc/systemd/system/ dizininde yeni bir dosya oluşturmalıyız. Servis, belirli aralıklarla ekrana "Merhaba ben servis!" yazısını yazsın. Servis adı olarak merhaba koyalım. Bunun için belirttiğimiz dizinde merhaba.service isimli dosyayı oluşturalım. nano veya benzer bir yazı editöründe bu dosyayı oluşturabilir ve düzenleyebilirsiniz.

Systemd servislerinin (unit) tanımları, dosya içindeki başlıkların altında bölümler içinde tanımlanmalıdır. Her bir bölüm köşeli parantezler içinde tanımlanır. [Unit] bölümü içinde servise ait bir takım bilgiler tanımlanır. Bu bilgilerden birkaçı aşağıda örnek olarak verilmiştir:

- Description: Servisin kısa bir tanımını içerir.
- Documentation : Servisin dökümantasyon adresleri tanımlanır.
- Requires: Servisin gereksinim duyduğu bileşenleri tanımlar. Bileşenler aktif olduğunda servis başlatılır. İhtiyaç duyulan bir öğe veya öğeler başarılı bir şekilde başlatılamamışsa bu servis de başlatılamaz.
- After: Servisin hangi servisten sonra başlayacağını belirtir.

[Install] bölümü içinde bir servisin kurulum anında enable veya disable edilirken kullanması gereken tanımları içerir. WantedBy tanımı ile servisin etkinleştirildiğinde hangi seviyede devreye gireceğini tanımlar. /etc/systemd/system klasöründe bir soft link oluşturarak başlangıçta başlatılmasını sağlar. Alias tanımı ile servisin adı tanımlanır.

Anahtar Kavram
Konfigürasyon: Bilgisayar sisteminin bilhassa fiziksel birimlerini gösterme, yapılandırma.

[Service] bölümü ile servisin çalışabilmesi için gereken konfigürasyonlar tanımlanır. Type alanı, prosesin nasıl çalışacağını ve tipini belirtir. Bunlar simple, forking, oneshot, dbus, notify ve idle değerleri olabilir. User alanı, servisin hangi kullanıcı ile çalışacağını, Group alanı, hangi grupta çalışacağını belirtir. ExecStart alanı servisin başlangıcında çalıştırmak istenen programın yolunu ve varsa parametrelerini içerir. ExecStop alanı ise servisin durdurulması durumunda çalıştırılacak komutları içerir.

Aşağıdaki komutlar ile merhaba isimli servisimizi oluşturalım:

```
cd /etc/systemd/system/
sudo nano merhaba.service
```

Aşağıdaki servis tanımı dosyasını merhaba.service dosyasının içine yazalım:

```
[Unit]
Description=Say Hello Program
After=network.target

[Service]
Type=simple
```

```
User=root
Restart=always
ExecStart=/usr/bin/sh /home/kaynak/merhaba_yaz.sh
ExecStop=/bin/kill ${MAINPID}
[Install]
WantedBy=multi-user.target
```

ExecStart içinde bu servisin /home/kaynak/merhaba_yaz.sh scriptini çalıştırdığını görüyoruz. Bu sebeple nano /home/kaynak/merhaba_yaz.sh komutu ile aşağıdaki bash scriptini dosyaya yazalım:

```
#!/bin/bash
echo “Merhaba Servisler!”
sleep 5
```

Bu script, ekrana bir yazı yazacak ve 5 saniye boyunca uyuyacaktır. 5 saniye sonunda program sonlanacaktır. Script dosyasının çalıştırılabilir olması için sudo chmod +x /home/kaynak/merhaba_yaz.sh komutunu çalıştırmalısınız.

Tüm bu hazırlıklardan sonra systemd için hazırladığımız tanımları systemd'ye tanıtmamız gerekli. Bilgisayarı yeniden başlatmadan sudo systemctl daemon-reload komutu ile servis dosyalarında yapılan değişiklikler tanıtılır. Yeni hazırladığımız servisi başlangıçta çalışması için enable etmemiz gerekecektir. sudo systemctl enable merhaba komutu, servisimizi enable edecektir. Dilerseniz sudo systemctl enable --now merhaba komutu ile servisi hemen başlatabilirsiniz. Servisin durumunu systemctl status merhaba komutu ile görebilirsiniz. Resim 28.de merhaba servisinin durumu çalışır olarak görüntülenmektedir:

```
● merhaba.service - Merhaba Yaz
     Loaded: loaded (/etc/systemd/system/merhaba.service; enabled; vendor pres>
     Active: active (running) since Thu 2021-08-26 00:47:48 +03; 4s ago
   Main PID: 217385 (sh)
      Tasks: 2 (limit: 2314)
     Memory: 392.0K
     CGroup: /system.slice/merhaba.service
             ├─217385 /usr/bin/sh /home/kaynak/merhaba_yaz.sh
             └─217386 sleep 5

Ağu 26 00:47:48 kaynak-VirtualBox systemd[1]: Started Merhaba Yaz.
Ağu 26 00:47:48 kaynak-VirtualBox sh[217385]: Merhaba Servisler!
```

Resim 28. Servis durumu

7.1.2. Linux Sunucu Yazılımları

Linuxde en çok kullanılan sunucu yazılımlarından birkaçının kurulumu anlatılmaktadır. Ssh, web sunucusu, veri tabanı sunucusu gibi en çok ihtiyaç duyulan temel sunucu yazılımları bu başlık altında ele alınmaktadır. Yazılımlar Ubuntu üzerinden örneklendirildiği için apt paket yöneticisi kullanılmaktadır.

Paket yöneticisi ile kurulum yapmadan önce sudo apt update komutu ile repositorylerin güncellemeyi unutmayalım. Ayrıca sudo apt upgrade ile kurulu paketlerin ve sistemin güncellemelerini yapabilirsiniz.

7.1.2.1. SSH Sunucusu

Ssh sunucusu, iki bilgisayar arasında güvenli olmayan bağlantı ile üzerinden güvenli ve şifreli bir bağlantı kurulmasını sağlar. OpenSSH, ssh sunucusu veya istemcisi için geliştirilen ücretsiz ve açık kaynak bir yazılımdır. Ssh sunucusunun kurmak için aşağıdaki işlemleri konsoldan gerçekleştirmelisiniz:

Kurulum için;

sudo apt install openssh-server

Ssh servisinin systemd ile servis olarak başlatılması için;

sudo systemctl enable ssh

Servisin durumunu öğrenmek için;

sudo systemctl status ssh

ssh servisi çalışmıyor ise çalıştırmak için;

sudo systemctl start ssh

Ssh sunucusu 22 numaralı tcp portunu dinler. İstemcinin ssh sunucusuna bağlanabilmesi için güvenlik duvarından 22 numaralı portu açmalıyız. Eğer güvenlik duvarı aktif ve port kapalı ise aşağıdaki komutlarla ilgili portu açabilir ve durumunu görüntüleyebilirsiniz.

sudo ufw allow ssh

sudo ufw enable

sudo ufw status

Ssh sunucusuna istemciden bağlantı yapabilmek için bir ssh client yazılımına ihtiyacınız vardır. ssh komutu ile istemcinizde kurulu olup olmadığını anlayabilirsiniz. Çoğu dağıtımda kurulu gelir. Kurmak için aşağıdaki komutları kullanabilirsiniz. Windows ile putty yazılımını kullanarak da bağlantı yapabilirsiniz.

sudo apt install openssh-client

Ssh client ile sunucuya bağlantı yapabilmek için kullanıcı adını ve ilgili sunucunun ip adresine veya domainine ihtiyacımız vardır. Aşağıdaki komut ile bağlantı kurabilirsiniz;

ssh user@sunucu-ip-adresi

7.1.2.2. Web Sunucusu

Web uygulamalarını bir sunucuda çalıştırabilmek için bir web sunucusuna ihtiyacımız vardır. Kullanıcıların web uygulamasına yaptıkları her bir isteği yakalayıp ilgili web uygulamasına ileten ve web uygulamasından üretilen cevabın kullanıcıya döndürülmesini sağlayan yazılımlara web sunucusu yazılımları denilebilir. Web sunucusu yazılımları, işletim sisteminin ağ soketlerini dinleyerek

gelen istekleri yakalarlar. Piyasada çok fazla web sunucusu yazılımları vardır. Bunlara örnek olarak Apache, Nginx, Caddy, Lighttpd yazılımları verilebilir.

İşletim sistemlerinde bir portun sadece bir proses tarafından dinlenebileceğini unutmamalıyız. Bu örnekte nginx 80 numaralı tcp portunu dinleyecektir. Sistemde bu portu dinleyen başka bir proses var ise nginx başlatılamayacaktır. Bir portun kullanımda olup olmadığını, kullanımda ise hangi proses tarafından kullanıldığını aşağıdaki komut ile öğrenebilirsiniz. Aşağıdaki komut 22 numaralı portu kullanan prosesleri listelemektedir:

```
sudo lsod -i :22
```

Resim 29.da 22 numaralı portun 246795 PID'li sshd tarafından kullanıldığını görebiliyoruz.

```
kaynak@kaynak-VirtualBox:/etc/systemd/system$ sudo lsof -i :22
COMMAND    PID USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
sshd    246795 root    3u  IPv4 8684795      0t0  TCP *:ssh (LISTEN)
sshd    246795 root    4u  IPv6 8684806      0t0  TCP *:ssh (LISTEN)
```

Resim 29. Port Kullanımı

Nginx yazılımını kurmak için aşağıdaki komut kullanılır:

```
sudo apt install nginx
```

Güvenlik duvarında 80 numaralı portu açabiliriz veya nginx ile gelen uygulama profillerini güvenlik duvarına tanıtarak da port açabiliriz. Aşağıdaki komut ile Nginx'in sadece 80 numaralı portunu tanımlayan "Nginx HTTP" profilini güvenlik duvarında açalım.

```
sudo ufw allow 'Nginx HTTP'
```

İpucu
Güvenlik duvarı konusuna 8. bölümde yer verilecektir.

Güvenlik duvarında tanımlanan kuralların durumunu aşağıdaki komut ile görebilirsiniz:

```
sudo ufw status
```

Resim 30.da ilgili kuralın tanımlandığını ve tanımlanan kuralların listesini görüntüleyebilirsiniz:

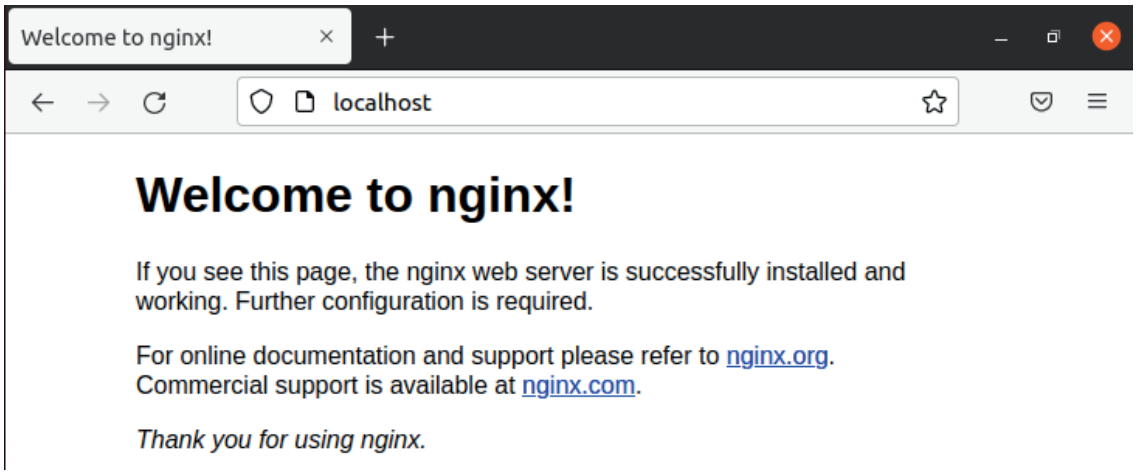
```
kaynak@kaynak-VirtualBox:/etc/systemd/system$ sudo ufw allow 'Nginx HTTP'
Kural eklendi
Kural eklendi (v6)
kaynak@kaynak-VirtualBox:/etc/systemd/system$ sudo ufw status
Durum: etkin

Alıcı                      Eylem       Gönderen
-----                      -----       --------
Nginx HTTP                 ALLOW       Anywhere
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

Resim 30. Güvenlik Duvarında Kural Tanımlama ve Durumu

Nginx web sunucusunun durumunu aşağıdaki komutla görebiliriz:

```
systemctl status nginx
```

Bir web tarayıcı kullanarak web sunucusuna bağlantı yapıp yazılımı test edebilirsiniz. Adres olarak sunucunun ip adresini yazmalısınız. Eğer aynı makine içinde bağlantı yapacak iseniz adres kısmına localhost yazarak da bağlantı sağlayabilirsiniz. İşletim sistemlerinde bulunan hosts dosyasında localhost alan adı 127.0.0.1 olarak tanımlanmıştır. 127.0.0.1, makinenin içeriden ulaşabileceğiniz ip adresidir. Resim 31.de nginx yazılımın çalıştığını görüntüleyebiliriz.



Resim 31. Nginx web sayfası

Nginx, varsayılan bir web sayfasını göstermektedir. Varsayılan ayarları dilerseniz /etc/nginx/nginx.conf dosyasından değiştirebilirsiniz. Detaylı bilgiye nginx dokümantasyonundan ulaşabilirsiniz.

7.1.2.3. Veri tabanı Sunucusu

Veri tabanı sunucuları uygulamalara veri saklama ve sorgulama hizmeti sunarlar. İlişkisel ve ilişkisel olmayan veri tabanları vardır. İlişkisel veri tabanlarına örnek olarak PostgreSQL, MySQL, Oracle, Microsoft SQL Server gibi veri tabanları verilebilir. İlişkisel olmayanlara ise MongoDB, Cassandra, Redis, HBase, Neo4j veri tabanı yazılımları örnek olarak verilebilir. Bu başlık altında ilişkisel veri tabanlarından PostgreSQL veri tabanı sunucusunun kurulumundan kısaca bahsedeceğiz.

PostgreSQL veri tabanını kurmak için aşağıdaki komutu kullanabilirsiniz:

```
sudo apt install postgresql postgresql-contrib
```

PostgreSQL veri tabanının gerçek bir sistemde verimli bir şekilde kullanılabilmesi için varsayılan ayarların yeterli olmayacağını unutmamalıyız. İlgili uygulamaya göre ve ilgili makinenin özelliklerine göre mutlaka ayarların bilinçli bir şekilde yapılması gerekmektedir. Aksi taktirde kaynaklarınızı verimli bir şekilde kullanamayabilirsiniz.

PostgreSQL kurulumu, postgres isimli bir kullanıcı hesabı ile yazılımı çalıştırmak üzere kurulum yapar. PostgreSQL'i kullanabilmek için bu hesap ile komutları çalıştırmak gerekecektir. Aşağıdaki komut ile postgres kullanıcısı olarak işlemlere devam edebiliriz.

```
sudo -i -u postgres
```

PostgreSQL veri tabanını kontrol edebilmek için gerekli olan psql istemcisine psql komutu ile erişebilirsiniz. Bu noktadan sonra artık veri tabanı komutları ile işlemlerinizi yapabilirsiniz. Örnek olarak yeni bir veri tabanı oluşturmak için aşağıdaki komutu kullanabilirsiniz:

```
create database ornek_db;
```

Psql istemcisini kullanmadan da aşağıdaki komut ile veri tabanı oluşturmak mümkündür:

```
createdb ornek_db
```

7.2. Windows Sunucuları

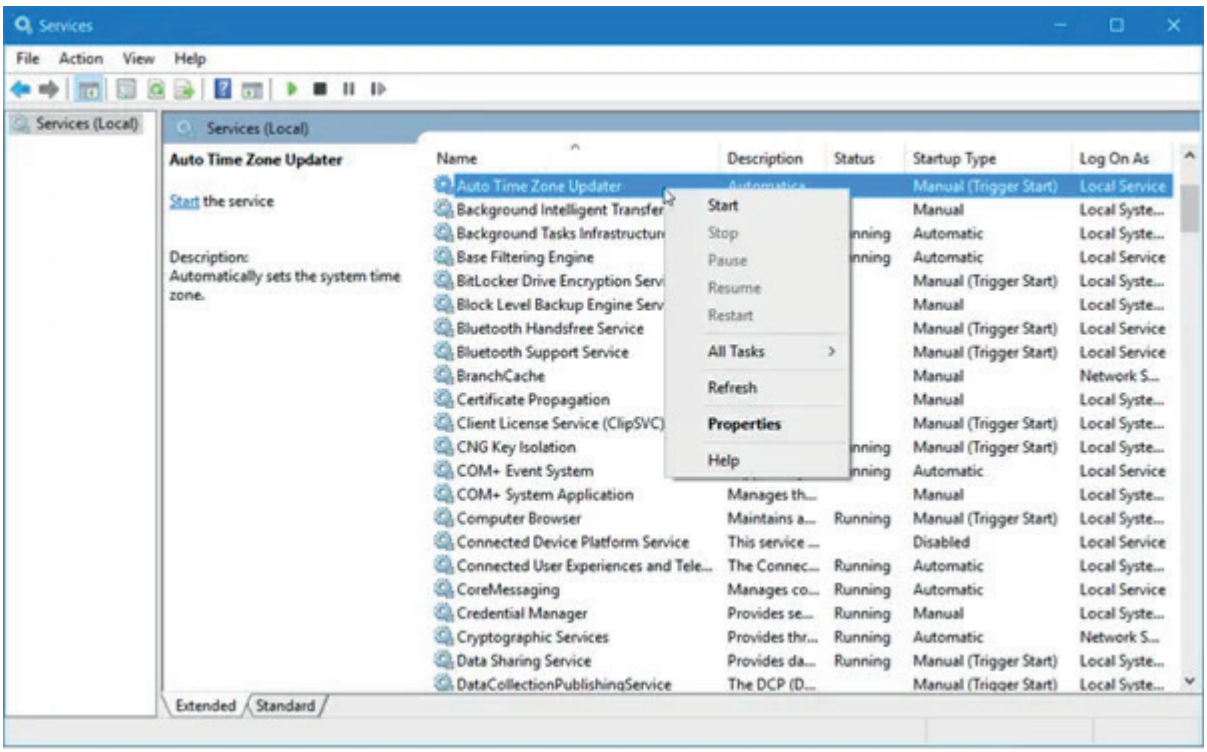
Windows sunucu işletim sistemleri, Linux işletim sistemlerinden sonra en çok kullanılan ikinci işletim sistemidir. Windows sunucu işletim sistemlerinin Linux'e göre çeşitli avantajları ve dezavantajları vardır. Avantajları ve dezavantajları maddeler halinde aşağıdaki gibi sıralayabiliriz:

Şekil 16. Windows Sunucu İşletim Sistemlerinin Avantaj ve Dezavantajları

Windows sunucu işletim sistemleri sürümleri aşağıdaki gibi listelenebilir:

- Windows Server 2003 (Nisan 2003)
- Windows Server 2003 R2 (Aralık 2005)
- Windows Server 2008 (Şubat 2008)
- Windows Server 2008 R2 (Ekim 2009)
- Windows Server 2012 (Eylül 2012)
- Windows Server 2012 R2 (Ekim 2013)
- Windows Server 2016 (Eylül 2016)
- Windows Server 2019 (Ekim 2018)
- Windows Server 2022

Windows, arka plan servislerini Services.msc isimli uygulama ile yönetmenize izin verir. Windows servislerini Linux'de olduğu gibi kolayca yönetemeyebilir veya kendi servisinizi yazmayabilirsiniz. Windows kapalı kaynak kodlara sahip olduğu için sadece size verilen komutlarla veya arayüzlerle işlemlerinizi yapabilirsiniz. Servisleri Resim 32. de göründüğü gibi, Services.msc isimli uygulama ile yönetebilirsiniz:

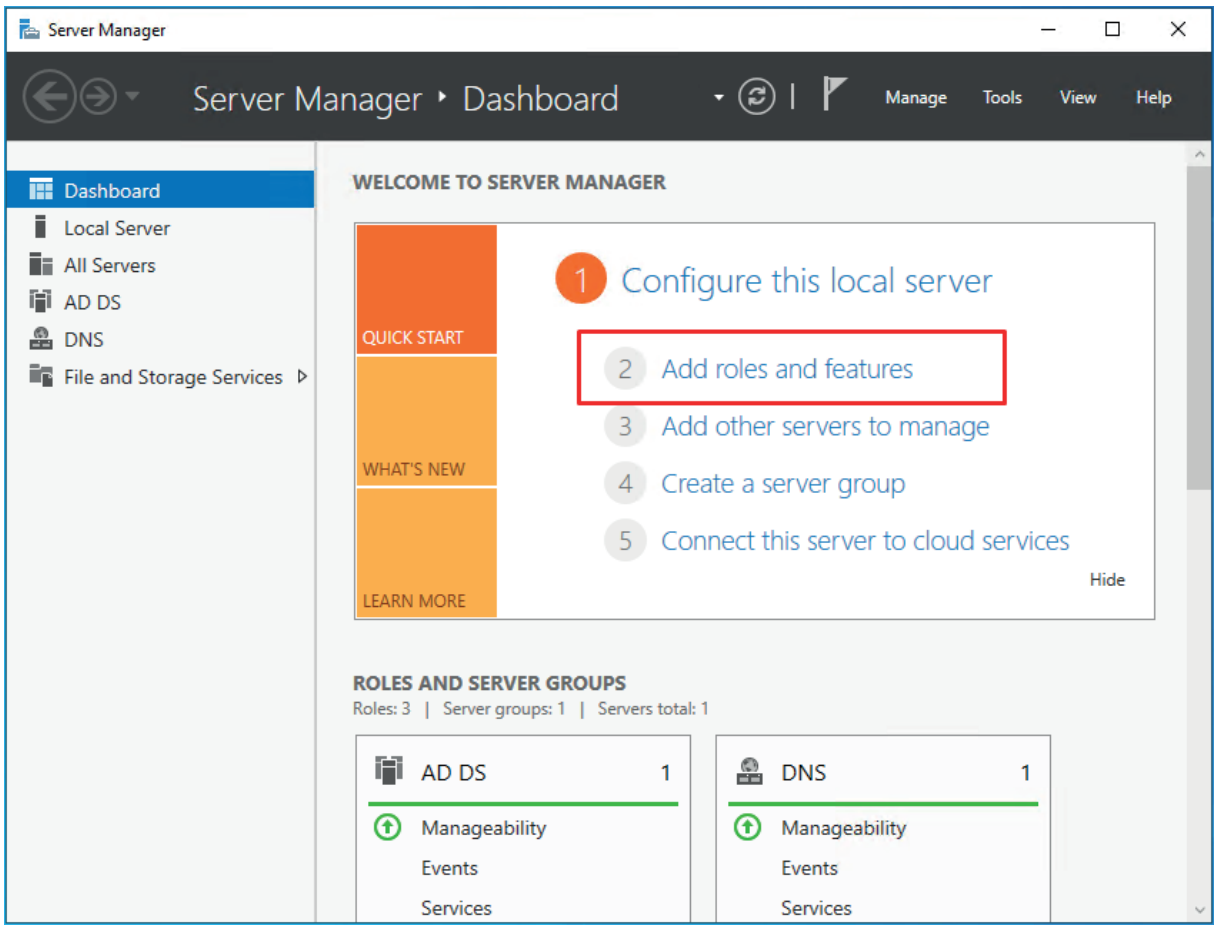


Resim 32. Windows Servisleri

Kendiniz yeni bir servisi oluşturmak istiyorsanız sc.exe yazılımından faydalanarak komut satırından yeni bir servis oluşturabilirsiniz. Aşağıdaki kod parçası yeni bir servis oluşturmak için gerekli örnek komutu göstermektedir:

sc.exe create NewService binpath= c:\windows\system32\NewServ.exe type= share start= auto depend= +TDI NetBIOS

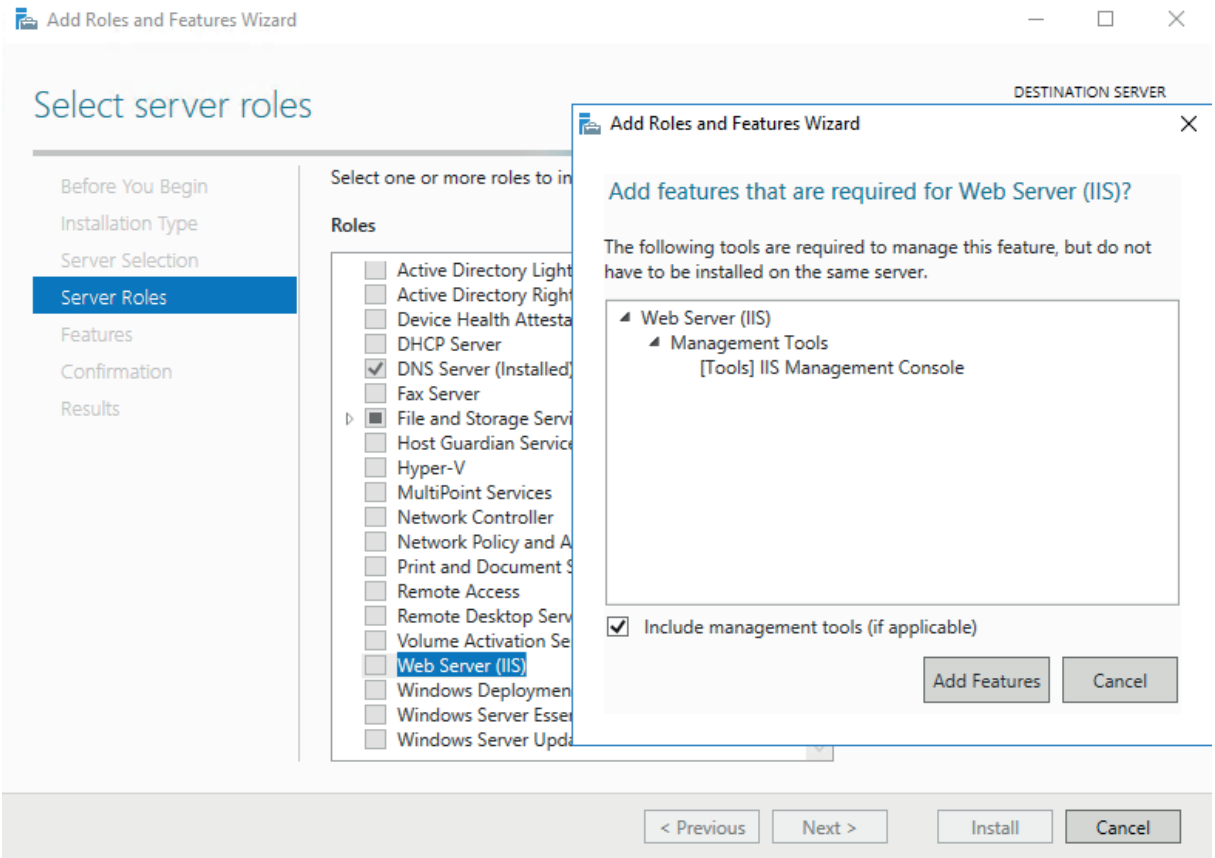
Windows sunucularda, web sunucu yazılımı için IIS (Internet Information Services) kullanılmaktadır. Elbette farklı üçüncü parti yazılımlar da kullanılabilir. Windows sunucularda, sunucu bileşenlerini Server Manager arayüzü ile tek bir ekrandan kurmak veya kaldırmak mümkündür. Resim 33.te Server Manager ekranı görülmektedir. Add roles and features ile sunucunuza yeni bileşenler eklenebilir. Role-based or feature-based installation seçeneği ile sihirbazdan kuruluma devam edilir. İlgili sunucu seçilerek işlemin hangi sunucuda yapılacağı belirtilmelidir. İstenirse uzak bir sunucuya aynı ekrandan kurulum yapılabilir.



Resim 33. Windows Server Manager

IIS seçenekler arasından seçilerek devam edilir. Resim 34.te görüldüğü gibi ilgili rol seçilir. Sihirbaz, kurulum esnasında ilgili modül için en temel bileşenleri

seçili olarak size sunacaktır. Farklı bileşenlerin eklenmesi isteniyorsa ekranlardan ilgili bileşenler seçilebilir.



Resim 34. IIS kurulumu rol seçimi

Kurulumdan sonra işletim sistemini yeniden başlatmanız gerekecektir. Internet Information Services Manager yazılımını kullanarak web sunucunuzu yapılandırabilirsiniz.

Windows işletim sistemlerinde PowerShell çok güçlü bir terminaldir. PowerShell ile Windows sunucularınızı yönetmeniz mümkündür. Örnek olarak aşağıdaki komut ile PowerShell üzerinden IIS kurulumu yapabilirsiniz.

```
Install-WindowsFeature -name Web-Server -IncludeManagementTools
```

7.3. Paket Yönetimi

7.3.1. Paket Sistemleri

Paketleme sistemlerinin temel birimi paket dosyalarıdır. Paket dosyaları yazılımı oluşturan sıkıştırılmış dosyalardan oluşmaktadır. Dosyalar veri dosyası, program dosyası veya dosya ile ilgili bilgiler barındıran metadata dosyaları da olabilir. Birçok paket, paket kurulumundan önce ve sonra yapılandırma görevlerini

üstlenen komut satırlarını da içermektedir. Paket dosyaları paket bakıcısı olarak bilinen kişi tarafından oluşturulur. Paket bakıcısı genellikle dağıtım satıcısının bir çalışanıdır.

Yazılımlar genellikle işlerini gerçekleştirebilmek için diğer yazılım bileşenlerine ihtiyaç duyarlar. Nadir de olsa başka yazılımların bileşenlerine ihtiyaç duymayan yazılımlarda vardır. Girdi-çıktı işlemleri birçok program tarafından kullanılan ortak faaliyetlerdir. Farklı birçok programın ihtiyaç duyduğu ortak işlemler "paylaşılabilir kütüphaneler" olarak isimlendirilen depolarda saklanırlar. Eğer bir yazılım paylaşılan bir kaynağa ihtiyaç duyuyorsa bu yazılımın bağımlı olduğunu söyleriz. Modern paket yönetim sistemlerinde bağımlılıklar paketin içerisinde bulunur ve ekstradan kullanıcının yüklemesine gerek yoktur.

Paket yönetim sistemi veya paket yöneticisi; paketlerin kurulum, güncelleme, konfigürasyon, kaldırılması işlemlerinin tutarlı bir şekilde yürütülmesini sağlayan sistemlerdir (Wikipedia, 2021).

Bir paket yöneticisi (Aksan, 2021);

- Paketin indirilmesi esnasında paketin doğruluğunu ve bütünlüğünü kontrol etmelidir.
- Paketin kaynağının kimlik doğrulamasını yapmalıdır.
- Güncelleme işlemlerini hatasız yapmalı, güncelleme takibini gerçekleştirebilmelidir.
- Bağımlılık yönetimini gerçekleştirmelidir.
- Paketler hataya sebep vermeyecek şekilde yapılandırılmalıdır ve çalıştırılmalıdır.

Paket yönetiminin önemini bir senaryo üzerinden görelim. Bir proje başlandığında ek kütüphanelere, kod parçacıklarına ihtiyaç duyulabilir. Bunlar projenin bağımlılıklarıdır. Bağımlılıklar önceliklerine göre sıralanır ve kurulum gerçekleştirilir. Belirli bir zaman dilimi içerisinde güncellemeler kontrol edilir, varsa değişiklikler uygulanır. Kullanılmayan dosyalar kaldırılır. Güncellemeler, uygulama çalışmasını bozmayacak şekilde olmalıdır. Uygulamanın yedeğini almak ya da uygulamayı taşımak isteyebiliriz. Tüm bu işlerin tek tek yapılması, kontrol edilmesi oldukça zordur. Paket yöneticisi bu süreçteki işlemleri üstlenirler ve tek bir komutla birçok işlem gerçekleştirilebilir.

7.3.2. Linux'te Paket Yönetimi

Farklı amaçlar için tasarlanmış yüzlerce farklı Linux dağıtımı vardır. Linux dağıtımlarının kalitesini belirleyen önemli parametrelerinden biri paket sistemleridir. Linux yazılım ortamı oldukça dinamiktir. Geliştiriciler tarafından sürekli güncelleme gerçekleştirilmektedir. En popüler Linux dağıtımlarının çoğu her altı ayda bir yeni sürüme geçiyor ve her gün yeni programlar ekleniyor.

Dinamik yapıya ayak uydurabilmek için güçlü bir paket yönetimi araçlarına ihtiyaç duyulmaktadır.

Paket yönetimi sayesinde kullanıcılar, sisteme kolaylıkla yazılımlar yükleyebilmekte veya mevcut yazılımların bakımını yapabilmektedir. Kullanıcılar, Linux dağıtıcılarının paketlerini yükleyerek ihtiyaç duyabilecekleri yazılımları kolaylıkla kullanabilmektedirler. Farklı dağıtımlar farklı paketleme sistemlerini kullanırlar ve genellikle bir dağıtımın desteklediği paketleme sistemi başka bir dağıtımla uyumlu değildir.

Dağıtım, yazılım geliştirilme döngüsünün farklı aşamalarında farklı depoda saklanabilirler. Dağıtım ilk başta test deposunda tutulur. Test aşamasındaki dağıtım; geliştiriciler, yazılım testi yapan kişiler, kullanıcılar tarafından incelenir, değerlendirilir. Hatalar bulunup test edildikten sonra genel bir dağıtım olarak piyasaya verilir.

Windows işletim sistemlerinde program kurulum dosyaları .exe uzantılıdır. Linux işletim sisteminde ise birkaç kurulum dosya uzantısı mevcuttur. .deb ve .rpm popüler paketleme biçimidir. .deb paketleme biçimi Debian Linux dağıtımı tarafından üretilmiştir. Ubuntu, Pardus gibi Linux dağıtımları tarafından kullanılmaktadır. .rpm paketleme biçimi Red Hat Linux dağıtımı tarafından geliştirilmiştir. Fedora, CentOS gibi Linux dağıtımları tarafından kullanılmaktadır. .deb ve .rpm paketleme biçimlerinde paketleme, güncelleme, bakım, bağımlılıkları içermemeleri nedeniyle bağımlılık takibi gibi işler zor olmaktadır. Flatpak, Snap, AppImage bağımlılıkları kendi içerisinde barındıran, tüm Linux dağıtımlarında kullanılmaya imkân veren yeni nesil paketleme sistemleridir.

Paketleme formatları ve araçları işletim sistemine göre değişkenlik gösterir. Debian, Ubuntu, Raspbian dağıtımlarında dosya formatı .deb uzantılıdır. Paket yönetiminde ise APT paket yöntemi kullanılır. APT paket yönetimi, gelişmiş paket yönetiminde en çok kullanılandır. APT paket yönetimi ile kolaylıkla paket kurulumu, arama işlemi, güncelleme işlemleri gerçekleştirilebilir.

Fedora, Red Hat dağıtımlarında dosya formatı .rpm uzantılıdır. “yum” komutuyla paket dosyaları ve depoları ile iletişim kurulmaktadır. Paket yönetimi görevlerinden birçoğu komut satırından gerçekleştirilebilir. Yaygın olarak kullanılan komutları inceleyelim (Shotts, 2013), (Özdemir, 2018).

- **Depoda paket bulmak**

Bir paketin adı veya açıklamasına göre paket bulunabilir. Birçok dağıtım kullanıcıya paket arama işlemi için kullanıcı arayüzleri sunar. Daha ayrıntılı bir arama işlemi konsol üzerinden gerçekleştirilir. Paketin bulunabilmesi için konsolda Tablo 12.de verilen komutların komut satırına yazılması yeterlidir:

Tablo 12. Linux Paket Bulma Komutları

Sistem	Komut
Debian/Ubuntu	apt-cache search arama_kelimesi
Red Hat/ CentOS	yum search arama_kelimesi
Fedora	dnf search arama_kelimesi

- **Linux depodan paket yüklemek**

Yüklemek istenilen paketin adı verilerek Linux depodan paket yüklemesi Tablo 13.te verilen komutların komut satırına yazılması yeterlidir. Paket yüklemesi ile paketin bağımlılıkları da yüklenir:

Tablo 13. Linux Depodan Paket Yükleme Komutları

Sistem	Komut
Debian/Ubuntu	sudo apt-get install paket_dosyası
Red Hat/ CentOS	sudo yum install paket_dosyası
Fedora	sudo dnf install paket_dosyası

Bir paket dosyası depodan değil de başka bir kaynaktan düşük seviyeli araç kullanılarak (bağımlılık çözümü olmayan) indirilebilir (bk. Tablo 14).

Tablo 14. Paket Dosyasından Paketin İndirilmesi

Sistem	Komut
Debian/Ubuntu	dpkg --install paket_dosyası
Red Hat/ CentOS	rpm -i paket_dosyası
Fedora	sudo dnf install paket_dosyası

- **Linux yüklü paketlerin kaldırılması**

Bir paket dosyası paketten kaldırılabilir (bk. Tablo 15).

Tablo 15. Linux Yüklü Paketlerin Kaldırılması

Sistem	Komut
Debian/Ubuntu	sudo apt-get remove paket_dosyası
Red Hat/ CentOS	sudo yum remove paket_dosyası
Fedora	sudo dnf erase paket_dosyası

- **Linux yüklü paketlerin güncellenmesi**

Linux yüklü paketlerinin güncellenmesi işlemi dikkatlice gerçekleştirilmesi gereken bir işlemdir. Güncelleme esnasında nelerin değiştiği takip edilmeli

ve yapılan bildirimler dikkatlice okunarak onay verilmelidir. Takip edilmeden gerçekleştirilen güncellemeler sonrasında çalışan dosyalarda sorunlar oluşabilir. Konfigürasyon dosyalarının güncellenmesi işlemi en sık rastlanan sorunlardandır. Yüksek seviyeli araçlar, bu önemli görevi tek bir adımda gerçekleştirebilirler (bk. Tablo 16).

Tablo 16. Linux Yüklü Paketlerin Güncellenmesi

Sistem	Komut
Debian/Ubuntu	apt-get update; apt-get upgrade
Red Hat/ CentOS	sudo yum update
Fedora	sudo dnf upgrade

Paket dosyasından bir paketin versiyonunun yükseltilmesi işlemi için Tablo 17.de verilen komutlar komut satırına yazılır:

Tablo 17. Paket Dosyasından Bir Paketin Güncellenmesi

Sistem	Komut
Debian/Ubuntu	dpkg --install paket_dosyası
Red Hat/ CentOS	rpm -U paket_dosyası

- **Linux yüklü paketlerin listelenmesi**

Sistemde kurulu tüm paketlerin listesini verir (bk. Tablo 18).

Tablo 18. Sistemde Kurulu Paketlerin Listelenmesi

Sistem	Komut
Debian/Ubuntu	dpkg --list
Red Hat/ CentOS	rpm -qa

- **Paket kurulum durumunun tespiti**

Bir paketin kurulu olup olmadığını tespit ederiz (bk. Tablo 19).

Tablo 19. Paketin Kurulum Durumunun Kontrol Edilmesi

Sistem	Komut
Debian/Ubuntu	dpkg –status paket_dosyası
Red Hat/ CentOS	rpm -q paket_dosyası

- **Kurulu paketler hakkında bilgilerin elde edilmesi**

Paketi yüklemeden paketlerin ayrıntılı açıklamalarını okumak fikir verir. Paketin ismi bilindiğinde, aşağıda verilen komutlar kullanılarak paketin açıklamalarını görebiliriz (bk. Tablo 20).

Tablo 20. Paket ile İlgili Ayrıntıları Görmek

Sistem	Komut
Debian/Ubuntu	apt-cache show paket_ismi
Red Hat/ CentOS	yum info paket_ismi

- **Kurulu dosyanın paketini bulmak**

Belirli bir dosyanın yüklenmesinden hangi paketin sorumlu olduğunu bulabilmek için verilen komutlar kullanılmaktadır (bk. Tablo 21).

Tablo 21. Dosyanın Paket Bilgisini Yakalamak

Sistem	Komut
Debian/Ubuntu	dpkg --search dosya_ismi
Red Hat/ CentOS	rpm -qf dosya_ismi

- **Paket yöneticileri yardım komutu**

Unix sistemlerde konsol üzerinden "man" komutu kullanılarak yardım alınabilir. man uygulamasında aşağı-yukarı ok tuşlarıyla komutlar gezinebilir. "man komut-ismi" yazılarak komut işlevi ile ilgili bilgiler elde edilebilir.

Tablo 22. "man" Komudunun Kullanımı

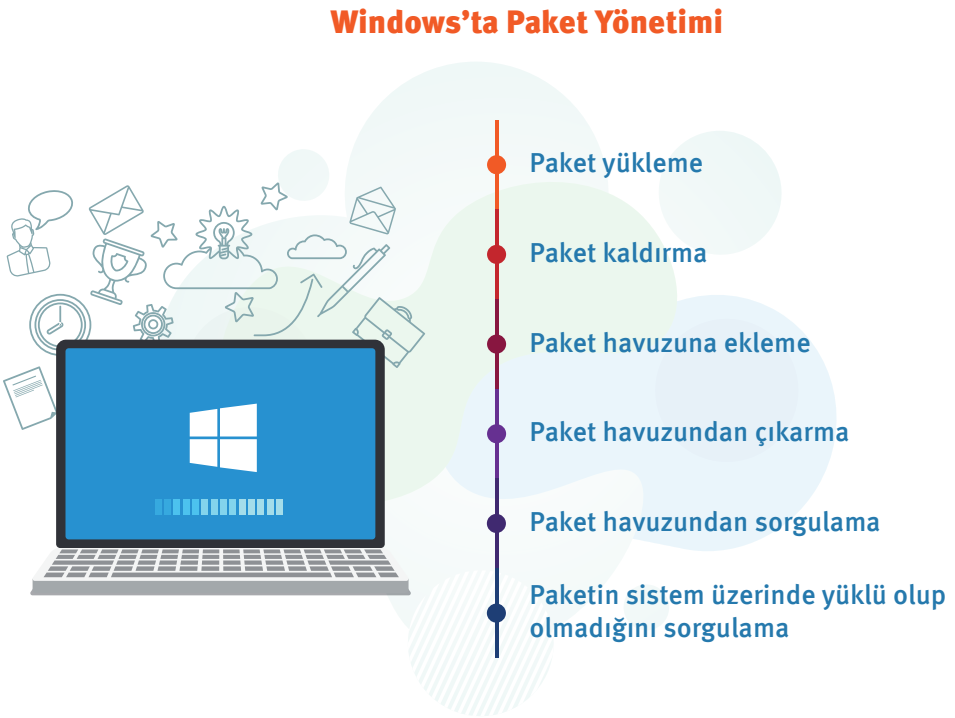
Sistem	Komut	Açıklama
Debian/Ubuntu	man apt-get	Paket güncelleme ve paketlerle çalışma
Red Hat/ CentOS	man yum	Paket aramak, sorgulamak

7.3.3. Windows'ta Paket Yönetimi

Microsoft dünyasında paket yönetimi için yaygın olarak MSI, MSP, MSU, APPX vb. paketler kullanılır. Microsoft, Windows için Linux tarzı bir paket yönetimi çerçevesi geliştirmeyi hedeflemiştir. Windows 10 ile OneGet paket yönetim çerçevesi gelmiştir. OneGet bir paket yöneticisi değildir. OneGet bir paket yönetimi yöneticisidir. OneGet, Powershell'in bir parçasıdır. OneGet; Chocolatey'in mevcut paketleri ile test edilmektedir, masaüstü uygulamalarını ve diğer yazılımları kolayca yükleyebilmeyi sağlar. OneGet, açık kaynaktır.

Chocolatey, Windows için bir paket yöneticisidir. Linux'de kullanılan "apt-get" komudu işlevini görür. Chocolatey ile dış kaynaktan bir dosya indirmeden

"powershell" kullanılarak ihtiyaç duyulan 3. parti uygulamalar, paketler mevcut sisteme yüklenebilir. Chocolatey ile indirilen paket, bağımlılıkları ile indirilir. Kullanıcıya iş bakımından zaman ve rahatlık kazandırır. Chocolatey ile uzakta bulunan bir bilgisayara paket yükleme işlemi gerçekleştirilebilir. Chocolatey, OneGet paket yönetim çerçevesine yerleşik olarak gelir.



Şekil 17. Oneget (Packagemanagement) Paket Yönetimi Modülü İle Gelen Cmdlet'leri Kullanarak Gerçekleştirilebilen İşlemler. (Aladağ, 2015)

Yaygın Olarak Kullanılan cmdlet'ler;

- ***Get-PackageSource;*** Kurulu paket kaynaklarının listesini görüntülememizi sağlar.
- ***Add-PackageSource;*** Depo eklemek için kullanılır.
- ***Remove-PackageSource;*** Depoları kaldırmak için kullanılır.
- ***Find-Package;*** Paket aramak ve keşfetmek için kullanılır.
- ***Install-Package -Name;*** İsmini belirttiğiniz paketi kurmanızı sağlar. Herhangi bir .exe dosyası aramanıza, .exe dosyasını bilgisayarınıza indirmenize hatta kurulum sihirbazıyla uğraşmanıza gerek kalmaz. Otomatik olarak paket indirilir ve kurulur. Birden fazla paket ismi belirtilebilir. Tek bir komutla birden çok yazılım yüklenebilir.
- ***Get-Package;*** Kurulan paketleri görmenizi sağlar.
- ***Uninstall-Package;*** Kurulan paketlerin kaldırılmasını sağlar.

- ***Save-Package;*** Paket, başka bir konuma indirilir.
- ***Get-PackageProvider;*** Paket yöneticilerini listeler. Örneğin; Chocolatey, NuGet.
- ***Register-PackageSource;*** Tanımlanan paket yöneticisi için paket kaynağı ekler.
- ***Unregister-PackageSource;*** Kayıtlı paket kaynağını kaldırır.
- ***Get-PackageProvider -Name Chocolatey;*** -Name parametresine Chocolatey değeri verilerek ilgili paket yönetici hakkında bilgi alınabilir.
- ***Find-Package -ProviderName chocolatey -Name Notepad;*** Chocolatey paket yöneticisi kullanılarak Notepad isminde bir paketin olup olmadığı kontrol edilir.
- ***Install-Package Notepad -Confirm;*** Notepad paketi yüklenirken kullanıcıdan onay almasını istediğimiz zaman kullanırız.
- ***Set-PackageSource -Name Chocolatey -Trusted*** komutu ile Chocolatey'in güvenilir olduğu bilgisini atadık.

Bu Bölümde Ne Öğrendik?

* Sunucu sistemlerinde Unix ve türevi işletim sistemleri büyük oranda kullanılmaktadır.
    * Linux işletim sistemlerinin çeşitli dağıtımları ortak Linux çekirdeği kullanırlar.
* Sunucular sürekli çalışan ve stabil bir sistem olmalıdır. Web, mail, veri tabanı, dns gibi servisler sunucular üzerinde hizmet vermektedirler. Bu uygulamalar sunucularda bir servis olarak çalıştırılırlar. Bu servisler Linux'de Systemd ile kontrol edilir. Systemd sistemde ilk başlatılan prosestir.
    * Windows'ta da servisler vardır. Windows servisleri Services.msc uygulaması veya PowerShell üzerinden kontrol edilebilmektedir. Linux ve Windows sistemler üzerinde herhangi bir uygulamayı servis olarak sistemde çalıştırmak mümkündür. Bunun için servis tanımlarının doğru bir şekilde sistemde tanımlanması gerekir.
* Sunucularda çalışan çok çeşitli yazılımlar mevcuttur. Bu yazılımlara örnek olarak; SSH sunucularından OpenSSH, web sunucularından nginx ve IIS, veri tabanı sunucularından PostgreSQL verilebilir.
    * Paketleme yönetimi olmadığında kullanıcı bir uygulama yüklemek istediğinde uygulamanın bağımlılıkları ile kurulması, güncellenmesi, konfigürasyonu, kaldırılması gibi işlemlerle tek tek kendisi uğraşır. Paket yönetim sisteminde işlevler tek bir komut satırı ile gerçekleştirilmektedir. Farklı işletim sistemlerinin kullandığı farklı paket yönetim sistemleri vardır. Bu sistemlerin kalitesi işletim sisteminin kalitesini belirleyen en önemli parametredir.
* Paketleme sistemlerinin temel birimi paket dosyalarıdır. Paket dosyaları; yazılımın konfigürasyon dosyası, metadata dosyası, program dosyası, veri dosyası gibi yazılımı oluşturan dosyalardan oluşur. Etkili bir paketleme sistemi ile tutarlı bir yönetim gerçekleştirilebilir.
    * Linux açık kaynak kodlu bir işletim sistemidir. Linux çekirdeğini kullanarak birçok farklı işletim sistemi geliştirilmiştir. Linux dağıtımları, geliştiriciler tarafından sürekli güncellenmektedir. Dinamik bir yapıya sahip Linux işletim sistemleri, güçlü bir paket yönetimi araçlarına ihtiyaç duyar. Farklı dağıtımlar farklı paketleme sistemlerini kullanırlar ve genellikle bir dağıtımın desteklediği paketleme sistemi başka bir dağıtımla uyumlu değildir.
* Windows kapalı kaynak kodlu işletim sistemidir. Microsoft, Windows işletim sistemi için Linux'deki gibi açık kaynak kodlu bir paket yönetim çerçevesi geliştirmeyi hedeflemiştir. Windows 10 ile OneGet gelmiştir.

* OneGet tam anlamıyla bir paket yöneticisi değildir. OneGet bir paket yönetimi yöneticisidir. OneGet, Powershell'in bir parçasıdır ve içerisinde Chocolatey gibi paket yöneticilerini içerir. OneGet paket yönetimi modülünde kullanılan cmdlet'ler ile paket yükleme, kaldırma, sorgulama, havuza eklemek, havuzdan çıkarmak ve paketin sistem üzerinde yüklü olup olmadığını sorgulama gibi işlevler tek bir komut satırı ile gerçekleştirilebilmektedir.

Kaynakça

(tarih yok).

(2021). Debian Manpages: https://manpages.debian.org/ adresinden alındı

Aksan, C. (2021). *Paket Yöneticisi Nedir? Neden İhtiyaç Duyulur?* ceaksan.com: https://ceaksan.com/tr/paket-yoneticisi adresinden alındı

Aladağ, M. (2015). *Windows 10 Yenilikleri – PowerShell 5 ile Paket Yönetimi OneGet*. cozumpark.com: https://www.cozumpark.com/windows-10-yenilikleri-powershell-5-ile-paket-yonetimi-oneget/ adresinden alındı

Alkar, A. (2015). Embedded system basics and application. Ankara.

Çobanoğlu, B. (2018). *Herkes için Python*. Pusula.

Doğu Akdeniz Üniversitesi, B. B. (2021). İşletim Sistemleri-Bellek Yönetimi.

Doğu Akdeniz Üniversitesi, B. V. (2021). İşletim sistemleri-Kilitlenmeler. Siirt.

Gülbağ, A. (2017). İşletim Sistemlerine Giriş.

javaTpoint. (2021). *OS*. javaTpoint: https://www.javatpoint.com/os-attributes-of-a-process adresinden alındı

Kaya, A. (2009). Symbian İşletim Sistemi. *Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri*. Şanlıurfa.

Özdemir, H. (2018). *Linux Paket Yöneticileri*. pythontr.com: https://www.pythontr.com/makale/linux-paket-yoneticileri-632 adresinden alındı

Saatçi, A. (2002). Bilgisayar işletim sistemleri. Ankara: Hacettepe Üniversitesi. http://hilmi.trakya.edu.tr/ders_notlari/os/isleti_sistemleri.pdf adresinden alındı

Samet, R. (2018). Bilgisayar Sistemleri. (A. Ü. Enstitüsü, Dü.)

Shotts, W. (2013). *The Linux Command Line- Second Internet Edition*. No Starch Press.

Silberschatz, A., Gagne, G., & Galvin, P. (tarih yok). *Operating System Concepts*. içinde John Wiley & Sons, Inc. 2021 tarihinde alındı

Taşcı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Taşçı, T. (2017). İşletim Sistemleri-Temel Bilgi Teknolojileri Kullanımı.

Türkoğlu, İ. (tarih yok). İşletim Sistemleri (Ders Notları). *Fırat Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Bölümü*. Türkiye. 2021 tarihinde alındı

Wikipedia. (2021). *Paket yönetim sistemi*. Wikipedia.org: https://tr.wikipedia.org/wiki/Paket_y%C3%B6netim_sistemi adresinden alındı

Yıldırım, S. (tarih yok). Bilgi Teknolojilerine Giriş3. Atatürk Üniversitesi. 2021 tarihinde alındı

8. GÜVENLİK VE YEDEKLEME

Gerçek güvenlik diye bir şey yoktur. Hiç olmamıştı.

Germaine Greer

Ders videosunu izlemek için QR kodu taratın.

Kazanımlar

- Kullanıcılar, tehditlere karşı alabileceği önlemleri öğrenebilir.
- Linux ve Windows işletim sistemlerinde güvenliği sağlamak için yaygın olarak kullanılan komutları öğrenebilir.
- Linux'de çekirdek seviyesinde güvenlik duvarını yapılandırabilir.
- Farklı yedekleme senaryolarını bilebilir ve uygulayabilir.
- Linux ve Windows işletim sistemlerinde kayıt dosyalarını inceleyebilir.

Başlamadan Önce

Bu bölümde, işletim sistemleri güvenliği tehditlerine karşı kullanıcıların alabileceği birtakım önlemler, işletim sisteminin kendi bünyesinde bulunan mekanizmalar, güvenliğin sağlanması için geliştirilen yazılım uygulamaları ve yedekleme senaryoları en çok tercih edilen Windows/Linux işletim sistemleri üzerinden anlatılacaktır.

Birlikte Düşünelim

1. Son kullanıcıların işletim sisteminin güvenliğini sağlamada sorumlulukları var mıdır?
2. Yedekleme hangi yöntemlerle gerçekleştirilebilir?
3. Plansız gerçekleştirilen yedekleme işleminin oluşturduğu problemler nelerdir?

8.1. Güvenlik

İşletim sistemleri sistem üzerinde çalışan diğer uygulamalara ve servislere güvenlik ilkesini temel alarak hizmet sunduğu için işletim sisteminin güvenliğinin sağlanması bütün sistemin güvenliği için temel oluşturmaktadır. İşletim sistemleri gizliliğin sağlanması, kullanılabilirliğinin devamı için çeşitli mekanizmalara sahiptir. İşletim sistemi güvenliği, işletim sistemini virüslerden, solucanlardan, tehditlerden, korsan saldırılarından kendini koruyabilmek için kullandığı önlemleri ifade eder.

Kullanıcıların tehditlere karşı alabileceği birtakım önlemler vardır. İşletim sistemlerinin beraberinde sunduğu güvenlik önlemleri her zaman yeterli olmayabilir. Bu nedenle güvenli bir işletim sistemini seçmek kullanıcının yapması gereken en temel önlemlerden birisidir. Kullanılan işletim sistemini güncel tutmak ve güncellemeleri takip etmek de oldukça önemlidir. Güncellemeler sayesinde kullanıcılar sistemlerini en yeni tehditlere karşı korumuş olurlar. İşletim sistemlerinin bileşenlerinin güncellenmesi dışında kullanıcıların sistemlerinde çalıştırdığı programlara da dikkat etmesi oldukça önemlidir. Bir elektronik posta, usb bellek veya herhangi bir web sayfasından indirilen içeriği belli olmayan çalıştırılabilir dosyalar sistem için önemli bir tehdittir. Kullanıcıların bu içerikleri bir antivirüs yazılımı ile kontrol etmesi veya sisteme bir antivirüs yazılımı kurması bu tip tehditlere karşı alınabilecek en güzel yöntem olacaktır. Güvenilmeyen hiçbir usb bellek, klavye, fare gibi cihazların bilgisayara takılmaması sistemin güvenliği açısından önemlidir. Bu tip cihazların donanım seviyesinde sisteme erişim sağladığını veya görünen cihazın bir başka cihazı taklit edebileceğini unutmamalıyız. Kötü amaçlı tasarlanan bir klavye, basılan tüm tuşları kayıt altına alabilir ve bu bilgileri dışarıya transfer edebilir.

8.1.1. Kullanıcı ve Gruplar

Çok kullanıcılı işletim sistemleri, birden fazla kullanıcının tek bir makineye aynı anda erişmesini sağlarlar. Farklı kullanıcılar aynı anda ağ üzerinden bağlantı yapabilir, çeşitli programları çalıştırarak işlemci, ram, disk gibi kaynakları ortak kullanabilirler. Tek bir makinenin birden fazla kullanıcı tarafından kullanılması güvenlik problemlerini de beraberinde getirmektedir. Herhangi bir kullanıcının sisteme bir virüs bulaştırması tüm sistemi etkileyecektir. Kullanıcıların, özellikle yetki sahibi olan kullanıcıların, güvenli parolalar kullanması ve parolalarını saklaması gerekir.

İşletim sistemlerinde bir kullanıcı tarafından çalıştırılan komutlar, o kullanıcının yetki seviyesinde çalıştırılır. Yönetici gibi ileri düzey yetki seviyesi gerektiren komutların ise kullanıcının yetki seviyesinin yükseltilerek çalıştırılması istenir. Linuxde sudo ile komuta başlamak, Windows'ta "yönetici olarak çalıştır" seçerek işlem yapmak gibi yetki seviyesinin yükseltilmesine örnek verebiliriz. Güvenlik açısından, yetki gerektirmeyen komutların asla yetki seviyesi yükseltilerek çalıştırılmaması gerekir.

Gruplar ile sistemde farklı yetkiler farklı başlıklar altında toplanabilir. Kullanıcılara doğrudan çeşitli yetkilerin verilmesi yerine bir gruba atanarak gruplar halinde yetkilendirilmesi kolay ve güvenilir bir yöntemdir. Kullanıcıların gereksiz gruplara dahil edilmemesi gerekir. Gereksiz yetki tanımı, güvenlik açısından her zaman bir tehdittir.

Çok kullanıcılı işletim sistemlerinde kullanıcılar ve gruplar farklı şekillerde sisteme tanımlanabilirler. Bu başlık altında Linux işletim sisteminde kullanıcı yönetimi ele alınmıştır. Diğer işletim sistemlerinde de kullanıcı ve grup yönetimi temelde benzer bileşenlerden oluşur.

Linux işletim sistemlerinde kullanıcı yönetimi için gerekli olan kullanıcı ekleme, silme, dondurma, parola değiştirme işlemleri için bilinmesi gereken komutlar aşağıda anlatılmaktadır. Bununla birlikte grup oluşturma, grup silme, gruba kullanıcı ekleme ve çıkarma işlemleri de aşağıdaki komutlarda verilmiştir:

Yeni kullanıcı oluşturmak için useradd komutu kullanılır. Aşağıdaki komut ile "Baran Kaynak" tam adı ile baran kullanıcısı oluşturulmuştur. Bu komutun superuser olarak çalıştırılması gerektiğini unutmamalıyız. Bu sebeple sudo komutu ile başlamalıyız. Kullanıcı eklendiğinde kullanıcı /etc/passwd dosyasına eklenir. Bu dosya tüm kullanıcıların bilgilerini saklar.

```
sudo useradd -c "Baran Kaynak" baran
```

Oluşturulan kullanıcının herhangi bir parolası yoktur ve kullanıcı kilitlidir. Kullanıcıya yeni bir parola atayarak kullanıcının kilidini kaldırabiliriz. Parolayı değiştirmek için "passwd kullanıcıadi" komutunu kullanabiliriz. Komut çalıştırıldığında belirlenmek istenen yeni parola ve parolanın doğrulaması sorulacaktır.

```
sudo passwd baran
```

Bir kullanıcıyı silmek için aşağıdaki komutu kullanabilirsiniz. Kullanıcıya ait ana dizini de silmek için -r parametresi komuta eklenmelidir.

```
sudo userdel baran
```

passwd komutu ile birçok farklı kullanıcı işlemleri yapılabilir. Farklı parametreler ile yapılabilecek işlemler aşağıda özetlenmiştir.

Kullanıcının durumunu kontrol etmek için;

```
sudo passwd -S baran
```

Kullanıcının parolası geçersiz kılınmış ise parola tekrar kullanılamaz. Kullanıcıya yeni bir parola verilmesi zorunludur. Kullanıcının parolasını geçersiz kılmak için;

```
sudo passwd -e baran
```

Kullanıcıyı kilitlemek için;

```
sudo passwd -l baran
```

Kullanıcının kilidini kaldırmak için;

```
sudo passwd -u baran
```

Kullanıcı ekleme anında Linux, bu kullanıcıya birincil bir grup ataması yapar. Özel bir grup seçilmemiş ise Linux, eklenen kullanıcının idsi ile aynı idye sahip

Anahtar Kavram

Parametre: Çok değişen, değişken.

olan yeni bir gruba atama yapar. Kullanıcı bu grubun tek üyesidir. Kullanıcı istenirse başka ikincil gruplara dahil edilebilir.

Sistemde grup 1 adında yeni bir grup oluşturmak için aşağıdaki komut çalıştırılmalıdır:

```
sudo groupadd grup1
```

baran isimli kullanıcıyı grup1'e dahil etmek için;

```
sudo usermod -aG grup1 baran
```

baran isimli kullanıcıyı grup1'den çıkarmak için;

```
sudo gpasswd -d baran grup1
```

grup1 isimli grubu silmek için;

```
sudo groupdel grup1
```

8.1.2. Dosya ve Dizin İzinleri

İşletim sistemlerinde ne kadar iyi güvenlik önlemleri alınmış olursa olsun dosya ve dizin izinlerinin doğru bir şekilde atanamamasından kaynaklanan güvenlik açıkları oluşabilir. Dosyaları oluşturan kullanıcılar bu dosyalar üzerinde izin belirleme hakkına sahiptir. Dosya ve dizinlere erişme yetkisi kullanıcı ve gruplar seviyesinde belirlenebilir. Her bir dosya ve dizin için okuma, yazma ve çalıştırma yetkileri tanımlanabilir. Çoğu çok kullanıcılı işletim sistemlerinde çalışma biçimi aynıdır.

Linux işletim sistemlerinde yöneticiler tüm dosya ve dizinlere erişim hakkına sahiptir. Bu durum yetkili kullanıcıların dikkatli olmasını gerektirir. Parolası ele geçirilen bir yetkili kullanıcı ile tüm sistemdeki dosyalara erişilebilir. Yetkili kullanıcının etik değerlere dikkat etmesi gerekir. Linux işletim sistemlerinde dosya ve dizin izinlerini daha yakından tanımaya çalışalım.

Terminalden ls -l komutu ile dosya ve dizinlerin izin bilgisine erişebilirsiniz. Resim 35.te dosya ve dizinlerin izin bilgileri ilk kolonda gösterilmektedir. d ile başlayan ifade dizin olduğunu, - ile başlayan ifade ise bunun bir dosya olduğunu gösterir. merhaba.sh dosyasının izin bilgisine baktığımızda -rwxrwxr-x şeklinde izinlerin tanımlandığını görmekteyiz. Bu bilgiyi 3'erli bloklara ayıralım. İlk karakterin dizin veya dosya olduğunu belirttiğini unutmayalım. – rwx rwx r-x bloklarında bulunan ilk blok, dosya sahibinin sırasıyla okuma, yazma ve çalıştırma izinlerini ifade eder. İkinci blokta yine ilk blok ile aynı ifade, rwx, yer almaktadır. Bu bilgi dosyanın grubuna ait izinleri ifade eder. Hem dosya sahibi hem de grubu dosyayı okuyabilir, dosyaya yazabilir veya silebilir ve çalıştırabilirler. Son blok, r-x ise diğer tüm kullanıcıları ifade eder. Bu örnekte r ifadesi ile tüm kullanıcıların okuma iznine sahip olduğunu görebilmekteyiz. Ancak ikinci karakterde w yerine – ifadesi yer almaktadır. Bu durum diğer tüm kullanıcıların yazma iznine sahip olmadığını gösterir. Son karakterde bulunan x ifadesi ise diğer tüm kullanıcıların dosyayı çalıştırma iznine sahip olduğunu gösterir.

Dizinlerde ise durum yine benzerdir. Ancak ufak farklılıklar vardır. r ifadesi, eğer x ifadesi var ise dizindeki içeriğin listelenebileceğini ifade eder. w ifadesi

dizinin içinde dosyaların oluşturulması, silinmesi ve yeniden adlandırılabilmesine izin verildiğini ifade eder. Ancak w ifadesinin yine r de olduğu gibi x şartını da sağlaması istenmektedir. x, bu dizine girilebileceğini ifade eder.

```
kaynak@kaynak-VirtualBox:~$ ls -l
toplam 44
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Belgeler
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Genel
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 İndirilenler
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Masaüstü
-rwxrwxr-x 1 kaynak kaynak  151 Ağu 20 17:37 merhaba.sh
-rwxrwxr-x 1 kaynak kaynak   46 Ağu 26 00:47 merhaba_yaz.sh
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Müzik
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 15 17:30 Resimler
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Şablonlar
drwxr-xr-x 2 kaynak kaynak 4096 Ağu 10 20:28 Videolar
drwxrwxr-x 3 kaynak kaynak 4096 Ağu 27 16:42 YeniDizin
```

Resim 35. Linux'da İzinler

rwx ifadesini aslında sistem ikili kodlar şeklinde yorumlar. Bu ifade her bir karakterin 1 ya da 0 olması ile birer bit ile temsil edilir. İzinler, rwx için 111, rw- için 110, r-x için 101 ve --- için 000 şeklinde ikili kodlarla çalışır. Görüldüğü üzere oldukça kolay anlaşılır bir yapı vardır. Tablo 23.te bu konuya örnekleri ile daha yakından bakalım. Tabloda birinci sütun izin ifadesini, ikinci sütun bu ifadenin ikilik karşılığını, üçüncü sütun onluk karşılığını ve son sütun ise izinleri sözlü anlatımını göstermektedir.

Tablo 23. Linuxde Dosya ve Dizin Örnekleri

İzin	İkili kodlama	10luk kodlama	Açıklama
- rwx rwx rwx	111 111 111	777	Hiçbir kısıtlama yoktur. Herkes tüm işlemleri yapabilir. Tercih edilmeyen bir durumdur.
- rwx r-x r-x	111 101 101	755	Dosya sahibi okuyabilir, yazabilir ve çalıştırabilir. Diğer tüm kullanıcılar dosyayı okuyabilir ve çalıştırabilir. Ortak kullanılan bir program için kullanımı uygundur.
- rwx --- ---	111 000 000	700	Dosya sahibi okuyabilir, yazabilir ve çalıştırabilir. Diğer kişilerin hiç bir işlem için izni yoktur. Programların sadece sahibi tarafından erişilebilmesi ve çalıştırılabilmesi için idealdir. Diğer tüm kullanıcılardan gizli tutulmuştur.
- rw- rw- rw-	110 110 110	666	Tüm kullanıcılar okuyabilir ve yazabilir. Kimse çalıştıramaz.
- rw- r-- r--	110 100 100	644	Dosya sahibi okuyabilir ve yazabilir. Diğer herkes dosyayı sadece okuyabilir.
- rw- --- ---	110 000 000	600	Dosya sahibi okuyabilir ve yazabilir. Diğerlerinin hiç bir izni yoktur. Bir dosyayı gizli tutmak için kullanılabilir.
d rwx rwx rwx	111 111 111	777	d ifadesi ile başladığı için dizini temsil eder. Hiç bir yetki kısıtlaması yoktur. Herhangi bir kişi dizinlerde yeni dosya oluşturma, listeleme ve dosyaları silme işlemlerini yapabilir. Güvenlik açısından tavsiye edilmez.
d rwx r-x r-x	111 101 101	755	Dizin sahibi tüm yetkilere sahiptir. Diğer tüm kullanıcılar dizinleri listeleyebilir ancak yeni dosya oluşturamaz ve dosyaları silemez.
d rwx --- ---	111 000 000	700	Dizin sahibi tüm yetkilere sahiptir. Diğer kişiler dizinde hiç bir işlem yapamazlar.

Dosyaların ve dizinlerin izinlerini değiştirmek için chmod komutu kullanılır. chmod 2 parametre alır. İlk parametre izinin değerini, ikinci parametre ise dosyanın yolunu belirtir. Aşağıdaki kod ile merhaba.sh dosyasını Sadece sahibi tarafından çalıştırılabilir ve okunabilir yapalım. Diğer tüm kullanıcılar için gizli olsun.

```
chmod 700 merhaba.sh
```

Bir dosyanın sahibinin oluşturan kişi olduğunu önceden ifade etmiştik. Dosyanın sahipliğini değiştirmek için chown komutu kullanılır. Aşağıdaki komut, merhaba.sh dosyasının sahibini baran kullanıcısı olarak değiştirmek için kullanılır:

```
chown baran /home/kaynak/merhaba.sh
```

Dosyanın sadece grubunu değiştirmek için ise : ile grup adı yazılır.

```
chown :grup1 /home/kaynak/merhaba.sh
```

Hem dosya sahibini hem de grup adını değiştirmek için parametre baran:grup1 şeklinde yazılabilir:

```
chown baran:grup1 /home/kaynak/merhaba.sh
```

Dizinin içinde başka dizinler ve dosyalar var ise -R parametresi ile rekürsif olarak aynı işlem gerçekleştirilebilir.:

```
chown -R baran /home/kaynak/Klasor1
```

8.1.3. Güvenlik Duvarı

Güvenlik duvarları (firewall), bilgisayar ağlarının güvenliğini sağlarlar. Ağ üzerinde gelen ve giden trafiği izleyerek, önceden belirlenmiş kurallar çerçevesinde, trafiğin geçişine izin verir veya engeller. Güvenlik duvarları donanımsal, yazılımsal veya bulut tabanlı olabilir. Her birinin artıları ve eksileri vardır. İşletim sistemlerinde yazılımsal güvenlik duvarı vardır. Bu yazılım işletim sisteminin çekirdeğindeki ağ fonksiyonlarını kullanarak ağ arayüzü ile sistem arasına girer. Bu uygulama, gelen ve giden tüm paketleri inceler. Kurallara göre paketlerin sisteme girmesine izin verir veya izin vermez. Aynı şekilde dışarı yönlü de paketlerin sistemden çıkıp çıkamayacağına da karar verir. Linux işletim sistemleri, iptables ve netfilter ile güvenlik duvarı yazılımını sunar. Windowsta ise Windows Firewall vardır. Unix, MacOS, BSD, Solaris işletim sistemlerinde ise ipfw kullanılmaktadır. Kullanıcılar işletim sistemleri içinde gelen bu bileşenlerin yerine veya yanında başka yazılımları da kullanabilirler.

Güvenlik duvarları, tanımlanan kuralları kadar güçlüdür. Kuralların eksik veya yanlış tanımlanması durumunda güvenlik duvarı yazılımının yapabileceği birşey kalmaz. TCP/IP protokolünde, 4. katmanda, kurallar port ve ip bazlı tanımlanırlar. Gelişmiş yeteneklere sahip 7. katmanda çalışan güvenlik duvarları da vardır. Bunlar uygulama seviyesinde paketin içeriğine göre karar verirler ve çalışma yapısı çok daha karmaşıktır. Bu başlık altında Linux'de iptables ile güvenlik duvarı yapılandırılmasını inceleyeceğiz.

Linux'de en yaygın kullanılan güvenlik duvarı yazılımı iptables'dır. Sisteme giren veya sistemden çıkan her bir paket için Linux çekirdeğinde bulunan netfilter'da bulunan kancalar (hook) çalıştırılır. netfilter'ı iptables yazılımı ile kontrol edebiliriz. iptables, netfilter projesinin içindedir. Kuralları iptables üzerinden oluşturarak çekirdek seviyesinde güvenlik duvarını yapılandırabiliriz.

iptables'da 3 tip zincir (chain) bulunmaktadır. Bunlar input, forward ve output'tur. Input, gelen bağlantılar için kullanılır. Bir web sunucusuna 80 numaralı porttan bağlantı isteği geldiği zaman, iptables, ilgili port ve ip adresi için yazılan kuralları eşleştirmeye çalışır. Forward, yerele iletilmeyecek olan paketler için işletilir. İşletim sisteminin kendisine değil, bir başka makineye gidecek olan paketler için yazılan kuralları eşleştirmeye çalışır. Modem, router gibi cihazlarda kullanımı yaygındır. Output ise giden bağlantılar için kullanılır. Output zincirinde makineden dışarı doğru yapılan bağlantı istekleri için kurallar eşleştirilir.

Zincirlerden anlaşılacağı üzere iptables ile tanımlanan kurallar ilgili zincirlerde sıraya konulur ve çalıştırılırlar (Debian Manpages, 2021). Kurallar, bağlantının ip adresi ve portu üzerinden eşleştirilirler. Eşleşen kural için 3 farklı aksiyon gerçekleştirilebilir. Bunlar; Accept (kabul), Drop (düşür) ve Reject

(reddet)'tir. Accept ile bağlantıya izin verilir. Drop ile bağlantı düşürülür ve sanki hiçbir şey olmamış gibi davranır. Karşı taraf, cevap gelmediği için sistemin varlığından haberdar olmaz. Reject, bağlantıya izin vermez ve karşı tarafa red bilgisini döndürür.

iptables ile örnek firewall kuralları yazalım. iptables -A komutu ile yeni kuralları tabloya ekleyebiliriz. -I kullanarak tabloda sırasını da belirterek kuralı istediğimiz sıraya ekleyebiliriz. -s parametresi source (kaynak) adresi tanımlar. -p parametresi, TCP veya UDP tipini ve -dport parametresi ise giden bağlantıdaki hedef portu tanımlar. Aşağıdaki kural ile 10.0.0.20 adresinden gelen tüm bağlantıları engelleyebiliriz:

```
iptables -A INPUT -s 10.0.0.20 -j DROP
```

10.0.0.0/24 ağında tanımlı olan tüm ip adresleri 10.0.0.1-10.0.0.255 aralığını ifade eder. Bu ip adreslerinin tamamını ise aşağıdaki komutla engelleyebiliriz:

```
iptables -A INPUT -s 10.0.0.0/24 -j DROP
```

Belirli bir ip adresinden port bazlı engelleme de yapılabilir. Aşağıdaki komut ile 10.0.0.20 ip adresinden 80 portuna gelen tüm bağlantılar engellenmiştir:

```
iptables -A INPUT -p tcp –dport 80 -s 10.0.0.20 -j DROP
```

Girilen tüm bu kuralların bilgisayar yeniden başladığında tekrar sisteme tanımlanabilmesi için kaydetmeliyiz. Ubuntuda sudo iptables-save komutu ile kaydedebilirsiniz. Tanımlı tüm kuralları listelemek için de iptables -L komutu kullanılır. iptables -F komutu tanımlı olan tüm kuralları silecektir.

8.1.4. Kayıtlar

İşletim sisteminde gerçekleşen olaylar, kayıt defterlerinde tutulur. Çalışan programlar, gerçekleşen olaylar, servisler, sistem kayıtları gibi çeşitli kayıtlar vardır. İşletim sistemleri bunları ayrı ayrı kayıt altına alırlar. Bir programın veya servisin hata vermesi ve kapanması durumunda bu kayıtlar incelenerek sorunun ne olduğu anlaşılabilir. Bir kullanıcının hangi tarihte sisteme giriş/çıkış yaptığı, şifre denemeleri gibi güvenlik kayıtları da sistemde tutulmaktadır. Windows işletim sistemlerinde Windows Event Viewer ile loglar incelenebilir. Linuxte ise /var/log klasörünün altında bulunan dosyalar okunarak incelenebilir. Linux kayıt dosyalarının neler içerdiğini inceleyelim:

- /var/log/syslog: Sistem hakkında genel bilgi ve mesajları içerir. Sistemdeki tüm olayları saklar.
- /var/log/auth.log: Sistemdeki kimlik doğrulama kayıtlarını saklar.
- /var/log/boot.log: Sistem boot (önyükleme) aşamasında oluşturduğu mesajları bu dosyada saklar.
- /var/log/kern: Çekirdekte üretilen mesajları ve olayları saklar.
- /var/log/dmesg: Sistemdeki aygıt sürücülerinde oluşturulan mesajlar saklanır.
- /var/log/faillog: Başarısız oturum açma kayıtlarını saklar.

- /var/log/cron: Tanımlanan cron işlerinin mesajlarını saklar.

Sistemde logların çok fazla üretilebileceğini ve çok büyük dosyalar olabileceğini unutmayalım. Bunun için tail komutu ile log dosyalarını incelemek kolaylık sağlayacaktır. tail -f komutu ile log dosyasını canlı olarak izlemeniz mümkündür.

8.2. Yedekleme

Sistemlerde veri kaybının yaşanmaması için yedeklemelerin planlanması ve gerçekleştirilmesi gerekir. Yedekleme, farklı türlerde gerçekleştirilebilir. Kullanıcı verileri yedeklenebilir, işletim sistemi seviyesinde yedekleme yapılabilir veya disk bölümü tamamıyla yedeklenebilir. Bunların dışında sanallaştırma ile anlık görüntüler alınabilir. Tüm bu farklı yedekleme senaryolarının amacı, olası problemlerde veri kayıplarını en az düzeye indirmektir. İşletim sistemlerinde yedek almak, her sistem için farklı olabilir. Windows işletim sisteminde Başlat > Ayarlar > Güncelleme&Güvenlik > Yedekleme menüsünde yedek alma aracına erişebilirsiniz. Harici bir disk seçerek yedekleme yapabilir, gerektiğinde bu diskten yedeklerinizi geri alabilirsiniz. Linux işletim sistemlerinde ise işletim sisteminin içinde gelen bir yedekleme aracı yoktur. Ancak ücretsiz ve açık kaynak birçok yedekleme çözümü vardır. rsync bunlardan birisidir. Bu aracı farklı işletim sistemlerinde de kullanmak mümkündür.

Disk seviyesinde yedek almak, bazen daha kullanışlı olabilir. Bunun için disk bölümünü tamamıyla okuyup hedef diske kopyalayan bir yazılım aracı seçilmeli ve kullanılmalıdır. Bu başlık altında sadece dosyaların nasıl yedeklenebileceğini rsync yazılımı ile Linux üzerinde inceleyeceğiz.

Rsync, komut satırından dosyalarınızı gösterilen bir hedefe senkronize eder (Debian Manpages, 2021). Araç senkronizasyonu gelişmiş algoritmalar ile gerçekleştirir. Hedefte bulunan dosyalar ile kaynakta bulunan dosyaları karşılaştırır ve farklarını hesaplar. Böylelikle sadece değişen dosyaları senkronize ederek zaman kazanmış olursunuz. Rsync, sadece dizinler arasında senkronizasyon yapmaz. İki farklı bilgisayar arasında ağ üzerinden senkronizasyon yapabilir. Ağ üzerinden ssh, rsync veya s3 protokolleri ile senkronizasyon yapabilir. İstendiği taktirde yedekler şifrelenerek hedefte saklanabilir. Yedekleme işlemleri bir cronjob oluşturularak periyodik olarak gerçekleştirilebilir.

Sisteme bir usb disk bağladığımızı varsayalım ve bu diske verilerimizi yedeklemek isteyelim. Bu durumda aşağıdaki komut kullanılabilir. -r parametresi işlemin tüm alt dizinler için de yapılacağını ifade eder. /home/kaynak dizini /media/kaynak/myusbdisk/yedekler dizinine yedeklenmektedir:

```
rsync -r /home/kaynak /media/kaynak/myusbdisk/yedekler
```

Bu komut çalıştırıldığında çalıştırma anı, dosyanın özelliklerine yazılır ve dosyalar sanki yedekleme anında oluşturulmuş gibi görünür. Dosyaların özelliklerini saklamak için -a (archive) parametresi kullanılmalıdır:

```
rsync -ra /home/kaynak /media/kaynak/myusbdisk/yedekler
```

Yedekleme işlemi gerçekleştirilirken yapılan işlemleri görmek için -v (verbose) parametresi eklenmelidir. Kopyalama işlemi hakkında detaylı bilgileri görüntülemek için -P (progress) parametresi kullanılabilir.

Bu Bölümde Ne Öğrendik?

* İşletim sisteminin güvenliğinin sağlanması bütün sistemin güvenliği için temel oluşturmaktadır. İşletim sistemini virüslerden, solucanlardan, tehditlerden, korsan saldırılarından koruyabilmek için işletim sistemleri yazılımları içerisinde mekanizmalar mevcuttur.

    * İşletim sistemlerinin beraberinde sunulan güvenlik önlemleri her zaman yeterli olmayabilir. Tehditlere karşı alınabilecek birtakım önlemler vardır. Bu önlemlerden bazıları şu şekildedir; güvenli bir işletim sistemi seçilmeli; kullanıcılar işletim sistemi yazılımını güncel tutmalı ve güncellemeleri takip etmeli; kullanıcılar, sistemlerinde çalıştırdığı programlara dikkat etmeli ve güncellemelerini takip etmeli; elektronik posta, usb bellek veya herhangi bir web sayfasından indirilen içeriği belli olmayan çalıştırılabilir dosyaların içerikleri bir antivirüs yazılımı ile kontrol edilmeli, antivirüs yazılımı kurulmalı; özellikle yetki sahibi kullanıcılar güvenli parolalar kullanmalı ve parolalarını saklamalı; yetki gerektirmeyen komutlar yetki seviyesi yükseltilerek çalıştırılmamalı; gereksiz yetkilendirmeler yapılmamalı; yetkiler için grup yapılarının kullanımı tercih edilmeli; dosya ve dizin izinleri kullanıcı ve gruplar seviyesinde belirlenmeli; kayıt defterleri kullanılmalı ve incelenmelidir.

* Windows ve Linux işletim sistemleri üzerinden bu aktivitelerin gerçekleştirilebilmesi için işletim sisteminin sunduğu arayüzler ve komutlar kullanılmaktadır.

    * Bilgisayar sistemlerinin güvenliğini sağlayabilmek için sadece işletim sisteminin güvenliğini ele almak yeterli değildir. Bilgisayar ağlarının güvenliğini sağlayabilmek için güvenlik duvarı kullanılmaktadır.

* Güvenlik duvarları donanımsal, yazılımsal veya bulut tabanlı olabilir. Her birinin artıları ve eksileri vardır. İşletim sistemlerinde yazılımsal güvenlik duvarı vardır.

    * Güvenlik duvarları, tanımlanan kuralları kadar güçlüdür. Kuralların eksik veya yanlış tanımlanması durumunda güvenlik duvarı yazılımının yapabileceği bir şey kalmamaktadır.

* TCP/IP protokolünde, 4. katmanda, kurallar port ve ip bazlı tanımlanırlar. Gelişmiş yeteneklere sahip 7. katmanda çalışan güvenlik duvarları da vardır.

    * Sistemlerde veri kaybının yaşanmaması için yedeklemelerin planlanması ve gerçekleştirilmesi gerekmektedir.

* Yedekleme, farklı türlerde gerçekleştirilebilir. Kullanıcı verileri yedeklenebilir, işletim sistemi seviyesinde yedekleme yapılabilir veya disk bölümü tamamıyla yedeklenebilir. Bunların dışında sanallaştırma ile anlık görüntüler alınabilir. Tüm bu farklı yedekleme senaryolarının amacı, olası problemlerde veri kayıplarını en az düzeye indirmektir.

Kaynakça

(2021). Debian Manpages: https://manpages.debian.org/ adresinden alındı