Linux'a Giriş

Yazarlar:
Tobias Elsner
Thomas Erker
Anselm Lingnau

Çevirmenler:
Z.Tuğçe Şırın
Mithat Göğebakan
Samet İlhan
Serdar Orazmamedov
Özcan Zafer Ayan
Emre Doğru
Sena Özben
Margulan Byekmurat
Sema Özvin
Ahmet Dellal
Emrah Şentürk
Ebru Akagündüz

May 24, 2016

İçindekiler

List of Tables

Önsöz

Linux Essentials is a new certification by the Linux Professional Institute (LPI) which is aimed especially at schools and universities in order to introduce children and young adults to Linux. The Linux Essentials certificate is slated to define the basic knowledge necessary to use a Linux computer productively, and through a corresponding education programme aid young people and adults new to the open source community in understanding Linux and open-source software in the context of the ITC industry. See appendix C for more informaton about the Linux Essentials certificate.

With this training manual, Linup Front GmbH introduces the first comprehensive documentation for Linux Essentials exam preparation. The manual presents the requisite knowledge extensively and with many practical examples and thus provides candidates, but also Linux newcomers in general, with a solid foundation for using and understanding the free Linux operating system as well as for attaining in-depth knowledge about running and administering Linux. In addition to a detailed introduction to the background of Linux and free/open-source software, we explain the most important Linux concepts and tools such as the shell, how to handle files and scripts, and the file system structure. Insights into system administration, user and permission management and Linux as a networking client round off the presentation.

Based on the content of this training manual, Linux Essentials alumni are well-prepared to pursue further certifications including the LPI's LPIC programme as well as vendor-specific certificates like those from Red Hat or Novell/SUSE.

The training manual is particularly suitable for a Linux Essentials preparation class at general-education or vocational schools, academies, or universities, but by virtue of its detailed approach and numerous exercises with sample solutions can also be used for self-study.

This courseware package is designed to support the training course as efficiently as possible, by presenting the material in a dense, extensive format for reading along, revision or preparation. The material is divided in self-contained chapters detailing a part of the curriculum; a chapter's goals and

prerequisites are summarized clearly at its beginning, while at the end there is a summary and (where appropriate) pointers to additional literature or web pages with further information.

Additional material or background information is marked by the "lightbulb" icon at the beginning of a paragraph. Occasionally these paragraphs make use of concepts that are really explained only later in the courseware, in order to establish a broader context of the material just introduced; these "lightbulb" paragraphs may be fully understandable only when the courseware package is perused for a second time after the actual course.

Paragraphs with the "caution sign" direct your attention to possible problems or issues requiring particular care. Watch out for the dangerous bends!

Most chapters also contain exercises, which are marked with a "pencil" icon at the beginning of each paragraph. The exercises are numbered, and sample solutions for the most important ones are given at the end of the courseware package. Each exercise features a level of difficulty in brackets. Exercises marked with an exclamation point ("!") are especially recommended.

Excerpts from configuration files, command examples and examples of computer output appear in typewriter type. In multiline dialogs between the user and the computer, user input is given in bold typewriter type in order to avoid misunderstandings. The "¡¡¡¡" symbol appears where part of a command's output had to be omitted. Occasionally, additional line breaks had to be added to make things fit; these appear as "¡¡". When command syntax is discussed, words enclosed in angle brack- ets ("¡Word¿") denote "variables" that can assume different values; material in brackets ("[-f ¡file¿]") is optional. Alternatives are separated using a vertical bar ("-a—-b").

Important concepts are emphasized using "marginal notes" so they can be easily located; definitions of important terms appear in bold type in the text as well as in the margin.

References to the literature and to interesting web pages appear as "[GPL91]" in the text and are cross-referenced in detail at the end of each chapter.

We endeavour to provide courseware that is as up-to-date, complete and error- free as possible. In spite of this, problems or inaccuracies may creep in. If you notice something that you think could be improved, please do let us know, e.g., by sending e-mail to

`courseware@linupfront.de`

(For simplicity, please quote the title of the courseware package, the revision ID on the back of the title page and the page number(s) in question.) We also welcome contact by telephone, telefax or "snail mail". Thank you very much!

Bölüm 1

Bilgisayarlar, Yazılım ve İşletim Sistemleri

Amaçlar

- Temel bilgisayar donanım bilgisini edinmek
- Farklı işletim sistemlerinin farkında olmak ve bu sistemlerin farklı yönleri ile benzer yönlerini tayin etmek

Önceden Bilinmesi Gerekenler

- Temel bilgisayar bilgisi işe yarar olacaktır.

1.1 Bilgisayar da neyin nesi?

Bilgisayarın ne olduğunun ayrıntılarına girmeden önce, bilgisayar camiasının dikkate değer kişilerinden birkaç alıntı ile işe başlayalım.

> Esasında Birleşik Devletler'de, gizlenmiş araştırma laboratuvarlarında yarım düzine kadar büyük bilgisayarlardan olsaydı; bu, ülke olarak ihtiyaç duyduğumuz şeylerin çaresine bakabilirdi. Howard H. Aiken, 1952

Howard Aiken bilgisayar alanında bir öncüydü ve IBM'nin ilk bilgisayarı "Harvard Mark I"in tasarımcısıydı. Modern anlayışa göre inşaa edilmiş ilk bilgisayarlar İkinci Dünya Savaşında şifrelenmiş mesajları deşifre etmek için ya da zor hesaplamaları yapmak için yapılmıştı; büyük, karmaşık ve hataya yatkınlardı. Bugünlerde bilgisayarlarda bulunan transistör ya da entegre devreler henüz

icat edilmemişti. Bu zamanlarda aydınlığa kavuşan şey, savaşın hemen sonrasında ortaya çıkan bir takım temel varsayımlarla oluşturulan "bilgisayar" olarak kabul edilebilecek bir cihazın varlığıydı.

- Bilgisayar veriyi 'otomatik olarak' çalıştırılan komutların sırasına göre işler,
- Programlar şartlı çalışmalara ve döngülere izin vermelidir,
- Bilgisayarın çalıştırdığı programı değiştirmek veya yerine başka bir şey yerleştirmek mümkün olmalıdır

Örnek olarak, çoğu teknolojik cihaz - televizyon setlerinden dijital kameralara çamaşır makinesine ya da arabalara kadar - bugünlerde neredeyse küçük bir bilgisayar sayılabilecek programlanmış kontrol birimleri içerir. Böyle olmasına rağmen bu cihazları "bilgisayar" olarak görmeyiz, çünkü bu cihazlar sadece düzenlenmiş ve değiştirilemez programları çalıştırırlar. Buna karşın bir hesap makinesi "verileri işlemek" için kullanılabilir ama - eğer "programlanabilir bir hesap makinesi" kadar pahalı değilse - bu otomatik olarak olmaz; bir insan tuşlara basmalıdır.

1950'nin başlarında, bilgisayarlar insanların ancak araştırma kurumlarında görmeyi bekleyeceği – Aiken'in öngördüğü gibi - yüksek derecede özelleştirilmiş cihazlardı. Zamanın bilim -kurgu filmleri gizemli dönen çarklarla dolu dolapların olduğu koridorları gösteriyordu. Tam olarak 70 yıl bile olmadan bu görüntü önemli bir ölçüde değişti. [1]

> Kimsenin evinde bir bilgisayara sahip olmasına gerek yoktur. Ken Olsen, 1977

Ken Olsen bir bilgisayar üreticisi olan ve 1970'lerdeki "küçük" kavramının "havalandırmalı bir makine odasına ve güç ünitesine ihtiyaç duymayan ve bir milyon dolardan daha ucuza mal olan" anlamına geldiği zamanlarda "küçük" bilgisayarların geliştirilmesinde öncü bir firma olan DEC (Digital Equipment Corporation) [2] firmasının yöneticisi idi. Donanım teknolojisinin gelişmesiyle 1970'lerin sonuna doğru "küçük" kavramı "iki insan tarafından taşınabilir" anlamına gelmeye başladı.

DEC Linux camiası için önemlidir çünkü Unix - kendi oluşumundan 20 yıl sonra Linus Torvald'a Linux'u başlatmak için ilham veren sistem - ilk olarak DEC PDP-8 ve PDP-11 bilgisayarları üzerinde geliştirilmiştir.

[1] Bu metinde belirtilen alıntının aslında Thomas J. Watson, IBM'in CEO'su, 1943'te "Dünya pazarında sadece beş bilgisayar için yer vardır." sözleriyle birlikte söylendiği tasvir edilir. Ne yazık ki bu hiçbir zaman doğrulanamamıştır. Ve eğer 1943'te gerçekten bunu söylediyse, bu en azından 10 yıl için geçerli olurdu.

[2] DEC 1998 yılında Compaq tarafından alındı, Compaq ise 2002' de Hewlett-Packard tarafından alındı.

Ayrıca 1970'ler ilk "ev bilgisayarları"nın varlığını gördü. Bu günümüzün kişisel bilgisayarlarıyla karşılaştırılamaz çünkü insanlar evde kendileri bilgisayarlarını lehimlemek durumundaydı (ki bu zamanlarda fiziksel olarak imkansız bir durum olurdu) ve bu bilgisayarlar çok nadir olarak kullanılabilir bir klavye ve iyi bir ekran ile geliyordu. Bu bilgisayarlar genellikle bir tamircinin eski kullandığı malzemelerden yapılıyordu, bir elektrikli tren setinden geriye kalanlar gibi, çünkü gerçekte çok kullanışlı değillerdi. Buna rağmen, bizim önceden yaptığımız tanımımıza göre "bilgisayar" olarak adlandırılıyorlardı çünkü özgürce programlanabiliyorlardı, bu zahmetli bir şekilde her şeyi tuşlardan girmek ya da (eğer o kadar şanslıysanız) bir ses kaset teypinden yüklemek anlamına gelse bile. Yine de bu bilgisayarlar tümüyle ciddiye alınmıyorlardı ve Ken Olsen'in cümlesi sık sık yanlış yorumlandı. O hiçbir şekilde küçük bilgisayarlara karşı değildi (hatta işi bu bilgisayarları satmaktı). Onun anlamadığı şey bütün ev işlerinin (ısıtma, aydınlatma, eğlence ve bunun gibi şeyler) bir bilgisayar tarafından kontrol edilmesi fikriydi – bu fikir o zamanlar sadece teoride vardı ama günümüzde oldukça uygulanabilir bir durumda.

1970'lerin sonunda ve 1980'lerde "ev bilgisayarları" parçalar topluluğundan kullanmaya hazır cihazlara ("Apple II" ya da "Commodore 64" gibi isimler hala bu işin içinde olan eski üyelerimize tanıdık gelebilir) dönüştü ve ofislerde de bu cihazlardan bulunmaya başladı. IBM tarafından ilk kişisel bilgisayar 1981'de duyuruldu ve Apple ilk "Macintosh"u 1984'te piyasaya sundu. Bunların dışındakiler, söylenildiği gibi, tarihte kaldı ama bilgisayar dünyasının sadece kişisel bilgisayarlardan ya da Mac'lerden oluşmadığı unutulmamalıdır. Devasa, odaları dolduran eski bilgisayarlar hala bulunmaktadır ama bu durum gitgide azalmakta ve büyük gruplarda günümüz bilgisayarlarından oluşmaktadır. Bununla birlikte temel ilke Howard Aiken'in zamanında beri değişmedi: Bilgisayarlar hala veriyi koşul ve döngüler içerebilen değişebilir programlara göre otomatik olarak işleyen cihazlardır. Ve işler bu şekilde yürümeye devam edecek gibi görünüyor.

Alıştırmalar

1. İlk kullandığınız bilgisayar neydi? Ne çeşit işlemci içeriyordu, ne kadar belleği vardı ve sabit diski ne kadar büyüktü (eğer bir sabit diski bulunuyorduysa - eğer sabit diski bulunmuyorduysa veri nasıl kalıcı olarak depolanıyordu)?

1.2 Bir Bilgisayarın Parçaları

Hadi bir bilgisayarın (ya da daha açık olmak gerekirse IBM-uyumlu bir kişisel bilgisisayarın) içine bakma fırsatını yakalayalım ve burada bulmamızın mümkün olduğu parçaları inceleyelim:

İşlemci ("CPU" Türkçe anlamıyla "Merkezi İşlem Ünitesi") bilgisayarın çekirdeğidir: burası bilgisayarı bilgisayar yapan, program kontrolündeki verilerin işlendiği yerdir. Bugünün işlemcileri genelde birkaç "çekirdek" içerir, bunun anlamı işlemcinin ana parçaları birden fazladır ve bağımsız olarak (bağımsız işlem yapabilme bilgisayarın işlem hızını ve buna bağlı olarak performansını arttırır) işlem yapabilirler. Genelde hızlı bilgisayarlar birden fazla işlemci içerirler. Bilgisayarlar normalde Intel ya da AMD işlemcilerine (detaylarda farklılık gösterebilir ama aynı programları çalıştırabilirler) sahiptir. Tabletler ve akıllı telefonlar genel olarak ARM işlemcilerini kullanırlar. Bunlar diğerleri kadar güçlü değildirler ama enerjiyi daha az tüketirler. Intel ve AMD işlemcileri direkt olarak ARM için hazırlanmış programları çalıştıramazlar, aynı durum ARM işlemcileri için de geçerlidir.

Bellek Bir bilgisayarın çalışma belleğine "RAM" (ya da random-access memory/rastgele erişimli bellek, buradaki rastgele bir gelişigüzelliği değil isteğe bağlı erişimi ifade eder) denir. Bu bellekte sadece işlenen veriyi değil aynı zamanda çalıştırılan programın kodunu tutar.

Bu fikir Howard Aiken'in döneminden bilgisayarın öncülerinden olan John von Neumann'a aittir. Kod ve veri arasında hiç farklılığın olmadığını öngörür – bu programların bizim adresleri ya da yemek tariflerini değiştirdiğimiz gibi kodu değiştirebileceği anlamına gelir. Eski günlerde, programlar kabloların yeri değiştirilerek ya da delikli kartlar oluşturularak yazılırdı ve bu programlar değiştirilemezdi.

Bugünün bilgisayarları normalde 1 gibibyte bellek belki de daha fazlasını kullanıyor. 1 gibibyte 2**30'a eşdeğerdir, bu 1,073,741,824 [3] byte eder ve gerçekten çok büyük bir sayıdır. Karşılaştırma olarak düşünürsek: Harry Potter ve Ölüm Yadigarları kitabı her sayfa karakter, boşluk ve noktalama işaretlerinden oluşan 1700 karakterden fazla olmak üzere yaklaşık olarak 600 sayfa içerir. Bu muhtemel olarak bir milyon karakter demektir. Bununla birlikte bir gibibyte 1,000 tane Harry Potter kitabına tekabül eder ve eğer sadece genç bir büyücünün kahramanlıkları ile ilgili değilseniz 1,000 kitap inanılmaz bir kütüphane oluşturur.

Ekran kartı Çok uzak olmayan bir geçmişte insanlar bilgisayarlarda bir çıktıyı oluşturmak için elektrikli bir daktiloyu kullanabiliyorsa mutlulardı. Eski ev bilgisayarları televizyon setlerine bağlılardı ve genelde berbat denilebilecek resimler oluştururlardı. Diğer elde ise bugün en basit "akıllı telefonlar" bile oldukça etkileyici grafikler sunuyorlar ve şu anda kullanılan kişisel

[3] İnsanlar genelde bunu "gigabyte" olarak söyler ama gigabyte normalden yüzde 7 daha azdır.

bilgisayarlardaki ekran kartları 1990'larda [4] pahalı bir spor arabaya ve küçük bir ev değerinde bir maliyete sahip olurdu. Bugünün sloganı "3D hızlandırma", ki bu durumda ekran gerçekte 3D olarak çalışmaz (ki bu bile gitgitde moda olmaya başladı) bilgisayarın içinde işlenen grafikler sadece sağ, sol, yukarı ve aşağıyı içermez – bilgisayar ekranında görülebilen yönler- aynı zamanda ön ve geri kısımları içerir. Görüntü gerçeklik oyunları için bir canavarın bir duvarın önünden mi arkasından mı çıkacağı çok önemlidir bununla birlikte görünür olsun ya da olmasın modern ekran kartlarının amacı bilgisayar işlemcisini diğer işler için serbest bırakmaktır. Güncel ekran kartları kendi işlemcilerine sahiptir, bunlar bilgisayarın kendi işlemcilerinden daha hızlı çalışırlar ama onlar kadar genel olarak kullanışlı değildirler.

Çoğu bilgisayar ayrı bir ekran kartı içermez çünkü bu bilgisayarların grafik donanımları işlemcinin bir parçasıdır. Bu bilgisayarı daha küçük, ucuz, sessiz yapar ve bunlar enerjiyi daha iyi kullanır ama grafik performansı çok iyi değildir. Eğer en yeni oyunları oynama bağımlısı değilseniz bu sizin için gerçek bir sorun teşkil etmez.

Anakart Anakart genel olarak dikdörtgen biçimindedir, bilgisayarın işlemcisinin, belleğinin ve grafik kartının takılı olduğu ve örneğin sabit disklerin, yazıcıların, bir klavyenin ve farenin ya da ağ kablolarının ve elektronik olarak gerekli olan her şeyin kontrolcüsünün bulunduğu bir levhadır. Bilgisayarlarda bulunan anakartlar birçok çeşitli boy ve renklerde [5] olabilir, örneğin oturma odasında bulunan bir video kaydedici olarak kullanılan küçük ve sessiz bir bilgisayarın anakartı ile çok fazla RAM'e ihtiyaç duyan ve birden fazla işlemcisi bulunan büyük sunucuların anakartları farklı boyut ve renklerde olabilir.

Güç Kaynağı Bir bilgisayar çalışmak için elektriğe ihtiyaç duyar, ne kadar elektriğe ihtiyaç duyduğu sahip olduğu bileşenlere bağlıdır. Güç kaynağı 240 V AC kaynağını bilgisayarın ihtiyaç duyduğu daha düşük değerlerdeki DC voltajına indirmek için kullanılır. Güç kaynağı bütün parçalar için yeterli elektriği üretecek şekilde seçilmelidir. Güç kaynağının bilgisayara verdiği elektriğin çoğu er ya da geç ısıya dönüşecektir bu yüzden bilgisayarlarda soğutma çok önemlidir. Basit tasarımlarda, genelde bir ya da iki fan pahalı elektronik parçalara hava üfler ya da sıcak havayı dışarı atar. Uygun bir tasarımla bilgisayarları fana ihtiyaç duymayacak şekilde yapmak mümkündür bu şekilde bilgisayarlar çok sesssiz çalışırlar ama bu tür bilgisayarlar hem oldukça pahalı hem de çok hızlı değillerdir (İşlemci ve ekran kartların hızlı olması genelde sıcak olması anlamına gelir).

[4] Bu arada bütün teşekkürler harika bilgisayar oyunlarının tükenmeyen popülerliğine gitsin. Kim bilgisayar oyunlarının işe yaramaz olduğunu düşünüyorsa bir dakikasını bunun üstünde düşünerek geçirmelidir.

[5] Gerçekten! Ama yinede kimse bilgisayarının anakartını rengine göre seçmemelidir.

Sabit Diskler Bilgisayarın belleği anlık çalışan süreçlerin (belgeler, dökümanlar, web sayfaları, geliştirilen programlar, müzik ve videolar,. . . - ve tabiki verinin üzerinde çalışan programlar) verilerinin depolandığı bir yerken kullanılmayan veriler sabit diskte depolanır. Bunun ana sebebi sabit disklerin genelde bilgisayarların belleklerinden çok daha fazla veri depolayabilmesidir – günümüzde sabit diskler terabytelar cinsinden ölçülmektedir.

Boyuttaki bu genişleme hızda yavaşlama ile doğru orantılı ilerler. Bellek erişim zamanı nanosaniyeler ile ölçülürken sabit disklerde bu durum milisaniyelerle ölçülür. Bu, bir metre ile 1000 kilometre arasındaki farka eşdeğerdir.

Geleneksel olarak, sabit diskler manyetik madde içeren dönen plakalar içerir. Okuma / yazma kafası bu maddeyi değişik yerlerden manyetize edebilir ve depolanmış verileri okuyabilir. Bu plakalar dakikada 4,500 ile 15,000 defa döner ve okuma/yazma kafası ile plaka arası bir dakikadır (en fazla3 nanosaniye). Bu sabit disklerin çok hassas olduğu anlamına gelir, çünkü eğer okuma/yazma kafası disk hala çalışırken plaka ile iletişime geçerse kesintiye uğramış kafa kırılır ve disk bozulur.

Taşınabilir bilgisayarlar için olan son moda sabit diskler, bilgisayarın düştüğünü anlayabilecek ve o anda sabit diskte oluşabilecek zararı önlemek için bilgisayarı kapatacak hızlandırılmış sensörlere sahiptirler.

Son moda SSD'ler ya da katı-durum diskleri, manyetik plakalar kullanmak yerine depolamak için "flash belleği" kullanır – bu içindekileri elektrik olmadan tutan bir bellek çeşididir. Katı-durum diskleri manyetik sabit disklerden daha hızlıdır fakat görece olarak daha pahalıdırlar. Bununla birlikte hareket eden parçaları yoktur, düşmeye dayanıklıdırlar, sabit disklere göre enerjiyi tasarruflu kullanırlar. Bu özellikleri taşınabilir bilgisayarlar için çok iyidir.

Flash bellekler belirli sayıda yazma işlemine tabii tutulabilir. Ölçümler bunun pratikte bir sorun teşkil etmediğini göstermiştir.

Bir sabit diski (manyetik ya da SSD) bilgisayara bağlamak için çeşitli yöntemler vardır. Şu anda en kullanılanı "serial ATA" (SATA) olarak adlandırılır, ayrıca "IDE" olarak da bilinir. Ayrıca sunucular SCSI ya da SAS disklerini kullanırlar. Harici diskler için, USB ya da eSATA (SATA' nın sağlam bağlantısı noktası olanı) kullanılabilir.

Sözü açılmışken: Gigabytelar ile gibibytelar (ya da terabytelar ile tebibytelar arasında) arasındaki fark en çok sabit disklerde fark edilir. Örneğin 100 GB disk alırsınız ama bilgisayarınıza bağladığınızda bir bakarsınız 93 GB görünüyor?! Bununla birlikte diskiniz arızalı değildir (çok şanslısınız) disk sürücüsünü üreten firma gigabyteları kullanırken bilgisayarınız muhtemel olarak alanları gibibyte cinsinden hesaplıyordur.

Optik Sürücüler Sabit disklerin dışında bilgisayarlar genelde okuma ve genelde yazma işlemlerini yapabilen optik sürücüleri içerirler. (CD-ROM,

DVD-ROM ve Blu-ray diskler) Mobil cihazlar bazen optik sürücüler için yeterli fiziksel hacme sahip olmazlar ama bu optik sürücüleri desteklemedikleri anlamına gelmez. Optik medyalar – isim verilere erişmek için kullanılan lazerden gelir- genellikle yazılım ve ürünlerin (müzik ya da filmlerin) dağıtımında kullanılırlar. Bu medyaların dağıtımı için internet etkin olarak kullanılmaya başlandığı için optik medyaların önemi azalmaktadır. Eski zamanlarda optik medyalar yedekleme için kullanılıyordu ama bugünlerde bu çok anlamsızdır çünkü CD-ROM ortalama olarak 700 MB lık ve DVD-ROM ortalama olarak 9 GiB veri tutabilirler. Bu sebeple 1 TBlık bir veri için tam bir yedekleme 1000 CD ya da 100 DVD gerektirir. (Blu-ray diskler 25 GB'a kadar veri tutabilir fakat blu-ray diskleri okuyan okuyucu mekanizmalar oldukça pahalıdır.)

Ekran Eski filmlerde hala yeşil renkli bilgisayar ekranlarını görebilirsiniz. Gerçekte yeşil ekranların hepsi ortadan kaybolmuş durumdadır, renkler oldukça gözde bir durumdalar ve yeni ekranlar eskiden sahip olduğumuz CRT'ler (tüplü monitörler) gibi devasa değiller. Sıvı kristal (LCD) teknolojisine dayanan ince ve şık monitörlere sahibiz artık. LCD'ler kendilerini masada daha az yer kaplamanın avantajıyla sınırlamıyorlar, aynı şekilde ekran ışıkları titreşmiyor ve kullanıcıları muhtemel zararlı radyasyonlara maruz bırakmıyor. Buna her yönden kazanma durumu da diyebiliriz. Değişik açılardan bakıldığında renk değişikliklerinin olması ve ucuz cihazlarda kötü ışık durumları birkaç dezavantajı bulunur.

Tüplü monitörlerde kullanılmayan bir resmin uzun bir süre ekranda kalmamasına dikkat edilirdi çünkü resim ekrana yazılabilir ve kalıcı bulanık bir fon olarak ekranda kalabilirdi. Ekran koruyucular belirli bir süre ekran kullanılmadığında şirin bir canlandırma ekrana gelir ve bu "yazılma" sorununu engellerdi. (klasik olan bir akvaryum balığıydı). LCD' ler artık bu sorunu yaşamamaktadır ama ekran koruyucular hala dekoratif olarak bulunmaktadır.

LCDler akıllı telefon boyutundan duvar boyutuna kadar bütün boyutlarda bulunabilir, en önemli özelliği çözünürlükleridir, bilgisayarlar genelde 1366 x 768 (yatay x dikey) ile 1920 x 1080 arasında bir çözünürlük sunarlar. (Daha düşük ve daha yüksek çözünürlükler mümkündür ama ekonomik ya da görsel olarak gerekli değillerdir.) Bilgisayarların büyük bir kısmı genişletilmiş çalışma ortamlarında birden fazla ekranı desteklerler.

Ayrıca bugün genel olarak yüksek çözünürlüklü televizyona karşılık gelen 16:9 ölçüsü bulunur – bilgisayarların büyük bir kısmı televizyon izlemek için kullanılmadığı için aslında bu saçma bir gelişmedir. Daha uzun ama dar ekranlar (4:3 gibi formatlar) sık kullanılan programlar için daha uygun olurlar.

Diğer Bileşenler Bilgisayara bizim anlattığımız bileşenlerden daha fazlasını bağlamanız mümkündür. Yazıcılar, tarayıcılar, kameralar, televizyon alıcıları, modemler, robotik kollar, komşularınızı rahatsız edecek küçük füze atıcılar ve

bunun gibi şeyler. Bu listenin gerçekten bir sonu yok ve bütün değişik cihazları burada anlatamayız. Yine de bu birkaç gözlem yapamayacağımız anlamına gelmez.

- Bir tane güzel yeniliklerden biri, örnek olarak, bağlantıların basitleştirilmesidir. Neredeyse bütün değişik sınıftaki cihazlar kendi arayüzlerini kullanıyorken (yazıcılar için paralel arayüzler, modemler için seri arayüzler, klavye ve fareler için PS/2 arayüzü, tarayıcılar için SCSI, ...) bugünlerde çoğu cihaz USB'yi (universal serial bus) kullanıyor. USB kısmen güvenilir ve kabul edilebilir bir hıza sahip olmakla birlikte bilgisayar çalışırken tak-kullan özelliğini destekler.

- Bir diğer yenilik bileşenlerin kendi içlerinde daha akıllı olması ile ilgili. Önceden pahalı yazıcılar bile elektrikli daktiloların kabul edilebilir derecede IQ derecesine sahip olan aptal cihazlardı ve programcıların istenen doğru çıktıyı alabilmek için yazıcıya tam olarak doğru kodu oldukça dikkatli bir şekilde göndermesi gerekiyordu. Bugün yazıcılar (en azından iyi yazıcılar) programcılar için daha az sorun yaratacak şekilde kendi desteklediklerı özgün programlama dillerine sahipler ve kendi içlerinde bir bilgisayar sayılabilirler. Bu durum çoğu diğer bileşen için aynıdır.

Alıştırmalar

1. Bilgisayarınızın içini açın (tercihen bir öğretmen ya da bir ebeveyn gözetiminde ve önce elektrik bağlantılarını kapatmayı unutmayın!) ve işlemci, bellek, anakart, ekran kartı, güç kaynağı ve sabit disk gibi önemli parçaları bulmaya çalışın. Bilgisayarınızın hangi parçalarından burada bahsedilmedi?

1.3 Yazılım

Bilgisayarın donanımı, içerdiği teknik parçalar, önemli olduğu gibi yazılımı da [6], çalıştırılabilir programlar, oldukça önemlidir. Bu yaklaşık olarak üç kategoriye bölünebilir.

Firmware bilgisayarın anakartında depolanır ve eğer uygun değilse değiştirilip yerine başka bir firmware kullanılabilir. Bu bilgisayarı açtıktan sonra bilgisayarı tanımlı olan bir duruma getirmek için kullanılır. Genelde saati ayarlamaya yarayan ya da anakarttaki bazı özellikleri açıp kapatmaya yarayan bir kurulum modunu uyandırmak için bir yol bulunur.

Kişisel bilgisayarlarda bu yazılım BIOS (Temel Girdi/Çıktı Sistemi) olarak adlandırılır, daha yeni sistemlerde bunun adı EFI olarak geçer.

[6] Donanımın tanımı: "Bilgisayarın vurulabilen parçalarına denir." (Jeff Pesis)

Bazı anakartlar normal Linux'dan daha hızlı açılan küçük bir Linux sistemi yüklü olarak gelirler, bu sadece Windows'u açmaya gerek kalmadan internette gezinme ya da DVD izleme gibi sınırlı işlevler görür.

İşletim sistemleri bilgisayarı kullanılabilir bir cihaz yapar: İşletim sistemi bellek, sabit disk, ayrı çalışan programların işlemcide kullanabilecekleri zaman ve diğer bileşenlere erişim gibi bilgisayar kaynaklarının kullanımının nasıl olacağını yönetir. Programların başlamasına ve durmasına izin verir ve aynı bilgisayarda olan farklı kullanıcılar arasında bir ayırıcı görevi görür. Bunların dışında giriş seviyesinde bilgisayarın yerel bir ağa ya da internete katılmasını sağlar. İşletim sistemi görsel bir arayüz sunar ve bu kullanıcıların bilgisayarın nasıl çalıştığı ile ilgili bir fikir verir.

Yeni bir bilgisayar aldığınızda bu genellikle önceden kurulmuş bir işletim sistemi ile gelir: Kişisel bilgisayarlar Microsoft Windows, Mac bilgisayarlar OS X, akıllı telefonlar genelde Android (bir Linux türevi) kullanır. Bununla birlikte işletim sistemi donanıma firmware gibi bağlı değildir ve herhangi birisi bir yenisiyle değiştirilebilir. Örneğin çoğu kişisel bilgisayarlara ve Mac bilgisayarlara Linux kurabilirsiniz.

Ya da Linux'u var olan bir işletim sisteminin yanına ek olarak da kurabilirsiniz, bu bir sorun yaratmaz.

Kullanıcı-seviye programlar kullanışlı bir şeyler yapmanızı sağlarlar. Bu belge yazmak, resim çizmek ya da değiştirmek, müzik bestelemek, oyun oynamak, internette gezinmek ya da yeni bir yazılım geliştirmek olabilir. Bu programlar uygulamalar olarak adlandırılırlar. Bunlara ek olarak genelde işletim sisteminin size sağladığı bazı araçlar bulunur, bunlar bilgisayar ayarlarında değişiklik yapmanıza izin verirler. Sunucular genelde diğer bilgisayarlara web, posta ya da veritabanı gibi bazı hizmetler sağlarlar.

1.4 En Önemli İşletim Sistemleri

1.4.1 Windows ve OS X

İşletim sistemi denildiğinde çoğu insanın aklına Microsoft Windows [7] geliyor otomatik olarak. Bu günümüzde bilgisayarlarının çoğunun Windows yüklü olarak satılmasından kaynaklanıyor. Bu kendi başına kötü bir şey sayılmaz, çünkü bilgisayar sahipleri ilk sistem yükleme işlemi için uğraşmak zorunda kalmazlar. Ama olaya başka bir açıdan bakarsak bu Linux gibi alternatif sistemlerin tanınmasında bir sorun yaratıyor.

Aslında bilgisayarı Windows yüklü olarak almak o kadar da zor değildir çünkü bilgisayarınızı Linux ile kullanmak isteyebilirsiniz ama bu durumda sıfırdan bir sistem inşa etmek durumunda kalırsınız. Teoride bakıldığında kullanılmamış bir Windows için bilgisayar üreticisinden paranızı geri alabilirsiniz

[7] Çoğu insan başka işletim sistemi olduğunun farkında bile değildir.

ama hepimiz biliyoruz ki bugüne kadar parasını geri almayı başarabilen çok fazla kişi olmadı.

Günümüzde kullanılan Windows 1990'lı yılların standartlarına göre oldukça iyi bir işletim sistemi olan "Windows NT"nin varisidir (Windows 95 gibi önceki sürümler mevcut bulunan Microsoft işletim sistemi MS-DOS'un grafiksel bir eklentisiydi ve o günün şartlarına göre oldukça ilkel sayılırdı). Nezaket Windows'u kritik bir şekilde eleştirmemize engel oluyor, ama Windows'un bir insanın işletim sisteminden beklediği her şeyi, grafik kullanıcı arayüzü ve çoğu cihazı destekleme özelliği gibi, ortalama olarak karşıladığını söyleyebiliriz.

Apple'ın Macintosh'u 1984'te piyasaya sürüldü ve o zamandan beri Mac OS olarak adlandırılan bir işletim sistemini kullanıyor. Yıllardan beri, Apple tabanda birçok değişiklik yaptı (günümüzün Mac'leri teknik olarak Windows bilgisayarlar ile aynı) ve bu değişikliklerin bazısı oldukça radikal oldu. 9 numaralı sürümüne kadar (9 da dahil olmak üzere) Mac OS zayıf bir yapıydı, örneğin aynı anda çalışan birden fazla programa yetersiz derecede destek olabiliyordu. Günümüzdeki Mac OS X deki X bir Romen "10" rakamı (X karakteri değil) ve bu sistem temelde BSD Unix özelliklerini taşıyor.

Şubat 2012'den beri, Macintosh işletim sisteminin resmi ismi "Mac OS X" den "OS X" e değişti. Eğer "Mac OS"un gitmesine izin verirsek ne demek istediğimiz daha iyi anlaşılabilir.

Windows ve OS X arasındaki büyük fark, OS X'in Apple bilgisayarları ile özel olarak satılması ve "normal" bilgisayarlarda çalışmıyor olması. Bu durum Apple'a çok kararlı bir sistem oluşturmayı sağlıyor. Bunun dışında Windows neredeyse bütün bilgisayarlarda çalışıyor ve hiç görülmeyecek şekillerde birleşebilecek çok geniş bir donanıma destek sunuyor. Bu yüzden Windows kullanıcıları bazen çözülmesi zor ya da bazen çözülmesi imkansız uyumsuzluk sorunları ile karşılaşabilirler. Buna rağmen Windows tabanlı bilgisayarlarda çok geniş bir donanım seçme imkanı bulunur ve böylece fiyatlar daha uygun seviyeye iner.

Windows ve OS X'in benzer noktaları ikisininde patentli bir yazılım olması: Kullanıcılar Microsoft ya da Apple önlerine neyi koyarsa onu kabul etmek zorundalar ve değil sistemi değiştirmek sistemin kendisini inceleyemezler bile. Bir güncelleme sistemine bağlıdırlar ve eğer üretici bir şeyi siler ya da başka bir şey ile değiştirirse buna uyum sağlamak zorundadırlar.

Burada bir fark bulunur: Apple genel olarak bir donanım üreticisidir ve OS X'i sadece Mac bilgisayarları alanlara sağlar (bu yüzden OS X Mac-olmayan bilgisayarlar için değildir). Diğer taraftan Microsoft bilgisayarları üretmez, sadece bilgisayarlarda çalışacak Windows gibi yazılımları satarak para kazanır. Bununla birlikte, Linux gibi bir işletim sistemi Apple'dan çok Microsoft'a bir tehdit oluşturur çünkü Apple alan insanların çoğu Apple bilgisayarı (tüm paketi) istedikleri için alırlar sadece OS X ile ilgilendikleri için değil. Bilgisayar dünyası, tablet ve diğer yeni moda Windows çalıştırmayan bilgisayarların

istilası altına girdi ve bu Microsoft'u büyük bir baskı altına sokuyor. Apple bu durumdan Mac'ler yerine iPhone ve iPad'leri satarak kolayca kurtulabilir ancak Windows olmadan Microsoft iflasın eşiğine gelebilir. [8]

1.4.2 Linux

Linux ilk başta Linus Torvalds tarafından bir merak duygusu ile başladı fakat sonra kendi hayatını kurmaya devam etti. Bu zamanlarda yüzlerce geliştirici (sadece öğrenci ve bu işle hobi için uğraşan insanlar değil ayrıca IBM, Red Hat ve Oracle gibi firmalardaki profesyoneller) tarafından geliştirildi.

Linux'ta 1970'lerde AT&T'nin Bell Laboratuvarında geliştirilen ve küçük (küçük tanımı için yukarıyı inceleyin) bilgisayarlar için tasarlanan bir işletim sistemi olan Unix'ten esinlenilmişti. Unix kısa sürede araştırma ve teknoloji alanında tercih edilen bir sistem olmuştu. Büyük bir bölümde Linux Unix'in tasarımını ve temel fikirlerini kullanmaktadır ve Unix yazılımını Linux üzerinde çalıştırmak oldukça kolaydır fakat Linux'un kendisi Unix kodlarını içermez ve bağımsız bir projedir. Windows ve OS X'in tersine Linux ekonomik olarak ayrı bir şirket tarafından desteklenmiyor. Linux özgürce alınabilir ve oyunu kuralına göre oynayan herkes tarafından kullanılabilir. (bir sonraki bölümde belirtildiği gibi) Bütün bunlarla birlikte Linux sadece kişisel bilgisayarlar üzerinde değil, telefonlardan (en popüler akıllı telefon işletim sistemi, Android, bir Linux türevidir) en büyük anaçatı bilgisayarlara (dünyanın en hızlı 10 bilgisayarının hepsi Linux üzerinde çalışır) kadar çoğu sistemde çalışır ve bu Linux'u modern bilgisayar tarihinde en esnek işletim sistemi yapar.

Linux tek başına bir işletim sistemi çekirdeğidir, uygulamaların ve özelliklerin kaynak kullanımını ayarlayan bir programdır. Uygulamaları olmadan gelen bir işletim sistemi çok kullanışlı olmadığından, insanlar genelde bir Linux dağıtımını yükler. Dağıtım, kararlı bir Linux'a sahip belirli uygulama, özellik, belgeleme ve diğer kullanışlı özellikleri barındıran bir pakettir. Buradaki güzel olan şey, Linux'un kendisi gibi, çoğu Linux dağıtımı özgürce kullanılabilir bir durumdadır ve ücretsiz ya da çok düşük bir fiyatla kullanılabilirdir. Bu şekilde binlerce dolarlık Windows ve OS X lisansı almadan ve lisans kısıtlamalarına maruz kalmadan bütün bilgisayarlarınıza, Ayşe teyzenizin ve arkadaşlarınız Zeynep ve Emre'nin bilgisayarlarına rahatça Linux dağıtımlarından istediğinizi kurabilirsiniz.

Linux ve Linux dağıtımları ile ilgili daha fazla bilgiyi ikinci bölümde bulabilirsiniz.

[8] Aslında gerçek savaş alanı Windows değil Office'dir – çoğu insan Windows' u hayranlıklarından kullanmaz, eğlence için kullanılabilen (ucuz) bir işletim sistemi olduğu için kullanır ve bu bilgisayarlar Office'i çalıştırır– fakat aynısı Apple ve Google'ı yer değiştirirsek de olur. Bir gerçek olarak, Office ve Windows Microsoft'un para kazanabildiği yegane ürünlerdir; bunun dışındaki Microsoft'un yaptığı her şey (muhtemelen Xbox oyun konsolu hariç) bir kayıptır.

1.4.3 Farklılıklar ve Benzerlikler

Aslında bu üç büyük işletim sistemi sadece kullanıcıya sundukları şeyler konusunda sadece detaylarda farklılık gösterirler. Bu üç sistemde kullanıcıya herkesin kullanabileceği şekilde dosyaları "sürükle ve bırak" şeklinde özelliklere sahip olan bir grafiksel kullanıcı arayüzü sunar. Çoğu popüler uygulamalar bu üç sistem içinde mevcuttur, bu yüzden zamanınızın çoğunu internet tarayıcısı, ofis uygulamaları ya da bir e-mail programında harcadığınız için aslında hangisini kullandığınızın bir önemi kalmaz. Bu bir avantajdır çünkü bu bir sistemden diğerine geçişi mümkün kular.

Grafiksel arayüz dışında, üç sistemde metin şeklinde girilen komutların sistem tarafından çalıştırılabileceği bir çeşit "komut satırı" sunar. Windows ve OS X'de bu özellik genelde sistem yöneticileri için kullanılabilir durumdadır, normal bir kullanıcının bunu kullanmaması gerektiği yönünde bir düşünce bulunur. Linux'ta komut satırı daha az saklanmış vaziyettedir, bu Unix'in bilimsel/teknolojik felsefesi nedeniyle olabilir. Aslında bir gerçek olarak, çoğu görev Linux'un (ve bir açıdan da OS X'in) sağladığı güçlü araçlarla birlikte komut satırından daha etkili bir şekilde çalıştırılabilir. Yeni bir Linux kullanıcısı olarak sizde komut satırını açıp, bunun güçlü ve zayıf yönlerini öğrenmelisiniz, grafiksel arayüzün güçlü ve zayıf yönlerini öğrenmeniz gerektiği gibi. İkisinin karışımı size oldukça büyük bir çok yönlülük kazandıracaktır.

Alıştırmalar

1. Eğer Windows ve OS X gibi patentli bir işletim sistemi ile deneyiminiz olduysa: En sık hangi uygulamaları kullanıyorsunuz? Bunlardan hangileri özgür yazılım?

1.5 Özet

Bugünün kişisel bilgisayarları, ister Linux tabanlı olsun ister Windows ya da OS X donanımları, temel konseptleri ve kullanım amaçları düşünüldüğünde farklılıktan çok benzerliklere sahipler. Hiç şüphe duymadan günlük işlerinizi yapmak için bu üç sistemden birini kullanabilirsiniz, hiçbiri açıkça ve itiraz edilemez bir şekilde "en iyi" değil. Bununla birlikte, bu kitap daha çok Linux hakkında, kalan sayfalar boyunca sistemin kullanımı ile ilgili olabilecek en fazla bilgiyi sağlamaya çalışacağız, sistemin gücüyle ilgili olan kısımları ortaya koyacağız ve zayıf olan yönlerinden bahsedeceğiz. Şimdilik Linux diğer iki sisteme karşı ciddi bir alternatif ve diğerlerinden değişik açılardan -özellikle bazı yönlerden- daha üstün. Sizin Linux ile ilgilendiğinizi görmekten mutluyuz ve umarız Linux'u öğrenirken, pratik yaparken ve kullanırken eğlenirsiniz. Eğer LPI'nin Temel Seviye Linux sertifikası ile ilgileniyorsanız sınavda başarılar dileriz!

Özet

- Bilgisayarlar verileri otomatik olarak çalıştıran, koşullu çalışmaya ve döngülere izin veren, değiştirilebilir programları barındıran cihazlardır.
- Bilgisayarın en önemli parçaları işlemci, bellek, ekran kartı, anakart, sabit disk ve buna benzer parçalardır.
- Bilgisayarda bulunan yazılımlar firmware, işletim sistemi ve kullanıcı seviyesi programlar olarak üç gruba ayrılabilir.
- En popüler işletim sistemi Microsoft Windows'tur. Apple bilgisayarları OS X olarak adlandırılan başka bir işletim sistemi kullanır.
- Linux bilgisayarlar için alternatif bir işletim sistemidir ve tek bir şirket tarafından değil çok sayıda gönüllü tarafından geliştirilir.
- Linux dağıtımları Linux işletim sistemi çekirdeğini uygulamalar ve belgelendirmeler ile kullanılabilir bir sisteme dönüştürür

Bölüm 2

Linux ve Özgür Yazılım

Amaçlar

- Linux ve özgür yazılımın temel ilkelerini öğrenmek
- Temel Özgür ve Açık Kaynak Kodlu Yazılım lisanslarını öğrenmek
- En önemli özgür uygulamalar hakkında bilgi sahibi olmak
- En önemli Linux dağıtımları hakkında bilgi sahibi olmak

Önceden Bilinmesi Gerekenler

- Bilgisayarlar ve işletim sistemleri hakkında temel bilgiler (Bölüm **??**)

2.1 Linux: Bir Başarı Öyküsü

Linux Torvalds[1], 1991 'in yazında, daha 21 yaşındayken Finlandiya Helsinki Teknik Üniversitesi 'nin bilgisayar bilimleri bölümünde öğrenim görüyordu. O zamanlar, yeni 386 'sı ile denemeler yapmak istiyordu ve işletim sistemi olmadan direk donanım üzerinde çalışan bir terminal emülatörü yazarak (Bu ona üniversitesindeki Unix sistemine erişimini sağlamıştı) kendisini oyalıyordu. Bu program zamanla ilk Linux işletim sistemi çekirdeğine dönüştü.

O zamanlarda Unix zaten 20 yaşındaydı ve üniversiteler ile araştırma kurumlarının seçtiği işletim sistemleri Unix'in birer çeşidiydiler.

[1]Linus Torvalds, etnik olarak bir Fin 'dir.(2010 'un Eylül ayında Amerikan vatandaşı olmuştur.) İsveççe konuşan bir azınlığın üyesidir. Kolayca telafuz edilebilen bir isme sahip olmasının nedeni budur.

Unix'in kendisi de aynen Linux gibi Ken Thompson ile Dennis Ritchie'nin Bell Laboratuvarlarındaki (AT&T nin araştırma kurumu) hobi amaçlı projelerinden biri sonucunda oluşmaya başladı. Çok kısa zamanda çok kullanışlı bir sisteme dönüştü ve büyük bir bölümü yüksek seviye bir dille yazıldığı için (C) oldukça kısa bir zamanda üzerinde geliştirildiği PDP-11 den başka bilgisayarlara da taşınabiliyordu. Ek olarak 1970 'lerde, sistemler çok küçük olduğundan ve bir derece basit olduğundan üniversitelerin çoğunda ders olarak işlenmeye başladı.

1970'lerin sonlarına doğru Berkeley'deki California Üniversitesi Unix'i VAX'a port etti, PDP-11'in mirasçısı, çeşitli değişiklik ve iyileştirmelerin yapıldığı bu sistem BSD(Berkeley Software Distribution) olarak duyuruldu. BSD türevleri hala bulunmaktadır.

Linux'un ilk versiyonunu geliştirmek için Linus, Minix'ten yararlandı. Minix Amsterdam Özgür Üniversitesindeki Andrew S. Tanenbaum tarafından öğretim amaçlı yazılmış bir sistemdir. Minix küçük tutulmuştu ve özgürce ulaşılabilir değildi. Bu yüzden ciddi bir işletim sistemi olarak tanınmamıştır[2].

25 Ağustus 1991' de Linus projesini herkese duyurdu ve dünyanın kalanını katılması için davet etti. Proje, Minix için alternetif bir işletim sistemi çekirdeği olarak görev yaptı.

O zamanlarda sistem bir isme sahip değildi. Linus "Freax"(Freak ve Unix'in karışımı olan bir kelime) diyordu. İlk başta Linux olarak tanıtmıştı ama sonra bunun çok egoistçe olduğunu düşündü. Linus' un sistemi üniversitenin ftp sistemine yüklendiğinde, Freax ismini beğenmeyen Linus' un okuldan arkadaşı Ari Lemmke ismi Linux olarak değiştirdi. Linus sonradan bu değişikliği onayladı.

Linux dikkate değer bir ilgiyi ve yardım edecek çok fazla gönüllü buldu. Linux 0.99 Aralık 1992'de GPL'nin ilk versiyonu altında lisanslandı ve Unix işlevselliğine sahip kullanılabilir bir işletim sistemi olmuş oldu.

Linux 2.0 1996 'nın başlarında ortaya çıktı. Çoklu işlemci desteği ve çalışma zamanında çekirdek modüllerini yükleme gibi bir çok önemli yenilikle birlikte geldi.

Bir diğer önemli değişiklik ise, Linux' un maskotu penguen "Tux" oldu. Linus Torvalds Avustralya' da bir penguenle karşılaşmış ve bundan çok etkilenmişti. Sarı ayakları ve gagasıyla oturan simgesel penguen Larry Ewing tarafından çizilmiş ve camiaya sunulmuştur.

Linux 2.6 gelişim sürecinde yeni bir yapılanmaya gitti. Önceki sürümlerde son kullanıcıya uygun sürümlerle geliştirici sürümleri ayrı olarak sunuluyordu. Linux 2.6'dan itibaren geliştirici çekirdekleriyle normal olanlar arasındaki çizgi kalktı ama gelecek sürümdeki geliştirmeler önceden sunulup resmi bir sürüm sunulmadan ayrıntılı bir şekilde test edildi.

[2]BSD zamanında özgürce kullanılabilir değildi. Ozamanlar kolayca erişilebilir olsaydı, belkide Linux 'a hiç başlanmamış olacaktı.

Bu yaklaşık olarak şu şekilde oluyor: Linux 2.6.37 yayınlandıktan sonra, Linus sonraki çekirdek için önerilen değişiklik ve gelişmeleri topladı ve kendi resmi sürümü olan Linux 2.6.38-rc1 sürümünü çıkardı. Bu sürüm çeşitli insanlar tarafından test edildi ve bütün değişimler ve geliştirmeler 2.6.38-rc2 sürümünde toplandı. Sonunda kod resmi olarak yayınlanacak şekilde kararlı bir hale geldi ve Linux 2.6.38 olarak yayınlandı ve bu işlem 2.6.39 ile devam etti.

Linus' un resmi sürümlerine ek olarak başka geliştiriciler tarafından geliştirilen Linux sürümleri bulunmaktadır. Örnek olarak Linus'un kendi sürümüne eklenilebilecek kadar iyi olduğu düşünülen yeni aygıt sürücülerinin geliştiği (birkaç geliştirme aşamasından sonra) "kademelendirilmiş ağaç" bulunmaktadır. Bir kere yayınlandıktan sonra Linux çekirdekleri belirli bir süre için düzeltmeleri kabul eder, bu yüzden 2.6.38.1, 2.6.38.2 ve bunun gibi sürümler bulunabilir. Temmuz 2011' de Linus 2.6.40 olarak hazırlanan sürümü Linux 3.0 olarak duyurdu. Bu çok büyük değişiklikler olmamasına rağmen numaralama işlemini basitleştirmek için yapılmış bir değişimdi.

Bugünlerde kullanıcı için yapılan çekirdek sürümleri 3.2-rc1 olarak adlandırılmaktadır bu yüzden düzenlemelerle gelen sürümler 3.1.1, 3.1.2,... olarak adlandırılırlar. "Linux" projesi bugün hiçbir şekilde bitmiş değildir. Linux dünya çapında programcılar tarafından sürekli geliştirilmekte ve bu durumdan memnun olan milyonlarca özel ve ticari kullanıcıya hizmet vermektedir. Bilgisayar endüstrisi içersinde Linux çekirdeği üzerinde önemli pozisyonlarda çalışan birçok kişi var ve bazıları çevresindeki en saygın profesyonel geliştiricilerdir.

Linux işletim sistemi doğası gereği çok yönlü donanım desteğine sahiptir, kesin bir kanıya varmamakla birlikte bütün platformlarda çalıştığı iddia edilebilir (akıllı telefonlar ve büyük sistemler de dahil olmak üzere) ve aynı zamanda Intel PC platformlarının örnek oluşturduğu donanım sürücüleriylede uyumludur. Linux sanayi ve akademi alanında yeni bir işletim sistemi geliştirmek için araştırmalara konu olan mevcut işletim sistemleri içersinde şüphesiz en yenilikçi olanıdır.

Çok yönlü olması Linux'u sanallaştırma ve "Bulut bilişim" gibi uygulamalar için tercih edilen işletim sistemi yapar. Sanallaştırma kendi işletim sisteminiz üzerinde çalıştırdığınız programlar yardımıyla tek bir gerçek ("fiziksel") bilgisayarda pek çok "sanal" makina kurup gerçek bilgisayarlar gibi kullanmayı mümkün kılar. Bu işlem kaynakları daha verimli kullanmamızı sağlar ve daha fazla esneklik imkanı vardır: Ortak sanallaştırma altyapıları ile sanal makineleri bir fiziksel makineden diğerine hızlıca taşımak mümkündür ve bu yapılar yükleme hataları ve arıza gibi durumları yönetmek için çok elverişlidir.

Bulut bilişim fikri büyük bilgisayar firmalarının sadece ihtiyaç anında ve kısa süreli lazım olan verileri depolamak için alternatif aramalarıyla oluştu çünkü bu çok maliyetli bir işti. Bulut bilişim kullanıcılarına sağlayıcılar

üzerindeki sanal makinalara erişim izni verilir, alanlara kullanıma bağlı olarak yükleme yapılabilir ve bu sistem, işlemi gerçek bir sistemde sürdürmekten çok daha tasarrufludur böylece 7/24 çalışacak bir bilgi işlem merkezinin kurulması için gerekli olan inşaat giderleri, personel giderleri, malzeme ve enerji giderleri gibi masraflardan kaçınılmış olunur.

Alıştırmalar

- İnternetten Andrew S. Tanenbaum ve Linus Torvalds ile ilgili olan meşhur tartışmayı bulun. Tanenbaum Linus Torvalds'ın Linux'u üretirken başarısız oldugunu düşünüyordu. Sen ne düşünüyorsun ?
- Linux çekirdeğinin kaynağı olan en eski kurulum kodunun sürüm numarası nedir bulabilir misiniz?

2.2 Bedava yazılım mı açık kaynak kodu mu?

2.2.1 Telif Hakkı Ve Özgür Yazılım

Ortaçağ boyunca kitap ve diğer yazılı ürünlerin çoğaltılması çok pahalı bir işti. Gerekli sürede elde yazabilen birini bulmak gerekiyordu – manastırlar incil kopyalamak için zamanın en iyi projelerinden birini yapmıştı(yazı yazmayı bilen rahipler vardı ve çok fazla boş zamanları vardı.) 16.yüzyılda matbaanın bulunması ile birlikte kopyalama işlemi daha basit ve ucuz oldu ve yayımcılık sektörü satmaya değer görülen her şeyi kopyalayıp dağıtmaya başladı. O zamanlarda yazarların neredeyse hiç hakkı yoktu, eğer yayımcılar onlara kendi çalışmalarını basmak için para öderse şanslıydılar. Gittikçe yayılan kopyalama orijinal yazıcıları kızdırdı, kendilerini aldatılmış hissettiler. Bu belirli çalışmalar için hükümetten özel haklar istemelerine neden oldu. Hükümet basılan yayınları denetlemek için bunu reddetmedi. Zamanla bu "özel haklar" yazarlara da geçti ve modern haliyle "telif hakkı" (ya da yazar hakları) olarak bilinen duruma dönüştü.

Telif hakkı bir işi yapan kişinin (bir kitabın yazarı, bir resmin ressamı, ...) o esere ne yapılacağı ile ilgili bütün haklara sahip olması demektir. Yazarlar kitaplarının basım haklarını yayıncılara verebilir ve kitaplarının piyasaya sürülmesini sağlayabilir; yazar basma, yayımlama, pazarlama ve buna benzer hiçbir şey ile uğraşmayarak yine de parasını kazanabilir, yayımcı ise kitap yazma derdinde olmadan para kazanabilir. Bu iki tarafında yararına olan bir durumdur.

Buna ek olarak "telif hakkı" "manevi hak" olarak belirlenebilir. Genelde bu hakları devretmek imkansızdır.

20. yüzyılda telif hakkı tasarımı uluslar arası olarak kabul gördü ve kayıt ve film endüstrisine de yansıdı. Bilgisayarın icadı ve internet durumu bir kere daha ciddi şekilde değiştirdi: Patentlerin amacı yayımcıyı diğer yayımcılara

karşı korumaktı, aniden bilgisayara sahip olan herkes dijital içerikleri kopyalamaya başladı.(yazılım, kitaplar, müzikler ve filmler gibi) Bu yayımcılar, müzik, filme ve yazılım firmaları için bir felaketti, çünkü bu kurumların iş modeli olan bu ürünleri satmak tehlikeye girmişti. O zamandan beri "içerik endüstrisi" daha sıkı patent yasalarına ve korsan kopyacılar için daha yüksek cezaların konulması yönünde sıkı çalışmalara başladı ve aynı zamanda telif haklarını ihlal edenler hakkında soruşturma açmaya çalışıyorlar (ve bir açıdan başarılı oluyorlar).

Günümüzde "fikri mülkiyet" sadece telif haklarını değil aynı zamanda marka ve patent haklarınıda içerir. Teknik süreçlerin belgelenmesi ve yayınlanmasından sonra patentler mucitlere buluşlarını kullanma ve yararlanma hakkı verir.(ör. buluşlarından yararlanma hakkını para karşılılığında başkalarına verebilirler). Ticari markaların popülerleğinin başkaları tarafından kullanılarak istismar edilmesi engellenir. Örneğin kimse kahverengimsi ve şekerli bir içeceği "Coca-Cola" ismi ile satma hakkına sahip değildir. Fikirler için üç ceşit "fikri mülkiyet" vardır ve birbirini tamamlayıcı niteliktedir, telif hakkı somut ifadesi ile fikirlerin gerçek çalışmalar ile ilgili kısmını şekillendirir ve ticari markalara kalitesiz iş uygulamalarını durdurma hakkı verir.

Seri üretim yapmak için telif hakkına sahip olunmalıdır ve bunun içinde çalışma küçük değişiklikler dışında son halini almış olmalıdır. Patentler patent ofisi tarafından yenilik açısından incelenerek verilir. Ticari markalar kesinlikle tescillenmiş olmalıdır fakat ürün veya hizmet sağlayıcısı kamu tarafından tanınana kadar belirli bir süre için logo kullanabilir.

Bilgisayar yazılımlarıda (yazılı bir çalışma şeklinde de olabilir, belirli bir seviyeye gelmiş yaratıcı bir düşüncede olabilir) telif hakları ile koruma altındadır. Bu telif hakkı sahibinin (programcı veya işveren) açık izni olmadan bir yazılımın bir bölümünü veya tamamını kopyalamanın yasadışı olduğu anlamına gelir.

Geçmişte bilgisayar yazılımı satışı çok yaygın değildi. Ya almış olduğunuz bilgisayar ile birlikte gelirdi(bu durumda fiyat epey yüksek olurdu, milyon dolarlar seviyesinde) ya da kendiniz yazmak zorundaydınız . 1960'larda ve 1970'lerde üniversiteler programları ya değiş tokuş ederdi ya da kopya programlar kullanırdı, 1976 yılında Bill Gates'in MITS Altair 8800 için dehşetle yapmaya çalıştığı BASIC yorumlayıcı gerçekten çok popüler oldu ve bununla çok övgü aldı ama kimse fiyatını sormayı akıl etmedi! Yazılım için para ödenmesi fikri kullanıcılar için tabiki mantıklı değildi hatta bazıları alay etti.

1970'li ve 1980'li yıllarda ofislerde bilgisayar kullanımı yaygınlaştı ve yazılım ihtiyacının artmasıyla birlikte yazılım satışı fikride yaygınlaştı. Bununla kalmayarak yazılım firmaları nasıl çalıştığı anlaşılmayan, düzenlemenin ya da incelemenin münkün olmadığı kapalı kaynak kodlu yazılımlar sattılar. Bir gün MIT'de araştırma görevlisi olan Richard M. Stallman 1960 ve 1970'lerin şartlarındaki bu durumun tam tersine paylaşım kültürünün öne çıkacağı bir

sistem üzerinde çalışmaya karar verdi. "GNU" [3] projesi tamamlanamadı ama bazı bileşenleri günümüzde bile Linux sistemlerinde kullanılmaktadır.

Richard M. Stallman (kısaca "RMS" olarak da bilinir) Özgür Yazılım fikrinin babasıdır. Bu bağlamda "Özgür" kelimesi "bedava" anlamına gelmez ama kullanıcı telif hakkıyla korunan yazılımlarda [4] yapamadığı çeşitli şeyleri yapabilir. RMS yazılım paketini "Özgür" olarak adlandırmak için 4 şart arar. "Dört Özgürlük" ile tanışın:

- Herhangi bir amaç için programı çalıştırmak özgürlüğü (özgürlük 0).
- Programın nasıl çalıştığını inceleme ve istediğinizi yapacak şekilde değiştirme özgürlüğü(Özgürlük 1)
- Kopyaları dağıtma özgürlüğü böylece etrafınızdakilerde faydalanabilir (Özgürlük 2)
- Programı geliştirme ve geliştirilmiş versiyonu(modifiye edilmiş genel versiyon) açık olarak olarak yayınlama özgürlüğü, böylece bütün topluluk faydalanır (Özgürlük 3).

Programın kaynak koduna erişmek 1. ve 3. özgürlükler için bir ön koşuldur. Özgür yazılım fikri genel olarak olumlu karşılanır, öyle olmasına rağmen RMS ve Özgür Yazılım Vakfı (FSF) hedefleri genellikle yanlış anlaşılır. 1990'ların sonunda, Eric S. Raymond, Bruce Perens ve Tim O'Reilly Open Source Initiative(OSI)(açık kaynak girişimini) hazırlandı, bazılarına göre bu durum özgür yazılım için daha iyi ve daha az ideojik bir pazarlama yöntemidir. FSF bu fikirlerinin "sulandırılması" konusunda hevesli değildi, FSF ve OSI çok benzer hedeflere sahip olmasına rağmen aradaki anlaşmazlıklar bugün bile sona ermemiştir (Bazı anlaşmalar önemli insanların karmaşık egolarına bağlı olabiliyor).

"Özgür yazılım" tanımının içindeki "özgür" "masrafsız" olarak yanlış anlaşılmaya müsaitken "açık kaynak kodlu yazılım" tanımının içindeki "açık kaynak kodu" bu tür kaynak kodlarının değiştirilip değiştirilmediğinin denetlendiği yönünde yorumlanabilir– Her ikisi de OSI temel ilkelerindendir. Bu anlamda, her iki terimde % 100 kesindir. Topluluktan sıklıkla "FOSS" (for free and open-source software) olarak bahsedilir ya da alternatif olarak "FLOSS" (özgür, libre ve açık kaynak yazılımı, FLOSS' daki "libre" özgürlük anlamına gelen ingilizce "liberty" kelimesini belirtmek için kullanılır.) kullanılır.

Herkes kopyalama ve değiştirme gibi haklara sahipse özgür yazılımdan nasıl para kazanılır? Bu çok doğal bir sorudur. Burada bir kaç örnek ("açık kaynak için iş modelleri") var:

[3] GNU "GNU's Not Unix" için bir kısaltmadır.

[4] Stallman 'ın örgütü, Özgür Yazılım Vakfı şöyle bir tanım yapıyor; "Free as in speech, not as in beer"

- Destek servisleri veya döküman sağlayabilirsiniz ya da özgür yazılım eğitimi vererek para kazanabilirsiniz (Bu yaratıcı firmalar için çok iyi bir iştir, Linup Front GmbH ve LPI gibi şirketler Linux sertifikaları satarak geçimini sağlıyor).

- Belirli müşterileriniz için özel iyileştirmeler veya eklentiler oluşturup harcadığınız zaman karşılığında para kazanabilirsiniz (yapmış oldugunuz geliştirmeler genel versiyonun bir parçası olur). Bunu eğer kendiniz için yapmıyorsanız bu çalışma özgür yazılım için başka bir alternatif olur.

  Özel yazılım geliştirmenin "geleneksel" modeli çerçevesinde, yazılımın orijinal üreticisi değişiklik yapma ve geliştirme hakkını elinde bulundurur. Böyle bir şirketin müşterisi olarak, üretici ürünü üretimden kaldırırsa ya da düpedüz kaybolursa diye kaygılanabilirsiniz (iflas edebilir ya da rakipleri tarafından satın alınabilir), çünkü bu büyük bir sorununuz var demektir ve geleceği olmayan bir yazılıma masraf yapmış olursunuz. Özgür yazılımda her zaman orjinal üreticilerden destek alacak birini bulursunuz–gerekirse diğer kullanıcılarla birlikte destek talep edebilirsiniz kimse yalnız kalmanızı istemez. Eğer bir paket yazılım satmak istiyorsanız önce FOSS şeklinde bir temel sürüm hazırlayıp insanları daha gelişmiş özelliklere sahip "tam sürüm" satın almaya ikna edebilirseniz, bu işi başarırsınız.

- Bu iki ucu keskin kılıç gidir: bir yandan bakıldığında aradığınız iş için çok sayıda bedava yazılım olması şüphesiz çok iyi bir şey olurdu fakat diğer taraftan bakıldığında genellikle ücretsiz versiyonlardaki önemli fonksiyonların kısıtlandığını görürsünüz ve işlevselleştirmek için çok fazla çalışmak gereklidir. Bu durumda ücretsiz (özgür) kelimesi bir çok sunucuda arama motorlarındaki popülerliği artırmak içim kullanılır ve bu durum yayıncıların "özgür yazılım dostu" olarak görünmek istemesiylede ilgilidir –yürümeden ilerleme kaydetmeye çalışmaktır.

2.2.2 Lisanslar

Bir yazılım nasıl "ücretsiz" veya "açık kaynak kodlu" hale gelir? Bahsettiğimiz bazı haklar –örneğin kopyalama ve değiştirme hakkı– kesinlikle çalışmanın sahibine aittir fakat bu kişi hakklarını başkalarına devredebilir. Bu yapmanın yollarından biri lisanslamaktır, yasal bir belge ile yazılımın bazı haklarının satın alan kişide veya indiren kullanıcıda olduğu belirtilebilir. Telif hakları bir yazılımı satın alan kullanıcı[5] (ya da yasal olarak indirme hakkına sahip başka biri) için sadece kurma ve çalıştırma izni verir. Buradan yazılımın kullanılmasının yapımcının iznine bağlı olduğu gerçeği çıkarılabilir–Buradan

[5] Aslında, bir çok ülkenin telif hakkı yasaları, "RAM 'e diskten bir program kopyalama ve çalıştırma" işlemi için bir telif hakkı olamıyacağına yönelik açıklamalar içerir.

satın almayan kullanıcıların kullanmasına izin verilmediği açıkça bellidir (tam tersine.yapımcıya para verirse parasıyla yazılımın haklarını takas etmiş olur). yazılımı kontrolsüz bir şekilde kopyalama,dağıtma,bir kısmını veya tamamını değiştirmenin yasak olduğu telif haklarında açıkça belirtilmiştir hak sahibi izin vermek istiyorsa bunu lisans içersinde belirtmelidir.

Fikri mülkiyet hakkına tabi programlar genellikle "son kullanıcı lisans sözleşmesi (EULA)" ile birlikte gelir alıcı yazılımı kullanmak için sözleşmeyi kabul etmek zorundadır. Yazılım satıcılarının EULA ile alıcıları yasaklaması aslında telif hakları ile izin verilen bir şeydir – yazılımı başkasına satarak kullanmasına izin vermek gibi, veya açık bir bir şekilde yazılımla ilgili kötü(aslında herhangi birşey) şeyler söylemek gibi. EULA sözleşmesinde oldukça fazla sayıdaki yasal engellemenin her iki tarafça da kabul edilmesi gerekiyor (en azından Almanya'da öyle). Örneğin, alıcı yazılımı almadan önce sözleşme şartlarını incelemek zorundadır ama sözleşme şartlarını kabul etmezse en azından parasının bir kısmı geri ödenmelidir.

Diğer taraftan, açık kaynak lisansı olan FLOSS için yazılımda yapılacak değişikleri telif hakkıyla yasaklamaktan başka yollar vardır. Genellikle yazılım kullanımını kısıtlamayı kimse istemez ama dağıtma ve değiştirme sürecini yönetirler. Bu şekilde alıcıyı en kötü durumda bile yanlız bırakmayarak onlardan daha iyi bir yazılım satın almalarını beklemezler. Bu nedenle, EULA'ların aksine, özgür yazılım lisansları genellikle sözleşme değildir alıcısı zorunluluk olduğunu açıkça kabul eder, yazılım yapımcıları adına tek taraflı beyanları ve onlar tarafından tanınan hakların dışında ekstra faydalardan oluşmaktadır, yasa aslında basit bir kullanım hakkıdır.

Artık temel kuralların yerine getirilmesi gereken hayvanat bahçelerinin yerinine ücretsiz yazılımlar ve açık kaynak kodlu yazılımlar bulunmaktadır. En iyi bilinen özgür yazılım lisansı Genel Kamu Lisansı(GPL) Ricsabit M. Stallman ve diğerler tarafından yürürlüğe kondu. OSI kendi görüşüne göre açık kaynak ruhunu somutlaştıran bu lisansları "onaylar", FSF lisansları onaylar ve "dört özgürlük" ile korur. Onaylı lisanslar listesi bu kuruluşların internet sayfalarında mevcuttur.

Eğer bir ücretsiz veya açık kaynak yazılım projesine başlamayı düşünüyorsanız, kesinlikle sizin için gereklerini yerine getiren ücretsiz bir FSF ya da OSI lisansı vardır. Bu lisansınız için FSF veya OSI'den onay almak gerekmediği anlamına gelirve bununla birlikte, zaten onaylanmış mevcut bir lisans kullanmak genellikle daha iyidir, mevcut lisanslar genellikle yasa uzmanları tarafından incelenmiş makul ve kabul edilebilir olduğu su geçirmezdir – yeni bir amatör sözleşme veya fikri mülkiyet hukuku hazırlarken daha sonra başınızı derde sokabilecek önemli ayrıntılar göz ardı olabilir.

Önemli bir gözlem olarakta ücretsiz veya açık kaynak yazılım savunucuları hiçbir şekilde yazılım için telif hakkını tamamen ortadan kaldırmak niyetinde değildir. Aslında, özgür yazılımcılar olarak biz telif hakklarının yazılımın

değiştirme ve dağıtma gibi haklarını yasal olarak yapımcısına verdiğini biliyoruz ve bazı koşullar sağlanarak kullanıcılarında bu tür haklara sahip olması gerektiğini düşünüyoruz. Telif hakkı olmasaydı, herhangi bir yazılım için herkez birbirine yardım edebilirdi ve "dört özgürlük" gibi merkez ilkeler olmadan bu tehlikeli olabilirdi çünkü insanlar hiç bir paylaşımda bulunmadan birikimlerini saklayabilirdi.

2.2.3 GPL

Linux çekirdeği ve diğer bütün "Linux" paketleri Genel Kamu Lisansı (GPL) altında dağıtılmaktadır. GPL, RMS tarafından GNU projesi için geliştirilmiştir başlamakta olduğu yazılım için GPL lisansı ile dağıtılan yazılımın GPL altında kalması gerekiyordu (bu lisans tip copyleft lisans olarak adlandırılır). Bu yaklaşık olarak aşağıdaki gibi çalışır:

- GPL Yazılımın kaynak biçiminde mevcut olması gerekir ve isteğe bağlı bir şekilde kullanılabilir olmasını amaçlar.
- Kaynağını değiştirmek için açıkça izin verilir kaynağı degiştirerek ya da değiştirmeden dağıtabilirsiniz ve GPL bir sonraki alıcıyada aynı hakları verir.
- Ayrıca GPL yazılımları çalıştıralabilir formlarda bile dağıtmak münkün (hatta satabilirsiniz). Bu durumda, kaynak kodu (GPL hakları ile birlikte) yazılımın çalıştırılabilir formu ile birlikte sunulmalıdır ya da istendiği takdirde belirli bir süreliğine paylaşılabilir.

  Bu bağlamda "Kaynak kodu" almak demek yazılımı bilgisayarda çalıştırmak için gereken her şeyi almak demektir. Özel durumlar ne demektir– örneğin kapalı bir bilgisayardaki değiştirilmiş Linux çekirdeğini uygun bir şekilde başlatmak için şifreleme anahtarları gerekebilir– bu hararetli bir tartışma konusudur.

  Eğer biri para ile GPL yazılım satın alırsa, doğal olarak sadece bütün bilgisayarlarında çalıştırma hakkına sahiptir, kopyalayamaz ve yeniden satamaz.(GPL lisansı altında).Bunun bir sonucu olarak da "koltuk başına" GPL yazılım satmak mantıklı bir iş değildir, bu durumun önemli bir sonucu olarakta fiyatlar açısından rahatlık sağlamasıdır bu sebeple Linux dağıtımları kullanmak mantıklıdır.
- Eğer bir GPL programına ait parçaları bir araya getirerek yeni program (bir "türetilmiş çalışma") yazarsanız, bu program GPL ile lisanslanmalıdır.

Burada da, hararetli tartışmalara sebep olan şey "türetilmiş program" yazmak için ne kadar GPL programı kullandığınızdır. FSF'ye göre,Bir program

dahilinde dinamik olarak kullanarak GPL kütüphanesi kullanılıyorsa o program GPL altındadır, herhangi bir GPL kod bu nedenle GPL altında olmalıdır hukuki anlamda bir "türetilmiş çalışma" olarak kabul edilemez. Bu durumlardan ne kadarının uydurma ne kadarının hukuken savunulabilir olduğu, prensip olarak, bir hukuk mahkemesinde tespit edilmelidir.

GPL, yazılımı kullanım için değil değiştirmek ve dağıtmak için kurallar belirlemektedir.

Şu anda yaygın olarak kullanılan iki GPL sürümü vardır.Yeni sürüm 3 (ayrıca "GPLv3" denir) 2007 Haziran ayında yayımlanmıştır ve eski sürümden (sürüm 2 (ayrıca "GPLv2")) farkları yazılım patentleri gibi alanlarda açıklamalar, diğer ücretsiz lisansları ile uyumluluk, "özgür" yazılımların teorik olarak değişiklik yapmanın imkansız olduğu Özel donanımlarda çalıştırılması ile ilgili tanımlarıdır ("Tivoisation",olan çekirdek, Linux tabanlı bir dijital PVR değiştirilemez ve yenilenemez). GPLv3 kullanıcılarınına başka şartlar eklemek için izin verir.–GPLv3 toplum içinde evrensel onayı almadı,dolayısıyla birçok proje (en belirgin, Linux çekirdeği) daha basit olan GPLv2'de kalmıştır. Buna ek olarak, birçok proje "GPLv2 veya sonraki bir sürümünü" altında kodunu dağıtmaktadır, dolayısıyla bu tür yazılımları dağıtırken veya değiştirirken hangi sürümünün kullanılacağına GPL'i takip ederek kendiniz karar verebilirsiniz.

Projeye katılmak için lisans kapsamında projenizle aynı bir projenin lisansını kullanmak özgür yazılım geliştiricileri arasında en iyi tarzı olarak kabul edilir, anonim kod kullanmakta ısrar eden bir çok proje için bu "resmi" bir yoldur. Bazı projelerde projeye kod veren yapımcılar ısrarla telif hakkı atamaları(ya da benzeri bir organizasyon) isterler. Bu adımın avantajı ise kodun telif hakkının ve telif hakkı ihlalinin projeye ait olmasıdır—telif hakkı sahipleri yasal dayanaklarına başvurabilir—muhattabını bulmak daha kolaydır. Beklenmeyen bişey olduğunda ya da olmasını istemediğiniz bişey olduğunda projenin lisansını degiştirmek kolaylaşır,böyle bişey olduğunda bunu yapma yetkisi sadece telif hakkı sahibindedir.

Linux çekirdeği proje durumunda olduğundan, açıkça telif ataması gerektirmez, kodlar binden fazla yazarın katkılarıyla oluşan bir toplu çalışma olduğundan lisans değişikliği çok zor ya da imkansızdır. Bu sorun GPLv3 tanıtımı sırasında tartışıldı.Geliştiriciler yasal kaynakların sıralanması için dev bir proje yapılmasını kabul etti ve yazarlardan Linux çekirdek kaynak kodunun her satırı için lisans değişikliği onayı alındı. Inatla karşı çıkan Bazı Linux geliştiricileri oldu, orada bulunayaman ya da vefat etmiş olan geliştiricilerde vardı onların kodlarının telif haklarını temizlemek için kodların yenilenmesi veya benzeri ile değiştirilmesi gerekiyordu. Bununla birlikte, en azından Linus Torvalds GPLv2 destekçisi olarak kaldı, bu nedenle uygulamada bir problem olmadı.

GPL ürünün olası fiyatı konusunda bir şart bulundurmaz. GPL yazılım kopyalarını vermek veya para karşılığında satmak kaynak kodlarını beraberinde

vermek ya da istendiğinde vermek şartıyla tamamen yasaldır ve alıcıda GPL haklarını alır.Bu GPL yazılımın ücretsiz olması gerekmediği anlamına gelir. GPL [GPL91] inceleyerek daha fazla bilgi bulabilirsiniz,dağıtılmış olan bazı GPL-ed ürünleri (Linux içeriğinde var) inceleyebilirsiniz. GPL bu anlamda ücretsiz lisansların en tutarlı olanı olarak kabul edilir-dediğimiz gibi-GPL tarafından sağlanan,altında yayımlanan, kod özgür kalmalıdır. Şirketler çeşitli vesilelerle kendi projelerine GPL kod dahil etmeye çalışıyorlar, bu GPL den kurtulmak için bir bahana degildir. telif hakkı sahibi olarak (en sık) FSF'nin sert eleştirilerinden sonra bu şirketler GPL ile uyumlu hale gelmiştir. En azından, Almanya'da GPL mahkeme kararıyla doğrulanmıştır–Linux kernel programcısı D-Link (ağ bileşenleri üreticisi, bu durumda bir Linux tabanlı NAS cihazı)'e karşı Frankfurt bölge mahkemesinde elde edebilir karar çıkarmıştır, cihazı dağıtırken GPL uyulamadığını iddia ederek dava açmıştı.

Neden GPL çalışır? Bazı şirketler GPL'e ağır kısıtlamalar getirmeye ya da geçirsiz kılmaya çalıştı. Örneğin, Amerika Birleşik Devletleri'nde, "Amerikan karşıtı" ya da "anayasaya aykırı" olarak adlandırıldı ve Almanya'da bir şirket geçersiz kılmak için rekabet hukuku kullanmaya çalıştı GPL sözde yasadışı fiyat belirleme yapıyormuş. Eğer bir şeyler yanlış ise hiç kimsenin bunu yapmayacağı fikti GPL ile kanıtlanabilir gibi görünüyor. Aslında bu saldırılın önemli bir kısmı göz ardı edilebilir: GPL olmasaydı, yazılımda yapımcısı dışından herhangi birinin değişiklik yapması doğru olmazdı, böyle dağıtımlar yapılamazdı ve yazılım satışlarıda telif hakları yasası ile sınırlandırılırdı. Yani eğer GPL yok olursa, kodlarla ilgili eğlendiğiniz ne varsa öncekinden daha kötü bir hal alır.

Bu noktada davalı "hayır" da diyebilir ama eğer "evet" dersede karar değişmez kodun herhangi bir telif hakkının olmaması onları gene haklı çıkarır. Bu rahatsız edici bir ikilemdir çünkü gerçekten bazı şirketler bu durumu kendi çıkarları için kullanmıştır-GPL anlaşmazlıkları çoğunlukla mahkeme dışında yaşanır.

Eğer bir yazılım üreticisi GPL ihlali yaparsa (örn. kendi projesine yüzlerce satır GPL kodu eklerse), bu o projenin bütün kodlarının artık GPL'den bağımsız olduğu anlamına gelmez. Yanlızca GPL kodunun lisanssız dağıtıldığı anlamına gelir, üreticiler bu sorunu çeşitli şekillerde çözerler:

- GPL kodunu kaldırarak kendi kodlarını koyarlar. Bu şekilde yazılımları GPL'den bağımsız olur.

- GPL kodunun telif hakkı sahibi ile pazarlık yapabilirsiniz (eğer varsa ve pazarlık yapmak isterse) örneğin, bir lisans ücreti ödemeyi kabul edersiniz. Ayrıca aşağıda birden çok lisans bölümüne bakınız.

- Onlar programın tümünü gönüllü olarak GPL altında yayınlayabilir ve böylece GPL koşullarına(en olası yöntem) bağlı kalınmış olur.

Bunlardandan bağımsız olarak, önceki ihlalleri için ödenecek zarar söz konusu olabilir. Sahipli yazılımın yazılım telif hakkı durumu herhangi bir şekilde bundan etkilenmez.

2.2.4 Diğer Lisanslar

GPL'e ek olarak, başka popüler diğer lisanslarda vardır. Işte kısa bir bakış:

BSD Lisansı BSD lisansı Berkeley'deki Kaliforniya Üniversitesinde başlatılmıştır Unix dağıtımları kasıtlı olarak çok basit tutulur: Yazılım alıcı universite(veya uzantısı, orijinal yazılım yazarı) tarafından yapıldığı izlenimi oluşturmadan istediğini yapmakta özgürdür. Program için mümkün olduğunca herhangi bir yükümlülük bulundurulmaz. lisans metni programın kaynak kodu ve içinde– Değiştirilmiş versiyonlarda veya dağıtılan diğer çalıştırılabilir formlarda– veya dökümanlarında korunmuş olmalıdır.

Bir yazılım paketi BSD lisanslı kod içeriyorsa, bu kodun herhangi bir promosyon materyalinde ya da bir sistemde kullanılması sorunu telif hakkı sahibini ilgilendirir. Bu "reklam cümlesi" bir köşede dursun.

GPL'in aksine BSD lisansı yazılımın kaynağını açık tutmaya çalışmaz, isteyen BSD lisanslı yazılımı aslında kendi yazılımının içine entegre edip binary olarak dağıtabilir (bu GPL yazılım olsaydı ilgili yazılımın kaynak kodunuda GPL altında dağıtmak gerekirdi).

Microsoft veya Apple gibi Ticari yazılım şirketleri, GPL yazılım konusunda daha az heveslidir fakat BSD lisanslı yazılımlar ile hiçbir sorunları yoktur. Örneğin Windows NT tarafından kullanılan TCP/IP network kodu (adapte şeklinde) BSD'dir ve Macintosh'un OS X işletim sistemi BSD çekirdeğinin biraz daha geniş parçalarını kullanır.

FOSS toplulukları içinde,GPL veya BSD lisanslarından hangisinin "daha özgür" olduğu konusunda uzun süreler farklı görüşler belirtilmiştir. Bir taraftan bakıldığında mantıklı bir alıcı olarak, BSD lisanslı yazılımlarla daha özgürsünüzdür ve bu nedenle BSD lisansı mutlak olarak daha fazla özgürlük aktarıyordur. Öte yandan GPL savunucularına göre kodların ücretsiz kalması herkesin kullanması için önemlidir ve başka türlü olursa özel sistemler içinde kaybolunur, GPL kodunu almak isteyenlerin GPL yazılım havuzuna birşeyler vermek zorunda olması bu büyük özgürlüğün göstergesidir.

Apache Lisansı Apache lisansı BSD lisansı gibidir, yazılımların değiştirilmiş sürümlerinin aynı lisansı kullanarak ya da özgür-açık kaynak kodlu yazılım olacak şekilde dağıtılması koşulunu barındırmamaktadır. BSD'den daha karmaşıktır ama ayrıca patentleri, ticari marka haklarını ve diğer detaylar ile ilgili kullanım koşulları içerir.

Mozilla Kamu Lisansı Mozilla lisansı (Firefox ve diğer yazılım paketlerine uygulanır) BSD lisansı ile GPL'in bir karışımıdır. Copyleft yanı zayıf kalan bir lisanstır, MPL lisansı altında alınan kodun MPL altında yayınlanması şart koşulurken (GPL gibi) eklenen kodlar MPL altında yayınlanmak zorunda değildir.

Creative Commons Özgür Yazılım camiasının başarısı hukuk profesörü Lawrence (Larry) Lessig'i cesaretlendirdi ve Lawrence yazılıma ek olarak diğer çalışmalar içinde aynı durum için başvuruda bulundu. Amaç kitaplar, resimler, müzikler ve filmler gibi kültürel havuzu oluşan eserlerin herkesin kullanımına, değiştirmesine ve dağıtmasına özgürce açık olmasıydı. Yaygın özgür yazılım lisansları sadece yazılım üzerine oldukları için yaratıcı-üretimler lisansı eserlerini halkın kullanımına bağışlayan kişiler için geliştirildi. Bunlarda da bazı kısıtlamalar bulunabilir; sadece çalışmanın gösterilmesine izin verilebilir, GPL gibi değiştirilmiş ürünün ilerde yapılabilecek değişikliklere açık olarak sunulması istenebilir ya da yayının ticari amaçla kullanılması engellenebilir.

Kamu Malı "Kamu malı" artık telif hakları altında olmayan kültürel eserler için uygulanır. Anglo-Saxon yasal kurallarına göre bir eserin yaratıcısı eserini (örneğin bir yazılım parçası) kendi bütün haklarından vazgeçerek kamu malı haline getirebilir ama bu her zaman bütün yasal çevrelerde geçerli olmaz. Örneğin Almanya'da eserler bu eser üzerinde en son çalışan kişinin ölümünden 70 yol sonra kamu malı haline geçer. İlk bilgisayar programının bu şekilde herkese açık duruma geçmesi için biraz daha beklememiz gerekiyor.

1930'dan sonra üretilen telif haksız ürünlerin kamu malı durumuna geçmesi için bir ihtimal söz konusudur. Birleşik Devletlerde, Parlemento telif hakkı terimini süresi geçmiş bir çizgi film üzerinde genişletti ve dünyanın geri kalanı bunu izlemekten genel olarak memnun. Neden Walt-Disney'in torunlarının torununun gerçek bir sanatçının yaratıcılığından para kazanabileceği pek açık değil ama bu genelde en iyi lobiyi yapanlar tarafından belirlenen bir durum elbette.

Çoklu Lisanslama Temel olarak bir yazılım paketinin telif hakkı sahibi bu paketi aynı anda başka lisanslar altında da sunabilir – örneğin özgür yazılım lisansı GPL geliştiriciler için, sahipli telif hakkı kaynak kodlarını açıkça sunmak istemeyen şirketler için. Elbette bu en çok diğer programcıların kendi programları ile kullanabilecekleri kütüphaneler için anlam ifade eder. Kim sahipli bir yazılım geliştirmek isterse GPL kısıtlamalarından paralı lisans alarak kurtulabilir.

Alıştırmalar

- GPL ile ilgili olan yargılardan hangileri doğru ya da yanlıştır?
  1. GPL yazılımı satılamaz.
  2. GPL yazılımı şirketler tarafından değiştirilerek kendi ürünlerinde kullanılamaz.
  3. GPL yazılım paketinin sahibi bu programı başka bir lisans altında da dağıtabilir.
  4. GPL geçerli bir lisans değildir çünkü lisans ancak kişi sözü edilen yazılım paketini aldığında lisansı görür. Bir lisansın geçerli olabilmesi için kişinin bu lisansı görüp yazılımı kullanmadan önce kabul etmesi gerekir.
- FSF'nin "dört özgürlüğünü" Debian Özgür Yazılım Kılavuzu ile karşılaştırın. (Bölüm **??**'e bakınız.) Hangi özgür yazılım tanımını daha çok beğendiniz ve neden onu daha çok beğendiniz?

2.3 Önemli Özgür Yazılımlar

2.3.1 Genel Bakış

Linux güçlü ve şık bir işletim sistemidir ama en iyi işletim sistemi bile üzerinde çalışacak programlar olmadan bir işe yaramaz. Bu bölümde normal Linux bilgisayarlarda bulunabilecek en önemli özgür yazılımlardan bir kısmını tanıtacağız.

Eğer belirli bir program söylediği şeyleri yapmıyorsa onları dikkate değer bulmuyoruz. Alanımız sınırlı ve LPI'nin ve sınavında geçen yazılım paketlerini anlatmaya çalıştık (ne olur ne olmaz).

2.3.2 Ofis ve Geliştirici Araçları

Çoğu bilgisayar muhtemelen ofis uygulamaları için kullanılıyordur, çünkü bu uygulamalar mektup ve anılarınızı, seminer kağıtlarınızı ya da yüksek lisans tezinizi yazmanızı sağlar, tablolar kullanarak verinin gelişimini gösterir ve buna benzer görevlere sahiptir. Kullanıcılar ayrıca bilgisayardaki zamanlarının çoğunu internette ya da mail okur ya da yazarken harcarlar. Bununla ilgili birçok özgür yazılım olmasına şaşmamalı.

Bu bölümdeki programların çoğu sadece Linux için değil ve Windows, OS X ve Unix türevleri içinde bulunmaktadır. Bu kullanıcıların işletim sistemlerini Linux ile değiştirmeden Microsoft Office, Internet Explorer ve Outlook yerine Libre Office, Firefox ya da Thurderbird kullanarak işlerini hallet-melerini sağlar. Eğer kartlarınızı doğru oynarsanız [6] kullanıcılar farkı görmeyebilir bile.

[6] Örneğin, Windows 8 beta yerine Ubuntu kurup, gerekli kişiselleştirmeleri yapabilirseniz, kullanıcıyı gerçekten onun Windows 8 beta olduğuna ikna edebilirsiniz.

OpenOffice.org Uzun yıllar boyunca özgür yazılım camiasının büyük ofis tarzı uygulamalar konusunda amiral gemisi konumundaydı. Yıllar önce "Star Office" olarak başlamıştı ve Sun tarafından satın alındıktan sonra özgür yazılım olarak dağıtılmaya başlanmıştı. (hafif inceltilmiş bir hali) OpenOffice.org bir insanın bir ofis programından bekleyeceği her şeye sahip; kelime işlemci, tablolar, sunum hazırlama, veritabanı - ve büyük rakibi Microsoft'un dosya formatlarıyla iş yapabilecek düzeyde.

LibreOffice Sun Oracle tarafından alındıktan sonra OpenOffice.org un geleceği belirsizdi ve OpenOffice.org un ana geliştiricilerinden bazıları birleştiler ve kendi OpenOffice.org sürümlerini yayınladılar. İki pakette şu an geliştirilmeye devam ediliyor (Oracle OpenOffice.org u Apache Yazılım Vakfına bağışladı), fakat ne zaman ve ne şekilde bir "birleşme" olacağı belli değil.

Şu an çoğu büyük Linux dağıtımı LibreOffice ile birlikte geliyor, çünkü LibreOffice daha hızlı bir şekilde geliştiriliyor ve daha önemlisi daha temiz bir sürüm.

Firefox Şu anda en popüler internet tarayıcısı ve Mozilla Vakfı tarafından dağıtılmaktadır. Firefox bir zamanların en tepede olan tarayıcısı Microsoft'un Internet Explorer'ından daha güvenli ve daha hızlı çalışmaktadır, daha çok iş yapar ve daha rahat bir kullanım sunar. Bunlara ek olarak kendi isteklerinize göre Firefox'u özelleştirmek için varolan eklentileri kullanabilirsiniz.

Chromium Google tarayıcısı Chrome'un özgür yazılım türevi. Chrome son zamanlarda Firefox ile yarışmaya başladı – Chrome'da eklentileriyle birlikte oldukça güçlü ve güvenli bir tarayıcı. Özellikle Google'ın desteği ile son zamanlarda hız kazanmaya başladı.

Thunderbird Mozilla Vakfı'nın dağıttığı bir e-posta programı. Altyapısının büyük bir kısmını Firefox tarayıcısıyla ortak kullanıyor ve Firefox gibi değişik amaçlar için kullanılabilecek çok sayıda eklenti sunuyor.

2.3.3 Görseller ve Çoklu Ortam Araçları

Görseller ve çoklu ortam Macintosh'un baskınlığı altında. (Windows tarafında da iyi yazılımlar sunulsa bile) Kabul etmek gerekirse, Linux hala Adobe Photoshop'a eşdeğer bir programın eksikliğini hissediyor fakat bu konudaki yazılımlarda o kadar da kötü değil.

Gimp resimleri düzenlemek için bir programdır. Photoshop'un eşdeğeri sayılmaz (örneğin bazı baskı öncesi özellikleri eksik) ama kesinlikle çoğu amaç

için kullanılabilir hatta Photoshop'un iyi bir şekilde yapmadığı web için gereken grafikleri hazırlamak için birkaç araç sunuyor.

Inkscape Gimp Linux'un Photoshop'u konumundayken Inkscape resimler için kullanılır, vektör tabanlı grafiklerin oluşturulması açısından güçlü bir araçtır.

ImageMagick neredeyse bütün görsel formattaki dosyaları birbirine çevirmeye yarayan bir yazılım paketidir. Ayrıca resimleri betik kontrollü bir şekilde düzenleme yolları sunar. Web sunucuları ve grafiklerin bir fare ve monitörle işlenmesi gereken diğer çevreler için harikadır.

Audacity ses dosyalarını düzenlemek için kullanılır, Windows ve Mac bilgisayarlarda da popülerdir.

Cinelerra ve KDEnlive ya da OpenShot gibi programlar "düzgün-olmayan video düzenleyiciler" olarak bilinir ve dijital görüntü kaydedicilerden, TV alıcılarından, web kameralarından video alabilir bunları düzenleyip değişik efektler koyabilir ve çıktıyı istenen formata dönüştürebilirler.

Blender sadece güçlü bir video düzenleyici değil ama ayrıca üç boyutlu görsel tasarımına izin veren bir programdır, profesyonel kalitede canlandırma filmler için kullanılabilecek bir yazılımdır.

Bahsetmemiz gereken bir konu var, o da Linux olmadan bugünün patlamalı Holywood filmlerinden hiçbiri üretilemezdi. Büyük stüdyoların özel efekt "üretim çiftlikleri" nin hepsi Linux tabanlıdır.

2.3.4 İnternet Servisleri

Linux olmadan internet çok tanınır olmayabilirdi: Google'ın yüzbinlerce sunucusu dünyanın bütün dünyanın en büyük hisse senedi değişim ticaret sistemini çalıştırmak gibi görevlerde, istenen performans sadece Linux ile sağlanabildiği için Linux kullanır. Gerçek şu ki, çoğu internet yazılımı önce Linux'ta geliştirilir ve çoğu üniversite araştırmaları açık kaynaklı Linux platformu üzerinde olur.

Apache açık ara internetteki en popüler web sunucusudur, bütün web sitelerinin yarısından falzası bir Apache sunucusu üzerinde çalışır.

MySQL ve PostgreSQL özgürce dağıtılan ilişkisel veritabanı sunucularıdır. MySQL web siteleri için en iyisiyken PostgreSQL bütün amaçlar için kullanılabilecek yenilikçi ve yüksek performanslı bir veritabanı sunucusudur.

Postfix güvenli ve çok güçlü bir mail sunucusudur. Ev ofislerinden büyük ISP'lere ya da Fortune 500 listesindeki şirketlere kadar kullanılabilecek bir sistemdir.

2.3.5 Altyapı Yazılımları

Bir Linux sunucusu yerel ağda çok kullanışlı olabilir: Güvenilirdir, hızlıdır ve düşük-bakım gerektiren yükleyip unutabileceğiniz bir yazılımdır. (düzenli yedeklemeler dışında tabi ki!)

Samba bir Linux makinesini Windows istemcileri için bir sunucuya dönüştürebilir. (Linux istemciler içinde bunu yapabilir elbette) Yeni Samba 4 ile bir Linux sunucusu Aktif Dizin alan kontrolcüsü olarak kullanılabilir. Güvenilirlik, performans ve cepte kalan lisans paraları oldukça ikna edicidir.

NFS Samba için bir Unix ortamıdır ve ağdaki diğer Linux ve Unix makinelere Linux sunucu diskine erişimi sağlar. Linux geliştirilmiş performans ve güvenliği ile modern NFSv4'ü destekler.

OpenLDAP orta ve büyük ağlar için bir dizin servisi görevi görür ve (serves as a directory service for medium and large networks and offers a large degree of redundancy and performance for queries and up-dates through its powerful features for the distribution and replication of data.)

DNS ve DHCP temel ağ altyapısıdır. BIND ile birlikte Linux DNS sunucularını destekler ve ISC DHCP sunucusu çok büyük ağlarda bile istemcilere IP adresi gibi ağ parametrelerini sağlayabilir. Dnsmasq küçük ağlar için kullanması kolay bir DNS ve DHCP sunucusudur.

2.3.6 Programlama Dilleri ve Geliştirme

Başlangıcından beri Linux hep geliştirme ortamları için geliştirilmiştir. Bütün önemli diller için derleyiciler ve yorumluyucular bulunur – GNU derleyici ailesi, örnek olara, C, C++, Objektif C, Java, Fortran ve Ada'yı destekler. Elbette Perl, Python, Tcl/Tk, Ruby, Lua ya da PHP gibi popüler betik dilleride desteklenir ve Lisp, Scheme, Haskell, Prolog ya da Ocaml gibi daha az yaygın kullanılan dillerde çoğu Linux dağıtımı tarafından desteklenir.

Çok zengin bir setten oluşan editörler ve yardımcı araçlar yazılım geliştirmeyi bir keyfe dönüştürür. Standart düzenleyici vi ve GNU Emacs ya da Eclipse gibi profesyonel geliştirme ortamlarıda Linux'ta hazırdır.

Linux ayrıca "gömülü sistemler" için olan geliştirme ortamları içinde uygundur, yani bilgisayarlar kullanıcının uygulamaları içinde çalışır bunlar Linux'un

kendi temeline ya da özelleştirirmiş bir işletim sistemine dayanır. Bir Linux bilgisayarda, ARM işlemcilerde çalışacak makine kodunu üretecek bir derleyici ortamını yüklemek kolaydır. Linux ayrıca Android akıllı telefonları için yazılım üretilmesi için kullanılır ve bu amaç için kullanılaran araçlar Google tarafından özgürce dağıtılır.

Alıştırmalar

- Hangi Özgür Yazılım projelerini duydunuz? Hangilerini kendiniz kullandınız? Telif hakkıyla satılan alternatiflerinden daha iyi mi yoksa daha kötü mü olduklarını düşünüyorsunuz? Eğer öyleyse, neden? Öyle değilse, neden?

2.4 Önemli Linux Dağıtımları

2.4.1 Genel Bakış

Eğer birisi "Bilgisayarımda Linux çalışıyor" derse genelde sadece Linux'u değil, Linux temelinde çalışan tamamlanmış bir yazılım ortamından bahsediyordur. Bu genelde kabuğu (bash) ve komut-satırı araçlarını, X.org görsel sunucuyu, KDE ya da Gnome görsel bir kullanıcı arayüzünü, LibreOffice, Firefox ya da Gimp gibi araçları ve diğer bir sürü kullanışlı programı içerir. Elbette bu araçların hepsinin Orijinal kaynakları internette bulunarak derlenebilir fakat çoğu Linux kullanıcısı önceden yapılmış bir yazılım setini ya da bir "Linux dağıtımını" tercih eder.

Linux dağıtımı 1992'nin başlarında ortaya çıktı – buunla birlikte bunlardan hiçbiri günümüzde geliştirilmiyor ve genellikle unutulmuş haldedirler. Hala çalışan en eski dağıtım olan Slackware ilk defa Temmuz 1993'te ortaya çıkmıştır.

Değişik amaç ve yaklaşımlara sahip birçok Linux dağıtımı bulunur. Bazı dağıtımlar şirketler tarafından dağıtılır ve muhtemelen sadece para için satılır, gönüllülük esasıyla geliştirilen dağıtımlarda bulunmaktadır. Bu bölümde en önemli genel amaçlı dağıtımları inceleyeceğiz.

Eğer en sevdiğiniz Linux dağıtımından bahsetmezsek bu onu beğenmediğimiz anlamına gelmez, sadece yerimiz ve zamanımızın kısıtlı olduğunu ifade eder. Eğer bir dağıtım listemizde yoksa bu onun kötü ya kullanışsız olduğu anlamına gelmez bu sadece o dağıtımın listemizde olmadığı anlamına gelir.

"DistroWatch" web sitesi (http://distrowatch.com/) en önemli Linux dağıtımlarını listeler ve dağıtım-yönelimli haberleri sağlar. Şu anda 317 tane dağıtımı içeriyor (evet tam üçyüzonyedi tane!) ama bunu okuduğunuz süre içerisinde muhtemelen bu rakam doğru olmayacak.

2.4.2 Red Hat

Red Hat (http://www.redhat.com/) 1993 yılında Linux ve Unix araçları sağlayan bir dağıtım şirketi olan "ACC Corporation" tarafından ortaya çıkarıldı. 1995'te şirket kurucusu, Bob Young, 1994'te Red Hat Linux diye adlandırılan bir Linux dağıtımını ortaya çıkaran Marc Ewing'in şirketini satın aldı ve onun şirketinin adını "Red Hat Yazılım" olarak değiştirdi. 1999'da Red Hat halka sunuldu ve şu an muhtemelen sadece Linux ve özgür yazılım tabanlı olan en büyük şirket.

Red Hat orijinal bireysel müşteri marketinden çekildi (son Red Hat Linux 2004'te yayınlandı) ve şimdi "Red Hat Enterprise Linux" (RHEL) adı altında şirketler için profesyonel bir dağıtım ile pazara girdi. RHEL sunucu başına lisanslanmıştır ama yazılıma para ödemezsiniz – GPL ve benzer özgür yazılım lisansları ile gelmektedir bunlar– ama zamanlanmış güncellemeleri ve problem desteği için para ödersiniz. RHEL veri merkezleri için çokça tercih edilir ve diğerleri dışında hata dayanıklılığı için destek verir. (uygun ek araçlarla)

"Fedora" (http://www.fedoraproject.org/) büyük kısmı Red Hat tarafından kontrol edilen bir dağıtımdır ve RHEL için bir "deneme yatağı" olarak hizmet eder. Yeni yazılım ve fikirler önce Fedora'da denenir ve kullanışlı olduğu ortaya çıkanlar er ya da geç RHEL'de yerini alır. RHEL'in aksine Fedore satılmak yerine internette bedava olarak indirilebilir haldedir, proje bir kısmı Red Hat bir kısmı camia geliştiricileri olan bir komite tarafından sürdürülür. Çoğu Fedore kullanıcısı için var olan yazılımlara odaklanmak ve yeni fikirler dağıtımın cazibeli kısmıdır, hatta bu sık sık gelen güncellemeler anlamına gelse bile. Fedora başlangıç seviyesi kullanıcıları ve güvenilir olması gereken sunucular için çok uygun değildir.

Red Hat yazılımlarını GPL gibi özgür yazılım lisansları altında dağıttığı için Red Hat'a lisans bedelini ödemeden aynı işlevleri gören bir sistem edinmek mümkündür. CentOS (http://www.centos.org/) ya da Scientific Linux (https://www.scientificlinux.org/) gibi dağıtımlar genelde RHEL tabanlıdır ama bütün Red Hat markası silinmiştir. Bu temelde aynı yazılımı Red Hat desteği olmadan elde etmeniz anlamına gelir.

CentOS özel olarak RHEL'e çok yakındır ve Red Hat size CentOS makineleriniz için destek satmaktan mutlu olacaktır. Bunun için RHEL yüklemenize bile gerek kalmaz.

2.4.3 SUSE

Alman şirketi SUSE "Gesellschaft für Software- und System-Entwicklung" ismi altında Unix desteği veren bir firma olarak 1992 yılında kuruldu. Ürünlerinden biri Patrick Volkerding'in Linux dağıtımı idi, ilk tamamlanmış Linux dağıtımı olan Slackware'den türetilmişti. Yavaş yavaş Red Hat'ten RPM paket yönetimi ya da /etc/sysconfig dosyası gibi bazı özellikler alarak Slackware'den ayrılmaya

başladı. Slackware gibi görünmeyen ilk S.u.S.E sürümü 1996'nin sürüm 4.6'sı idi. SuSE(isimdeki noktalar bir zaman sonra kayboldu) kısa sürede Alman dilindeki bir Linux dağıtımı olarak öne çıktı ve "Kişisel" ve "Profesyonel" olmak üzere iki tane toplu set halinde dağıtılmaya başlandı.

Kasım 2003'te Amerikan yazılım şirketi Novell SuSE'yi 210 milyon dolara aldığını ilan etti; anlaşma sonuçlandırıldı. (Bu noktadan sonra SuSE SUSE olarak değişti) Nisan 2011'de Novell SuSE ile birlikte Attachmate adlı bir firma tarafından alındı, bu firma terminal emülatörleri, sistem monitörleme ve uygulama taşıma ve Linux ve açık kaynak camiaları tarafından önceden keşfedilmemiş işler yapıyordu. O zamandan beri Novell birisi SUSE olmak üzere iki ayrı iş kolunda çalışmaya devam etti. araçları konusunda çalışıyordu.

Red Hat gibi SUSE'de bir şirket Linux'u önerir, SUSE Linux Enterprise Server (SLES, http://www.suse.com/products/server/). Ayrıca SUSE Linux Enterprise Desktop (SLED) dağıtımıda bulunur, bu dağıtım masaüstü iş istasyonlarında kullanılmak üzere tasarlanmıştır. SLES ve SLED içerdikleri paketler açısından değişiklik gösterirler; SLES daha çok sunucu yazılımına yönelikken SLED daha interaktif yazılımları kullanır.

Şekil 2.2: Debian Projesinin Örgütsel Yapısı

SUSE'de, bireysel kullanıcılar için bir dağıtım çıkarıyor ve bu serbestçe kullanılabilir bir durumdadır, "openSUSE" (http://www.opensuse.org/), eski zamanlarda dağıtım optik medyalarda dağıtıldıktan birkça ay sonra indirmek için internete konulurdu. Red Hat'in aksine SUSE hala lisanslı ürünler içeren paket ürünler satar. Fedora'nın aksine openSUSE hala kısa yaşam süreleri içeren ciddi bir platformdur.

SUSE dağıtımının fark edilir ürünlerinden biri kapsamlı bir grafiksel sistem yönetim aracı olan "YaST" dır.

2.4.4 Debian

İki büyük Linux dağıtımı firması olan Red Hat ve Novell/SUSE'nin aksine Debian projesi (www.debian.org) amaçları yüksek kalite bir Linux dağıtımı olan ve Debian GNU/Linux olarak adlandırılan bir dağıtımı herkes için kullanılabilir yapmak olan gönüllülerin çalışmasının bir sonucudur. Debian projesi Ian Murdock tarafından 16 Ağustos 1993 yılında duyuruldu, isim kendi ismi ile o zamanlar kız arkadaşı (şimdi eski karısı) olan Debra'nın isimlerinin bir birleşimi olarak ortaya çıktı. (Bu sebeple debb-ian olarak telaffuz ediliyor) Şu anda proje 1000'den fazla gönüllü yardımına sahiptir.

Debian üç belgeye dayanmaktadır:

- Debian Özgür Yazılım Kılavuzları (DFSG – Debian Free Software Guidelines) hangi yazılım projesinin özgür olarak kabul edilebileceğini tanımlar. Bu önemlidir, çünkü sadece DFSG-özgür yazılımları Debian GNU/Linux dağıtımına kesin olarak katılabilir. Proje ayrıca özgür olmayan yazılımlarıda

içerir, bunlar dağıtımın sunucusunda DFSG özgür yazılımlarından kesin bir şekilde ayrı tutulur. İkincisi main diye adlandırılan alt dizinin non-free bölümündedir. contrib olarak adlandırılan bir orta alan bulunur, bu bölüm kendi içinde DFSG yazılımı olup özgür olmayan yazılımlar olmadan çalışamayan yazılımları içerir.

- Toplumsal Sözleşme projenin amaçlarını tanımlar.
- Debian Tüzüğü projenin organizasyon kısmını tanımlar. (Şekil 2.2'ye bakınız)

Herhangi bir zamanda Debian GNU/Linux'un en az üç sürümü bulunur. Yeni ya da düzeltilmiş sürüm paketleri unstable bölümüne konulur. Eğer belirli bir zaman aralığında pakette önemli hatalar olmadığı anlaşılırsa bu testing bölümüne kopyalanır. Genelde testing bölümünün içeriği "dondurulmuş" olarak geçer ve çok ayrıntılı bir şekilde test edilir. En sonunda stable olarak yayınlanır. Debian GNU/Linux'un en sık eleştirilen özelliklerinden birisi stable sürümler arasındaki zaman uzunluğudur, ama genelde bu bir avantaj olarak kabul edilir. Debian projesi sadece Debian GNU/Linux'u indirmeye sunar; medyalar üçüncü parti üreticilerden sağlanır.

Organizasyonunun bir fazileti olarak, özgür olması ve özgür ve özgür olmayan yazılımlar arasında temiz bir ayrım yapması nedeniyle Debian GNU/Linux türetilmiş projeler için bir temeldir. Bu türetilmiş projelerden en popülerleri Knoppix (Linux'u yüklemeden deneyebileceğiniz bir Live CD), SkoleLinux (özellikle okullar için ayarlanmış bir Linux sürümü) ya da ticari bir sürüm olan Xandros'tur. Limux'da, Münih şehir yönetimince kullanılan masaüstü Linux türevi, bir Debian GNU/Linux türevidir.

2.4.5 Ubuntu

En popüler Debian türevlerinden biri olan Ubuntu Güney Afrikalı bir girişimci olan Mark Shuttleworth tarafından bir İngiliz firması Canoncial Ltd. aracılığı ile sunulmuştur. ("Ubuntu" kelimesi Zulu dilinden gelmektedir ve kabaca "insanlık" demektir.) Ubuntu'nun amacı Debian GNU/Linux tabanlı düzenli aralıklarla güncellenen güncel, yetenekli, anlaması-kolay bir Linux sürümü sunmaktır. Örnek olarak Debian on ya da daha fazla mimariye yönelik çalışabilmesine rağmen Ubuntu sadece üç mimariye hizmet sunar.

Ubuntu, Debian GNU/Linux'un kararsız sürümüne dayanır ve büyük bir parçası yazılımlar için aynı araçları kullanır ama Debian ve Ubuntu yazılım paketleri her zaman için karşılıklı olarak uyumlu olmak zorunda değildir. Ubuntu her altı ayda bir döngüde yeni sürüm yayınlar ve her iki yılda beş yıl boyunca güncelleme alan bir uzun-süreli-destek (LTS – long-term support) sürüm sunar.

Bazı Ubuntu geliştiricileri ayrıca Debian projesinde de aktif olarak katılımcıdır ve bu belli derecede yazılım alışverişini destekler. Ama bütün Debian geliştiricileri yaratıcılık adına yaptığı kısaltmalar konusunda çok hevesli değildir, çünkü Debian çok çaba gerektirsede kapsamlı çözümleri arar. Ek olarak Ubuntu Debian kadar kendisini özgür yazılıma borçlu hissetmez; Debian'ın tüm altyapu yazılımları özgürce bulunabilecekken bu Ubuntu için her zaman geçerli değildir.

Ubuntu çekici bir masaüstü sistemi olmak dışında RHEL ya da SLES gibi sunucu alanında daha yerleşik bir sistem olmanın peşinde ve bunu uzun süreli sürümler ve iyi bir destek sunarak yapmayı planlıyor. Canoncial şirketinin bundan nasıl para kazanacağı pek belli değil, proje başladığından beri genelde Mark Shuttleworth'un kendi internet sertifika kanıtlamasını, Thawte, Verisign'a sattığındandan beri iyice dolmuş olan özel kasasından destekleniyor.

2.4.6 Diğerleri

Burada bahsettiğimiz dağıtımların dışında birçok dağıtım bulunmaktadır. Red Hat ve SUSE'nin küçük rakipleri Mandriva Linux ya da Turbolinux bunlardan bazılarıdır. Gentoo gibi kaynak koduna odaklanan sürümlerde bulunmaktadır. Güvenlik duvarlarından oyunlara ya da çoklu ortam platformlara ya da çok karışık sistemler için kullanılabilecek birçok değişik amaç için birçok değişik "çalışan sistemler" bulunmaktadır.

Ayrıca Linux dağıtımı olarak kabul edilebilecek Android'den de bahsetmek gerekir. Android Google tarafından sağlanan ve Google'ın Java sürümüne (Dalvik) dayanan bir kullanıcı uzayı ortamı (GNU, X, KDE gibi normal dağıtımlarda bulunan kullanıcı arayüzleri yerine) barındıran bir Linux işletim sistemi çekirdeği taşır. Bir Android telefonu ya da tableti kendisini kullanıcıya Debian ya da openSUSE'nin çok aksi bir yönde tanıtır ama yinede tartışılabilir bir şekilde bir Linux sistemidir.

Çoğu Android kullanıcısı sistemlerini telefon ya da tabletlerine yüklenmiş olarak alıp hiç değiştirmiyorlar bu sistemi ama çoğu Android tabanlı cihazlar (bazen hackleme ile) varolan alternatif bir Android sürümünün yüklenmesine izin verir. Çoğu cihaz için en güncel Android sürümünü elde etmenin yolu cihaz üreticisinin ya da telefon servis sağlayacısının resmi bir güncel sürüm yayınlaması ile olur.

2.4.7 Farklılıklar ve Benzerlikler

Çok fazla Linux dağıtımı olmasına rağmen en azından büyük dağıtımların günlük hayatta kullanımları oldukça benzer hale geldi. Bu bir parça neredeyse aynı temel programları kullanıyor olmalarından dolayıdır – örneğin komut satırı yorumlayıcısı neredeyse her zaman bash olarak bulunur. Diğer açıdan bakıldığında standartlar büyüme hızını kısıtlamaya çalışır. Bu Dosya Sistemi Hiyerarşi Standardı ya da Linux Standart Tabanını kapsar, LST üçüncü parti

yazılımların oldukça fazla Linux sürümünde kullanılabilmesi için Linux'un temel bir sürümünün bulunması gerektiğini söyler.

Ne yazık ki, LST beklenildiği ve olması gerektiği gibi bir başarı olmadı – sık sık Linux'un gelişimini yavaşlatmak ya da durdurmak amaçlı olduğuna dair yanlış anlaşıldı ve çeşitliliği azaltmak (çoğu dağıtımlar LST ortamını kendi ortamlarıyla aynı anda paralel olarak sunsa bile) hedeflenen üçüncü parti yazılımları kendisine çekmek için tasarlanmıştı ancak bu yazılımlar kendi yazılım paketlerini genelde RHEL ve SLES gibi ana şirket dağıtımları için kullanılabilir hale getirdi ve sadece bu ortamları desteklemeye başladı. SAP ya da Oracle'ı diyelim ki Debian GNU/Linux üzerinde çalıştırmak mümkün olsa da, RHEL ve SLES için yapılan lisans harcaması ile bu büyük yazılım paketleri için yapılacak lisanslama bedeli temel olarak çok fark edilebilir bir fark yaratmaz.

Dağıtımların farklı olduğunun fark edildiği bir alan yazılım paketlerinin yönetilmesi (yükleme ve kaldırma) ve dağıtımla gelen paketlerin dosya biçimidir. Bu konuda iki genel yaklaşım bulunur. Birisi Debian GNU/Linux ("deb") ve diğeri de Red Hat ("rpm") tarafından geliştirilen dosya biçimidir. Genel olarak ikiside birbirine açıkça bir üstünlük sağlamaz ama yine de güçlü yönleri değişimlerini engellemektedir. Deb yaklaşımı Debian, Ubuntu ve diğer Debian türevlerinde kullanılırken Rad Hat, SUSE ve bunlardan geliştirilen dağıtımlar rpm üzerinde çalışır.

Tablo 2.1: En önemli Linux dağıtımlarının karşılaştırması (Şubat 2012 itibariyle) –

İki yaklaşımda yazılım paketleri arasındaki bağımlılıklarda sistemi kararlı tutmaya yardımcı ve sistemde bir paketin silinmesinde diğerlerinin bağlılıklarının silinmemesini sağlayan oldukça kapsamlı bir bağımlılık yönetimi sunar. (ya da bir yazılımın bağımlılıkları önceden kurulmuşsa bunların tekrar kurulmamasını sağlar.)

Windows ve OS X kullanıcıları yazılımlarını bazı kaynaklardan [7] elde ederler. En azından Debian, openSUSE ya da Ubuntu gibi büyük dağıtımlar kullanıcılarına çok kapsamlı bir listeden direkt olarak paket yönetim araçlarıyla program kurabilmelerine olanak sağlarlar. Bu dağıtımların depolarına erişime izin verir ve bir ağ üzerinden bağlılıkları ile birlikte bir programın kurulabilmesi için arama seçeneği sunar.

Aynı temel paket biçimlerini kullansalarda (deb ya da rpm) her zaman dağıtımlar arasında paketleri kullanmak mümkün değildir – paketler paket biçimlerinden daha çok dağıtımların nasıl çalıştığı ile ilgili bilgiler içerir. Bu tümüyle yapılamaz bir durum değildir elbette (örneğin Debian GNU/Linux ve Ubuntu doğru koşullarda birbirlerinin paketlerini kullanabilir) ama bir kural

[7] Şu anda Apple ve Microsoft, Akıllı telefonlardan bildiğimiz Merkezi bir yazılım deposunun kendilerine getirmeye oldukça istekliler.

olarak, bir sistem ne kadar sistemde derine etki ediyorsa bir sorun oluşması o kadar büyüktür – sadece birkaç çalıştırma satırı içeren programlar ve onların belgeleri daha az sorun yaratırken makinanın başlangıç servisine gönderilen bir sistem servisi çok daha fazla sorun yaratabilir.

Tablo 2.1 en önemli Linux dağıtımlarının genel özelliklerini gösterir. Daha fazla bilgi için DistroWatch'a ya da dağıtımların kendi internet sitelerine göz atmanızı tavsiye ederiz.

Özet

- İlk Linux sürümü Linus Torvalds tarafından geliştirilip özgür yazılım olarak internette yayınlandı. Bugün dünya genelindeki yüzlerce geliştirici sistemi güncellemek ve geliştirmek için birlik oluyorlar.
- Özgür yazılım, yazılımları istediğiniz amaçlar için kullanmanıza olanak sağlar, kodu değiştirebilirsiniz ve başka insanlara değiştirilmiş ya da değiştirilmemiş kopyaları dağıtabilirsiniz.
- Özgür yazılım lisansları alıcıya başka türlü sahip olamayacakları hakları verir çünkü genelde lisans anlaşmaları alıcının haklarını kısıtlamaya yöneliktir.
- GPL çok popüler bir özgür yazılım lisansıdır.
- Özgür yazılım için olan diğer yaygın lisanslar bulunmaktadır. BSD lisansı, Apache lisansı ya da Mozilla Genel lisansı bunlara dahildir. Yaratıcı-yaygınlık lisansı yazılım dışında kültürel işler için kullanılır.
- Her amaç için kullanılabilinecek çok geniş bir ölçekte özgür ve açık kaynak kodlu yazılım bulunmaktadır.
- Birçok farklı Linux dağıtımı bulunmaktadır. Bunlardan en popüler olanları Red Hat ve Fedora, SUSE ve openSUSE, Debian ve Ubuntu'dur.

Bölüm 3

Linux ile İlk Adımlar

Amaçlar

- Temel Linux işlevselliğini denemek
- Metin editörü kullanarak dosyaları oluşturmayı ve düzenlemeyi öğrenmek

Önceden Bilinmesi Gerekenler

- Diğer işletim sistemlerini temel düzeyde bilmek

3.1 Giriş ve Çıkış yapmak

Linux, kullanıcılar arasında ayrım yapar. Bunun sonucu olarak bilgisayar açıldıktan hemen sonra bilgisayarı kullanmaya başlayamayabilirsiniz. Önce bilgisayara kim olduğunuzu söylemelisiniz. Yani giriş yapmalısınız. Sistem verdiğiniz bilgiye dayanarak ne yapacağına ya da ne yapmayacağına karar verir. Elbette sisteme erişim hakkına ihtiyaç duyarsınız (bir "hesap"). Sistem yöneticisi size geçerli bir kullanıcı adı ve parola atamış olmalıdır. Parola sizin hesabınıza sadece sizin kullandığınıza emin olmanızı sağlar. Onu gizli tutmalı ve başka kimseye söylememelisiniz. Sizin kullanıcı adınızı ve parolanızı bilen birisi sistemden tüm dosyalarınızı silebilir veya onları okumanızı engelleyebilir, e-mail atmanızı engelleyebilir yani genel olarak istemeyeceğiniz herşeyi yapabilir.

Bazı modern Linux dağıtımları işiniz kolaylaştırır ve sadece sizin kullandığınız bilgisayarda giriş sürecini atlamanıza izin verir. Eğer bu gibi bir sistem kullanıyorsanız giriş yapmak zorunda değilsiniz fakat bilgisayarınız doğrudan sizin oturumunuzla başlar. Bunun avantajları elbette vardır ama

sadece sizin bilgisayarınıza üçüncü bir erişimin olmadığını öngörürsek. Kaybolmayı veya çalınmayı önleyen diğer mobil sistemlerde veya dizüstü bilgisayarlarda bundan kaçınılmalıdır.

Görsel ortama giriş yapmak Günümüzde giriş ekranları görselleşmiştir ve giriş adımları görsel bir ekranda yer almıştır. Bilgisayarınız sizin kullanıcı adınızı ve parolanızı girmenizi sağlayan bir form gösterir.

Parolanızı girdiğinizde sadece yıldız görüyorsanız endişelenmeyin. Bunun anlamı bilgisayarınızın girişi yanlış anladığı değil, sizi izleyen insanlar için parolanın görünmesini zorlaştırmak istemesidir.

Giriş yaptıktan sonra bilgisayarınız sizin için görsel bir oturum açar. Görsel oturumun anlamı uygulama programlarına erişebileceğiniz menü ve simgelerin olmasıdır. Linux'un desteklediği çoğu görsel ortamlar, zamanından önce kapanan oturumu geri yüklemek için "oturum yönetimi"ni destekler. Bu sizin hangi programı çalıştırdığınızı, ekranda hangi pencerede olduğunuzu veya hangi dosyaları kullanıyor olduğunuzu bilmenizi gerektirmez.

Görsel ortamdan çıkış yapmak Çalışmanızı bitirdiyseniz veya başka bir kullanıcı için bilgisayarı kullanmaya uygun hale getirmek istiyorsanız çıkış yapmanız gerekir. Bu çok önemlidir. Çünkü oturum yöneticisi gelecekteki kullanımınız için sizin o andaki oturumunuzu kaydeder. Detaylı olarak çıkışın nasıl çalışacağı görsel ortama bağlıdır. Fakat kural olarak bir menü vardır. Şüphelendiğinde dökümanlara veya sistem yöneticinize başvurun.

Metin konsoluna girişi yapmak Masaüstü sistemlerin aksine, sunucu sistemleri sıklıkla sadece bir metin konsolunu destekler veya gereken zamandan daha fazla kalmak istemeyeceğiniz gürültülü makine odaları içinde kurulumu destekler. Bu gibi bilgisayarlara ağ yoluyla giriş yapmayı tercih edeceksiniz. Her iki durumdada görsel giriş ekranı göremeyeceksiniz ama bilgisayar size kullanıcı isminizi ve parolanızı doğrudan soracak. Mesela bunun gibi birşey görebilirsiniz.

```
computer login: _
```

Eğer bilgisayarın "computer" ismiyle sormasını istersek bu böyledir. Burada kullanıcı adınızı girmelisiniz ve "enter" tuşuna basmalısınız. Daha sonra bilgisayar size parolanızı soracaktır.

```
password:
```

Burada parolanızı girin. Bu kez parolayı yazarken hiçbirşey görmeyeceksiniz. Eğer hem kullanıcı adını hemde parolanızı doğru girdiyseniz sistem girişinizi

kabul edecektir. Daha sonra kabuk(komut satırı yorumlayıcısı) başlar ve komutları girmek ve programları çağırmak için klavyeyi kullanmanıza izin verir. Giriş yaptıktan sonra "ev dizini" sizin dosyalarınızın bulunduğu yer ekrana gelir.

Eğer güvenli kabuk kullanırsanız, mesela başka bir makineye ağ yoluyla giriş yaptığınızda bağlandığız bilgisayardaki kullanıcı adıyla sizin kullandığınız bilgisayarın kullanıcı adını aynı olarak varsayar ve kullanıcı adını sormaz. Detaylar bu kılavuzun kapsamı dışındadır. Güvenli kabuk Linux Administration II kitabında detaylı olarak anlatılmıştır.

Metin konsolundan çıkış yapmak metin konsolunda "logout" komutunu kullanarak çıkış yapabilirsiniz.

```
$ logout
```

Sistemden çıkış yaptığınızda metin konsolu üzerinde sistem başlama mesajını ve diğer kullanıcı giriş iletisini gösterir. Güvenli kabuk oturumuyla yerel bilgisayarınızdan başka bir komut iletisini elde edebilirsiniz.

Alıştırmalar

- Sisteme giriş yapmayı deneyin. Daha sonrada çıkış yapın. Kullanıcı adı ve parolanızı sistem dökümanları içerisinde bulacaksınız veya öğretmeniniz size söyleyecek?
- Eğer varolmayan bir kullanıcı adıyla giriş yapılırsa veya yanlış parola girilirse ne olur? Herhangibir sıradışı birşey farkettiniz mi? Sistemin böyle davranmasına sebep olan şey nedir?

3.2 Masaüstü Ortamı ve Tarayıcı

3.2.1 Görsel Masaüstü Ortamı

Görsel ortama giriş yaptığınızda Linux bilgisayarınız modern bilgisayarlarda gördüğünüzden daha farklı bir masaüstü göstermeyecektir.

Malesef bizim için burada iki Linux'un aynı olduğunu belirlemek mümkün değil. Resmi görsel ortamlarla gelen Windows veya Macintosh sistemlerinin aksine Linux sistem kurulumunda bir çok görsel ortam arasından birini seçmeni öneren dağıtımlarla gelir.

- KDE ve GNOME, benzer bir "look and feel" ile birlikte geniş kapsamlı uygun uygulamaları sağlamaya çalışan masaüstü ortamlarıdır. KDE ve GNOME'un hedefi hazır sistemlerin daha iyisini ve karşılaştırılabilir

deneyimlerini sunmaktır. KDE ve GNOME yenilikçi özellikler içerir. KDE' de arka plandaki dosyaların ve dökümanların indekslerini anlamsal arama özellikleri vardır ve geçen ay ispanyada çekildiğim fotoğraflara diskte nerede saklandığına bakmaksızın uygun erişim izni verdiği varsayılır [1]. Kabaca konuşursak, KDE ileri bilgili kullanıcılar için geniş kapsamlı özelleştirmeye odaklanır, GNOME ise basitlik ve kullanılabilirlikle ilgilenir. Mesela mümkün olmayan veya daha az değiştirilebilir şeyleri sağlamaya yönelimlidir.

- LXDE ve XFCE "lightweight" ortamlardır. Temel yaklaşımlarda KDE ve GNOME'a benzer fakat daha çok kaynakların ekonomik kullanımına yönelmiştir, bu yüzden anlamsal arama gibi çeşitli yük getiren servisler içermezler.

- Tüm masaüstü ortamını kullanmaktansa pencere yöneticilerinin bir grubunu kullanmayı tercih edebilirsiniz. İlk olarak KDE'den önce bu bakış açısında bir standartlaşma kurulmuştur. Bu böyle şeyleri yapmak için genel bir yol olmuştur. Fakat bugün Linux kullanıcılarının çoğunluğu görsel ortamların dahada iyileştirilmesine inanır.

Aynı görsel ortamı kullanan iki farklı dağıtım bile olsa bu onların aynı görüneceği anlamına gelmez. Genellikle görsel ortamlar onların temalarına dayalı görünüm özelleştirmesine daha fazla yer ayırır ve dağıtımlar bunu kendini diğerlerinden ayırmak için kullanır. Arabaları göz önüne alalım. Tüm arabalar neredeyse 4 tekerleğe bir rüzgarlığa sahip fakat siz hiçbir zaman BMW'yi Citroen veya Ferrari ile karıştırmazsınız.

Kontrol çubuğu herhangi bir olayda kontrol çubuğu(yatay araç çubuk(dock), panel yada neye sahipsen) ekranın ya en üstünde yada en altında senin önemli uygulama programlarına giriş yapmana izin veren veya bilgisayarı kapatmana yada çıkış yapmanı sağlayan menülerdir. KDE windowstaki başlangıç butonuna benzeyen bir panele bağlıdır. Panelde aynı windowstaki başlangıç butonu gibi programların menüsünü açar. GNOME başlangıç butonu kullanmaz fakat ekranın en üstüne menü çubuğunu taşır. En önemli yada en çok kullandığınız programlar ekranın solundaki çek-bırak menüye eklenip kolayca erişilebilir.

Dosya Yöneticisi Görsel ortamlar genellikle disk üzerinde dizinlere (dosyalara) erişmenize izin veren ve onların içeriklerini alt dizinlere ayırmanıza ve dosyaları işletmenize olanak sağlayan bir dosya yöneticisine sahiptir. Buradaki yöntemlerin diğer görsel sistemlerden çokta fazla farkı yoktur. Bir pencereyi bir dizinden

[1] Microsof bir kaç kez bunu yapacağını vaat etti ama sonraki Windows versiyonu piyasaya çıkmadan önce yenilik listesinden çıkarıldı.

diğerine sürükleyerek kopyalayabilir veya taşıyabilirsiniz ve isterseniz farenin sağ butonuna tıklayarak içerik menüsünü açabilir dosyaya farklı işlemler uygulayabilirsiniz. Deneyin.

Dock sıklıkla kullanılan dosyalar ve programlar genellikle ekranın altında saklanır veya hızlı erişim için ekranda sabit bir yere yerleştirilir.

Sanal masaüstleri çoğu Linux tabanlı görsel ortamların genel bir özelliği Windows ve OS X'in sunmadığı sanal masaüstleridir. Bu sanal masaüstleri ekranda boş bir alanda çoğaltılabilir. Arka ve ön planda çalıştırılarak program pencerelerinin kendi seçimleriyle masaüstleri bir çok kez simule edilebilir. Bunlar çalıştığın program veya dökümanların masaüstünde istediğin yerde olabilmesine izin verir, web tarayıcın ve eposta okuyucun için masaüstünde yer ayırabilirsiniz ve programlanmış masaüstünde pencerelere yeniden düzenleme zorunluluğun olmadan çabucak eposta mesajlarını oluşturabilirsin.

3.2.2 Tarayıcılar

Modern bilgisayarlar üstünde en ünlü programlardan biri web tarayıcılarıdır. İyiki en popüler tarayıcılar açık kaynaklı programlardır ve Firefox veya Google Chrome Windows veya OS X ve Linux için uygundur (dağıtımınız muhtemelen Google Chrome sunmaz fakat açık kaynak çeşidi olan Chromium'u sunar, aralarında çokta bir fark yoktur) internet girişi gibi bir giriş için uygulama menüsüne baktığınızda tarayıcıyı bulabilirsiniz.

Ticari durumlara rağmen Debian GNU Linux sistemler ve çeşitli türetimleri üzerinde Firefox tarayıcısının "Iceweasel" olarak adlandırılmıştır. Çünkü Mozilla vakfı, Firefox üreticileri tarayıcının tamamlanmamış versiyonunun dağıtımına yalnızca kod resmi versiyona uyduğunda izin verir. Debian projesi tamir ve güvenlik problemleri haklarını elinde tuttuğundan beri kopyalama ve ticari dağıtımları çok ciddi bir şekilde ele alır. Buda ismin değişmesi zorunluluğunu getirir. (Diğer dağıtımlar resmi versiyonda kalır veya isim hakkıyla ilgilenmez yeni bir isim bulur.)

3.2.3 Terminaller ve Kabuklar

Görsel Linux ortamı içerisinde bile sıklıkla bir kabuk içerisinde metinsel komutları girebileceğin erişime uygun bir terminal penceresi mevcuttur (bu klavuzun geri kalanı çoğunlukla kabuk komutlarından bahseder, bu nedenle bunlara ihtiyacınız olabilir).

Çoğu Linux masaüstü ortamları üstünde bir terminal penceresi bir fare tıklaması uzaklığındadır. Debian GNU/Linux üstündeki KDE içinde konsole isimli bir giriş sistem altındaki başlangıç menüsünde vardır. Sistem bir

kabuğu yürüten uygun programları açar ve metinsel komutları yürütür. Benzeri yöntemler diğer masaüstü ortamları ve dağıtımlar üstündede vardır.

Alıştırmalar

- Bilgisayarınızda hangi görsel ortam kurulu? Bir dosya yöneticisi aç ve dosya veya dizin simgesi üzerinde sağ tıkladığında ne olduğunu anlamaya çalışın. Eğer arka planda simgeler arasında boş bir pencere üstünde sağ tıklarsanız ne olur? Bir dosyayı bir dizinden diğerine nasıl taşırsın? Yeni bir dosya veya dizin nasıl oluşturursun? Bir dosyaya nasıl yeni bir isim verirsin?

- Bilgisayarınızda hangi web tarayıcısı kurulu? Birden fazla var mı? Tarayıcıyı başlatmayı deneyin ve onların çalıştığından emin olun.

- Bir terminal penceresi açıp kapatın. Terminal pencere programın aynı pencere içinde birden çok oturumu destekler mi? (tab ile alt pencereleri kulanmak mümkün.)

3.3 Metin Dosyalarını Oluşturmak ve Düzenlemek

Betik veya program yazmanız, sistem yöneticisi olarak yapılandırma dosyalarını düzenlemeniz veya basitçe bir alışveriş listesini not ediyor olmanız farketmez. Metin dosyalarını düzenlemede Linux en iyisidir. Bu yüzden Linux kullanıcısı olarak ilk adımlarınızdan birisi metin dosyalarının nasıl oluşturulduğunu ve düzenlendiğini öğrenmek olmalıdır. Bunun için seçtiğiniz araç metin editörüdür.

Linux metin editörleri deüişik şekillerde ve renklerde gelir. Burada terminal içinde çalışan acemi işi metin editörü (beginner-proof) GNU Nano'nun en önemli özelliklerini anlatarak kolay bir çıkış yolu sunacağız.

Elbette popüler görsel arayüzler, menülerle, araç çubuklarıyla bulunmaktadır. Bu kullanışlı iyi şeyler Windows üzerindeki Notepad veya daha iyisiyle karşılaştırılabilecek programlardır. KDE üstündeki Kate veya Gnome üstündeki gedit gibi. İki sebepten dolayı burada bu editörlere detaylı olarak bakmayacağız.

- Bu programlar kendilerini daha detaylı olarak açıklamaya eğilimlidirler ve gereğinden fazla ilginizi buraya çekmek istemiyoruz.

- Her zaman görsel arayüz kullanacak bir pozisyonda olmayacaksınız . belkide güvenli kabuk kullanarak uzakta bir bilgisayarda çalışıyor olabilirsiniz veya sunucu odasında sunucu konsolunun önünde duruyor olabilirsiniz ya da şansınıza sadece bir metin ekranına sahipsinizdir.

Her halukarda hayatının geri kalanı boyunca tek bir editör kullanmak zorunda değilsiniz. Hiçkimse Nano gibi bir editörden başka bir seçeneğiniz yokmuş gibi sizi görsel masaüstü bilgisayarınız üstünde görsel bir editör kullanmaktan alıkoyamaz.

Gelenekçi Linux tutkunları Nano gibi birşeyi küçümserler: gerçek Linux profesyoneli için doğru seçim editörü vi ("vi ay" diye okunur), klavye özelliği ok tuşlarına güvenmeyen ve makine odalarını doldurmuş metin terminallerinin yeşil ışığında zamana direnmiş yaşayan bir fosile benzetilir. Eğer sistem yöneticisi üzerinde bir kariyere girişirseniz yakın veya uzak gelecekte vi ile tanışmalısınız. En azından orta seviyede vi bilmek gerekir. Bu yüzden vi özellikle Linux'ta ve her Unix çeşidi üstünde geniş olarak kullanılmaya değer tek editördü. Fakat şu anda bu böyle değil.

GNU Nano pico isimli editörün basit bir klonudur. pico PINE eposta paketinin bir bölümüdür. (PINE kabul edilmiş tanımlara göre ücretsiz bir yazılım değildir bu yüzden GNU projesi yeni bir editör yazdı. Bu arada PINE'ın ileri sürümü alpine adı altında özgür olarak mevcuttur ve pico'nun özgür bir versiyonunuda taşır.) Çoğu dağıtım GNU Nano veya pico'yu önerir buda basitlik içindir. Bölümün geri kalanında GNU Nanodan bahsedeceğiz. Pratik olarak söylediğimiz herşey picoyada uygulanabilir.

Orjinal pico ile karşılaştırırsak GNU Nano özellikleri bazı uzantılara sahiptir.(bu uzantılar bir sürpriz göz önüne alarak gelmemiştir. Zaten isme göre çoktan mevcuttur. En önemli 3 editörden biridir.) fakat bunların çoğu bizimle doğrudan ilgilenmez. Gerçekten çok açık olan sadece bir uzantı vardır. GNU Nano uluslar arası hale getirilmiştir(internationalized). Böylece bir sistem üstünde diğer taraftan almanca dilini kullanmak için kurarsınız. Almanca olarak metinleri mesajla yazabilirsiniz.

GNU Nano çoğunlukla terminal penceresi içerisinde hazır olarak başlar. (bölüm 3.2.3 aşağıdaki bir komut kullanır.)

```
$ nano myfile
```

($ burada sadece komut satırının[prompt] biçimlendirilmiş kısaltmasıdır. Burada sisteminizi daha süslü gösterebilir ve $ girmeniz gerekmez. Enter tuşuna basarak komutu tamamlamayı unutmayınız.) Sonuç olarak figure 3.1'e benzeyen birşey görebilirsiniz. O en üstte işaretlenmiş bir satırla çoğu boş bir pencere ve en aşağıda yardım satırlarıyla, komutların özet açıklamaları olan önemli bir liste, olarak gösterilmiştir. Yardım satırının hemen üstündeki satır durum satırıdır. Durum satırı diske veri kaydettiğin zaman dosya isimlerini girebileceğin Nanoda gözükecek mesajlardır.

Ekranda daha fazla kullanılabilir boşluğa ihtiyacınız varsa yardım satırını gizlemek için "alt + x " basabilirsiniz.(space tuşunun solundaki "alt" tuşuna basılı tutarken x'e basın) tekrar basarsanız yardım satırını tekrar gösterir. (Eğer dahada boşluk istersen yardım satırının hemen altındaki top satırını "alt + o" kullanarak gizleyebilirsiniz.)

Metin girişi ve değişmesi Yeni bir metin girmek için Nano penceresinin içinde yazmaya başlanır. Eğer hata yaparsan "backspace" tuşu imleçin solundaki karakteri silecektir. Metinin içerisinde dolaşmak için ok tuşlarını kullanabilirsiniz. Mesela başlangıça yakın birşeyi değiştirmek için. Yeni bir şey yazarsan imleçin işaret ettiği yerde gözükecektir. "del" tuşu imleçten sonraki karakteri kaldırır ve satırın geri kalanının bir pozisyon sola hareket etmesine sebep olur. Bazı Nano versiyonları fareyi destekler,böylece görsel ekranda Nanoyu kullanırsanız veya metinsel ortam fareyi yönetirse, metin içerisinde bir yere imleçi yerleştirmek için o noktaya tıklayabilirsiniz. Fare desteğini açmak için "alt + m" basabilirsiniz.

Metni kaydetme Metin düzenlediğinde veya giriş yaptığında "ctrl + o" kullanarak kaydedebilirsiniz. Nano size dosya için bir isim sorar. (bölüm 6 sonunda dosya isimleri hakkında daha fazlasını göreceğiz.)Nano daha sonra dosya ismiyle metni kaydeder.

Nanodan çıkış Nanodan "ctrl + x" kullanarak çıkış yapabilirsiniz. Kaydedilmemiş veriler taşıyorsa Nano metni kaydedip etmeyeceğini sorar. Cevap "y" ise Nano ismini sorar veya "n" ise Nano hemen çıkar buda kaydedilmemiş verilerin atılmasına sebep olur.

Dosya ekleme Farklı bir dosya (önceden varolan) o anki metin dosyanıza "ctrl + r" kullanılarak eklenebilir. Eklediğin dosya imleçin pozisyonundan itibaren eklenecektir. Nano doğrudan sorar yada "ctrl + t" kullanarak dosya göz atıcısını açıp dosya seçmenizi sağlar.Bu arada bu dosyayı kaydettiğinizde("ctrl + o" kullanılarak)de çalışır.

Kesme ve yapıştırma Bir tampon içerisine satırı kaydetmek ve satırın taşıdığı imleci kaldırmak için "ctrl + k" komutunu kullanabiliriz.(dikkat: Nano her zaman imleçin satırdaki pozisyonuna bakmaksızın satırın tümünü kaldırır) "ctrl + u" tamponun içeriğini tekrardan yapıştıracaktır. Dikkatsizlik sonucu "ctrl + k" 'ya basarsanız veya basitçe satırı taşımak yerine kopyalamayı isterseniz "ctrl + u" kullanabilirsiniz.

Yapıştırmalar her zaman imlecin olduğu yerden itibaren gerçekleşir. Eğer imleç satırın ortasındaysa "ctrl + u" ya bastığınızda tampondan gelen satır, imlecin sağ tarafına yapıştırılır. İmleçin sağında ne olursa olsun yeni satır orjinal satır haline gelir.

Bir satırda bir çok kez "ctrl + k" ya basarak tampona daha fazla satır taşıyabilirsiniz. Bu satırların hepsi birden yapıştırılacaktır. Satırın sadece bir bölümünü kesmek istiyorsanız imleçin karşılık geldiği pozisyonda "ctrl + ^" (bazı klavyelerde "alt+a" ya basılarak elde edilebilir) basın. Ondan sonra imleçi keseceğiniz bölümün sonuna taşıyın. Nano yardımseverlikle kesmek için

seçmiş olduğunuz metini işaretleyecektir. Sonrasında o bölgeyi "ctrl + k" kullanarak tampona taşıyın. İmleçin altındaki karakterin kesilmemiş olduğunu unutmayın. Sonra "ctrl + u"ya basarak tamponun içindenki herhangibir yere üsttekileri yapıştırabilirsiniz.

Metin arama Eğer "ctrl + w" ya basarsan Nano sana metnin bir parçasını sormak için durum satırını kullanır. İmleç ondan sonra senin dökümanının içindeki metnin bir parçasında olduğu yere gider ve onun o anki pozisyonundan başlayarak devam eder.

Çevrimiçi yardım "ctrl + g" kullanarak Nanonun ana yardım ekranını görebilirsiniz. Ana yardım ekranın editörün temellerini ve çeşitli klavye komutlarını açıklar. (Burada açıkladığımızdan daha fazla komut var.) Yardım ekranını "ctrl + x" kullanarak kapatabilirsiniz. Bunlar GNU Nanonun en önemli özellikleridir. Bunları denemek en iyisidir. İstediğiniz gibi deneyebilirsiniz.

vi başlığına geri dönün (Linux gurularının editörü olduğunu hatırla) Eğer macera istiyorsanız sisteminizde vim editörünün kurulu olduğundan emin olun. Bu editör vi'nin go-to gerçekleştirimidir. (Çok nadir olarak az sayıda kişi Linux üstünde orjinal BSD vi kullanır.) vimtutor programını başlatın ve yarım saatinizi vi'ye girişiniz için harcayın (Linux dağıtımınıza bağlı olarak vimtutor'u ayrı bi paket olarak indirebilirisiniz. Şüphe içindeyseniz sistem yöneticinize veya bilen herhangibirine sorun).

Alıştırmalar

- GNU Nano'yu başlatın ve aşağıdaki metni girin.

```
Roses are red,
Violets are blue,
Linux are brilliant,
I know it is true.
```

  Bu metni roses.txt olarak kaydedin.

- önceki alıştırmadaki metinden alttaki satırı kesin

```
 Linux is brilliant
```

  Üç kere yapıştırıp metnin aşağıdaki gibi gözükmesini sağlayın.

```
Roses are red,
Violets are blue,
Linux are brilliant,
Linux are brilliant,
Linux are brilliant,
I know it is true.
```

Bu satırların ilkinin içindeki “is” in “i” üstündeki imleç pozisyonunu işaretleyin. 3. Satırdaki “is”in “i”sine yönlendirin ve işaretli bölgeyi kaldırın.

Bu bölümdeki komutlar

logout kabuk oturumunu sonlandırır

pico PINE/Alpine paketinden basit bir metin editörü

Özet

- Linux sistemi kullanmadan önce kullanıcı ismi ve parolanı girmek zorundasınız. Sistemi kullandıktan sonra tekrar çıkış yapmalısınız.
- Linux çeşitli görsel ortamlar sunar. bu görsel ortamların çoğu birbirine benzerdir ve açıkça birbirnden üretilmişlerdir.
- Terminal penceresi görsel ortam içerisindeki metinsel kabuk komutlarını girmene izin verir.
- GNU Nano basit bir metin editörüdür.

Bölüm 4

Kim Korkar Kabuktan

Amaçlar

- Komut-satırı kullanıcı arayüzünün önemini anlamak
- Bourne-Again Shell (Bash) komutları ile çalışmak
- Linux komutlarının yapısını anlamak

Önceden Bilinmesi Gerekenler

- Temel bilgisayar kullanım bilgisi faydalı olur.

4.1 Neden?

Linux (Unix gibi), diğer modern işletim sistemlerinden farklı olarak klavye ile metinsel komutlar girme fikrine dayanır. Windows tarzı, 15 yıldan bu yana izleyicilerini grafik arayüzlerin vazgeçilmez en önemli bileşen olduklarını düşündürterek beyin yıkayan sistemler kullananlar için çok eski bir teknik gibi gelebilir. Linux ortamına Windows ortamından geçen çoğu kişi için komut satırı arayüzü ilk başta karşılaştırmalı olarak bakılınca, bir 21. yy. insanının cep telefonu şebekesi olmayan, kötü masa adabı ve ölümcü dişçileri olan Kral Artur sarayına geçişindeki durum gibi bir "kültür şoku"dur.

Ama herşey o kadar göründügü gibi kötü değildir. Günümüzde Linux için de Windows'ta, Mac OS X'te olduğu gibi onlara eşit veya bazı noktalarda daha iyi kullanım sunan görsel arayüzler var. Öte yandan, görsel arayüzler ve metin odaklı komut satırı birbirini dışlayan değil, aslında tamamlayıcıdırlar ("Her iş için doğru araç" felsefesine göre).

Günün sonunda gelişmekte olan Linux kullanıcısı olmak dışında aynı zamanda "kabuk" olarak bilinen metin odaklı kullanıcı arayüzüne alışmış olacaksınız. Elbette kimse sizin görsel masaüstü kullanmanızı engellemek istemez. Ancak, kabuk ile yapabileceğimiz son derece güçlü, karmaşık operasyonları görsel olarak yapmak oldukça güçtür. Kabuğu ihmal etmek arabanın birinci vitesi dışında tüm viteslerini gereksiz saymak gibidir [1]. Tabi ki birinci vitesle de ulaşmak istediğiniz yere ulaşırsınız fakat bu zaman kaybına ve gereksiz gürültüye sebep olur. Öyleyse bu işlemin Linux altında nasıl yapıldığını niçin ögrenmeyelim? Şimdi iyice dikkat ederseniz size bu konuyla ilgili birkaç püf noktası aktaralım.

4.1.1 Kabuk nedir?

Kullanıcılar doğrudan doğruya işletim sisteminin çekirdeğiyle iletişim kuramazlar. Bu sadece "sistem çağrıları"nı kullanan programlarla mümkündür. Ancak, bir şekilde bu tür programları başlatabiliyor olmanız gerekir. Kabuk, klavyeden girilen komutları (genellikle) okuyup onları (örneğin) çalıştırılabilir komutlar haline getiren, özel bir kullanıcı programı olarak bu görevi üstlenir. Kabuk, bilgisayara "arayüz" hizmeti vererek asıl işletim sistemini bir kabuk gibi örter (kabuklu deiz hayvanının adındaki gibi) ve onu görünmekten saklar. Elbette ki kabuk, işletim sistemine erişen programlar arasından sadece bir tanesidir.

Bugünün KDE gibi görsel "masaüstleri" de "kabuk" olarak kabul edilebilir. Klavyeden girilen metin komutları yerine fare ile girilen grafik komutlarını okur, metin komutlarının belli bir dil bilgisini takip etmesi gibi aynı şey fareden alınarak gerçekleştirilir. Örneğin, fare yardımıyla nesneleri tıklayarak seçersiniz ve ne yapacağınıza karar verirsiniz: açmak, kopyalamak, silmek vs.

1960ların sonlarındaki Unix modelinde bile kabuk vardı. En eski kabuk 1970'lerin ortalarında "Unix'in 7. sürümü" için Stephen L.Bourne tarafından geliştirildi. "Bourne kabuğu" olarak isimlendirilen bu kabuk temel işlevleri yerine getirip yaygın olarak kullanılıyordu, ama bugün bu kabuk orijinal haliyle nadiren görülür. Diğer klasik Unix kabukları C kabuğunu içerir, bu kabuk Berkeley'de bulunan California Üniversitesi'nde C programlama dili ile oluşturulmuştur ve büyük ölçüde Bourne kabuğu ile uyumlu olmasına rağmen işlevsellik açısından daha gelişmiş Korn kabuğudur (David Korn tarafından geliştirildi, AT&T'de de geliştirildi).

Linux sistemlerinde standart, Bourne-again kabuğudur, kısacası *bash*. Bu kabuk, Brian Fox ve Chet Ramey tarafından Özgür Yazılım Vakfı'nın GNU projesi altında geliştirilmiş olup, Korn ve C kabuklarının birçok işlevini birleştirir.

[1] Bu benzetim elle vites kullanan Avrupalılar ve diğer insanlar içindir; bizim Amerikan okuyucularımız tabi ki otomatik vites kullanır. Eskiden hepsinin Windows kullanıyor olması gibi

Bahsettiğimiz kabukların haricinde başka kabuklar da mevcuttur. Unix üzerinde kabuk diğer programlar gibi sıradan bir uygulama programıdır, yeni bir tane yazmak için özel bir ayrıcalığa gerel yoktur. Sadece kabuğun diğer programlarla nasıl iletişim kurması gerektiği yönetmeliğinin kurallarına bağlı kalmanız yeterlidir.

Kabuklar kullanıcı komutlarını okumak için etkilesimli olarak çağrılabilirler (genellikle "terminal" üzerinde). Pek çok kabuk bir de dosyadan henüz işlenmemiş komut dizilerini okuyabilir. Bu tür dosyalara "kabuk betikleri" denir.

Kabuk aşağıdaki adımları takip eder:

1. Terminalden (veya dosyadan) komut okumak
2. Komutları onaylamak
3. Komutu doğrudan çalıştırmak veya karşılık gelen programı çalıştırmak
4. Sonucu ekrana (veya başka yere) vermek
5. 1. adımdan devam etmek

Standart komut döngüsü dışında, kabuk genellikle programlama dili gibi ilave özellikler de içerir. Bu da döngüler, durum kontrolleri ve değişkenleri kapsayan (daha çok kabuk betiklerinde, az sıklıktı da etkileşimli kullanımda) karmaşık komut yapıları içerir. Son zamanlarda kullanılan komutlar arasında dönerek yeniden kullanabilme gibi gelişmiş bir özellik kullanıcının hayatını kolaylaştırmaktadır.

Kabuk oturumları genellikle *exit* komutu ile sonlandırılabilir. Bu işlem oturum açtıktan hemen sonra elde ettiğimiz kabuk için de geçerlidir.

Her ne kadar yukarıda bahsettiğimiz gibi farklı kabuklar olsa da biz çoğu Linux dağıtımında standart kabuk olarak "bash" üzerinde odaklanalım. LPI sınavları da özellikle *bash* kullanımına işaret eder.

Alıştırmalar

1. Oturumunu kapatın ve tekrar açın, sonra da "*echo $0*" komutun çıktısını giriş kabuğunda kontrol edin. "*bash*" komutu ile yeni bir kabuk başlatın ve "*echo $0*" komutunu tekrar girin. İki komutun çıktısını karşılaştırın. Alışılmadık herhangi birşey farkettiniz mi?

4.2 Komutlar

4.2.1 Neden Komutlar?

Bir birgisayarın eylemleri, işletim sistemi ne olursa olsun üç adımla tanımlanabilir:

1. Bilgisayar kullanıcının veri girmesi için bekler
2. Kullanıcı komut seçer ve klavye ya da fare aracılığıyla komutu girer
3. Bilgisayar komutu gerçekleştirir

Linux sisteminde kabuk bir "*komut istemini*" görüntüler, bu da komutların girilebileceğini gösterir. Bu istem genellikle geçerli bir kullanıcı ve ana bilgisayar adını, mevcut dizini ve son karakteri içerir:

```
joe@red:/home > _
```

Bu örnek,*joe* kullanıcısının *red* adındaki bilgisayarın */home* dizininde bulunduğunu ifade etmektedir.

4.2.2 Komut Yapısı

Bir komut aslında karakterler dizisinden oluşur ve tuşuna basılmasıyla komut, kabuk tarafından değerlendirilir. Çoğu komut belli belirsiz İnglizce'den esinlenilmiştir ve onlara adanmış bir "komut dilini" oluşturur. Bu dilde komutlar belli kurallara, "sözdizimine" uymalıdır ki kabuk bunları yorumlayabilsin.

Kabuk komut satırını yorumlayabilmek için ilk önce satırı sözcüklere ayırır. Gerçek hayatta olduğu gibi sözcükler boşluklarla ayırılır. Satırdaki ilk sözcük genellikle asıl komuttur. Satırdaki geri kalan sözcükler detaylı olarak ne yapılmak istendiğini belirten parametrelerdir.

Kabuğun büyük ve küçük karakterleri birbirinden ayırabiliyor olması DOS ve Windows kullanıcılarını şaşırtabilir. Linux komutları genellikle sadece küçük harflerle (istisnalar kuralı kanıtlar) yazılır ve aksi halde anlaşılmaz. Ayrıca Bölüm **??**'e bakın.

Komutları sözcüklere ayırırken, kabuk için sözcüklerin arasında bir veya daha fazla boşluk karakterinin olması aynı şeydir. Aslında, kabuk için sözcükler arasında boşluk karakterinin olup olmamasının önemi yoktur; bir satır üzerinde dizme yapan karakterine de izin verilmiştir. Bu karakterin önemi de komutları dosyalardan okurkendir, çünkü kabuk doğrudan doğruya sekme karakterini girmeye izin vermez (en azından çemberlerin içinden atlamadan).

Bilgisayara girdiğiniz komut tek satıra sığmayacaksa bunu birkaç satırda ifade etmek de mümkün. Ama bunun bir komut girdisi olarak anlaşılmaması için satır sonlandırıcıdan önce (Enter) "Token \" karakteri yazılmalıdır. Komutların parametreleri kabaca ikiye ayırmak mümkündür:

- Tire("-") ile başlayan parametreler seçenekler diye isimlendirilir. Bunlar genelde, isteğe bağlıdır, detaylar söz konusu komuta göre değişir. Bunlara mecazi olarak "anahtarlar" demek mümkündür. Bunlar komutun bazı özelliklerinin açılıp kapanmasını sağlarlar. Eğer komuta birkaç tane seçenek eklenmek istenirse bunları ayrı ayrı tire karakteri ("-a -l -F") ile

yazabileceğimiz gibi tek tire karakteri ile de yapılabilir ("-alF"). Konsol komutları birden fazla seçenek alabilir. Bunların bazısı tek karakterle yazılabilen seçenekler olurken kimisi okunabilirliği arttırmak için uzun şekilde yazılan (genellikle de tekil karakter karşılıklarına ek olarak) seçeneklerdir. Uzun seçenekler çoğu zaman iki tire karakteri ile başlarlar ve birleştirilemezler: "foo –bar –baz".

- Tire ile başlamayan parametreler "bağımsız değişken" olarak adlandırılır. Bu da çoğu zaman komutun işlemesi gerektiği dosyaların adlarına karşılık gelir

Genel komut yapısı aşağıdaki gibi gösterilebilir:

- Komut - "Ne yapılacak?"
- Seçenekler - "Nasıl yapılacak?"
- Bağımız Değişkenler - "Ne ile yapılacak?"

Genellikle seçenekler komutlardan sonra, delillerden önce gelir. Ancak, komutların tümü bu şekilde kuralın işlemesini şart koşmazlar. Bazıları delil ve seçenekleri keyfi olarak karıştırabilirler ve bütün seçenekler komuttan sonra gelmiş gibi davranırlar. Diğerleri ile birlikte komut satırı sırayla işlenirken seçenekler de karşılaşıldığında dikkate alınır.

Geçerli Unix sistemlerinin (Linux dahil) komut yapısı 40 yıllık bir süreç içinde büyük bir gelişme sağladı ve bu nedenle bazen çeşitli tutarsızlıkların ve küçük sürprizlerin görülmesi doğaldır. Biz de daha tutarlı olması gerektiğine inanıyoruz ama 30 yıllık geçmişe sahip kabuk betiklerini tamamen göz ardı etmek zordur. Bu nedenle sık sık görülen küçük garipliklere hazırlıklı olun.

4.2.3 Komut Tipleri

Kabuklar içerisinde aslında iki tip komut vardır:

Dahili komutlar Bu komutlar kabuğun kendisi tarafından sunulmaktadır. Bourne-again hızlı gerçekleştirilebilen 30 kadar dahili komut içerir. Kabuğun durumunu değiştiren bazı komutlar (exit veya cd gibi) dışarıdan temin edilemez.

Harici komutlar Kabuk bu tür komutları kendi kendine çalıştırmaz ama çalıştırılabilir dosyaları başlatır. Ki bu tür dosyalar genelde /bin veya /usr/bin dizinleri altında bulunurlar. Bir kullanıcı olarak kendi programlarınızı temin edip kabuğun diğer tüm harici komutları çalıştırdığı gibi kendi programlarınızın çalıştırılmasını sağlayabilirsiniz.

Komutunuzun türünü öğrenmek için *type* komutunu kullanabilirsiniz. Bu komuta bağımsız değişken olarak olarak komut adı verirseniz çıktı olarak size komutun türünü ya da karşılık gelen dosya ismini verir, örneğin

```
$ type echo
echo is a shell builtin
$ type date
date is /bin/date
```

(*echo* komutu ilginç bir komut olup kendisine verilen parametreleri çıktı olarak verir:

```
$ echo Thou hast it now, king, Cawdor, Glamis, all
Thou hast it now, king, Cawdor, Glamis, all
```

date komutu da mevcut tarih ve saati ayarlanmış olan saat dilimi ve dile göre gösterir:

```
$ date
Mon May 7 15:32:03 CEST 2012
```

echo ve *date* komutları hakkında daha fazlası için **??** . Bölüme bakınız.)
help komutuyla dahili Bash komutları ile ilgili bilgiyi alabilirsiniz:

```
$ help type
type: type [-afptP] name [name ...]
  For each NAME, indicate how it would be interpreted if used as a
  command name.

  If the -t option is used, ‘type’ outputs a single word which is one of
  ‘alias’, ‘keyword’, ‘function’, ‘builtin’, ‘file’ or ‘’, if NAME is an
```

Alıştırmalar

1. Aşağıdaki programlardan hangileri harici ve hangileri kabuk içerisinde dahili olarak temin edilir: *alias, echo, rm, test*?

4.2.4 Daha da fazla kural

Yukarıda belirtildigi gibi, kabuk, komut girildiğinde büyük-küçük harflere duyarlıdır. Bu durum sadece komutlar için geçerli olmayıp seçenekleri ve parametreleri (çoğunlukla dosya isimlerini) de kapsar.

Şunu da göz önünde bulundurmalısınız ki kabuk, bazı karakterlerini özel giriş karakterleri olarak algılar. En önemlisi de daha önce belirtilmiş olan boşluk karakteri komut satırındaki sözcükleri ayırmak için kullanılmaktadır. Özel anlamı olan diğer karakterler şunlardır:

```
$&;(){}[]*?!<>"'
```

Tabi bunları kullanmak isteyebiliriz. Bu durumda onlar kabuk tarafından özel karakterler gibi yorumlanmasın diye karakterlerin önüne "\" işareti eklenir. "\" işareti ile sadece tek bir özel karakteri kabuktan kaçırabilirsiniz. Birden fazla özel karakterin kaçırılması içinse tek tırnak yada çift tırnak ('...','"...") kullanılır. Örneğin:

```
$ touch 'New File'
```

Tırnak işareti olduğu için komut tek bir dosya olan "New File" için geçerlidir. Tırnak işareti olmasaydı "New" ve "File" diye iki ayrı dosya olarak algılanırdı.

Burada özel karakterlerin hepsini açıklayamayız. Çoğu bu kılavuzun farklı yerlerinde karşımıza çıkacaktır ya da Bash belgelerini kontrol edin.

Bu bölümdeki Komutlar

bash "Bourne-Again-Shell" - etkileşimli komut yorumlayıcısı

date Tarihi ve zamanı görüntüler

echo Boşlukla ayrılan tüm parametrelerini standart çıktıya yazar

help bash komutları için yardımı görüntüler

type Komutun türünü gösterir (dahili, harici, takma ad)

Özet

- Kabuk kullanıcı komutlarını okur ve onları çalıştırır.
- Çoğu kabukta programlama dilinin özellikleri vardır. Ayrıca henüz işlenmemiş komut dizileri içeren kabuk betiklerini destekler.
- Komutlar bağımsız değişken ve seçeneklere sahip olabilir. Seçenekler komutun *nasıl* çalıştığını belirlerken bağımsız değişkenler *ne* üzerine çalıştığını belirler.
- Kabuklar dahili, kabuğun içerisinde tanımlı, komutları ve harici, ayrı süreçler olarak başlatılan çalıştırılabilir dosyalara kaşılık gelen, komutları ayırt edebilir.

Bölüm 5

Yardım Almak

Amaçlar

- Kılavuz ve bilgi sayfalarıyla çalışabilmek
- Nasıl Yapılır kısmını anlamak ve onları bulabilmek
- Diğer en önemli bilgi kaynaklarına aşina olmak

Önceden Bilinmesi Gerekenler

- Linux'a Genel Bakış
- Temel Linux komut satırı kullanımı (örneğin daha önceki bölümlerde bahsedilenler)

5.1 Kendi kendine yardım

Linux güçlü ve karışık bir sistemdir ve kural olarak da güçlü ve karışık sistemler karmaşıktır. Belgeler, bu karmaşıklığı yönetebilmek için önemli bir araçtır. Linux'ün birçok kısmı (ne yazık ki hepsi değil) geniş olarak belgelenmiştir. Bu bölüm bu belgelere nasıl ulaşılacağının bazı yötemlerini açıklar.

Linux'ta "Yardım" birçok durumda "kendi kendine yardım" anlamına gelir. Özgür Yazılımın kültürü toplulukta boş zamanlarını geçiren diğer insanların zamanını ve iyi niyetini kılavuzun ilk birkaç paragrafında zaten açık olarak anlatılmış olan şeyleri sormamanıza dikkat çeker. Linux kullanıcısı olarak en azından mevcut belgelendirmelerin genel bilgisini edinmede ve acil durumlarda yardım almak için yapılacakları bulmada iyisinizdir. Eğer size düşeni yaparsanız göreceksiniz ki genellikle insanlar çikmazdan kurtulmanız

için yardım edeceklerdir. Ama baskalarının onların yerine işlerini yapmasını bekleyen tembel bireyler hoşgörüyle karşılanmazlar.

İyice araştırılmamış sorularınıza ve problemlerinize haftanın yedi günü birinin cevap vermesini istiyorsanız çok sayıda bulunan "ticari" destek tekliflerinden faydalanmanız gerekir. Bunlar bütün ortak dağıtımlar için mevcut olup bu destek ya dağıtım satıcısı tarafından ya da yan şirketlerle verilmektedir. Farklı servis satıcılarını karşılaştırın, hangisinin fiyatı ve servis anlaşması size uyuyorsa seçin.

5.2 *help* Komutu ve –*help* Seçeneği

bash içerisinde dahili komutlar hakkındaki daha detaylı bilgi *help* komutuna bağımsız değişken olarak komutun adı verilerek öğrenilebilir:

```
$ help exit
exit: exit [n]
    Exit the shell with a status of N.
    If N is omitted, the exit status
    is that of the last command executed.
$ _
```

Daha detaylı açıklama için kabuğun rehber sayfasında ve bilgi belgelerinde mevcuttur. Bu bilgi kaynakları daha sonra bu bölümde ele alınacaktır.

Bunun yerine harici komutların (programların) birçoğu *–help* seçeneğini destekler. Çoğu komut kısaca kendileriyle kullanılan parametrelerini ve sözdizminlerini sıralar.

Her komut –help seçeneğini desteklemeyebilir; sık sık çağrılan seçenekler -h veya -?, ya da yanlış bir seçenek veya geçersiz komut satırı belirtirseniz yardım ekrana gelecektir. Ne yazık ki genel kuralı yoktur.

5.3 Çevrim İçi Kılavuz

5.3.1 Genel Bakış

Nerdeyse her komut satırı programı ayar dosyaları ve sistem çağrıları vs. yanında "kılavuz sayfası" (ya da "man sayfası") ile gelir. Bu metinler genelde yazılımla beraber kurulur ve incelemek için "*man <isim>*" komutu kullanılır.

Buradaki *<isim>* açıklanmasını istediğiniz komutun yada dosyanın adıdır. Örneğin "*man bash*", anılan iç kabuk komutlarının listesini üretir.

Ancak, kılavuz sayfalarinin bazi dezavantajlari vardir: Bu kılavuzların çoğu sadece İngilizcedir; farklı dillere çevirilenleri vardır ama tam değil. Üstelik açiklamalar çoğunlukla karışıktır. Her kelime çok önemlidir, bu durum da yeni başlayanlar için belgeleri erişilebilirlikten uzaklaştırır. Ek olarak, özellikle

uzun belgelerin yapısı anlaşılmaz olabilir. Öyle olsa bile, bu belgelerin değeri göz ardı edilemez. Kağıtlarla kullanıcıyı canından bezdirmek yerine sistemle beraber çevrim içi kılavuz mevcuttur.

Birçok Linux dağıtımlari komut satırından çağrılabilen her komut için bir kılavuz sayfasının olması gerektiği felsefesini takip eder. Bu durum KDE ve GNOME görsel masaüstü ortamına ait programlar için aynı ölçüde geçerli değildir. Bunların pek çoğunun hiç bir kılavuz sayfasının bulunmaması yanında bazıları da görsel masaüstü çevresinde bile berbat bir şekilde belgelendirilmiştir. Bu programların çoğunun gönüllüler tarafından katkıda bulunulduğu gerçegi sadece zayıf bir bahanedir.

5.3.2 Yapı

Yardım sayfalarının yapısı kabaca Tablo **??** taslağını takip eder. Yine de her kılavuz sayfası orada belirtilen her bölümü içermeyebilir. Özellikle de, EXAMPLES kısmı sık sık kısa kısa verilmiştir.

BUGS başlığı genellikle yanlış anlaşılır: Düzeltilen uygulamanın içindeki *hataları* okuyun; elbette burada belgelendirilen genellikle komutun aldığı yaklaşımdaki kısıtlamalardır ve siz bu kısıtlamaların makul bir çabayla kaldırılamadığını bir kullanıcı olarak bilmelisiniz. Örneğin, *grep* komutunun belgeleri düzenli ifadelerin çeşitli yapılarının bulunması grep sürecinin çok bellek kullanmasına yol açabildiğini belirtir. Bu *grep* komutunun uyguladığı arama yönteminin sonucudur ve önemsiz, kolayca giderilebilen hata değildir.

Yardım sayfaları *groff* adında bir program tarafından metni görüntülemek veya yazdırmak için özel bir girdi biçimi olarak yazılır. Kılavuz sayfaları /usr/share/man dizininin *man*n alt dizini içinde saklanır. n Tablo **??**'de verilen bölüm numaraları içindir.

Diğer dizinlerdeki yardım sayfalarını, *man* komutu tarafından sırasıyla taranacak dizinleri içeren, MANPATH çevresel değişkenini ayarlayarak birleştirebilirsiniz. *manpath* komutu MANPATH değişkeninin nasıl ayarlanabileceği hakkında ipuçları barındırır.

5.3.3 Bölümler

Her kılavuz sayfası "kılavuzu" kapsayan "bölüme" aittir (Tablo **??**). 1., 5. ve 8. bölümler en önemlileridir. Aramayı daraltmak için *man* komutu satırına bölüm numarası verebilirsiniz. Örneğin, "*man 1 crontab*" *crontab* komutu için kılavuz sayfasını görüntülerken, "*man 5 crontab*"-*crontab* dosyalarının biçimlerini açıklar. Kılavuz sayfalarını işaret ederken, bölüm numarasını parantez içinde belirtmek gelenekseldir; biz, *crontab*(1) *crontab* komut kılavuzu, ve *crontab*(5) dosya biçimi açıklaması aradaki farkı ayırt ederiz.

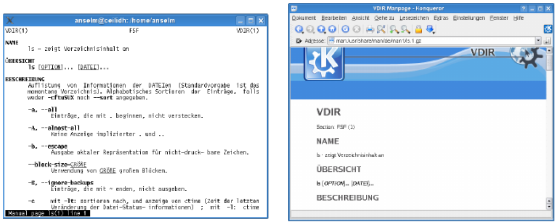

Figure 5.1: Metin terminali (sol) ve Konqueror (sağ) içerisinde bir kılavuz sayfası

-a seçeneğiyle *man* verilen isme göre bulunan kılavuzu gösterir; seçenek vermezsek ilk bulunan sayfayı (genellikle bölüm 1) gösterir.

5.3.4 Kılavuz Sayfalarını Görüntülemek

Kılavuz sayfalarını metin terminalde görüntülemek daha üzerinde duracağımız *less* ile gerçekleştirilir. Bu aşamada kılavuz sayfasında yukarı ok ↑ ve aşağı ok ↓ tuşları ile gezinebileceğinizi bilmek önemlidir. Metnin içindeki bir anahtar kelimeyi / tuşuna basıp ardından kelimeyi girip Enter tuşuna basarak arayabilirsiniz. Her geri dönüş tuşuna bastığınızda bir sonraki bulunan kayda sıçrar (eğer varsa). Kabuğa *q* tusuna basarak geri dönebilirsiniz.

KDE web tarayıcısı, Konqueror, kullanarak güzel biçimlendirilmiş uygun kılavuz sayfaları elde etmek mümkündür. Basitçe "*man:/<isim>*" (hatta "*#<isim>*") URL adresini tarayıcının adres satırına girin. Bu yöntem aynı zamanda KDE komut satırı için de geçerlidir.

Amaçsızca sayısız kılavuz sayfasını aramadan önce, konu hakkında *apropos* yardımıyla genel bir bilgi edinmek daha mantıklıdır. Bu komut basitçe şöyle çalıştırılır "man -k"; her ikisi de komut satırında verilen bir anahtar kelimenin "NAME" bölümlerine ilişkin tüm kılavuz sayfalarını arar. Bunun sonucunun çıktısı tüm man sayfalarını, adı veya açıklama kısmında anahtar kelimeyi de içerecek şekilde bir liste görüntüler.

Bu konuyla alakalı olan bir başka komut *whatis* komutudur.Bu komut da tüm man sayfalarını arar ama yukarıda belirttiğimiz komuttan farkı *whatis* aramayı anahtar kelimeye göre değil de komutun (dosya,...) *adına* göre yapar. Bu istenilen komut, sistem çağrıları vs. hakkında kısaca bir açıklama görüntüler. Özellikle söz konusu man sayfasının(lar) "NAME" bölümünün ikinci kısmını verir. *whatis* ile "man -f" eşdeğerdir.

Alıştırmalar

1. *ls* komutu için kılavuz sayfasını görüntüleyin. Metin tabanlı *man* komutunu kullanın ve eğer mümkünse - Konqueror tarayıcısını da kullanın.

2. Sisteminizde hangi kılavuz sayfaları (en azından "NAME" bölümlerine göre) süreçlerle iş yapar.

3. (Ileri düzey) Kuramsal bir komutun kılavuz sayfasını yazmak için metin editörü kullanın. Önceden *man*(7) kılavuz sayfasını okuyun. Kılavuz sayfasının görünürlüğünü hem ekranda ("*groff -Tascii -man <dosya> — less*" komutunu kullanarak) hem de yazılı çıktı olarak ("*groff -Tps -man <dosya> — gv -*" gibi birşey kullanın) kontrol edin.

5.4 Bilgi Sayfaları

Bazı komutlar için - genellikle karmaşık olanlar için sıradan kılavuz sayfaları yerine (ya da ilave olarak) "bilgi sayfaları" mevcuttur. Bunlar genellikle daha geniş olup World Wide Web'e benzer şekilde hiper metin prensibine göre kurulmuştur.

Bilgi sayfaları fikri GNU projesi ile birlikte ortaya çıkmıştır; o yüzden onlar en sık FSF (Özgür Yazılım Vakfı) ile yayımlanan veya GNU projesine ait yazılımlarla gelir. Aslında "GNU sistem" için sadece bilgi belgeleri olması gerekiyordu; ancak GNU, FSF himayesinde geliştirilmeyen bir sürü yazılımları da kendi içine alır, ve GNU araçları çizgileri daha kesin olan sistemlerde kullanıldığından FSF bazı durumlarda taviz vermeye başladı.

Kılavuz sayfalarının dengi olan bilgi sayfaları "*info <komut>*" komutu (bilgi programını içeren paket açıkça kurulmuş olması gerekebilir) kullanılarak görüntülenir. Ayrıca, bilgi sayfaları *emacs* editöründe veya KDE web tarayıcısı Konqueror'da URL'ler yardımıyla "*info:/<komut>*" görüntülenebilir.

Bilgi sayfalarının bir avantajı, kılavuz sayfaları gibi kaynak formatında yazılmış olmalarıdır. Bilgi sayfaları PDF ve PostScript formatında yazdırılabilir veya ekranda işlenebilir. *groff* yerine, TEX dizgi programı çıktı işlemi için hazırlanabilir.

Alıştırmalar

1. *ls* programı için bilgi sayfasına bakın. Metin tabanlı bilgi tarayıcısı ve, varsa, Konqueror tarayıcısını deneyin.

2. Bilgi sayfaları günümüzde HTML dosyalarının World Wide Web'de olduğu gibi hiper metnin ilkel formunu kullanır. Bilgi sayfaları neden HTML ile yazılmamıştır?

5.5 NASIL Belgeleri

Kılavuz ile bilgi sayfaları arasındaki ortak problem şudur: Kullanıcılar kullanacakları programın adını bilmek zorundadırlar. Hatta *apropos* ile arama

yapmak şans oyunu gibi birşeydir. Ayrıca, her problem tek bir komut kullanılarak çözülemez. Bu nedenle bunlar "komut odaklı" belgeler yerine genellikle "problem odaklı" olarak adlandırılır. Nasıl Yapılır kısmı bunlara çözüm üretmek için tasarlandı.

Nasıl Belgeleri kendilerini tek bir komutla kısıtlamayan geniş kapsamlı belgelerdir, ama sorunların çözümü için tam yaklaşımları açıklamaya çalışırlar. Örneğin, DSL yoluyla Linux sisteminin internete nasıl bağlanacağını detaylı olarak açıklayan "DSL Nasıl" belgesi vardır, ya da Linux için astronomi yazılımını tartışan "Astronomi NASIL" vardır. Nasıl Yapılır kısımlarının birçoğu İnglizce aslını genellikle geriden takip etse de bunlar başka dillerde de mevcuttur.

Çoğu Linux dağıtımı Nasıl Belgelerinin (veya büyük bir alt kümesinin) yerel olarak kurulu olmasını sağlar. Bunlar dağıtıma özel dizinlerin altında bulunurlar. /usr/share/doc/howto SUSE dağıtımları için, /usr/share/HOWTO Debian GNU/Linux içindir. Tipik olarak düz metin veya HTML dosyaları içerir. Nasıl Yapılır'ların geçerli tüm sürümleri ve diğer PostScript veya PDF biçimindekilerinin hepsi internet üzerinde "Linux Belgelendirme Projesi" (`http://www.tldp.org`) altında bulunabilir. Bu adres ayrıca diğer Linux belgelerini de sunar.

5.6 Ek Bilgi Kaynakları

Neredeyse her kurulu olan yazılım için ilave belgeleri veya örnek dosyaları /usr/share/doc ya da /usr/share/doc/packages (kullandığınız dağıtıma bağlı olarak değişir) altında bulunur. Çoğu GUI uygulamaları (KDE veya GNOME paketlerindeki gibi) "yardım" menüsü sunar. Üstelik birçok dağıtım uzmanlaştırılmış "yardım merkezleri" sunar. Sistem üzerindeki çoğu belgelere uygun erişimi sağlar.

Yerel sistemden bağımsız olarak WWW ve USENET arşivlerinin de arasında bulunduğu, pek çok belge Internet ortamında bulunmaktadır.

Linux için daha ilgi çekici sitelerden bazıları:

`http://www.tldp.org/` "Linux Belgelendirme Projesi", kılavuz sayfaları ve Nasıl Yapılır'dan sorumlu (diğer şeylerin yanı sıra).

`http://www.linux.org/` Linux meraklıları için genel portal.

`http://www.linuxwiki.de/` Linux ile ilgili her şey için serbest biçimli metin bilgi veritabanı (Almanca)

`http://lwn.net/` Haftalık Linux Haberleri - her türlü Linux haberleri için belki de en iyi web sitesi. Yeni gelişmeler, ürünler, güvenlik açıkları, basındaki Linux savunuculuğu vs. ayrıca her persembe günü geçmiş haftaların araştırmalarının yer aldığı çevrimiçi dergi de bulunur. Günlük haberlere ücretsiz olarak ulaşılabilir iken haftalık yayınlamalara belli bir ücret ödenmesi gerekiyor (aylık 5$'dan başlayan fiyatlarla). İlk çıktığı haftadan sonra o yayınları ücretsiz olarak erişilebilir hale getiriyorlar.

`http://freecode.com/` Bu site yeni (genel olarak serbest) çıkan yazılım paketlerini tanıtır. Buna ek olarak ilginç projeler veya yazılım paketleri için sorguları sağlayan bir veritabanı var

`http://www.linux-knowledge-portal.de/` LWN ve Freshmeat dahil diğer Linux sitelerinden haber başlıklarını toplar.

Eğer Internette veya Usenet arşivlerinde bulmadıysanız aradığınızı, sorunuza cevabı posta listelerinde ya da Usenet gruplarında soru sormak her zaman mümkündür. Bu forumlardaki çoğu kullanıcıların daha önce cevaplanmış ya da belgelendirmede veya "SSS" (Sıkça Sorulan Sorular) kısmında olan birşeyi sormanıza kötü tepki verebileceklerini unutmayın. Probleminizin detaylı açıklamasını hazırlayın, kayıt dosyalarından ilgili ayrıntıları verin çünkü karmaşık problemleri sizde olduğu gibi uzak mesafeden çözmek zordur (karmaşık olmayan problemleri kendinizin çözebiliyor olmanız lazım).

Haber arşivlerini `http://groups.google.com/` (eskiden DejaNews) adresinde bulabilirsiniz.

Linux için ilgi çekici *haber grupları* İngilizce için *comp.os.linux.** veya Almanca için *de.comp.os.linux.** hiyerarşilerinde bulunabilir. Birçok Unix grupları Linux konuları için uygundur; kabukla ilgili soruların kabuk programlama için ayrılan grupta sorulması gerekir, Linux grubunda değil, çünkü kabuklar genellikle Linux'a özgü değildir.

Linux tabanlı posta listelerini örneğin, *majordomo@vger.kernel.org* adresinde bulabilirsiniz. LIST denilen listeye katılmak için önce "subscribe LIST" adresine e-posta atmanız gerekir. Sistemde mevcut yorumlanmış tüm posta listeleri `http://vger.kernel.org/vger-lists.html` adresinde bulabilirsiniz.

Görünüşte anlaşılmayan problemleri çözmenin yolu hata mesajının Google'da aratmaktır (ya da güvendiğiniz baska bir arama motoru). Eğer faydalı bir sonuç alamazsanız, arama yaptığınız sorguda sadece size özel duruma bağlı kısımları kaldırın (mesela alan adları gibi). Google'da aramanın avantajı sadece ortak web sayfalarını indekslemek değil bunun yanında posta liste arsivlerini de indeksler. O yüzden sizin gibi sorunu yaşayan bir başka birilerinin bulunması da olası bir durum.

Açık kaynak kodlu yazılımların büyük miktarda belgelendirmelerinin olması tek büyük faydaları değildir ayrıca çoğu belgelendirmenin de en az

yazılımının kendisi gibi kısıtlı olmasıdır. Bu, yazılım geliştiricileri ve belge yazarları arasındaki işbirliğini kolaylaştırır ve belgelerin diğer dillere çevrilmesi daha kolaydır. Aslında, programcı olmayanlar için özgür yazılım projelerine destek verebilmeleri için bol fırsat bulunmaktadır, örn. iyi belgelemeler yazmak. Özgür yazılım faaliyet alanında programcılara verilen saygıyı belge yazarlarına da vermeye çalışılmalı. Bu durum kaymasının başlamış olması henüz tamamlandığı anlamına gelmemektedir.

Bu Bölümdeki Komutlar

apropos NAME bölümünde verilen anahtar kelimeyi içeren tüm kılavuz sayfalarını görüntüler

groff Gelişmiş dizgi programı

help *bash* komutları çevrim için yardımı görüntüler

info Karakter tabanlı terminalde GNU Bilgi sayfalarını görüntüler

less Metinleri sayfa sayfa (kılavuz sayfaları gibi) görüntüler

man Sistem kılavuz sayfalarını görüntüler

manpath Sistem kılavuz sayfalarının aranacağı yolu belirler

whatis Açıklamasında verilen belirli bir anahtar kelime ile kılavuz sayfalarını bulur

Özet

- "*help <komut>*" dahili bash komutlarını açıklar. Birçok harici komut *–help* seçeneğini destekler.
- Çoğu program kılavuz sayfalarıyla gelir. Bunlar *man* komutuyla incelenebilir. *apropos* verilen anahtar kelimelere göre tüm kılavuz sayfalarını arar, *whatis* kılavuz sayfa isimlerine bakar.
- Bazı programlar için bilgi sayfaları kılavuz sayfalarına bir alternatiftir.
- Nasıl Yapılır belgeleri problem tabanlı bir belgeleme olustururlar.
- World Wide Web ve USENET dünyasında Linux ile ilgili çok sayıda ilginç kaynaklar var.

Table 5.1: Kılavuz sayfasının bölümleri

Bölüm	İçerik
NAME	Komutun adı ve kısa açıklama
SYNOPSIS	Komutun sözdizimi açıklaması
DESCRIPTION	Komutun etkisi ile ilgili açıklama
OPTIONS	Kullanılabilir seçenekler
ARGUMENTS	Kullanılabilir bağımsız değişkenler
FILES	Yardımcı dosyalar
EXAMPLES	Örnek komut satırları
SEE ALSO	İlgili konulara çapraz referanslar
DIAGNOSTICS	Hata ve uyarı mesajları
COPYRIGHT	Komutun yazarları
BUGS	Bilinen komut sınırlamaları

Table 5.2: Kılavuz sayfası konuları

Bölüm	İçerik
1	Kullanıcı komutları
2	Sistem çağrıları
3	C dili kütüphane işlevleri
4	Aygıt dosyaları ve sürücüleri
5	Ayar dosyaları ve dosya formatları
6	Oyunlar
7	Çeşitli (örneğin groff makroları, ASCII tabloları, ...)
8	Yönetici komutları
9	Çekirdek işlevleri
n	"Yeni" komutlar

Bölüm 6

Dosyalar: Bakım ve Besleme

Amaçlar

- Linux kurallarıyla birlikte dosya ve dizin isimleriyle uğraşmaya alışmak
- Dosya ve dizinlerle çalışmada önemli komutları bilmek
- Kabuk dosya adı arama şablonlarını kullanabilmek

Önceden Bilinmesi Gerekenler

- Kabuk kullanımı
- Metin editörü kullanımı

6.1 Dosya ve Yol İsimleri

6.1.1 Dosya İsimleri

Linux gibi işletim sisteminin en önemli hizmetlerinden birisi de, veriyi sabit disk veya USB sürücüler gibi kalıcı hafızaya depolamak ve sonrasında onları kullanmaktır. Bu işi insan açısından katlanılabilir kılmak için, benzer veriler cihaz üzerinde "dosyalar" altında toplanır.

Eğer bu bile sizin için önemsiz gözüküyorsa, bu anlatılanların size hiçbir getirisi yoktur. Eskiden bazı işletim sistemleri, bir verinin bilgisini almak için disk üzerindeki iz sayıları gibi ayrıntıları belirtmeyi zorunlu kılıyordu.

Bu yüzden, size dosyalarla nasıl uğraşılacağını anlatmaktan önce, Linux'un dosyaları nasıl isimlendirdiğini açıklamalıyız.

Linux dosya isimlerinde, aslında bilgisayarınızın gösterebileceği her karakteri (hatta daha fazlasını) kullanmanıza olanak tanır. Ancak, karakterlerden bazılarının özel anlam içerdiğinden beri, dosya isimlerinde bu karakterleri kullanmanızı tavsiye ederiz. Yalnızca iki karakterin kullanılamaz: bölü ("/") işareti ve 0 byte (ASCII değeri 0 olan karakter). Boşluk, inceltme işareti ya da dolar işaretleri gibi karakterler serbestçe kullanılabilir, fakat kabuk tarafından yanlış yorumlanmakyan kaçınmak için komut satırında "\" veya tırnak işaretleri kullanılarak bu karakterlerden kaçılır.

Acemilerin kolayca tuzağa düşeceği olay ise, Linux'un dosya isimlerinde büyük/küçük harfleri ayırt edebilmesidir. Büyük ve küçük harfli dosyaların ayrı gösterildiği fakat aynı davranıldığı Windows işletim sisteminin tersine Linux *x-files* ve *X-Files* dosya isimlerini iki farklı dosya olarak ele alır.

Limitlerin "dosya sistemine" bağlı olduğundan beri, Linux altında byte'lar halinde düzenlenen dosya adları "çok uzun", belli bir üst limit yoktur, olabilir(Linux'ta bu düzenlemeyle ilgili birçok yöntem vardır). Normal sınır 255 karakterdir, fakat 3 satırdan daha fazla uzunlukta dosya isimleri standart metin terminalinde size bir karmaşıklık oluşturmamalıdır.

DOS ve Windows bilgisayarlardan başka bir fark da Linux, dosyanın tipini belirlemek için sonekleri kullanmaz. Bu yüzden, dosya ismindeki nokta tamamen sıradan bir karakterdir. Bir metin dosyasını *mumble.txt* olarak kaydetmede serbestsiniz, fakat prensip olarak sadece *mumble* kelimesi de kabul edilebilir. Bu kullanım tabi ki sizin tamamen uzantı ile adlandırmanızı engellememelidir, sonuçta bu kullanım dosya içeriğini tanımlamanızda kolaylık sağlar.

Bazı programlar girdi dosyalarında özel uzantı kullanmayı zorunlu kılar. Örneğin C derleyicisi olan *gcc*, "*.c*" ile biten C kaynak kodu, "*.s*" ile biten makine dili kaynak kodu ve "*.o*" ile biten ön derlenmiş nesne dosyaları isimlerini dikkate alır.

Dosya isimlendirmede özgürce inceltme işareti ve diğer özel karakterleri kullanabilirsiniz. Yine de, eğer dosyalar diğer sistemlerde kullanılacaksa özel karakterleri kullanımından kaçmak en iyisidir, çünkü diğerlerinde dosyaların aynı isimle gösterileceğinin bir garantisi yoktur.

Özel karakterlerin ne olduğu sizin yerel ayarlarınıza bağlıdır, çünkü ASCII karakter setini (Çoğunlukla içinde İngilizce dili, rakamlar ve yaygın özel karakterlerin bulunduğu 128 karakter) aşan karakterlerin temsili için genel bir standart yoktur. Çokça kullanılan kodlamalara örnek olarak ISO 8859-1 ve ISO 8859-15 (sırasıyla ISO-Latin-1 ve ISO-Latin-9 olarak bilinir) bununla birlikte gelişigüzel ve çok doğru olmayarak adına "Unicode" denilen genellikle "UTF-8" karakter seti ile kodlanan ISO 10646. Y kodlaması sistemde yürürlükte iken dizine baktığınız zaman X kodlamasıyla oluşturduğunuz dosyalar tamamen farklı gözükebilir. Bu konuların üzerinde çok durursanız, size tüm konu hiçbir şey gibi gelir.

Bazen kendinizi yanlış karakter setiyle kodlanmış bir yığın dosya ile uğraşıyor olarak bulabilirsiniz. Çeşitli karakter setleri arasında dönüşüm yapabilen *convmv* programı bu konuda size yardım edebilir. (Çoğu dağıtımın standart kurulumunun bir parçası olmadığından, muhtemelen kendiniz yüklemeniz gerekecektir.) Ancak, bunu tüm bölümü bitirdiğimiz zaman yapmanız daha uygun olur, çünkü daha normal *mv*'den bahsetmedik bile...

Aşağıdaki tüm karakterleri dosya isimlendirmelerinde özgürce kullanabilirsiniz:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789+-._
```

Ancak aşağıdaki ipuçlarına dikkat etmelisiniz:

- Linux ve daha eski Unix sistemleri arasında dosya taşımalarını mümkün kılmak için dosya isimleri en fazla 14 karakter olmalıdır. (Bu kurala uymak gerçekten önemlidir.)

- Dosya isimleri her zaman harf veya rakamlardan biriyle başlamalıdır; diğer 4 karakterleri sadece dosya içerisinde sorunsuzca kullanabilirsiniz.

Bu kuralları anlamanın en kolayı birkaç örneği incelemektir. Uygun dosya isimlerine örnek olarak

```
X-files
foo.txt.bak
50.something
7_of_9
```

verilebilir. Buna karşılık aşağıdaki kullanımlar problem oluşturabilir (muhtemelen):

```
-10F  ‘‘- ’’ ile başlıyor, özel karakter var
.profile  Gizli dosya olacaktır
3/4-metre  Kural dışı karakter içerir
Smörrebröd  Inceltme işareti içerir
```

Diğer bir tuhaf özellik de, dosya isimlerinin nokta (".") ile başlaması bazı yerlerde dosyanın gözden kaçmasına neden olacak, örneğin dizin içerisindeki dosyaların listelendiği zaman, bu tür dosya isimleri "gizli" gibi düşünülür. Bu özellik genellikle programların içerik ayarlarında ve dizin listelemelerde çok daha önemli dosyaların kullanıcının kafasını karıştırmamak amacıyla kullanılır.

DOS ve Windows uzmanları için: Bu sistemler dosyanın isminden bağımsız olarak "dosya özniteliklerinin" "gizli" olması ile izin verir. Linux ve Unix böylesi bir işi desteklemez.

6.1.2 Dizinler

Potansiyel olarak birçok kullanıcı aynı Linux sisteminde alıştığından, tüm dosya isimlerinin sadece bir tane olması problem teşkil edebilirdi. Sue kullanıcısı halihazırda aynı isimde bir dosyaya sahip olduğu için Joe kullanıcısının *letter.txt* isimli bir dosya oluşturmasını oldukça güç hale sokar. Ayrıca Joe'nun, Sue'nun tüm dosyalarını okuyamamasını garanti eden uygun bir yol bulunması gerekir.

Bu nedenle, Linux dosyaları gruplamak için aşamalı "dizin" fikrine destek verir. Aynı dizindeki dosyalar dışında, dosya isimlerinin tüm sistemde eşsiz olması gerekmez. Özellikle bu, sistemin Joe ve Sue için farklı dizinler atayabilmesini, ve her birinin kendi dosyaları hakkında diğerlerininkilerin isimlerini dert etmeden çağırabilmeleri anlamına gelir.

Ek olarak, Joe'ya Sue'nun *dizinine* (veya tam tersi) erişmesini yasaklayabiliriz ve dizin içerisindeki kişisel dosyalar hakkında endişelenmemize gerek kalmaz.

Linux'ta dizinler basit dosyalar olsa da, "düz" dosyalara erişmek için kullandığınız yöntemler ile onlara erişemeyebilirsiniz. Ancak, daha önce bahsettiğimiz dosya isimleri kurallarını(önceki bölüme bakınız) dizin isimlendirmelerinde de göz önünde bulundurmanız gerekir. Sadece bölü ("/") karakterinin, dizin isimlerinden dosya isimlerini ve dizinleri birbirlerinden ayrımak için kullanıldığını bilmeniz yeterlidir. Örnek olarak: *joe/letter.txt*, *joe* dizinindeki *letter.txt* dosyasıdır.

Dizinler ağaç-benzeri (yaratıcı bir şekilde "dizin ağacı" denen) yapı ile tasvir edilen diğer dizinleri içerebilir (bu "aşamalı" teriminden daha önce de bahsetmiştik). Bir Linux sistemi, ağacın kökleri gibi bir yapıda olan "kök dizini" denilen özel bir dizin içerir. "/"(bölü) ile ifade edilir.

İsminin aksine, kök dizininin, sistem yöneticisi anlamına gelen *root* ile alakası yoktur. Sadece isimleri benzerdir.

Burada bölünün çift görevi vardır, hem kök dizininin adıdır, hem de dizin isimleri arasında ayraç görevi üstlenir. Bu konuya birazdan geri döneceğiz.

Yaygın Linux dağıtımlarındaki temel yükleme, genellikle tek bir dosya hiyerarşisinde bulunan ve bunun çokça belirli düzene göre yapılandırılmış olduğu onbinlerce dosyayı içerir. Bu dosya hiyerarşisine Bölüm **??**'da daha çok yer vereceğiz.

6.1.3 Mutlak ve Göreceli Yol İsimleri

Linux sistemindeki her dosya, kök dizini tarafından başlayan, bir dosyaya (dosyanın kendisine) ulaşıncaya kadar devam eden tüm dizinlerin adının geçtiği ve sonunda dosyanın kendisinin bulunduğu dosya isimleriyle tasvir edilir. Örneğin, */home/joe/letter.txt* ismi, *joe* dizini altında bulunan, *home* dizini içerisinde konumlanan, daha sonra da evebeyn olan kök dizininin yer aldığı

letter.txt dosyasına karşılık gelir. Kök dizini ile başlayan isimlere "mutlak yol ismi" adı verilir. İsimler dizin ağacı içerisinde dizin ve dosya isimleri barındıran "yol" vasıtasıyla tanımlandığından "yol isimleri" lafını kullanırız.

Linux sisteminde her sürecin bir "mevcut dizini" (genellikle "çalışma dizini" diye adlandırılır) vardır. Dosya isimleri bu dizinin içerisinde aranır; *letter.txt* "mevcut dizin içerisindeki letter.txt" için, *sue/letter.txt* "sue dizini içerisindeki, letter.txt" için bir kısaltmadır. Mevcut dizinden başlayan bu tarz isimlere "göreceli yol isimleri" adı verilir.

Göreceli yol isimlerinin mutlak olduğunu söylemek önemsizdir: Mutlak dosya yolu bir "/" ile başlar; diğer hepsi görecelidir.

Mevcut dizin evebeyn ve çocuk süreçler arasındaki "miras" olarak aktarılır. Yani kabuktan yeni bir kabuk başlattığınızda(veya başka bir program), bu yeni kabuk, sizin başlatmak için kullandığınız kabuk dizini ile aynı mevcut yolu kullanır. Yeni kabuğunuzda diğer dizine *cd* komutunu kullanarak geçebilirsiniz, fakat eski kabuğun mevcut dizini değişmez. Eğer yeni kabuktan ayrılırsanız, eski kabuğun mevcut dizinine (değişmemiş dizine) geri dönersiniz.

Göreli yol isimlerinde(hatta mutlak olanlarda bile) iki kullanışlı kestirme vardır: ".." ismi her zaman dizin ağacında bir bilinmeyen olan mevcut dizinin üst dizinini işaret eder. Örneğin, */home/joe*, */home* durumudur. Bu sıklıkla mutlak yol ismine başvurmak zorunda kalmadan, mevcut dizinden, dizin ağacı gibi gösterilen "yan dal"'a ayrılmak için kullanışlı bir yoldur. Farzedelim ki */home/joe* dizininin *letters* ve *novels* isminde altdizinleri olsun. *letters* sizin mevcut dizininiz gibi olsun, *novels* dizininin içerisinde *ivanhoe.txt* dosyasına kaba bir şekilde mutlak yol ismi olan */home/joe/novels/ivanhoe.txt*'yi kullanmaksızın *../novels/ivanhoe.txt* şeklinde göreceli yol ismi belirterek referans belirtebilirsiniz.

Ikinci kestirme ise size açıkça bir anlam ifade etmiyor gibi gelebilir: "." ismi dizin içerisinde bulunur ve her zaman dizinin kendisini ifade eder. Neden zaten içinde bulunduğumuz dizine işaret eden böyle bir metoda ihtiyacımızın olduğunun doğrudan bir cevabı yoktur, fakat tabi ki bu durumların bize kattığı kullanışlı beceriler vardır. Örneğin, belki biliyorsunuzdur (ya da Bölüm **??**'a bakınız) kabuğun harici komutlar için program dosyalarını ararken kullandığı çevresel dizin değişkeni olan *PATH* vardır. Diyelim ki bir yazılım geliştiricisisiniz, bir program çalıştırmak istiyorsunuz, *prog* olarak isimlendirdiğimiz, (a) mevcut dizininiz içerisindeki bulunan ve (b) *PATH* içerisinde yer almayan (Güvenlik nedenleri için her zaman iyi bir fikirdir) dizininiz içerisindeki *prog*'u çalıştıralım, halâ kabuğunuzdayken

```
$ ./prog
```

diyerek dosyanızı, mutlak yol ismini vermek zorunda kalmadan program olarak başlatabilirsiniz.

Linux kullanıcısı olarak sisteme girdiğinizden hemen sonra "ev dizinine" ulaşırsınız. Sistem yöneticisi sizin hesabınızı oluşturduğunda bu dizinin ismine karar verir, fakat genellikle kullanıcı isminizle aynı olur ve /home dizini altında yer alır, *joe* kullanıcısı için atanan */home/joe* gibi.

6.2 Dizin Komutları

6.2.1 Mevcut Dizin: cd & Co.

Mevcut dizini değiştirmek için *cd* kabuk komutunu kullanabilirsiniz: Basitçe parametre olarak arzu edilen dizin ismi verilir:

```
$ cd letters         letters dizinine değiştirir
$ cd ..     Bir üstteki dizine değiştirir
```

Eğer parametresiz kullanacak olursanız ev dizininize erişirsiniz:

```
$ cd
$ pwd
/home/joe
```

Mevcut dizinin mutlak yolunun çıktısını almak için *pwd* ("print working directory") komutunu kullanabilirsiniz.

Ayrıca mevcut dizininizi büyük olasılıkla kendi istemcinizin bir kısmı olarak görebilirsiniz. Sistem ayarlarınıza bağlı olarak

```
joe@red:~/letters> _
```

gibi bir satırla karşılaşabilirsiniz. *∼/letters /home/joe/letters* ifadesi için kısaltmadır; tilda ("∼") ise mevcut kullanıcı ev dizininizi gösterir.

"*cd -*" komutu *cd* komutu ile son kullanılmış dizine tekrar geri dönmeyi sağlar. Bu kullanışlı komut iki dizin arasında peş peşe geçiş yapmayı sağlar.

Alıştırmalar

1. Kabukta *cd* komutu dahili mi, harici mi komuttur? Neden?

2. Kabuk kılavuz dosyasında *pushd*, *popd*, *dirs* komutlarını okuyunuz. Bu komutların tasvir edildiği gibi çalıştığına ikna olunuz.

6.2.2 Dosya ve Dizinleri listeleme –ls

Dizin ağacı etrafında yolunuzu bulmak için hangi dosya ve dizinlerin mevcut dizin içerisinde olduğunu bilmeniz gerekir. *ls*("list") komutu tam da bu iş için vardır.

Parametresiz kullanımda dosyaların isme göre sıralandığı çok sütunlu çıktı gösterilir. Renkli ekranların günümüzde standart haline gelmesiyle, dosya

isimleri ve türlerinin farklı renklerde görüntülenmesi gelenek haline geldi (Daha dosya tipleri hakkında henüz konuşmadık; bu konuya Bölüm **??**'da değineceğiz).

Neyse ki, günümüzde birçok dağıtım renkli kullanımı bünyesinde barındırıyor. Tablo **??**'de en yaygın gösterimler vardır.

Siyah-Beyaz ekranlarda, ki hala günümüzde var, -F veya -p parametreleri önerilir. Bunlar dosya türleri ile ilişkili dosya isimlerine özel karakter eklerler. Bu karakterlerin bazılarına Tablo **??**'de değinmiştik.

Gizli dosyaları (nokta ile başlayan isimlileri) *-a* ("all") parametresiyle görüntüleyebilirsiniz. Bir başka kullanışlı seçenek ise *-l*'dir(küçük harf olarak "L", "long"(uzun) anlamı taşır, "1" rakamı değildir). Bu sadece dosya isimlerini sıralamakla kalmaz ayrıca her dosya için ek bilgi de içerir.

Bazı Linux dağıtımlarında önceden ayarlanmış bazı yardımcı parametrelerin kombinasyonu olan kısaltmalar vardır; SUSE dağıtımlarında, örneğin sadece basit bir *l* kullanımı "*ls -alF*"'nin kısaltmasıdır. "*ll*" ve "*la*" *ls*'nin farklılaşmış bazı kısaltmalarıdır.

Aşağıdaki örnekte *ls*'nin *-l*'li ve *-l*'siz kullanımı vardır:

```
$ ls
file.txt
file2.dat
$ ls -l
-rw-r--r-- 1 joe users 4711 Oct 4 11:11 file.txt
-rw-r--r-- 1 joe users 333 Oct 2 13:21 file2.dat
```

İlk durumda, bütün gizli olmayan dosyalar listelendi; ikinci durumda ise ek bilgiler eklendi.

Uzun formatın farklı parçalarının anlamları vardır: İlk karakter dosya türünü verir (Bölüm **??**'a bakınız); sade dosyalar "-", dizinler "d" şeklinde vb.(Tablo **??**'deki "karakter türleri").

Sonraki 9 karakter erişim izinlerini gösterir. Sonraki dosyanın adedini, dosyanın sahibini(joe), ve dosyanın grubunu(users) gösterir. Byte olarak dosyanın boyutundan sonra, dosyanın içerdiği son değiştirilme bilgisini tarih ve saat olarak görebilirsiniz. Ve tabi ki dosyanın adı da görüntülenir.

Kullandığınız dile bağlı olarak özellikle tarih ve saat sütunları örmeklerimizden(asgari dil ortamı "C" kullanarak oluşturduğumuz) tamamen farklı görünebilir. Bu genellikle interaktif kullanımda sorun çıkarmaz, fakat eğer kabuk betiğinizin bir bölümünde "ls -l"'nin çıktısını almaya çalışıyorsanız büyük bir sıkıntı ortaya çıkabilir. (Gelişmiş Linux kılavuzunu çalışmaktan önce, kabuk betiklerinde ön tanımlanmış dil çevreselini kullanmanızı öneririz.)

Eğer dizin için ekstra bilgi görmek isterseniz(/tmp dizini gibi), "ls -l /tmp"'nin size gerçekten bir yardımı dokunmaz, çünkü ls /tmp içerisindeki tüm dosyaların verisini listeleyecektir. -d parametresi kullanarak /tmp'nin kendisinin bilgisini elde edebilirsiniz.

ls burada bahsedilenlerden çok daha fazlasına destek verir; önemli olanlarından bir kaçı Tablo **??**'de gösterilmişti.

Linux Essentials ve LPI-101 gibi LPI sınavlarında, kimse sizden ls'nin tüm 57 tane parametresini bilmenizi beklemez. Ancak en önemli yarım düzine kadarını—yaklaşık Tablo **??** içeriğindeki kadarını— ezberlemek isteyebilirsiniz.

Alıştırmalar

1. */boot* dizinini hangi dosyaları içerir? Alt dizinleri var mıdır? Varsa hangileridir?

2. Dosya ismini parametre olarak alan *ls* ile dizin ismini parametre olarak alan *ls* arasındaki fark nedir?

3. Eğer dizin adı programda geçiyorsa, dizindeki dosyaların bilgisini görüntülemekten *ls*'yi sadece dizinleri göstermesi için hangi parametre ile çalıştırırız? (İpucu:Belgeye bakınız.)

6.2.3 Dizin Oluşturma ve Silme: mkdir ve rmdir

Dosyalarınızı düzenli bir şekilde tutmak istediğinizde yeni dizinler oluşturmak iyi bir fikirdir. Dosyalarınızı ilişkili konuya göre (örnek olarak) bu "klasörler" altında toplayabilirsiniz. Tabi ki ileri bir yapılandırma için, bu dizinler altında ilave dizinler oluşturabilirsiniz, hevesinizi, keyfi engeller tarafında sınırlamayınız.

Yeni dizinler oluşturmak için —emphmkdir komutu kullanılabilir. Bağımsız değişlen olarak bir veya daha çok dizin ismi alır, aksi halde yeni dizin oluşturmak yerine sadece bir hata mesajı alırsınız. İç içe dizinleri tek adımda oluşturabilmek için *-p* parametresini kullanabilirsiniz, aksi halde komut, oluşturduğunuz yoldaki dizinlerin hepsini sonuncusu hariç varmış gibi farzedecektir. Örneğin:

```
$ mkdir pictures/holiday
mkdir: cannot create directory ‘pictures/holiday’: \\
No such file or directory
$ mkdir -p pictures/holiday
$ cd pictures
$ ls -F
holiday/
```

Bazen bir dizine artık ihtiyaç duyulmaz. Dağınıklığı azaltmak için *rmdir* ("remove directory") komutu ile o dizini silebilirsiniz.

mkdir komutunda olduğu gibi en az bir silinecek dizin adı verilmelidir. Ayrıca silinecek dizinler boş olmalı, içerlerinde dosya veya dizinler barındırmamalıdır. Diğer komutta olduğu gibi bunda da her ismin sonundaki son dizin siliniz.

```
$ rmdir pictures/holiday
$ ls -F
pictures/
```

-p seçeneği ile en sağdakinden başlayarak boş olan tüm dizinler tek bir hamle ile silinebilir.

```
$ mkdir -p pictures/holiday/summer
$ rmdir pictures/holiday/summer
$ ls -F pictures
pictures/holiday/
$ rmdir -p pictures/holiday
$ ls -F pictures
ls: pictures: No such file or directory
```

Alıştırmalar

1. Ev dizininiz içerisinde *dir1*, *dir2* ve *dir3* isimli dizinleri içeren *grd1- test* isimli bir dizin oluşturun. *grd1- test/dir1* içerisine geçerek "hello" yazan *hello* isimli bir dosya (örneğin metin düzenleyici ile) oluşturun. *grd1- test/dir2* içerisinde "howdy" yazan *howdy* isimli bir dosya oluşturun. Bu dosyaların olmadığını kontrol edin. *dir3* isimli dizini *rmdir* komutu ile silin. Daha sonra *dir2* isimli dizini *rmdir* kullanarak silmeye çalışın. Ne oldu ve neden?

6.3 Dosya Arama Şablonları

6.3.1 Basit Arama Şablonları

Sık sık birden çok dosyaya aynı anda bir komutu uygulamak istersiniz. Örneğin, eğer "p" ile başlayıp ".c" ile biten tüm dosyaları *prog1* dizininden *prog2* dizinine kopyalamak istediğinizde, her dosyanın ismini açık bir şekilde yazmak, eğer en azından birçok dosya ile uğraşıyorsanız, oldukça sıkıcı olabilir. Kabuk arama şablonları kullanarak bu iş çok rahat bir şekilde yapılır.

Kabuk komut satırında belirlediğiniz bir kelimeyi yıldız ile kullanarak bunu sağlayabilirsiniz. Örneğin:

```
prog1/p*.c
```

Kabuk bu bağımsız değişkeni asıl program içerisinde bağımsız değişken ile eşleşen dosya adlarının sıralı bir listesi ile değiştirir. "Eşleşme", asıl dosya adında, yıldız yerine keyfi karakterlerin istenen uzunlukta sıralanmasına denir. Örneğin:

```
prog1/p1.c
prog1/polly.c
prog1/pop-rock.c
prog1/p.c
```

yukarıdaki isimler uygundur (örnekteki son ismin özelliği için kısa not — "istenen uzunluk", "sıfır uzunluğu" içerebilir!). Yıldız ile eşleştirilemeyecek bir karakter sizce ne olabilir?—taksim işareti; genellikle arama şablonundaki yıldız karakterini bulunduğunuz dizin ile sınırlamak daha iyidir.

echo komutunu ile kolay bir şekilde bu arama şablonlarını test edebilirsiniz.

```
$ echo prog1/p*.c
```

komutu herhangi bir zorunluluk, kural vb. olmaksızın eşleşen dosya isimlerini çıktı olarak verir.

Eğer gerçekten *dizin ağacında* belirli bir dizin ile başlayan tüm dosyalara bir komut uygulamak isterseniz, bunun için yapılacak bir çok yöntem vardır. Bu konuya **??**'te değineceğiz.

"*" arama şablonu "mevcut dizindeki tüm dosyaları" tanımlar, nokta ile başlayan gizli dosyalar hariç. Olası zahmetli sürprizleri engellemek için, şayet açıkça ".*" gibi anlam içeren dosyaları aramıyorsanız, arama şablonları gizli dosyaları göz ardı ederek çalışır.

DOS veya Windows[1] gibi işletim sistemlerinin komut satırında yıldız kullanımı ile rastgelmiş ve "*.*" kullanarak dizindeki tüm dosyalara işaret etmiş olabilirsiniz. Linux altında bu doğru değildir, "*.*" şablonu "nokta içeren tüm dosyalar" ile eşleşir, fakat nokta zorunlu değildir. Linux'taki karşılığı dediğimiz gibi "*"'dır.

Soru işareti arama şablonu sadece tek bir keyfi karakter ile eşleştirmek için kullanılır (yine taksim işareti hariç). Örnekteki gibi bir kalıp

```
p?.c
```

alttaki isimleri eşler:

```
p1.c
pa.c
p-.c
p..c
```

Unutmayın mutlaka bir karakter bulunmalıdır, "hiçbir şey" seçeneği burada geçerli değildir.

Özellikle çok önemli bir olaya dikkat etmelisiniz: Arama şablonunun genişlemesi kabuğun sorumluluğu altındadır! Çalıştırdığınız komutlar genellikle arama şablonları hakkında birşey bilmez ve ilgilenmezler de. Onlar yol isimlerini listeleyerek bakmaya başlar, fakat nereden geldikleri ile ilgilenmezler, örneğin, doğrudan mı yazılmış ya da bir arama sonucu mu diye kontrol etmezler.

Bu arada, kimse arama şablonlarının sonuçlarının yol isimleri kadar yorumlanabilir olduğunu söylemez. Örneğin, eğer bir dizin içerisinde "-l" isminde dosya içeriyorsa, "ls *"'nin o dizinde kullanılması size ilginç ve belki de sürpriz bir sonuç verecektir. (bkz: Alıştırma 6.9)

Eğer kabuk, arama şablonlarıyla eşleşen isimlerde dosyaları bulamazsa ne olur? Bu durumda söz konusu komut, arama şablonunu pas geçer; kendi arma işi ne ise onu yapar. Genellikle bu tarz arama şablonları dosya isimleri gibi yorumlanmışlardır, fakat söz konusu "dosya" ise bulunamaz ve hata mesajı bildirilir. Ancak, arama şablonları ile kullanışlı işler yapabileceğiniz birçok

[1] Muhtemelen CP/M için çok gençsinizdir.

komut vardır, bütün bunlarla, gerçekten meydan okuyabileceksiniz, kabuğun çalıştırdığı komut, kendi genişlemesini engel olmaz. (İpucu:tırnak işaretleri).

6.3.2 Karakter Sınıfları

Arama şablonlarında eşleştirilen karakterlerin daha hassas tanımlanması için "karakter sınıfları" önerilir: Aşağıdaki biçimdeki bir arama şablonunda

```
prog[123].c
```

köşeli parantez içindeki karakterler, içerisinde tam olarak o karakter geçen isimlerle eşleşir. Bu sebeple örnekteki şablon

```
prog1.c
prog2.c
prog3.c
```

ile eşleşir, fakat bunlar ile değil

```
prog.c  Tam olarak bir karakter gerekli
prog4.c  4 sıralandırmada yok
proga.c  a harfi de olmaz
prog12.c  Tam olarak bir karakter, lütfen
```

Daha kullanışlı bir gösterimle özel aralıkları belirleyebilirsiniz.

```
prog[1-9].c
[A-Z]bracadabra.txt
```

İlk satırdaki köşeli parantezdekiler tüm rakamlarla, 2. satırdaki de büyük harfle başlayanlarla eşleşir.

Yaygın karakter kodlamalarında harflerin bitişik olmadığını unutmayın:

```
prog[A-z].c
```

gibi şablon sadece *progQ.c* ve *progx.c* değil aynı zamanda *prog_.c* ile de eşleşir.(ASCII tablosunu kontrol ediniz, örn. "*man ascii*" kullanarak.) Eğer "sadece büyük ve küçük harf" eşleştirmek istiyorsanız, bunu

```
prog[A-Za-z].c
```

kullanarak gerçekleştirebilirsiniz.

```
prog[A-Za-z].c
```

şeklindeki bir yapı, harfler gibi şüpheli gözükse de, çift noktalı karakterleri yakalamaz.

Daha fazla kolaylık sağlamak için, "bunların dışındaki tüm karakterler" gibi yorumlanan olumsuz karakter sınıfları belirleyebilirsiniz:

```
prog[!A-Za-z].c
```

gibi bir ifade "g" ve "." arasında harf olmayan karakterli eşler. Her zamanki gibi, bölü işareti hariç.

6.3.3 Küme Parantezleri

Aşağıdaki ifadelerdelerdeki küme parantezlerinin genişlemesi

```
{red,yellow,blue}.txt
```

uzak bir ilişli olmasına rağmen kabuk arama şablonları ile ifade edilir. Kabuk bunu

```
red.txt yellow.txt blue.txt
```

olarak değiştirir.

Genellikle, küme parantezleri içerisinde virgüllerle ayrılmış metin içeren bir kelime, herbir kelimenin küme parantezi içerisindeki bir parça ile teker teker değiştirilmesi ile oluşan kelimeler ile değiştirilir. *Bu değiştirilme tamamen komut satırında metin tabanlıdır ve herhangi dosya veya dizinin tamamen olup veya olmamasından bağımsızdır*, sistemde gerçekte yol ismi olarak bulunan isimleri üreten arama şablonlarından farklıdırlar.

Bir kelime için kartezyen çarpım olarak sonuçlanacak birden fazla küme parantezi ifadeniz olabilir, diğer kelimeler bütün olası kombinasyonlardır:

```
{a,b,c}{1,2,3}.dat
```

ifadesi

```
a1.dat a2.dat a3.dat b1.dat b2.dat b3.dat c1.dat c2.dat c3.dat
```

sonucunu üretir.

Bu kullanışlıdır, örneğin, sistematik olarak yeni dizinler oluşturmak için işe yarar, arama şablonları burada yardımcı olamaz, onlar sadece zaten varolan şeyleri bulabilir:

```
$ mkdir -p revenue/200{8,9}/q{1,2,3,4}
```

Alıştırmalar

1. Aşağıda mevcut dizindeki dosyalar listelenmiştir

   ```
    prog.c prog1.c prog2.c progabc.c prog
   p.txt p1.txt p21.txt p22.txt p22.dat
   ```

   Bu isimler hangi arama şablonuyla eşleşir? (a) *prog*.c*, (b) *prog?.c*, (c) *p?*.txt*, (d) *p[12]**, (e) *p**, (f) *.**

2. "*ls*" ile "*ls **" arasındaki fark nedir? (İpucu: İkisini de alt dizinler içeren bir dizinde deneyin)

3. Aşağıdaki komut neden gösterilen çıktıyı üretir:

```
$ ls
-l file1 file2 file3
$ ls *
-rw-r--r-- 1 joe users 0 Dec 19 11:24 file1
-rw-r--r-- 1 joe users 0 Dec 19 11:24 file2
-rw-r--r-- 1 joe users 0 Dec 19 11:24 file3
```

4. "*" kullanımında neden nokta ile başlayan dosya isimleri eşlenmez?

6.4 Dosyalar ile ilgili işlemler

6.4.1 Kopyalama, Taşıma ve Silme — cp ve Arkadaşları

İstediğiniz dosyaları cp ("copy")("kopyala") komutu kullanarak kopyalayabilirsiniz. iki temel yaklaşım vardır:

Eğer cp'ye kaynak ve hedef dosya isimlerini (iki argüman) söylerseniz kaynak dosyanın içeriğinin 1:1 kopyasını hedef dosyada oluşturabilirsiniz. Eğer hedef dosya zaten varsa genelde cp üzerine yazılması gerekip gerekmediğini sormaz, fakat sadece yapar—Uyarı (veya -i parametresi) bunun için vardır.

Ayrıca hedef dosya ismi yerine hedef dizin ismi verebilirsiniz. Kaynak dosya eski ismini koruyarak o dizine kopyalanmış olacaktır.

```
$ cp list list2
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 joe users 2500 Oct 4 11:11 list
-rw-r--r-- 1 joe users 2500 Oct 4 11:25 list2
-rw-r--r-- 1 joe users 8765 Oct 4 11:26 passwd
```

Bu örnekte, ilk list dosyasının tam kopyasını list2 isminin altına oluşturduk. Sonra /etc/passwd dosyasını bulunduğumuz dizine (hedef dizin ismi olan nokta ile temsil edildi) kopyaladık. En önemli cp parametreleri Tablo **??**'te listelenmiştir. Tek kaynak dosyası yerine, kaynak dosyalarının daha uzun listesine(veya kabuk genel arama şablonuna) izin veriliyor. Ancak, bu yöntemde farklı isimli dosyaları kopyalamak olanaksızdır, fakat sadece farklı dizine kopyalanabilir. DOS'ta her TXT dosyasının aynı isimde ve BAK uzantısıyla "COPY *.TXT *.BAK" kullanarak yedek kopyalarını almak mümkün iken, Linux "cp *.txt *.bak" komutu genellikle bir hata mesajıyla sonuçlanacaktır.

Bunu anlamak için, kabuğun bu komutu nasıl çalıştırdığını kafanızda canlandırmak zorundasınız. Önce tüm genel arama şablonları için uygun dosya isimleriyle birlikte değiştirmeye çalışır, örneğin *.txt ile letter.txt ve letter.txt'yi. *.bak'ın *.txt uzantısında bağımlılığı ve mevcut dizinde*.bak için eşleşmeyen dosya isimleri sonucunda ne olur? Ama sonuç neredeyse hiç DOS kullanıcısının beklentisi olmayacak! Genellikle kabuk cp komutunda varolan bu dizin isminden(olası muhtemel) beri cp'nin görüş açısından başarısız olduğu genişletilmemiş *.bak genel arama şablonunu son argüman olarak alıp geçecektir.

cp komutu dosyanın tam kopyasını yaparken, fiziksel olarak saklama ortamı üzerinde dosya çoğaltılıyor ya da yeni, özdeş bir tanesi farklı depolama ortamında oluşturuluyor, mv ("move")("taşıma") komutu dosyayı farklı bir yere taşır veya adını değiştirmeyi sağlar. Bu kesinlikle dizin içeriğine bağlı bir işlemdir, tabi dosya farklı bir dosya sistemine taşınmadıkça — örneğin sabit disk bölümünden USB diske kopyalamak gibi. Bu olayda dosyanın fiziksel olarak taşınması, dosyanın yeni yere kopyalanması ve eski olanın silinmesi gereklidir.

mv'nin söz dizimi ve kuralları cp'ninkiler ile özdeştir — yine yalnızca bir tane yerine kaynak dosyalarının listesini belirtebilirsiniz, ve bu durumda komut sizden son argüman olarak dizin adını bekler. Temel fark mv dosyalar kadar iyi bir şekilde dizinleri de yeniden isimlendirebilirsiniz. mv'nin -b, -f, -i, -u ve -v parametreleri cp ile tanımlanmış özelliklere karşılık gelir.

```
$ mv passwd list2
$ ls -l
-rw-r--r-- 1 joe users 2500 Oct 4 11:11 list
-rw-r--r-- 1 joe users 8765 Oct 4 11:26 list2
```

Bu örnekte asıl dosya olan list2 passwd dosyasının yeniden isimlendirilmesiyle değiştirilmiştir. cp gibi mv de eğer hedef dosya adı varsa onaylamak için sormaz, acımasızca üstüne yazar.

Dosyaları silmek için olan komuta rm ("remove")("kaldırma") denir. dosyayı silmek için bulunduğunuz dizinde yazma yetkinizin olması gerekir. Bu nedenle hatta size ait olmayan dosyaları bile düzgün bir şekilde kaldırabildiğiniz ev dizininizde "malikane efendisi" sayılırsınız.

Dosya üzerinde yazma yetkisi, diğer deyişle, tamamen alakasız silme işleminde ilgili dosyadaki gibi, dosyanın hangi kullanıcı veya grupta olduğuna bağlıdır. rm işini cp veya mv gibi acımasızca ele alır—onaylama olmaksızın dosya sisteminden tamamen silinen söz konusu dosyalar gibi. Özellikle dikkatli olmalısınız, bilhassa kabuk genel arama şablonları kullandığınızda. Linux'ta dosya ismindeki nokta karakterinin DOS'ta özel bir anlamı yoktur. Bu nedenle, "rm *" komutu bulunan dizinde tüm gizli olmayan dosyaları siler. Alt dizinler zarar görmeden kalacaktır; ayrıca "rm -r *" ile onlar da silinebilir.

Sistem yöneticisi olarak, "rm -rf /" gibi bir komut ile tüm sistemi çöp

yapabilirsiniz; son derece önem gereklidir! “rm -rf foo*” yerine “rm -rf foo *” yazmak kolaydır.

rm'nin sildiği bütün dosyalarda bazı isimlilerin silinmemesi gerekiyorsa,”rm -i” biraz daha dikkatli bir şekilde işler:

```
$ rm -i lis*
rm: remove 'list'? n
rm: remove 'list2'? y
$ ls -l
-rw-r--r-- 1 joe users 2500 Oct 4 11:11 list
```

Bu örnek, her dosya için, rm silinmesi gerekip gerekmediğini (”evet” için ”e”) ve (”hayır” için ”h”) olarak sorduğunu anlatır.

KDE gibi masaüstü ortamları genellikle dosya yöneticisinden geri dönüşümü olası olan dikkatsizce silinen dosyaları görüntüleyen ”çöp kutusu” gösterimine destek verir. Komut satırı için benzer yazılım paketleri vardır.

Ek olarak -i ve -r parametrerin, rm cp'nin -v ve -f parametrelerinin kullanımına izin verir ve benzer sonuçlar üretir.

Alıştırmalar

1. Kendi ev dizininizde myservices denilen /etc/services dosyasının kopyasını oluşturun. Bu dosyayı srv.dat olarak tekrar isimlendirin ve /tmp dizinine kopyalayın (yeni adını koruyarak). Dosyanın iki kopyasını da silin.

2. mv neden -R parametresine sahip değildir(cp'nin sahip olduğu gibi)?

3. Farzedelim ki dizinlerinizden biri ”-file” isminde bir dosya içersin (isminin başında tire karakteri bulunsun). Bu dosyayı silmek için nasıl bir yol izlersiniz?

4. Eğer ”rm *”'ye kazara kurban gitmesini istemediğiniz yerde bir dizininiz olsa

   ```
   $ > -i
   ```

   gibi içerisinde ”-i” denilen bir dosya oluşturabilirsiniz (Bölüm **??**'de daha fazla detay açıklanacaktır). Eğer ”rm *” komutunu çalıştırsanız ne olur? Neden?

6.4.2 Dosyaların bağlanması — ln ve ln -s

Linux dosyalara "bağlantılar" adı verilen referanslar oluşturmaya izin verir, ve böylece aynı dosyaya birkaç isim atanabilir. Fakat bu neyi amaçlar? Dosya ve dizin isimleri için olan kısayollar, uygulamaların erişmesinde oluşturulması istenmeyen dosya silme işlemleri için güven, programcılar için kolaylık, geniş dizin ağaçları için alan tasarrufu sağlar ve birkaç sürümlerinde sadece küçük farklılıklarla mevcut olmalıdır.

ln("link")("bağlantı") komutu varolan dosyaya (ilk argüman) ek olarak, yeni bir isim(ikinci argüman) ataması yapar.

```
$ ln list list2
$ ls -l
-rw-r--r-- 2 joe users 2500 Oct 4 11:11 list
-rw-r--r-- 2 joe users 2500 Oct 4 11:11 list2
```

Dizinin şimdi list2 isminde yeni bir dosya içerdiği görülür. Aslında aynı dosyaya sadece iki tane referans vardır. Bu bilgi referans sayıcısı tarafından "ls -l" çıktısının ikinci sütununda ipucu olarak verilir. Değeri 2'dir, dosyanın gerçekten 2 tane isme sahip olduğunu belirtir. İki dosya isimlerinin gerçekten aynı dosyaya referans olup olmadığını, sadece "ls -i" komutu kullanarak tespit edebilirsiniz. Bu durumda, ilk sütundaki dosya numarası, iki dosya için de özdeş olmalıdır. Dosya numaraları, ayrıca inode (indeks düğümü) numaraları da denir, dosya sisteminde dosyaları eşsiz olarak tanımlar:

```
$ ls -i
876543 list 876543 list2
```

"Inode", "indirection node"'un kısaltmasıdır. Dosya numaraları, dosyaların hakkında ismi haricinde tüm bilgiyi tutar. Her dosya için kesinlikle bir tane dosya numarası vardır.

Eğer dosyalardan birinin içeriğini değiştirmek isterseniz, diğerinin içeriği de değişir, fakat aslında sadece bir tane dosya vardır (eşsiz dosya numarası 876543 olan). Biz sadece dosyaya başka bir isim verdik.

Dizinler dosya numaraları için dosya isimlerini eşlemede basit bir tablo görünümü üstlenirler. Açıkçası tabloda farklı isim içeren fakat aynı dosya numarası olan birçok girdi olabilir. İsim ve dosya numarasıyla birlikte olan dizin girdisine "bağlantı" adı verilir. İki bağlantısı bulunan dosya için, "orijinal" isme sahip olanı bulmanın imkansız olduğunu farketmelisiniz, ln komutu içerisindeki ilk parametre örnek olarak verilebilir. Sistemin bakış açısından iki isim tamamen farksızdır ve birbirine eşittir.

Bu arada, dizinlere bağlantı atamak Linux'ta geçerli değildir. Yalnızca, sistemin baktığı her dizin için sadece "." ve ".." birer istisnadır. Dizinlerin ayrıca dosya olduğu ve kendi dosya numaralarına sahip olduklarından beri,

dosya sisteminin nasıl içten bir şekilde uyguğunun izini sürebilirsiniz (Ayrıca bkz. Alıştırma 6.19).

İki dosyadan birinin silinmesi, dosya numarası 876543 için olan isimlerin azalmasına yol açar. Referans sayıcısını 0 değerine ulaşmadan önce dosyanın içeriği aslında silinmiş olur.

```
$ rm list
$ ls -li
876543 -rw-r--r-- 1 joe users 2500 Oct 4 11:11 list2
```

Dosya numaralarının sadece aynı fiziksel dosya sistemi (disk bölümü, USB sürücü, ...) içerisinde eşsiz olduğundan beri, bu tür bağlantılar sadece dosyanın bulunduğu aynı dosya sistemi üzerinde geçerlidirler.

Dosyanın içeriğinin silinmesi hakkındaki açıklama tam olarak doğru değildir: Eğer son dosya adı silinmişse, dosya artık açılamayabilir, fakat eğer bir süreç hala dosyayı kullanıyorsa, dosyanın açık bir şekilde kapanana veya sonlanmasına kadar devam edebilir. Unix sisteminde program sonlandırıldığında dosyanın yok olması geçici dosyaların işlemesi için yaygın bir deyimdir: Yazma, okuma için oluşturduğunuz dosyayı sonra hemen silebilirsiniz. Daha sonra dosyaya veriyi yazabilir ve devamında geri atlayıp başa dönerek tekrar okuyabilirsiniz.

ln'yi sadece iki dosya ismi argümanları değil ayrıca bir veya daha fazlası ile çalıştırabilirsiniz. İlk durumda, orijinal olarak aynı isimli bağlantı mevcut dizinde oluşturulacaktır (dosyanın konumlandığı yerde bi gerçekten farklı olmalı), ikinci durumda tüm isimlendirilen dosyalar son argüman olarak verilen dizinde orijinal adları "bağlanmış" olacaktır (mv gibi).

Hepsi bu kadar değil ancak: Linux sistemlerinde iki farklı tip bağlantı türü vardır. "sıkı bağ" denilen tür, ln komutu için ön tanımlı durum altında açıklanmıştır. Dosyayı teşhis etmek için daima dosya numarasını kullanır. Ek olarak, sembolik bağlantılar da vardır(ayrıca "sıkı bağlara" zıt olarak "dayanıksız bağ" adı verilir). Sembolik bağlantılar gerçekten bağlantı ismindeki "hedef dosya"yı içerirler, bayrak ile birlikte dosyanın sembolik link olduğunu belirtir ve erişim de hedef dosyaya yeniden yönlendirilerek yapılmalıdır.

Hard bağlantıların aksine, hedef dosya sembolik bağlantı hakkından hiçbir "şey" bilmez. Sembolik bağlantının oluşturulması veya silinmesi hiçbir şekilde hedef dosyayı etkilemez; ancak hedef dosya silindiğinde sembolik bağlantı "asılı" kalır, yani hiçbir yere referans olmaz (erişim sonucunda bir hata mesajı görüntülenir).

Sıkı bağların aksine, sembolik bağlantılar farklı fiziksel dosya sistemlerinde dosyalardaki gibi dizinlere bağlantı oluşturmaya izin verirler. Uygulamada, yol isimlerinin anlamı yoluyla bağın izinin tutulmasından beri, sembolik bağlantılar kolay olmasından dolayı sıklıkla tercih edilir.

Dosya veya dizin isimleri değiştiğinde fakat isteğe bağlı olarak geriye uyumluluk istenmesi açısından sembolik bağlantılar popülerdir. Örneğin kullanıcının

posta kutuları (okunmamış elektronik posta içerir) /var/mail dizini altında depolanmalıdır. Geleneksel olarak, bu dizine /var/spool/mail denmiştir, ve birçok programın içinde bu değer doğrudan koda gömülüdür. /var/mail'e geçişi hafifletmek için, dağıtım /var/mail'e işaret eden /var/spool/mail adı altında sembolik bağlantı içermiş bir biçimde ayarlanabilir (Sıkı bağlantıların dizinlere uygulanmasının geçerli olmadığı zamanlarda, sıkı bağlantılar kullanarak bunu yapmak imkansız olacaktır).

Sembolik bağlantı oluşturmak için, ln'ye -s parametresini vermelisiniz:

```
$ ln -s /var/log short
$ ls -l
-rw-r--r-- 1 joe users 2500 Oct 4 11:11 liste2
lrwxrwxrwx 1 joe users 14 Oct 4 11:40 short -> /var/log
$ cd short
$ pwd -P
/var/log
```

"-s" parametresi ile oluşturulan "dayanıksız bağlantılar"'ın aksine, ln komutu -b önceden bahsettiğimiz -b, -f, -i, ve -v parametrelerine destek verir. Artık ihtiyaç kalmayan sembolik bağlantıları kaldırmak için, düz dosyaları siler gibi kullanılan rm ile silinebilir. Bu işlem bağlantının hedefinden ziyade bağlantıya etki eder.

```
$ cd
$ rm short
$ ls
liste2
```

Alıştırmalar

1. Kendi ev dizininizde, keyfi içerikli bir dosya oluşturun(örneğin "echo Hello ¿ /hello" veya bir metin editörü kullanarak). link denilen dosyaya bir sıkı bağlantı oluşturun. Dosyanın iki tane isme sahip olduğundan emin olun. Dosyayı metin editörüyle değiştirmeye çalışın. Ne oluşur?

2. Geçenki alıştırmadaki dosyaya /symlink denilen bir bağlantı oluşturun. Sembolik bağlantının çalıştırılması yardımıyla dosyaya erişimin yapılıp yapılmadığını kontrol edin. Eğer sembolik bağlantının işaret ettiği dosyayı silerseniz ne olur?

3. Hangi dizinin .. bağlantısı "/" dizinine işaret eder?

4. Aşağıdaki komutu ve çıktısına göz atınız:

```
$ ls -ai /
2 . 330211 etc 1 proc 4303 var
2 .. 2 home 65153 root
4833 bin 244322 lib 313777 sbin
228033 boot 460935 mnt 244321 tmp
330625 dev 460940 opt 390938 usr
```

Açıkça / ve /home dizinleri aynı dosya numarasına sahip olduğu görülüyor. Hani iki dizin açıkça aynı dizinde bulunamazdı? Bu olguyu açıklayabilir misiniz?

5. Dizinlere sıkı bağlantı yapılmasının geçerli olmadığına önceden değinmiştik. Bunun nedeni nedir?

6. "ls -l "'nin çıktısında, 'nin altdizininin alt dizinler içermediğini nasıl söyleyebilirsiniz?

7. (Zeka oyunu/araştırma alıştırması:) Diskte sembolik bağlantı veya sıkı bağlantı için ne gereklidir? Neden?

6.4.3 Dosya İçeriğinin Görüntülenmesi—more ve less

Metin dosyalarının ekranda kullanışlı ve olanaklı olarak görüntülenmesinde, uzun dökümanları sayfa sayfa görmenizi sağlayan more komutu çok yardımcı bir araçtır. Çıktı, bir ekrandan sonra ve "–More–"'un son satırda(zaten görünen dosyanın yüzdesi tarafından tespit edilir) görüntülenmesinden sonra durur. Çıktı bir tuşa bastıktan sonra devam eder. Çeşitli tuşların anlamları Tablo **??**'te gösterilmiştir.

more ayrıca bazı parametreleri alır. -s("squeeze")("sıkıştırma") ile, boş satırların bulunduğu alanlar sadece bir tanesi kalacak şekilde sıkıştırılır, -l parametresi sayfa çıktısını yoksayar (genellikle "L̂" olarak tanımlanır) aksi halde çıktıyı durdurur. -n<number> <sayı> parametresi ekrandaki gösterilen satır sayılarını ayarlar, aksi halde more terminal tanımlamasına referans oluşturan TERM'den sayıyı alır.

more'un çıktısında hala çıktının başlangıcına doğru geri gitmenin genel olarak imkansız olması gibi can sıkan sınırlamalar vardır. Bu sebeple, gelişmiş sürümü olan less (hafif bir kelime oyunu var— "less is more" ("less daha fazla") olarak düşünün) daha fazla [böyle!] günümüzde çoklukla görülür. less her zamanki gibi yön tuşlarıyla metin çevresinde dolaşmanıza imkan tanır, arama alışkanlıkları geliştirilmiştir ve metininin başına/sonuna gitmeye izin verir. En yaygın kullanılan klavye komutları Tablo **??**'te özetlenmiştir.

Bölüm **??**'te bahsettiğimiz gibi, less genellikle man aracılığıyla kılavuz sayfalarını görüntüleyen bir program gibi hizmet eder. Böylece kılavuz sayfalarını incelerken tüm komutlar mevcuttur.

6.4.4 Dosya arama - find

"Burada foobar dosyası olmalı ... fakat nereye koymuştum?" şeklinde kim böyle bir hisse kapılmadı ki? Tabi ki sıkıcı bir şekilde tüm dizinlerinizi elinizle gezebilirsiniz. Fakat eğer bir şeyi kullanışlı kılmak için yardım etmezse Linux, Linux olmaz.

find komutu dizin ağacında özyinelemeli bir şekilde verilen kriterlere göre eşleşen dosyaları arar. "Özyinelemeli" alt dizinlerin, alt dizinleri şeklinde göz önünde bulundurarak devam eden anlamı içerir. find'ın çıktısı eşleşen dosyaların sonrasında diğer programlara aktarılabilecek olan yol isimlerini içerir. Sonraki örnek komut yapısını tanımlar:

```
$ find . -user joe -print
./list
```

Bu komut mevcut dizinde tüm alt dizinlerin dahilinde joe kullanıcısına ait olan dosyaları arar. -print komutu terminalde sonucu görüntüler (bizim örneğimizde tek dosya). Kolaylık için, eğer eşleşen dosyalarla ne yapacağınızı belirlemediyseniz, -print varsayılan olacaktır.

find kendi işini yaparken birkaç argümana ihtiyacı olduğunu göz önünde bulundurun.

Başlangıç dizini özenle seçilmelidir. Eğer kök dizinini seçerseniz, gerekli dosya(lar) —eğer varsa— kesinlikle bulunacaktır, fakat arama uzun zaman alır. Tabi ki sadece bu dosyaları uygun ayrıcalıklı yerlerde ararsınız. Başlangıç dizini için tam yol adı vermek çıktıdaki dosya isimlerinin mutlak olmasına neden olur, başlangıç dizini için göreceli yol adları doğrultusunda göreli yol adları üretir. Tek başlangıç dizini yerine, dönüşte aranacak dizinlerin listesini belirleyebilirsiniz.

Bu parametreler dosyalar üzerinde detaylı bir şekilde gereksinimleri tanımlar. Tablo **??**'da en önemli denemeler gösterilmiştir. find belgeleri çok daha fazlasını açıklar.

Eğer çoklu denemeler aynı zamanda verilirse, eşleşecek dosyaların hepsi dolaylı olarak birlikte mantıksal VE işlemine uğrarlar. find ek mantıksal operatörlere destek verir (Tablo **??**'ye bakınız).

Mantıksal operatörleri denerken hatalardan kaçmanın yolu, denemelerin parantezlerle kapatılmasıdır. Parantezler tabi ki kabuktan kaçırılmış olmalıdır:

```
$ find . \( -type d -o -name "A*" \) -print
./.
./..
./bilder
./Attic
$ _
```

Bu örnek dizine işaretçi olan veya "A" ile başlayan, ya da her ikisi koşulundaki dosyaların isimlerini listeler.

Eylemler Önceden bahsettiğimiz gibi, arama sonuçları ekranda -print parametresiyle görüntülenebilir. Buna ek olarak, -exec ve -ok adında dosya isimleri içeren komutları çalıştıran 2 parametre vardır. -ok ve -exec arasındaki tek fark, -ok gerçekten komutu çalıştırmadan önce bilgi için kullanıcıya sorar; -exec ile üstü kapalı bir şekilde bu işlemin yapıldığı varsayılır. Biz -exec'i ele alırken kendimizi kısıtlayacağız.

-exec parametresini yöneten bazı genel kurallar şunlardır:

- -exec'in devamındaki komut noktalı virgül (";") ile sonlandırılmak zorundadır. Birçok kabuklarda noktalı virgülün özel karakter olduğundan beri, find'ı görünebilir yapmak için noktalı virgül kaçırılmış olmak zorundadır (örneğin "\" veya yorumların kullanımı gibi).
- Komutun içerisindeki iki süslü parantez ("{}") bulunan isimle, dosya ismini yer değiştirir. Bu dosya adlarında boşluk sorunlarını önlemek için tırnak içinde parantez içine almak en iyisidir.

Örneğin;

```
$find . -user joe -exec ls -l '{}' \;
-rw-r--r-- 1 joe users 4711 Oct 4 11:11 file.txt
$ _
```

Bu örnek mevcut dizin(ve altında) içerisindeki tüm dosyaların her biri için için test kullanıcısına ait olanları arar ve "ls -l" komutunu çalıştırır. Aşağıdaki daha mantıklı olacaktır:

```
$ find . -atime +13 -exec rm -i '{}' \;
```

Bu etkileşimli olarak mevcut dizin(ve altındaki) son 2 haftadır erişilmeyen dosyaları siler.

Bazen - yukarıdaki son örnekteki gibi— bulunan her bir dosya adı için -exec kullanarak yeni bir süreç başlatmak çok verimsiz olur. Bu durumda

komutu çalıştırmadan önce olası bütün dosya isimlerini toplayan xargs komutu kullanışlı olacaktır.

```
$ find . -atime +13 | xargs -r rm -i
```

xargs standart girdiyi, maksimum (yapılandırılabilir) karakter veya satıra kadar okur ve bu materyali belirlenen komut için argüman olarak kullanır (bu örnekte rm). Girdide, argümanlar boşluk karakterleri (tırnaklar veya "̈kullanarak kaçırılabilir) veya yeni satırlar tarafından ayrılmıştır. Komut girişi bitirmeye gerektiği kadar çağrılır.— -r parametresi sadece eğer find gerçekten dosya ismi gönderdiyse çağrılmasını sağlar; aksi halinde en azından bir kez çalıştırılmış olacakatır.

Garip dosya isimleri find/xargs kombinasyonunun kullanımında sorun oluşturabilir, örneğin ayrıcılar olarak sorun olabilen boşluklar veya gerçekten de, yeni satırlar içerenler. Sihirli çözüm find'da "-print"'in yaptığı gibi dosya isimlerini çıktıya veren, fakat yeni satırlar yerine null byte'lar kullanarak ayıran "-print0" parametresinin kullanımıyla sağlanır. Null byte'ın yol isimlerinde geçerli karakter olmadığından beri, karışıklık artık mümkün değildir. xargs bu tarz girdideki gibi "-0" parametresi kullanarak çağırılmak zorundadır:

```
$ find . -atime +13 -print0 | xargs -0r rm -i
```

Alıştırmalar

1. Sisteminizdeki 1MB'dan daha büyük olan tüm dosyaları bulun ve isimlerini çıktıya verin.

2. Bir olağandışı isime sahip dosyayı find kullanarak nasıl silebilirsiniz (örneğin eski kabukların başa çıkamadığı görünmez kontrol karakterleri veya umlautlar)?

3. (Kitap aracılığıyla ikinci kez) Nasıl kendinize ait bir kez çıktığınızda silinen dosyaları koruma altına alırsınız?

6.4.5 Çabukça Dosyaları Bulma - locate ve slocate

find komutu birçok farklı kritere bağlı olarak dosyaları arar fakat başlangıç dizini altındaki dizin ağacını tamamen gezmek zorundadır. Ağaç boyutuna göre, bu dikkate değer bir zaman alır. Tipik bir uygulamada, belirli isimlerle aranan dosyalar—hızlandırılmış bir yöntem vardır.

locate komutu kabuk genel arama şablonunda eşleşen isimlere ait düm dosyaları listeler. En sıradan durumda, bu basit karakterlerin dizisidir:

```
$ locate letter.txt
/home/joe/Letters/letter.txt
```

```
/home/joe/Letters/grannyletter.txt
/home/joe/Letters/grannyletter.txt~
```

Bununla birlikte oldukça önemli bir faydası olan locate (LPIC1 müfredatının parçasında bu olgu vurgulanmıştır), tüm Linux dağıtımlarında varsayılan yüklemenin bir parçası değildir. Örneğin, eğer bir SUSE dağıtımı kullanıyorsanız, locate kullanabilmek için önce açıkça findutils-locate paketini kurmanız gerekir.

"*", "?", ve "[...]" karakterleri kabukta yaptıkları gibi locate'te de aynı görevi üstlenirler. Fakat genel arama karakterleri olmayan sorgu herhangi bir yerdeki şablonun içerdiği tüm dosya isimleri ile yerleşirken, genel arama karakterleriyle yapılan sorgu sadece tamamen, baştan sona, şablonun tasvir ettiği isimleri geri döndürür. Bu nedenle locate için olan şablon sorguları genellikle "*" ile başlar:

```
$ locate "*/letter.t*"
/home/joe/Letters/letter.txt
/home/joe/Letters/letter.tab
```

Kabuk genel arama karakterleri içeren locate sorgularının açılmaya çalışılmasından kabuğu korumak için etrafına tırnaklar koyduğunuzdan emin olun. Bölü ("/") özel olarak işlenmez:

```
$ locate Letters/granny
/home/joe/Letters/grannyletter.txt
/home/joe/Letters/grannyletter.txt~
```

locate çok hızlıdır çünkü dosya sistem ağacında gezinmez, fakat önceden ubdatedb programı kullanılarak oluşturulması gereken dosya isimleri "veritabanı"'nı kontrol eder. Bu son veritabanı güncellemesinden beri sisteme eklenen dosyaların locate tarafından yakalanamayacağı anlamına gelir ve aksi bir şekilde bu zamanda silinmiş olan dosyaların isimlerini de çıktıya verebilir.

locate'in varolan dosyalarını sadece "-e" parametresi kullanarak alabilirsiniz, fakat bu yöntem locate'in hız avantajını reddeder.

updatedb program locate için veritabanı oluşturur. Bunun önemli büyüklükte zaman aldığından beri, sistem yöneticiniz genellikle bunu sistem çok iş yapmayacağı zaman (tahminen gece geç saatte) çalıştırılacak şekilde ayarlar. Bunun için gerekli olan cron hizmeti Advanced Linux'ta detaylarıyla anlatılacaktır. Şimdi birçok Linux dağıtımları ubdatedb'nin aralıklarla sık sık sık çalıştırılmasına neden olurlar.

Sistem yöneticisi olarak, veritabanı kurarken dikkate alınması gereken dosyaları updatedb 'ye söyleyebilirsiniz. Dağıtımınıza bağlı olarak neler olur detaylıca açıklayalım: updatedb kendi kendine konfigürasyon dosyasını okuyamaz, fakat ayarları komut satırından ve (kısmen) çevresel değişkenlerden alır.

Buna rağmen, birçok dağıtım genellikle ilgili olan ortam değişkenlerinin bulunduğu /etc/updatedb.conf veya /etc/sysconfig/locate gibi dosyaları okuyan kabuk betiğinden updatedb'yi çağırır.

Örneğin /etc/cron.daily gibi bir dosya bulabilirsiniz (detaylar, dağıtımınıza bağlı olarak değişiklik gösterebilir).

Örnek olarak updatedb'nin belirli dizinlerdeki aramalarına ve bu arama işleminde diğer dosyaları atmalasona sebep olabilirsiniz; program ayrıca sadece bir bilgisayarın veritabanı oluşturnası gerektiği gibi, birçok bilgisayar tarafından kullanılan ve onların kendi kök dizinlerindeki veritabanlarında bulunan "ağ dosya sistemi"'ni belirlemenize imkan tanıyor.

Önemli bir yapılandırma ayarı da kullanıcı kimliğiyle updatedb'nin çalıştırılmasıdır. Temelde iki olasılık vardır:

- updatedb root gibi çalışır ve böylece veritabanındaki her dosyaya giriş yapabilir. Bu ayrıca kullanıcıların birbirlerininkine erişemediği dizinlerdeki dosya isimlerinin de görüntülenmesini sağlar, örneğin, diğer kullanıcıların ev dizinleri.

- updatedb "hiçkimse" kullanıcıları gibi sınırlı ayrıcalıklarla çalışır. Bu durumda, sadece "hiçkimse" kullanıcısı tarafından okunabilen dizinlerdeki isimler görüntülenir.

slocate programı —alışılmış locate'e alternatif olarak— bu problemi dosyanın sahibi, grubu ve izinlerinin ve buna ek olarak dosyanın isminin de veritabanında bulunması sayesinde atlatır. Sadece slocate'i hangi kullanıcı çalıştırıyorsa o dosya ismini çıktıya verir, adeta söz konusu dosyaya erişmek gibi. slocate updatedb programıyla birlikte gelir, fakat bu sadece slocate için tahsis edilen bir isimdir.

Birçok durumda, slocate de ayrıca locate komutu gibi çalıştırılabilir.

Alıştırmalar

1. README çok yaygın bir dosya ismidir. Sisteminizde README denilen tüm dosyaların mutlak yol isimlerini bulunuz.

2. Kendi ev dizininizde yeni bir dosya oluşturun ve locate çağırıldığında bu dosyanın listelenmeyeceği konusunda kendinizi ikna edin (uygun egzotik bir dosya adı kullandığınızdan emin olun. updatedb'i çalıştırın (büyük olasılıkla yönetici yetkileri kullanarak). locate sonradan sizin dosyanızı bulabildi mi? Dosyayı silin ve bu adımları tekrarlayın.

3. Kendinizi slocate'in /etc/shadow gibi dosyaları normal kullanıcı olarak çalıştırılıp arama yaptığına ikna edin.

Bu Bölümdeki Komutlar

cd Kabuğun mevcut çalışma dizinini değiştirir

convmv Karakter kodlamaları arasında dosya isimlerini dönüştürür

cp Dosyaları kopyalar

find Verilen belirli kriterler ile eşleşen dosyaları arar

less Metinleri(kılavız sayfaları gibi) sayfalar halinde görüntüler

ln "sıkı" veya sembolik bağlantıları oluşturur

locate Dosya ismi veritabanında verilen isme göre dosyaları bulur

ls Dosya bilgisi veya dizin içeriklerini görüntüler

mkdir Yeni dizinler oluşturur

more Metin verilerini sayfalar halinde görüntüler

mv Dosyaların ismini değiştirir veya istenilen dizinlere taşır

pwd Mevcut çalışma dizininin adını görüntüler

rm Dosyaları veya dizinleri siler

rmdir Boş dizinleri siler

slocate Dosya ismi veritabanında verilen isme göre hesaptaki dosya izinlerini alarak dosya arar

updatedb locate için dosya ismi veritabanı oluşturur

xargs Standart girdiden komut satırlarını oluşturur.

Özet

- Dosya isimleri neredeyse tüm olası karakterleri içerebilir. Ancak taşınabilirlik için, harfler, rakamlar ve bazı özel karakterler kullanmaya kendinizi alıştırmalısınız.
- Linux dosya isimlerinde büyük/küçük harfleri ayırt eder.
- Mutlak yol isimleri her zaman bölü işareti ile başlamalı ve söz konusu olan dosya veya dizin, kökten başlayarak dizin ağacıyla birlikte verilmelidir. Göreceli yol isimleri "mevcut dizin"'den başlar.

- cd komutu kullanarak kabuğun mevcut dizinini değiştirebilirsiniz. Ve ismini pwd kullanarak görüntüleyebilirsiniz.
- ls dosya ve dizinler hakkında bilgileri görüntüler.
- mkdir ve rmdir kullanarak dizin oluşturabilir veya silebilirsiniz.
- cp, mv ve rm komutları dosya ve dizinleri kopyalar, taşır ve siler.
- ln komutu "sıkı" ve "sembolik" bağlantılar oluşturmanıza olanak tanır.
- more ve less terminal çıktısında dosyaları (ve komut çıktısını) sayfalar halinde görüntülemeye yarar.
- find belirli kriterlere göre eşleşen dosya ve dizinleri arar.

Table 6.1: ls'deki bazı dosya gösterimleri

Dosya Tipi	Renk	Sonek (ls -F)	Harf türü (ls -l)
düz metin	siyah	none	-
çalıştırılabilir dosya	yeşil	*	-
dizin	mavi	/	d
link	cyan	@	l

Table 6.2: Bazı ls parametreleri

Parametre	Sonuç
-a veya –all	Aynı zamanda gizli dosyaları da görüntüler
-i veya –inode	Eşsiz dosya numarasını görüntüler (inode number)
-l veya –format=long	Ek bilgi görüntüler
-o veya –no-color	Çıktı renk-kodlamasını atlar
-p veya -F	Eklenen özel karakterle dosya türünü işaretler
-r veya –reverse	Sıralamayı ters yapar
-R veya –recursive	Alt dizinlerle birlikte öz yinelemeli olarak listeler (DOS: D
-S veya –sort=size	Boyuta göre sıralar (büyükten küçüğe doğru)
-t veya –sort=time	Değiştirilme tarihine göre sıralar (yeniden eskiye doğru)
-X veya –sort=extension	Dosya uzantısına göre sıralar ("dosya türüne" göre)

Table 6.3: *cp* seçenekleri

	Seçenek	Sonuç
-b	(yedek)	Varolan hedef dosyalarının yedek kopyalarını isimleri ile tildayı
-f	(zorla)	Varolan hedef dosyaları sorgu mesajı vermeden üzerine yazar.
-i	(etkileşimli)	Varolan hedef dosyalarının üzerine yazılıp yazılmayacağını sora
-p	(korumak)	Kaynak dosyanın bütün özelliklerini korumak için kopyalamaya
-R	(özyinelemeli)	Dizinleri tüm içerikleriyle birlikte kopyalar
-u	(güncelleme)	Yalnızca kaynak dosya hedef dosyadan yeni ise kopyalar(veya h
-v	(ayrıntılı)	Ekrandaki tüm faaliyeti görüntüler

Table 6.4: more için klavye komutları

Tuş	Sonuç
	Scrolls up a line
	Scrolls up a screenful
b	Scrolls back a screenful
h	Displays help
q	Quits more
/ <kelime>	Searches for <kelime>
! <komut>	Executes <komut> in a subshell
v	Invokes editor (vi)
+ l	Redraws the screen

Table 6.5: less için klavye komutları

Tuş	Sonuç
veya e veya j veya	Scrolls up one line
f veya	Scrolls up one screenful
veya y veya k	Scrolls back one line
b	Scrolls back one screenful
veya g	Jumps to the beginning of the text
veya + g	Jumps to the end of the text
p <percent>	Jumps to position <percent> (in %) of the text
h	Displays help
q	Quits less
/ <word>	Searches for <word> towards the end
n	Continues search towards the end
? <word>	Searches for <word> towards the beginning
+ n	Continues search towards the beginning
! <command>	Executes <command> in subshell
v	Invokes editor (vi)
r veya + l	Redraws screen

Table 6.6: find için deneme durumları

Deneme	Açıklama
-name	Dosya adı örneğini belirler. Tüm kabuk arama şablonu karakterleri geçerli
-type	Dosya türünü belirler (bkz. Bölüm **??**). İçerdikleri: b blok aygıt dosyası c karakter aygıt dosyası d dizin f düz dosya l sembolik bağlantı p FIFO (adlandırılmış yöneltme) s Unix alan soketi
-user	Kullanıcıya ait olan dosyayı belirler. Kullanıcı adları, sayısal kullanıcı kim
-group	Gruba ait olan dosyayı belirler. -user gibi grup adları, sayısal grup kimlikl
-size	Ayrıntılı dosya boyutunu belirler. Sade sayılar 512-byte'lık bloklar halinde
-atime	(ing. access) son erişim(okuma/yazma) tarihi bazında dosyaları aramaya i
-mtime	(ing. modification) değiştirme zamanına göre seçer.
-ctime	(ing. change) son dosya numarası zamanının değiştirilmesine göre seçer(in
-perm	Dosyanın eşleşmesi gerektiği yetkiler kümesini belirler. Yetkiler sekizlik say
-links	Uygun dosyaların eşleştiği referans sayı değerini belirler.
-inum	Verilen dosya numarasıyla eşleşen bağlantıları bulur.

Table 6.7: find için mantıksal operatörler

Parametre	Operatör	Anlam
!	Değil	Sonraki deneme eşleşmemelidir
-a	Ve	-a'nın sağ ve sol yanındaki iki deneme de eşleşmelidir
-o	Veya	-o'nun sağ veya sol yanındaki iki denemeden en azından biri

Bölüm 7

Düzenli İfadeler

Amaçlar

- Basit ve gelişmiş düzenli ifadeleri formüle edebilmek ve anlamak
- grep programı ve egrep, fgrep varyasyonlarını öğrenmek

Önceden Bilinmesi Gerekenler

- Linux, kabuk ve komutların temel bilgisine sahip olmak(önceki bölümlerden)
- Dosya ve dizinleri yönetmek (Bölüm **??**)
- Metin editörünü kullanmak (Bölüm **??**)

7.1 Düzenli ifadeler: Temeller

Çoğu linux komutları metin işleme için kullanılır-"buna benzer bütün satırlara xyz yap" biçimindeki şablonlar ile tekrar tekrar karşılaşırız. Metin parçalarını, çoğunlukla dosyalardaki satırları tanımlamak için kullanılan çok güçlü bir araç, "düzenli ifade" [1] olarak adlandırılır. İlk bakışta düzenli ifadeler kabuğun dosya ismi arama şablonlarına benzer görünür (Bölüm **??**), fakat düzenli ifadeler farklı şekilde işler ve daha çok olanak sağlar.

Düzenli ifadeler, kendileri de düzenli ifadeler olarak nitelendirilen daha ilkel ifadelerden özyinelemeli olarak oluşturulur. En basit düzenli ifadeler

[1] Bu bilgisayar bilimlerinden bir terimdir ve "harflerin" birleşiminden, bir harf kümesi içindeki seçeneklerden ve bunların sınırsız tekrarından oluşan karakter dizisi kümelerinin davranış yöntemlerinden birini belirtir. Düzenli ifadeleri tanıyan yordamlar programlama dili derleyicileri gibi çoğu programın temel yapı taşlarıdır. Düzenli ifadeler Unix tarihinde çok erken ortaya çıkmışlardı; ilk Unix geliştiricilerinin çoğu bilgisayar bilimleri geçmişine sahipti, yani bu fikir onlara yabancı değildi.

harfler, rakamlar ve genel karakter kümesideki diğer karakterlerdir ve bunlar kendilerini temsil ederler. Örneğin "a", "a" karakteriyle eşleşen düzenli ifadedir. "abc" düzenli ifadesi, "abc" karakter dizisi ile eşleşir. Karakter sınıfları kabuk arama şablonlarına benzer şekilde tanımlanabilir; dolayısıyla "[a-e]" düzenli ifadesi, a'dan e'ye herhangi bir karakterle eşleşir ve "a[xy]b" düzenli ifadesi "axb" veya "ayb" ile eşleşir. Kabukta olduğu gibi aralıklar birleştirilebilir-"[A-Za-z]" düzenli ifadesi tüm büyük ve küçük harflerle eşleşirama bir aralığın tümleyeni biraz farklı şekilde oluşturulur:" [^abc]" düzenli ifadesi "a", "b" ve "c" dışındaki tüm karakterlerle eşleşir (kabukta "[!abc]" ile oluşturulur). Nokta "." kabuk arama şablonundaki soru işaretine karşılık gelir, sadece 1 karakterle-bunun tek istisnası "\n" karakteridir.-eşleşecektir. Bu yüzden "a.c", "abc", "a/c" gibi ifadelerle eşleşir ama aşağıdaki gibi birden çok satır yapısıyla eşleşmez.

```
a
c
```

Çoğu program satır satır işlendiğinden ve çoklu satır yapılarını işlemek daha zor olacağından çoklu satır yapısına yer verilmez (bazen bunu yapma imkanı olsaydı güzel olmayacağını söylemiyoruz).

Kabuk arama şablonlarının her zaman bir dosya isminin başından başlayarak eşleşmesi gerekiyorken düzenli ifadelere dayanarak satırları seçen programlarda düzenli ifadenin satırın herhangi bir yeri ile eşleşmesi yeterlidir. Ancak bu kısıtlanabilir: Bir şapka karakteri ("^") ile başlayan düzenli ifadeler yalnızca satır başında eşleşme yapar ve dolar ("$") işaretiyle biten bir ifade sadece satır sonunda eşleşme yapar. Her bir satırın sonundaki alt satır karakteri yoksayılır. Böylece "xyz" ile biten tüm satırları seçmek için "xyz\n$" yazmak zorunda kalmak yerine "xyz$" kullanabilirsiniz.

Daha ciddi konuşursak "^" ve "$" işaretleri sırasıyla satırın başlangıcındaki ve satırın sonunda yeni satır karakterinin hemen solundaki görünmez kavramsal karakterlerle eşleşir.

Son olarak, "*" kendisinden önce gelen düzenli ifadenin isteğe göre defalarca tekrar edilebileceğini (veya hiç kullanılmayabileceğini) göstermek için kullanılabilir. Yıldızın kendisi girdideki hiçbir karakterin yerine geçmez ama sadece kendisinden önce gelen ifadeyi değiştirir-sonuç olarak kabuk arama şablonu "a*.txt", "^a.*\\.txt " düzenli ifadesine karşılık gelir (burada düzenli ifadenin başlangıcında ve girdi satırının sonunda "anchoring" olduğunu ve kaçırılmamış nokta karakterinin herhangi bir karakterle eşleştiğini hatırlayın). Tekrarlama birleştirmeden daha önceliklidir;" ab*" bir "a"nın peşinden gelen bir veya daha fazla (veya hiç) "b"yi ifade eder , "ab"nin birden fazla sayıda tekrarı olmaz.

7.1.1 Düzenli İfadeler: Ekstralar

Bir önceki bölümdeki açıklamalar düzenli ifadeler kullanan neredeyse tüm Linux programları için geçerlidir. Birçok program ilave işlevsellik veya gösterim kolaylığı sağlayan farklı eklentileri destekler. En iyi gerçekleştirimler, şu an Tcl, Perl, Python gibi modern betik dilleri içinde bulunur. Bu dillerin gerçekleştirimleri bilgisayar bilimlerinde kullanılan özgün düzenli ifadelerin gücünü şimdiye kadar aşmıştır.

Bazı yaygın eklentiler:

Kelime ayıraçları "\<" bir kelimenin başı (harf olmayan bir karakterin harften önce geldiği yer) ile eşleşir. Benzer olarak, "\>" kelimenin sonu (harf olmayan bir karakterin bir harfi takip ettiği yer) ile eşleşir.

Gruplama Parantezler "((...))" düzenli ifadelerin birleşiminin tekrarına izin verir. "a(bc)*" ifadesi "a"dan sonra "bc"nin bir veya daha fazla tekrarı ile eşleşir.

Alternatif Dikey çubukla ("|") birden çok düzenli ifade arasından seçim yapılabilir. "hava(alanı|limanı|sahası)" ifadesi, "havalimanı", "havasahası" ve "havaalanı" ile eşleşebilir. Fakat tek başına "hava" ile eşleşmez.

Seçimli ifadeler soru işareti ("?") düzenli ifadelerin seçimli olmasını sağlar, yani ya bir kez geçer ya da hiç geçmez. "uçak(savar)?" ifadesi ya "uçak" ya da "uçaksavar" ile eşleşir.

En az bir tekrar ("+") işareti önceki düzenli ifadenin en az bir kez geçmek zorunda olması haricinde ("*") operatörü ile tamamen aynıdır.

Belirli tekrar sayıları Kıvrık parantezler içinde en az ve en çok tekrar sayılarını belirtebilirsiniz. "ab{2,4}" ifadesi "abb","abbb" ve "abbbb" ifadeleri ile eşleşir ama "ab" yada "abbbbbb" ifadeleri ile eşleşmez. En az veya en çok sayıyı es geçebilirsiniz. En az sayı yoksa 0 kabul edilir, en çok sayı yoksa "sonsuz" kabul edilir.

Geri-referans "\\n" benzeri bir ifadeyle girdinin düzenli ifade içindeki "n"inci sayılı parantez içindeki ifade ile eşleşen parçasını belirtebilirsiniz. Mesela "(ab)\\1" ifadesi "abab" ile eşleşir. "(ab*a)x\1" ifadesini işlerken parantezler "abba" ile eşleştiyse tüm ifade "abbaxabba" ile eşleşir (başka hiçbirşeyle eşleşmez). Daha fazla ayrıntı GNU grep'in dökümanları içinde mevcuttur.

Aç gözlü olmayan (eşleşme): “*” , “+” ve “?” operatörleri genellikle açgözlüdür, yani mümkün olduğunca çok sayıda girdiyle eşleşmeye çalışırlar: ”^a.*a” ifadesi “abacada” giriş karakter dizisine uygulandığında “aba” yada “abaca” ile değil, “abacada” ile eşleşir. Ancak aynı operatörlerin açgözlü olmayan uyarlamaları da vardır. “*?” ,”+?” ve “??” girdinin mümkün olduğu kadar azıyla eşleşmeye çalışırlar. Örneğimizde “a.*?a” , “aba” ile eşleşebilir. Kıvrık parantez operatörü de aç gözlü olmayan uyarlama sağlayabilir.

Her program her eklentiyi desteklemez. Tablo **??** bazı önemli programları kısaca gösterir. Emacs, Perl ve Tcl burada tartışılmayan bir çok eklentiyi destekler.

Table 7.1: Düzenli İfade Desteği

Eklenti	GNU Grep	GNU egrep	trad egrep	vim	emacs	Perl	Tcl
Kelime ayıraçları	•	•	•	•[1]	•[1]	•[4]	•[4]
Gruplama	•[1]	•	•	•[1]	•[1]	•	•
Alternatif	•[1]	•	•	•[2]	•[1]	•	•
Seçimli ifadeler	•[1]	•	•	•[3]	•	•	•
En az bir tekrar	•[1]	•	•	•[1]	•	•	•
Sınırlar	•[1]	•	o	•[1]	•[1]	•	•
Geri-referans	o	•	•	o	•	•	•
Aç gözlü olmayan eşleşme	o	o	o	•[4]	•	•	•

[1]Kendinden önce “\” işareti gelmelidir, örn: “ab+” yerine “ab\+”.
[2]Parantez kullanımı gerekmez, alternatifler daima tüm ifadeye aittir.
[3]“?” yerine “\=” kullanılır.
[4]Sözdizimi tamamen farklıdır (kendi dökümanlarına başvurun).

7.2 Dosya içinde metin arama – grep

Düzenli ifadelerin kullanıldığı en önemli Linux programlarından biri grep'tir. Verilen düzenli ifade ile eşleşen metinleri bir veya daha fazla dosya içindeki satırlarda arar. Eşleşen satırlar çıktı olarak verilir, eşleşmeyenler atılır.

Grep'in iki çeşidi vardır. Geleneksel olarak sadeleştirilmiş fgrep (“fixed (sabit)”) düzenli ifadelere izin vermez-karakter dizileri ile sınırlıdır- ama çok hızlıdır. egrep (“extended (genişletilmiş)”) ek düzenli ifade operatörlerine izin verir ama biraz daha yavaştır ve daha fazla hafızaya ihtiyaç duyar.

Bu gözlemler bir dereceye kadar doğru olabilir. Özel olarak grep ve egrep düzenli ifadeleri değerlendirmek için tamamiyle farklı algoritmalar kullanır. Bu da girdinin büyüklüğüne ve düzenli ifadenin yapısına ve boyutuna bağlı olarak çok farklı performans sonuçlarına yol açar. Yaygın Linux dağıtımlarında grep'in 3 çeşidi de aslında aynı programdır; arama şablonları için izin verdikleri söz dizimiyle ayrılırlar.

grep'in sözdizimi, aramak için en az bir düzenli ifade ister. Bunu, içinde arama yapılacak metin dosyasının veya dosyalarının isimleri izler. Eğer hiç

dosya ismi belirtilmemişse grep standart girdiye başvurur (Bölüm **??**'e bakın).

Girdi içerisinde aranacak düzenli ifade Bölüm **??**'deki temel düzenli ifadelerin yanında Bölüm **??**'deki gelişmiş düzenli ifadelerin çoğunu da içerebilir. Ancak grep ile "\+","\?" ve "\{" operatörlerinden önce ters bölü gelmelidir. (egrep için bu gerekli değildir.) Maalesef hiç aç gözlü olmayan operatör yoktur.

Eğer düzenli ifade kabuk arama şablonuna benziyorsa ve tek bir karakter dizisinden daha karmaşıksa kabuğun düzenli ifadeyi genişletmeye çalışmasını önlemek için düzenli ifadeleri tek tırnak içine almalısınız.

Düzenli ifade ve dosya isimlerinden başka komut satırına çeşitli seçenekler geçilebilir. (Tablo **??**'ye bakın.)

Table 7.2: grep ile kullanılabilecek seçenekler

	Seçenek	Sonuç
-c	(sayı)	Eşleşen satırların sayısını verir.
-i	(yoksayma)	Küçük ve büyük harfler eşittir.
-l	(listeleme)	Eşleşen satırları değil, eşleşen dosyaların isimlerini yazdırır.
-n	(sayı)	Çıktıda eşleşen satırların numaralarını verir.
-r	(özyinelemeli)	alt dizinlerdeki dosyaları da arar.
-v	(ters çevir)	Sadece düzenli ifadeyle eşleşmeyen satırları yazdırır.

-f ("dosya") seçeneğiyle arama şablonu bir dosyadan okunabilir. Eğer verilen dosya birden çok satır içeriyorsa her satır kendi başına eşzamanlı olarak aranacak bir arama şablonu olarak değerlendirilecektir. Bu, özellikle sık kullanılan arama şablonlarının kullanımını oldukça basitleştirir. Yukarıda bahsedildiği gibi fgrep arama şablonu olarak düzenli ifadelerin kullanılmasına izin vermez. Öte yandan egrep düzenli ifadeler için çoğu eklentinin kullanımına izin verir (Tablo **??**).

Son olarak grep için bazı örnekler. frog.txt dosyası Grimm Kardeşler'in peri masalı Kurbağa Kral'ı içerir. (Ek **??**'ye bakın). frog karakter serisini içeren tüm satırlar en kolay şöyle bulunur.

```
$ grep frog frog.txt
Frog stretching forth its big, ugly head from the water. >>Ah,old
>>Be quiet, and do not weep,<< answered the frog, >> I can help you, but
>>Whatever you will have, dear frog,<<said she,>>My clothes,my pearls
```

Açıkça sadece "frog" kelimesini ("bullfrog" veya "frogspawn" gibi varyasyonlarını bulmayacak) aramak için kelime ayıracı eklentisi kullanılmalıdır:

```
$ grep \<frog\> frog.txt
frog stretching forth its big, ugly head from the water. >> Ah, old
```

(Bunun İngilizce olması farketmez. Dil ne olursa olsun "frog" geçen her satır gösterilir) "frog" ile *başlayan* satırları göstermek aşağıdaki kadar basittir:

```
$ grep ^frog  frog.txt
frog stretching forth its big, ugly head from the water. >> Ah, old
frog, that he had caused three iron bands to be laid roun his heart,
```

Farklı bir örnek: `/usr/share/dict/words` dosyası ingilizce kelimelerin bir listesini içerir (genellikle "sözlük" [2] olarak adlandırılır) Burada üç veya daha fazla "a" içeren tüm kelimeleri arıyoruz:

```
$ grep {n 'a.*a.*a' /usr/share/dict/words
8:aardvark
21:abaca
22:abacate
..(7030 kelime gizlendi)...
234831:zygomaticoauricularis
234832:zygomaticofacial
234834:zygomaticomaxillary
```

(sırasıyla: bir afrika hayvanı (orycteropus afer), iplik yapımında kullanılan muz ağacı (musa textilis), avokadonun Brezilyacası (perse sp.), bir yüz kası, iki tıbbi sıfat)

Daha karmaşık düzenli ifadelerle birlikte grep'in neden bir satırı yazdırıp diğerini yazdırmadığı çabucak belirsizleşebilir. Bir dosyadaki eşleşen parçaların farklı renkte gösterilmesini sağlayan --color seçeneği kullanılarak bu belirsizlik bir dereceye kadar azaltılabilir.

```
$ grep --color root /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

`export GREP\_OPTIONS='--color=auto'` benzeri bir komut (örneğin ∼/.profile dosyası içinde) bu seçeneği kalıcı olarak etkinleştirir; auto argümanı eğer çıktı bir boruya veya dosyaya gönderilirse renk çıktısını gizler.

Alıştırmalar

1. "?" ve "+" düzenli ifade operatörleri gerçekten gerekli midir?
2. `frog.txt` dosyası içinde "king" veya "king's daughter" kelimelerini içeren satırları bulun.
3. `/etc/passwd` içinde sistem üzerindeki kullanıcıların listesi vardır. Dosyanın her bir satırı ":" karakteri ile ayrılmış sıralı alanlardan oluşur. Her satırdaki son alan o kullanıcının giriş kabuğunu verir. Giriş kabuğu olarak bash kullanan bütün kullanıcıları listeleyen grep komut satırı yazın.
4. `/usr/share/dict/words` dosyası içinde şu 5 sesli harfi "a","e","i","o" ve "u" sırasıyla (önünde, aralarında ve sonunda sessiz harfler olabilir) içeren kelimeleri arayın.

[2] Sözlüğün boyutu kullanılan dağıtıma göre oldukça farklı olabilir

5. `frog.txt` dosyasında en az 4 harfli kelimelerin iki defa geçtiği satırları yazdıran bir komut verin.

Bu bölümdeki komutlar

egrep Belirli düzenli ifadelerle eşleşen satırlar için dosyaları arar. Genişletilmiş düzenli ifadelere izin verir.

fgrep Belirli içerikte satırlar için dosyaları arar. Düzenli ifadeler kullanılamaz.

grep Verilen düzenli ifadeyle eşleşen satırlar için dosyaları arar.

Özet

- Düzenli ifadeler karakter dizilerinin kümelerini tanımlamak için çok güçlü bir yöntemdir.
- `grep` ve ailesi düzenli ifadelerle eşleşen satırlar için bir dosyanın içeriğini arar.

Bölüm 8

Standart G/Ç ve Filtre Komutları

Amaçlar

- Kabuk G/Ç yönlendirmelerinde uzmanlaşmak
- En önemli filtre komutlarını öğrenmek

Önceden Bilinmesi Gerekenler

- Kabuk işlemleri
- Metin düzenleyicisi kullanımı (Bölüm **??**)
- Dosya ve dizin kullanma (Bölüm **??**)

8.1 G/Ç Yönlendirmeleri ve Komut Boru Hatları

8.1.1 Standart Kanallar

Çoğu Linux komutu - grep ve Bölüm **??**'deki arkadaşları gibi - girdi verilerini okumak, bir şekilde düzenlemek ve bu düzenlemenin sonucunu çıktı olarak vermek üzerine tasarlanmıştır. Örnek olarak, eğer aşağıdaki gibi bir komut girerseniz;

```
$ grep xyz
```

Klavyeden metin girdisi yapabilirsiniz ve grep sadece "xyz" harf grubunu içeren kısımları vurgulayacaktır.

```
$ grep xyz
abc def
xyz 123
xyz 123
aaa bbb
YYYxyzZZZ
YYYxyzZZZ
Ctrl + d
```

(Sondaki klavye tuş kombinasyonu grep'e girdi girişinin bittiğini haber verir.)

grep "standart girdiden" verileri okur – bu durumda klavyeden – ve bunu "standart çıktıya" yazar – bu koşulda konsol ekranına ya da yüksek ihtimalle görsel bir arayüzdeki bir terminal programına –. Bu standart yolların üçüncüsü "standart hata çıktısı"dır; grep' in ürettiği veriler standart çıktıya verilirken, standart hata çıktısı hata mesajlarını alır. (Örneğin, bulunmayan bir girdi dosyası ya da düzenli ifadelerdeki bir yazım hatası gibi)

Bu bölümde bir programın standart çıktısını nasıl bir dosyaya yönledirebileceğini ya da başka bir programın standart girdisini nasıl bir dosyadan alabileceğini öğreneceğiz. Daha da önemlisi, bir programın çıktısını direk olarak (dosyaya yönlendirmek gibi bir ara yöntem kullanmadan) nasıl başka bir dosyanın girdisi olarak kullanabileceğinizi öğreneceksiniz. Bu, tek başlarına düşünüldüklerinde hepsi oldukça basit olan Linux komutlarının birleşerek daha karmaşık yapılar oluşturmalarına olanak sağlar. (Bunu bir Lego seti gibi düşünün.)

Bu bölümde bu konuyu derinlemesine incelemeyeceğiz. Çok karmaşık kabuk betikleri oluşturmak için sıradaki rehberi, İleri Düzey Linux, okuyun, o rehberde Unix araçları kullanılarak kabuk için betikler oluşturmak anlatılmaktadır. Bu bölümde komut satırında temel Linux komutlarının zekice birleştirilmesini öğreneceksiniz.

Table 8.1: Linux'ta Standart Kanallar

Kanal	İsim	Kısaltma	Aygıt	Kullanım Alanı
0	standart girdi	stdin	klavye	Programların girdisi
1	standart çıktı	stdout	ekran	Programın çıktısı
2	standart hata cıktısı	stderr	ekran	Programın hata mesajları

Standart kanallar Tablo 8.1 de gösterilmiştir. Tabloda belirtilen stdin, stdout ve stderr standart girdi, standart çıktı ve standart hata çıktısı için olan teknik terimlerdir. Bu yollar daha sonra kullanacağımız gibi sırasıyla 0, 1 ve 2 numaralarına atanmışlardır.

Kabuk, bahsedilen program hiçbir şeyi fark etmeden, ayrı komutlar için bu standart kanalları yönlendirebilir. Çıktı ekranda ya da terminal penceresi dışında belirtilen bir dosyada yazıyor olabilir, ancak uygulamalar her zaman bu standart kanlları kullanır. Bu dosya başka bir cihaz olabileceği gibi (yazıcı

gibi) çıktıyı alacak metin dosyasını belirlemek mümkündür. Dosyanın o anda varolması bile gerekmez, gerektiği zaman kendiliğinden oluşturulur.

Standart girdi yolu da aynı şekilde yönlendirilebilir. Bir program girdisini klavyeden değil belirtilen bir dosyadan (bu başka bir aygıt ya da dosya olabilir) alır.

Klavye ve çalıştığınız terminal ekranı (ister Linux konsolu olsun, ister seri porttaki bir terminal, ister görsel arayüzde çalışan bir terminal penceresi, isterse de güvenli kabuk olarak bilinen ağdan bağlanılan bir oturum olsun) /dev/tty dosyası ile erişilebilir – veri okumak isteniyorsa bu klavye, cıktı vermek isteniyorsa bu ekran olarak kullanılır (tersi bayağı saçma olurdu). Şu komut:

```
$ grep xyz /dev/tty
```

Bu bölümde önceden verdiğimiz grep komutuna denktir. Bu konuyla ilgili daha fazla bilgiyi Bölüm **??**'da "özel dosyalar"da bulabilirsiniz.

8.1.2 Standart Kanalları Yönlendirmek

Standart çıktı kanalı ">" (büyüktür işareti) operatörü kullanılarak yönlendirilebilir. Aşağıdaki örnekte, "ls -laF" komutunun çıktısı filelist adı verilen bir dosyaya yönlendirilmektedir; ekran çıktısı şöyle gözükür:

```
$ ls -laF >filelist
$ _
```

Eğer filelist dosyası mevcut değilse oluşturulur. Ancak eğer bu isimde bir dosya varsa dosyanın üzerine yazılır. Kabuk bunu bahsedilen program çalıştırılmadan önce ayarlar – komut çağrısında yazım hataları varsa ya da program hiç çıktı oluşturmasa bile yine de çıktı dosyası oluşturulur. (bu durumda filelist dosyası boş olacaktır)

Eğer kabuk çıktı yönlendirmelerinde varolan dosyaların üzerine yazılmasını engellemek istiyorsanız, "set -o noclobber" komutu verilerek varolan dosyalar korunabilir. Bu durumda, eğer çıktı varolan bir dosyaya yönlendirilirse bir hata oluşacaktır.

Filelist dosyasına her zamanki yollarla bakabilirsiniz, mesela less kullanarak:

```
$ less filelist
total 7
drwxr-xr-x 12 joe users 1024 Aug 26 18:55 ./
drwxr-xr-x 5 root root 1024 Aug 13 12:52 ../
drwxr-xr-x 3 joe users 1024 Aug 20 12:30 photos/
-rw-r--r-- 1 joe users 0 Sep 6 13:50 filelist
-rw-r--r-- 1 joe users 15811 Aug 13 12:33 pingu.gif
-rw-r--r-- 1 joe users 14373 Aug 13 12:33 hobby.txt
-rw-r--r-- 2 joe users 3316 Aug 20 15:14 chemistry.txt
```

Eğer filelist dosyasının içeriğine dikkatlice bakarsanız, filelist adında boyutu 0 olan bir dizin girdisi görebilirsiniz. Bu kabuğun çalışma şeklinden kaynaklanır: Kabuk, komut satırını işlerken önce çıktı yönlendirmesini fark eder ve yeni bir filelist dosyası oluşturur (ya da varolan dosyanın içeriğini siler). Komutun (bu örnekte ls) standart çıktısını terminal yerine filelist dosyasına bağlayarak komutu çalıştırır.

ls çıktısındaki dosyanın uzunluğu 0 görünür çünkü ls, filelist dosyası ile ilgili bilgiye dosyaya hiçbir şey yazılmadan önce bakar, filelist satırından önce üç dosya olmasına rağmen. Bunun nedeni ls'nin önce bütün dizin girdilerini okuyup, bunları dosya adına göre sıralamasından ve ancak bunları tamamladıktan sonra dosyaya yazmaya başlamasıdır. Bu sebeple ls sadece yeni oluşturulmuş (ya da içi boşaltılmış) filelist dosyasını görür.

Eğer bir komutun çıktısını varolan bir dosyaya dosyanın önceki içeriğini değiştirmeden eklemek istiyorsanız >> operatörünü kullanabilirsiniz. Eğer dosya yoksa, bu durumda da oluşturulacaktır.

```
$ date >> filelist
$ less filelist
total 7
drwxr-xr-x 12 joe users 1024 Aug 26 18:55 ./
drwxr-xr-x 5 root root 1024 Aug 13 12:52 ../
drwxr-xr-x 3 joe users 1024 Aug 20 12:30 photos/
-rw-r--r-- 1 joe users 0 Sep 6 13:50 filelist
-rw-r--r-- 1 joe users 15811 Aug 13 12:33 pingu.gif
-rw-r--r-- 1 joe users 14373 Aug 13 12:33 hobby.txt
-rw-r--r-- 2 joe users 3316 Aug 20 15:14 chemistry.txt
Wed Oct 22 12:31:29 CEST 2003
```

Bu örnekte, on anki tarih ve saat filelist dosyasına ekleniyor.

Bir komutun standart çıktısını yönlendirmenin bir başka yolu geri imlerdir (‘...‘) Bu işlem **komut değiştirme** olarak da adlandırılır. Geri imlerin içindeki bir komutun standart çıktısı geri imler arasındaki komut (ve geri im işaretleri) ile değiştirilerek komut satırına eklenir ve değişiklikten sonraki komut çalıştırılır.

Örneğin:

```
$ cat dates
22/12 Get presents
23/12 Get Christmas tree
24/12 Christmas Eve
$ date +%d/%m
23/12
$ grep ‘date +%d/%m‘ dates
23/12 Get Christmas tree
```

” ‘date‘ “ yerine daha uygun bir yazım $(date) kullanmaktır. Bu yöntem, iç içe geçmiş bu tip değiştirme çağrılarını daha kolay yazmayı sağlar. Bununla birlikte bu söz dizimi sadece bash gibi modern kabuklar tarafından desteklenir.

Standart girdi kanalını yönlendirmek için < işaretini (küçüktür işareti) kullanabilirsiniz. Bu, klavye girdisi yerine belirtilen dosyanın içeriğini okuyacaktır.

```
$ wc -w <frog.txt
1397
```

Bu örnekte, wc filtre komutu frog.txt dosyasındaki kelimeleri sayar.

Birden fazla girdi dosyasını birleştirmek için << gibi bir yönlendirme bulunmaz, bunu yapmak için cat komutunu kullanmanız gerekir.

```
$ cat file1 file2 file3 | wc -w
```

("|" operatörü ile ilgili daha geniş bilgiyi gelecek bölümde anlatacağız.) Ancak çoğu program bir ya da birden fazla dosya ismini komut satırı girdisi olarak kabul eder.

Elbette, standart girdi ve standart çıktı aynı anda eş zamanlı olarak yönlendirilebilir. Aşağıdaki örnekte kelime sayacının çıktısı wordcount olarak adlandırılan bir dosyaya yazılır.

```
$ wc -w <frog.txt >wordcount
$ cat wordcount
1397
```

Standart girdi ve standart çıktı kanalları dışında ayrıca standart hata çıktısı kanalı bulunur. Eğer bir programın çalışmasında hatalar meydana gelirse, buna karşılık gelen mesajlar bu kanala yazılır. Böylece standart çıktıyı dosyaya yönlendirdiğiniz zaman bile hata mesajlarını görebilirsiniz. Eğer standart hata çıktısının da bir dosyaya yazılmasını istiyorsanız yönlendirme operatörünü kullanırken kanal numarasını belirtmelisiniz - stdin (0<) ve stdout (1>) için bu isteğe bağlı bir seçenektir fakat stderr için (2>) yazımını kullanmak zorunludur. >& operatörünü her iki çıktıyı da yazdırmak için kullanabilirsiniz.

```
make >make.log 2>&1
```

Bu komut make komutunun standart çıktısını ve standart hata çıktısını make.log dosyasına yönlendirir.

Dikkat Burada sıralama çok önemlidir. Aşağıdaki iki komut tümüyle farklı sonuçlar oluşturur. İkinci durumda standart hata çıktısı standart çıktı nereye giderse oraya yönlendirilecektir (/dev/tty, standart çıktının gittiği yer), sonra standart çıktı make.log'a gönderilir ama bu standart hata çıktısının gideceği yönü değiştirmez.

```
make >make.log 2>&1
make 2>&1 >make.log
```

Alıştırmalar

1. -U seçeneğini ls komutunun çıktısını sıralamadan vermesi için kullanabilirsiniz. "ls -laU >filelist " yazmanızdan sonra bile dosya içerisindeki çıktıda filelist dosyasının boyutu 0 olarak gözükür. Bunun nedeni ne olabilir?

2. "ls /tmp" ve "ls /tmp >ls-tmp.txt" komutlarının çıktılarını karşılaştırınız. Bir fark görüyor musunuz? Eğer fark varsa bunu nasıl açıklarsınız?

3. Neden bir dosyayı bir adımda yeni haliyle değiştirmek mümkün değildir? Mesela "grep xyz file > file" neden mümkün değildir?

4. "cat foo >>foo " komutunda foo'nun boş bir dosya olmadığını varsayarsak bu komuttaki sorun nedir?

5. Kabukta, bir hata mesajını nasıl standart hata çıktısı olarak alırsınız?

8.1.3 Komut Boru Hatları

Çıktı yönlendirmesi bir programın sonucunu kaydederek başka bir komutta işlemek için sıkça kullanılır. Bununla birlikte bu çeşit ara kayıtlar çok sıkıcı sayılmazlar ama işleri bittiği zaman silmeyi unutmamak gerekir. Bu sebeple, Linux komutları birbirine doğrudan borularla bağlama yöntemi sunar: Bir programın çıktısı otomatik olarak başka bir programın girdisi olur.

Birkaç komutun arasındaki yönlendirmeler |operatörü kullanılarak yapılır. "ls -laF" komutunun çıktısını bir dosyaya yönlendirip sonra bu dosyaya less kullanarak bakmak yerine, aynı işlemi tek adımda ara dosya kullanmadan yapabilirsiniz:

```
$ ls -laF | less
total 7
drwxr-xr-x 12 joe users 1024 Aug 26 18:55 ./
drwxr-xr-x 5 root root 1024 Aug 13 12:52 ../
drwxr-xr-x 3 joe users 1024 Aug 20 12:30 photos/
-rw-r--r-- 1 joe users 449 Sep 6 13:50 filelist
-rw-r--r-- 1 joe users 15811 Aug 13 12:33 pingu.gif
-rw-r--r-- 1 joe users 14373 Aug 13 12:33 hobby.txt
-rw-r--r-- 2 joe users 3316 Aug 20 15:14 chemistry.txt
```

Bu komut yönlendirmeleri neredeyse istenilen herhangi bir uzunlukta olabilir. Ayrıca en sonda oluşacak olan sonuç bir dosyaya yönlendirilebilir:

```
$ cut -d: -f1 /etc/passwd | sort | pr -2 >userlst
```

Bu komut yönlendirmesi bütün kullanıcı isimlerini /etc/passwd dosyasının : ile ayrılan ilk sütunundan alıp bunları alfabetik şekilde sıralayıp sonucu iki

sütun halinde userlst dosyasına yazar. Burada kullanılan komutlar bölümün devamında açıklanacaktır.

Bazen veri akışını bir komut yönlendirmesi içindeki ara bir noktada depolamak gerekebilir, örneğin bir devredeki ara sonuç başka görevler için kullanışlı olabilir. tee komutu veri akışını kopyalar ve bir kopyayı standart çıktıya diğerini ise bir dosyaya gönderir. Komut ismi tesisatçılıkta aynı şekilde kullanılan bir tür borudan esinlenmiştir (Tablo **??**'ye bakın).

Hiç seçenek belirtilmediğinde tee komutu belirtilen dosyayı oluşturur ya da böyle bir dosya varsa üzerine yazar; -a (ingilizcede birleştirme anlamına gelen "append") seçeneği kullanılarak varolan dosyaya ekleme yapılabilir.

```
$ ls -laF | tee list | less
total 7
drwxr-xr-x 12 joe users 1024 Aug 26 18:55 ./
drwxr-xr-x 5 root root 1024 Aug 13 12:52 ../
drwxr-xr-x 3 joe users 1024 Aug 20 12:30 photos/
-rw-r--r-- 1 joe users 449 Sep 6 13:50 content
-rw-r--r-- 1 joe users 15811 Aug 13 12:33 pingu.gif
-rw-r--r-- 1 joe users 14373 Aug 13 12:33 hobby.txt
-rw-r--r-- 2 joe users 3316 Aug 20 15:14 chemistry.txt
```

Bu örnekte içinde çalışılan dizinin içeriği hem list dosyasına hem de ekrana yazılır. (list dosyası ls komutunun çıktısında görünmez çünkü tee komutu dosyayı ls komutunun çalıştırılmasından sonra oluşturur.)

Alıştırmalar

1. Aynı ara sonucu birden fazla dosyaya aynı anda nasıl yazardınız?

8.2 Filtre Komutları

Unix' in - ve dolayısı ile Linux'un - temel fikirlerinden biri "araç takımı" ilkesidir. Sistem, her biri (kavramsal olarak) basit bir görevi yerine getiren çok sayıda sistem programı ile birlikte gelir. Bu programlar başka programları oluşturmak için "inşa blokları" olarak kullanılabilir ve böylece programcıları gerekli fonksiyonları kendi başlarına geliştirme derdinden kurtarır. Örnek olarak, programlar kendi sıralama yöntemini içermek yerine Linux'un onlara sağladığı sort komutunu kullanır. Bu modüler yapı birkaç avantaja sahiptir:

- Bütün zamanlarını yeni sıralama fonksiyonları üretmekle harcaması gerekmeyen programcıların hayatını kolaylaştırır.
- Eğer sort bir hata düzeltmesi veya performans iyileştirmesi geçirirse, sort kullanan bütün programlar bundan faydalanır – ve çoğu durumda programlarda herhangi bir değişiklik yapılması bile gerekmez.

Table 8.2: cat seçenekleri

Seçenek	Sonuç
-b	(ing. number non-blank lines) Çıktıdaki boş olmayan tüm satırları 1'den başlayarak numaralandırır.
-E	(ing. end-of-line) Her satırın sonunda $ işareti gösterir (Satır sonundaki boşluk karakterlerini yakalamak için faydalıdır).
-n	(ing. number) Çıktıdaki tüm satırları 1'den başlayarak numaralandırır.
-s	(ing. squeeze) Çıktıdaki art arda gelen boş satırları tek bir boş satır ile değiştirir.
-T	(ing. tabs) d "|".
-v	(ing. visible) "M-a".
-A	(ing. show all) -vET ile aynı.

Girdilerini standart girdiden alıp çıktılarını standart çıktıya veren araçlara "filtre komutları" ya da kısaca "filtreler" denir. Girdi yönlendirmesi olmazsa filtre komutu, girdisini klavyeden okuyabilir. Klavye girişinin bittiğini belirtmek için terminal tarafından "dosya sonu" olarak algılanılan + d tuş kombinasyonunu kullanmalısınız.

Bunun sadece klavye girdisi için geçerli olduğunu unutmayın. Diskteki dosya + d karakterini (ASCII 4) içerebilir ve sistem bunun dosyanın sonu olduğunu düşünmez. Bu davranış Ctrl-Z (ASCII 26) karakterinin bir metin dosyası içinde bile olsa garip anlam içerdiği oldukça popüler bir işletim sisteminin davranışının tersidir.

"Normal" komutların çoğu, önceden belirtilmiş olan grep gibi, eğer üzerinde çalışacakları bir dosya ismi belirtilmezse filtreler gibi çalışır.

Bu bölümün geri kalanında bu şekilde çalışan çoğu önemli komuta aşina hale geleceksiniz. Bazı komutlar tam olarak filtre komutları sayılmasa da, boru hatları için önemli bloklar oluşturmakta kullanılırlar.

8.3 Dosyaları Okuma ve Yazma

8.3.1 Metin Dosyalarının Çıktıları ve Birleştirme - cat

cat (ing. "concatenate": bitiştirmek) komutu gerçekte komut satırından ismi verilen birden fazla dosyayı tek bir dosyada birleştirmek için kullanılır. Eğer sadece bir dosya ismi belirtirseniz, bu dosyanın içeriği standart çıktıya yazılacaktır. Eğer hiç dosya ismi belirtmezseniz cat standart girdiyi okur - bu kullanışsız görünebilir ama cat satırları numaralandırma, satırları sonlarını ve

özel karakterleri görünür yapma ya da birden fazla boş satırı teke indirme gibi seçenekler sunar. (Tablo **??**)

Söylemeye gerek olmasa bile belirtmekte yarar var, cat sadece metin dosyalarında anlaşılır sonuçlar ortaya çıkarır. Eğer bu komutu diğer çeşit dosyalar için kullanırsanız (mesela /bin/cat gibi bir ikili dosya için) çıktı tamamlandığında kabuk iletisinin - en azından bir metin terminalinde - okunamayan karakterlerden oluşması kesin gibidir. Bu durumda normal karakter kümesini (çıktıyı görmeden yazmak zorunda kalabilirsiniz) **reset** komutu ile geri getirebilirsiniz. Eğer cat çıktısını bir dosyaya yönlendirirseniz bu sorunla tabii ki karşılaşmazsınız.

"En Kullanışsız cat Kullanımı Ödülü" cat'i gereksiz yere kullanan insanlara verilmelidir. Çoğu durumda komutlar sadece standart girdiyi okumakla kalmaz, dosya isimlerini de kabul eder, bu sebeple cat tek bir dosyayı standart girdiden bu komutlara göndermek için gerekli değildir. "cat data.txt |grep foo" gibi bir komut gereksizdir çünkü bunun yerine basitçe "grep foo data.txt" yazabilirsiniz. Hatta grep sadece standart girdiden okuyabiliyor olsaydı bile "grep foo <data.txt " daha kısa olurdu ve ayrıca cat sürecine ihtiyaç olmazdı.

Alıştırmalar

1. Bir dizinin "garip" isimlere sahip dosyalara sahip olduğunu nasıl kontrol edebilirsiniz (örneğin isminin sonunda bir boşluk karakteri olan ya da ortasında görünmez kontrol karakteri bulunduranlar gibi)?

8.3.2 Başlangıç ve Bitiş - head ve tail

Bazen dosyanın sadece bir parçasıyla ilgilenirsiniz: Doğru dosya olup olmadığını görmek için ilk birkaç satır ya da özellikle kayıt dosyalarında son birkaç satır. Head ve tail komutları tam olarak bunu verirler - varsayılan olarak argümanlar arasında verilen dosyaların her birinin ilk on ve son on satırını (veya alışıldığı gibi standart girdilerinin ilk veya son 10 satırını) gösterirler. -n seçeneği gösterilecek satır sayısını değiştirmenizi sağlar: "head -n 20" ilk yirmi satırı değer olarak döndürürken "tail -n 5 data.txt" data.txt dosyasının son beş satırını döndürür.

Geleneksel olarak istediğimiz n sayıdaki satırı direk olarak "-n" şeklinde belirtebiliz. Resmi olarak bu yönteme artık izin verilmiyor ancak head ve tail' in Linux versiyonları hala bu özelliği destekliyorlar.

-c seçeneğini sayma işleminin satır olarak değil byte olarak yapılması için kullanabilirsiniz: "head -c 20" standart girdinin ne kadar satıra sahip olursa olsun ilk 20 bytelık kısmını gösterir. Eğer "b", "k", ya da "m" harflerini eklerseniz (sırasıyla bloklar, kibibytelar ve mebibytelar) sayım 512, 1024 ya da 1048576 ile çarpılarak yapılır.

head ayrıca eksi işaretini kullanmanıza izin verir: "head -c -20 " standart girdinin son 20 bytelık kısmı hariç tümünü gösterir.

head' in bu münasebetsizliğine karşılık, tail de head'in desteklemediği bir şey yapar: eğer satır numaraları "+" ile başlıyorsa, verilen satırdan itibaren bütün verileri görüntüler.

```
$ tail -n +3 file           satır 3 ve sonrasındakiler
```

tail komutu ayrıca önemli olan -f seçeneğini destekler. Bu seçenek tail komutunun dosyanın o anki sonunu yazdırmasının ardından dosyaya daha sonra eklenecek olan satırları da yazdırması için beklemesini sağlar. Bu yöntem bazı kayıt dosyalarını gözetlemek istediğiniz zaman çok işe yarar. Eğer tail -f komutuna birden fazla dosya ismi verirseniz, her çıktı satırı bloğu için hangi dosyaya yeni veri eklendiğini gösteren bir başlık satırı koyar.

Alıştırmalar

1. Standart girdinin sadece 13. satırını nasıl çıktı olarak alabilirsiniz?

2. "tail -f" komutunu deneyin: Bir dosya oluşturun ve bu dosya üzerinde "tail -f" komutunu çalıştırın. Sonra başka bir pencereden ya da sanal bir konsoldan dosyaya bir şeyler ekleyin (örneğin echo >> ... kullanarak) ve tail komutunun çıktısını gözlemleyin. Tail birden fazla dosyayı aynı anda eşzamanlı olarak izlediğinde nasıl bir çıktı veriyor?

3. İzlenen dosya küçülürse "tail -f" komutuna ne olur?

4. Aşağıdaki komutların çıktısını açıklayın:

   ```
   $ echo Hello >/tmp/hello
   $ echo "Hiya World" >/tmp/hello
   ```

   İlk echo komutundan sonra aşağıdaki komutu başka bir pencerede çalıştırdığınızda ne olur?

   ```
   $ tail -f /tmp/hello
   ```

8.4 Veri Yönetimi

8.4.1 Sıralı Dosyalar - sort ve uniq

sort komutu metin dosyaları içindeki satırları önceden belirtilmiş bir kritere göre sıralamaya yarar. Varsayılan ayar her satırın ilk birkaç karakterinin

ASCII karakter seti değerlerine[1] göre artan şekilde (A-Z) sıralanmasıdır. Almancadaki üzeri çift noktalı karakterlerin sıklıkla hatalı sıralanmasının sebebi budur. Örneğin, "Ä" nın karakter kodu 143'tür, bu yüzden sıralamada karakter kodu 91 olan "Z" karakterinden sonra gelir. Küçük "a" karakteri bile büyük "Z" karakterinden "daha büyük" sayılır.

Elbette sort değişik dillere ve kültürlere göre davranabilir. Alman kurallarına göre sıralama yapmak için LANG, LC_ALL ya da LC_COLLATE çevre değişkenlerinden birini "de", "de_DE" ya da "de_DE@UTF-8" (seçiminiz kullandığınız dağıtıma göre değişir) olarak değiştirebilirsiniz. Eğer kullandığınız sıralama kurallarını sadece bu defaya mahsus değiştirmek istiyorsanız aşağıdaki şekilde kullanabilirsiniz:

```
$ ... | LC_COLLATE=de_DE.UTF-8 sort
```

LC_ALL değeri LC_COLLATE üzerinde üstündür. Aynı şekilde LC_COLLATE de LANG değerinden üstündür. Ayrıca Alman sıralama düzeni kullanmanın bir yan etkisi de sıralama yaparken harflerin boyutunun göz önüne alınmamasıdır.

Aksini belirtmezseniz, sıralama bütün girdi satırı göz önüne alınarak alfabetik göre yapılır. Yani iki satırın ilk karakterleri eşitse satırdaki ilk değişiklik gösteren karakter sıralamayı belirler. Elbette sort sadece bütün satıra göre değil, özelleştirilmiş bir şekilde belirli sütunlara ya da (hayali) bir tablonun alanlarına göre sıralama yapabilir. Alanlar 1'den başlayarak numaralandırılır, "-k 2" ile sıralama için ilk alan görmezden gelinerek her satırın ikinci alanı dikkate alınır. Eğer iki satırın ikinci alanları eşitse ve "-k 2,3" şeklinde daha özelleştirilmiş bir seçenek belirlenmediyse satırın kalanına bakılır. Ayrıca aynı sort komutunda birden fazla -k seçeneği belirtilebilir.

Bunlara ek olarak, sort artık kullanılmayan eski bir sıralama belirtecini de destekler: Alanlar sıfırdan başlayarak numaralandırıldığında ilk alan "+m" ve son alan "-n" olarak belirtilir. Ayrıca son alan açıkça belirtilmelidir. Üstteki örnek "+1 ", "+1 -3 " ve "+1 -2 " şeklinde yazılabilir.

Boşluk karakteri alanlar arasında ayırıcı işlevi görür. Eğer art arda birden fazla boşluk varsa sadece ilki ayırıcı olarak değerlendirilir; geri kalanlar takip eden alana ait değerler olarak kabul edilir. Örnek olarak Lameborough Track and Field Club'ın düzenlediği yıllık maratonun katılımcılarının isimlerinden oluşan bir listeyi ele alalım. Başlangıç olarak ilgili çevre değişkenlerini sıfırlayarak sistemin standart dil ortamının ("POSIX") kullanıldığından emin oluruz. (4. sütun koşucunun forma numarasıdır.)

```
$ unset LANG LC_ALL LC_COLLATE
$ cat participants.dat
Smith Herbert Pantington AC 123 Men
```

[1] Elbette ASCII sadece 128 karakter içerir. Burada aslında ASCII ile beraber 128'den sonraki karakter kodları için kullanılmakta olan uzantı (örn: ISO-8859-1 veya diğer adıyla ISO-Latin-1) kastedilmektedir.

```
Prowler Desmond Lameborough TFC 13 Men
Fleetman Fred Rundale Sportsters 217 Men
Jumpabout Mike Fairing Track Society 154 Men
de Leaping Gwen Fairing Track Society 26 Ladies
Runnington Vivian Lameborough TFC 117 Ladies
Sweat Susan Rundale Sportsters 93 Ladies
Runnington Kathleen Lameborough TFC 119 Ladies
Longshanks Loretta Pantington AC 55 Ladies
O'Finnan Jack Fairing Track Society 45 Men
Oblomovsky Katie Rundale Sportsters 57 Ladies
```

Önce soyadına göre sıralamayı deneyelim. Bu aslında kolay bir örnek çünkü soyadı kısmı zaten her satırın başında:

```
$ sort participants.dat
Fleetman Fred Rundale Sportsters 217 Men
Jumpabout Mike Fairing Track Society 154 Men
Longshanks Loretta Pantington AC 55 Ladies
O'Finnan Jack Fairing Track Society 45 Men
Oblomovsky Katie Rundale Sportsters 57 Ladies
Prowler Desmond Lameborough TFC 13 Men
Runnington Kathleen Lameborough TFC 119 Ladies
Runnington Vivian Lameborough TFC 117 Ladies
Smith Herbert Pantington AC 123 Men
Sweat Susan Rundale Sportsters 93 Ladies
de Leaping Gwen Fairing Track Society 26 Ladies
```

Listedeki iki küçük sorunu fark etmiş olmalısınız: "Oblomovsky" "O'Finnan" ın önünde olmalıydı ve "de Leaping" listenin sonunda değil başında yer almalıydı. Eğer sıralama kuralları olarak "İngilizce"yi belirtirsek bu sorunlar kaybolur:

```
$ LC_COLLATE=en_GB sort participants.dat
de Leaping Gwen Fairing Track Society 26 Ladies
Fleetman Fred Rundale Sportsters 217 Men
Jumpabout Mike Fairing Track Society 154 Men
Longshanks Loretta Pantington AC 55 Ladies
Oblomovsky Katie Rundale Sportsters 57 Ladies
O'Finnan Jack Fairing Track Society 45 Men
Prowler Desmond Lameborough TFC 13 Men
Runnington Kathleen Lameborough TFC 119 Ladies
Runnington Vivian Lameborough TFC 117 Ladies
Smith Herbert Pantington AC 123 Men
Sweat Susan Rundale Sportsters 93 Ladies
```

(en_GB "İngiliz İngilizcesi" için, en_US ise "Amerikan İngilizcesi" için kullanılan kısaltmadır ve her ikisi de burada aynı işi görür) Şimdi ilk isme göre sıralama yapalım:

```
$ sort -k 2,2 participants.dat
Smith Herbert Pantington AC 123 Men
```

```
Sweat Susan Rundale Sportsters 93 Ladies
Prowler Desmond Lameborough TFC 13 Men
Fleetman Fred Rundale Sportsters 217 Men
O'Finnan Jack Fairing Track Society 45 Men
Jumpabout Mike Fairing Track Society 154 Men
Runnington Kathleen Lameborough TFC 119 Ladies
Oblomovsky Katie Rundale Sportsters 57 Ladies
de Leaping Gwen Fairing Track Society 26 Ladies
Longshanks Loretta Pantington AC 55 Ladies
Runnington Vivian Lameborough TFC 117 Ladies
```

Bu örnek sort'un yukarıda anlatılan özelliğini gösterir: İlk tanımlanan boşluk ayırıcı olarak kabul edilir, diğerleri takip eden alanın değerlerini oluşturur. Gördüğünüz gibi ilk isimlerden sadece soyisimleri aynı uzunlukta olanlar alfabetik olarak listelendi. Bu durum -b seçeneği kullanılarak düzeltilebilir, bu seçenek art arda gelen boşluk karakterlerini tek bir boşluk olarak değerlendirir.

```
$ sort -b -k 2,2 participants.dat
Prowler Desmond Lameborough TFC 13 Men
Fleetman Fred Rundale Sportsters 217 Men
Smith Herbert Pantington AC 123 Men
O'Finnan Jack Fairing Track Society 45 Men
Runnington Kathleen Lameborough TFC 119 Ladies
Oblomovsky Katie Rundale Sportsters 57 Ladies
de Leaping Gwen Fairing Track Society 26 Ladies
Longshanks Loretta Pantington AC 55 Ladies
Jumpabout Mike Fairing Track Society 154 Men
Sweat Susan Rundale Sportsters 93 Ladies
Runnington Vivian Lameborough TFC 117 Ladies
```

Bu sıralı liste hala biraz hatalı; bunun için 8.14 numaralı alıştırmaya bakın.

Aşağıdaki örnekte de görüldüğü gibi sıralama alanı daha detaylı bir şekilde belirtilebilir:

```
$ sort -br -k 2.2 participants.dat
Sweat Susan Rundale Sportsters 93 Ladies
Fleetman Fred Rundale Sportsters 217 Men
Longshanks Loretta Pantington AC 55 Ladies
Runnington Vivian Lameborough TFC 117 Ladies
Jumpabout Mike Fairing Track Society 154 Men
Prowler Desmond Lameborough TFC 13 Men
Smith Herbert Pantington AC 123 Men
de Leaping Gwen Fairing Track Society 26 Ladies
Oblomovsky Katie Rundale Sportsters 57 Ladies
Runnington Kathleen Lameborough TFC 119 Ladies
O'Finnan Jack Fairing Track Society 45 Men
```

Burada participants.dat dosyası ikinci alanın ikinci karakterine göre, yani ilk adlarının ikinci karakterine göre (çok anlamlı!) azalan şekilde (-r) sıralanmıştır. Bu örnekte de başta gelen boşluklar -b seçeneği kullanılarak göz ardı edilmelidir. (Örnek 8.14 teki hata burada da kendini tekrar ediyor.)

-t seçeneği ile ("terminate", Türkçesi "durdur") alan ayırıcısının olarak istenen bir karakteri seçebilirsiniz. Bu seçenek alanlar zaten veri olarak boşluklar içerebileceği için iyi bir fikir olabilir. Aşağıda örnek dosyamızın için daha kullanışlı (ancak daha zor okunan) bir hali bulunuyor:

```
Smith:Herbert:Pantington AC:123:Men
Prowler:Desmond:Lameborough TFC:13:Men
Fleetman:Fred:Rundale Sportsters:217:Men
Jumpabout:Mike:Fairing Track Society:154:Men
de Leaping:Gwen:Fairing Track Society:26:Ladies
Runnington:Vivian:Lameborough TFC:117:Ladies
Sweat:Susan:Rundale Sportsters:93:Ladies
Runnington:Kathleen:Lameborough TFC:119:Ladies
Longshanks:Loretta: Pantington AC:55:Ladies
O'Finnan:Jack:Fairing Track Society:45:Men
Oblomovsky:Katie:Rundale Sportsters:57:Ladies
```

"LC_COLLATE=en_GB sort -t: -k2,2" kullanılarak ilk isme göre sıralamak artık doğru sonucu veriyor. Ayrıca katılımcının numarasına göre sıralamak (4 numaralı alan) kulüp isminde kaç tane boşluk olursa olsun daha kolaydır:

```
$ sort -t: -k4 participants0.dat
Runnington:Vivian:Lameborough TFC:117:Ladies
Runnington:Kathleen:Lameborough TFC:119:Ladies
Smith:Herbert:Pantington AC:123:Men
Prowler:Desmond:Lameborough TFC:13:Men
Jumpabout:Mike:Fairing Track Society:154:Men
Fleetman:Fred:Rundale Sportsters:217:Men
de Leaping:Gwen:Fairing Track Society:26:Ladies
O'Finnan:Jack:Fairing Track Society:45:Men
Longshanks:Loretta: Pantington AC:55:Ladies
Oblomovsky:Katie:Rundale Sportsters:57:Ladies
Sweat:Susan:Rundale Sportsters:93:Ladies
```

Elbette "sayı" sıralaması aksi belirtilmedikçe alfabetik olarak yapılır – "117" ve "123" "13" ten önce, o da "154"den önce gelir. Bu durum sayısal karşılaştırma yapmaya zorlayan -n seçeneği ile düzeltilebilir.

```
$ sort -t: -k4 -n participants0.dat
Prowler:Desmond:Lameborough TFC:13:Men
de Leaping:Gwen:Fairing Track Society:26:Ladies
O'Finnan:Jack:Fairing Track Society:45:Men
Longshanks:Loretta: Pantington AC:55:Ladies
Oblomovsky:Katie:Rundale Sportsters:57:Ladies
Sweat:Susan:Rundale Sportsters:93:Ladies
Runnington:Vivian:Lameborough TFC:117:Ladies
Runnington:Kathleen:Lameborough TFC:119:Ladies
Smith:Herbert:Pantington AC:123:Men
Jumpabout:Mike:Fairing Track Society:154:Men
Fleetman:Fred:Rundale Sportsters:217:Men
```

Table 8.3: sort seçenekleri

Seçenek		Sonuç
-b	(boşluk, ing. blank)	Alanların içindeki başta gelen boşluk karakterlerini dikkate almaz.
-d	(sözlük, ing. dictionary)	Sözlük sırasına göre sıralar, örn. sadece harf, rakam ve boşluk karakterleri dikkate alınır.
-f	(katla, ing. fold)	Büyük ve küçük harfleri eşit olarak değerlendirir.
-i	(yoksay, ing. ignore)	Yazdırılamayan karakterleri dikkate almaz.
-k alan1 [, alan2]	(anahtar, ing. key)	alan1'den başlayıp alan2'ye kadar (dahil) sıralar.
-n	(sayısal, ing. numeric)	Alanın içeriğini sayı olarak kabul eder ve sayısal değerine göre sıralar, başta gelen boşluklar dikkate alınmaz.
-o	(çıktı, ing. output)	Sonuçları dosyaya yazdır, dosya ismi orjinal girdi dosyası ile aynı olabilir.
-r	(ters, ing. reverse)	Azalan şekilde sırala, örn: Z'den A'ya doğru
-t <karakter>	(durdur, ing. terminate)	Alan ayıracı olarak karakteri kullan.
-u	(Tekil, ing. unique)	Art arda gelen aynı çıktı satırlarından sadece ilkini yaz.

sort'un bazı önemli seçenekleri Tablo **??**'te gösterilmiştir, ayrıca programın belgelerini incelemek de faydalı olabilir. Sort size çok zaman kazandırabilecek çok yönlü ve güçlü bir komuttur.

uniq komutu girdide verilen art arda gelen aynı satırların sadece ilkini (veya tercihinize göre sonuncusunu) yazdırmak gibi önemli bir işlev sağlar. Neyin "aynı" olarak kabul edileceği her zamanki gibi özel seçenekler kullanılarak değiştirilebilir. Uniq gördüğümüz komutların çoğundan farklıdır, bir tane dışında isteğe bağlı girdi dosyası kabul etmez; eğer ikinci bir dosya ismi verilirse bunu çıktının yazdırılacağı dosya olarak farz eder (eğer ikinci dosya ismi verilmezse çıktı standart çıktıya yazılır). uniq çağrısında hiç dosya ismi verilmemişse, beklendiği gibi standart girdiden okumaya başlar.

Uniq girdi dosyasındaki tüm eşit satırlar art arda sıralanmış olduğunda en iyi şekilde çalışır. Eğer durum bu değilse her bir satırın çıktıda sadece bir kez görüneceği kesin olmaz.

```
$ cat uniq-test
Hipp
```

```
Hopp
Hopp
Hipp
Hipp
Hopp
$ uniq uniq-test
Hipp
Hopp
Hipp
Hopp
```

Bunu "sort -u" komutunun çıktısıyla karşılaştırın:

```
$ sort -u uniq-test
Hipp
Hopp
```

Alıştırmalar

1. participants0.dat dosyasındaki (alanların ":" ile ayrıldıkları dosya) katılımcı listesini klüp isimlerine ve her klüpteki oyuncuları da soyad ve ilk adlarına göre (bu sıra ile) sıralayın.

2. Katılımcı listesini klüp isimlerine göre artan bir şekilde (a-z) klüplerin oyuncularını da numaralarına göre azalan şekilde nasıl sıralarız? (İpucu: Sort programının belgelerini inceleyin!)

3. Örneklerde sürekli iddia ettiğimiz hata nedir ve hangi sebepten dolayı oluşur?

4. Aşağıdaki dosya isimlerine sahip bir dizin:

   ```
   01-2002.txt 01-2003.txt 02-2002.txt 02-2003.txt
   03-2002.txt 03-2003.txt 04-2002.txt 04-2003.txt
   <<<
   11-2002.txt 11-2003.txt 12-2002.txt 12-2003.txt
   ```

   ls komutunun çıktısını tarihsel sıraya göre doğru dizen sort komutunu yazın:

   ```
   01-2002.txt
   02-2002.txt
   <<<
   12-2002.txt
   01-2003.txt
   <<<
   12-2003.txt
   ```

8.4.2 Sütunlar ve Alanlar - cut, paste vb.

Grep ile bir metin dosyası içindeki satırları bulup kesebilirken cut komutu bir metin dosyasında sütun yönelikli çalışır. İki farklı çalışma yöntemi vardır:

Birinci ihtimal sütunların kesin bir şekilde ayrılmış olmasıdır. Bu sütunlar bir satırda tek bir karaktere karşılık gelirler. Bu gibi sütunları kesmek için -c (ing. column) seçeneği ile sütun numarası verilmelidir. Birden fazla sütunu tek seferde kesmek için sütun numaraları virgül ile ayrılır. Sütun aralıkları bile belirtilebilir.

```
$ cut -c 12,1-5 participants.dat
SmithH
ProwlD
FleetF
JumpaM
de LeG
```

Bu örnekte ilk ismin ilk karakteri ve son ismin beş karakteri ayrılır. Ayrıca çıktının her zaman için girdideki sütunlar ile aynı sıraya göre gerçekleştiğini görebiliriz. Eğer seçilmiş sütun sıraları birbirleri üstüne gelse bile, her girdi karakteri sadece bir defa yazdırılır:

```
$ cut -c 1-5,2-6,3-7 participants.dat
Smith
Prowler
Fleetma
Jumpabo
de Leap
```

İkinci yöntem bir ayraç karakteri ile ayrılmış göreceli alanları kesmektir. Eğer bu göreceli alanları kesmek istiyorsanız cut, -f (ing. field) seçeneğine ve istenen alan numarasına ihtiyaç duyar. Sütunlar için geçerli olan kuralların hepsi alanlar içinde geçerlidir. -c ve -f seçenekleri birbirine karşılıklıdır, aynı anda her ikisini birden kullanamazsınız.

Varsayılan ayırıcı tab karakteridir, diğer ayırıcılar -d (ing. delimiter) seçeneği ile belirtilebilir:

```
$ cut -d: -f 1,4 participants0.dat
Smith:123
Prowler:13
Fleetman:217
Jumpabout:154
de Leaping:26
```

Bu yolla katılımcıların soyadları (birinci sütun) ve numaraları (dördüncü sütun) listeden alınır. Yukarıdaki örnekte okunabilirlik için çıktının sadece ilk birkaç satırı gösterilmektedir.

Sözü açılmışken, –output-delimeter seçeneğini kullanarak çıktı alanları için girdi ayıracından farklı bir ayırıcı belirleyebilirsiniz:

```
$ cut -d: --output-delimiter=': ' -f 1,4 participants0.dat
Smith: 123
Prowler: 13
Fleetman: 217
Jumpabout: 154
de Leaping: 26
```

Eğer gerçekten sütun ya da alanların sırasını değiştirmek istiyorsanız, awk ve perl gibi büyük silahları ortaya çıkarmak zorundasınız. Bunu şimdi anlatacağımız paste komutu ile yapabilirsiniz ama bu biraz can sıkıcı olabilir.

Dosyalar sütunlar yerine alanlara göre işlenecekse -s (ing. seperator, ayırıcı) seçeneği kullanışlı olur. "cut -f" ayırıcı karakterini içermeyen satırları bulduğunda çıktıya bunları bütün halinde verir; -s bu satırları gizler.

Paste komutu belirtilen dosyalardaki satırları birleştirir, bu nedenle sıklıkla cut komutu ile birlikte kullanılır. Hemen fark edeceğiniz gibi paste bir filtre komutu değildir. Yine de dosya adı gelmesi gereken yere "-" işaretini koyarak paste komutunun dosya yerine standart girdiden okumasını sağlayabilirsiniz. Komutun çıktısı daima standart çıktıya gönderilir.

Söylediğimiz gibi, paste satır satır çalışır. Eğer iki dosya adı belirtildiyse ilk dosyanın ilk satırı ve ikincinin ilk satırı (bir tab karakterini ayırıcı olarak kullanarak) çıktının ilk satırı olacak şekilde birleştirilir. Aynı işlem dosyadaki bütün satırlar için tekrarlanır. Başka bir ayırıcı kullanmak için -d seçeneğini kullanabilirsiniz.

Örnek olarak; maraton koşucuları listesini katılım numaraları öne gelecek birleştirip yeni bir dosya oluşturabiliriz:

```
$ cut -d: -f4 participants0.dat >number.dat
$ cut -d: -f1-3,5 participants0.dat \
>
| paste -d: number.dat - >p-number.dat
$ cat p-number.dat
123:Smith:Herbert:Pantington AC:Men
13:Prowler:Desmond:Lameborough TFC:Men
217:Fleetman:Fred:Rundale Sportsters:Men
154:Jumpabout:Mike:Fairing Track Society:Men
26:de Leaping:Gwen:Fairing Track Society:Ladies
117:Runnington:Vivian:Lameborough TFC:Ladies
93:Sweat:Susan:Rundale Sportsters:Ladies
119:Runnington:Kathleen:Lameborough TFC:Ladies
55:Longshanks:Loretta: Pantington AC:Ladies
45:O'Finnan:Jack:Fairing Track Society:Men
57:Oblomovsky:Katie:Rundale Sportsters:Ladies
```

Bu dosya "sort -n p-number.dat" kullanılarak numaralara göre düzgn bir şekilde sıralanabilir.

-s (seri, ing. serial) ile verilen dosyalar sırayla işlenir. İlk olarak, ilk dosyanın bütün satırları (aralarında bir ayırıcı karakter bulunacak şekilde)

tek bir satır haline getirilir ve sonra ikinci dosyadaki bütün satırlar çıktının ikinci satırını oluştururlar.

```
$ cat list1
Wood
Bell
Potter
$ cat list2
Keeper
Chaser
Seeker
$ paste -s list*
Wood Bell Potter
Keeper Chaser Seeker
```

list* arama karakteriyle eşleşen bütün dosyalar - bu durumda list1 ve list2 - paste kullanılarak birleştirildi. -s seçeneği bu dosyaların tüm satılarlarının çıktının bir sütununu oluşturmasını sağlar.

Alıştırmalar

1. participants.dat dosyasının (eşit uzunlukta sütunlar halinde olanının) katılımcı numarası ve kulüplerinin olmadığı bir halini oluşturunuz.
2. participants0.dat dosyasının (: işaretinin ayırıcı olarak kullanıldığı halinin) katılımcı numarası ve kulüplerinin olmadığı bir halini oluşturunuz.
3. Katılımcılar0.dat dosyasının alanlarının ":" ile değil ",␣" (virgül ve boşluk işareti) ile ayrıldığı bir halini oluşturunuz.
4. Sisteminizde kaç tane grup kullanıcılar tarafından birincil grup olarak kullanılıyor? (Kullanıcıların birincil grupları /etc/passwd dosyasındaki dördüncü alandadır.)

Bu Bölümdeki Komutlar

cat Dosyaları birleştirir (diğer işlevlerin yanında)

cut Girdisindeki sütun ya da alanları ayırır

head Bir dosyanın başlangıcını gösterir

paste Farklı girdi dosyalarındaki satırları birleştirir

reset Bir terminalin karakter kümesini mantıklı bir değere sıfırlar

sort Girdisini satır satır sıralar

tail Bir dosyanın sonunu gösterir

uniq Bir dosyadaki art arda gelen eş satırları teke indirir

Özet

- Her Linux programı standart G/Ç yollarını, stdin, stdout ve stderr'i destekler.
- Standart çıktı ve standart hata çıktısı > ve >> operatörleri ile, standart girdi < operatörü ile yönlendirilebilir.
- Boru hatları komutların standart çıktı ve girdilerini (ara dosyalar olmadan) direkt olarak bağlamak için kullanılabilir.
- tee komutunu kullanarak yönlendirmelerdeki ara sonuçları dosyalarda saklayabiliriz.
- Filtre komutları (ya da filtreler) kendi standart girdilerini okuyup değiştirip sonucu standart çıktıya yazarlar.
- sort sıralama için kullanılan çok yönlü bir programdır.
- cut komutu girdideki her satırdan belirtilen aralıktaki sütun ya da alanları ayırır.
- paste ile dosyaların satırları birleştirilebilir.

Bölüm 9

Kabuk Hakkında Daha Fazla Bilgi

Amaçlar

- Kabuk ve ortam değişkenleri hakkında bilgi edinmek

Önceden Bilinmesi Gerekenler

- Temel kabuk bilgisi (Bölüm **??**)
- Dosya yönetimi ve basit komutlar (Bölüm **??**, Bölüm **??**)
- Metin editörü kullanımı (Bölüm **??**)

9.1 Basit Komutlar: sleep, echo ve date

Deneyim kazanmak için bazı araçları temel komutlarla açıklayacağız:

sleep Bu komut argüman olarak belirlenen saniye süresince hiçbir şey yapmaz. Eğer kullandığınız kabuk biraz ara versin istiyorsanız bu komutu kullanabilirsiniz.

```
$ sleep 10
       Yaklaşık 10 saniye hiçbir şey olmaz
$ _
```

echo Bu komut argümanların çıktılarını verir, argümanlar boşluklarla ayrılmış olmalıdır. Kabuk değişkenlerin referanslarını değiştirebildiğinden beri ilginç ve kullanışlıdır (bkz. Bölüm **??**). Bu konuya benzer bir örnek:

```
$ p=Planet
$ echo Hello $p
Hello Planet
$ echo Hello ${p}oid
Hello Planetoid
```

(Doğrudan bir değişkenin değerine bir şey eklenmek istenirse ne yapılacağı ikinci örnekle açıklanmıştır.)

Eğer echo komutu –n seçeneği ile çağırılırsa o satır sonlandırıcıdır, sonraki çıktıyı yazmaz.

```
$ echo -n Hello
Hello_
```

date Date komutu o anki tarih ve saati gösterir. "date –help" komutu çağırarak çıktının biçimini önemli ölçüde belirleyebilir veya "man date" komutunu kullanarak çevrimiçi belgeler okuyabilirsiniz.

(İkinci kez bu kılavuzu okurken:) Özellikle önemli şehir veya zaman dilimi ismini TZ çevre değişkeniyle ayarlarsanız tarih dünya saati olarak hizmet vermektedir (genellikle başkent).

```
$ date
Thu Oct 5 14:26:07 CEST 2006
$ export TZ=Asia/Tokyo
$ date
Tue Oct 5 21:26:19 JST 2006
$ unset TZ
```

/usr/share/zoneinfo çevresinde köklenen bilgiyle geçerli şehir isimleri ve zaman dilimlerini öğrenebilirsiniz.

Sistem saatini ayarlama: Her kullanıcı sistem saatini okumak için izinliyken, sadece sistem yöneticisi olan root date komutunu kullanarak sistem saatini değiştirebilir ve MM, DD, hh, mm biçimindeki argümanlardan MM takvim ayı, DD takvim günü, hh saat ve mm de dakikadır. İsteğe bağlı olarak nadiren iki basamaklı yıl (yüzyıl için muhtemelen bir ya da iki) ve saniye (nokta ile ayrılmış) gerektiğinde ekleyebilirsiniz.

```
$ date
Thu Oct 5 14:28:13 CEST 2006
$ date 08181715
date: cannot set date: Operation not permitted
Fri Aug 18 17:15:00 CEST 2006
```

Date komutu yalnızca Linux sisteminin saatini değiştirir. Bu saat bilgisayarın anakartı üzerinde CMOS saatine aktarılmamış olabilir. Bu nedenle bu işlemi gerçekleştirmek için özel bir komut gerekli olabilir. Birçok dağıtım sistem kapatıldığında bunu otomatik olarak yapar.

Alıştırmalar

1. Varsayalım ki şu an 22 Ekim 2003, saat 12:34 ve 56. saniye olsun. Aşağıdaki çıktıyı elde etmek için durum biçimlendirme komutları ve date dökümanlarını çalışır:

   (a) 22-10-2003

   (b) 03-294 (WK43) (İki haneli yıl, yıl içinde gün sayısı, takvim haftası)

   (c) 12h34m56s

2. Şu anda Los Angeles'da saat kaç?

9.2 Kabuk Değişkenleri ve Ortamı

Çoğu yaygın kabuklar gibi bash diğer programlama dillerinde bulunan özelliklere sahiptir. Örneğin; değişkenlerde sayıları veya matinlerin bir parçasını depolamak mümkündür ve geri alınabilir. Değişkenler kabuk işlemlerini çeşitli şekilde denetlerler.

Ayarlama değişkenleri: Kabuk içinde bir değişken "foo=bar" gibi bir komut ile ayarlanır(Bu komut foo değişkenini metin değeri bar değişkenine ayarlar). Önünde veya eşittir işareti arkasında boşluk eklememeye dikkat edin! Önünde dolar işareti ile değişken adını kullanarak değişkenin değerini alabilirsiniz:

```
$ foo=bar
$ echo foo
foo
$ echo $foo
bar
```

(farka dikkat)

Kabuk değişkeninden ortam değişkenini ayırt edebilirsiniz. Kabuk değişkenleri tanımlandığı kabukta görülür. Harici bir komut başlatılır ve oradan kullanılabilirken diğer taraftan ortam değişkenlerine çocuk süreçler aktarılır (Çocuk süreçler bir kabuk olmak zorunda değildir. Her Linux süreci ortam değişkenlerine sahiptir). Tüm kabuğun ortam değiikenleri aynı zamanda kabuk değişkenidir; fakat aksi geçerli değildir.

Export komutunu kullanarak varolan kabuk değişkenini bir ortam değişkeniyle ifade edebilirsiniz.

```
$ foo=bar               foo şu anda kabuk değişkeni
$ export foo            foo şu anda ortam değişkeni
```

Veya aynı zamanda kabuk ve ortam değişkeni gibi yeni bir değişken tanımlayabilirsiniz

```
$ export foo=bar
```

Aynı anda birden fazla değişkenler için aynı çalışır:

```
$ export foo baz
$ export foo=bar baz=quux
```

Export komutuyla tüm ortam değişkenlerini görüntüleyebilirsiniz (hiçbir parametre almadan). Ayrıca env komutu da (hiçbir parametre almadan) mevcut ortamı görüntüler. Tüm kabuk değişkenleri (ortam değişkenlerini de içererek) set komutu kullanılarak görüntülenebilir. En yaygın değişkenler ve onların anlamları tablo 9.1 de gösterilmiştir.

Aynı zamanda set komutu farklı ve harika birçok şey yapar. Kabuk programlamayı kapsayan Advanced Linux Linup Ön Eğitim klavuzunda tekrar karşılaşacaksınız.

Env aslında sadace ortam süreçlerini göstermek yerine işlemek için tasarlanmıştır. Aşağıdaki örneği inceleyin:

```
$ env foo=bar bash          Foo ile çocuk kabuğu başlat
$ echo $foo
bar
$ exit                      Ana kabuğa geri dön
$ echo $foo
                             Tanımlı değil
$ _
```

En azından bash ile (ve ilişkiler) gerçekte genişletilmiş bir ortamla komutları çalıştırmak için env komutuna ihtiyaç duymazsınız.-Temelde aynı şeyi yapar.

```
$ foo=bar bash
```

Ancak env komutu da geçici ortam değişkenleri kaldırmak için izin verir (nasıl?). Bir değişken silme: Eğer yeterli kabuk değişkeni varsa unset komutunu kullanarak silebilirsiniz. Ayrıca bu çevreden onu siler. Eğer ortamdan bir değişkeni silip kabuk değişkeni olarak tutmak isterseniz "export -n" kullanın:

```
$ export foo=bar  foo ortam değişkeni
$ export -n foo  foo kabuk değişkeni (sadece)
$ unset foo  foo sonsuza dek gider ve kaybolur
```

9.3 Komut Tipleri - Yeniden Yüklemeler

Kabuk kontrol: Kabuk değişkenlerinin bir uygulaması kabuğun kendisinin kontrolündedir. İşte başka bir örnek: **??** . bölümde de tartıştığımız üzere kabuk iç ve dış komutları ayırır. Kabuk dış komutları PATH ortam değişkeninin değerini oluşturan dizinlerde hangi çalıştırılabilir programa karşılık geldiğini arar. İşte PATH için genel bir değer:

```
$ echo $PATH
/home/joe/bin:/usr/local/bin:/usr/bin:/bin:/usr/games
```

Bireysel dizinler listedeki kolona göre ayrılmış, bu nedenle listenin örneği olarak beş dizin oluşur. Eğer bir komut girişi yaparsanız

```
$ ls
```

kabuk bunun iç komut olmadığını bilir (kendi iç komutlarını bilir) ve bu nedenle PATH'i dizini en solundan araştırmaya başlar. Özellikle, aşağıdaki dosyaların mevcut olup olmadığını kontrol eder:

```
/home/joe/bin/ls  yok ...
/usr/local/bin/ls  hala şans yok ...
/usr/bin/ls  tekrar şans yok ...
/bin/ls  anlaşıldı!
     /usr/games dizini kontrol edilmez.
```

Bu /bin/ls dosyası adı üstünde ls komutu çalıştırmak için kullanılır.

Tabiki bu arama oldukça karışık süreçlerdir. Bunun sebebi kabuğun gelecek için hazırlık yapmasıdır: Eğer daha önce ls komutunun uygulası olarak /bin/ls dosyası tespit edilmiş ise, şu an için bu yazışmalar hatırlanır. "hashing" süreci çağrılır ve bunun ls komut tipini uygulayarak meydana geldiğini görebilirsiniz.

```
$ type ls
ls is hashed (/bin/ls)
```

Hash komutu hangi bash'de "hashing" olan emrediyor size söyler ve ne sıklıkla na zaman çağrıldığını söyler. "hashing-r" ile kabuğun karma belleğini silebilirsiniz. Diğer bir kaç seçeneğe bash klavuzundan bakabilir ya da "help hash" kullanarak öğrenebilirsiniz.

Açıkça söylemek gerekirse PATH değişkenin bir ortam değişkeni olmasına gerek yoktur - kabuk değişkeni mevcut kabuk için iyi durumda (bkz. alıştırma 9.5).

Her halükarda bir ortam değişkeni olarak tanımlamak için uygundur (genellikle de kabukları). Böylece kabuğun çocuk süreçlerinin istenilen değeri kullanılır.

Eğer kabuk kullanan belirli bir dış komut için programın hangisi olduğunu tam olarak bilmek istiyorsanız which komutunu kullanabilirsiniz:

```
$ which grep
/bin/grep
```

which, kabuğu aynı yöntemde kullanır - ilk olarak PATH dizininde başlar ve söz konusu dizinin istenilen komutu ile aynı ada sahip bir yürütülebilir dosya içerip içermediğini denetler.

which kabuğun iç komutları hakkında hiçbir şey bilmiyor olmasına rağmen "which test" komutu "/usr/bin/test" döndürür. Bu aslında iç komutların önceliğe sahip olduğundan çalıştırılabileceği anlamına gelmez. Eğer emin olmak istiyorsanız "type" kabuk komutunu kullanmanız gerekir.

whereis komutu çalıştırılabilir programların adlarını döndürür; ama aynı zamanda belge (man pages), kaynak kodu ve söz konusu komut(lar) ile ilgili farklı dosyaları da döndürür. Örneğin:

```
$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /etc/passwd.org /usr/share/passwd >
</usr/share/man/man1/passwd.1.gz /usr/share/man/man1/passwd.1ssl.gz>
</usr/share/man/man5/passwd.5.gz
```

Bu örnekte whereis(1) komutunu açıklayan (kabaca) doğrudan kodlanmış bir yöntem kullanılmıştır.

Alıştırmalar

1. Çocuk süreçlerin çalışması için Ortam ve kabuk değişkenlerine geçirerek aşağıdaki komutların sırasıyla çalıştığına dair kendinizi ikna edebilirsiniz.

   ```
   $ foo=bar  foo kabuk değişkeni
   $ bash  yeni kabuk (çocuk süreç)
   $ echo $foo
   foo tanımlı değil
   $ exit  ana kabuğa geri dön
   $ export foo  foo ortam değişkeni
   $ bash  yeni kabuk (çocuk süreç)
   $ echo $foo
   bar  ortam değişkeni ile birlikte geçti
   $ exit  ana kabuğa geri dön
   ```

2. Çocuk süreçte bir ortam değişkeni değişirse ne olur? Aşağıdaki komutları sırasıyla inceleyin:

```
$ foo=bar  foo kabuk değişkeni
$ bash  yeni kabuk (çocuk süreç)
$ echo $foo
bar  ortam değişkeni ile birlikte geçti
$ foo=baz  yeni değer
$ exit  ana kabuğa geri dön
$ echo $foo  ne alacağız?
```

3. PATH bir ortam değişkeni yerine "sadece" basit bir kabuk değişkeni ise kabuğun komut satırında aramayı çalıştırdığına da emin olun. Eğer PATH tamamen silinise ne olur?

4. Aşağıdaki komutları ele almak için hangi çalıştırılabilir programlar kullanılır: fgrep, sort, mount, xter

5. Sistemdeki hangi dosyalar "crontab" komutu için belgeleri içerir?

9.4 Uygun Bir Araç Olarak Kabuk

Kabuk birçok Linux kullanıcıları için en sık kullanılan araç olduğundan beri, geliştiricileri tarafından kullanımını rahat hale getirmek için birçok sorunu çözmek üzere tartışıldı. Daha kullanışlı bazı önemsiz şeyler buradadır:

Komut Editörü Basit bir metin editöründeki gibi komut satırlarını düzenleyebilirsiniz Bu sayede, enter tuşu kullanarak giriş bitmeden önce keyfi karekterler ekleyebilir veya silebilir ve giriş satırının etrafında imleci hareket ettirebilirsiniz. Bu editörün davranışları Linux üzerinde en popüler editörlerin "set-o vi" ve "set-o emacs" komutlarını kullanarak uyarlanabilir.

Durdurulan Komutlar Linux komutları çevresinde çokça isim karıştırmak veya yanlış bir parametreye geçmek kolaydır. Haliyle çalıştırılma anında iptal komutu olmalıdır. Bu işlem için sadece eş zamanlı olarak +c tuşuna basılmalıdır.

Tarihçe Kabuk "tarih" in bir parçası olarak şimdiye kadarki komutları hatırlar. ve tuşlarını kullanarak bu komutların listesinde hareket edebilirsiniz. Yukarıda açıklandığı gibi önceki komutu bulursanız kullanarak tekrar çalıştırabilir veya düzenleyip kullanabilirsiniz. +r komutunu kullanarak "adım adım" listeyi arayabilirsiniz - sadece karakter dizisi yazdığınızda ve kabuk bu diziyi içeren en son çalıştırılan komutu gösterir. Uzun dizilerde arama daha hassastır.

Sistem oturumunuzu kapattığınızda, kabuk gizli dosya ∼/.bash_history içinde geçmişini depolar ve daha sonra giriş yaptıktan sonra tekrar kullanılabilir hale getirir. (Söz konusu isim HISTFILE değişkenini ayarlayarak farklı bir dosya adı kullanabilir.)

"plain" dosyasında depolanan geçmişin saklanır olduğunun bir sonucu olarak bu dosya metin editörü kullanılarak düzenlenebilir (Bölüm **??**'de anlatıldığı gibi). Öyleki komut satırına yanlışlıkla şifrenizi girdiniz, el ile tarihi kaldırabilirsiniz - özellikle, sistem başıboşsa ev dizinlerinden herhangi birini herkes görebilir.

Otomatik Tamamlama Otomatik olarak komut ve dosya adlarını tamamlamak Bash kabuğunun muazzam bir yeteneğidir. Eğer tuşuna bastığınızda kabuk benzersiz bir tespit ile eksik bir girdi var ise devamını tamamlar. Bir komutun ilk sözcüğü için, bash geçerli veya belirtilen dizindeki tüm dosyaların komut satırı geri kalanı içinde tüm çalıştırılabilir programları düşünmektedir. Çeşitli komutlar veya dosyaların adları aynı başlıyorsa, kabuk adını mümkün olduğu kadar tamamlar sonrasında komut veya dosya adı hala eksik olabilir. İkinci bir tuşuna basıldığında kalan olasılıkları listeler.

Belirli programlara kabuğun tamamlama mekanizmasını uygulamak mümkündür. Örneğin; bir FTP istemcisi, komut satırında dosya adları yerine yakın zamanda ziyaret ettiği FTP sunucularının adlarını sunabilir. Ayrıntılar için bash belgelerine bakın.

Tablo 9.2 de bash içindeki en önemli tuşlar hakkında bir göz gezdirilmiştir.

Bir satır üzerinde çoklu komutlar Aynı giriş hattı üzerinde çeşitli komutlar girmek için mükemmel ücretsizdir. Sadece bir noktalı virgül kullanarak onları ayırmak gerekir:

```
$ echo Today is; date
Today is
Fri 5 Dec 12:12:47 CET 2008
```

Bu örnekte birinci komut çalıştırılmış olduğunda, ikinci komut yürütülür.

Koşullu Yürütme Bazen ilk komutun doğru çalışıp çalışmadığına bağlı olarak ikinci komutun çalıştırılması kullanışlıdır. Her Unix süreçleri doğru çalışan veya ne tür hataların meydana geldiğini belirten geri dönüş değeri dödürür. Önceki durumda, dönüş değeri 0; ikincisi de, 0 dan farklıdır. $? değişkeniyle kabuğun çocuk sürecinin geri dönüş değeri bulanabilir.

```
bash  çocuk süreç başlatıldı ...
$ exit 33  ... ve hemen tekrar çıkldı
exit
$ echo $?
33  yukarıdaki çıkış değeri
$ _
```

Ama bunun sonrakiler üzerinde etkisi yoktur.

İki komut arasında, "ayırıcı" olarak && kullanılır (aksi halde noktalı virgül nerde ise), ilk komuttan başarılı bir şekilde çıkıldığı zaman, ikinci komut yalnız yürütülür. Bunu görebilmek için kabuğun -c seçeneğini kullanabilirsiniz, bununla birlikte komut satırı üzerinden çocuk kabuğa bir komut iletilir (Etkileyici, değil mi?):

```
$ bash -c "exit 0" && echo "Successful"
Successful
$ bash -c "exit 33" && echo "Successful"
  -33 hiç başarılı değil
```

Aksine "ayırıcı" olarak ||kullanımında ilk komut başarılı bitmediği taktirde ikinci komut yalnız yürütülür.

```
$ bash -c "exit 0" || echo "Unsuccessful"
$ bash -c "exit 33" || echo "Unsuccessful"
Unsuccessful
```

Alıştırmalar

1. "echo "Hello!"" komutunda ne yanlış? (İpucu: Form komutları "! -2" Ya da "! ls" ile tecrübe edin.)

9.5 Dosyadan Komutlar

Bir dosya içinde kabuk komutları depolamayabilir ve bunu en bloc ile yürütebilirsiniz. (Elverişli bir dosya oluşturma **??**'de öğrenilecektir.) Parametre olarak verilen dosya adını sadece kabuğun çağırması gerekir.

```
$ bash my-commands
```

Bunun gibi bir dosya kabuk betiği çağırır ve çok kısa özetleyebiliriz ki kabuk burada çok geniş programlama özelliklerne sahiptir. (Advanced Linux Linup Ön Eğitim klavuzunda detaylı olarak kabuk programlama açıklanıyor.)

Büyülü sözler ekleyerek dosyayı "çalıştırılabilir" yaparak

```
#!/bin/bash
```

bash komutunun başa eklenmek zorunda kalma durumunu engelleyebilirsiniz.

```
$ chmod +x my-commands
```

(Bölüm **??**'de chmod ve erişim haklarıyla ilgili daha fazla bilgi bulabilirsiniz.) Bundan sonra

```
$ ./my-commands
```

komutu yeterli olacaktır.

Eğer yukarıdaki gibi bir kabuk betiği çağırılırsa bash komutunu başa ekleyerek veya dosyayı çalıştırarak geçerli kabuğun çocuk süreci bir alt kabukta çalıştırılır.

Bu da demektir ki, kabuk ya da ortam değişkenlerindeki değişikliklerin geçerli kabukta etkisi yoktur. Örneğin; dosyanın atama satırı içerdiğini varsayalım

```
foo=bar
```

Aşağıdaki komut dizisi dikkate alın:

```
$ foo=quux
$ bash assignment  foo=bar içerir
$ echo $foo
quux  Değişiklik yok; atama sadece altkabukda oldu
```

Bu genellikle bir özellik olarak kabul edilir, ama her zaman için (şimdi ve sonrasında) dosya komutlarının mevcut kabuğu etkileyecek olması oldukça gereklidir. Bu da çalışır: Eğer o anki kabukta bunlar doğrudan yazılırsa tam olarak kaynak komut dosyası satırlarını okur -tüm değişkenlerdeki değişiklikleri- dolayısıyla geçerli kabuk da etkili olur:

```
$ foo=quux
$ source assignment  foo=bar içerir
$ echo $foo
bar  Değişken değişti!
```

Lafı gelmişken kaynak komutu için farklı bir isim ".". (doğru okunmalı- nokta) Bu sebeple

```
$ source assignment
```

eşdeğerdir.

```
$ . assignment
```

Dış komutlar için program dosyaları gibi dosyaların okuması kaynak veya . kullanarak PATH değişkeni tarafından verilen dizinleri aramasıyla olur.

9.6 Programlama Dili Olarak Kabuk

Bir dosyadan kabuk komutları çalıştırmak için mümkün olduğunca iyi bir şey olduğuna emin olmak gerekir. Bununla birlikte her zaman aynı şeyleri yapmak zorunda olmadan kabuk komutlarnı daha iyi yapılandırmak mümkündür. Örneğin; komut satırı parametreleri elde edilebilir. Avantajları açıktır: Sık kullanılan prosedürleri, sürekli yazmak sıkıcı olacağından kaydedebilirsiniz ve nadiren kullanılan süreçleri bu kaydın dışında bırakın çünk bu sayede hataları hataları önemli ölçüde önleyebilirsiniz. Kabuğun als programlama dilini açıklamak için burada yeterli bir alana sahip değiliz; ama bir kaç örnek bunun için yeterli olacaktır.

Komut satırı parametreleri Bir kabuk betiğine komut satırı parametrelerini ilettiğinizde, kabuk $1, $2 gibi değişkenleri kullanabilir hale gelir. Aşağıdaki örneği inceleyin:

```
$ cat hello
#!/bin/bash
echo Hello $1, are you free $2?
$ ./hello Joe today
Hello Joe, are you free today?
$ ./hello Sue tomorrow
Hello Sue, are you free tomorrow?
```

$ * bir kerede tüm parametreleri içerir ve $ # parametrelerin sayısıdır:

```
$ cat parameter
#!/bin/bash
echo $# parameters: $*
$ ./parameter
0 parameters:
$ ./parameter dog
1 parameters: dog
$ ./parameter dog cat mouse tree
4 parameters: dog cat mouse tree
```

Döngüler For komutu kelimelerin bir listesi üzerinden yineleme ile döngü oluşturmayı sağlar (boşluklarla ayrılmış).

```
$ for i in 1 2 3
> do
> echo And $i!
> done
And 1!
```

```
And 2!
And 3!
```

Burada, i değişkeniyle sıralı listelenmiş varsayılan değerler, komutlar arasından do komutu ile yürütülür.

Eğer kelimeler bir değişken alınırsa bu daha eğlenceli hale gelir:

```
$ list='4 5 6'
$ for i in $list
> do
> echo And $i!
> done
And 4!
And 5!
And 6!
```

Eğer "in ..." yazmazsanız, döngü komut satırı parametreleri üzerinde dolaşır:

```
$ cat sort-wc
#!/bin/bash
# Sort files according to their line count
for f
do
echo `wc -l <"$f"` lines in $f
done | sort -n
$ ./sort-wc /etc/passwd /etc/fstab /etc/motd
```

("wc -l" komutunu komut satırına geçirilen standart giriş veya dosya(lar) satırları saymaktadır.) İş hattı kullanarak sıralamak için döngüyü standart çıktıya yönlendirme unutulmamalıdır!

Alternatifler Sadece belirli koşullar altında belirli komutları çalıştırmak için daha önce de anlatılan ||ve && operatörlerini kullanabilirsiniz.

```
#!/bin/bash
# grepcp REGEX
rm -rf backup; mkdir backup
for f in *.txt
do
grep $1 "$f" && cp "$f" backup
done
```

Betik, örneğin; parametre olarak iletilen düzenli ifade ile eşleşen en az bir satır içerirse (for döngüsü sağlar) ve ismi .txt ile sonlanıyorsa sadece backup dizine bir dosya kopyalar.

Alternatifler için yararlı bir araç olarak çeşitli şartlarda kontrol sağlayan test komutu vardır. Koşul gerçekleşirse çıkış kodu 0 döndürür (başarılı), gerçekleşmemesi durumunda sıfırdan farklı bir çıktı verir (başarısız). Örneği inceleyin:

```
#!/bin/bash
# filetest NAME1 NAME2 ...
for name
do
test -d "$name" && echo $name: directory
test -f "$name" && echo $name: file
test -L "$name" && echo $name: symbolic link
done
```

Bu betik dizin yerine kullanılan her bir dosya veya sembolik dizin için parametrelerin ve çıkışları olarak iletilen dosya isimlerinin sayısına bakar.

Test komutu hem bin/test teki gibi serbest çalışan bir program olarak hem de bash ve diğer komutlardaki gibi dahili olarak mevcuttur. Çok daha garip testler söz konusu olduğunda bu değişmeler ince farklılıklar gösterebilir. Şüpheniz varsa, belgeleri okuyun.

Eğer bir koşulla, birden fazla komutu bağımlı kılacak bir komut varsa kullanabilirsiniz (uygun ve okunabilir bir şekilde). "[...]" yerine "test ..." yazılabilir:

```
#!/bin/bash
# filetest2 NAME1 NAME2 ...
for name
do
if [ -L "$name" ]
then
echo $name: symbolic link
elif [ -d "$name" ]
echo $name: directory
elif [ -f "$name" ]
echo $name: file
else
echo $name: no idea
fi
done
```

Eğer komut sinyali "başarılı" ise (çıkış kodu 0) komutlar yürütülür, sonrasında gelen elif, else veya fi komutları sona erer. Öte yandan sinyal "başarısız" ise sonraki elif komutu değerlendirilmeye alınacak ve çıkış kodu olarak kabul edilecektir. Eşleşen fi komutuna ulaşılana kadar kabuk modele devam eder.

If veya elseif komutlarından hiç biri "başarı" ile sonuçlanmadı ise sonrasında else komutu yürütülür. Bunlar gerekli değilse elif ve başka dallanmalar ihmal edilebilir.

Daha fazla döngü Başlangıçta döngü için sabit bir dönme sayısı belirlenir (Listedeki sözcük sayısı). Ancak, sık sık bir döngü ne sıklıkta çalıştırılmalıdır; Bu gibi net olmayan durumlar ile en başta ilgilenmek gerekir. Komut çalışırken (If gibi) "başarılı" yada "başarısız" olduğu belirlenebilir. Bunu halledebilmek için kabuk while komutunu sunar. Sonuç başarılı ise "bağlı" komutlar çalıştırılır, başarısız ise döngüden sonraki komutlar çalıştırılacaktır.

Aşağıdaki betik gibi bir dosyayı okur

```
Aunt Maggie:maggie@example.net:the delightful tea cosy
Uncle Bob:bob@example.com:the great football
```

(Adını Komut satırına veren) ve her satırda bir teşekkürler e-postası oluşturur (Linux günlük yaşamda çok yararlıdır):

```
#!/bin/bash
# birthday FILE
IFS=:
while read name email present
do
(echo $name
echo ""
echo "Thank you very much for $present!"
echo "I enjoyed it very much."
echo ""
echo "Best wishes"
echo "Tim") | mail -s "Many thanks!" $email
done <$1
```

read komutu giriş dosyasındaki satırları okur ve her satırı üç kolon olarak bölümlendirir. Bu kolonlar isim email ve döngü içinde bulunan mevcut değişkenlerdir. Mantığa aykırı olarak döngüye yeniden yönlendirilen giriş sonda bulunabilir.

Bu betiği zararsız e-posta adresleri ile test edin, ilişkilerinizi karışık hale getirir!

Alıştırmalar

1. Arasındaki fark (olduğu kadar döngü çalıştırma söz konusu olduğunda) nedir?

   ```
   for f; do ...; done
   ```

ve

```
for f in $*; do ...; done
```

(Eğer gerekirse deneyin)

2. sort-wc betiğinde neden

   ```
   wc -l <$f
   ```

   yerine

   ```
   wc -l $f
   ```

   kullanıyoruz?

3. grepcp değiştirilebilir. Bu şekilde dikkat edilmesi gereken dosyaların listesi de komut satırından alınır (İpucu: Shift kabuk komutu $ dan ilk komut satırı parametresini siler ve oluşacak açığı kapatmak için diğerlerini yukarı çeker. Kaydırmadan sonra, önceki $2 $1 olur, $3 de $2 olur ve böyle devam eder).

4. Neden filetest komutunun sembolik linkler için (sembolik bağ <yerine sadece> foo) çıktısı yoktur?

   ```
   $ ./filetest foo
   foo: file
   foo: symbolic link
   ```

Bu Bölümdeki Komutlar

. Komut satırına giriş yapılırsa kabuk komutlarını içeren dosyaları okur

date Süreç ortamı Çıkışlarını veya ayarlanmış ortam programlarını başlatır

export Tanımlananları ve ortam değişkenlerini yönetir

hash bash deki "görülen" komutları gösterir ve yönetir

set Kabuk değişkenlerini ve seçeneklerini yönetir

source Komut satırına giriş yapılırsa kabuk komutlarını içeren dosyaları okur

test Komut satırında mantıksal ifadeleri değerlendirir

unset Kabuk veya ortam değişkenlerini siler

whereis Çalıştırılabilir programları, el ile oluşturulmuş sayfaları ve kaynak kodu verilen programları arar

which PATH boyunca programları arar

Özet

- sleep komutu argüman olarak belirtilen saniye sayısı için bekler.
- echo komutu argümanları çıktılar.
- Tarih ve saat date komutu kullanılarak tespit edilebilir.
- bash gibi çeşitli özellikler komutu ve dosya adını tamamlama, komut satırı düzenlemesi, takma adları ve değişkenler gibi interaktif kullanımı destekler.

Table 9.1: Önemli Kabuk Değişkenleri

Değişken	Anlamı
PWD	Geçerli dizinin ismi
EDITOR	Kullanıcının tercih ettiği editörün ismi
PSI	Kabuk komutu yönetme şablonu
UID	Geçerli kullanıcının kullanıcı ismi
HOME	Geçerli kullanıcının ev dizini
PATH	Dış komutlar gibi uygun çalıştırılabilir programlar içeren dizinlerin list
LOGNAME	Geçerli kullanıcının kullanıcı ismi (tekrar)

Table 9.2: bash içindeki tuşlar

Tuşlar	İşlevleri
veya	En son komutlar arasında gezinir
+r	Komut geçmişini arar
-	Geçerli komut satırı içindeki İmleci hareket ettirir
+a	Komut satırının başına gider
+e	Komut satırı sonuna atlar
	Sırasıyla, imlecin altındaki/önündeki karakteri siler
+t	İmlecin Önündeki ve altındaki iki karakterin yerlerini değiştirir
+l	Ekranı temizler
+c	Komuta ara verir
+d	Son giriş (giriş kabukları için:kapalı oturum)

Bölüm 10

Dosya Sistemi

Amaçlar

- "dosya" ve "sistem dosya" kavramların anlamak
- Farklı dosya türlerin tanımak
- Linux sistemin ağaç dizi yollarını öğrenmek
- Dizi ağacı içerisinde harici dosyanın nasıl entegre edileceğini öğrenmek

Önceden Bilinmesi Gerekenler

- Linux temel bilgileri (Önceki Konulardan)
- Dosyalar ve Dizinleri işlemesi (Bölüm **??**)

10.1 Terimler

Genelde dosya, verilerin kendi içerisinde bulunan toplamlarıdır. Dosya içindeki veri türlerine bağlı herhangi bir kısıtlama yoktur; bir dosya metin birkaç harftan oluşabilir veya kullanıcın tam iş hayatın birden çok megabyte içeren arşivden oluşur. Dosyalar düz metin içermeleri gerekmez. Görüntü, Ses,.. çalışabilir uygulamalar ve diğer pek çok dosyalar bir depo üzerine yerleştirilir. Bir dosya veri türünü tahmin etmek için dosyanın içinde bulunan dosya komutun kulanabilir:

```
$ file /bin/ls /usr/bin/groups /etc/passwd
/bin/ls: ELF 32-bit LSB executable, Intel 80386,
  version 1 (SYSV), for GNU/Linux 2.4.1,
dynamically linked (uses shared libs), for GNU/Linux 2.4.1, stripped
/usr/bin/groups: Bourne shell script text executable
/etc/passwd: ASCII text
```

/usr/share/file alt dizinindeki kurallara uygun dosya sistemini tahmin eder. yönetici /usr/share/file/magic alt dizini kuralların bulunduğu bir metin dosyası bulundurur. Kendi kurallarınızı /etc/magic alt dizinine koymak şartıyla tanımlayabilirsiniz. Detaylar için magic(5)'e bakınız. Uygun bir şekilde işleyebilmesi için bir Linux sistemi binlerce farklı dosyaya ihtiyaç duyar. Bunlar sistemin çeşitli kullanıcıları tarafından oluşturulmuş ve sahip olunan çeşitli dosyalardır.

Uygun bir şekilde işleyebilmesi için bir Linux sistemi binlerce farklı dosyaya ihtiyaç duyar. Bunlar sistemin çeşitli kullanıcıları tarafından oluşturulmuş ve sahip olunan çeşitli dosyalardır.

Bir dosya sistemi depolama alanı üzerinde veri düzenleme ve yönetimini belirler. Bir sabit disk temelde sistemin tekrar bulabilmesi gereken byte ları içerir, hatta çok büyük,esnek ve verimli dosyalar olsalar bile. Dosya işletim sistemi detayları farklılık gösterebilir (Linux bunların bir çoğunu bilir. ext2, ext3 , ext4, ReiserFS, XFS, JFS, btrfs vb.) fakat kullanıcıya sunulan şey farklı türdeki ve isimdeki dosya ve dizin adlarıyla oluşturulmuş bir ağaç hiyerarşi yapısıyla aynıdır (ayrıca bkz. Bölüm **??**).

Linux topluluğunda, terim "dosya sistemi" birkaç anlamı taşır. Burada sunulan anlama ek olarak—"bir ortam üzerinde bayt düzenleme yöntemi", bir dosya sistemi her zaman bizim "dizin ağacı" olarak nitelendirdiğimizi göz önünde bulundurur. Buna ek olarak, bu veriler ile birlikte, belirli bir ortamda (bir sabit disk bölümü, USB anahtar, ...) genellikle bir "dosya sistemi" adı verilir. Örneğin bizim söylediğimiz anlamda sabit bağlantılar (Bölüm **??**) sabit disk veya sabit disk ve USB anahtarı arasındaki iki farklı bölümler arasında, yani "dosya sistemi sınırlarının ötesinde" işe yaramaz.

10.2 Dosya Türleri

Linux sistemlerinde temel öncül "Herşey bir dosyadır". Bu ilk bakışta şaşırtıcı görünebilir fakat çok kullanışlı bir konsepttir. Altı dosya türü şu şekilde sıralanabilir:

Yalın dosyalar Bu dosya grubu metinleri, grafikleri, ses dosyalarını içerir. Aynı zamanda çalıştırılabilir dosyaları da içerir. Yalın dosyalar editors, cat, shell output redirection gibi alışılmış araçlar kullanılarak oluşturulabilir.

Dizinler Aynı zamanda "dosyalar" olarak bilinen dizinler, bildiğimiz gibi depolamaya yardımcı olur. Bir dizin temelde dosya adları ve ilişkili düğüm numaralarını veren bir tablodur. Dizinler mkdir komutu kullanılarak oluşturulurlar.

Sembolik Linkler Farklı bir dosyaya erişmek için belirtilen yolardır. (Windows' taki "kısayollar"a benzer). Ayrıca bkz Bölüm **??**. Sembolik linkler ln -s kullanılarak oluştulurlar.

Aygıt Dosyaları Bu dosyalar, disk sürücüleri gibi isteğe bağlı cihazlar için arabirim olarak görev. Örneğin, /dev/fd0 dosyası ilk disket sürücüyü temsil eder. Her yazma veya böyle bir dosyaya okuma erişimi gelen cihaza yönlendirilir. Aygıt dosyaları mknod komutu kullanılarak oluşturulur; Bu genellikle sistem yöneticisi önceliğidir ve bu yüzden bu kılavuzda daha ayrıntılı olarak açıklanmamıştır.

FIFOlar Genellikle "adlandırılmış yöneltim" olarak bilinirler. Kabuk yöneltmeleri gibi, ara dosyaları kullanmadan süreçler arasındaki doğrudan iletişimi sağlarlar. Bir işlem FIFO'yu yazma, diğeri de okuma için açar. Kabuğun program açısından dosya gibi hareket eden ardışık düzen için kullandığı yönlendirmelerin aksine isimsizdir – Dosya sistemi içinde bulunmazlar sadece bağlantılı süreçlerin arasında bulunurlar – FIFO ların dosya isimleri vardır ve gelişigüzel programlar tarafından dosyalar gibi açılabilirler. Bunun yanısıra, FIFO ların erişim hakları olabilir (yönlendirmelerin yoktur). FIFO'lar mkfifo emri kullanarak oluşturulurlar.

Unix-alan soketleri FIFOlar gibi, Unix-alan soketlerinde de süreçler arası iletişim yöntemi vardır. TCP/IP üzerinden gerçek ağ iletişimleriyle aynı programlama arayüzünü kullanıyorlar.Diğer taraftan, Unix-alan soketleri TCP / IP'ye göre çok daha verimlidir. FIFOların aksine, Unix alan soketleri iki yönlü haberleşmeyi-katılımcı da veri gönderip alabilir- sağlar. Eğer X sunucusu ve istemcileri aynı bilgisayarda ise Unix-alan soketleri X11 grafik sistemleri tarafından kullanılırlar. Unix-alan soketlerini oluşturmak için herhangi bir program yoktur.

Alıştırmalar

1. Sisteminizi çeşitli dosya türü örnekleri çerçevesinde kontrol ediniz (Tablo 10.1 Size sorulardaki dosyaları nasıl tanıyacağınızı gösterecek.)

10.3 Linux Dizin Ağacı

Bir Linux sistemi yüz binlerce dosya çeşidinden oluşu. rizleri tutmak için dizin yapıları ve Linux sistemi içeren bir dosya için belli kurallar vardır,F ilesystem Hierarchy Standard (FHS). Çoğu Linux dağıtımında bu standart(ufak sapmalarla) FHS'ye uygundur. FHS bütün dizinleri açıklar.

immediately below the file system's root as well as a second level below /usr . Dosya sistemi ağacı kök diziniyle başlar, "/" (root dizini /root ile karıştırılmamalı, root kullanıcının ev dizini). The root directory contains either just subdirectories or else additionally, if no /boot directory exists, the operating system kernel.

Kök dizinin alt dizinlerini listelemek için "ls -la /" komutunu kullanabilirsiniz. Sonuç Şekil 10.1 gibi görünmelidir. Bireysel alt dizinler FHS'ye uyar,bu nedenle yaklaşık olarak her dağıtım aynı dosyaları içerir. Şimdi dizinlerden bazılarına daha yakından bakalım:

```
$ cd /
$ ls -l
insgesamt 125
drwxr-xr-x   2 root root  4096 Kas 27 11:53 bin
drwxr-xr-x   3 root root  4096 Kas 30 10:53 boot
drwxr-xr-x   2 root root  4096 Oca  6  2012 cdrom
drwxr-xr-x  15 root root  4380 Ara 11 17:29 dev
drwxr-xr-x 164 root root 12288 Ara 18 09:27 etc
drwxr-xr-x  89 root root  4096 Eki 18 13:08 home
drwxr-xr-x  25 root root  4096 Eki 19 15:00 lib
drwxr-xr-x   2 root root  4096 Eki 19 14:33 lib64
drwx------   2 root root 16384 Oca  6  2012 lost+found
drwxr-xr-x   3 root root  4096 Eki 19 15:49 media
drwxr-xr-x   2 root root  4096 Eki  9  2011 mnt
drwxr-xr-x   5 root root  4096 Kas 29 10:29 opt
dr-xr-xr-x 205 root root     0 Ara  6 09:31 proc
drwx------  12 root root  4096 Ara 14 20:26 root
drwxr-xr-x  26 root root   880 Ara 18 14:15 run
drwxr-xr-x   2 root root 12288 Ara 12 09:05 sbin
drwxr-xr-x   2 root root  4096 Haz 21  2011 selinux
drwxr-xr-x   2 root root  4096 May 11  2012 srv
dr-xr-xr-x  13 root root     0 Ara  6 09:31 sys
drwxrwxrwt  16 root root 12288 Ara 18 15:32 tmp
drwxr-xr-x  11 root root  4096 Oca  6  2012 usr
drwxr-xr-x  15 root root  4096 Ara  6 09:30 var
```

FHS hakkında çok fazla fikir birliği vardır fakat sadece Linux üzerindeki bir şeyi bağlamak kadardır, yani o kadar da fazla değildir. Bir yandan çoğunlukla

sadece üretici tarafından dokunulan ve FHS'nin bütün ayrıntılarına uymanın hiçbir şey kazandırmadığı Linux sistemleri vardır(Örneğin geniş bant yönlendirici ve PVR). Diğer yandan kendi sisteminizde ne isterseniz yapabilirsiniz, fakat sonuçlarına hazırlıklı olmanız gerekir. Dağıtıcınız Dosya Hiyerarşi Sistemi'ne uymanız halinde bir sıkıntı çıkmayacağını size garanti eder, fakat aynı zamanda kurallara uymadan yaptığınız işlemlerden çıkan hatalardan dolayı şikayet etmemenizi bekler. Örneğin, eğer /usr/bin dizinine bir program yüklüyorsanız ve söz konusu olan dosya gelecek sistem güncellemesi esnasında üzerine fazla yazılırsa, FHSye göre /usr/local/bin dizinine kendi programlarınızı koymanız beklenmediği için bu sizin kendi hatanızdır.

İşletim sistemi çekirdeği-/boot /boot dizini asıl işletim sistemini içerir: vmlinuz Linux çekirdeğidir. /boot dizininde ayrıca boot yükleyicisi için gereken diğer dosyalar vardır (LILO yada GRUB).

Genel araçlar -/bin /bin dizini içinde sistemi başlatmak için gerekli olan en önemli çalıştırılabilir programları(çoğunlukla sistem programları) vardır. Bu örneğin mkdir ve mount komutlarını içerir. Bu programların bazıları o kadar önemlidir ki sadece sistem başlaması esnasında değil, sistem çalışırken de gereklidir - ls ve grep gibi. /bin dizini aynı zamanda zarar görmüş sistemi tekrar çalıştırmak için gerekli olan programları içerir eğer sadece kök dizinini içeren dosya sistemine ulaşılabilirse. Sistem sistem tamiri ya da başlaması esnasında gerekli olmayan ek programlar /usr/bin dizini içinde bulunabilirler.

Özel sistem programları-/sbin /bin dizini gibi /sbin dizini de sistemi onaracak ya da başlatacak prgramları içerir. Ancak çoğu parça için bunlar sadece kök dizini(root) tarafından kullanılabilen sistem konfigrasyonu araçlarıdır. Normal kullanıcılar sistemi sorgulayacak programların bazılarını kullanabilirler, fakat hiçbir şeyi değiştiremezler. /bin dizininde olduğu gibi bu dizinle birlikte daha fazla sistem programlarını içeren /usr/sbin denilen bir dizin vardır.

Sistem kütüphaneleri-/lib Burası dosyalar ve bağlantılar gibi /bin ve /sbin içinde bulunan programlar tarafından kullanılan "paylaşılmış kütüphaneler"dir. Paylaşılmış kütüphaneler çeşitli programlar tarafından kullanılan kodun parçalarıdır. Bazı süreçler aynı temel parçaları kullandığı için böyle kütüphaneler birçok kaynağı kurtarır ve bu temel parçalar belleğe sadece bir kere yüklenmeli: ek olarak bir kere sistemde bulunduklarında ve tüm programlar merkezi bir dosyadan söz konusu olan kodu yakaladığında böyle kütüphanelerde arızları onarmak daha kolaydır. Tesadüfen /lib/modules dizini altında çekirdek modülleri vardır(aygıt sürücüleri, dosya sistemleri ya da ağ protokolleri kullanımında

gerekli olmayan çekirdek kodu gibi). Bu modüller gerek duyulduğunda çekirdek tarafından yüklenebilirler ve bazı durumlarda kullanım sonrası kaldırılabilirler.

Aygıt dosyaları - /dev bu rehber ve alt dizinleri aygıt dosyaları için çoklu kayıtları içerir. Aygıt dosyaları kabuk(genellikle komut satırı kullanıcıları yada programcılar için ulaşılabilir olan sistemin bir parçası) ve çekirdek içerisindeki aygıt sürücüleri arasında arayüz oluştururlar. Diğer dosyalar gibi içerikleri yoktur fakat aygıt numaraları aracılığıyla çekirdek içerisindeki bir sürücüye referans olurlar.

Daha önceki zamanlarda Linux dağıtıcıları için her kişiye göre araç için /dev dizini içinde bir aygıt içermesi alışılmıştı. Böylece bir laptop Linux sistemi bile her biri 63 bölümlü, 8 ISDN adaptörü, 16 seri halinde ve 4 paralel arayüzü ve bu tür özellikleri taşıyan 10 sabit disk için gerekli olan araç dosyalarını içerirdi. Bugün bu eğilim her hayali aygıt için bir aygıtı olan fazla dolu /dev dizinlerinden ve gerçekte bulunan araçlar için girişleri içeren çalışmakta olan çekirdeğe yakın olarak bağlanan sistemler için uzaktadırlar. Bu alanda sihirli kelime udev'dir ve Linux Administration kısmında daha detaylı anlatılacaktır.

Linux karakter araçları ve blok araçları arasında ayrım yapar. Örneğin bir karakter aracı ağdır,faredir ya da bir modemdir -tek karakterleri işleyen ya da sağlayan bir araç. Bir blok aracı ise bloklarda bilgileri işler- bu byte'ların tek başına okunamadığı, 512'li gruplar halinde okunduğu disket ya da sabit diskleri içerir. Aygıt dosyaları "ls -l" dizini içinde "c veya b" çıkışlarıyla etiketlenir.

```
crw-rw-rw-  1 root root 10, 4 Oct 16 11:11 amigamouse
brw-rw----  1 root disk  8, 1 Oct 16 11:11 sda1
brw-rw----  1 root disk  8, 2 Oct 16 11:11 sda2
crw-rw-rw-  1 root root  1, 3 Oct 16 11:11 null
```

Dosya uzunluğu yerine bu liste 2 tane numarayı içerir. İlki aygıtın türünü belirten ve çekirdek sürücüsü bu aygıtın sorumluluğunda olduğu zaman yöneten "asıl aygıt numarası"dır. Örneğin tüm SCSI sabit diskleri 8 numaralı başlıca aygıta sahiptir. İkincisi ise "ikincil aygıt numarası"dır. Bu benzer ya da ilgili aygıtlar arasında ayrım yapan ya da bir diskin çeşitli bölünmelerini belirten sürücü tarafından kullanılır.

Birkaç tane göze çarpan uydurulmuş aygtlar vardır. null device, /dev/null, aslında gerekli olmayan program çıkışı için bir çöp kutusu gibidir, fakat aşağıdaki şekildeki gibi bir yerde yönetilmelidir.

```
$ program >/dev/null
```

Aksi halde ağda gösterilen programın standart çıkışı göz ardı edilir. Eğer /dev/null okunursa boş bir dosya gibi davranır ve hemen dosya sonuna döner. /dev/null yazan ve okuyan tüm kullanıcılar için ulaşılabilir olmalıdır.

/dev/random ve /dev/urandom dizinlerindeki "aygıtlar" sistemde gürültüden oluşturulan şifrelemeyle ilgili kalitenin rastgele byte'larına döner-anahtar basımı gibi tahmin edilemeyen olaylar arasındaki aralıklar gibi. /dev/random dizininden gelen bilgiler yaygın şifrelemeyle ilgili algoritmaları için anahtarlar oluşturmak için uygundur. /dev/zero dosyası boş byte'ların sınırsız bir isteğine döner; bunları örneğin dd komutlu dosyaların üzerine yazmak için veya oluşturmak için kullanabilirsiniz.

Yapılandırma dosyaları - /etc /etc dizini çok önemlidir; çoğu program için yapılandırma dosyalarını içerir. Örneğin /etcinittab ve /etc/init.d/* dosyaları sistem dosyalarını başlatmak için gerek duyulan çoğu belirli bilgileri içerir. Aşağıda en önemli dosyaların daha detaylı tanımları bulunmaktadır-onların birkaçı dışında, sadece kök dizini kullanıcısının yazma izni vardır fakat herkes onu okuyabilir.

/etc/fstab bu tüm yerleştirilebilir dosya sistemini ve onların özelliklerini tanımlar(tür,ula şekli vs.).

/etc/hosts bu dosya TCP/IP ağının yapılandırma dosyalarından biridir. IP adreslerine gelen ağ misafilerinin isimlerini haritalandırır.küçük ağlarda ve bağımsız misafirlerde bu yeni bir sunucu yerleştirir.

/etc/inittab bu dosya init programı için ve sistem başlangıcı için yapılandırma dosyasıdır.

/etc/init.d/* bu dizin çeşitli sistem servisleri için "init kodlarını" içerir. Bunlar başlatmak için ya da sistem kapandığında sistem dosyalarını durdurmak için kullanılırlar. Red Hat dağıtımlarında bu dizine /etc/rc.d/init.d denilmektedir.

/etc/issue bu dosya giriş yapmak için kullanıcıya sorulmadan önce karşılamayı içerir. Yeni bir sistemin yüklenmesinden sonra çoğunlukla satıcının ismini içerir.

/etc/motd bu dosya kullanıcı başarılı bir şekilde giriş yaptıktan sonra ortaya çıkan "günün mesajını" içerir. Sistem yöneticisi bu dosyayı önemli bilgileri ve olayların kullanıcılarını bildirmek için kullanır.[1]

/etc/mtab bu yerleştirme noktalarını içeren tüm yerleştirilmiş dosya sistemlerini içeren bir listedir. Bu dosya an itibariyle yerleştirilmiş tüm dosya sistemlerini içermesinden dolayı /etc/fstab dosyasından ayrılmaktadır, /etc/fstab dosyası sadece yerleştirilmesi mümkün olan dosya sistemleri

[1] There is a well-known claim that the only thing all Unix systems in the world have in common is the "message of the day" asking users to remove unwanted files since all the disks are 98% full.

için ayarları ve seçenekleri içerir. Hatta bu liste komut satırı aracılığıyla dosya sistemlerini yerleştirebildiği için ayrıntılı değildir. Dosyaların durgun olması gereken /etc içindeki bir dosyaya bu tür bilgileri yerleştirmemeliyiz.

/etc/passwd burada sistemce bilinen tüm kullanıcıların bir listesi vardır, kullanıcının belirli bilgilerinin çeşitli araçlarını toplamak için. Dosyanın ismine rağmen modern sistemlerde şifreler bu dosya içinde depolanmaz fakat /etc/shadow denilen başka bir dosya içinde depolanır. /etc/paswd'un aksine bu dosya normal kullanıcılar tarafından okunamaz.

Aksesuarlar-/opt Bu dizin gerçekten 3. şahıs yazılımına yöneliktir - bölgesel olarak yüklenen dosyalar ya da dağıtım dosyalarıyla çelişmeden yüklenebilmesi gereken tüm paketler. Böyle yazılım paketleri /opt (packages) gibi bir alt dizin meydana getirirler. Haklı olarak /opt dizini boş bir disk üzerine dağıtım yüklendikten sonra tamamen boş olmalı.

"değişmeyen dosyalar"-/usr burada programları ve sistemin onarımı ya da başlatılması için önemli olmayan bilgi dosyalarını içeren çeşitli alt dizinler vardır. En önemli dizinler şunları içerir:

/usr/bin başlatma için önemli ya da önemsiz olan sistem programları

/usr/sbin kök dizini için daha fazla sistem programları

/usr/lib /bin ya da /sbin için kullanılmayan daha fazla kütüphane

/usr/local bölgesel sistem yöneticisi tarafından yüklenmiş dosyalar için dizin. /opt diziniyle benzerlik gösterir-dağıtım buraya hiçbir şey koymaz

/usr/share mimari olarak bağımsız veri. Esas olarak Intel, Sparc ve PowerPC ana makinelerinden oluşan bir Linux ağı merkezi bir sunucu üzerinden bu dosyanın tek bir kopyasını paylaşabilir.

/usr/share/doc belgeleme, örneğin HOWTO

/usr/share/info bilgi sayfaları

/usr/share/man alt dizinlerde elle yapılan sayfalar

/usr/sc çekirdek ve diğer programlar için kaynak kodu(eğer mümkünse).

/usr ismi çoğunlukla yanlış olarak Unix sistem kaynaklarının kısaltması olarak düşünülür. Esasen bu dizin bilgisayarların büyük ve yavaş bir sabit diskleri olduğu zamandan daha küçük ve daha hızlı sabit diskleri olduğu zamana kadarki süreçte türemiştir. Büyük disk daha az sıklıkla kullanılan ya da çok büyük olan dosyalar ve programlar için bir bellek olarak görev yaparken

bütün sık kullanılan programlar ve dosyalar küçük diske gider. Bugün bu ayrım bir şekilde değişebilir: Dikkatle /usr dizinini kendi bölmesinin üzerine yerleştirebilirsin ve bu bölmeyi "salt okunur" olarak değiştirebilirsin. Disk belleğinin düşen fiyatları bunu artık gerekli kılmamasına rağmen uzak bir sunucudan /usr'yi içe aktarmak mümkündür (yaygın Linux dağıtımları bunu hiçbir şekilde desteklemez).

Çekirdek içine bir pencere-/proc bu en ilginç ve önemli dizinlerden biridir. /proc uydurulmuş dosya sistemidir: diskte boşluk meydana getirmez fakat onun alt dizinleri ve dosyaları birisi içerikleriyle ilgilendiğinde ya da ilgilenirse çekirdek tarafından oluşturulur. bilgisayarın donanımı hakkında çekirdeğin sahip olduğu diğer bilgilerin yanı sıra işleyen süreçler hakkında da bir çok bilgi bulacaksınız. Örneğin bazı dosyalarda bütün bir donanım analizi bulacaksınız. En önemli dosyalar şunları içerir:

/proc/cpuinfo işlemcinin tipi ve saat frekansı hakkında bilgiler içerir.

/proc/devices bu başlıca aygıt numaralarını içeren, çekirdek tarafından desteklenen aygıtların tüm listesini içerir.

/proc/dma kullanımdaki DMA kanallarının bir listesidir. Bugünün PCI temelli sistemlerinde ne ilgi çekici ne de önemlidir.

/proc/interrupts kullanımdaki bütün donanım kesintileridir. Bu kesinti numaralarını, tetiklenen kesintilerin numaralarını ve belirli kesintilerle uğraşan sürücüleri içerir(Bir kesinti sadece çekirdekte bir kesinti olduğunda bu listede yer alır).

/proc/ioports /proc/interrupts gibi fakat I/O bağlantı yerleri içindir.

/proc/kcore bu dosya bilgisayarın bütün RAM'lerini uygun hale getirir ve çekirdeğin hatalarını ayıklamak için gereklidir. Okumak için yetkili kullanıcı (root) ayrıcalıklarını gerektirir. Bundan uzak durursanız iyi edersiniz.

/proc/loadavg son 1,5 ve 15 dakika içindeki işlemci yüklemesini ölçen 3 numarayı içerir. Bu değerler genellikle uptime programı sayesinde verilir.

/proc/meminfo bellek ve takas kullanımını gösterir. Bu dosya free program tarafından kullanılır.

/proc/mounts tüm yerleştirilmiş dosya sistemlerinin başka bir listesidir. Çoğunlukla /etc/mtab'la özdeştir.

/proc/scsi bu dizinde uygun SCSI aygıtlarını listeleyen scsi isminde bir dosya vardır. SCSI hakkında bilgi veren bir 0(1,2,. . . ,aynı türün çoklu adaptörleri için) dosyası içeren sistemde SCSI ana makine adaptörünün her türü için başka bir alt dizin vardır.

/proc/version model numarasını ve şu anki çekirdeğin derleme tarihini içerir.

Back when /proc had not been invented, programs like the process status display tool, ps, which had to access kernel information, needed to include considerable knowledge about internal kernel data structures as well as the appropriate access rights to read the data in question from the running kernel. Since these data structures used to change fairly rapidly, it was often necessary to install a new version of these programs along with a new version of the kernel. The /proc file system serves as an abstraction layer between these internal data structures and the utilities: Today you just need to ensure that after an internal change the data formats in /proc remain the same—and ps and friends continue working as usual.

Donanım kontrolü-/sys Linux çekirdeği versiyon 2.6'dan bu yana bu dizini belirtmiştir. /proc gibi çekirdeğin talebi üzerine uygun hale getirilir ve geniş bir alt dizinleri sıralaması içinde uygun donanım üzerinde istikrarlı bir görüşe izin verir. Aynı zamanda çeşitli özel dosyalar aracılığıyla donanım üzerinde yönetim işlemlerine destek sağlar.

Teorik olarak bireysel girişlerle hiçbir alakası olmayan /proc dizinindeki tüm girişler /sys dizinine yavaşça geçirilmeli. Bu stratejik amaç gerçekleştirildiğinde bu, birisinin tahmini olduğundandır.

Dinamik olarak değişen dosyalar-/var bu dizin farklı dizinler boyunca dağıtılan dinamik olarak değişen dosyaları içerir. Çeşitli programları yerine getirdiğinde kullanıcı çoğu zaman bilgileri oluşturur(çoğunlukla gerçeğin farkında olmadan). Örneğin biçimlendirilmiş man sayfaları sonradan lazım olurlar diye bir süreliğine etrafta tutulurken man komutu sıkıştırılmış kılavuz sayfası kaynaklarını sıkıştırılmamış hale getirmeye sebep olur. Benzer olarak bir belge basıldığında yazıcıya gönderilmeden önce baskı bilgisi depolanmalı, örneğin /var/spool/cups içerisinde. /var/log içindeki dosyalar giriş ve çıkış zamanlarını ve diğer sistem olaylarını kaydeder, /var/spool/cron düzenli otomatik komut yürütmeleri hakkında bilgi içerir ve kullanıcının okunmamış mailini /var/mail'de tutar.

Linux'te sistem log dosyaları genellikle "syslog" servisi tarafından işlenir.sys logd denilen bir program diğer programlardan gelen mesajları kabul eder ve bunları başlangıç ve önceliklerine göre sonraları bulabileceğiniz /var/log dizini altında dosyalar halinde sınıflandırır. Dosyalar dışında syslog servisi aynı zamanda mesajlarını başka yerde yazabilir. Örneğin konsola ya da ağ aracılığıyla

bilgi merkezinizden tüm log mesajlarını birleştiren merkezi bir "yönetim istasyonu" olarak hizmet eden başka bir bilgisayara.

/sys/logd dizininin yanısıra bazı Linux dağıtımları klogd servisi içerir. Görevi işletim sistemi çekirdeğinden gelen mesajları kabul etmek ve onları sys logd dizinine doğru geçirmektir.diğer dağıtımlar kendi sys logd dizinleri bu işi kendileri yapabildikleri için ayrı bir klogd dizinine ihtiyaçları yoktur.

Linux çekirdeği sistem başlatılmadan önce onları kabul edecek sys logd dizinini çalıştırmak için yeterince uzakta mesajların tüm sınıflandırmasını yok eder. Mesajlar hala önemli olduğu için Linux çekirdeği onları içten depolar ve dmesg komutunu kullanarak onlara ulaşabilirsiniz.

Geçici dosyalar-/tmp bazı faydalar geçici dosya alanına ihtiyaç duyar, örneğin bazı editörler ya da türler. Tmp dizininde tüm programlar geçici bilgileri yerleştirirler. Bazı dağıtımlar sistem başlatıldığında tmp dizinini temizlemek için kurulabilirler: Bu yüzden kalıcı önemi olan hiçbir şeyi bu klasöre koymamalısınız.

Alışılmışa göre, /tmp dizini sistem başlangıcında boşaltılır fakat öyle değildir. Dağıtımınızın ne yaptığını kontrol etmelisiniz.

Sunucu dosyaları-/srv aşağıda çeşitli sunucu programları tarafından sunulan dosyaları bulacaksınız.

```
drwxr-xr-x 2 root root 4096 Sep 13 01:14 ftp
drwxr-xr-x 5 root root 4096 Sep 9 23:00 www
```

Bu dizin nispeten yeni bir buluştur ve oldukça mümkündür ki sisteminizde henüz bulunmamaktadır. Ne yazık ki web sayfaları FTP sunucusunun dökümanları için başka bir yer yoktur ki FHS yazarları bunun üzerinde uzlaşabilirler, bu sayede bu dosyalar olmadan bir sistem üzerinde tamamen farklı bir yerde sonlandırabilirler, örneğin /usr/local ya da /var alt dizinlerinde.

CD-ROM ya da Disketlere ulaşım–/media bu dizin çoğunlukla otomatik olarak meydana getirilir; CD-ROM'lar ve disketler için yerleşim noktaları olarak görev yapan /media/cdrom ve /media/floppy gibi ilave edilen bpş dizinleri içerir. Donanım başlangıcınıza dayanarak /media/dvd gibi daha fazla dizinleri eklemekte kendinizi özgür hissedebilirsiniz, eğer bunlar yerleşim noktaları olarak bir anlam ifade ediyorlarsa ve dağıtım satıcınız tarafından önceden yüklenmemişse.

Diğer depolama ortamlarına ulaşım–/mnt bu dizin ilave edilen depolama ortamlarının geçici yerleştirilmesi için yerleşim noktası olarak görev yapar. Red Hat tarafından yapılan bazı dağıtımlarla CD-ROM ve disket

için medya yerleşim noktaları /media'nın altında olmak yerine burada ortaya çıkabilir.

Kullanıcı ev dizinleri–/home bu dizin kök dizini dışında tüm kullanıcıların ev dizinlerini içerir.

Eğer birkaç 100'den fazla kullanıcınız varsa tüm ev dizinlerini home dizinlerinin hemen alttakiler olarak tutunması özel koruma ve etkinlik açısından mantıklıdır. Örneğin daha uzak alt bölünme için ayırıcı bir özellik olarak kullanıcıların başlıca grubunu kullanabilirsiniz.

```
/home/support/jim
/home/develop/bob
```

Yöneticinin ev dizini–/root Yöneticinin ev dizini kök dizininde kuruludur. Bu, dogrudan kök alt dizinin dışında yönetici evinde konumlandırılmayan gözlenen bir farklılıkla diğer kullanıcılara benzer tamamen doğal bir ev yöneticisidir.

Kayıp özelliği lost+found (FHS tarafından yönetilen sadece iç hat dosya sistemleri) bu yönetici herhangi bir yöneticiye ait olmayan ama uygun görünen dosyalar için kullanılır. Dosya sistemini uygunluğunu kontrol edici,aynı dosya sistemi üzerinde ost+found yöneticide öyle dosyalara bağlantılar yaratır. Bu yüzden sistem yöneticisi dosya gerçekten nereye ait bulabilir; lost+found, dosya sistem uygunluğunu kontrol edicinin sabit bir yerin içinde bulabilme ümidiyle yaratılmıştır. Kural gereği, iç hat dosya sistemleri üzerinde bu her zaman dosya endeksi numarasını kullanır.

Dizin düzenlemesi için başka bir yöntem de şöyledir: FHS dosyaları ve dizinleri 2 kritere göre böler-bölgesel olarak ulaşılabilir olmaya ihtiyacı var mı , başka bir bilgisayarda durabilirler mi ve ağ aracılığıyla ulaşılabilirler mi, bağlamları durgun mu ya da sistem çalışırken değişirler mi?

Bu bölünmenin arkasındaki düşünce sistem yönetimini yalınlaştırmaktır: dizinler dosya sunucularına taşınabilirler ve merkezi olarak sürdürülebilirler. Dinamik bilgiler içermeyen dizinler salt okunura yerleştirilirler ve kazalara daha fazla elverişlidir.

Alıştırmalar

1. Sisteminiz "olağan" yerlerde kaç tane program içeriyor?

2. Eğer grep komut satırında birden fazla dosya ismiyle çağrılmışsa, her eşleşen hattın önüne söz konusu dosyanın adını verir. bir kabuk joker kalıbı(.txt gibi) ile grep'i çalıştırmak büyük olasılıkla bir problemdir,

grep çıkışının kesi formatı öngörülemediğinden hangi programların ne şekilde çalışacağını bilemez. Arama kalıbı tek dosya ismini genişletse bile nasıl doya adı çıkışını zorlayabilirsiniz. (ipucu: /dev içerisinde çok faydalı bir dosya bulunmaktadır.)

3. "cp foo.txt /dev/null" komutu temelde hiçbir şey, ama "mv foo.txt /dev/null" uygun erişim izinleri varsayarak /dev/null by foo.txt ile yer değiştirir. Neden?

4. Sisteminizde,(Varsa) hangi yazılım paketleri /opt altına yüklüdür? Hangileri üretici taraından sağlanmıştır ve hangileri üçüncü parti ürünlerdir? Sizce üretici üçüncü parti ürünün teaser versiyonunu /opt içine veya farklı bir yere yüklemeli mi? Ne düşünüyorsunuz?

5. Neden /proc'ta köklü dizin ağacının yedek kopyalarının bulundurulması önerilmez?

10.4 Dizi Ağacı ve Dosya Sistemleri

Bir Linux sistemi dizin ağacı diskler, CD-ROM diskler, USB anahtarları ve taşınabilir MP3ler gibi depolama birimleri üzerinde birden fazla bölüm boyunca uzanır. Microsoft Windows'taki yolunuzu biliyorsanız, muhtemelen bu sorunu Linux üzerinde harfler sayesinde farklı "sürücüler" tanımlaması ile çözebileceğinizin farkındasınız, mevcut tüm disk bölümleri ve medya "/" ile başlayan dizin ağacına entegre edilmiştir.

Genel olarak hiçbir şey sizi tam bir Linux sistemini tek bir sabit disk bölümüne yüklemenizi engelleyemez. Ancak, /home bölümleme dizinini kendi bölümüne koymak yaygındır. Bu yaklaşımın avantajı, Linux dağıtıcınız, tamamen kendi veri güvenliği (basitçe doğru zamanda dikkatli olmanız gerekmektedir, yani dağıtımcı yükleyicisinden yükleme için hedef bölüm(leri) aldığınız zaman.) hakkında endişelenmenize gerek kalmadan sıfırdan fiili işletim sistemini tekrar yükleyebilmenizdir. Bu aynı zamanda yedek kopya oluşturma işlemini kolaylaştırır.

Büyük sunucu sistemleri üzerinde, diğer dizinleri, tipik olan sunucu sistemleri Cally /tmp, /var/tmp'a veya /var/spool'a kendi bölümlerini atamanız da oldukça olağandır. Amaç önemli bölümleri tamamen doldurmayla kullanıcıları rahatsız edici sistem işlemlerini engellemektir. Örneğin /var doluysa, diske herhangi protokol mesajları yazılamasın, bu yüzden kullanıcıları /var/tmp dosyasındaki okunmamış büyük boyutlu maillerden, yazılmamış yazma işlemlerinden veya büyük dosyalardan korumak istiyoruz. Öte yandan, tüm bu bölümler sistemi kaplamak eğilimindedir.

More information and strategies for partitioning are presented in the Linup Front training manual, Linux Administration I.

/etc/fstab dosyası sistemin nasıl çeşitli disk bölümlerinden oluştuğunu açıklar. Başlangıç boyunca, sistem normal bir kullanıcı olarak endişelenmemeniz gereken çeşitli sistem dosyalarını doğru yerlere yerleştiriyor (Linux buna "bağlamak" diyor). Aslında neye ilgi duyduğunuzu, yani nasıl CD-ROM diskler ve USB tuşlarına erişilebileceğinizi ve bunların da monte edilmesi gerekir. Dolayısıyla biz bu başlığı kısaca örterek en iyisini yapmış oluruz her ne kadar gerçekten de kullanıcının ülkesi olsa da.Bir orta monte etmek için, orta için her iki aygıt dosyası ismine (genellikle /dev/sda1 gibi bir blok aygıt) ve ortanın içeriğinin görünmesi gereken dizin ağacındaki bir dizine ihtiyacınız vardır. Bu herhangi bir dizin olabilr.

Başka bir ortayı üzerine bağladığınız orjinal içeriğe erişemeseniz bile dizin boş olmak zorunda değildir (siz ortayı yeniden bağladıktan sonra tekrar görünecektir).

Prensip olarak, birileri /etc gibi nemli sistem dizinine çıkarılabilir ortam monte etmeli (ideal olarak parolasız kök dizini girişi içeren passwd diye adlandırılan). Dizin ağacı içinde rasgele yerlerde dosya sistemlerinin montajı zaten kökü gibi, bu gibi maskaralık gerek olacaktır sistem yöneticisi, sınırlı olmasının nedeni budur. Bu yüzden dosyaları keyfi yerlere dizin ağaçlarıyla birlikte monte etme sistem kullanıcısına kısıtlanmıştır.

Daha önce, biz /dev/sda1 dosyasına "orta aygıt dosyası" denir dedik. Bu gerçekten sistemdeki ilk SCSI diskinin ilk bölümüdür - gerçek isim tamamen kullandığınız ortanın türüne göre değişir. USB anahtarları için hala bariz bir isimdir. Bu bilgiyle-aygıt adı ve bağlama noktası- bir sistem kullanıcısı ortayı aşağıdakileri takiben notme edebilir:

```
# mount /dev/sda1 /media/usb
```

Bu ortamda dosyası olarak adlandırılan bir dosya olarak /media/usb/ dosya görünür olacağı anlamına gelir dizin ağacında. Aşağıdaki gibi bir komutla:

```
# umount /media/usb Note: no ‘‘n’’
```

Kullanıcı aynı zamanda ortayı tekrar bağlayabilir.

Bu Bölümdeki Komutlar

dmesg çekirdeğin arabellek mesajının içeriğini çıkarır

file Kurallar dosyası(1)'e göre dosyanın içeriğinin türünü tahmin eder

free ana bellek ve takas alanı kullanımını görüntüler

klogd Çekirdeik log mesajlarını kabul ede

mkfifo FIFO'ları oluşturur (named pipes)

mknod Aygıt dosyalarını oluşturur

syslogd sistem log mesajlarını işler

Özet

- Dosyaların bir ad altında depolanan verilerinin, kendine yeten koleksiyonları vardır. Linux aynı zamanda diğer nesneler ve aygıtlar için dosya soyutlamayı kullanır
- Bir disk üzerindeki veri ve idari bilgilerin düzenlenmesi yöntemine "dosya sistemi" denir.
- Aynı zamanda dizinlerin ve dosyaların üzerindeki veriler ile birlikte sistemin tam hiyerarşi yapısını veya belirli bir depolama ortamını kapsar.
- Linux dosya sistemleri (iki çeşittir FIFO'lar ve UNİX alan soketleri): düz dosyaları, dizinleri, sembolik bağları ve aygıt dosyalarını içerir
- Dosya sistemi Hiyerarşisi Linux sistemi tarafından birçoğunun anlamını önemli bir dizinlerini ve dağılımlarını açıklar.

Table 10.1: Linux Dosya Türleri

Tür	ls -l	ls -F	Nasıl oluşturulur
düz metin	-	isim	çeşitli programlar
dizin	d	isim/	mkdir
sembolik link	l	isim@	ln -s
aygıt dosyası	b veya c	isim	mknod
FIFO (pipe)	p	name |	mkfifo
UNIX domain socket	s	isim=	komut yok

Table 10.2: FHS'ye Göre Dizin Ayrımı

	Statik	Dinamik
yerel	/etc, /bin, /sbin, /lib	/dev, /var/log
uzak	/usr, /opt	/home, /var/mail

Bölüm 11

Dosyaları Arşivlemek ve Sıkıştırmak

Amaçlar

- "Arşivleme" ve "Sıkıştırma" koşulların öğrenmek
- tar'ı kullanabiliyor olmak
- "gzip" ve "bzip2" kullanarak dosyaları sıkıştırmak ve açabiliyor olmak
- Dosyaları zip ve unzip ile işleyebilmek

Önceden Bilinmesi Gerekenler

- Kabuk yöntemini kullanmak (Bölüm **??**)
- Dosyalar ve dizinleri kullanmak (Bölüm **??**)
- Filtreleri kullanmak (Bölüm **??**)

11.1 Arşivleme ve Sıkıştırma

"Arşivleme" birden fazla dosyayı toplayıp tek bir parça haline getirir. Tipik uygulamalar Manyetik kasete üzerinde bir dizin ağacı depolar. tape—the magnetic tape drive appears within Linux as a device file onto which the output of the archival program can be written. Diğer taraftan, arşivlenen verileri kasetin sürücü cihaz dosyalarından okuyup arşivden çıkartarak dizi ağaçlarını yeniden yapılandırabilir. İlgili programların çoğu, arşivleri çözüp oluşturabildiğinden beri her iki işlemi de arşivleme başlığı altında inceliyoruz.

Burada bizi ilgilendiren kısmı sadece kayıpsız sıkıştırmadır, sıkıştırılmış verilerimizi yeniden inşa ederek orijinal formuna sokabilmemizi sağlar.

Başka bir deyişle, yüksek dereceli bir sıkıştırmada gereksinimleri terk ederek orijinal haline getirmeyi başarabilmesi bir alternatiftir. Buna "Lossy" (kayıplı) sıkıştırma yöntemi denir ve aynen JPEG resimleri ve "MPEG-1" ses katmanlarındaki gibi sıkıştırılırlar. Işin ilginç yanı olan gereksiz verilerden kurtulmanın yolu; Örnek bir mp3 verisini alalım, bu parcalarin bir kisim sinyallerini atalim, "psycho-acoustic model" e dayanarak sadece insanların duyabileceği derecede ve bunu dinleyenlerin anlayamayacağı kadar kayıp vererek kodlamak mümkündür. JPEG bir birlerine benzeyen çizgilerle çalışır.

Çalışma süresinin kodlanması aşağıdaki karakter dizinin temsil eden basit bir örnek alırsak;

```
ABBBBAACCCCCAAAABAAAAAC
```

daha yoğun şekilde

```
A*4BAA*5C*4AB*5AC
```

Burda, "*4B " sıralı dört tane "B" karakter. Bu basit yaklaşım "run-length encoding"(uzun kodlama çalıştırma) olarak adlandırılır ve bugün bile örneğin faks makinelerinde (düzeltmeleriyle birlikte) bulunur. "Real" (gerçek) gzip veya bzip2 givi sıkıştırma yazılımları gelişmiş yeni metotlar kullanırlar.

Bu tür birleşmiş arşivleme ve sıkıştırma yazılımlar Windows dünyasında sık ölçüde kullanırlar. (PKZIP, WinZIP vs), aynı adım genellikle Linux ve Unix dünyasında ayrı olarak ele alınır. Arşivi ilk kez kullanırken tar çıkış sıkıştırma işleminden önce tar ile dosyaların bir dizi olduğunu söylemek popüler bir yöntemdir, gzip-PKZIP ve benzerleri kendi üzerindeki her bir dosyayı sıkıştırır ve sonra bir tek büyük bir dosya içine sıkıştırılmış dosyaları toplar.

Bu yöntemin PKZIP ve bağlantılı yöntemlere göre avantajı bu yöntemin yüksek sıkıştırma oranı veren birkaç orijinal dosya arasında yer alabilmesidir. Ancak bu da şu durumda bir dezavantaj sayılabilir; eğer dosya hasar görmüşse tüm arşiv bu noktadan başlayarak işe yaramaz hale gelebilir.

Doğal olarak Linux'ta bile kimse sizi ilk sıkıştırmanızdan ve onları arşivlemenizden alıkoymaz. Maalesef bu yöntem diğerindeki kadar kullanışlı değildir

Elbette Linux'ta de Windows dünyasındaki rar ve zip gibi kullanışlı sıkıştırma programları vardır.

Alıştırmalar

1. Neden run-length kodlama örneğinde AA yerine *2A i kullanılır?
2. run-length kodlama metodun kullanarak "A*2B****A" dizinini nasıl gösterebilirsiniz?

11.2 tar Kullanarak Dosya Arşivlemek

tar ismi "kaset arşivin 'den türemiştir. Uygulama bireysel dosyaları arşiv dosyasına koyar ve birinden diğerine ek bilgileri not alır (tarih, erişme izni, sahibi, ...). Her ne kadar tar'ın başlangıçta manyetik teyp sürücüleri ile birlikte kullanılabilir olması gereksede, tar arşivleri çeşitli mediyalar üzerine doğrudan yazılmışda olabilir. Diğer kullanımlar arasında tar dosyaları Linux ve diğer özgür yazılım paketleri için kaynak kodu yaymak için standart biçimidir.

Linux üzerinde yaygın olarak kullanılan katman GNU uygulaması diğer Unix türevleri olan tar uygulamalarında bulunan çeşitli uzantıları içerir. Örneğin, GNU tar birçok medyayı kapsayan çok hacimli arşiv oluşturarak bunu destekler. Bu, çoklu cilt arşivler bile küçük arşivler yalnızca çok kirli Tabii ki, diskete yedek kopyalarını sağlar. Elbette ki sadece küçük arşivler için önemli olan bu çoklu cilt arşivleri bile diskete yedekleme yapılmasına izin verir.

Küçük bir açıklama: Bölme (split) komutu arşiv dosyaları gibi büyük dosyaları kesebilmeyi sağlar ve böylece dosya uygun parçalar haline getirilerek disketlere kopyalanabilir ya da e-maille yollanabilir ve cat kullanılarak geldikleri şeklinde yeniden birleştirilebilirler.

tar'ın avantajları: Bunun kullanımı basittir, güvenli ve iyi çalışır, evrensel olarak tüm Unix ve Linux sistemlerinde çalışır. Bunun dezavantajları basit arızalardan dolayı problemlere yol açabilir, ve tar'ın tüm sürümleri aygıt dosyalarını depolayamayabilir. (Sisteminizin tümüne geri yüklemek istediğinizde).

tar arşivleri, dosyalar ve tüm dizin hiyerarşileri içerir. Eğer Windows medyası ağ üzerinden dizin ağaçları içinde takılı ise, içerikleri tar kullanarak arşivlenir. tar kullanılarak oluşturulan arşivler normalde sıkıştırılmamıştır, Ama başka sıkıştırma yazılımlar kullanarak sıkıştırılabilir (bugüne kadar her zaman gzip veya bzip2 kullanılmıştır). Bunlar yedek kopyalamalar ile ilgili iyi bir fikir değildir, sıkıştırılmış arşiv verilerin bit hatalar her zaman geri kalan verilerin kayıplarına yol acar. Typical suffixes for tar archives include .tar , .tar.bz2 , or .tar.gz , depending on whether they have been compressed not at all, using bzip2 , or using gzi . ".tgz" uzantısı zipped (sıkıştırılmış) tar veri formatı genel olarak DOS dosya sistemi üzerinde depolanması gerekiyor. tar 'in söz dizimi;

```
tar <options> <file>||<directory> ...
```

Önemli tar seçenekleri:

-c ("create") yeni arşiv oluşturur

-f file <file> üzerinden yeni arşiv oluşturur (veya mevcut arşivden okur)

<file> düz dosyalarda veya bir aygıt dosyalarda bulunması (diğerlerinden yanı sıra

-M multi-volume arşivleri kullanmak

-r arşive dosyalar ekler (manyetik kaseteler için değil)

-t arşiv içindekileri görüntüler

-u arşiv içindeki kendi sürümünden daha yeni olan dosyaları değiştirir. Eğer dosyaların hepsi oluşmamışsa, bunlar daha önceden eklenmiştir (manyetik kaseteler için değil)

-v Verbose modulu—tam o sırada tar'ın ne yapıyor olduğunu görüntüler

-x Dosyalar ve Dizinleri arşivden çıkarır

-z gzip kullanarak arşivi sıkıştırır veya sıkıştırmadan çıkarır

-Z sıkıştırma kullanarak arşivi sıkıştırır veya açar (normalde linux üzerinde mevcut değil)

-j bzip2 kullanarak arşivi sıkıştırır veya sıkıştırmadan çıkarmamasının option syntax tar'ın söz dizim seçenekleri oldukça sıra dışıdır

Bunda -f gibi parametre olanlar dahil olmak üzere tekbir tire sonra birkaç seçenekleri "paketlemek" (başka bir yerde olduğu gibi) mümkündür.

İlk "seçenek paketi" önünde gösterişi çizgi dışarı bırakabilirsiniz. Sık sık aşağıdaki gibi komutları göreceksiniz:

```
tar cvf ...
```

Yine de, bunu önermiyoruz.

Aşağıdaki örnekte arşivler tüm dosyaları ile birlikte asıl dizinde data isim ile devam eden data. tar dosyası kullanıcının ana sayfasında bulunmaktadır.

```
# tar -cvf ~/data.tar data* data1
data10
data2
data3
```

-c seçeneği yeniden arşiv oluşturup düzenler, "-f /data.tar" isimli arşivin ismini verir. -v seçeneği sonucu hakkında hiçbir değişiklik yapamaz; sadece dosyaların isimleri hakkında arşivlenmiş halini ekranda görüntüler. (eğer dosyalardan birisi arşivlenmişe bu da gerçekten bir dizin ise, dizinin tüm bilgilerine arşivin içine dizinler eklenecektir.

Tar tüm bir dizinin arşivlenmesini de sağlar. Bunu çevreleyen dizinden yapmak daha iyidir. Arşivin içine bir altdizin oluşursa buda arşivi paketten çıkarırken yeniden oluşacaktır. Aşağıdaki örnekte daha detaylı olarak gösterecektir.

```
# cd /
# tar -cvf /tmp/home.tar /home
```

Sistem yöneticisi root home.tar isimli /home dizininde bir arşiv depolarsa (tüm kullanıcı bilgileri). Bu da /tmp dizin içerisine saklanır.

Eğer dosyalar ve dizinlerin mutlak path isimlerin kullanarak verilirse, tar path ismine bağlı otomatik olarak depolar (başka bir sözle "/" ile başlayan her bir isimleri kaldırılmıştır). Başka bir bilgisayarda arşivden paket çıkarırken bu tür problemleri önler (Uygulama 11.6'ya bakın).

Arşivin "tablo içeriklerini" -t seçeneğin kullanarak görüntüleyebiliriz:

```
$ tar -tf data.tar data1
data10
data2
```

-v seçeneği tari daha detaylı verir:

```
$ tar -tvf data.tar
-rw-r--r-- joe/joe            7 2009-01-27 12:04 data1
-rw-r--r-- joe/joe            8 2009-01-27 12:04 data10
-rw-r--r-- joe/joe            7 2009-01-27 12:04 data2
```

-x komutu ile verileri açabilirsiniz:

```
$ tar -xf data.tar
```

bu durumda tar terminal üzerinde hiç çıktı üretmez. -v parametresini kullanmanız gerekir:

```
$ tar -xvf data.tar data1
data10
data2
```

Eğer arşiv hiyerarşi bir dizin içerirse, bu ana dizini yeniden oluşturur (Eğer hatırlarsanız tar tüm mutlak isimlerinden bir tane path ismine bağlı üretir). Herhangi dizine arşivleri çıkartabilirsiniz, her zaman bu yapıyı tutar.

Paketi çıkarma sırasında dosya veya dizin isimlerini değiştirebilirsiniz. Bu durumda söz konusu olan sadece dosyalar ve dizinler paketten açılacaktır. Ancak, arşivin içindeki isimlerin tam olarak eşleşmesine dikkat edin:

```
$ tar -cf data.tar ./data
$ tar -tvf data.tar
drwxr-xr-x joe/joe          0 2009-01-27 12:04 ./data/
-rw-r--r-- joe/joe          7 2009-01-27 12:04 ./data/data2

$ mkdir data-new                                    ./ missing
$ cd data-new
$ tar -xvf ../data.tar data/data2
tar: data/data2: Not found in archive
tar: Error exit delayed from previous errors
```

Alıştırmalar

1. Ana dizinizdeki dosyaların liste bilgilerin bir dosya içinde depolayın. O dosyanın tar arşivini oluşturun. Orijinal dosya ve onun arşivini bir biriyle karşılaştır. Ne önerirsiniz?

2. Üç veya dört boş dosya oluşturup onu yeni oluşturduğunuz arşive ekleyiniz

3. Asıl dosyanızı silin ve tar'ın içindekileri arşivden çıkartın.

4. Eğer bir dosya veya muhafaza edilmesi için dizinin adı mutlak bir yol adı olarak verilirse neden GNU tar profilaktif olarak yol adı başlangıcında silmez? ve etc-backup.tar (a) mutlak yol isimleri varsa ne olacağını hayal edin,(b) başka bir bilgisayara aktarılır ve orada çözülür.

11.3 gzip ile Dosya Sıkıştırma

Linux in en yaygın kullanılan sıkıştırma uygulaması Jean-Loup Gailly ve Mark Adler tarafından geliştirilmiştir. Bu tek bir dosya sıkıştırması için kullanılmıştır (yakınlarda adı geçmiştir, çokça dosyaları tek arşivde içermesi mümkündür).

gzip uygulaması ("GNU zip"in kısaltılmışı) 1992'de sıkıştırma problemleri önlemek için ortaya çıkmıştır, standart Unix sürümlerin özel sıkıştırma aracısıdır. Sıkıştırma Lempel-Ziv-Wlch algoritmasına dayanmıştır (LZW), patent hakları US tarafından saklıdır 4,558,302. Bu patenti Sperry (later Unisys) şirketine ait olup bu da 20 Haziran 2003 de süresi sona ermişti. Diğer taraftan, gzip DEFLATE metodu Phil Katz [RFC 1951] tarafından kullanılmıstır, artık LZW patentine sahip olmamasına dayalı bunun yanı sıra LZ77 isimin alan Huffmen schema kodlaması özgür patent olmuştur. ayrıca, LZW den daha iyi çalışmaktadır.

gzip compressi kullanarak sıkıştırılmış dosyaları sıkıştırmadan çıkarır, çünkü Sadece Unisys tarafından patenti korunmuştur. Bu tür dosyaları ".Z" uzantıları ile ayırt edebilirsiniz.

gzip PKZIP ve benzer Windows uygulamaları "ZIP" ismiyle karıştırılmamalıdır. Bu tür uygulamalar derhal dosyaları sıkıştırıp ardından arşivleyebilir; gzip sadece sıkıştırma işiyle uğraşır ve arşivleme işini tar veya cpio uygulamalarına bırakır .—gzip, ZIP paketleri tam olarak hangi bir dosya DEFLATE yöntemle arşivlenmiş bunları arşivden çıkarabilir.

gzip bir tek dosyaları isler ve değiştirir, dosyaların uzantı isimlerine. gz ismini ekler. Bu değişikler dosyanın orijinal halinden daha az hafıza tutar. Birkaç dosya bir tek arşiv içine sıkıştırılmalıdır, tar ve gzip bileşik olmalıdır.

gzip in en önemli tercihleri şunlardır:

-c sıkıştırılmış dosyasını standart sıktıysa gönderir, asil dosyanın yerine konur; asilin kalanı değişmemektir.

-d sıkıştırılmış dosyanı çıkarmak (öteki türlü: gunzip gzip -d gibi çalışır)

-l ("list") sıkıştırılmış dosyanın adı, paket boyutu gibi önemli bilgilerini görüntüler

-r ("recursive") alt dizindeki dosyaları sıkıştırır

-S <suffix> uses the specified suffix in place of .gz

-v her bir dosyanın isimi ve sıkıştırma oranının çıktılarını gösterir

-1 ... -9 Bir bir sıkıştırma oranı belirtir -1 (veya –fast) Çok hızlı çalışır ama iyi sıkıştırmaz, -9 (veya –best) çok iyi sıkıştırır ama yavaş; varsayılan ayarları ise -6'dır.

Aşağıdaki komutu letter.tex dosyasını sıkıştırır, letter.tex.gz isimli sıkıştırılmış dosya depolar ve asıl dosyası silinir:

```
$ gzip letter.tex
```

Dosyayı paketten açarken

```
$ gzip -d letter.tex
```

ya da

```
$ gunzip letter.tex
```

letter.tex.gz (-S .t) yerine letter.tex.t olarak sıkıştırılmış dosya ve dosyanın sıkıştırma oranın cıktısı (-v):

```
$ gzip -vS .t letter.tex
```

-s komutu sıkıştırmadan cıkarma üzerine belirtir, "gzip -d" olana kadar a .gz :

```
$ gzip -dS .t letter.tex
```

Eğer tex uzantılı tüm dosyaları all.tar.gz şeklinde sıkıştırmak istersek, aşağıdaki komut kullanılır

```
$ tar -cvzf tex-all.tar.gz *.tex
```

Unutmayın; tar asıl dosyayı silmez! Bu sadece paketten çıkarmak için kullanılır.

```
$ tar -xvzf tex-all.tar.gz
```

Alıştırmalar

1. Alıştırma 11.3 deki en iyi sıkıştırması ile tar arşivin sıkıştırın.
2. Sıkıştırılmış arşivin içeriklerini denetlemek. tar arşivinden asıl yerine geri getirin.
3. Nasıl bir şekilde ana dizininizdeki tüm içerikleri paketleyip gzip dosyasını sıkıştırır?

11.4 bzip2 ile Dosya Sıkıştırmak

bzip2 Julian Seward tarafından geliştirilen büyük ölçüde gzip'e uyumlu bir uygulamadır. Ancak, bu başka yöntemler içinde kullanılır yüksek derece fazladan zaman ve hafıza kullanılarak sıkıştırır (sıkıştırmadan açarken de bir farkı yoktur).

bzip2 "Burrows-Wheeler dönüşümünü" kullanır kodlamada sık sık ortaya çıkan dizinleri tek karakteristik dizilere çevirir. Bu sonuçlar "local frequency" e tek karakteristik olarak sıralanır, çalışma uzunluğu hesaplandıktan sonra, Huffman şemasına göre kodlanır. Huffmen kodu yoğun bir şekilde dosyaya yazılır.

Bzip'den ne haber? bzip aritmetiksel kodlama kullanan bzip2 ye göre daha önce çıkmış olup dönüşümü Huffman kodlamasından sonra bloklamıştır. Ancak, geliştiricisi aritmetiksel kodlamaya karar verdi bu nedenle onu çevreleyen uygulamalar çeşitli patent sorunlar yaşadı.

gzip, bzip2'ler sıkıştırmak için bir veya birden fazla dosya isimlerin parametre olarak kabul eden .bz2 uzantı isimile biten dosyalar, sıkıştırılmış sürümlerin yerine geçer.

-c ve -d seçenekleri eponymous seçeneklerine karsı gelip. Ancak, "seçeneklerin kalitesi" -1 den -9 a farklı çalışır: onlar o anki sıkıştırmasın kullanarak blok

boyutun belirler. Varsayılan değeri 9 dur, -1 olunca önemli bir hız kazancı sunmaz.

-9 900kib boyutta bloklar. Bunun karşılığı hafıza kullanımı yaklaşık 3.7 MiB Sıkıştırır (7.6 MiB sıkıştırır), çağdaş bir donamın sorun çıkarmamalıdır. Blok Boyutlarının artırılmasından bir avantaj elde edilebilir gibi görünmüyor. Sıkıştırma üzerinde blok boyutu seçimini vurgulamakta yararı sıkıştırmadan çıkarmada gerekli olan hafıza miktarı boyutunu belirler, aklınızda tutmanız gerekeni eğer çok az hafızalı bilgisayarda çok byte bilgisayarla .bz2 dosya hazırlamada bzip2 (1) daha ayrıntılı olarak açıklar.

Karşılaştırdığımızda da gzip ve gunzip,bunzip2 dosyaları açmak için, bzip2 de dosyaları sıkıştırmak için kullanılır. (Bunlar gzip2'in başka bir ismidir: Hem "bzip2 -d " kullanarak ta dosyaları açabilirsiniz.)

11.5 zip ve unzip kullanarak Dosya Arşivleme ve Sıkıştırmak

Windows da veya internette veri alış verişte genellikle widespread ZIP dosya biçimi kullanılmalı (bugünlerde Windows üzerinde birçok dosya arşiv programları da .tar.gz ile uyumludur). Linux'da iki ayrı uygulama vardır zip (arşivleri oluşturma) ve unzip (arşivleri açma).

Dağıtıma baglı olarak bu uygulamaları ayrı ayrı kurmanız gerekir. Debian GNU/Linux'da, örneğin, zip ve unzip iki ayrı paket bulunur.

Zip programının arşivleme tarzı size PKZIP gibi programların sıkıştırma tarzını hatırlatabilir. En basit değişle, komut satırına geçirilen dosyaları toplar:

```
$ zip test.zip file1 file2
adding: file1 (deflated 66%)
adding: file2 (deflated 62%)
$ _
```

(Burada test.zip çıkan arşivin adıdır.)

-r seçeneğini kullanarak zip'e alt dizinlerin içine inmesi komutunu verebilirsiniz.:

```
$ zip -r test.zip ziptest
adding: ziptest/ (stored 0%)
adding: ziptest/testfile (deflated 62%)
adding: ziptest/file2 (deflated 62%)
adding: ziptest/file1 (deflated 66%)
```

-@ seçeneğiyle zip kendi standartları çerçevesinde arşivlenecek dosyaların adlarını okur:

```
$ find ziptest | zip -@ test
adding: ziptest/ (stored 0%)
adding: ziptest/testfile (deflated 62%)
adding: ziptest/file2 (deflated 62%)
adding: ziptest/file1 (deflated 66%)
```

(Arşiv dosyasının adında. zip son ekini kullanmayı unutabilirsiniz.) zip bir arşive dosya eklemenin 2 yolunu bilir. Dosyayı sıkıştırma olmadan depolanmış kabul eder ve saklanır (sıkıştırılmış dosyanın yüzde kaç sıkıştırıldığını mesela "% 62 düşürülmüş", arşiv içinde kendi orijinal dosyanın boyutu düşürülmüş iken sadece %38 olduğu anlamına gelmektedir). siz -0 seçeneğini kullanmadıkça zip otomatik olarak daha mantıklı bir yaklaşım seçer.

Eğer var olan bir zıp dosyasını onun ilk parametresiyle çağırır ve başka bir şey belirleyemezseniz, arşivlenecek dosyaların (aynı adları taşıyan mevcut dosyaların üzerine yazılmış olan) mevcut içeriğinin üstüne arşive eklenir. Bu durumda zıp, tar ve çipodan farklı davranır. Temiz bir arşivleme isterseniz ilk olarak dosyayı kaldırmanız gerekir.

Dosyaları aptalca eklemenin yanı sıra, zıp operasyonun diğer birkaç morlarını da destekler: -u seçeneği arşivi komut satırında belirtilen dosyayla aynı ismi taşıyan bir dosya daha varsa sadece yeni olanı arşiv dosyalarına ekleyerek günceller(adı geçen dosyalar henüz arşive hiçbir şekilde alınmamışsa).-f seçeneği arşiv dosyalarını komut satırında üzerine yazılmış daha yeni versiyonlarıyla canlandırır (bütünüyle yeni dosyalar arşive eklenmez). -d seçeneği arşiv içindeki dosyaları siler ve kişilerin isimleri komut satırında dosya adlarını değerlendirir.

zıp'in yeni sürümleri aynı zamanda -FS (dosya senkronizasyon) modunu destekler. Bu mod -u'nun yaptığını yaparak arşivi dosya sistemine senkronize eder, fakat aynı zamanda komut satırı üzerindeki henüz isimlendirilmemiş dosyaları siler(ya da -r durumunda aranan bir dizinin parçasıdır). Bu metodun avantajı yeniden yapılandırılmış bir arşivinkiyle karşılaştırıldığında arşivde önceden var olan değişmemiş dosyaların tekrar sıkıştırılmış olmasını gerektirmez.

zip her çeşit seçeneği destekler ve "zip -h" yi kullanarak listeye bakabilirsiniz (ya da -h2'yi kullanarak daha ayrıntılı bir listeye bakabilirsiniz). Aynı zamanda zip(1) man sayfası da çok bilgilendiricidir.

zip her çeşit seçeneği destekler ve "zip -h" yi kullanarak listeye bakabilirsiniz (ya da -h2'yi kullanarak daha ayrıntılı bir listeye bakabilirsiniz). Aynı zamanda zip(1) man sayfası da çok bilgilendiricidir.

Arşivin için gözetmenin en iyi yolu -v seçeneğin kullanmak, içinde ne olduğunu gormek altdizinler için zorluk çıkarabilir.

Bu tablonun düzenlenmesi lazım

```
$ unzip -v test                  The.zip suffix maybe omitted
Archive: Length --------
0 16163 18092 35147 --------
69402
test.zip Method ------ Stored Defl:N Defl:N Defl:N
Size Cmpr ------- ---- 00% 6191 62% 6811 62% 12119 66% ------- ---
25121 64%
Date ---------- 2012-02-29 2012-02-29 2012-02-29 2012-02-29
Time CRC-32 ----- -------- 09:29 00000000 09:46 0d9df6ad 09:01 4e46f4a1
Name
----
ziptest/ ziptest/testfile ziptest/file2 ziptest/file1 -------
4 files
```

unzip le arşivin ismini çağırması arşivi paketten açmasına karşılık gelir:

```
$ mv ziptest ziptest.orig
$ unzip test
Archive: test.zip
    creating: ziptest/
  inflating: ziptest/testfile
  inflating: ziptest/file2
  inflating: ziptest/file1
```

-d komutunu kullarak asıl dizinden başka bir dizin içine arşivi paketten açmak. Zorunlu kalsa bu dizin yeniden oluşturulur:

```
$ unzip -d dir test
Archive: test.zip
    creating: dir/ziptest/
  inflating: dir/ziptest/testfile
  inflating: dir/ziptest/file2
  inflating: dir/ziptest/file1
```

Eğer komut satırında belirli dosya isimi varsa, o zaman sadece bu dosyaları paketten çözülecektir:

```
$ rm -rf ziptest
$ unzip test ziptest/file1
Archive: test.zip
   inflating: ziptest/file1
```

(bu durumda ziptest dizini ine oluşacaktır)

Alternatif olarak, -x komutun kullanırsanız seçilmiş olan dosyalar haricindekiler paketten çözülmüş olacaklar:

```
$ rm -rf ziptest
$ unzip test -x ziptest/file1
Archive: test.zip
  creating: ziptest/
 inflating: ziptest/testfile
 inflating: ziptest/file2
```

Ayrıca kabuk arama desenleri kullanarak paketi çözebilirsiniz (veya paketten çözmektende önleyebilirsiniz) :

```
$ rm -rf ziptest
$ unzip test "ziptest/f*"
Archive: test.zip
  inflating: ziptest/file2
  inflating: ziptest/file1 $ rm -rf ziptest
$ unzip test -x "*/t*"
Archive: test.zip
    creating: ziptest/
  inflating: ziptest/file2
  inflating: ziptest/file1
```

(Note the quotes, which are used to hide the search patterns from the actual shell so unzip gets to see them.) Unlike in the shell, the search patterns refer to the complete file names (including any "/").

As is to be expected, unzip also supports various other options. Look at the program's help information using "unzip -h" or "unzip -hh", or read unzip(1).

Alıştırmalar

1. Ana dizinize birkaç dosya oluşturun ve onları zip ile arşivleyin. "unzip -v" kullanarak arşivin içeriğine bak. /tm dizinine arşivi paketten çıkarın

2. unzip kulanarak paketten çözüyor olduğunuz dosya eğer dosya sistem içerisinde daha önceden oluşmuşsa ne olurdu?

3. zip arşivi file.zip iki altdizinleri barındırıyor olsun bunların içinde (örnegin .c, .txt, .dat). Öyle bir unzip komutun verin ki içerikler içinden .txt dosyaların ayrı bir yerde açsın.

Bölüm 12

Sistem Yönetimine Giriş

Amaçlar

- Sistem yöneticisi rolü hakkında fikir sahibi olmak
- İşletim sistemi çekirdeği ve süreçlerin temllerini anlamak
- Paket yönetimi kavramlarını bilmek

Önceden Bilinmesi Gerekenler

- Temel kabuk kullanımı (Bölüm **??**)
- Linux dosya sistemi yapısı ile ilgili bilgiler (Bölüm **??**)

12.1 Sistem Yönetimi Temelleri

Bir sistem yöneticisi ne yapar? Bilgisayarları yapılandırma, yazılım kurulumu (ve bazen kaldırmak) çevre birimlerini bağlamak ve kullanılabilir yapmak, yedekleme yapmak (ve bazen bunları geri yüklemek) kullanıcı hesaplarını eklemek ve kaldırmak, sorunları olan kullanıcılara yardım etmek, bu görevlerin etkileyici bir listesidir. Eski günlerde kişisel bilgisayarda, bilgisayar kullanıcısı aynı zamanda yöneticiydi. bu fikir windows gibi sistemlerde uzun süre devam etmiştir (hatta birkez windows farklı kullanıcılardan fikir edinmiştir, yönetici hesaplarını kullanmak için ortak olmak zorunludur, çünkü sadece çeşitli programaların ilgili ayrıcaklıkları kullanılabileceği varsayılmıştır). Unix –linux'un ilham aldığı sistem– bazı kullanıcıları desteklemek için tekrar kuruldu ve dolayısıyla normal kullanıcılarla yönetici arasındaki ayrım ve bilgisayarı geleneğiyle işletim sistemi ve bunun gibi çok daha köklü bir hal aldı.

Lpi Linux Temelleri sınavında sistem yönetiminden vurgulanmamaktadır fakat en azından genel bir bakış olmalıdır - belki sistem yöneticisi olmamaktadır, fakat sizin için sistem yöneticisinin ne yaptığının daha iyi anlamanızı sağlar ya da bir noktada sistem yöneticisi olur. LPI LPIC sertifika parça bölümleri sistem yöneticileri için önemli bir noktadır. Özellikle LPIC-2 ve LPIC-3.

Bu bölümde bir Linux bilgisayarın hemen kullanımı ile ilgili daha az olan birkaç konudan söz edeceğiz. Fakat örneğin bir resim elde etmek için bilgisayarda ne çalıştıracağını ("neden bilgisayarım yavaş") ve yazılımın bilgisayarda nasıl yönetildiğini içerir. Biz burada ki büyük resimden sonra detaylara bakmıyoruz.

Linux sisteminde, sistem yöneticisi özel bir kullanıcı hesabı erişimine sahiptir, root. Bu hesap erişim kontrollerinden muaftır, aksi yapıldığında da (Bölüm **??**). Bu nedenle sistemden bütün dosyalara erişilebilir. Bu gereklidir, örneğin, yeni yazılım kurmak için - "Normal" kullanıcıların sistem dizinleri program dosyalarını okuma ve yürütme olabilir. Fakat, diğer kullanıcıların zararına manipülasyonlar önlemek amacıyla, bunlar yazılmamalıdır. Yöneticinin ayrıca yedek kopyalarını oluşturmak için herhangi bir kullanıcının dosyalarını okuması gerekir (ve bir yedeği geri yüklemesi gerekiyorsa bunları geri yazması için).

Sistemdeki tüm dosyalara yazma yetkisinin sisteme ciddi zarar riski içerdiği açıktır. Eğer root olarak oturum açıyorsanız, Linux

```
# rm -rf /
```

gibi tüm sistem dosyalarını yok eden bir komutu kullanmanızı engelleyemez (ve zarar verebilmek için daha incelikli yolları vardır). Dolayısıyla sadece onlara ihtiyacımız olduğunda root yetkilerinden yararlanmak gereklidir. root olarak giriş yaptığında RIGHT OUT olduğunda web'de sörf yapılabilir veya posta okunabilir.

Eğer root olarak sistemdeki tüm dosyaları okuyabilirseniz, günaha yenik düşebilirsiniz, diyelim ki, patronunuzun (veya eşinizin) epostasını düzenli olarak kontrol edebilirsiniz. Yapmayın. Bu güvensiz olabilir (en azından eş durumunda) ve/veya yasadışı (en azından patronun durumunda) ve linux ve sistem yönetimi ile eğlenceli yağma yükümlü olduğundan bu sorunun her türlü içine alınabilirsiniz. Evde ve işyerinde huzur için söylenecek bişey. Katılan kişi ile iştişare içinde izole kısa bir gözetlemeye almakta yanlış birşey yoktur. Fakat bunun alışkanlık haline gelmesine izin vermeyin. Şüpheniz olduğunda, Peter Parker'ı düşünün, a. k. a. Spider-Man: "büyük güç büyük sorumluluk getirir."

root olarak doğrudan girişten sakınmalısınız (özellikle de grafiksel bir ekran üzerinde). Bunun yerine, root olarak çalışan bir kabuk elde etmek için, normal bir kullanıcı olarak başlatılmış bir terminal oturumunda, su komutunu kullanın.

```
$ /bin/su -
Password: secret Password for root
# _
```

root kabuğundan çıkıldıktan (exit veya +d kullanarak) sonra başlangıçta su çağrılan kabukta tekrar sonlandırılır. Bazı dağıtımlarda başka bir root hesabı olmadan elde edilmeye çalışılır. Ubuntu örneğin de olduğu gibi sistem sırasında oluşturulan ilk kullanıcının önüne sudo koyarak root ayrıcalıklarıyla tek bir komutta yürütmeyi sağlar.

```
$ sudo less /var/log/syslog Peruse system log
```

(Gerekirse bu kullanıcı diğer kullanıcılar için bu ayrıcalığı genişletebilir.) Büyük görevler için, yönetici ayrıcalıklarıyla çalışan bir kabuk elde etmek için "sudo -i" kullanmak mümkündür.

Çoğu linux dağıtımı sinyali root haklarıyla çalışan olduğunda kabuk "#" istemiyle biterek çıktılanır. Başka bir şey görüntülenirse—genellikle "$" veya ">", kabuk ayrıcalığı olmadığında.

Alıştırmalar

1. su komutunu deneyin. Neden örnek programı çağırmak için bir mutlak yol adı kullanıyor?

2. su için root parolasını bilmeniz gerekir. sudo genelde kendi parolanızı sorar. Hangisi daha iyi?

12.2 Sistem Yapılandırması

Diğer sistemlerde sadece veritabanında özel araçlar yoluyla değiştirilmiş ve "bit-rot" (windows kayıt defteri) duyarlı olan kendi yapılandırma bilgilerini gömerken, linux genellikle /etc dizini içindeki metin dosyalarında sistem genelindeki yapılandırma girişlerini bulundurur (Birkaç örneği Bölüm **??**'de görülebilir). Burada sistem yöneticisi bunları değiştirmek veya genişletmek için kendi seçtikleri bir metin düzenleyicisi kullanabilir. Örneğin, yeni bir kullanıcı, kullanıcı adı, sayısal kullanıcı kimliği veya /etc/passwd dosyasına ev dizini ismi gibi ilgili parametreleri katarak eklenebilir. Yeni bir sabit disk aygıt dosyası adını ve disk görünmesini gerekenr dizini belirten bir satırına /etc/fstab ekleyerek yapılandırılmış olabilir.

Bir linux sistemi yaygın farklı kaynak yazılım bileşenlerinin karmaşık bir sistemidir. Bunların bazıları linux kendisinden daha yaşlıdır) Tarihsel olarak gelişmiş kurulumlar /etc içindeki farklı yapılandırma dosyaları çok düzgün olmayan bir biçimde yapılandırmasını takip eder. Bazıları satırlar tarafından organize edilmektedir, diğerleri bölümleri ayraçlar ile ayrılmış olarak içerir,

hatta diğerleri XML dosyalarını ya da yürütülebilir bir kabuk betiğidir. Kesinlikle tüm bu farklı biçimleri ile uğraşması gerekir, yöneticiler için bir sıkıntıdır. Yazılım paketleri her türlü değiştirilmesi gerekir fakat bu kolay değildir.

Bununla, birlikte bazı çok yaygın usuller: Örneğin, çoğu yapılandırma dosyalarında "#" ile başlayan satırlar açıklama satırlarıdır.

Sistem yapılandırmasını farklı metin dosyaları ile yönetme fikri başlangıçta eski moda bir fikir gibi görünse de bunun bazı somut avantajları vardır:

- Tek bir yazılım paketi veya hizmet yapılandırması hataları ile bütün olarak genellikle sisteme zarar verilmesi mümkün değildir (Sistemin işlevseliği için çok gerekli yapılandırma dosyaları vardır tabi ki onların hataları olabilir, örneğin, sistemi yeniden önyükleme yapamayan hale getirebilir. Ama bunlar en iyi ihtimalle küçük bir azınlıktır).

- Çoğu yapılandırma dosyası yorumlara izin verir. Bu sayede bireysel yapılandırma dosyalarının detaylarının nerede meydana geldiğinin doğrudan belgelemesini sağlar ve böylece ekip işbirliğini daha kolay yapar ya da kişinin kendi unutkanlık nedeniyle olan kazalar engellenir. Bu kesinlikle hatırlamak zorunda kalmanızdan çok daha iyidir menu y bir x girişi olduğunda sekme z bir iletişim açmanızı sağlar. Kesinlikle kontrol edilmesi gereken bir kutusu vardır. Aksi halde hiçbirşey çalışmaz (kağıt parçalarının bu tür bilgeliği vardır. En çok ihtiyaç duyulduğunda kaybolma eğilimi vardır).

- Git veya Mercurial gibi sürüm denetim sistemini için metin dosyalarını "kontrol" edebilir ve böylece sadece çeşitli dosyaları kapsayan büyük değişiklikler belgelenemez. Fakat aynı zamanda gerekirse bunları düzenli bir şekilde geri alır. Bu aynı zamanda merkezi bir sunucu üzerinden bir bilgisayarın tam Yapılandırmasını depolamak için uygunluğu sağlar. Böylece bilgisayarın herhangi bir nedenle yeniden yüklenmesi gerekirse derhal kullanılabilir - diyelim ki felaket bir donanım hatasında sonra. Veri merkezlerinde bu çok avantajlar sağlar, özellikle tüm yapılandırma değişikliklerinde ayrıntı bir "denetim" istediğinde.

- Metin dosyaları yönetilmesi gereken bilgisayarlara merkezi bir sunucudan yapılandırma dosyaları dağıtarak tüm bilgisayar ağlarının uygun şekilde yapılandırılmasını sağlar. "puppet" veya "salt " gibi sistemlerde yapılandırma dosyaları için "şablonlar" yapmak mümkündür. Onlar tamamen bireysel bilgisayarların elle yapılandırmasını önlemek için dağıtıldığı zaman hedef bilgisayar için uygun bilgilerin kullanıldığı örneklemedir. ("sneaker net") Bu da, büyük ağların yönetimini kolaylaştırır aynı zamanda daha küçük yüklemeye de yardımcıdır.

Alıştırmalar

1. /etc dizinini araştırın. Pek çok dosyayı oradaki kılavuz sayfalarını "man fstab" gibi birşey ile deneyin. /etc nin içindeki dosyalar normal kullanıcı olarak okunamaz mı, neden ?

12.3 Süreçler

Yürütülmekte olan bir programa "süreç" denir. Program kodunun kendisinin yanında (söz konusu işlemci için makine dilinde) bir süreç şu anda kullanımdaki dosyalar gibi verilerin yanı sıra idari bilgi için çalışan depolama ortamı içerir. (ortam değişkenleri için) Geçerli bir dizin ve bir işlem numarası veya sistemi içinde benzersiz süreci tanımlayan "PID". İşletim sistemi çekirdeği süreçlerin oluşturulmasından, işlemci zamanı ve depolama atamasından ve çıktıktan sonra temizlemeden sorumludur. Süreçler dosyaları, aygıt ya da ağa erişmek için işletim sistemi çekirdeğinin içine çağırabilir.

Yeni süreçler mevcut süreçler sırasında ortaya çıktı. Bakteri veya yaşamın diğer düşük biçimlerinin aksine iki neredeyse özdeş kopyalarına ayrıldı. ("neredeyse özdeş", çünkü bir süreç "ana" ve diğerleride "çocuk" olarak kabul edilir.) Buna ek olarak, süreci farklı bir programı çalıştırmak için ayarlayabiliriz: örneğin, kabuktan ls komutu çağıracaksak, kabuk programı çalıştırmadan önce çocuk süreç oluşturulur. Bu kod (diğer şeyler arasında) olası bir giriş/çıkış yönlendirme düzenlemesi yapar ve sonra /bin/ls program dosyası ile yenilenir. ls programın sonunda çocuk süreç sona erer ve kabuk size sonraki komutu sorar.

PID ile ilk işlem, önyükleme sırasında işletim sistemi çekirdeği tarafından oluşturulur. Kurala göre bu, /sbin/ init programıdır ve aynı zamanda "init süreci" olarak da adlandırılır. init süreci sistem önyüklemesinden ve örneğin, arka planda çalışan sistem servisleri için ek işlemleri başlatılmasından sorumludur.

ps Sistem üzerinde çalışan süreçler hakkında bilgi edinmek için "ps" komudunu kullanabilirsiniz. En basit şekilde, ps komutu, mevcut terminalinizde (ya da günümüzde, grafik ekranınız üzerindeki mevcut terminal penceresinde) çalışan tüm süreçleri gösterir.

```
$ ps
PID TTY STAT TIME COMMAND 997 pts/8 S 0:00 -bash
1005 pts/8 R 0:00 ps
$_
```

PID ve COMMAND kolonları kendilerini belirtir. TTY terminale isim verir ("pts/something" genellikle terminal penceresi anlamına gelir), TIME

şimdiye kadar süreçler tarafından kullanılan işlemci zamanı ve STAT "süreç durumu"dur.

Linux'ta bir süreç, aşağıdaki durumların biridir. Yani:

Çalıştırılabilir (R) Sürece işlemci zamanı tahsisi edilmiş olabilir.

Uykuda (S) Sürec genellikle bir tuşa basma ya da diskten veri okuma gibi giriş/çıkış olaylarını bekler.

Derin (bölünemez) uykuda (D) Süreç bir olay bekliyor ve rahatsız edilemez. Süreçler çok uzun süre bu durumda kalmamalıdır çünkü sadece sistem önyükleme tarafından silinebilirler. Eğer böyle olursa, bu genellikle bazı hatalardan kaynaklanmaktadır.

Geçici olarak durdurulmuş (T) Süreç geçici olarak sahibi veya bir yönetici tarafından durdurulabilir, ama daha sonra çalışmaya devam edebilir.

Zombi (Z) Süreç gerçekte tamamlanmıştır, ancak çıkış kodu henüz ana süreç tarafından yakalanamamıştır. Bu süreç "ölmek" anlamına gelir ama sistem içinde ölümsüz kalır. Gerçeğin aksine zombiler sorun değildir çünkü onlarda, süreç tablosunda bir yuva başka kaynakların yerini kaplayamaz. Eğer sisteminiz çok fazla zombiyle kaplıysa, bu ilk etapta süreçleri oluşturan program ile ilgili bir sorun olduğunu gösterir. Bu program sonlandırmada zombileri yok etmek gerekir.

ps bilgilerini sağlayan kontrol parametrelerini kullanın. Örneğin, belirli bir süreç hakkında bilgi edinmek için bir işlem numarası girebilirsiniz:

```
$ ps 1
PID TTY STAT TIME COMMAND
1 ? Ss 0:00 init [2]
```

l seçeneği, bir işlem hakkında daha ayrıntılı bilgi verir:

```
$ ps l $$
F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND
0 1000 3542 3491 20 0 21152 2288 - Ss pts/8 0:00 /bin/bash
```

("$$" kabukta "mevcut süreci" gösterir)

UID, sürecin sahibinin sayısal kimliği (bkz. Bölüm **??**), PPID Sürecin "ana" bir süreç kimliği. PRI önceliği- sayı yükseldikçe öncelik düşecektir(!)- VSZ Çalışma belleğinde boyut (KiB içinde) ve RSS RAM içindeki geçerli büyüklük (üstelik KiB içinde).

Sürecinin bir parçası diske taşınmış olabilir, böyle bir durumda VSZ ve RSS, özdeş değildir. Bir disk bölümünde veya dosya üzerinde takas alanı ekleyerek Linux bilgisayarında mevcut çalışan belleği büyütebilirsiniz.

ps komutu dikkate alınan süreçlerin seçimini ve her süreç için bilginin çıktı boyutunun, tipinin kontrol edilmesinde bir çok seçeneği destekler. ps(1) kılavuz sayfasını okuyun.

ps ve benzer programlar genellikle /proc üzerine monte edilen ve işletim sistemi çekirdeği tarafından uygun hale getirilen proc dosya sisteminden bilgileri elde eder. Bu dizin içinde "dosyalar" süreçleri ve sistemin diğer özellikleri hakkında güncel bilgileri içerir. (Ayrıca bkz. Bölüm **??**)

free Free komutu sistem belleği hakkında bilgi sağlar:

```
$ free
          total       used       free     shared buffers  cached
Mem: 3921956 1932696 1989260          0      84964  694640
-/+ buffers/cache: 1153092 2768864
Swap:    8388604       0 8388604
```

"Mem:" satırı bilgisayarın yaklaşık 4 GB RAM'e sahip olduğunu (altında "toplam"; işletim sistemi çekirdeği burada görünmeyen bazı belleği de doldurur) ve de yaklaşık yarısının dolu olduğunu söyler ("used" ve "free" alanlarına bakın). İşletim sistemi diskte veri depolamak için yaklaşık 700 MiB kullanır ve ikinci satır size bu boş ve kullanılan hafızayı nasıl etkilediğini söyler. Üçüncü satırda takas alanı ("Swap:") kullanımı anlatılmaktadır (Bu makinede 8 GB, üzerinden).

Modern Linux makinelerde "paylaşılan" sütunu her zaman sıfırdır; ve bu nedenle ihmal edilebilir.

free komutu da çeşitli seçenekleri destekler. Örneğin; dostça bir çıkış biçimi üretmek için:

```
$ free --human  ‘‘-h’’ de olur
total used free shared buffers cached
Mem: 3,7G 1,9G 1,8G 0B 84M 678M
-/+ buffers/cache: 1,2G 2,5G
Swap: 8,0G 0B 8,0G
```

Burada free komutu bilgisayar dostu mebibyte ve gibibyte'ı ifade etmek için "M" ve "G" birimleri kullanır. –si seçeneği on yetkilerine geçiş yapar (mega ve gigabyte).

top Son olarak, "top" komutu sürekli güncellemeleri ile ps ve free komutlarının bir birleşimi gibidir. Sistem ve süreç bilgilerini içeren bir bilgi tam ekran görüntülenir: Şekil 12.1 de bir örnek gösterilmektedir:

şekil 12.1 eklenecek

- Çıktının üst kısmında yer alan, birinci sıra şimdiki duvar saati zamanını ve "uptime"ı gösterir. Örneğin, sistemin yeniden başlatılmasından sonra geçen zamanın süresi (burada, dört gün, on dört saat, ve değiştir) ve giriş yapmış olan kullanıcıların sayısı (burada "11" çok ciddiye alınmamalı; son penceredeki her oturum bir kullanıcı gibi sayılır). Sağ tarafta üç numara bulunmakta, sözüm ona sistem yüklemelerini tanımlayan yükleme ortalamaları.

  Yükleme ortalamaları yürütülebilir işlemleri (state R) belirtir, son dakikanın, son beş dakikanın ve son on beş dakikanın ayrı ayrı ortalamasını alır. Bu değerlerin faydası gözde büyütülmemeli!; size gerçekten o kadarını söyleyemezler. Eğer son dakikanın değeri yüksekse ve son 15 dakikadan biri düşükse, bu durumda sisteminiz aniden daha fazlasını yapmak zorunda kalacak; eğer son dakikanın değeri düşük fakat son 15 dakikanın yüksekse, sisteminiz yapılacak çok şey için kullanır fakat bu şu an aşılmış durumdadır.

  Eğer yükleme ortalamaları sürekli olarak sisteminizdeki işlemci çekirdekleri sayısından aşağıdaysa, bunun anlamı siz gereksiz yere pahalı bir işlemci için fazla harcama yapmışsınız demektir. Sekiz çekirdekli sistem üzerinde, örneğin, 8 civarındaki değerler (bu geleneksel olarak bir sistem yöneticisinin omurgasından soğuk parçaları düşürecektir tamamiyle normaldir); uzun bir zaman periyodu boyunca değerler 8'in çok altındaysa bu çok acıklı bir durumdur.

- İkinci satır işlemlerin sayısını ve çeşitli işlem evrelerinde nasıl dağıldıklarını verir.

- Üçüncü satır CPU kullanım tipine göre yüzdeler içerir: "us" çalıştırılan kodun çıktısı ve "sy işletim sisteminin çekirdeğindeki koddur.

- Takip eden iki satır temelde -free- nin çıktılarına tekabül eder.

- Ekranın alt tarafı "ps l" e benzer işlem listesidir. Üst kısım gibi her birkaç saniyede güncellenir ve aksi istenmediği taktirde, CPU zamanı kullanım yüzdelerine göre işlemleri sıralamıştır (sistemin üzerinde en çok vakit harcadığı işlem liste başı olur)

  Eğer m tuşuna basarsanız, liste hafıza kullanımına göre sıralanacaktır - en çok yer kaplayan işlem en üstte olur. p tuşu ile, CPU zaman listesine geri dönebilirsiniz.

  h tuşunu top içinde yardım sayfasını görüntülemek için kullanabilirsiniz. Top ana sayfası çıktıları ve olağan tuş kombinasyonlarını açıklar ve size işlem listesi içeriklerini gereksinimlerinize göre nasıl adapte edeceğinizi gösterir.

Alıştırmalar

1. ps komutunun ax seçeneği ile sistemdeki bütün işlemleri görüntüleyebilirsiniz. Listeye göz atın. Hangi işlemleri farkettiniz?

2. shell oturum içinde uzun yürütülen bir işlem başlatın ("sleep 120" e benzer yapılabilir). Başka bir oturumda "ps ax" ı çağır ve bu süreci çıktıya yerleştirmeyi dene. (Hint: grep arkadaşınızdır).

3. top'ı kullanarak hangi sürecin şu an, en çok CPU zamanını kullandığını bul. Hangi işlem en çok hafıza kullanıyor?

12.4 Paket Yönetimi

Modern Linux dağıtımları normalde kalabalık (tipik olarak binlerce) her biri sistemin işlevselliğinde (çalıştırılabilir programlar, kitaplıklar, belgelendirme,...) belirli bir bölüm için gerekli her şeyi barındıran "paketler" içermektedir. Başlangıçta Linux bilgisayarı yapılandırırken, siz yönetici olarak hangi paketlerin bilgisayara yükleneceğini belirleyebilirsiniz ve tabiki daha sonra yayımcınızdan keyfi olarak paketleri her zaman ekleyebilirsiniz yada kullanılmayanları kaldırabilirsiniz.

İşlevselliğin paketler şeklinde nasıl bölündüğünün ayrıntıları dağıtıma bağlıdır. Kütüphanelerle genelde bir "run-time paketi" ve bir "geliştirme paketi" arasında ayrım yapılır. Run-time paketi diğer programların kütüphaneyi kullanabilmesi için yüklenmesi zorunlu olan dosyaları içerir (bir .so dosyası içinde asıl dinamik olarak yüklenilebilir kütüphane gibi). Geliştirme paketini kurmaya sadece eğer, kütüphaneyi kullanan yeni yada var olan programları derlemeyi hedeflerseniz, ihtiyaç duyarsınız. C derleyicisi bilgileri içerir bu durum kütüphaneyi kullanmaya ihtiyaç duyar ("include files"), hata ayıklama için istatiksel olarak ilişkilendirirlebilir bir kütüphane, ya da kütüphane içeriği hakkında belgelendirme. Eğer belgelendirme büyükse başka bir pakete bölünebilir.

Örneğin, burada Debian GNU/Linux 6.0'a ("Squeeze") göre rsvg kütüphanesi (SVG format grafikleri ile ilgilenen) paket bölünmüş:

```
librsvg2-2  Gerçek (run-time) kütüphanesi
librsvg2-dev  geliştirme paketi
librsvg2-bin  Command-line programs
librsvg2-dbg  Komut satırı programları
librsvg2-doc  belgeleme
librsvg2-common  Daha fazla Komut satırı programları
python-rsvg  Python dili bağlama
libimage-librsvg-perl  Perl dili bağlama
```

Her linux bilgisayarında [1], bilgisayarın farkında olduğu ve yakın zamanda yüklenmiş olan paketler hakkında bilgiler içeren bir "paket veritabanı" vardır. Dağıtıcınızın "depoları" ya da paketler içeren sunucu ile paket veritabanınızı periyodik olarak eşzamanlayabilirsiniz ve bu şekilde bilgisayarınızdaki paketlerden hangilerinin güncel olmadığını bulabilirsiniz çünkü dağıtımcılar yeni versiyonları önerirler. Paket yönetim sistemimi genelde size söz konusu paketleri seçerek güncelleme fırsatı verir.

Bunun pratikte ne kadar iyi işleyeceği (bir kez daha) dağıtımınıza bağlıdır. Özellikle, konu göründüğünden daha karmaşık olabilir: bir paketin yeni versiyonu bir kütüphanenin (kendi paketinde kullanılabilir olan) mutlaka yeni versiyonda yüklenilmesine ihtiyaç duyabilir ve eğer kurulu olan başka bir program tam olarak bu kütüphanenin eski versiyonuna ihtiyaç duyarsa bu problemlerin ortaya çıkmasına sebep olabilir. Bazen bir paketin sistemde başka bir yerde ciddi değişiklikler yapmadan güncellenebilmesi mümkün olmayabilir. İyi paket yönetim sistemleri bu gibi durumları saptar ve yönetici olarak sizi uyarır ve/veya size müdahale etme şansı sunar.

Bölüm **??**'de de belirtildiği gibi, büyük Linux dağıtımları farklı paket yönetim sistemlerinden herhangi birini kullanabilirsiniz. Her iki paket yönetimi sistemi de kendi araçlarıyla ve paket dosyaları için kendi formatıyla gelen Debian GNU/Linux'un paket yönetim sistemi ve türetimleri, Red Hat, SUSE kullanılarak RPM paket yönetimi gibi sistemlerdir. Prensip olarak, aynı sorunu çözmek, ancak paket yönetimi için kullanılan komutlar gibi ayrıntıda farklılaşır.

Örneğin; RHEL, Fedora veya openSUSE gibi RPM tabanlı sistemlerde, yüklenen tüm paketlerin listesini aşağıdaki komutu kullanarak görüntüleyebilirsiniz.

```
$ rpm --query --all  ‘‘-qa’’
```

Debian tabanlı sistemlerde bu komut yerine aşağıdaki komut gereklidir.

```
$ dpkg --list  ‘‘-l’’
```

Paket veritabanları genellikle /var/lib altında bulunur. Debian gibi sistemlerde /var/lib/dpkg (/var/cache/apt yüklenen her paket dosyası ile birlikte depo sunucularının içerik tablolarını içerir) ve RPM tabanlı sistemlerde /var/lib/rpm içindedir.

Bugün, dpkg ve rpm gibi programlar bir paket yönetim sistemi "temelini" oluşturur. Yöneticier temel programlar üzerine inşa edilen daha uygun araçlar tercih ederler ve bu programlar örneğin paket depolarına kolay erişim ve paketler arasındaki bağımlılıkların otomatik çözümlerini içerir. Debian dünyasında, "Aptitude" ve "Synaptic" kullanılırken, RPM tarafında Red Hat Zypper üzerinde

[1] Those using one of the major distributions, at any rate—there are a few distributions that seem to get by without a package management system, but these are for nerds.

YUM ve SUSE diye adlandırılan bir program kullanılır(paket yönetimi genel yönetici aracı YaST içine entegre edilmiş olsa bile).

Bu araçlardan bazıları alttaki paket yönetim sistemi ile bağımsızdır. "PackageKit" de örneğin, Debian veya RPM paket yönetiminin her ikisi de kullanılmayabilir. Bundan başka kontrollü şartlar altında paket güncelleştirmesi veya yüklenmesi için yönetici ayrıcalıkları olmadan izin verebilir.

Alıştırmalar

1. Sisteminizde kaç paket yüklü? Yukarıda gösterilen rpm ve dpkg çağrısını kullanın ve çıkış satırlarını sayın. (Dikkat: "dpkg –list" de yüklü olması gereken paketleri görüntüler fakat daha yeni sürümleri silebilir veya yerine geçirebilir. "ii" ile başlayan çıktı satırlarını sayar.)

Bu Bölümdeki Komutlar

dpkg Debian GNU/Linux paket yönetim aracı

free Serbest ana bellek ve takas alanı kullanımını görüntüler

ps Çıktıların süreç durum bilgisi

rpm Çeşitli Linux dağıtımları tarafından kullanılan paket yönetim aracı

su Farklı bir kullanıcının kimliğini kullanarak bir kabuk başlatır

sudo Normal kullanıcıların, yönetici ayrıcalıklarına sahip belirli komutları çalıştırmasına izin verir

top Süreç yönetimi ve kontrolü için ekran yönelimli araç

Özet

- Linux "normal" kullanıcı ve sistem yöneticisini root ile birbirinden ayırır. root ile olağan ayrıcalık kontrolleri muaf edilir.
- Normal bir kullanıcı olarak, su veya sudo ile geçici yönetici ayrıcalıkları elde edebilirsiniz.
- Bir Linux bilgisayarın konfigürasyonu /etc dizini içindeki metin dosyaları içinde yer alır.
- Süreçler yürütülmekte olan programlardır.
- ps ve top gibi komutlar mevcut sistem durumunu daha yakından görmemizi sağlar.

- Önemli Linux dağıtımları da GNU/Linux ya da orjinal Red Hat tarafından geliştirilen RPM paket yönetim sistemi kullanır.
- Temel araçlara dayanarak çoğu dağılımlar, yönetmek, yükleme ve bağımlılıkları dahil yazılım paketleri silebilmeniz için uygun yazılımı sunarlar.

Bölüm 13

Kullanıcı Yönetimi

Amaçlar

- Linux'un kullanıcı ve grup konseptini kavramak
- Linux'ta grup ve kullanıcı bilgilerinin nasıl depolandığını öğrenmek
- Kullanıcı ve grup yöneticiliğini komutlarını kullanmayı öğrenmek

Önceden Bilinmesi Gerekenler

- Düzenleme dosyalarını kullanma hakkında bilgi

13.1 Temeller

13.1.1 Neden Kullanıcılar?

Bilgisayarlar eskiden büyük ve pahalıydı ama bugünlerde bir çalışma ofisini bilgisayarsız düşünmek neredeyse olanaksız ve artık hemen hemen her evin odasında bir bilgisayar mevcut. Bilgisayar sistemi, bir aile için annenin, babanın ve çocukların dosyalarını farklı dizinlere koymaları için yeterli olabildiği gibi farklı kullanıcıları birbirinden ayırmaya ve bu kullanıcılara bazı özel haklar vermeye elverişli olmalıdır.Şirketin maaş bordroları bilgilerine bakan Geliştirme Departmanından Ms. Jones'un işi gibi, gelecek senenin ürünleri hakkında detaylı bilgiye erişen İnsan Kaynaklarından Mr Smith'in de işi var. Ve evde olmalarına rağmen gizli kalması her ikisi için de önem arz ediyor - Örneğin Noel hediye listesi yada genç bir kız çocuğunun günlüğü doğal olarak meraklı gözlerce açılmaması gerektiği gibi.

Genç bir kızın günliğünün Facebook üzerinden bütün dünyaya açık olduğu gerçeğini göz ardı edeceğiz, konu bu olsa dahi, bütün dünya genç kızın günlüğüne

kesinlikle hiç birşey yazmayacak. (Ki Facebook bile farklı kullanıcıların fikirlerini destekliyor.)

Farklı kullanıcıları birbirinden ayırma özelliğinin öneminin bir diğer sebesi ise çeşitli haklardan yoksun, daha az değiştirilebilir ve bilgisayar sisteminin, çeşitli açılardan görünür olmaması gerçeğinden gelmektedir. Bu yüzden Linux, sistem yöneticisi için kullanıcı şifreleri ve bilgilerinin ortak kullanıcılardan gizli tutmasını sağlayan ayrı bir kullanıcı kimliği kullanır (root). Eski windows sistemlerinin bir sıkıntısı- e-mail aracılığıyla yada rastgele internette gezinirken elde edilen ve bütün sistem üzerinde hasara yol acan programlar - Linux'ta başınıza gelmeyecektir çünkü ortak kullanıcı olarak bilgisayarda uyguladığınız herhangi birşey bütün sisteme zarar veren bir işlem sıfatında olmayacaktır.

Fakat bu tamamen doğru değildir: Arada bir normal kullanıcılara yöneticilere atanan özel yetkileri yapmasını sağlayan bir hata açığa çıkar. Bu tür hatalar son derece kötü/tehlikelidir ve genellikle bulunduktan sonra çabucak düzeltilebilir, fakat göz ardı edilmemesi gereken bir husus vardır ki bu tür bir hata (bug) uzun bir süre boyunca sistemde saptanmamış olarak kalmış olabilir. Bu yüzden Linux'ta (diğer işletme sistemlerinde olduğu gibi), dağıtıcınızın desteklediği kritik sistem parçalarının, örneğin çekirdek gibi, en son sürümlerini çalıştırmak/yüklemek zorundasınız.

Linux, normal kullanıcılar tarafından gerçekleştirilen yetkisiz girişlerden sistem ayarlarını korusa da sizi sistemin beynini kapatmanız için yanıltmamalı. Size önerimiz (Örneğin: Kullanıcı grafik ara-yüzüne root olarak girmemek gibi). 'X' Websitesine girmenizi ve kredi kartı numaranızı soran ek olarak kredi kartınızın şifrenizi girmenizi isteyen E-posta mesajları size Linux'tayken bile ulaşabilir ve başka yerde olduğu gibi aynı şekilde bu tür e-posta mesajlarını dikkate almamanız gerekmektedir.

Linux kullanıcı hesapları vasıtasıyla farklı kullanıcıları birbirinden ayırır. Ortak dağıtıcılar yükleme sırasında genelde 2 tane kullanıcı hesabı açar, şöyle ki yönetimsel görevler için root ve normal kullanıcı için ise farklı bir kullanıcı hesabı. Siz (yönetici olarak) daha sonra daha fazla kullanıcı hesabı ekleyebilirsiniz yada daha geniş bir şebekedeki istemci bilgisayarda başka bir yerde depolanan kullanıcı bilgileri otomatik olarak karşınıza çıkabilir.

Linux, kullanıcı hesaplarını ayırır, kullanıcıları değil. Örneğin, kimse sizi web'te dolaşmak ya da e-postaları okumak için kullandığınız farklı bir kullanıcı hesabını kullanmaktan alı koymaz. Eğer internetten yüklediğiniz şeylerin önemli bilgilerinize ulaşmayacağından emin olmak istiyorsanız biraz ustalıkla web tarayıcısı ve e-posta programını normal programlar [1] arasında 'Dolaşma hesabı' adı altında çalıştırabilirsiniz.

Linux altında, her kullanıcı hesabına, kullanıcı kimliği adında (UserID

[1] Which of course is slightly more dangerous again, since programs runninig on the same screen can communicate with one another

ya da kısada UID), tek bir numara atanmıştır. Üstelik her kullanıcı hesabı metinsel kullanıcı ismi (root veya joe) gibi insanların akıllarında daha kalıcı olmasını kolaylaştıran özellikler taşır. Çoğu yerde sayım yaptığı zaman (Örneğin; giriş yapıldığında, yada dosyaların ya da sahiplerinin listelerinde) Linux, münkün mertebede ilk olarak metinsel ismi kullanacaktır.

Linux çekirdeği metinsel kullanıcı isimleri hakkında herhangi birşeyden haberdar değildir; dosya-sistemindeki işlem-verisi ve sahiplik-verisi sadece Kullanıcı kimliği'ni kullanır (UID). Bu, eğer kullanıcı hala sistemde dosyaları mevcutken UID'yi silmişse ve UID başka bir kullanıcıya atanmışsa bazı sorunlara yol açabilir. Sonuç olarak, bu kullanıcı bilgisayara giren bir önceki kullanıcıdan geriye kalan dosyalara sahip olur yani bir nevi ona kalır.

Başka kullanıcı isimlerine aynı (rakamsal) UID (Kulanıcı kimliği) atamak herhangi bir teknik probleme yol açmaz. Bu kullanıcılar, UID'ye ait bütün dosyalara eşit anlamda erişim iznine sahiptir fakat her kullanıcı özel bireysel şifrelerini diledikleri gibi oluşturabilirler. Fakat bunu yapmanız önerilmez yaparsanız da dikkatli olmanız gerekmektedir.

13.1.2 Kullanıcı ve Gruplar

Bilgisayarla çalışmak için öncelikle giriş yapmanız gerekmektedir. Bu sistemin sizi tanımanıza ve kullanıcı hesabı için belirlenmiş yetkilerin size atanmasını sağlayacaktır. Oturumunuz boyunca (Giriş ve çıkış yaptıgınız ana kadar) yaptığınız herşey kullanıcı adınız altında gerçekleşir. Ek olarak, her kullanıcıya dosyalarını depolayabileceği ve yöneteceği ve başka kullanıcıların genellikle okumaya ve yazmaya izinlerinin olmadığı ait 'Ev Dizin'leri vardır. Sadece sistem yöneticisi (root) bütün dosyaları okuyabilir ve yazabilir).

Hangi Linux dağıtımını kullandığınıza bağlı olarak (Cue: Ubuntu), sisteme açıkça giriş yapmanıza gerek olmayabilir. Çünkü bilgisayar sizi tanıyacaktır ve nedeninin bu olduğunu farzedecektir. Sistem güvenliği yerine kolaylığı tercih ederseniz, bu özel işlem sadece bilgisayarınıza erişebilecek bir başka kişinin olmadığı zaman mantıklı olabilir.

Bazı kulanıcılar belirli sistem kaynaklarına ve dosyalarına giriş izni sağlamak için bir grup oluşturabilirler. Linux, grupları ya belirlenmiş sabit isimleriyle yada kullanıcılar için geçici giriş işlemlerine benzer yöntemlerle belirler. Grupların kullanıcılar gibi 'ev dizin'leri yoktur ama yönetici olarak isteğe bağlı olarak belirli guruplara göre makul yetkilerle directory'ler oluşturabilirsiniz.

Gruplarda rakamsal isimlerle tanılanmıştır (Group IDs yada GIDs)

Kullanıcı isimlerinin Kullanıcı Kimlikleriyle ilişkilendirildiği gibi, Grup isimleride Grup Kimlikleriyle ilişkilendirilir: Linux Çekirdeği sadece ilk GrupKimliğini tanır ve işlem bilgisindeki yada sistem dosyasındaki ilk GrupKimliğini depolar.

Her kullanıcı, birincil gruba ve muhtemel ikincil veya ek gruplara bağlıdır. Ortak bir ortamda,

Erişim kontrolü amaçları için,bütün gruplar eşit ağırlıktalar ve her kullanıcı her zaman üyesi olduğu gruplardan elde ettiği yetkileri sonuna kadar kullanıyor. Birincil ve ikincil gruplar arasındaki tek fark kullanıcı tarafından yakın zamanlarda oluşturulan dosyalar genellikle [2] kendi birinci grubuna atanmıştır.

Linux Kernel'in 2.4'cü sürümüne kadar, bir kullanıcı ek olarak en çok 32 grubun üyesi olabiliyordu. Linux'un 2.6'cı versiyonundan itibaren ikincil grupların sayısı sınırsız olarak değiştirildi.

Kullanıcıların UID'lerini (Kullanıcı kimliklerini),birincil ve ikincil grupları ve GIDs eşlerini/aynılarını (Grup kimliklerini) aşağıdaki ID programı aracılığıyla bulabilirsiniz.

```
**$ id
uid=1000(joe) gid=1000(joe) groups=24(cdrom),29(audio),44(video),_
_ 1000(joe)
$ id root
uid=0(root) gid=0(root) groups=0(root)
```

-u, -g, ve –G seçenekleri ile, ID nin kendisi hesapların UID'sini , birincil grubun GID'sini yada ikinci grubun GID'sini vermek için kendini zorlar. (Bu seçenekler kombine edilemez) Ek olarak –n seçeneğiyle numaralar yerine ID lerin isimlerini elde edersiniz.

```
$ id -G
1000 24 29 44
$ id -Gn
joe cdrom audio video
```

'groups' komutu "id –Gn" komutu gibi aynı sonucu verir. 'last' komutu ile bilgisayarınıza kimin giriş yaptıgını ve ne zaman yaptığını öğrenmek için kullanabilirsiniz (veya ağ vasıtasıyla giriş yapılması durumunda nereden olduğu hakkında bilgi almak için)

```
$ last
joe pts/1 pcjoe.example.c Wed Feb 29 10:51 still logged in
bigboss pts/0 pc01.example.c Wed Feb 29 08:44 still logged in
joe pts/2 pcjoe.example.c Wed Feb 29 01:17 - 08:44 (07:27)
sue pts/0 :0 Tue Feb 28 17:28 - 18:11 (00:43)

-----
reboot system boot 3.2.0-1-amd64 Fri Feb 3 17:43 - 13:25 (4+19:42)
```

Ağ'daki oturumlar için, üçüncü sütun istemci/ana bilgisayarın ismini belirler. ":0" grafiksel ekranı gösterir (Birinci X sunucusunu, emin olmak için – çünkü 1'den fazla olabilir)

[2] The exception occurs where the owner of a directory has decreed that new files and subdirectories within this directory are to be assigned to the same group as the directory itself. We mention this strictly for completeness.

Bilgisayarın ne zaman başladığını gösteren “reboot” girdisi/komutunu’da not alın. 3. Sütun “uname-r” ile sağlanan Linux işletim sistemi çekirdeğinin sürüm numarasını barındırır.

Kullanıcı ismiyle, “last” komutu belirli kullanıcı hakkında bilgi elde etmemizi sağlar.

```
$ last
joe pts/1 pcjoe.example.c Wed Feb 29 10:51 still logged in
joe pts/2 pcjoe.example.c Wed Feb 29 01:17 - 08:44 (07:27)
```

Bu gizli bilgi rastgele bir biçimde sıradan kişilerin kullanımına sunulabiliyor ve bu da sizin açınızdan bir takım problemlere yol açabilir. Linux dağıtıcınızın kullanıcıların veri gizliliğini korumak için koyduğu standart ayarları değiştirip kullanıcıların bilgilerini daha iyi korumak istiyorsanız aşağıdaki komutu kullanarak sayaç bilgilerine bakan ’last’ komutu dosyasının genel okuma yetkisini kaldırabilirsiniz.

```
# chmod o-r /var/log/wtmp
```

Yönetici izni olmayan kullanıcılar ise komutu kullandıklarında aşağıdaki değerleri elde edeceklerdir.

```
$ last
last: /var/log/wtmp: Permission denied
```

13.1.3 İnsan ve Pseudo-kullanıcılar

Normal kullanıcılardan başka -sistemi kontrol eden insanlar- kullanıcı ve grup kavramı, sistemin çeşitli parçalarına bazı yetkiler vermek içinde kullanılır. Bu şu anlama geliyor,gerçek kullanıcıların kişişel hesaplarının yanısıra, insan kullanıcılara benzemeyen başka hesaplarda vardır ama bunlara yönetici işlev yetkileride verilmiştir. Bu hesaplar kendi grupları ve hesaplarıyla bir nevi kendilerine işlevsel görevler belirlerler.

Linux işletim sistemini yükledikten sonra, /etc/passwd ve /etc/group dosyalarında bir kaç tane grup ve sahte kullanıcı hesabıyla karşılaşabilirsiniz.Bildiğiniz üzere en önemli görev root (Yönetici) kullanıcısı ve adını verdiği gruplarındır. Root’un UID (Kullanıcı kimliği) ve GID (Grup kimliği) numaraları sıfırdır.

Root (Yönetici) yetkileri UID 0’a bağlıdır, GID 0’a herhangi bir özel yetki verilmemiştir.

Bazı yazılım sistemlerine, bilgisayar parçalarına ya da cihazlara (örneğin; yazıcılar, kasetler, ya da floppy sürücüler) ait olan sahte kullanıcılar (örneğin; Usenet-News INN adında kullanıcı ismi vardir yada Postfix posta servisi için postfix adı altında ek olarak birer kullanıcı ismi açılmıştır). Eğer gerekliyse, ’su’ komutu ile diğer kullanıcı hesaplarında olduğu gibi bu hesaplarada erişebilmeniz mümkün. Bu yan kullanıcılar, yönetici hesabı olmadan dosya

sahipliğine bağlı özel gereksinimlere erişim yetkileri sağlamak için dizin ve dosya sahipleri kadar önemli ve gereklidirler. Aynı gruplara Appkies'ler; disk grubunun üyeleri, örneğin, sistem disklerine erişmelerini engellemek için bazı engeller vardır.

Alıştırmalar

1. işletme sistemi çekirdeği çeşitli kullanıcı ve grupları birbirinden nasıl ayırır?
2. Eğer bir UID (Kullanıcı kimliği)'ye 2 farklı kullanıcı ismi verilmişse ne olur? Bu mümkün müdür?
3. Sahte kullanıcı nedir? Örnek veriniz.
4. (ikinci metinde) Root (Yönetici) güvenmek istemeyeceğiniz şifreli bir grup diskine kullanıcı atamak sizce kabul edilebilir mi? Neden?

13.2 Kullanıcı ve Grup Bilgisi

13.2.1 /etc/passwd dosyası

/etc/passwd dosyası sistem kullanıcı veritabanıdır. Sistem üzerindeki her kullanıcı için bir girdi vardır, Linux kullanıcı isminin özelliklerini taşıyan bir dize, ya da 'gerçek' bir isim vesaire. Sistem ilk yüklendikten sonra,dosya sahte kullanıcıların bir çoğunun girdilerini sistemde barındırır.

/etc/passwd dosyasındaki girdiler aşağıda bulunana formatta ki gibidirler.

```
<user name>: <password>: <UID>: <GID>: <GECOS>: <home directory>: <shell
```

Bu isim küçük harflerle ve sayılardan oluşturulmalıdır; ilk olarak bir harf kullanılmalıdır. Unix sistemleri genelde ilk 8 harfi dikkate alır, Linux'ta böyle bir sınırlandırma yoktur fakat çeşitli ağlarda dikkate alınması gerekmektedir.

Çift noktaları, noktalama işaretlerini ve kullanıcı isimlerinde ki özel harfleri sistem izin verse bile kullanmayınız. Kullanıcı isimleri oluşturan bütün araçların hepsi seçici değildir ve /etc/passwd dosyasını elinizle değiştirebilirsiniz. İlk bakışta sorunsuz çalışıyor görünsede ileri ki zamanlarda farklı yerlerde çeşitli problemlere yol açabilir.

Sadece sayılardan ve büyük harflerden oluşan kullanıcı isimlerini kullanmaktan kaçınınız. İlk olarak sahiplerine giriş yaparken sorun yaratır ve eğer rakamsal kullanıcı ismi hesabın rakamsal UID (Kullanıcı kimliği) siyle eşit değilse karışıklığa sebep olabilir. 'ls -l' gibi komutlar '/etc/passwd' dosyasında UID'ye karşılık eşdeğer bir girdi olup olmadığını gösterir, ve 'ls' çıktısında UIDS'in tamamen rakamsal kullanıcı isimlerinden ibaret olduğunu söylemek doğru olmaz.

Genellikle, bu alan kullanıcının şifrelenmiş parolasını barındırır. Bugün, çoğu Linux dağıtıcıları, şifreleri bazı özel yetkili programlar ya da yönetici tarafından erişilebilip, herkes tarafından okunan /etc/shadow dosyasında toplanan, /etc/passwd dosyasında saklamak yerine 'gölge parolalar' kullanırlar. "/etc/passwd" dosyasında bir 'x' biraz dikkat gerektirebilir? Her kullanıcı 'passwd' programınından faydalanarak şifresini değiştirebilir.

Rakamsal kullanıcı tanımlayıcısı -0 ile 232 arasında bir sayı- 1. Genelde, 0 ila 99 (99 dahil) arasındaki UID numaralar sistem için ayrılmıştır, 100 ile 499 arasındaki sayılar ise yazılım paketlerinin sahte/yan kullanıcılar oluşturması için kullanılır. Yaygın dağıtıcılarda gerçek kullanıcıların UID numaraları 500 ya da 1000'den itibaren başlar. Çünkü, sistem kullanıcıları isimleriyle değil UID (Kullanıcı kimlikleriyle) birbirinden ayırt eder.Çekirdek, farklı isimlere sahip fakat UID leri aynı olan iki hesap varsa (Konu özel yetkilerle alakalı olduğu sürece) iki hesabıda tamamen aynı şekilde değerlendirir. Kullanıcı isimlerini gösteren komutlar ("'ls -l'" ya da 'id') giriş yaparken kullanıcının hangi ismi yada UID yi kullandığını gösterir.

Kullanıcının birinci grubunun giriş yaptıktan sonraki GID'si.

Novell/SUSE dağıtıcıları (diğerleri arasında) bütün kullanıcıların tıpkı paylaşılan birinci grup gibi kullanıcılara benzeyen tek bir grup oluşturur. Bu yöntem kolay kavranabileceği gibi oldukçada geliştirilmiştir(pekiştirilmiştir.

Debian GNU/Linux ya da Red Hat gibi diğer birçok dağıtıcı ne zaman sistemde bir hesab oluşturulursa hesabın UID (Kullanıcı kimliği) leriyle aynı olan GID (Grup kimliği) lerle yeni bir hesap oluşturur. Bunun amacı ise bütün kullanıcıları aynı grup kullanıcıları yerine koyan daha gelişmiş yetkiler vermektir.Aşağıdaki duruma bir göz atın: Jim (kullanıcı ismi jim) Yönetici Sue'nun (kullanıcı ismi sue) kişisel yardımcısıdır. Bu durumda Sue'nun başkalarının giriş yetkisinin bulunmadığı ev dizinine erişim yetkisinin bulunması gerekiyor.

Red Hat ve Debian&Co'nun 'bir grup bir kullanıcı' yöntemine göre Jim gruba girebilir ve Sue'nun dosyalarını diğer grup üyeleri düzenleyip okunmalarını sağlayabilir fakat başkaları için değil. 'Herkes için tek bir grup' yöntemiyle de en baştan yeni bir grup oluşturulması ve Jim ve Sue'nun hesaplarını uygun bir şekilde tekrardan ayarlamak gerekecekti.

/etc/passwd dosyasındaki atamalarda, her kullanıcı bir grubun üyesi olmak zorundadır.

Kullanıcının ikincil/diğer grupları (eğer geçerliyse) /etc/group dosyasındaki girdiler tarafından belirlenir.

Bu açıklama alanı 'GECOS Alanı' olarakta bilinmektedir.

GECOS - "General Electric Comprehensive Operating System" yani Genel elektrik kapsamlı işletim sisetmi anlamına gelmektedir ve Linux'la hiç bir alakası yoktur fakat Unix eskiden bu alanı Gecos uzaktan iş girdi servisi için uyumluluk verisi saklamak için /etc/passwd dosyasına eklerdi.

Bu alan kullanıcı hakkında çeşitli bilgileri, 'reel' ismi ve telefon numarası

ya da ofis numarası isteğe bağlı veriler gibi bilgileri içerir. Bu bilgiler 'Mail' yada 'Finger' gibi programlar tarafından kullanılır. Tam isim, genellikle 'news' ya da 'mail' yazılımı sayesinde gönderici adresinde görülebilir.

Teorik açıdan,GECOS alanımızdaki içeriği (kullanıcı olarak) değiştirmenize yarayan 'chfn' adında bir program vardır. İşe yarayıp yaramadığı tartışmalara yol açsa da, ortak bir ortamda kimse insanların isimlerini istedikleri gibi değiştirmelerine izin vermek istemez.

Bu dizin kullanıcıların dosyalarını depolayabilecekleri kişisel alanlarıdır.Yeni oluşturulan dizin kesinlikle boş değildir, çünkü kullanıcı genellikle temel gereçler açısından profil dosyalarının numaralarını alır. Kullanıcı giriş yaptığı zaman,kullanıcı kabuğu ana dizinini o anda geçerli olan dizini olarak kullanır örneğin kullanıcı giriş yatığı anda dosyaları burada depolanır.

Programın ismi doğrulamayı geçtikten sonra 'login' komutuyla başlatılır, bu genellikle kabuktur. 7.alan en son satır boyunca genişler/yayılır.

Kullanıcı bu girdiyi 'chsh' programı yardımıyla değiştirebilir. Uygun programlar /etc/shells dosyasında gösterilmiştir. Eğer kullanıcının parametleri olan etkileşimli bir kabuğu (shell'i) yok ise genelde yaygın olan '/bin/true' klasörüne girilebilir. Bu alan genelde /bin/sh kabuğunun başlatılması durumunda boş bırakılabilir.

Eğer çizgisel bir ortamda oturum açarsanız etkileşimli kabuk (shell) gerekli olmadığı sürece kullanıcı adınız altında çeşitli programlar çalışacaktır. /etc/passwd dosyasındaki kabukl girdisi

Burada gösterilen alanlar boş olabilir. Gerekli olanlar sadece kullanıcı ismi,UID,GID ve ana dizindir. Çoğu kullanıcı hesaplarında bütün alanlar doldurulur ama yan/ek kullanıcılar bu alanların sadece bir kısmını kullanabilir.

Ana dizinler genelde /home klasörü altında yer alır ve isimlerini kullanıcılarının isimlerinden alırlar. Genelde bu kullanıcının ismi verilen ana dizini bulmayı kolaylaştıran bir yöntemdir. Teorik olarak,bir ana dizin kullanıcının belirlediği herhangi bir isim altında herhangi bir dosyada ya da klasörde yer alabilir.

Daha büyük sistemlerde /home ve 'kullanıcı ismi' dizini arasına bir yada birden fazla seviye dizin seviyesi koyulur. Aşağıdaki gibi;

```
/home/hr/joe Joe from Human Resources
/home/devel/sue Sue from Development
/home/exec/bob Bob the CEO
```

Bunun birkaç sebebi vardir. Bir yandan bölüm içindeki sunucu üzerinde ana dizini bulunan kullanıcının bölümünü korumayı kolaylaştırırken, bir yandan da diğer istemci bilgisayarlar için bunu mevcut kılar. Bir yandan, Unix (ve bazı Linux) işletim sistemlerinde binlerce dosya içeren dizinlerle işlem yapmak eskiden çok yavaştı ve bu /home dizininde bir kaç bin girdi üzerinde

istenmedik bir olaya ve dosyaların zarar görmesine sebep olabilirdi. Fakat şuanki Linux sistem dosyalarında (ext3 with dir_index and similar) bu artık bir yaşanan bir sorun olmaktan çıktı.

Göz önünde bulundurmanız gerekir ki, /etc/passwd dosyalarını elle değiştirip düzenlememelisiniz. Kullanıcı hesaplarını oluşturup düzenlemenize yardımcı olacak bir kaç program zaten var.

Genelde, kullanıcı verilerini /etc/passwd dosyasından başka bir yerde depolamak münkün. Çoklu (binlerce) kullanıcısı olan sistemlerde kullanıcı verilerini bağlantısal veritabanında depolamak tercih edilebilirken türdeş olmayan çoklu ortam kullanıcı veritabanlı LDAP dizinine bağlı ağlarda, kullanıcı verilerini bağlantısal veritabanında depolamak tercih konusu olmayabilir. Bunun detayları ise bu derse dahil değildir.

13.2.2 /etc/shadow dosyası

Güvenlik için,hemen hemen bütün Linux dağıtıcıları '/etc/shadow' dosyasında ("shadow passwords") kodlanmış kullanıcı parolaları depolar. Bu dosya normal kullanıcılar tarafından okunamazken, yöneticilerin yanı sıra'shadow' gruplar tarafından okunabilir fakat sadece root tarafından okunup değiştirilebilir.Eğer dosyayı normal kullanıcı olarak açmayı denerseniz bir hata oluşur ve sonucunda dosya açılmaz.

/etc/shadow dosyasını kullanmak zorunlu değildir fakat kullanmanız önerilir. Ancak shadow passwords (gölge parolaların) sunduğu ekstra güvenlik devre dışı bırakıldığı yerlerde, örneğin NIS kullanıcı verilerini diğer sunuculara aktarmak için kullanıldığında (Özellikle türdeş olmayan Unix ortamlarında) bir takım sistem ayarları değşikliği yapılması gerekebilir.

Format (biçim), Bu Dosya aşağıdaki formatta her kullanıcı için bir satır bulundurur.

```
<user name>:<password>:<change>:<min>:<max>
:<warn>:<grace>:<lock>:<reserved>
```

Örneğin:

```
root:gaY2L19jxzHj5:10816:0:10000::::
daemon:*:8902:0:10000::::
joe:GodY6c5pZk1xs:10816:0:10000::::
```

Aşağıda bireysel alanların anlamlarını öğreneceğiz.

username Bu /etc/passwd dosyasındaki bir girdiye eşdeğer/uymak zorundadır.Bu alan 2 dosyayı birbirine bağlar.

password Kullanıcının kodlanmış parolasıdır.Boş bir alan genelde kullanıcının parolasız oturum açabileceği anlamına gelir.Bir yıldız (*), yada ünlem

(!) işareti söz konusu olan kullanıcının oturum açmasını engeller. Kullanıcının hesabını uyan parolanın başına ünlem yada yıldız işareti ekleyerek tamamen silmeden kitlemek daha yaygındır.

change Son parola değişikliğinin tarihi, örneğin: gün olarak 1 ocak 1970'den itibaren

min Parolanın değiştirilme ihtimalinden önceki en son parola değişikliğinden itibaren geçen asgari zaman süresi.

min parolanın değiştirilmesine gerek kalmadan ne kadar süre daha geçerli kalacağını gösteren azami süre.Bu süre geçtikten sonra kullanıcı parolasını yenilemek zorundadır.

warn Kullanıcının parolasını bitiş tarihinden önce değiştirmesini hatırlatan gün sayısı.Genelde bu uyarı oturum açıldığında karşınıza çıkar.

grace Bitiş tarihinden itibaren hesaba katılarak,kullanıcıya parolasını değiştirmesi için tanınan ekstra süre.

lock Hesabın net olarak hangi tarihte kapanacağını gösteren tarih. 1 ocak 1970'dan itibaren

Parola kodlamasıyla ilgili bazı düşünceler var.Kodlanan bir parolanın tekrar deşifre edileceğini düşünebilirsiniz. Böyle birşey /etc/shadow dosyasının bir kopyasını elde etmeye başaran bir hacker'a bütün sistem hesaplarını açması demek olurdu. Ancak, durum gerçekte böyle değil, çünkü dönüşü olmayan birşeydir. Çünkü herhangi bir Linux şifresinin deşifre edilmiş gösterimini şifrelenmiş formatından kurtarmak/elde etmek imkansızdır. Kodları deşifre etmenin tek yolu olası parolaları şifreleyip daha sonra /etc/shadow dosyasındaki parolalarla uyuşup uyuşmadığını kontrol etmektir.

Parolanızın harflerini 95 tane görünür ASCII harflerinden oluşturduğunuzu farz edelim (büyük ve küçük harfler ayrılmıştır). Bu 95 tane tek harfli parolanın olduğu anlamına gelir ve 95x95 = 9025 tanede 2 harfli parolanın olduğu anlamına gelir vesaire. 8 harfli bir parola oluşturduğunuzu düşünürsek buda 6.6 katrilyon adet şifre yapar. Eğer saniyede 250,000 bin tane parolayı tekrardan kodlayıp ve olası parolayı bulmak için bu parolaları tek tek denediğinizi farzedersek hesaplara göre 841 yılınız olması gerekiyor, bu da mümkün değil.

Ancak bu hala güvende olduğunuz anlamına gelmez. Geleneksel yöntemleri ('crypt' ya da 'Des' denilen) artık pek yaygın değil. Bu yöntemler [3] sadece

[3]If you must know exactly: The clear-text password is used as the key () to encrypt a constant string (typically a sequence of zero bytes). A DES key is 56 bits, which just happens to be 8 characters of 7 bits each (as the leftmost bit in each character is ignored). This process is repeated for a total of 25 rounds, with the previous round's output serving as the new input. Strictly speaking the encryption scheme used isn't quite DES but changed in a few places, to make it less feasible to construct a special password-cracking computer from commercially available DES encryption chips.

girilen parolanın ilk 8 harfine bakmak gibi kötü olan yönleri var ve deneyimli olan bir hacker bugünlerde yeterince alana sahip olan bir disk alarak önceden kodlanmış 50 milyon parolanın önbelleğini bu alanda oluşturabilir. Bir parolayı çözmek için sadece önbelleklerindeki kodlanmış parolayı/ya da parolaları aramaları yeterli, bu yöntem kolayca ve hızlıca yapılabilir ve uyuşan açıkmetin parolayı okuyabilir.

Eğer işleri biraz daha zorlaştırırsak, yeni eklenen kodlanmış bir parola girildiğinde, sistem genelde şifrelenmiş parola için sunulan 4096 ihtimalden birini seçen 'salt' adında bir öğe/unsur ekler. 'Salt' ın temel amacı X kullanıcısının /etc/shadow dosyasının içeriğine bakarak, kendi parolasının Y kullanıcısının parolasıyla aynı olduğunun farkına varmasından dolayı kaynaklanacak darbelerden kaçınmaktır. Güzel bir etki için, önceki paragrafta belirtildiği gibi hacker'in önceden kodlanmış sözlüğü için gerekli olan alan 4096 unsuruyla boşa gitmiştir.

Bugünlerde şifreleme yöntemleri genellikle 12 bit yerine 48 bit salt kullanan ve istenilen uzunlukta parolalar oluşturulmasına izin veren MD5 algoritmasına dayalıdır. Şifreleme 'crypt'den daha yavaş bir şekilde çalışır ve asıl amaçtan biraz bağımsız olsada,hackerlari belli bir dereceye kadar engeller.

Çeşitli yönetici parola parametlerinden pek fazla birşey beklemeyin. Bu parametreler metin konsolu oturum açma işlemiyle kullanılır ama sistemin diğer parçalarının buna dikkat edip etmeyeceği sizin belirlediğiniz ayarlara bağlıdır (örneğin şekilsel giriş ekranı gibi). Kısa aralıklardaki kullanıcı şifrelerini aynı şeye zorlamakta pek birşey ifade etmez - bu genellikle 'bob1, bob2, bob3,' sırasıyla sonuçlanır ya da kullanıcılar 2 parola arasında seçim yapar. 'mimal interval' En kısa aralık

Sistem yöneticisi olarka başa çıkmanız gereken sorun sisteminize ait parolaları zorla kırmak isteyen insanlar değildir. Genel olarak 'social engineering (toplum mühendisliğini) kullanmak daha faydalıdır. Şifrenizi çözmeye çalışan akıllı biri a, b ve diğer harflerle değilde eşinizin ilk ismi, çocuklarınızın ilk isimlerini, arabanızın plakasını,ya da köpeğinizin doğum gününü vs. tahmin ederek parolanızı tahmin etmeye çalışır. Ya da hala insanların bugün hala tuzağına düştüğü telefon aramaları ; 'Merhaba,sizi IT departmanından arıyorum. Güvenlik sistemiyle ilgili bir test yapıyoruz ve acilen kullanıcı adınızı ve şifrenizi öğrenmemiz gerek.'

13.2.3 /etc/group dosyası

Genelde Linux grup bilgisini /etc/group dosyasında saklar. Bu dosya sistemdeki her grup /etc/passwd dosyasındaki kolonlarla ayrılan ve alanlardan oluşan girdiler gibi tek satırlık girdiler içerir.Daha doğrusu /etc/group dosyası her satırda 4 tane alan içerir

```
<group name>: <password>: <GID>: <members>
```

Anlamları aşağıdaki gibidir:

group name Grup ismi dizinlerdeki sıralamalar için kullanılır.

password Belirlenen grup için isteğe bağlı parola. Bu parola, üyesi olmadıkları gruba /etc/shadow yada /etc/group dosyasıyla 'newgrp' komutunu kullanarak grubun üyesi olduğunu farz eder. Yanlış girilen bir '*' ya da geçersiz bir harf normal kullanıcıları söz konusu olan gruba geçişlerini engeller. Bir 'X' ayrı bir parola dosyası olan /etc/gshadow'a aittir.

GID grubun rakamsal grup tanımlayıcısı

members Kullanıcıların comma-seperated listesi. Bu liste bu grubun üyesi olan fakat GID alanında yada etc/passwd girdisinde farklı değerleri olan ve grubu ikinci grup olarak kullanan kullanıcıların listesini içerir (Bu grubu birincil grup olarak kullanan üyeler ve kullanıcılarda bu listede yer alabilir fakat gerekli değildir).

Bir /etc/group dosyası aşağıdaki şekilde olabilir:

```
root:x:0:root
bin:x:1:root,daemon
users:x:100:
project1:x:101:joe,sue
project2:x:102:bob
```

Yönetici girdileri ve idari grupları için 'bin group' girdileri sistemdeki ek kullanıcıların hesaplarına benzerler. Buna benzer gruplara bir çok dosya verilmiştir. Diğer gruplar sadece kullanıcı hesapları bilgilerini içerirler.

UID'ler gibi,GIDs ler belirli bir değerden başlarlar,genelde 100'dür.Geçerli bir girdi için,en azından birinci yada 3. alan (grup ismi ve GID) kısmı doldurulmalıdır. Böyle bir girdi, GID'ye (/etc/passwd dosyasındaki kulanıcının birincil GID alanında oluşabilecek) metinsel bir isim verir.

Şifre ya da üyelik alanları sadece kullanıcılara ikincil grup olarak verilen gruplar için doldurulmalıdır. Üyelik listesinde belirtilen kulanıcılar 'newgrp' komutunu kullanarak GIDs'lerini değiştirmek istediklerinde sistem onlara şifrelerini sormaz. Eğer şifrelenmiş bir parola verilmişse, üyelik listesideki girdisi olmayan kullanıcılar grubun üyesi olduğunu göstermeleri için bu parolayı kullanarak işlemi doğrulayabilirler.

Uygulamada, grup parolaları hemen hemen hiç kullanılmıyorsa, idareci olarak parolalardan kaynaklanacak herhangi bir faydayı ya da avantajı düzeltmesi çok zordur. Bir yandan söz konusu kullanıcıların gruplarına şifre vermek daha uygunken, diğer yandan bütün grup üyeleri tarafından bilinen tek bir parola güvenlik sistemini kusuruz kılmaz.

Eğer güvenli olmak istiyorsanız, unutmayınki her grubun parola slotunda ('*') asterisk vardir ve bunu kullanabilirsiniz.

13.2.4 /etc/gshadow dosyası

Kullanıcı veritabanına gelince, grup veritabanı için gölge parola uzantısı vardır. /etc/group dosyasındaki şifrelenmiş fakat herkesin okuyabildiği grup parolaları etc/gshadow adındaki ayrı bir dosyada depolanır. Burada ayrıca grup hakkındaki diğer bilgiler depolanır, örneğin gruba üye ekleyen veya bu üyeleri kaldırma yetkisi olan yöneticilerin isimleri gibi.

Alıştırmalar

1. /etc/passwd dosyasının 2. sütununda hangi değerle karşılaşırsınız? Neden bu değer buradadır ya da neden bu değeri bulursunuz?

2. Metin konsoluna geçin (+F1'i kullanarak) ve oturum açmayı deneyin, yalnız kullanıcı isminizi küçük harflerle girin. Uyguladıktan sonra ne olduğunu söyleyin.

13.3 Kullanıcı Hesaplarını ve Grup Bilgisini Yönetme

Yeni bir linux dağıtımı sisteme yüklekdikten sonra, genellikle yan kullanıcı hesapları ve yönetici için sadece root hesabı bulunur. Diğer kullanıcı hesapları sonradan oluşturulur (genelde çoğu dagıtıcı kullanıcıları en azından bir tane normal kullanıcı hesabı açmaları için zorlarlar/şart koşarlar).

Yönetici olarak, gerekli bütün kullanıcılar için hesap oluşturmak ve düzenlemek sizin işinizdir. Bunu kolaylaştırmak için, Linux bünyesinde kullanıcı düzenlemele ayarları için bir kaç araç bulundurur. Bunu yapmak kolay bir iştir fakat bu araçların nasıl kullanılacağı hakkında biraz bilgiye sahip olmanız gerekmektedir.

13.3.1 Kullanıcı Hesabını Oluşturmak

Yeni kullanıcı hesabı olşturma işlemi genelde aynıdır ve aşağıdaki adımlardan oluşur;

1. /etc/passwd yada olası bir ihtimalle /etc/shadow dosyaları içinde yeni girdiler oluşturmalısınız

2. Gerekirse /etc/group dosyasında bir yada birkaç girdi oluşturabilirsiniz.

3. Ana dizin oluşturmalısınız, bir kaç temel gerekli dosyayı içine kopyalayın ve gurubun sahipliğini yeni kullanıcıya devredin.

4. Gerekirse kullanıcı ismini diğer veritabanına giriniz, örneğin, disk kotaları, veritabanı tablolara ve özel uygulamalara özel yetkiler atar.

Yeni hesaba eklenen bütün dosyalar sadece metin dosyalarıdır. Her adımı metin düzenleyicisi elle kullanarak uygulayabilirsiniz. Fakat, bu iş karmaşık olduğundan dolayı sıkıcı gelebilir, bu yüzden 'useradd' programı vasıtasıyla sistemin size yardımcı olmasını sağlar.

En basitinden, 'useradd' programını yeni kullanıcının kullanıcı ismini vermek için kullanabilirsiniz. Ek olarak, bir çok kullanıcı parametreleri girebilirsiniz, belirtilmemiş parametreler için (genelde UID), en uygun standart ayarlar otomatik olarak sistem tarafından seçilecektir. İstek üzerine, kullanıcının ana dizini programın /etc/skel dizininden aldığı dosyalarla donatılır ve oluşturulur. 'useradd' komutunun sözdizimi;

```
useradd [<options>] <user name>
```

Aşağıdaki seçenekleride kullanabilirsiniz :

-c comment GECOS alan girdisi

-d home directory Eğer bu seçenek yoksa, /home/(username) kullanılır/varsayılır

-e date Bu tarihte hesap otomatik olarak devre dışı bırakılır ("yyyy-mm-dd" formatında"

-g group Kullanıcının birinci grubu (isim yada GID). Bu grup sistemde bulunmak zorundadır.

-G Supplementary groups (names or GIDs). These groups must also exist.

-s shell Yeni kullanıcının oturum açma kabuğu

-u UID Yeni kullanıcının rakamsal UID'si.Bu UID "-o" seçeneği verilmemişse kullanımda olmayabilir.

-m Ana dizin oluşturur ve temel dosyaları içine kopyalar.Bu dosyalar,başka bir dizin "-k(directoy" ismini kullanmıyorsa /etc/skel dosyasından gelir.

Örneğin;

```
# useradd -c "Joe Smith" -m -d /home/joe -g devel -k /etc/skel.devel
```

Komutu Joe Smith adındaki bir kullanıcı için bir hesap oluşturur ve 'devel' grubuna atar. Joe'un ana dizini /home/joe şeklinde oluşturulur ve /etc/skel.devel dosyaları içine kopyalanır.

-D seçeneğiyle (SUSE dağıtımlarında, –show-defaults) yeni kullanıcıların özelliklerine standart değerler verebilirsiniz. Ek seçenekler olmadan, standart değerler aşağıdaki gibidir:

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
```

Bu değerleri -g,-b,-f,-e ve -s seçenekleriyle sırasıyla aşağıdaki gibi değiştirebilirsiniz.

```
# useradd -D -s /usr/bin/zsh          zsh as the default shell
```

Listedeki son 2 değer değiştirilemez.

useradd programı düşük seviyede/gelişmemiş bir araçtır. Gerçek hayatta, tecrübeli bir yönetici olarak muhtemelen 'useradd' programıyla yeni kullanıcı hesapları eklemeyeceksinizdir ama yerel politikaları kapsayan bir kabuk betiği yardımıyla bu işi kolaylıkla halledebilirsiniz (böylelikle bu komutları yada değerlerin hepsini bilmeniz ya da hatırlamanız gerekmiyecek). Ancak, shell scriptini kendiniz öğrenmeniz gerekecek eğer Debian GNU/Linux ya da türevlerini kullanıyorsaniz (Aşağıya bakınız)

Dikkat edin: Her Linux dağıtımında useradd denilen program olsada diğer araçlar detayları açısından farklılık gösterebilir.

SUSE dağıtımlarının useradd programı /etc/passwd dosyası yerine kullanıcıları isteğe bağlı olarak bir LDAP dizinine koyacak şekilde uygun hale getirir (Bu sebeptendir ki -D seçeneği başka yerde olabileceği gibi soru sormaz ya da standart değerleri düzenlemez - LDAP işlemlerini uygulamak hala tartışılır) Bu konunun detayları bu elkitabında yer almamaktadır.

Debian GNU/Linux ve Ubuntu'da, useradd programı mevcuttur fakat kullanıcı hesabı oluşturmak için adduser programını kullanmak tavsiye edilir. Adduser programının avantajı ise Debian GNU Linux un kurallarına uymasıdır ve dahada fazlası asıl hesabı oluşturmanın yanı sıra isteğe bağlı diğer uygulamaları çalıştırması/uygulayabilmesidir. Örneğin, bir kullanıcı bir web sunucusunun döküman ağacında bir dizin oluşturabilir böylelikle yeni kullanıcı burada dosyalarını yayınlayabilir ya da kullanıcı otomatik olarak veritabanı sunucusuna yetkili giriş sağlayabilir.

13.3.2 passwd Komutu

Passwd komutu kullanıcılar için parola ayarlar, eğer root (yönetici) olarak oturum açtıysanız, o zaman

```
#passwd joe
```

kullanıcı John için yeni bir parola sorar (ekranda tekrarlanmaması için parolayı/kullanıcıyı iki kere girmeniz gerekmektedir)

Passwd komutu normal kullanıcılar tarafından kendilerine ait parolalarını değiştirmeleri içinde kullanılabilir (başka kullanıcıların şifrelerini değiştirmek sadece root (yöneticinin) yetki alanındadır.

```
$ passwd
Changing password for joe.
(current) UNIX password: secret123
Enter new UNIX password: 321terces
Retype new UNIX password: 321terces
passwd: password updated successfully
```

Yeni bir parola oluşturabilmek için önce normal kullanıcılar kendilerine ait parolaları bir kez doğru olarak girmeleri gerekir. Bu bilgisayarınıza giren başka kişilere bilgisayarınıza girmelerine izin vermemeniz için gayet kullanışlı bir yöntem.

Diğer yandan, passwd komutu /etc/shadow dosyasındaki çeşitli ayarları düzenlemenize yardımcı olur. Örneğin, passwd komutunun -s seçeneğiyle kullanıcının 'parola durumuna' bakabilir.

```
# passwd -S bob
bob LK 10/15/99 0 99999 7 0
```

Çıktıdaki ilk alan kullanıcı ismidir, ardından parola durumu gelir "PS" ya da "P" eğer parola belirlenmişse, kilitlenmiş bir hesap için sonuç "LK" ya da "L" ve parolası olmayan bir kullanıcı için "NP" sonucu. Diğer alanlar,sırasıyla, son parola değişiminin tarihi, parolayı değiştirmek için gereken en az ve en fazla aralık, bitiş uyarı aralığı ve parola süresinin bitiş tarihinden sonra kapanan hesaptan önceki'ekstra zaman süresi'.

passwd komutu seçenekleriyle bazı ayarları değiştirebilirsiniz. Aşağıda bir kaç örnek bulabilirsiniz;

```
# passwd -l joe Lock the account
# passwd -u joe Unlock the account
# passwd -m 7 joe Password change at most every 7 days
# passwd -x 30 joe Password change at lesat every 30 days
# passwd -w 3 joe 3 days grace period before password expires
```

/etc/shadow dosyasındaki şifrelenmiş parolanın önüne bir ünlem "!" koyarak -l ve -u değerleriyle hesapları kilitleyebilir veya kullanıma açabilirsiniz. Veritabanındaki 'kodlanmış parolayla eşleşen oturum üzerine/içine herhangi birşey girmek mümkün değildir. Bu yüzden, normal oturum açma işlemiyle sisteme erişim sağlamak engellenmiştir. Eğer "!" kaldırılırsa, asıl parola tekrar yürürlüğe girer. Yalnız, kullanıcı veritabanındaki kodlanmış parolayla eşleşmeyen

başka araçlarla kullanıcıların sisteme erişim sağlayabileceğini unutmamamız lazım, örneğin public-key doğrulaması olan güvenlik kabuğu gibi.

/etc/shadow dosyasında kalan ayarlara "chage" komutu uygulanması gerekir.

```
# chage -E 2009-12-01 joe Lock account from 1 Dec 2009
# chage -E -1 joe Cancel expiry date
# chage -I 7 joe Grace period 1 week from password expiry
# chage -m 7 joe Like passwd -m
# chage -M 7 joe Like passwd -x (Grr.)
# chage -W 3 joe Like passwd -w (Grr, grr.)
```

(chage, passwd komutunun değiştirebileceği bütün ayarları değiştirebilir)

Eğer seçenek isimlerini hatırlayamıyorsanız, chage komutunu sadece kullanıcı hesabının ismiyle kullanın. Program size değiştirmeniz ya da doğrulamanız için bir dize güncel değer sunar. Eğer yönetici bile düz metin bir parola alamıyor. Hatta tüm parolaların saklandığı /etc/shadow dizini bile yardımcı olmuyorsa. Bir kullanıcı parolasını unutursa genellikle passwd komutu kullanarak yeniyebilir.

Eğer root parolanızı unuttuysanız. Gerektiği takdirde kök olarak oturum açmak için bir kurtarma kabuğuna veya boot linux önyükleme icin disk veya CD ile /etc/shadow root bölgesini temizlemek için bir editör kullanabilirsiniz.

Alıştırmalar

1. Joe adlı kullanıcının sifresini degistirin.etc/shadow dizininde nasıl bir degisiklik yapmamız gerekir.

2. Dumbo adlı kullanıcı sifresini unuttu. Nasıl yardımcı olursunuz?

3. joe adlı kullanıcının sifresini bir veya iki hafta icerisinde degistirilmesi gerekmektedir. Fakat bir ikinci hafta varmadan ikigun once kullanıcı uyarılmalıdır. Bunun icin kullanıcı ayarlarını kontrol edin.

13.3.3 Kullanıcı Hesabını Silmek

Bir kullanıcı hesabı silmek için, /etc/passwd ve /etc/shadow dan kullanıcının girdilerini kaldırmak, /etc/group daki kullanıcının tüm referanslarını silmek ve kullanıcının ev dizininin yanı sıra oluştulan veya kullanıcı tarafından sahip olunan diğer dosyaların kaldırılması gerekir. Eğer kullanıcı varsa, /var/mail'de gelen mesajlar için bir posta kutusu gibi onu da silmiş olmalıdır.

Bu adımları otomatikleştirmek için yine uygun bir komut vardır. userdel komutu bir kullanıcı hesabını tamamen siler. Sözdizimi:

```
userdel [-r] <kullanıcı ismi>
```

-r seçeneği, kullanıcının ev dizinini (içeriği de dahil olmak üzere) ve /var/mail deki posta kutusunu silmeyi sağlar, kullanıcıya ait diğer dosyaları-mesela crontab dosyalar- el ile silinmelidir. Belirli bir kullanıcıya ait dosyaları hızlı bir şekilde bulmak ve kaldırmak için aşağıdaki komut kullanılır.

```
find / -uid <UID> -delete
```

-r seçeneği olmadan sadece kullanıcı veritabanından kullanıcı bilgilerini kaldırır, evdizini yerinde kalır.

13.3.4 Kullanıcı Hesabını ve Grup Bilgisini Değiştirmek

Kullanıcı hesapları ve grup atamaları geleneksel olarak /etc/passwd ve /etc/group dosyaları düzenlenerek değiştirilebilir. Ancak, çoğu sistem aynı amaç için usermod ve groupmod gibi komutlar içerir, daha güvenli ve kullanımı daha uygun olduğundan bunu tercih etmelisiniz.

usermod programı çoğunlukla useradd gibi aynı seçenekleri usermod kabul eder fakat kullanıcı hesaplarındaki mevcut değişiklikler yerine yenilerini oluşturur. Örneğin aşağıdaki komut ile kullanıcının birincil grubunu değiştirebilirsiniz.

```
usermod -g <group> <user name>
```

Dikkat! Eğer mevcut bir kullanıcı hesabının UID'sini değiştirmek istersen, /etc/passwd de doğrudan <UID> alanını düzenleyebilirsiniz. Ancak, aynı zamanda chown kullanarak yeni UID'ye kullanıcı dosyalarını transfer etmelisiniz. "chown -R tux /home/tux" ile kullanıcı tux'a kullanıcı tux'un ev dizini altında sahip olduğu tüm dosyaları yeniden kazandırıp sonra hesap için UID değiştirmiş olursunuz. "ls -l" metinsel bir isim yerine sayısal bir UID görüntüler. Bu, bu dosyaların UID için herhangi bir kullanıcı ismi olmadığını ima eder. chown kullanarak bunu düzeltebilirsiniz.

13.3.5 Kullanıcı Bilgisini Doğrudan Değiştirmek – vipw

Vipw komutu dogrudan /etc/passwd duzenlemek icin bir duzenleyici(vi veya farklı bir editor) cagırır. Söz konusu olan dosyanın diger kullanıcaları da etkilemesi icin e.g kullanılarak etkilendirilebilir. -S secenegi ile /etc/shadow dizinide duzenlenebilir.

The actual editor that is invoked is determined by the value of the VISUAL environment variable, alternatively that of the EDITOR environment variable; if neither exists, vi will be launched.

Alıştırmalar

1. Test adı verilen bir kullanıcı olusturun. Test hesabını değiştirin ve anan dizinde (örneğin /tmp) farklı bir dizinde bir kac dosya olusturun. Test UID root degistirmeye calısın. Kullanıcı test listlerinde gordunuz durumu degerlendirin.

2. Dağıtım grafiksel aracını kullanarak test1 adında bir kullanıcı olusturun useradd komutu ile test2 ve test3 adında yeni kullanıcıları elle oluşturun. Yapılandırma dosyalarına bakınız. Yeni hesapları kullanarak bir dosya olusturun bu üç hesaptan herhangi birini kullanarak yeni dosyalar sorunsuzca oluşturabilir miyim?

3. Kullanıcı test2 hesabını silin ve bu kullanıcıya ait sistemde bir dosya kalmadığından emin olunuz.

4. Kullanıcı test1 UID degistirin.Baska ne yapmamız lazım?

5. /home/test1 gelen /home/kullanıcı/ kullanıcı test1 ev dizini değiştirmeye çalışınız..

13.3.6 Grup Oluşturmak, Değiştirmek ve Silmek

Kullanıcı hesapları gibi, çeşitli yöntemlerden herhangi birini kullanarak grup oluşturabilir. Yeni kullanıcı hesapları oluştururken "manuel" yöntemi çok daha hızlı ve kolaydır: Grupların ana dizinleri olmadığından, /etc/group dosyayı düzenlemek ve yeni bir satır eklemek için genellikle herhangi bir metin düzenleyicisi kullanmak yeterli olur. Grup şifreleri kullanıldığında, başka bir giriş /etc/gshadow 'a eklenmelidir.

Bu arada, gruplar için dizin oluşturmada yanlış bir şey yoktur. Grup üyeleri orada kolektif calışma meyvelerini yerleştirebilir. Yaklaşık olarak kullanıcı ana dizinleri oluşturmaya benzemektedir, ancak yapılandırmada hiçbir temel dosyanın kopyalanması gerekmemektedir. Grup yönetimi için, kıyas yoluyla useradd, usermod, ve userdel, groupadd, groupmod, ve groupdel programları vardırki doğrudan /etc/group ve /etc/gshadow düzenlemek için.

Groupadd ile sadece doğru komut parametreleri vererek yeni groupadd gruplar oluşturabilirsiniz:

```
groupadd [-g <GID>] <group name>
```

-g seçeneği, belirli bir grup numarasını belirtmenize olanak sağlar. Daha önce belirtildiği gibi bu pozitif bir tam sayıdır. 99'a kadar olan değerler genellikle sistem için ayrılmış gruplardır. Eğer –g belirtilmezse, bir sonraki serbest GID kullanılır.

Doğrudan /etc/group groupmod yazmak zorunda kalmadan groupmod ile varolan gruplar düzenleyebilirsiniz.

```
groupmod [-g <GID>] [-n <name>] <group adı>
```

"-g <GID>" seçeneği grubun GID'sini değiştirir. Çözülemeyen dosya grubu atamaları manuel olarak ayarlanması gerekir. "-n <name>" seçeneği GID değiştirmeden grup için yeni isim düzenler; elle ayarlamalar gerekmez.

Grup girişlerini kaldırmak için bir araç vardır. Bu groupdel olarak adlandırılır:

```
groupdel <group name>
```

Burada da, bu dosya sistemini kontrol etmek ve ayarlamak için mantıklı "orphaned" group dosya atamaları için chgrp komutu vardır. Kullanıcıların birincil grupları silinmiş olamaz-söz konusu kullanıcılar ya önceden silinmiş ya da farklı bir birincil gruba atanmak için zorlanmış olabilir.

Gpasswd komutu özellikle grup şifreleme işlemleri için kullanılır ve passwd komutuna benzer. Sistem yöneticisi, bir veya daha fazla grup, idari grup yöneticisi, grup üyeliği listesinin yönetimi temsil edebilir. Grup adminleri aynı zamanda gpasswd komutunu kullanır:

```
gpasswd -a <user> <group>
```

adds the ¡user¿ to the ¡group¿, and

```
gpasswd -d <user> <group>
```

removes him again. With

```
gpasswd -A <user>,... <group>
```

Sistem yöneticisi grubu yöneticileri olarak hizmet verecek olan kullanıcılara gösterebilir.

Bir süredir SUSE dağıtımları gpasswd'i dağıtımlarına dahil etmiyorlar. Onun yerine modifiye kullanıcı sürümleri ve bu grubun yönetim araçları vardır ki LDAP dizininden işleyebilir.

Sistem yöneticisi olarak, vigr komutu doğrudan kullanarak grub veritabanını değiştirebilirsiniz. Bu "özel" erişim vigr için bir editör çağırarak, vipw gibi çalışır /etc/group. Benzer şekilde, "vigr -s" /etc/gshadow erişim sağlar.

Alıştırmalar

1. Gruplara ne için ihtiyac duyulur örnek veriniz.
2. Bir grubun tüm üyelerinin erişebileceği bir dizin oluşturabilir miyiz?
3. Bir grup testi olusturun.Bu grubun uyesi sadece kullanıcı test1 olmalıdır. Bu gruba bir parola verin. Kullanıcı test1 ve test2 olarak oturum açın ve yeni bir grup olusturmaya çalıştığınızda olusan durumu analiz edin.

Bu Bölümdeki Komutlar

adduser Yeni kullanıcı hesapları oluşturmak için uygun komutu (Debian)

chfn Kullanıcıların kullanıcı veritabanında GECOS alanını değiştirmenizi sağlar

gpasswd Bir grup yöneticisi bir grup üyeliğini değiştirmek ve grup parolasını güncellemek için izin verir

groupadd Sistemi grup veritabanına kullanıcı grupları ekler

groupdel Sistem grubu veritabanından grupları siler

groupmod Sistemi grup veritabanında grubunda girişleri degistirir

groups Bir kullanıcı bir üyesi olduğu grupları gösterir

id Bir kullanıcının UID ve Gids görüntüler

last Oturum açmış son kullanıcıları listeler

useradd yeni bir kullanıcı ekler

userdel kullanıcı siler

usermod veri tabanındaki bilgileri günceller

vigr /etc/group /etc/gshadow önlemek için "Dosya kilitleme" ile düzenleme yapabilmesi sağlar

Özet

- Sisteme erisim kullanıcı hesapları tarafından yonetilir.
- Bir kullanıcı hesabının bir sayisal UID ve (en az)bir metinsel kullanıcı adı vardır.
- Kullanıclar gruplar olusturabilir.Bu grupların adları ve sayısal GID'leri vardır.
- "Pseudo-users" ve "pseudo-groups" erişim haklarını daha da geliştirmek icin hizmet verilmektedir.
- Merkezi kullanıcı veritabanı (normal) /etc/passwd dosyasında saklanır.
- Kullanınıcın parolaları /etc/shadow dosyasında saklanır ve normal kullanıcılar bu dosyada yer alan parametrelere erisemezler.
- Grup bilgileri /etc/group ve /etc/gshadow dosylarında saklanır.

- Sifrele passwd komutu kullanılarak degistirilir.
- Chage programı /etc/shadow parametrelerini yonetmek icin kullanılır.
- Kullanıcı bilgileri vipw kullanarak ya da diger komutlar (useradd, usermod, userdel) ile değiştirilir.
- Grup bilgileri groupmod, groupdel, groupadd ve gpasswd komutları kullanılarak düzenlenebilir.

Bölüm 14

Linux Erişim Kontrol Sistemi

Amaçlar

- Linux erişim kontrol/ayrıcalık mekanizmalarının anlaşılması
- Dosya ve dizinlere erişim hakkı atamak için yetkili olmak
- SUID, SGID ve "sticky bit"in öğrenilmesi

Önceden Bilinmesi Gerekenler

- Linux kullanıcı ve grup konseptinin bilinmesi (Bölüm **??**)
- Linux dosya ve dizinlerinin bilinmesi

14.1 Linux Erişim Kontrol Sistemi

Erişim Kontrol Sistemi Birkaç kullanıcı aynı bilgisayar sistemine erişmek istedikleri zaman, A kullanıcısının B kullanıcısının özel dosyalarına erişememesini sağlamak için dizinler, dosyalar ve işlemler için erişim kontrol sistemi olmalıdır. Bunun sonucunda Linux, Unix izinlerinin standart sistemini uygulamaya koyar.

İzinleri Ayırmak Unix geleneklerinde, her dosya ve her dizin, bir kullanıcıya (sahibine) ve bir gruba kesinlikle atanmıştır. Her dosya, sahibi, grup üyeleri (kısaca group olarak atanmıştır) ve tüm diğer kullanıcılar (others) için izinlere ayırmayı destekler. Okuma, yazma ve çalıştırma izinleri ayrı ayrı bu

üç kullanıcı takımı için seçilir haldedir. Sahibi, bir dosyanın erişim izinlerini belirleyebilir. Eğer ki sahibi, uygun izinleri verirse grup ve diğerleri de bir dosyaya erişebilir.

Erişim Modu Bir dosyanın erişim izinlerinin genel toplamı, erişim modu olarak adlandırılır.

Erişim Kontrolü Genellikle erişilebilir ortamda özel veya grup-içi veri depolayan birçok kullanıcılı bir sistemde, bir dosyanın sahibi, uygun erişim kontrolü ile dosyalarını okurken veya değiştirirken başkalarını da tutabilirsiniz. Bir dosya için haklar, sahibi, grubu ve diğerleri için ayrı ayrı bağımsız olarak tespit edilebilir. Erişim izinleri, grubun birlikte çalıştığı dosyalara, bir grup işbirlikçi süreç sorumlulukları eşlemek için kullanıcılara izin verir.

14.2 Dosya ve Dizinler İçin Erişim Kontrolü

14.2.1 Temel Bilgiler

Dosya İzinleri Sistemdeki her dosya ve dizin için, Linux, 3 kullanıcı sınıfının her biri için (sahibi, grup ve diğer) erişim haklarını ayırmaya izin verir. Bu haklar okuma, yazma ve çalıştırma izinlerini içerir.

Dosyalara göre, bu izinler kabaca adlarının ima ettiklerini kontrol ederler. Her kim okuma iznine sahipse bir dosyanın içeriğine bakabilir, her kim yazma iznine sahipse içeriğin değiştirilmesine izin verilmiştir. Çalışma izni, bir dosyanın bir işlem gibi çalışması için gereklidir.

Binary yani makine kodlu bir program çalıştırılırken çalışma izni gereklidir. Kabuk betikleri veya "yorumlanabilir" programların diğer türlerini içeren dosyalar için, ayrıca okuma izni gerekir.

Dizin İzinleri Dizinler için, bazı şeyler bir bakıma farklı görünmektedir. Okuma izni bir dizinin içeriğine bakarken gereklidir. Örneğin; ls komutu çalıştırılırken. Dizin içerisinde dosyaları oluşturmak, silmek ya da adlandırmak için yazma iznine ihtiyaç duyarsınız. Çalıştırma izni "use" diye isimlendirilen "cd" diye komut mantığıyla alınır veya dizin ağacında altta kalan dosyaların isimleri ile ilgili yol ismini kullanır.

Okuma iznine sahip olduğun dizinlerde, dosyanın adını okuyabilirsin fakat dosyalarla ilgili herhangi bir şey bulamazsın. Eğer ki bir dizin için sadece çalıştırma hakkında sahipseniz, dosya isimlerini bilmeniz şartıyla onlara erişebilirsiniz.

Genelde bir dizine yazma ve çalışma izni atamak için mantıklı olunmalıdır. Buna rağmen, bazı özel durumlarda yararlı olabilir.

Dosya silinmek üzere ise bir dosya üzerinde bu yazma iznini önemini vurgulamak tamamen önemsizdir. Dosyanın olduğu dizine yazma izni gereklidir.

Bir dosyayı silmek sadece dizinden gerçek dosya bilgilerinin referansını silmektir, bu tamamen bir dizin işlemidir. Eğer ki bir dosyayı yazma izni olmadan silmek isterseniz rm komutu sizi uyarmaz fakat işlemi onaylarsanız ve dizine yazma izni verirseniz işlem başarılı olurken hiçbir şey olmayacaktır. (Like any other Unix-like system, Linux has no way of "deleting" a file outright; you can only remove all references to a file, in which case the Linux kernel decides on its own that no one will be able to access the file any longer, and gets rid of its content.)

Eğer ki dosyaya yazma izni verir fakat dizinine vermezseniz, o dosyayı tamamen silemezsiniz. Buna rağmen 0 byte'a kesebilirsiniz ve böylece içeriğini silebilirsiniz. Oysaki dosya özünde hala mevcuttur. Her bir kullanıcı için Linux, erişim haklarını en uygun belirler. Örneğin, bir dosyanın grup üyelerine dosya için okuma izni verilmezse fakat diğerlerine verilirse, grup üyeleri dosyayı okuyamaz. Aslına bakılırsa eğer ki diğerleri dosyayı görebiliyorsa, grup üyeleri de diğerlerinin parçası anlamına gelir, ki ona dahi okuma izni verilmelidir. Fakat bu uygulanmaz.

14.2.2 Erişim İzinlerini Kontrol Etmek ve Değiştirmek

Haklar, kullanıcılar ve grup atamaları ile ilgili bilgi elde etmek isterseniz ls –l komutunu uygulayabilirsiniz.

```
$ ls -l
-rw-r--r-- 1 joe users 4711 Oct 4 11:11 datei.txt
drwxr-x--- 2 joe group2 4096 Oct 4 11:12 testdir
```

Tablonun ilk sütunundaki karakter dizeleri kullanıcının, grubun ve diğerlerinin erişim haklarını ayrıntılı anlatır. (ilk karakter sadece dosya türüdür ve izinleri ile ilgisi var). Üçüncü sütun sahibinin, dördüncü sütun da grubun adını verir.

Erişim dizelerindeki "r", "w", "x" , sırasıyla okuma, yazma ve çalıştırma izinlerini ifade eder. Eğer ki listede "-" varsa, yerini tutan kategorinin ilişkili ayrıcalığa sahip olmadığını gösterir. Bunun sonucu olarak, "rw-r–r–" dizesi, kullanıcı için okuma ve yazma, grup ve diğerleri için de sadece okuma izninin verildiğini belirtir.

Dosya sahibiyseniz, chmod komutu ile bir dosyanın erişim haklarını ayarlayabilirsiniz. Dosya sahibi için "u", grup üyeleri için "g", diğer herkes için "o" kısaltmaları ile 3 kategori belirtebilirsiniz. İzinler kendilerine zaten bahsedilen kısaltmalar "r", "w"ve "x" tarafından verilmektedir. "+", "-" ve "=" kullanarak, herhangi bir izni ekleyebilir, mevcut izinlerden çıkartabilir ya da önceden ayarlananların tümünü değiştirebilirsiniz.

```
$ chmod u+x file          sahibi için çalıştırma izni verir
$ chmod go+w file         grup ve diğerleri için yazma izni verir
$ chmod g+rw file         grup için okuma ve yazma izni verir
```

```
$ chmod g=rw,o=r file     okuma ve yazma izni verir,
                          Grubun çalışma iznini siler;
                          Diğerleri için sadece okuma izni verir
$ chmod a+w file , ugo+w ile eşdeğerdir.
```

Aslında, belirtilen izinler oldukça karmaşıktır. Chmod ile ilgili bütün detayları bulmak için bilgi dokümantasyonlarına bakabilirsiniz.

Bir dosyanin yada dizinin erişim haklarını değiştirmeye izin veren tek kullanıcı o dosyanın sahibidir. Bu ayrıcalık, gerçek izinlerin bağımsızlığındandır. Sahibi, onların tüm izinlerini alır fakat geri verirken onları tutmaz.

Chmod komutunun genel yazımı

```
chmod [<seçenekler>] <izinler> <isim>...
```

Bir çok dosyaya yada dizine isteğinize göre verebilirsiniz. En önemli seçenekler şu şekilde:

-R Eğer ki bir dizine verilmişse , dizinlerin altındaki dizin ve dosyaların izinleri de değiştirilmiş olacaktır.

–reference=<isim> Dosyanın erişim izinlerini kullanır. Bu durumda hiçbir izin komut ile verilmemelidir.

Bir dosyanın erişim modunu sembolik yerine sayısal olarak da belirtebilirsiniz (biraz yukarda söylenilmişti). Uygulamada bu, bir dosyanın yada dizinin tüm izinlerini ayarlarken çok yaygındır ve şöyle çalışır. Üç izin, üç basamaklı sekizli sayı olarak temsil edilebilir. İlk basamak sahibinin haklarını, ikinci basamak grup haklarını ve üçüncü basamak da diğerlerinin haklarını tanımlar. Bu basamakların her biri okuma izni 4, yazma izni 2 ve çalışma izni 1 olan bireysel izinlerin toplamından türetilmiştir. “ls -l” ve sekizli form ile yaygın erişim modlarına birkaç örnek:

```
rw-r--r-- 644
r-------- 400
rwxr-xr-x 755
```

Sayısal erişim modlarını kullanarak, sadece bir kerede tüm izinleri ayarlayabilirsiniz—Diğerlerini tek başına bırakırken bireysel hakların ayarlanması veya yok edilmesi için başka bir yol yoktur, bunu sembolik gösterimde ”+” veya ”-” operatörlerini kullanırken yaptığımız duruma benzetebiliriz. Bundan dolayı, komut

```
$ chmod 644 file
```

Sembolik olarak eşittir

```
$ chmod u=rw,go=r file
```

14.2.3 Dosya Sahibi ve Grubunu Belirleme—chown ve chgrp

Chown komutu, bir dosya yada dizinin sahibini ve grubunu ayarlamaya izin verir. Değişikliğin uygulanabilmesi için bu komut istenilen kullanıcı ismi ve/veya grup ismi ve dosya veya dizin ismini alır. Şu şekilde çağrılır :

```
chown <kullanıcı adı>[:][<grup adı>] <ad>...
chown :<grup adı><ad>...
```

Hem kullanıcı hem de grup adı verilirse, ikisi de değiştirilir. Eğer ki sadece kullanıcı adı verilirse, grup olduğu gibi kalır. Eğer ki kullanıcı adından sonra iki nokta geliyorsa , dosya kullanıcının birincil grubuna atanır. Eğer ki grup isminin önünde iki nokta varsa, sahibi değişmeden kalır. Örneğin:

```
# chown joe:devel letter.txt
# chown www-data foo.html      yeni kullanıcı www-data
# chown :devel /home/devel     yeni grup devel
```

Chown ayrıca nokta yerine kullanıcı eski bir sözdizimi olan iki noktayı da destekler.

Diğer kullanıcılara yada isteğe bağlı gruplara dosya vermek için root olmak zorundasınız. Bunun ana sebebi, eğer ki sistem kotayı kullanırsa (her kullanıcı depolama alanının belli bir miktarını kullanabilir), normal kullanıcılar aksi taktirde birbirlerini rahatsız edebilir.

Chgrp komutunu kullanarak, normal bir kullanıcı olarak dosyanın grubunu değiştirebilirsiniz - yeni grubun bir üyesi ve dosya sahibi olma şartıyla.

```
chgrp <group name><name>...
```

Bir dosyanın sahibini ya da grubunu değiştirirken, çeşitli kategoriler için erişim izinlerini değiştirmezsiniz.

Chown ve chgrp aynı zamanda dizin hiyerarşisinin parçası için yinelemeli değişiklikleri uygulamak için -R yi destekler.

Tabiki bir dosyanın izinlerini, grubunu, sahibini, çok popüler olan dosya tarayıcısını (Konqueror yada Nautilus gibi) kullanarak da değiştirebilirsiniz.

Alıştırmalar

1. Bir dosya oluşturun. Bu dosyanın grubu nedir? İkincil gruplarından birine dosya atamak için chgrp kullanın. Gruba üye olmayan bir kullanıcıya dosya atamak isterseniz ne olur?

2. Çeşitli tarayıcıların dosyanın izinleri, sahibi, grubu gibi ayarları için sunduğu mekanizmaları karşılaştırın. Belli başlı farklılıklar var mı?

14.3 Süreç Sahipliği

Linux depolama ortamı, üzerindeki verilerin, nesneler gibi sahipliliğini göz ününde bulundurur. İşlemlerin sistem üzerinde de sahipleri vardır.

Birçok komut sistem belleğinde bir süreç oluşturur. Normal kullanım sırasında birbirlerinden korunan bir çok süreç bulunmaktadır. Kendi sanal adres alanı içindeki tüm verileri ile birlikte her süreç, kullanıcı ve sahibe sahip olan bir sürece atanmıştır. Bu genellikle süreci başlatan kullanıcıdır fakat süreçler yönetici ayrıcalıkları kullanarak oluşturulur ki onların sahipliği değiştirilebilir ve SUID mekanizması aynı zamanda bunu yapma yetkisine sahiptir.

"-u" seçeneği kullanılarak çağrıldığında ise süreçlerin sahipleri ps program tarafından gösterilir.

```
# ps -u
USER PID %CPU %MEM SIZE RSS TTY STAT START TIME COMMAND
Bin 89 0.0 1.0 788 328 ? S 13:27 0:00 rpc.portmap
test1 190 0.0 2.0 1100 28 3 S 13:27 0:00 bash
test1 613 0.0 1.3 968 24 3 S 15:05 0:00 vi XF86.tex
nobody 167 0.01.4 932 44 ? S 13:27 0:00 httpd
root 1 0.0 1.0 776 16 ? S 13:27 0:03 init [3]
root 2 0.0 0.0 0 0 ? SW 13:27 0:00 (kflushd)
```

14.4 Yürütülebilir Dosyalar İçin Özel İzinler

"ls -l" komutu ile dosyaları listelerken, bazen normal rwx den farklı izin ayarlarıyla karşılaşabilirsiniz. Örneğin;

```
-rwsr-xr-x 1 root shadow 32916 Dec 11 20:47 /usr/bin/passwd
```

Bunun anlamı nedir? Burada biraz konunun dışına çıkmak zorundayız.

Passwd dizininin normal erişim modunu taşıdığını varsayalım:

```
-rwxr-xr-x 1 root shadow 32916 Dec 11 20:47 /usr/bin/passwd
```

Normal bir kullanıcı olan joe, parolasını değiştirmek ister ve passwd programını çağırır. Ardından, reddedildi mesajını alır. Sebebi nedir? Passwd işlemi yazmak için /etc/shadow dosyasını açmaya çalışır ve root bu dosyayı açmaya çalışana dek başarısız olur. Bu başka yoldan mümkün olmaz. Herkes parolaları keyfi değiştirmek ister ve örneğin root parolası değiştirilir.

Bymeans of the set-UIDbit (frequently called "SUIDbit", for short) a program can be caused to run not with the invoker's privileges but those of the file owner—here, root. In the case of passwd, the process executing passwd has write permission to /etc/shadow, even though the invoking user, not being a

system administrator, generally doesn't. It is the responsibility of the author of the passwd program to ensure that no monkey business goes on, e. g., by exploiting programming errors to change arbitrary files except /etc/shadow, or entries in /etc/shadow except the password field of the invoking user. On Linux, by the way, the set-UID mechanism works only for binary programs, not shell or other interpreter scripts.

Bell Labs used to hold a patent on the SUID mechanism, which was invented by Dennis Ritchie

SUID

. Originally, AT&T distributed Unix with the caveat that license fees would be levied after the patent had been granted; however, due to the logistical difficulties of charging hundreds of Unix installations small amounts of money retroactively, the patent was released into the public domain.

By analogy to the set-UID bit there is a SGID bit, which causes a process to be executed with the program file's group and the corresponding privileges (usually to access other files assigned to that group) rather than the invoker's group setting.

The SUID and SGID modes, like all other access modes, can be changed using the chmod program, by giving symbolic permissions such as u+s (sets the SUID bit) or g-s (deletes the SGID bit). You can also set these bits in octal access modes by adding a fourth digit at the very left: The SUID bit has the value 4, the SGID bit the value 2—thus you can assign the access mode 4755 to a file to make it readable and executable to all users (the owner may also write to it) and to set the SUID bit.

You can recognise set-UID and set-GID programs in the output of "ls -l" by the symbolic abbreviations "s" in place of "x" for executable files.

14.5 Dizinler İçin Özel İzinler

Oluşturucunun kimliğine göre, sahiplik dosya atama prensibinden başka bir istisna var: Dizin sahibi kendi hedef dizini gibi, o dizin aynı gruba sahip olmalı diye dosyalara komut verebilir. Dizinler üstündeki SGID biti ayarlanarak (kurularak) belirlenmiş olabilir. (Dizinler gibi çalıştırılamaz, SGID biti böyle şeyler için kullanılmaya uygundur.)

SGID bitiyle dizin erişim izni değiştirilemez. Ona başvurmak için kategori içindeki izni yazılmış olmalıdır. (sahip, grup, diğerleri) Eğer, örneğin kullanıcı, ne dizinin sahibi ne de dizin grubunun üyesi değilse, dosyaların orada oluşturulabilir olması için dizin grubu 'diğerleri' için yazılabilir olmalıdır. Eğer kullanıcı hiç bir biçimde grubun üyesi olmasa bile, SGID içinde oluşturulan dosya sonra o dosya dizinine sahip olur.

Bir dizindeki SGID bit için tipik bir uygulama bir dizindir. Bu, bir "proje grubu" için dosya deposu olarak kullanılır. (Sadece) proje grubu üyelerinin

dizindeki tüm dosyaları yazıp ve okuyabilme ve yeni dosyalar oluşturabilme yeteneğine sahip oldukları varsayılır. Bunun anlamı, projeye katkıda bulunan tüm kullanıcıları bir proje grubuna eklemeye ihtiyacınız olduğudur (ikinci bir grup yeterli olacaktır).

```
# groupadd project          yeni grup oluştur
# usermod -a -G project joe joe grubun içinde
# usermod -a -G project sue sue grubun içinde
```

Şimdi dizin oluşturabilir ve grubu onu atayabilirsiniz. Sahibi ve gruba tüm izinler verilmiş , diğerlerine verilmemiştir. Aynı zamanda SGID biti ayarlayabilirsin.

```
# cd /home/project
# chgrp project /home/project
# chmod u=rwx,g=srwx /home/Project
```

Şimdi, eğer ki kullanıcı hugo /home/Project içine bir dosya oluşturursa, dosya grup projesine atanmak zorundadır.

```
$ id
uid=1000(joe) gid=1000(joe) groups=101(project),1000(joe)
$ touch /tmp/joe.txt Test: standard directory
$ ls -l /tmp/joe.txt
-rw-r--r-- 1 joe joe 0 Jan 6 17:23 /tmp/joe.txt
$ touch /home/project/joe.txt project directory
$ ls -l /home/project/joe.txt
-rw-r--r-- 1 joe project 0 Jan 6 17:24 /home/project/joe.txt
```

There is just a little fly in the ointment, which you will be able to discern by looking closely at the final line in the example: Dosya doğru bir gruba aittir fakat proje grubunun diğer üyeleri bununla beraber sadece onu okuyabilirler. Eğer proje grubunun tüm üyelerinin de yazabilmesini istiyorsanız, ya chmod komutunu uygulamanız veya gruba yazma izni atanmış olması için umask komutu ile ayarlayabilirsiniz. (Alıştırma 14.4'e bakınız).

SGID mod sadece yeni dosyalar oluşturulduğu zaman sistem davranışını değiştirir. Mevcut dosyalar başka yerde olduğu gibi her yerde aynı şekilde çalışır. Bunun anlamı, örneğin, that a file created outside the SGID directory keeps its existing group assignment when moved into it (whereas on copying, the new copy would be put into the directory's group).

Chgrp programı her zaman SGID dizininde çalışır. Dosyanın sahibi, onu herhangi bir gruba üye olarak atayabilir. Sahibi, dizin grubunun üyesi değilse, chgrp kullanarak gruba dosyayı koyamaz. Dosya dizinin içinde baştan oluşturulmalıdır.

Bu, dizinde SUID bit ayarıyla mümkündür. Bu izin bir anlam ifade etmez.

Linux dizinler için, sahibinin silebildiği ya da dizin içindeki dosyaların kaldırılabildiği özel modu destekler.

```
drwxrwxrwt 7 root root 1024 Apr 7 10:07 /tmp
```

Bu t modu, "sticky bit", genel dizinler ortak kullanımda olduğunda, bir sorunu çözmek için kullanılır. Bir dizin için yazma izni, bir kullanıcıya, erişim moduna ve sahibine bakılmaksınız diğer kullanıcıların dosyalarını silme izni verir. Örneğin, /tmp dizinleri temel bir zemindir ve birçok program, geçici dosyalarını burada oluşturur. Bunu yapmak için, tüm kullanıcıların bu dizine yazma izni vardır. Bu, herhangi bir kullanıcının burada dosya silme iznine sahip olduğu anlamına gelir.

Genelde, bir dosyayı silerken yada adını değiştirirken, sistem, dosyanın erişim izinlerini dikkate almaz. Eğer ki "sticky bit" dizin üzerinde ayarlıysa, o dosya sonradan sadece, dosya sahibi, dizin sahibi yada root tarafından silinebilir. "Sticky bit"in ayarlanabilir yada silinebilir olmasını, "+t" yada "-t" sembolleri ile belirtebiliriz. Sekizlik gösterimde SUID ve SGID gibi aynı rakamla 1 değerine sahiptir.

"Sticky bit" adı, daha önceden Unix sistemleri için kullanılan ek bir anlamdan türemektedir. O zamanda, programlar başlatıldığında programın tamamı swap alanına kopyalanırdı ve sonlandırıldıktan sonra tamamen silinirdi. Sticky bitli program dosyaları silinmek yerine takas alanında bırakılabilirdi. Bu, bu programların çağrılmasımı hızlandırır çünkü hiçbir kopya yapılmamıştır. En çok Unix, Linux gibi sistemler sayfalamaya ihtiyaç duyar. Programın çalıştırılabilir dosyasındaki kodun sadece gerçekten gerekli olan bu parçalarını alır. Bu takas alanına hiç bir şey kopyalamaz, Linux'taki gibi, sticky bit asla orjinal anlamına sahip değildir.

Alıştırmalar

1. Özel "s" ayrıcalığı ne demek? Onu nerede bulabilirsiniz? Kendi oluşturduğunuz dosya üzerinde bu ayrıcalığı ayarlayabilir misiniz?

2. Umask ayarlamak için hangi umask çağrısı kullanılır? Örneğin, proje dizinindeki dosyaları okumak ve yazmak için proje grubunun tüm üyelerine izin vermek.

3. Özel "t" ayrıcalığı ne demek? Onu nerede bulabilirsiniz?

4. (Programcılar için) Uygun bir komut çağırarak C programı yazınız. Bu programı SUID root ayarlayınız ve onu çalıştırdığınız zaman ne olacağını gözlemleyiniz.

Bu Bölümdeki Komutlar

chrgp Bir dosyanın yada dizinin grubunu ayarlar

chmod Dizin yada dosya için erişim modu ayarlar

chown Bir dizinin yada dosyanın grubunu ya da sahibini ayarlar

Özet

- Linux, dosya sahibi, dosya grubunun üyeleri ve tüm diğerleri için ayrı ayrı dosya okumayı, yazmayı, çalıştırmayı destekler.
- Bir dosyanın izinlerinin toplamı erişim modu olarak adlandırılır.
- Her dosya (ve dizin) bir sahibe ve gruba sahiptir. Erişim hakları—okuma,yazma ve çalıştırma izinleri—bu iki gruba ve "others"a ayrı ayrı atanmıştır. Sadece sahibierişim haklarını ayarlamaya izin verebilir.
- Erişim hakları sistem yöneticisine (root) uygulanmaz. Root, tüm dosyaları okuyabilir ve çalıştırabilir.
- Dosya izinleri chmod komutunu kullanılarak manipüle edilebilir.
- Chown kullanarak, sistem yöneticisi kullanıcıları ve keyfi dosyaların grup atamalarını değiştirebilir.
- Normal kullanıcılar, farklı gruplara dosyalarını atamak için chgrp kullanabilirler.
- SUID ve SGID bitleri dosya sahibi veya dosya grubu ayrıcalıklarına sahip programların yürütülmesine izin verir.
- SGID biti dizinin grubuna atanacak olan dizine yeni dosyaların gelmesine yol açar. (oluşturulan kullanıcının birincil grubunun yerine).
- Sticky bit bir dizinde sadece sahibine (ve sistem yöneticisine) dosyaları silme izni verir.

Bölüm 15

Linux Ağ Yapılandırması

Amaçlar

- Temel Ağ Kavramlarını Öğrenmek
- Linux Bilgisayarlarda Tümleşik Ağ Gereksinimlerini Anlamak
- Önemli Komut Sorunlarını Çözmeyi Öğrenmek
- Önemli Ağ Servislerini Öğrenmek

Önceden Bilinmesi Gerekenler

- Dosya kullanma (Bölüm **??**) ve metin editörü kullanımı
- Linux dosya sistemi yapısını bilmek (Bölüm **??**)
- TCP/IP ve ağ servisleri hakkında bilgi sahibi olmak yardımcı olur

15.1 Ağ Temelleri

15.1.1 Giriş ve Protokoller

21. yüzyılda, yerel ağ bölgesi aracılığıyla bilgisayarlarda internete bağlanmak gerçek bir eğlencedir. Bu ağ ev DSL yönlendiricisi ya da kablosuz ağ olabilir. İnternete girmenin çeşitli yöntemleri vardır fakat siz hangi yolu kullanıyor olsanızda o yöntemlerin hemen hepsi birbirine benzer çalışır.

İnternet TCP/IP denilen protokol ailesi tabanlıdır. Bir protokol bir bilgisayarın diğer bilgisayarlarla nasıl konuşucağı konusunda bir anlaşmadır ve belirli elektriksel, optik ya da radyo sinyallerine ve cevaplarından web sunucusuna kadar bir çok şeyi kapsayabilir (ama bunların hepsi aynı anda olmaz). Çok ince

ayrıntılarıyla kesmeden bu üç protokolü birbirinden ayırmak mümkündür. Bu üç farklı protokol şöyledir:

Orta Erişim Protokolü Ağ kartının bir seviyesinde ve kablolarla veri iletimini yönetir. Bunlara örnek olarak Ethernet (LAN için) ya da I.E.E. 802.11 gibi WLAN protokolleri içerirler.

Haberleşme Protokolü Farklı ağ durumları arasında haberleşmeyi yönetir. İngiltere'den Avusturalya'daki bir web sitesine erişmek isterseniz haberleşme protokolü olan TCP ve IP protokolleri veri iletimini düzenler. Aslında gönderdikleriniz "indirme altında" ya da bunun tam tersi olarak ulaşır.

Uygulama Protokolleri Verilerinizi alan Avustralya'daki alıcının bu verilerle bir şeyler yapabileceğine emin olur. Örneğin web uygulama protokolü, HTTP, sizin bir koala resmi almanıza imkan tanır, Avusturalya'daki sunucu isteğinizi yorumlar ve size koala resmini yollar (kanguru resmini yollamak yerine), bu arada İngiltere'deki tarayıcı sizin gerçekten resim aldığınızı, aldığınız verinin hata mesajı olmadığını çözer.

Bu çok katmanlı yapı her katmanın hemen bir altındaki ve bir üstündeki ile haberleşmesi konusunda ciddi avantaja sahiptir. Uygulama protokolü, HTTP, İngiltere'den Avustralya'ya ne kadar byte göndereceiğini bilmeye gerek duymaz (iyi ki!), çünkü haberleşme protokolünü temsil eder. Haberleşme protokolü sırayla byteların gerçek iletiminin orta erişim protokolüne ayrılır.

Bu noktada bilgisayar ağının organizasyonu için yedi katmandan az olmayan ISO/OSI referans modeline başvurmaya alışılmıştır. Özetini ilgili yerlerde detaylarıyla size ayıracağız.

TCP/IP uygulama protokolü-yanında HTTP, bunlar SMTP (mail için olan haberleşme protokolü), SSH (uzaktaki makinede oturum açmak için), DNS(host ismini çözümlemek için) ya da ISP (internet tabanlı telefonculuk) ve diğer pek çoğunu içerirler-genelde ya TCP ya da UDP tabanlıdır. UDP bağlantısız güvenilmez iletişim sağlarken (veri iletim sırasında kaybolabilir, ya da hedef yere gönderildiğinden farklı sırada varması gibi), TCP bağlantıya yönelik, güvenilir bir servis sunar (bu tüm verilerin diğer uca uygun sırayla varacakları anlamına gelir).

Uygulamalar hangi haberleşme protokolünün daha kullanışlı olduğunu belirlerler. Webde veri iletimi sırasında hata oluşsun istemezsiniz (yazının bir kısmı, resim ya da indirilen yazılımın bir kısmı kaybolabilir, bu can sıkıcı felaket bir durumdur), bu nedenle TCP doğru seçimdir. Televizyonlar ya da sesli sohbetler için, genellikle servislerde küçük bırakmalarla yaşama tercih edilebilir (a pixelated picture or a brief burst of static) bu her şeyi durdurmaya çekmek olduğuna göre veriler iletilirken oluşan hatalar için sistem düzenlenir.

Bu noktada UDP daha anlamlıdır. UDP aynı zamanda DNS gibi servislerde küçük isteklerin hızlı ve kısa cevap olduğu yerlerde daha iyidir.

Üçüncü TCP/IP protokolü,ICMP sorun çözme ve kontrol amaçlı kullanılır. Normalde direkt olarak kullanıcı düzeyindeki uygulamalarda kullanılmaz.

15.1.2 Adresleme ve Yönlendirme

Usulüne göre her bilgisayar tekil bir adres kullanarak internet üzerinden erişime izin verir (bilgisayarınız belirtilmiş bir sunucuya istek gönderir ve bu isteğinize cevabı herhangi bir şekilde bulur). Bu adres IP adresidir. Hala çok popüler olan şema IPv4'tür. IPv4 dört tane sekizlik bir dizidir. Numaralar 0-255 arasındadır. Örneğin: 192.168.178.10 gibi.

Her kişi basitçe isteğine göre IP alamaz. Bunun yerine IP'ler bire bir atanır. ISP'niz ev bağlantınız için dikkat edecekken, şirketlerde bu işi sitemciler ya da ağ yöneticileri yapacaktır.

Bir şirket muhtemelen tüm zamanlarda aynı IP adresini alır. Öte yandan ISP'niz kesin olan sınırlı bir süre için size sadece ödünç bir adres verecektir, sonraki zamanlarda ise farklı bir adres alacaksınız. Bir yanda da müşterilerinin sayısından daha az adres kullanarak ISP'nizi etkinleştirir (her müşteri her zaman çevirim içi olmadığından beri) ve öte yandan ise bu servislerin sürekli IP adreslerisinizi değiştirme sıkıntısına engel olduğu varsayılır.

IP adresleri gökten düşmezler fakat-mümkün olduğu kadar-önlem ile atanırlar, farklı ağların arasında verilerin değişimi düzenli olarak korunurlar, ya da "yönlendirme", bundan basittir. Buna daha sonra tekrar daha ayrıntılı bakacağız.

Bilgisayarlar genelde daha fazla IP adresine sahiptir. Geri dönüş aygıt adresi 127.0.0.1 olarak her bilgisayarda mevcuttur, bilgisayarlar ona başvuru yaparlar ancak o dışarıdan erişilebilir değildir. Bir servis 127.0.0.1 adresine sadece istemci tarafından erişebilir ve aynı makinede çalışarak bağlanır.

Bu söylenenler kullanışsız ve saçmadır ama aslında seçkin anlamdadır. Örneğin kullanıcılara mail gönderebilecek bir web sitesi geliştiriyor olabilirsiniz (kullanıcı hesabı için içerikteki aktifleştirme linkine tıklayın gibi). Bu test sizin geliştirme makinenizde yapılır, geri dönüş aygıt adresi üzerinde sadece mail kabul edilsin diye yerel mail sunucusunu yükleyebilirsiniz—internetteki spamların istilasından projenizi kormuak için mükemmel bir uygun olacak

Bilgisayarların üzerindeki Ethernet ya da Wifi IP adresinize sahip uygun bir arayüzdür (en azından bilgisayarınız şimdilik arayüze bağlıysa). Ek olarak hiçbir şey test ya da uzman yapılandırmalarınız için ağ arayüzüne sizden ek bir adres atamasını önlemez.

X bilgisayarının internette diğer Y bilgisayarı ile bağlanabilmesinin hazırlanması için rekabet oluşur, bu Y'nin IP adresinin bilinmesi şartıyladır. (Eğer X İngiltere'de ve Y'de Avusturalya'da ise X'in kaç baytının Y'ye ulaşması gerektiği %100 açık değildir.) Bunu yapmak büyülü kelime olan "routing (yönlendirme)" ile mümkündür. Ve yönlendirme yaklaşık olarak bunun gibi çalışır:

- X bilgisiyarınız Y'nin IP adresini çözebilir (DNS bu işe yarar). 10.11.12.13 olduğunu varsayalım.

- X bilgisayarı IP adresi olan 10.11.12.13'ü çözdüğünde yerel ağdaki ile değerler aynı değildir (Y'nin Avusturalya'da olduğunu düşünmemizde hiç garip yan yok) Bu X bilgisayarının direkt olarak Y'ye veri gönderemediği anlamına gelir (büyük sürpriz).

- X bilgisayarının ağ yapılandırması "varsayılan rota" içerir, aksi halde verilerle ne yapacağını bilemediğinden direkt olarak hedef yere gönderilemez, tarif olarak bilinir. Bu görünüş olarak belki buna benzeyebilir "Bir şeyler gönderdiğinizde hemen Z bilgisayarına iletilemez". Z bilgisayarına aynı zamanda "varsayılan ağ geçidi" denilir.

- Z bilgisayarı –belki DSL yönlendiriciniz– Y bilgisayarına direkt olarak verileri iletemez. Ancak Z bilgisayarı, X bilgisayarı gibi adreslenmemiş bilgisayarlara direkt bağlanarak elindeki verilerle ne yapacağını bilir, yani onu ISP'niz üzerinden gönderir.

- Verileriniz bilgisayara gelene kadar bu oyun bir kaç düzey daha devam eder. Adreslenmiş veriler 10.11.x.y şeklinde Avusturalyan ISP'sine gönderilmelidir (ona "Billabong-Net" diyelim). Yani veriler burada gönderildi.

- Billabong-Net 10.11.12.13 e gelen verilerin en yüksek hedefe nasıl iletileceğini (umarım) bilir. Özellikle Y bilgisayarı Billabong-Net's alıcısınn birine yerleşmiş olabilir ya da alıcısının alıcısına yerleşmiş olabilir. Fakat doğru yapılandırılımışsa dogru çözecektir.

- Y'den X'e doğru alınan herhangi bir yanıt benzer geri dönüş aygıtını kullanır.

Burada önemli bir gözlem, veri geçişi olduğunda gerçek rota X'den Y'ye verileri aldığını belirtir. X bilgisayarı orta hedeflerin tam listesini belirtmeye gerek duymaz, fakat her ara hedefin doğru şeyi yaptığına güvenir. Aksine, her bilgisayar sadece "yerel" bilgiye ihtiyaç duyar ve tüm internetin nerede olduğunu bilmeye gerek duymaz - bu oldukça imkansız olurdu.

Haberleşme protokolü olarak IP özelliğinin birini bir ilke içerisinde yönlendirebildiğin bir paketten bir sonrakine değiştirirsin (hatta genellikle herhangi bir olay olmaz). Bu internet bağlantı kesintileri, sıkışıklığı ve tepkilerine izin verir.

Bu esasında "yavaş giden e-posta"ya benzer. Sydney'deki gerçek rota doğum günü kartınıza ulaşımı tutar ve bunu bilmeye gerek duymaz; komşunuzun posta kutusuna basit ve doğru bir şekilde damgalanmış zarfı koyabilirsiniz.

Günün sonunda bu şu anlama gelir: Linux bilgiasayar 3 şeyi bilemeye gerek duyar, kendi IP'sine, direkt olarak ulaşabilsin diye ayarlanmış adrese, ve dinlenmek için olan ağ geçidine. The set of addresses that your computer

can reach directly is subnet mask described by the IP address of your local network together with a which today is most commonly specified as a number.

Bilgisayarınızın IP adresinin 192.168.178.111 olduğunu düşünün yerel ağda ise 192.168.178.0/24. Burada, 24 alt ağda maskeler. İlk 24 biti adresinizi belirtir (yani ilk üçlü sekizlik, şöyleki 192.168.178). Son sekizlik de ise (ya da sonraki 8'li 32 yapar) yerel adresiniz mevcuttur. Bu tüm bilgisayarların 192.168.178.0 dan 192.168.178.255 arasında olduğu anlamına gelir- olşuturdularsa- direkt olarak ulaşabilirler; varsayılan ağ geçidi (yerel ağdaki adres sahip olmalı) diğer IP adreslerine ulaşmayı kullanmalıdır.

Aslında, örneğimizdeki 192.168.178.0 ve 192.168.178.255 adresleri özel anlamlara sahip olduklarında beri bilgisayarlar için mevcut değildir. Fakat sadece bütünlük için onlardan bahsediyoruz.

Bu ilkede elle bilgisayarınızın ağ yapılandırmasının bir parçası gibi (Detaylar dağıtımınıza bağlıdır) bütün ağ parametrelerini ayarlayabilirsiniz – IP adres, alt ağ (maskeleme) ve varsayılan ağ geçidi. Ancak, çok muhtemel olarak "DHCP sunucusu" size bir ağ sağlar, bu mevcut olan bu parametrelerle bilgisayarınıza endişelenmeden ayarlar yapılacaktır.

Bundan dolayı, Linux bilgisayarınaz LAN istemcisi eklemeyi isterseniz, Ethernet kablosu uygun sockete takılmalı ya da wifi doğru bir şekilde girilmedilir. Herhangi bir problem varsa, sistemciniz ya da ağ yöneticiniz yüksek sesle ve ısrarla çağırılmalıdır.

15.1.3 İsimler ve DNS

IP adresleri güzel ve önemlidir, fakat kullanması oldukça sakıncalıdır. It would be very aggravating if you had to enter (much less remember) the address 213.157.7.75 to access the server offering the latest version of this manual. shop.linupfront.com is that much handier.

Bu yol uygun olmayan IP adreslerini uygun isimlere DNS üzerinden verir (ya da tam tersi). DNS host isimleri için global veritabanı dağıtımıdır, IP adresler ve çeşitli maddeleri DNS sunucusu üzerinden erişebilir.

DNS ayarını kendi Linux'unuza yapabilirsiniz, tabiki-, aynı zamanda oldukça makul bir şekilde şirketinizdeki web ve mail sunucusu boyutlandırılmış LAN çalışıyor. Bu genellikle iyi bir fikirdir. Ancak, bu konular LPI2 kitabında daha geniş bahsedilmektedir.

IP adres bölümünden önce DNS sunucularının adresleri ile birlikte diğer temel ağ parametreleri konusu vardır. Alt ağ, varsayılan ağ geçidi - her Linux makine sahip olmalıdır. gerçekte hiyerarşi şöyle işler:

- The "root-level name servers" know about the part of a name on the very right—like .de , .com , .tv , whatever—and know which name servers are in charge of the content of these zones.

- The name servers for .de (by way of an example) know all the names of the form something.de and can tell which name servers know about names below those names.

- The name servers for a name like something.de (which are usually situated at the company in question or their ISP) know the IP address for a name like www.something.de and can supply it if required.

Bu ismin çözüldüğü anlamına gelir (Örneğin isim:shop.linupfront.de), bilgisayarınızın ilk olarak root düzeyindeki isim sunucularına isim sunucuları için olan görevleri sorar. Sonra linupfront.de için olan görevleri sorar. Son olarak linupfront.de adresi için linupfront.de isim sunucusunu sorar.

Aslında çalışan bilgisayarınız değildir, onun DNS'i sizin bilgisayarınızı kullanır. Fakat bilgisayarınızın ilkelerini küçültmez.

Tabi ki bu oldukça ilgili bir şemadır, and this is why your system keeps any answers around for a while. shop.linupfront.de'ye uyan IP adresini çözdüyseniz, the assumption is that this will stay the same for a while, so the resolution process is only repeated after that time has expired.

The advantage of this scheme is that we at Linup Front are free to dispose of names "below" linupfront.de and can add them to our DNS server as we wish. Diğer insanlar buradan direkt alır. It would be much more of a hassle to have to petition the "Internet office" for a new name, and to have to wait for it to be added to the official list. (Think of changes to the land register and how long these usually take.)

15.1.4 IPv6

IP as a communications protocol has been around for something like 30 years, and we have found out in the meantime that some assumptions that were made back then must have been somewhat naive. Mesela, IPv4(şu anki sürümü) prensip olarak 2^32 ya da yaklaşık olarak 4 milyar adrese izin verir. Due to limitations in the protocol as well as various awkwardnesses with their distribution, there are very few if any unused IPv4 addresses left and in an age where nearly everybody carries an Internet-enabled smartphone and even more people would like to have one, this is a definite problem.

Bu problemi hafifletmenin yolları vardır, mesela, internette tek olmayan her şey-cep telefonları internetin her yerinden görülebilir olan bir IP adresi alır. Instead, the operators wall off their networks from the actual Internet in order to be able to distribute more addresses (cue "network address translation", NAT). These methods are fairly disgusting and do imply other problems.

IPv6, the designated successor to IPv4 [1], has been available since the late 1990s. IPv6 does away with various restrictions of IPv4, but ISPs are still

[1] IPv5 asla varolmamaıştır

somewhat reluctant to roll IPv6 out comprehensively. Linux does deal very well with IPv6, and since you can quite happily run IPv4 and IPv6 in parallel, there is nothing preventing you from setting up an IPv6-based infrastructure in your company (or even your domestic LAN—many DSL routers support IPv6 by now). Here are some of the more important properties of IPv6:

Genişletilmiş adres uzayı IPv6 32 bitlik adres yerine 128 bitlik adresi kullanır, görülebilir gelecek için yeterli olması beklenmektedir (şanslar oldukça iyidir) IPv6 adresler, IPv6 addresses are notated by writing down chunks of two bytes in hexadecimal (base 16), iki nokta üst üste kullanımıyla ayrılmıştır:

```
fe80:0000:0000:0000:025a:b6ff:fe9c:406a
```

Her dörtlü karakter bloğunun başındaki sıfırlar kaldırılmış olabilir.

```
fe80:0:0:0:25a:b6ff:fe9c:406a
```

birçok dizi bloğunun başındaki sıfırlar "::" ile yer değiştirebilir:

```
fe80::25a:b6ff:fe9c:406a
```

The loopback address, i. e., the moral equivalent to IPv4's 127.0.0.1 , is ::1.

Adres Ataması IPv4 ile, ISP'niz aynı IPv4 adresini atar ya da en fazla bir kaç tane atar. (Gerçek büyük bir şirkette olmağınız müddetçe ya da diğer ISP ve daha olanlar için adres oldukça zor bulunurdur.) Bilgisayarınız için daha fazla adres gerekmezse, dolambaçlı olmaya ihtiyaç duyarsınız. With IPv6, you are instead assigned a complete network, namely a "subnet prefix" that fixes only 48 or 56 of the possible 128 address bits. Öyleyse her alt ağ grubunun 2^64 lük adresi serbestçe atanmıştır, ve bu muhtemelen kullanılabilecek olandan daha fazladır. (IPv4'e göre tüm internet sadece 2^32 adresleri kullanırbunun küçük bir kısmı)

Basit Yapılandırma Simple configuration While with IPv4 a computer must be assigned a local IP address—possibly with the aid of a DHCP server—, using IPv6 a computer can assign itself an address that is suitable to communicate with other computers in the immediate vicinity. With IPv6, a computer can also, without DHCP, locate routers in the neighbourhood which are prepared to forward data to the Internet. This avoids various problems with DHCP on IPv4. DHCP, IPv4 deki çeşitli problemleri önler. Bu arada IPv6 "varsayılan rota" kullanmaz.

Diğer gelişmeler The format of IP datagrams was changed to enable more efficient routing. In addition, IPv6 defines methods to change a network's subnet prefix much more easily than a network's address could be changed in IPv6—this is also an attempt to simplify routing. Furthermore, IPv6 supports encrypted networks (IPsec) and mobility where computers—think of cell phones—can migrate from one network to another without changing addresses or interrupting existing connections ("mobile IPv6").

Uygunluk IPv6 nın tanıtılması sadece IP-protokollerine etki eder. TCP ve UDP ya da diğer uygulamala protokolleri değişmeden kalır. Aynı zamanda IPv6 ve IPv4 ün paralel çalışabiliceğinden bahsetmiştik.

Alıştırmalar

1. Hangi orta erişim protokollerini biliyorsunuz? (Ethernet ya da IEE-802.11 hariç) Hangi haberleşme protokollerini biliyorsunuz? (IP,TCP ve UDP haricinde) Yukarıda bahsedilen istisnalardan hangi uygulama protokollerini biliyorsunuz?

2. 10.11.12.0/22 ağda kaç IP adresi kullanılabilirdir? (İlk ve son adresleri çıkarın çünkü onların özel anlamları vardır.)

15.2 Ağ İstemcisi Olarak Linux

15.2.1 Gereksinimler

Zaten Linux tabanlı bir bilgisayara ağ oluşması için istemci (IPv4) olarak gerekli şartları ekledik. (Bu sadece Linux Temelleri ile ilgili durumlarda kullanılır.)

- Bilgisayar için IP adresi gereklidir.
- Ağ adresi ve alt ağ maskesi gereklidir.(yani yerel IP adresini kasteder.)
- Varsayılan ağ geçidinin adresi yerel ağ üzerindedir.
- Adresler DNS sunucusunundur.

LAN kablosu bağlandığında ya da kablosuz olarak açıldığında bilgisayarınız ayarları otomatik olarak DHCP üzerinden açılırken ayarlayacaktır. Durum böyle değilse, bu verilerle kendiniz yapılandırmalısınız. Detaylı bilgi dağıtımınıza bağlıdır:

Debian Debian GNU/Linux tan türetilmiş bir dağıtımdır, ağ yapılandırması /etc/network/interfaces dosyası içindedir. Bu yapı kendi içersinde büyük ölçüde açıklayacıdır, ve örnek yorumlar /usr/share içersinde vardır.

Ubuntu doc/ifupdown/examples/network- interfaces.gz. Bir miktar belgelendirme intefaces(5) içinde vardır.

Suse Suse dağıtımlarında, YAST üzerinden ağ verilerinizi yapılandırabilirsiniz. (Network Devices/Network Cards bakınız). Aksi halde /etc/sysconfig/network dosyası altında yapılandırmalar vardır.

RedHat RedHat dağıtımı ve türevlerinde /etc/sysconfig/network- scripts dizinlerinde yapılandırmalar vardır.

Aynı zamanda ifconfig komutunu kısa süreli denemeler için kullanabilirsiniz. Ağ arayüzündeki eth0 a 192.168.178.111 gibi IP adresleri atanır.

```
# ifconfig eth0 192.168.178.111 netmask 255.255.255.0
```

The local network is 192.168.178.0/24 —the 255.255.255.0 is a roundabout method of writing down the subnet mask. Varsayılan ağ geçidi (örneğin 192.168.178.1) route komutunu kullanılarak yapılandırılır.

```
# route add -net default gw 192.168.178.1
```

Bu ayarlar makine yeniden başlatılana kadar devam eder. DNS sunucunun adresi genellikle /etc/resolv.conf dosyasına yazılır, siz de buna yakın bir şey görüyor olabilirsiniz:

```
# /etc/resolv.conf
search example.com
nameserver 192.168.178.1
nameserver 192.168.178.2
```

(the “search example.com” will append “example.com” to any names specified without a period—so if you use www , the name being actually looked up will instead be “www.example.com”).

Sistem yapılandırmanız otomatikse – örneğin Wifi kullanırken – genellikle /etc/resolv.conf içine yazar. Dağıtımınızın belgelendirmesini kontrol ederseniz isim sunucularının nasıl ve nerede yapılandırıldığnı çözebilirsiniz.

15.2.2 Sorun Çözme

İnternete girmenin basit yaklaşımı çalışmıyorsa ya da başka problemler oluşuyorsa – web sitesine ulaşırken sonu gelmez gecikmelerin oluşması gibi– ya da anlaşılamayan

bağlantı arızası – sistemcinize ya da ağ yöneticinize danışmalısınız ya da, genellikle bu konu ile ilgili daha tanıdık kim varsa onunla konuşursunuz. (En azından LPIC2 sınavına geçene kadar insanlar bu konuda problemleri varsa size getirecekler.)

Diğer bir yandan, On the other hand, it always goes down well if you have excluded the most obvious problems yourself or narrowed the error down somewhat. This may save your administrator some work, or, if nothing else, lets you appear to your administrator like someone to be reckoned with rather than a complete rookie.

Bu bölümün geri kalanı pek çok hata çözüm aracını ve onların kullanımını açıklar.

ifconfig Ağ yapılandırması için ifconfig komutuyla size daha yeni tanıttık. ifconfig aynı zamanda ağ arayüzünün ayarlarını sorgular.

```
$ /sbin/ifconfig eth0
eth0   Link encap:Ethernet HWaddr 70:5a:b6:9c:40:6a
       inet addr:192.168.178.130 Bcast:192.168.178.255
                                                  Mask:255.255.255.0
       inet6 addr: 2002:4fee:5912:0:725a:b6ff:fe9c:406a/64
                                                  Scope:Global
       inet6 addr: fe80::725a:b6ff:fe9c:406a/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
       RX packets:112603 errors:0 dropped:0 overruns:0 frame:0
       TX packets:98512 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
```

Çeşitli adresler daha ilginç bitler içerir.

- İlk satırdaki çıktı "donanım" ya da "MAC adresi" ni içerir. Bu çıktı arayüz üreticisi tarafından atanmıştır(buradaki, Ethernet arayüzüdür)
- İkinci satıra IPv4 adresi atanmıştır. Oldukça sağ taraftaki oldfashioned/tedious daki gösterimde alt ağ maskesi vardır.
- 3 ve 4. satırdaki adresler çeşitli IPv6 adresleri içerir. 3.sü alt ağ ön ekini içerirken (bu eke internet üzerinden erişebilir), 4. satır bilgisayara atanan yerel adresi içerir.

Yakından bakarsanız, her IPv6 adresinizin ikinci yarısının MAC adresiniz olduğunu tanıyacaksınız.

The "UP " at the start of the fifth line denotes that the interface is actually switched on.

ifconfig komutunu parametre olmadan çalıştırırsanız, bilgisayarınızdaki tüm aktif ağlar hakkında bilgi veren bir çıktı alırsınız. "-a" parametresini verdiğinizde aynı zamanda o anda aktif olmayan ağları da görüntülersiniz.

ping komutunu bilgisayarınız ve diğerleri arasındaki düşük seviyeli(IP) bağlantıları kontrol etmek için kullanabilrsiniz. ping kontrol protokolünü kullanır, ICMP, diğer bilgisayarlar için "hayat belirtisi" sorar. Bu belirtiler bilgisayarınızda geriye ulaşıyorsa, bilgisayarınızın verileri diğer bilgisayara ulaştırdığını biliyorsunuzdur, ve diğer bilgisayarda sizin bilgisayarınıza veri gönderebilir (bu bir şart anlamına gelmez).

Basit durumda, pin diğer bilgisayarı ismiyle çağırır you'd like to communicate with:

```
$ ping fritz.box
PING fritz.box (192.168.178.1) 56(84) bytes of data.
64 bytes from fritz.box (192.168.178.1): icmp_req=1 ttl=64 time=3.84ms
64 bytes from fritz.box (192.168.178.1): icmp_req=2 ttl=64 time=5.09ms
64 bytes from fritz.box (192.168.178.1): icmp_req=3 ttl=64 time=3.66ms
64 bytes from fritz.box (192.168.178.1): icmp_req=4 ttl=64 time=3.69ms
64 bytes from fritz.box (192.168.178.1): icmp_req=5 ttl=64 time=3.54ms
Stop the program using Ctrl+c
...
--- fritz.box ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 3.543/3.967/5.095/0.575 ms
```

Burada her şey sıralıdır. 5 paketin hepsi geriye ulaşırlar, sırası doğrudur, ve iletim zamanları yerel ağ için mantıklı olmalıdır. If instead you see nothing for a while before appears, something is fishy — the target computer cannot be contacted.

```
From 192.168.178.130 icmp_seq=1 Destination Host Unreachable
From 192.168.178.130 icmp_seq=2 Destination Host Unreachable
From 192.168.178.130 icmp_seq=3 Destination Host Unreachable
```

Uzaktaki bilgisayara bağlanamazsanız, bilgisayarınız ve uzaktaki bilgisayar arasında herhangi bir yerde bir hata olabilir. Sistematik yaklaşımlar için aşağıdaki taktikleri uygulayabilirsiniz:

- ping komutunu kullanarak geri dönüş aygıtınının ulaşıp ulaşamadığını kontrol edebilirsiniz, 127.0.0.1. Çalışmazsa, bilgisayarımızda bazı şeyler yanlıştır.
- Ağ arayüz adresine ulaşıp ulaşamadığını "ping" i kullanarak kontrol edebilirsiniz. Şimdilik internet erişimini kullanıyorusunuz(ya da kullandığınıza inanıyorsunuz). Gerekirse aşağıdakine benzer bir şekilde adresi bulabilirsiniz

  ```
  $ /sbin/ifconfig eth0
  ```

Çalışıyor olmalı, çalışmadıysa, öyleyse yerelinizde bir problem vardır.

- pingi kullarak yerelinizdeki varsayılan ağ geçidini alıp almadığınızı kontrol edebilirsiniz. (Adresinizi kalbinizden bilmiyorsanız, çözebilmek için route'u kullanın

```
$ /sbin/route
Kernel IP routing table
Destination   Gateway    Genmask   Flags Metric Ref Use Iface
default       fritz.box  0.0.0.0   UG    0      0   0   eth0
link-local    *          255.255.0.0 U   1000   0   0   lo
192.168.178.0 *          255.255.255.0 U 0      0   0   eth0
```

default girdi altında Destination nasıl kullanacağını anlatır–buradaki "ping fritz.box "). Çalışmıyorsa ve buna

```
Destination Host Unreachable
```

benzer bir mesaj alıyorsanız o zaman yerel ağınızda bir problem vardır.

If you can, ask a colleague who is just accessing the Internet, or try another computer: If everything seems to be OK there, then again your computer is likely to be the culprit. Otherwise—and quite likely in that case, too—it is time for the system administrator.

Mesela varsayılan rotanız yanlış olabilir ve yanlış bilgisayara işaret ediyor olabilir. (Ağ yapılandırması elle yapıldığında bu daha çok muhtemel bir durumdur) That would be unlikely to impact the users of other computers, where the configuration is probably correct.

Aslında doğru bir şekilde varsayılan ağ geçidine ulaşıyorsanız, o zaman problem ya LAN'ımız dışındadır, ya internet üzerinde herhangi bir yerde ya da "protokol yığınınızdan" uzakta bir yerlerdedir. Mesela pingi kullanarak uzaktaki web sunucusuna ulaşmak kolay olabilir, fakat (şirketiniz?) internet erişiminiz web üzerinden doğrudan erişime izin vermeyebilir çünkü "vekil sunucu"yu kullandığınızı zannedersiniz (ve yapılandırmayı unuttuysanız). Sistem yöneticiniz yardımcı olacaktır.

Ağ bağlantınız bazen çalışır bazen de "ping -f" yi kullanarak test edemezsiniz (kablolar dolaşabilir?, kemirgenlerin zarar vermesi?). Klasik olarak saniye başına paket göndermenin yerine, onun yapabildiği kadar hızlı verileri ping gönderir. Çıktıda bir nokta için her paket gönderilmiştir ve bir ters boşluk karakteri için her paket alınmıştır. Eğer bir paket kaybettiyseniz, noktaların satırı uzatması olacaktır.

If you're not root but an ordinary user, you must make do with a minimal interval of 0.2 seconds between two packets sent. Sistem yöneticisiyseniz sadece ağ taşmış olabilir.

IPv6 bağlantınızı kontrol etmek için ping yerine ping6 komutunu kullanabilirsiniz.

```
$ ping6 ::1
```

dig dig komutu DNS isim kararlılıklarını test etmek için kullanılır. Özel olarak belirtmediğiniz sürece aksi halde, IP adresinize uygun gelen isimleri bulmaya çalışır.

```
$ dig www.linupfront.de
; <<>> DiG 9.8.1-P1 <<>> www.linupfront.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34301
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.linupfront.de.    IN     A

;; ANSWER SECTION:
;www.linupfront.de.    3600   IN    CNAME   s0a.linupfront.de.
s0a.linupfront.de.     3600   IN    A       31.24.175.68

;; Query time: 112 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Mar 1 16:06:06 2012
;; MSG SIZE rcvd: 69
```

Bu çıktı "QUESTIONS SECTION" kısmında www.linupfront.de için bakıyorsak bize anlatır(biz zaten neresi olduğunu bilmediğimizde). The "ANSWER SECTION" lets on that actually no IP address corresponds to www.linupfront.de , but that www.linupfront.de is in fact a "nickname" for the computer called s0a. linupfront.de (a popular approach). s0a.linupfront.de , however, has the address 31.24.175.68 . Son olarak, sonraki kısımda 127.0.0.1 üzerinde DNS sunucudan cevap geldiğini anlatır.

If there is no answer for some time and then something like appears, then there is something rotten in the state of Denmark.

```
; <<>> DiG 9.8.1-P1 <<>> www.linupfront.de
;; global options: +cmd
;; connection timed out; no servers could be reached
```

Sizin /etc/resolv.conf daki ayarlarınız yanlıştır ya da isim sunucunuz yapması gerekeni yapamaz.

Komut satırında bahsedildiği gibi özel sunucu isimlerini sorabilirsiniz.

```
$ dig www.linupfront.de @fritz.box                    Frage fritz.box
```

Tabiki bu isim çözümü pahalı bir DNS sorgusunu gerektirmez (veya bir tavuk yumurta probleminiz var demektir). Şüphelendiğinizde direkt olarak IP adresinizi belirtebilirsiniz.

```
$ dig www.linupfront.de @192.168.178.1
```

DNS yolunuzunu biliyorsanız, RR tipindeki diğer A kayıtlarına bakmak için dig'i kullanabilirsiniz. Komut satırında ne istiyorsanız dig sadece onu anlatır.

```
$ dig linupfront.de mx
```

To find the name belonging to a given IP address (if any), -x seçeneğiyle belirtmelisiniz.

```
$ dig +short -x 31.24.175.68
s0a.linupfront.de.
```

(dig +short seçeneğiyle çok kısa bir çıktı verir.)

dig tabiki bir çok şey daha yapabilir, fakat sadece sunucu isimlerini yapılandırmak isteyenler ya da korumak isteyenler bu komutu çalıştırırlar. Eğer bu karşı koyamazsanız, yeterli bilgi dig(1) de vardır.

netstat netstat komutu İsviçre bıçağının bir çeşididir. Bu komut bilgisayarınızın tüm ağ bilgilerini sınıflarındırarak çıktı verir. Bu komutu eğer sadece "netstat" olarak çalıştırırsanız bütün aktif ağların bir listesini alırsınız.

```
$ netstat
```

This includes not only TCP connection, but also local connections via Unix domain sockets, which are as stiflingly voluminous as they are utterly boring. Daha değişik bir kullanım ise "netstat -tl" dir. Bu şekilde kullanılınca TCP portlarının bir listesi elde edilir. Bu çıktıdaki servisler bilgisayara gelen ağ bağlantılarını dinliyorlardır.

```
$ netstat -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp   0      0      eol:domain    *:*             LISTEN
tcp   0      0      ceol-eth.fri:domain    *:*    LISTEN
tcp   0      0      *:ssh         *:*             LISTEN
tcp   0      0      ceol:ipp      *:*             LISTEN
tcp   0      0      ceol:postgresql    *:*        LISTEN
tcp   0      0      ceol:smtp     *:*             LISTEN
```

TCP ve UDP portları aynı bilgisayardaki portları kullanmaya ya da aynı anda birkaç servise erişmeye izin verir. Birçok protokol sabit port numarası kullanır. Bunların listesi /etc/services dosyası içinde vardır.

Bu örnekteki çıktı bilgisayarın sağladığı DNS sunucu (domain servisi), yazım için olan bir CUPS sunucu (ipp servisi) ve mail sunucu (smtp servisi)nun ceol adresi üzerinde olduğunu anlatır. Hatta bu çıktı tüm yapılandırılımış adreslerden ssh (güvenli kabuk) sunar.

"netstat -tl" bağlantı hata çözümlerinde ağ servisleri ile önemli bir araçtır. Eğer servisler burada görünmüyorsa fakat aslında onun olduğunu düşünüyorsanız, bu bazı şeylerin yapılandırmasının yanlış olduğunu belirtir - possibly it does not use the correct address/port, or something went catastrophically wrong when the service was started so it is not running at all.

"-u" seçeneği "-t" nin yerinde olduğunda UDP tabanlı servisleri gösterir, ve "-p" seçeneği kullanıldığında isimleri ve süreçlerin servislere sağladığı PID ler gösterilir. Komutu root kullanıcı olarak çağırırsanız, sadece ikinci özellik mevcut olur.

"-n" seçeneği IP adresleri ve port numaraları yerine isimleriyle birlikte her şeyi gösterecektir. Bu bazen daha açıklayıcıdır, at least as long as you have a working knowledge of the port numbers.

```
$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address        Foreign Address State
tcp   0      0      127.0.0.1:53         0.0.0.0:*       LISTEN
tcp   0      0      192.168.178.130:53   0.0.0.0:*       LISTEN
tcp   0      0      0.0.0.0:22           0.0.0.0:*       LISTEN
```

"netstat -s" aşağıdakine gibi bir istatistik gösterir.

```
Ip:
    145845 total packets received
    8 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    145837 incoming packets delivered
    138894 requests sent out
    16 outgoing packets dropped
    172 dropped because of missing route
Icmp:
    30 ICMP messages received
    0 input ICMP message failed.
```

Ve "netstat -r" komutu aslında "route" komutuna benzer (parametre olmadan).

Alıştırmalar

1. ping komutunu kullanarak emin ol iyi bilinen bir sunucuya ulaşablirsin (belki www.google.com olabilir).
2. dig komutunu kullanarak IP adresinin www.heise.de adresine uygunluğunu kontrol edin.
3. dig's trace seçeneği yönetici seviyesi isim sunucularından başlayarak programın bir isim için arama zincirini tamamlamasını belgelemesine neden olur. Bazı ilginç isimler için dene ve ara basamakları kontrol edin.
4. Bilgisayarınız hangi ağ servisini sunar? Bunların hangisini diğer bilgisayarlardan ulaşabilir?

Bu bölümdeki komutlar

dig DNS bilgisi arar (bunun için çok uygundur)

ifconfig ağ arayüzünü yapılandırır

netstat ağ bağlantısı hakkında bilgi gösterir, suncular, yönlendirmeler gibi

ping ICMP'yi kullanarak temel ağ bağlantısını kontrol eder

ping6 temel ağ bağlantısını kontrol eder (IPv6 için)

route Linux çekirdeğinin yönlendirme tablosunu yönetir

Özet

- Ağ protokollerinin üç çeşidi vardır: orta erişimli protokoller, haberleşme protokolleri, ve uygulama protokolleri.
- Bir protokol yığınında her protokol hemen bir üstündeki ya da bir altındaki ile haberleşebilir.
- TCP/IP haberleşme protokolü olan IP, TCP (güvenilir ve bağlantı yönlendirilebilir ve UDP (güvenli değil ve bağlantısız), kontrol protoklü ICMP, çok sayıda TCP veya UDP tabanlı uygulama protokolleri içerir.
- Bilgisyarların internette tekil bir IP adresleri vardır.
- Yönlendirmeler bilgisayarların arasında birbirlerine direkt bağlanmadan haberleşmelerini sağlarlar.
- Ağa bağlanırken, Linux bilgisayar bir IP adres ister, alt ağdaki adres ile maskelenir, varsayılan bir ağ geçidi, ve adresin üzerinde bulunduğu en az bir adres vardır.

- DNS IP adreslerini ev sahibi isimlerine eşleştiren ya da bunun tam tersidir(diğer şeyler arasında).
- IPv6, IPv4 ün varisidir, IPv6 da çeşitli sınırlamalar kaldırılmıştır ve çeşitli gelişmeler içerir.
- ifconfig ve route ağı elle yapılandırmada kullanılır. Dağıtımlar ağ yapılandırması için çeşitli şemalar kullanırlar.
- ifconfig, ping, dig ve netstat programları iyi ağ sorunu çözümleri için kullanılır.