

Open Source Enterprise Centre  
GNU/Linux Sertifikasyon Programı

Linux 101 Ders Notları

# İçindekiler

<b>1</b>	<b>Linux, Knoppix ve bu eğitim hakkında</b>	<b>1</b>
1.1	Neden Linux ?	1
1.2	Knoppix Linux	1
1.3	Bu eğitim neleri içeriyor?	2
<b>2</b>	<b>bash'e giriş</b>	<b>2</b>
2.1	Kabuk	2
2.2	bash mi kullanıyorsunuz?	2
2.3	bash hakkında	2
2.4	"cd" kullanımı	3
2.5	Yollar	3
2.6	Mutlak Yollar	3
2.7	Görelî/Bağıl yollar	3
2.8	".." kullanımı	3
2.9	".." kullanımı, devamı	4
2.10	Görelî/Bağıl Yol Örnekleri	4
2.11	"." Kavramı	4
2.12	cd ve ana dizin	4
2.13	Diğer Kullanıcıların Home Dizinleri	4
<b>3</b>	<b>Linux Komutlarının Kullanımı</b>	<b>5</b>
3.1	"ls" tanımı	5
3.2	Uzun izin listeleri	5
3.3	Uzun izin listeleri, devam	5
3.4	Dizinlere bakmak	5
3.5	Özyineli listeler ve inode listeleri	6
3.6	inode'ları anlamak, bölüm 1	6
3.7	inode'ları anlamak, bölüm 2	6
3.8	inode'ları anlamak, bölüm 3	6
3.9	mkdir	7
3.10	mkdir -p	7
3.11	touch	7
3.12	echo ve yönlendirme	7
3.13	echo ve yönlendirme	7
3.14	cat ve cp	8
3.15	mv	8
<b>4</b>	<b>Linkler yaratmak ve dosyaları silmek</b>	<b>8</b>
4.1	Katıbağlantılar (hard links)	8
4.2	Katıbağlantılar, devamı	8
4.3	Sembolik bağlantılar	8
4.4	Sembolik bağlantılar, devamı	9
4.5	Sembolik Bağlantılar, Ayrıntılar	9
4.6	Sembolik Bağlantılar, Ayrıntılar	9
4.7	Sembolik Bağlantılar, Ayrıntılar	9
4.8	Sembolik Bağlantılar, Ayrıntılar	9
4.9	Sembolik Bağlantılar, Ayrıntılar	10
4.10	Sembolik Bağlantılar, Ayrıntılar	10
4.11	rm	10
4.12	rmdir	10
4.13	rm ve izinler	10

<b>5</b>	<b>joker karakterler, Giriş</b>	<b>11</b>
5.1	Joker karakterler (wildcards), Giriş . . . . .	11
5.2	Joker karakterler, Giriş, devamı . . . . .	11
5.3	Uyuşmazlıkları(non-matches) Anlamak . . . . .	11
5.4	UyuşmazlıklarıAnlamak, devamı . . . . .	11
5.5	Joker karakter sözdizimi: * . . . . .	11
5.6	Joker karakter sözdizimi: ? . . . . .	12
5.7	Joker karakter sözdizimi: [] . . . . .	12
5.8	Joker karakter sözdizimi: [!] . . . . .	12
5.9	Joker karakter ile ilgili uyarılar . . . . .	12
5.10	Joker karakter ile ilgili uyarılar, devamı . . . . .	13
5.11	Tek Tırnak ile Çift Tırnağın Kıyaslanması . . . . .	13
<b>6</b>	<b>Düzenli İfadeler</b>	<b>13</b>
6.1	Düzenli ifade nedir? . . . . .	13
6.2	Glob kıyaslaması . . . . .	13
6.3	Basit bir dizi parçası . . . . .	13
6.4	Basit karakter dizisi parçasımanlamak . . . . .	13
6.5	Metakarakterler . . . . .	14
6.6	[] Kullanımı . . . . .	14
6.7	[^] Kullanımı . . . . .	14
6.8	Farklılaşan sözdizimi . . . . .	14
6.9	"*" metakarakteri . . . . .	15
6.10	Satır Başıve Satır Sonu . . . . .	15
6.11	Tam Satır Düzenli İfadeler . . . . .	15
<b>7</b>	<b>FHS ve dosyalarıbulmak</b>	<b>15</b>
7.1	Dosya Sistemi Hiyerarşi Standardı . . . . .	15
7.2	İki Bağımsız D.S.H.S Kategorisi: . . . . .	16
7.3	/usr dizinindeki ikincil hiyerarşi . . . . .	16
7.4	Dosyalarıbulmak . . . . .	16
7.5	PATH . . . . .	17
7.6	PATH'i düzenlemek . . . . .	17
7.7	"which" hakkında her şey . . . . .	17
7.8	find . . . . .	18
7.9	find ve joker karakterler . . . . .	18
7.10	find ile küçük büyük harf ayırmıyapmadan aramak . . . . .	18
7.11	find ve düzenli ifadeler . . . . .	18
7.12	find ve türler . . . . .	18
7.13	find ve 'mtime'lar . . . . .	18
7.14	-daystart seçeneği . . . . .	19
7.15	-size seçeneği . . . . .	19
7.16	Bulunan dosyaların işlenmesi . . . . .	19
7.17	locate . . . . .	20
7.18	updatedb kullanımı . . . . .	20
7.19	slocate . . . . .	20
<b>8</b>	<b>Süreç (process) Kontrolü</b>	<b>20</b>
8.1	xeyes'ıbaşlatmak . . . . .	20
8.2	Süreci durdurmak . . . . .	21
8.3	fg ve bg . . . . .	21
8.4	"&" Kullanımı . . . . .	21
8.5	Birden çok sayıda arkaplan süreci . . . . .	21
8.6	Sinyallere giriş . . . . .	22
8.7	SIGTERM ve SIGINT . . . . .	22
8.8	Büyük ölüm . . . . .	22
8.9	nohup . . . . .	22
8.10	ps komutu ile süreçleri listelemek . . . . .	23
8.11	Hem ormanihem de ağaçlarıgörebilmek . . . . .	23

8.12	"u" ve "l" ps seçenekleri . . . . .	23
8.13	"top" kullanımı . . . . .	23
8.14	nice . . . . .	24
8.15	renice . . . . .	24
<b>9</b>	<b>Yazı İşleme</b>	<b>24</b>
9.1	Daha uzun bir boru hattı . . . . .	25
9.2	Metin işleme kasırgasıbaşıyor! . . . . .	25
9.3	echo . . . . .	25
9.4	cat, sort ve uniq . . . . .	26
9.5	wc, head, ve tail . . . . .	26
9.6	tac, expand ve unexpand . . . . .	26
9.7	cut, nl ve pr . . . . .	26
9.8	tr, sed ve awk . . . . .	26
9.9	od, split, ve fmt . . . . .	26
9.10	Paste, join, ve tee . . . . .	26
9.11	Yönlendirme (redirection) Kasırgası . . . . .	27
9.12	>> Kullanımı . . . . .	27
<b>10</b>	<b>Sistem ve ağ dökümantasyonu</b>	<b>27</b>
10.1	Linux sistem döküman tipleri . . . . .	27
10.2	Kılavuz sayfaları . . . . .	27
10.3	Kılavuz sayfaları, devamı . . . . .	28
10.4	Kılavuz sayfa bölümleri . . . . .	28
10.5	Çoklu kılavuz sayfaları . . . . .	28
10.6	Doğru kılavuz sayfayıbulmak . . . . .	28
10.7	apropos ile ilgili herşey . . . . .	28
10.8	Kılavuz Yolu (MANPATH) . . . . .	29
10.9	GNU bilgisi . . . . .	29
10.10	Linux Dökümantasyon Projesi . . . . .	29
10.11	LDP'ye genel bakış . . . . .	29
10.12	Mail Listeleri . . . . .	30
10.13	Haber grupları . . . . .	30
10.14	Satıcıfirmalar ve üçüncü-parti web siteleri . . . . .	30
10.15	Linux danışmanları . . . . .	30
10.16	Yazılım ve donanım sağlayıcıları . . . . .	31
10.17	Geliştiricilerin kaynakları . . . . .	31
<b>11</b>	<b>Linux izin modeli</b>	<b>31</b>
11.1	Bir kullanıcı, bir grup . . . . .	31
11.2	"ls -l" incelemesi . . . . .	31
11.3	Ben kimim? . . . . .	32
11.4	Hangi grupların üyesiyim? . . . . .	32
11.5	Kullanıcıve Grup HaklarınıDeğiştirmek . . . . .	32
11.6	Sahipliğin Özyineli Olarak Değiştirilmesi . . . . .	32
11.7	chmod Komutuna Giriş . . . . .	32
11.8	kullanıcı/grup/diğerleri Parçacık Yapısı . . . . .	33
11.9	İzinleri Sıfırlamak . . . . .	33
11.10	Sayısal Kalıplar . . . . .	33
11.11	Sayısal izin ifadelerinde sözdizim yapısı . . . . .	33
11.12	umask . . . . .	34
11.13	suid ve sgid'ye giriş . . . . .	34
11.14	suid . . . . .	34
11.15	suid/sgid uyarıları . . . . .	35
11.16	suid ve sgid'nin değiştirilmesi . . . . .	35
11.17	İzinler ve dizinler . . . . .	35
11.18	Dizinler ve sgid . . . . .	35
11.19	Dizinler ve silme işlemi . . . . .	36
11.20	Özel ilk basamak . . . . .	36

<b>12 vi Editörü</b>	<b>36</b>
12.1 Vi'a başlangıç . . . . .	36
12.2 Dosya açma, kaydetme ve çıkma . . . . .	37
12.3 Vi'da yazma . . . . .	37
12.4 Silme ve kopyalama . . . . .	37
12.5 Arama ve eşleştirme . . . . .	38
12.6 Diğer dosyaların metne eklenmesi . . . . .	38
12.7 Kabuk komutlarının çalıştırılması . . . . .	38
<b>13 Linux hesap yönetimi</b>	<b>38</b>
13.1 /etc/passwd dosyası . . . . .	38
13.2 /etc/passwd hakkında ipuçları . . . . .	39
13.3 /etc/shadow dosyası . . . . .	39
13.4 /etc/group dosyası . . . . .	39
13.5 Gruplara ilişkin notlar . . . . .	39
13.6 Elle kullanıcıve grup yaratma . . . . .	40
13.7 /etc/passwd dosyasınıdüzenlemek . . . . .	40
13.8 /etc/shadow dosyasınıdüzenlemek . . . . .	40
13.9 Parola belirleme . . . . .	40
13.10/etc/group dosyasınıdüzenlemek . . . . .	41
13.11Home dizin yaratma . . . . .	41
13.12Hesap yönetim araçları . . . . .	41
<b>14 Kullanıcıortamınıayarlama</b>	<b>41</b>
14.1 "fortune" programına giriş . . . . .	41
14.2 -login'i anlamak . . . . .	42
14.3 İnteraktifliğin test edilmesi . . . . .	42
14.4 /etc/profile ve /etc/skel . . . . .	43
14.5 export . . . . .	43
14.6 Değişkenlerin ihraç için işaretlenmesi . . . . .	43
14.7 Export ve set -x . . . . .	44
14.8 "set" ile değişkenlerin ayarlanması . . . . .	44
14.9 "Unset" ve "FOO=" farkı . . . . .	44
14.10Komutların çalışmalarımetkilemek için ihraç işlemi . . . . .	44
14.11"env" kullanımı . . . . .	45
<b>15 Midnight Commander (MC)</b>	<b>45</b>
15.1 Dizin panelleri arasında gezinme . . . . .	45
15.2 Fare desteği . . . . .	45
15.3 FTP işlemleri . . . . .	46
<b>16 Linux Dosya Sistemi</b>	<b>46</b>
16.1 Blok Aygıtlar . . . . .	46
16.2 Tüm Diskler ve Bölümler . . . . .	46
16.3 fdisk Kullanımı . . . . .	46
16.4 fdisk Kullanımı, Devamı . . . . .	47
16.5 Fdisk ve Ötesi . . . . .	48
16.6 Dosya Sistemlerini Yaratmak . . . . .	48
16.7 Dosya sistemlerini mount etmek . . . . .	48
16.8 Takas bölümü oluşturmak ve kullanmak . . . . .	49
16.9 Mount edilmiş dosya sistemlerini görmek . . . . .	49
16.10Mount Seçenekleri . . . . .	49
16.11fstab, Giriş . . . . .	50
16.12Örnek bir fstab Dosyası . . . . .	50
16.13Örnek bir fstab, Devamı . . . . .	51
16.14Dosya sistemlerini Unmount Etmek . . . . .	51
16.15fsck'ya Giriş . . . . .	52
16.16fsck ile İlgili Problemler . . . . .	52
16.17ext2 Dosya Sistemi . . . . .	52

16.18ext3 Dosya Sistemi . . . . .	52
16.19ReiserFS Dosya Sistemi . . . . .	53
16.20XFS Dosya Sistemi . . . . .	54
16.21JFS Dosya Sistemi . . . . .	54
16.22VFAT Dosya Sistemi . . . . .	54
<b>17 Sistemi Başlatmak</b>	<b>54</b>
17.1 MBR . . . . .	54
17.2 Çekirdek Açılış İşlemi . . . . .	54
17.3 /sbin/init Programı . . . . .	55
17.4 LILO'yu İnceleyelim . . . . .	55
17.5 LILO Kullanımı . . . . .	55
17.6 LILO Hakkında Önemli Not . . . . .	55
17.7 GRUB'ı İnceleyelim . . . . .	56
17.8 GRUB Kullanımı . . . . .	56
17.9 dmesg . . . . .	56
17.10/var/log/messages . . . . .	56
17.11Tek-kullanıcıModu . . . . .	57
17.12Tek-kullanıcı(Single-user) Modunu Kullanmak . . . . .	57
17.13Çalışma Seviyelerini Değiştirmek . . . . .	57
17.14Sistemi Düzgün Kapatmak . . . . .	57
17.15Sistemi Aniden Kapatmak . . . . .	58
17.16ÖntanımlıÇalışma Seviyesi . . . . .	58
<b>18 Çalışma Seviyeleri</b>	<b>58</b>
18.1 Tek-kullanıcıModu . . . . .	58
18.2 Çalışma Seviyeleri . . . . .	58
18.3 elinit Komutu . . . . .	59
18.4 Çalışma Seviyesi Etiketleri . . . . .	59
18.5 "now" ve "halt" . . . . .	59
18.6 init Konfigürasyonu . . . . .	59
<b>19 Dosya Sistemi Kotaları</b>	<b>60</b>
19.1 Kotalara Giriş . . . . .	60
19.2 Çekirdek Desteği . . . . .	60
19.3 Dosya Sistemi Desteği . . . . .	60
19.4 Kotaların Konfigürasyonu . . . . .	60
19.5 Kotaların Konfigürasyonu, Devamı . . . . .	60
19.6 quota Komutu . . . . .	61
19.7 KotalarıGörüntülemek . . . . .	61
19.8 edquota . . . . .	61
19.9 edquota, Devamı . . . . .	61
19.10edquota'yıAnlamak . . . . .	62
19.11Değişiklikler Yapmak . . . . .	62
19.12KotalarıKopyalamak . . . . .	62
19.13Grup Kısıtlamaları . . . . .	62
19.14repquota Komutu . . . . .	62
19.15repquota Seçenekleri . . . . .	63
19.16KotalarıGörüntülemek . . . . .	63
19.17Mühletleri Belirlemek . . . . .	63
19.18KotalarıAçılışta Kontrol Etmek . . . . .	63
<b>20 Sistem Logları</b>	<b>64</b>
20.1 syslogd'ye Giriş . . . . .	64
20.2 LoglarıOkumak . . . . .	64
20.3 LoglarıKuyruktan Takip Etmek . . . . .	64
20.4 Log Dosyalarınıngrep ile Kullanmak . . . . .	65
20.5 Log Dosyalarının Özeti . . . . .	65
20.6 syslog.conf . . . . .	65

20.7 syslog.conf, Devamı . . . . .	65
20.8 Yeniden Yükleme ve Ek Bilgiler . . . . .	65
20.9 Güvenlik Notu . . . . .	66
20.10logrotate . . . . .	66
20.11İleri Başlık: klogd . . . . .	66
20.12İleri Başlık: Alternatif Loglayıcılar . . . . .	66

# 1 Linux, Knoppix ve bu eğitim hakkında

## 1.1 Neden Linux ?

Bu eğitime geldiğinizde göre hepinizin kafasında üç aşağı beş yukarı oluşmuş bir Linux fikri vardır. Hatta muhtemelen bazılarınız Linux kurmuştur bile. Fakat kurmuş olanlarınızın bile kafasında 'Neden Linux ?' sorusu vardır.

Gerçekten 'Neden Linux ?' Bu soruya cevap vermeden önce bir miktar Linux'u yüzeysel bile olsa tanımak gerekmektedir.

Linux 1992 yılında Linus Torvalds tarafından başlatılan bir projedir. Temel olarak 1970'lerden beri süregelen Unix işletim sisteminin evlerde yaygın olan i386 (PC) tabanlı makinalarda çalışmasını hedeflemiştir. Bu başlangıcın devamında proje birçok kişi tarafından desteklenmiş ve şu anda binlerce yazılımcının bir şekilde ilgilendiği dev bir proje halini almıştır.

Linux diye bildiğimiz dağıtımların hepsi, Linux kerneli (çekirdeği) ve onun etrafındaki GNU yazılımlardan oluşan bir Unix sistemidir aslında. O zaman asıl sorumuz neden Unix olacaktır ?

Unix temelde Windows ve DOS kullananlarımızın komut satırı diye bildiği ortamdan ve X-Windows sisteminden oluşur. Bu ikisinin de geçmişi hem DOS'tan hem de Windows'tan çok daha eskilere dayanır.

Unix'in 1970'lerden beri temel aldığı en önemli özellik çok görevli (multi-task) ve çok kullanıcı (multi user) olmasıdır. Bütünleşik bir işletim sistemi olan Unix, bu iki özelliğini hem komut satırı ortamında hem de GUI ortamı olan X-Windows'da korur. Dahası X-Windows'da yapabileceğiniz hemen herşeyi komut satırından (konsoldan) gerçekleştirebilirsiniz.

Az önce bahsettiğimiz gibi Unix'in temel özelliği çok görevli ve çok kullanıcı olmasıdır. Bu sebeple Unix'de çok ciddi bir kullanıcı kısıtlaması vardır. Kullanıcılar, ancak izinleri olan harddisk alanında, izinleri olan komutları, kullanmalarına izin verilen hafıza miktarında ve hatta CPU gücü ile yaparlar. Bir kullanıcının bir programının çökmesi bir başka kullanıcıyı ve makinanın genel işletimini engellemez.

Otuz yıldır geliştirilen bu özellikler Unix'i diğer işletim sistemlerinden ayırmakta ve dünyanın en çok tercih edilen sonucu platformu yapmaktadır.

İşte bu noktada Unix'le tanışıklığı biraz daha ilerlemiş bazıları 'iyi de kullanması kesinlikle daha zor, bir çok işi konsoldan yapıyorsunuz, Windows gibi GUI'den herşeyi halletmek varken, 2006 senesinde ben niye konsolu kullanayım' diyebilir.

Burada bir şeye açıklık getirmek lazım. Konsol, dinazorluktan, ya da eski kafalılıktan hala kullanılan bir şey değildir. Konsol kullanılmaktadır, zira bir sistem yöneticisinin işlemleri en hızlı ve en kolay yoldan yapmasının yolu konsoldur. Konsoldan bir iki satır ile hallettiğiniz bir emri GUI'den yapmak genelde çok daha uzun zamanınızı alır. Belki daha da önemlisi, konsol 'ne yapması gerektiğini bilenler' içindir, şuraya buraya tıklayayım da nasıl olsa hallederim bu işi diyerek sistemi kurcalayanlar için değil.

Bir başka önemli nokta da Unix'in gücünü biraraya gelmiş binlerce küçük programdan alıyor olmasıdır. Bu kurs süresince size bu programları ihtiyacımız doğrultusunda nasıl birbiri ile ilintileyeceğimizi, nasıl istediğimizi gibi sonuçlar alacağımızı göstereceğiz.

Kısa bir örnek ls komutu olabilir. DOS ortamındaki dir komutunun karşılığı olan ls komutu size basitçe bir dizinin içeriğini gösterir. Diyelim ki bu dizin uzun ve içeriğini sayfa sayfa görmek istiyoruz. Bu noktada herhangi bir text dosyasını bize sayfa sayfa gösteren more komutu ile ls komutunu birleştiririz. En basitinden ls -al — more dediğinizde ls komutunun çıktısı takip edebileceğiniz gibi sayfa sayfa karşınıza gelir. Mesela ls komutu bir dizindeki dosyaları gösterirken, ls — wc -l komutu ls komutunun çıktısını wc (wordcount) komutuna göndererek o dizinde kaç dosya olduğunu gösterir. Görüyorsunuz iki alakasız komutla ls komutuna yeni fonksiyonlar kazandırdık. Dahası bir dizinde kaç dosya olduğunu bilgisayara saydırdık. Kolaylıkla bir başka komutu da bu sisteme ekleyip, her dosyayı bizim için teker teker yazıcıya yollamasını sağlayabilirdik veya başka bir şey. Birkaç kelimelik bu tür komutlarla yapabileceğiniz şeylerin ne kadar derine gidebildiğini görseniz şaşarsınız. Dahası bir kabuk ortamı gibi bir sahte-programlama dili ile birleştirildiğinde bu komutlar gerçekten uygulama düzeyinde işlerin altından kalkabilirler.

Linux kullanmayı öğrenmenizin sebeplerinden biri işte budur. Gerektiği gibi bilgisayarınızı kontrol edebilmeniz, ona istediğiniz işlemleri yaptırabilmeniz. Kendinizi ona değil onu kendinize uydurabilmeniz.

## 1.2 Knoppix Linux

Bu kurslar boyunca Knoppix adında bir dağıtım kullanacağız. CD'den boot eden bir dağıtım olan Knoppix için bilgisayarınızda harddiskiniz olması bile gerekli değil, zira 96MB ve üstü RAM'mız varsa Knoppix harddiskinizi kullanmaz bile. Dahası Knoppix ile yaptığımız bir yanlışlıkla harddiskinizdeki bilgilere zarar verme ihtimaliniz çok düşüktür. İşletim sistemini ise bozamazsınız zaten, çünkü o CD Rom'da yazılamaz bir halde duruyordur.



Öte yandan yaklaşık 2GB yazılım yüklü olan Knoppix cd'si hemen hemen bütün ihtiyaçlarınıza cevap verebilecek özelliklere sahiptir. Burada yaptığımız işleri, eve gittiğinizde kolayca deneyebilir, Linux kurulumu yapmadan (ileride onu da yapacağız) Linux kullanabilirsiniz.

Bilgisayarınızdaki donanımları tanıması, Usb-diskinizden, ethernet kartınıza kadar onları tanıyıp kullanabilmesi Knoppix'in en önemli yanlarından biri. Dahası daha açılıştan dili Türkçe olarak seçebilir ve kullanabilirsiniz. Ethernet, kablomodem, DSL ve seri modem ile Internet'e çıkabilen Knoppix dağıtımı sadece öğrenme amaçlı değil, kullanım açısından da eşsizdir. Belki tek açmaz/problem, ülkemizde çok sayıda bulunan winmodem tabir edilen PCI modemlerle uyumsuz olmasıdır. Yukarıda saydığımız diğer yöntemlerle ve external modemlerle Internet'e çıkmakta ve işlemlerinizi yapmakta bir sorun yaşamazsınız. Bütün bu sebeplerle biz de Knoppix dağıtımını kullanacağız eğitim boyunca.

Gelelim eğitimin içeriğine.

### 1.3 Bu eğitim neleri içeriyor?

"Linux Temellerine" hoşgeldiniz. Bu kurda sizlere bash (Linux'daki standart kabuk -shell-)’i tanıstırarak, ls, cp, mv gibi standart Linux komutlarının tüm potansiyelini nasıl kullanabileceğinizi gösterecek, Linux'un izin ve sahiplik modellerini ve daha birçok kavramı anlatacağız.

Bu kurun sonunda, Linux temelleri konusunda sağlam bir zemin elde etmiş olacak ve hatta bazı temel Linux sistem yönetim görevlerini öğrenmeye hazır halde olacaksınız. Bütün bu serinin sonunda ise bir Linux sistem yöneticisi olmak için gerek duyabileceğiniz bütün bilgilere sahip olacaksınız.

## 2 bash'e giriş

### 2.1 Kabuk

Eğer daha önce bir Linux sistemi kullandıysanız sisteme girdiğinizde aşağıdakine benzer bir komut satırı ile karşılaşacağınızı bilirsiniz:

```
$
```

Bu komut satırı çok farklı görünebilir. Sisteminizin üzerinde çalışılan makina (hostname) bilgisini, içinde bulunduğunuz dizini veya her iki bilgiyi birden yansıtır olabilir. Ancak nasıl görünürse görünsün kesin olan bir şey vardır: Bunu görüntüleyen programa "kabuk" (shell) denir ve çok büyük ihtimalle sisteminizde sizi karşılayan bash kabuğudur.

### 2.2 bash mi kullanıyorsunuz?

bash kullamp kullanmadığınızı şu şekilde kontrol edebilirsiniz:

```
$ echo $SHELL
/bin/bash
```

Eğer üstteki ifade bir hata verirse veya yukarıdaki örneğe benzemeyen bir sonuç verdiyse, bash'ten farklı bir kabuk kullanıyor olabilirsiniz. Bu durumda bile bu eğitimin büyük bir bölümü hala geçerli olacaktır, fakat bash'a geçiş yapmanız bu eğitim sürecinde sizin için önemli bir avantaj olacaktır. (chsh komutunu kullanarak kabuğu değiştirmek için gerekli bilgi için eğitim serimizin 2. Bölümüne bakın.)

### 2.3 bash hakkında

"Bourne-again shell" ifadesi için bir tür kısaltma olan Bash, pek çok Linux sistemde öntanımlı kabuk olarak karşınıza çıkar. Kabuğun görevi verdiğiniz komutlara uyarak Linux sisteminizle etkileşime geçmenizi sağlamaktır. Komut vermeyi bitirince kabuğa exit ya da logout komutu ile kendini sonlandırmasını söyleyebilirsiniz, hemen ardından karşınıza login ekranı gelir. Bu arada, kabuk ile çalışırken Control-D tuş kombinasyonuna basarak da bash kabuğunu sonlandırabilirsiniz.

## 2.4 "cd" kullanımı

Sizin de farkına varmış olabileceğiniz gibi bash komut satırı dünyanın en heyecan verici şeyi sayılmaz. İsterseniz şimdi de biraz dosya sistemi içinde gezinmek için bash kullanalım. Komut satırında iken aşağıdaki komutu verin (\$ karakterini yazmadan):

```
$ cd /
```

Bu komut ile bash kabuğuna / içinde çalışmak istediğimizi söylemiş olduk. Bu aynı zamanda kök (root) dizin olarak da bilinir; dosya sistemindeki tüm dizinler bir ağacı oluştururlar ve "/" bu ağacın en tepesinde ya da bu ağacın kökü kabul edilir.

cd komutu o anda çalışmakta olduğunuz dizini belirler. cd komutunun bir adı da "halihazırda çalışılan dizin (current working directory)" dir.

## 2.5 Yollar

bash kabuğunun halihazırda çalıştığı dizini görmek için yazılabilecek komut:

```
$ pwd  
/
```

Yukarıdaki örnekte, cd komutuna cevap olarak gelen / elemanına yol (path) denir ve cd komutuna gitmek istediğimiz yeri söyler. Burada, / elemanı mutlak yol (absolute path) olarak kullanılmış. Gidilecek olan dizinin, dosya sistemi ağacında, kökten itibaren ifade edilişine mutlak yol denir.

## 2.6 Mutlak Yollar

Birkaç mutlak yol örneği:

```
/dev  
/usr  
/usr/bin  
/usr/local/bin
```

Gördüğünüz gibi bütün mutlak yol ifadelerinin ortak noktası / ile başlamalarıdır. Örneğin /usr/local/bin yolunu yazarak, cd komutuna önce / dizinine gitmesini, oradan usr dizinine, daha sonra da usr dizininin altında local ve onunda altında bin dizinine gitmesi söyleniyor. Mutlak yol ifadeleri, her zaman / den başlayarak kullanılır.

## 2.7 Göreli/Bağıl yollar

Diğer yol türü göreli yoldur. Bash, cd ve diğer komutlar bu tür yolları daima içinde bulunduğunuz dizine göre yorumlar. Göreli yollar kesinlikle / ile başlamaz. Yani eğer /usr dizini içinde isek:

```
$ cd /usr
```

/usr/local/bin dizinine gitmek için:

```
$ cd local/bin  
$ pwd  
/usr/local/bin
```

yazabiliriz.

## 2.8 ".." kullanımı

Göreli yolların içinde bir ya da daha çok .. dizini bulunabilir. ".." dizini özel bir dizindir ve bulunduğumuz dizinin üstündeki dizini gösterir. Yukarıdaki örnekten devam edecek olursak:

```
$ pwd  
/usr/local/bin  
$ cd ..  
$ pwd  
/usr/local
```

Gördüğünüz gibi şimdi içinde bulunduğumuz dizin /usr/local dizinidir. Bu şekilde içinde bulunduğumuz dizine göre "geriye" doğru gidebiliriz başka bir deyişle "yukarı" doğru çıkabiliriz.

## 2.9 “..” kullanımı, devamı

Ek olarak, içinde bulunduğumuz dizinin de yer aldığı yani bir üst dizini temsil eden “..” dizinini de kullanabiliriz. Örneğin:

```
$ pwd
/usr/local
$ cd ../share
$ pwd

/usr/share
```

## 2.10 Göreli/Bağıl Yol Örnekleri

Görelî yollar biraz karmaşık görülebilir. Burada sonuçta varılacak hedef dizinin görülmeyeceği bir kaç örnek mevcuttur. Bu komutlar yazıldıktan sonra hangi dizin içerisine gidileceğini tahmin etmeye çalışınız:

```
$ cd /bin
$ cd ../usr/share/zoneinfo

$ cd /usr/X11R6/bin
$ cd ../lib/X11

$ cd /usr/bin
$ cd ../bin/
```

Şimdi bunları bir de yazarak nereye gittiğine bakınız ve doğru tahmin edip etmediğinizi gözlemleyiniz.

## 2.11 “.” Kavramı

cd konusunu bitirmeden önce birkaç küçük şeyden daha bahsetmek gerekmektedir. İlk olarak, “.” ile temsil edilen ve “halihazırda bulunduğunuz dizin” anlamına gelen özel bir sembol daha vardır. Bu dizin cd komutuyla birlikte kullanılmaz, genellikle bulunduğunuz dizindeki bazı programları çalıştırmak için kullanılır:

```
$ ./program
```

Yukarıdaki örnekte halihazırda çalışılan dizin (ardından) program isimli (çalıştırılabilir) dosya çalıştırılacaktır.

## 2.12 cd ve ana dizin

Ana dizinimizi değiştirmek isteseydik şöyle yazabilirdik:

```
$ cd
```

Parametresiz kullanımda cd, kullanıcıyı home dizinine götürecektir. root kullanıcılar için home dizini /root ve diğer kullanıcılar için ise /home/kullanıcıadı şeklindedir. Peki ya acaba kendi home dizinimiz içerisinde bir dosya belirlemek istersek ne yapmak gerekir? Mesela bir dosya argümanını program komutuna parametre olarak göndermek isteyelim. Eğer dosya kendi home dizinimiz içerisinde yer alıyorsa, yazılabilecek komut aşağıdaki şekilde düşünülebilir:

```
$ ./program /home/knoppix/dosyam.txt
```

Fakat yukarıdaki örnekte olduğu gibi her zaman tam yolu yazmak çok verimli olmayabilir. Neyse ki ~(tilda) karakteri yardımıyla aynı işi daha kolay bir şekilde halletmek mümkündür. Şöyle ki:

```
$ ./program ~/dosyam.txt
```

## 2.13 Diğer Kullanıcıların Home Dizinleri

Kabuk için ~ komutunun, kullanıcının home dizini anlamına geldiğini söylemiştik. Bunun yanında, yine ~ kullanılarak başka kullanıcıların da home dizinlerine erişmek mümkündür. Örneğin, knoppix’in home dizininde yer alan ogrencidosyasi.txt isimli dosyaya erişmek istiyorsak aşağıdaki komut yazılabilir:

```
$ ./program ~knoppix/ogrencidosyasi.txt
```

## 3 Linux Komutlarının Kullanımı

### 3.1 "ls" tanımı

Şimdi ls komutuna hızlıca göz atacağız. Muhtemelen ls komutunu önceden biliyorsunuz ve sadece onu yazmanın halihazırda bulunduğunuz dizinin içeriğini listeleyeceğini biliyorsunuz:

```
$ cd /usr
$ ls
bin doc etc games include info lib local man sbin share src X11R6
```

-a seçeneğini belirterek bir dizindeki dosyaların tümünü, gizli dosyalar - ile başlayanlar- da dahil olmak üzere görebilirsiniz. Aşağıdaki örnekte de görebileceğiniz gibi, ls -a . ve .. özel dizin linklerini de gösterir:

```
$ ls -a
.  .. bin doc etc games include info lib
local man sbin share src X11R6
```

### 3.2 Uzun dizin listeleri

ls komut satırında bir ya da birden fazla dosya ya da dizin belirtebilirsiniz. Eğer bir dosya belirtirseniz, ls sadece o dosyayı gösterecektir. Eğer bir dizin belirtirseniz, ls bu dizinin içeriğini gösterecektir. -l seçeneği, dosya haklarını, sahibi olan kişi ve grupları, değişikliğe uğradığı zamanı ve boyut bilgilerini görmek istediğinizde işinize yarayacaktır.

### 3.3 Uzun dizin listeleri, devam

Aşağıdaki örnekte -l seçeneğini /usr dizininin tam bir listesini almak için kullanıyoruz.

```
$ ls -l /usr
toplam 706
drwxr-xr-x  2 root    root      303104 2003-03-23 21:10 bin
drwxr-xr-x  4 root    root      73728 2003-03-23 19:50 doc
drwxr-xr-x  3 root    root       2048 2002-07-04 18:54 etc
drwxr-xr-x  2 root    root      8192 2003-03-23 16:36 games
drwxr-xr-x 63 root    root     32768 2003-03-22 19:14 include
lrwxrwxrwx  1 root    root        10 2002-08-08 15:02 info -> share/info
drwxr-xr-x 124 root    root    217088 2003-03-23 21:10 lib
drwxrwsr-x  9 root    staff     2048 2001-05-12 02:04 local
drwxr-xr-x  6 root    root     2048 2002-08-21 22:09 man
drwxr-xr-x  2 root    root    53248 2003-03-22 22:20 sbin
drwxr-xr-x 194 root    root    24576 2003-03-23 20:30 share
drwxrwsr-x  4 knoppix knoppix   2048 2003-03-23 18:47 src
drwxr-xr-x  6 root    root     2048 2003-02-13 08:36 X11R6
```

İlk sütun listedeki her eleman için izinleri gösterir. Bu bilgiyi nasıl yorumlamanız gerektiğini birazdan açıklayacağız. Bir sonraki kolon ise her bir dosya sistemi nesnesinin sahip olduğu linkleri gösterir, daha sonra değinmek üzere bu özelliği şimdilik geçeceğiz. Üçüncü ve dördüncü sütunlar ise sırasıyla dosyanın sahibini ve ait olduğu grubu belirtir. Beşinci sütun ise dosya büyüklüğüdür. Altıncı sütun "son müdahale edilen tarih" veya "mtime"dır. Son sütunda ise nesnenin ismi vardır. Eğer nesne bir sembolik link ise, - > işareti ve devamında sembolik linkin işaret ettiği adresi görürsünüz.

### 3.4 Dizinlere bakmak

Bazen bir dizinin içine değil de kendisine bakmak istersiniz. Bu gibi durumlar için -d seçeneğini kullanabilirsiniz, bu sayede ls dizinlerin içine değil kendilerine bakar.

```
$ ls -dl /usr /usr/bin /usr/X11R6/bin ../share
drwxr-xr-x 194 root    root    24576 2003-03-23 20:30 ../share
lrwxrwxrwx  1 root    root      12 2003-03-23 21:05 /usr -> /KNOPPIX/usr
drwxr-xr-x  2 root    root   303104 2003-03-23 21:10 /usr/bin
drwxr-xr-x  2 root    root   20480 2003-03-22 18:36 /usr/X11R6/bin
```

### 3.5 Özyineli listeler ve inode listeleri

Bir dizinin kendisine bakmak için `-d` kullanabileceğinizi gördünüz ancak `-R` ile tam tersini de yapabilirsiniz yani sadece bir dizinin içinde değil zincirleme olarak o dizinin içinde bulunan tüm dizinlerin ve onların da içinde bulunan ve... Bu şekilde mevcut tüm dizinlerin içine bakabilirsiniz! Bunun çıktısını burada göstermiyoruz çünkü böyle bir komutun çıktısı epey uzun olur ancak siz kendi sisteminizde hemen `ls -R` ve `ls -Rl` komutlarının deneyip sonucu görebilirsiniz.

Son olarak, `-i` seçeneği dosya sistemindeki nesnelerin inode numaralarını göstermek için kullanılabilir:

```
$ ls -i /usr
6603072 bin    6603408 games    6603776 lib      6604116 sbin    6602956 X11R6
6603184 doc    6603524 include  6603888 local    6604230 share
6603296 etc    6603644 info     6604004 man     6604346 src
```

### 3.6 inode'ları anlamak, bölüm 1

Bir dosya sistemindeki her nesneye özgün bir indeks numarası verilir, bu indeks'e de inode denir. Bu biraz önemsiz, ya da gereksiz detay gibi görünebilir ama bir çok dosya sistemi işlemini anlamamanın temeli inode mantığını anlamayı gerektirir. Örnek olarak, her dizinde gördüğümüz `..` bağlantılarını düşünelim. Gerçekten `..` dizininin ne olduğunu tam olarak anlayabilmek için `/usr/local`'ın inode numarasına bir bakalım:

```
$ ls -id /usr/local
6603888 /usr/local
```

`/usr/local` dizininin inode numarası 6603888 imiş, şimdi de, `/usr/local/bin/..` 'in inode numarasına bakalım:

```
$ ls -id /usr/local/bin/..
6603888 /usr/local/bin/..
```

### 3.7 inode'ları anlamak, bölüm 2

Gördüğümüz gibi, `/usr/local/bin/..` ile `/usr/local`'ın inode numaraları aynı! Açığa çıkan bu bilgiden yararlanabileceğimiz nokta şu: Eskiden, `/usr/local`'ı dizinin kendisi olarak düşünürdük. Ama şimdi, keşfettik ki aslında 6603888 inode'u dizinin kendisi, ayrıca da bu inode'a işaret eden iki tane de dizin tanımı (link'i) var. Hem `/usr/local`, hem `/usr/local/bin/..` de 6603888 inode'una işaret ediyor.

### 3.8 inode'ları anlamak, bölüm 3

Aslında `ls -dl` komutu ile 6603888 numaralı inode'a kaç kere bağlantı yapıldığını, başka bir deyişle aynı yeri gösteren kaç farklı ismin dosya sisteminde mevcut olduğunu görebiliriz:

```
$ ls -dl /usr/local
drwxrwsr-x    9 root    staff          2048 2001-05-12 02:04 /usr/local
```

Soldan ikinci sütuna bakacak olursak görürüz ki `/usr/local` (inode 6603888) dizinine 9 bağlantı (link) vardır. Benim dizinimde bu inode'a bağlı olan birkaç yolu göstermem gerekirse:

```
/usr/local
/usr/local/.
/usr/local/bin/..
/usr/local/games/..
/usr/local/lib/..
/usr/local/sbin/..
/usr/local/share/..
/usr/local/src/..
```

### 3.9 mkdir

Yeni dizinler yaratmak için kullanılan mkdir komutuna hızlı bir göz atalım. Aşağıdaki örnek /tmp altında 3 yeni dizin yaratır, tic, tac ve toe.

```
$ cd /tmp
$ mkdir tic tac toe
```

Varsayılan olarak, mkdir komutu üst dizinleri sizin için yaratmaz; yaratılması istenilen dizinin tüm yolunda sondan bir önceki dizin'e kadar bütün dizinlerin var olması gereklidir. Bunun anlamı /de/ne/me dizinlerini yaratmak istiyorsanız, üç ayrı mkdir komutu çalıştırmalısınız:

```
$ mkdir de/ne/me
mkdir: 'de/ne/me' dizini oluşturulamıyor: Böyle bir dosya ya da dizin yok
$ mkdir de
$ mkdir de/ne
$ mkdir de/ne/me
```

### 3.10 mkdir -p

Ancak, mkdir'in faydalı opsiyonu -p ile mkdir'i dizin yolunda var olmayan üst dizinleri yaratmasını sağlayabilirsiniz, aşağıda görüldüğü gibi:

```
$ mkdir -p de/ne/me2
```

mkdir komutu hakkında daha detaylı bilgi almak için man mkdir yazarak mkdir'in yardım sayfasına başvurabilirsiniz. Bu yöntem burada bahsedilen hemen hemen her komut için geçerli olacaktır, (örnek olarak man ls), sadece cd komutu bash kabuğunun içinde geldiği için bunun dışında kalır.

### 3.11 touch

Şimdi kopyalama, yeniden adlandırma, dosya ve dizin taşımak için kullanılan cp ve mv komutlarına göz atacağız. Buna başlamadan önce /tmp de bir dosya oluşturmak için touch komutunu kullanacağız:

```
$ cd /tmp
$ touch benikopyala
```

touch komutu varolan bir dosyanın "mtime" değerini günceller (ls -l çıktısındaki 6. sütunu hatırlayın). touch komutuna parametre olarak verilen dosya mevcut değilse boş bir yeni dosya yaratılacaktır. Yukarıdaki komutun sonucunda sıfır boyutlu bir /tmp/benikopyala oluşturmuş oluyorsunuz.

### 3.12 echo ve yönlendirme

Şimdi artık dosya varolduğuna göre bu dosyaya biraz veri ekleyelim. Bunu echo komutu ile yapabiliriz, bu komut kendisine geçilen argümanları alıp bunu standart çıktıya basar. Önce echo komutunun kendisine bir göz atalım:

```
$ echo "ilk dosya"
ilk dosya
```

### 3.13 echo ve yönlendirme

Ve şimdi aynı echo komutunu bu sefer çıktı yönlendirme tekniği ile kullanalım:

```
$ echo "ilkdosya" > benikopyala
```

Yukarıda gördüğümüz büyüktür işareti kabuğa echo'nun çıktısını benikopyala isimli bir dosyaya yönlendirmesini söyler. Bu dosya eğer mevcut değilse yaratılır, yok eğer mevcut ise üzerine yazılır. ls -l komut ile benikopyala dosyasının 9 byte büyüklüğünde olduğunu görebiliriz çünkü "ilkdosya" sözcüklerini, boşluk karakterini ve yenisatır karakterini barındırır:

```
$ ls -l benikopyala
-rw-r--r-- 1 knoppix knoppix 9 2003-03-25 16:33 benikopyala
```

### 3.14 cat ve cp

Terminalde dosyanın içeriğini görüntülemek için cat komutunu kullanın:

```
$ cat benikopyala
ilkdosya
```

Şimdi cp komutunun en basit şeklini kullanarak benikopyala dosyasının benikopyaladi isimli bir kopyasını çıkarabiliriz:

```
$ cp benikopyala benikopyaladi
```

Detaylı olarak incelersek görürüz ki bu ikisi gerçekten de birbirinden ayrı dosyalardır, inode numaraları farklıdır:

```
$ ls -i benikopyala benikopyaladi
8919 benikopyaladi      8901 benikopyala
```

### 3.15 mv

Şimdi "benikopyaladi" dosyasını "benitasidi" olarak isimlendirmek için mv komutunu kullanalım. Bu sefer inode numarası aynı kalmakla birlikte bu inode numarasına sahip olan dosya ismi değişmiş olacaktır.

```
$ mv benikopyaladi benitasidi
$ ls -i benitasidi
8919 benitasidi
```

Taşınan (move) bir dosyanın inode numarası, söz konusu dosya kaynak dosya ile aynı dosya sisteminde bulunduğu sürece aynı kalır. Bu eğitimin 3. bölümünde dosya sistemlerini daha detaylı olarak inceleyeceğiz.

## 4 Linkler yaratmak ve dosyaları silmek

### 4.1 Katı bağlantılar (hard links)

Daha önce izin girişleri ve inode'lar arasındaki ilişkiden bahsederken bağlantı terimini kullanmıştık. Aslında Linux altında iki tip bağlantı çeşidi vardır. Daha önce tartıştığımız bağlantılar katı bağlantılardır. Sistem üzerindeki herhangi bir inode herhangi bir miktarda katı bağlantıya sahip olabilir, ve tüm katı bağlantılar ortadan kalkana kadar da sistem üzerinde var olmaya devam edecektir. Yeni katı bağlantılar ln komutu kullanılarak yaratılabilir:

```
$ cd /tmp
$ touch ilklink
$ ln ilklink ikincilink
$ ls -i ilklink ikincilink
8940 ilklink      8940 ikincilink
```

### 4.2 Katı bağlantılar, devamı

Görebildiğiniz gibi, katı bağlantılar bir dosyayı ilişkilendirmek için inode seviyesinde çalışmaktadır. Linux sistemlerde, katı bağlantıların bazı kısıtlamaları vardır. Bir kısıtlama, sadece dosyalara katı bağlantılar yapmaya izin verir, dizinlere vermez. "." ve ".." sistem tarafından yaratılmış dizinlere ilişkilendirilmiş katı bağlantılar olmasına rağmen bu doğrudur. "root" kullanıcısı bile olsanız, kendinize ait dizinlere ilişkilendirilmiş yeni katı bağlantılar yaratamazsınız. Katı bağlantıların ikinci kısıtlaması da dosya sistemleri arasında geçiş yapılamamasıdır. Bunun anlamı, / ve /usr dizinleri farklı dosya sistemlerinde bulunuyorsa /usr/bin/bash dizininin içinde /bin/bash dizininin içindeki bir dosyaya katı bağlantı yapamazsınız.

### 4.3 Sembolik bağlantılar

Pratikte, sembolik bağlantılar katı bağlantılardan çok daha sık kullanılırlar. Sembolik bağlantılar özel bir tip dosya olup başka bir dosyaya doğrudan inode üzerinden değil de ismi ile ilişkilendirilir. Sembolik bağlantılar bir dosyanın silinmesine engel olmazlar, hedef dosya ortadan kaybolursa, sembolik bağlantı kullanılamaz hale gelir ya da "kırılır".

#### 4.4 Sembolik bağlantılar, devamı

Bir sembolik bağlantı `ln` komutuna `-s` parametresi verilerek yaratılabilir.

```
$ ln -s ikincilink ucunculink
$ ls -l ilklink ikincilink ucunculink
-rw-r--r--  2 knoppix  knoppix  0 2003-03-25 16:39 ilklink
-rw-r--r--  2 knoppix  knoppix  0 2003-03-25 16:39 ikincilink
lrwxrwxrwx  1 knoppix  knoppix 10 2003-03-25 16:40 ucunculink -> ikincilink
```

`ls -l` çıktısında sembolik bağlantılar normal dosyalardan üç şekilde ayırdedilebilir. Birincisi, bu çıktıdaki ilk sütun dosyanın bir sembolik bağlantı olduğunu belirtmek için `l` karakterini farkedebilirsiniz. İkincisi, sembolik bağlantının boyutu ilişkilendiği hedef dosyasının karakter sayısını verir (yukarıdaki örnekte ikincilink). Üçüncüsü, son kolon hedef dosyanın ismini görüntüler.

#### 4.5 Sembolik Bağlantılar, Ayrıntılar

Sembolik bağlantılar genelde katı bağlantılardan daha esnektir. Dizinler de dahil olmak üzere herhangi bir çeşit dosya sistemi nesnesi için sembolik bağlantı yaratmanız mümkündür. Ayrıca, sembolik bağlantıların yürütülmesi yollara bağlı olduğu için, mükemmel bir şekilde başka bir dosya sistemi üzerinde yer alan nesneye de sembolik bağlantı oluşturabilirsiniz. Fakat bu durum da sembolik bağlantının anlaşılmasını biraz zorlaştırabilir.

#### 4.6 Sembolik Bağlantılar, Ayrıntılar

`/tmp` isimli dizin altında, `/usr/local/bin` isimli dizine sembolik bağlantı yaratmak istediğimiz bir durum düşününüz. Bu durumda yazmamız gereken:

```
$ ln -s /usr/local/bin bin1
$ ls -l bin1
lrwxrwxrwx  1 knoppix  knoppix 14 2003-03-25 16:42 bin1 -> /usr/local/bin
```

Ya da alternatif olarak:

```
$ ln -s ../usr/local/bin bin2
$ ls -l bin2
lrwxrwxrwx  1 knoppix  knoppix 16 2003-03-25 16:43 bin2 -> ../usr/local/bin
```

#### 4.7 Sembolik Bağlantılar, Ayrıntılar

Gördüğünüz gibi, her iki sembolik bağlantı da aynı dizini göstermektedir. Fakat, eğer ikinci sembolik bağlantı başka bir dizin içerisine taşınırsa, bu bağlantı bağlı yol yüzünden "kırılmış" (broken) olacaktır.

```
$ ls -l bin2
lrwxrwxrwx  1 knoppix  knoppix 16 2003-03-25 16:43 bin2 -> ../usr/local/bin
$ mkdir yenidir
$ mv bin2 yenidir
$ cd yenidir
$ cd bin2
bash: cd: bin2: Böyle bir dosya ya da dizin yok
```

`/tmp/usr/local/bin` dizini mevcut olmadığı için, artık `bin2` bağlantısı ile dizin değiştiremeyiz. Bir başka deyişle `bin2` bağlantısı artık kırılmıştır.

#### 4.8 Sembolik Bağlantılar, Ayrıntılar

Bu yüzden bazen bağlı yol bilgisi ile sembolik link yaratmaktan kaçınmak iyi bir fikir olabilir. Fakat sembolik linklerin kullanışlı olduğu bir çok durum da vardır. `/usr/bin` altında yer alan bir program için başka bir isim yaratmak istediğiniz bir örneği ele alalım:

```
$ ls -l /home/knoppix/.Xdefaults
-rw-r--r--  1 knoppix  knoppix 893 2000-02-28 03:26 /home/knoppix/.Xdefaults
```



## 4.9 Sembolik Bağlantılar, Ayrıntılar

root kullanıcısı olarak, ".Xdefaults" için "xd" adında başka bir isim yaratmak istiyor olabilirsiniz. Bu örnekte, komut satırlarında yer alan # işaretinden de açıkça anlaşıldığı gibi root haklarına sahibiz.

```
# cd /home/knoppix
# ln -s /home/knoppix/.Xdefaults xd
```

## 4.10 Sembolik Bağlantılar, Ayrıntılar

Bu çözüm çalışacak olsa da, her iki dosyayı da /usr/local/bin altına taşımaya karar verdiğimiz anda bu durum problem yaratacaktır:

```
$ mv /home/knoppix/.Xdefaults /home/knoppix/xd /usr/local/bin
```

Çünkü sembolik bağlantımız için tam yol kullanmış durumdayız, xd halen /home/knoppix/.Xdefaults dosyasını göstermektedir ki bu bağlantı artık mevcut değildir, yani kırılmıştır. Bağlı yol ve tam yolların her ikisinin de çok faydalı olduğu durumlar vardır. Sonuçta kendi uygulamanıza en uygun olan tipi seçmeniz gerekmektedir. Genellikle bağlı yol ya da tam yol düzgün bir şekilde çalışır. Bu durumda aşağıdaki örnek çalışacaktır:

```
$ cd /home/knoppix
$ ln -s .Xdefaults xd
$ ls -l xd
lrwxrwxrwx    1 knoppix  knoppix          10 2003-04-11 14:54 xd -> .Xdefaults
```

## 4.11 rm

Artık cp, mv ve ln komutlarının nasıl kullanıldığını biliyoruz, şimdi nesneleri dosya sistemi üzerinde kaldırmayı öğrenmenin zamanı gelmiştir. Normalde bu işlem rm komutuyla yapılmaktadır. Dosyayı kaldırmak için basitçe bunları komut satırında belirlemek gerekmektedir.

```
$ cd /tmp
$ touch dosya1 dosya2
$ ls -l dosya
$ ls -l dosya1 dosya2
-rw-r--r--    1 knoppix  knoppix          0 2003-03-25 16:53 dosya1
-rw-r--r--    1 knoppix  knoppix          0 2003-03-25 16:53 dosya2
$ rm dosya2
$ ls -l dosya1dosya2
ls: dosya2: Böyle bir dosya ya da dizin yok
-rw-r--r--    1 knoppix  knoppix          0 2003-03-25 16:53 dosya1
```

## 4.12 rmdir

Dizinleri kaldırmak için iki seçeneğiniz var. Dizinin altındaki bütün nesneleri yok edebilir ve sonra rmdir kullanarak dizinin kendisini kaldırabilirsiniz:

```
$ mkdir benimdir
$ touch benimdir/dosya1
$ rm benimdir/dosya1
$ rmdir benimdir
```

## 4.13 rm ve dizinler

rm komutunun özyineli seçeneklerini kullanıp rm'e belirttiğiniz dizini içerdiği nesnelerle birlikte kaldırmasını söyleyebilirsiniz:

```
$ rm -rf benimdir
```

Genel olarak bir dizin ağacını kaldırmak için tercih edilen rm -rf 'dir. rm -f kullanırken dikkatli olun, çünkü bunun gücü iyi veya kötü niyetli olarak kullanılabilir.

## 5 joker karakterler, Giriş

### 5.1 Joker karakterler (wildcards), Giriş

Günlük Linux kullanımında çok defa bir operasyonu tek seferde birden fazla nesne için çalıştırmak isteyebileceğiniz (rm gibi) durumlarla karşılaşabilirsiniz. Böyle durumlarda, aşağıdaki gibi bütün dosyaları komut satırına yazmak çok gereksizdir:

```
$ rm dosya1 dosya2 dosya3 dosya4 dosya5 dosya6 dosya7 dosya8
```

### 5.2 Joker karakterler, Giriş, devamı

Bu problemi çözmek için, Linux'un kendi içinde yer alan joker karakter desteğinin getirdiği avantajdan yararlanabilirsiniz. Aynı zamanda (tarihsel nedenlerden dolayı) "globbing" olarak da adlandırılan bu destek, joker karakter modelini kullanarak tek seferde birden fazla dosyayı belirlemenize olanak sağlar. Bash ve diğer Linux komutları joker karakter modeline bakarak ve dosya sistemi üzerinde bu modele uyan dosyaları bularak komutu yorumlar. Bu durumda diyelim ki bulunduğumuz dizinde dosya1, dosya2, ... , dosya8 isimli dosyalar varsa bu dosyaları aşağıdaki şekilde yok edebilirsiniz:

```
$ rm dosya[1-8]
```

Ya da daha basitçe ismi dosya ile başlayan bütün dosyaları yok etmek istediğinizde aşağıdaki gibi yazabilirsiniz:

```
$ rm dosya*
```

### 5.3 Uyuşmazlıkları (non-matches) Anlamak

Veya /etc içinde yer alan ve ismi g ile başlayan tüm nesneleri listelemek istediğinizde aşağıdaki gibi yazabilirsiniz:

```
$ ls -d /etc/g*
/etc/gateways      /etc/gnome-vfs-mime-magic  /etc/gshadow
/etc/gconf         /etc/gpm-root.conf        /etc/gtansrc
/etc/gdm           /etc/grep-dctrl.rc        /etc/gtk
/etc/ggi           /etc/groff                 /etc/gtk-2.0
/etc/gimp          /etc/group                 /etc/guid-resolv.conf
/etc/gnome         /etc/group.org
/etc/gnome-vfs-2.0 /etc/gs.Fontmap
```

Peki ya acaba herhangi dosya sistemi nesnesi ile uyuşmayan bir yol belirlendiğinde ne olacaktır? Aşağıdaki örnekte, /usr/bin altında yer alan, asdf ile başlayan ve jkl ile biten bütün dosyaları listelemek istiyoruz:

```
$ ls -d /usr/bin/asdf*jkl
ls: /usr/bin/asdf*jkl: Böyle bir dosya ya da dizin yok
```

### 5.4 Uyuşmazlıkları Anlamak, devamı

Bu durumu şöyle açıklayabiliriz. Normalde mevcut dosya sisteminde yer alan bir ya da daha fazla nesne ile uyuşan bir model belirlediğimizde kabuk, yazdığımız modeli her uyuşan nesne boşlukla ayrılacak şekilde listeler. Fakat model herhangi bir nesne ya da nesnelerle uyuşmuyorsa, kabuk ifadeyi olduğu gibi bırakır. Ve sonuçta ls, /usr/bin/asdf\*jkl dosyasını bulamaz ve hata verir. Buradaki kural şudur: Kullanılan model, sadece modelin belirlediği nesne ya da nesneler sistemdeki nesnelerle uyuşuyorsa açılmaktadır.

### 5.5 Joker karakter sözdizimi: \*

Artık joker karakterlerin nasıl çalıştığını anladığımıza göre, biraz da joker karakterlerin sözdizimine göz atalım. joker karakterler arasında birkaç özel karakter kullanabilirsiniz; işte bunlardan bir tanesi:

```
*
```

Yıldız (asterix) \* işareti sıfır veya daha fazla karakterle eşleşecektir. Bunun anlamı \* olan yere herhangi bir şey gelebileceğidir. Örnekler:

- /etc/g\* belirtimi /etc'nin altında g ile başlayan tüm nesnelerle eşleşir.
- /tmp/my\*1 belirtim /tmp'nin altında my ile başlayan ve 1 ile biten tüm nesnelerle eşleşir.

## 5.6 Joker karakter sözdizimi: ?

?

Soru işareti '?' herhangi bir tek karakterle eşleşir. Örnekler:

- dosyam? belirtimi dosyam ile başlayıp bir tane daha karakter içeren nesnelerle eşleşir.
- /tmp/notlar?txt belirtimi hem /tmp/notlar.txt, hem /tmp/notlar\_txt dosyasıyla eşleşir, eğer mevcutlarsa.

## 5.7 Joker karakter sözdizimi: []

[]

Bu joker karakterler '?' ile benzerdir, ancak daha fazla kesinlikte ayırım yapma şansını verir. Bu joker karakterleri kullanmak için, [] arasında eşlemek istediğiniz tüm karakterleri koymalısınız. Sonuçta elde edilen eşleşme bu karakterlerin herhangi birinin tek eşleşmesini verecektir. - (eksi) işaretini bir aralık vermek için kullanabilirsiniz, hatta birden fazla aralığı birlikte kullanabilirsiniz. Örnekler:

- dosyam[12] belirtimi dosyam1 ve dosyam2 ile eşleşecektir. Wildcard, çalışılan dizindeki en bu dosyalardan en az bir tanesi mevcut olacak şekilde genişletilecektir.
- [Cc]hange[Ll]og belirtimi Changelog, ChangeLog, changeLog, ve changelog ile eşleşecektir. Gördüğünüz gibi, köşeli parantez joker karakterlerini kullanmak küçük / büyük harf eşleşmelerindeki kombinasyonları yakalamak için faydalı olacaktır.
- ls /etc/[0-9]\* komutu /etc 'nin altında rakamla başlayan tüm dosyaları listeler.
- ls /tmp/[A-Za-z]\* komutu /tmp 'nin altında küçük ya da büyük harfle başlayan tüm dosyaları listeler.

## 5.8 Joker karakter sözdizimi: [!]

[!]

[!] yapısı [] yapısına benzemektedir. Ancak bu sefer [] parantezde yer alan karakterle eşleştirme yerine, parantezler içerisinde bulunmayan karakterler ile eşleştirme yapılmaktadır.

Örnekler:

```
rm dosyam[!9]
```

bu komut ismi dosyam9 olan dosya dışındaki tüm dosyaları yok eder.

## 5.9 Joker karakter ile ilgili uyarılar

Burada joker karakter kullanırken dikkat edilmesi gereken bazı noktalara değineceğiz. Bash wildcard ile ilgili karakterler (?, [, ], \*) yazıldığı zaman buna göre özel bir işlem yapacağından, bir komuta parametre geçirirken bu karakterler kullanılacaksa dikkat edilmelidir. Örneğin, [fo]\* ifadesini içeren bir dosya yaratmak istediğimizde aşağıdaki yazılı olan komut yapmak istediğimiz şeyi gerçekleştirmeyecektir:

```
$echo [fo]* > /tmp/yenidosyam.txt
```

Eğer [fo]\* modeli, dizin içerisinde herhangi bir dosya ile eşleşiyorsa, bu durumda /tmp/yenidosyam.txt dosyası içerisinde görmeyi beklediğiniz [fo]\* ifadesi yerine bu eşleşen dosyaları göreceksiniz. O halde çözüm nedir? Bunun için gerekli çözümlerden birisi karakterlerinizi tek tırnak içerisinde yazmanızdır. Bu yaklaşım, kabuğa bu karakterler üzerinde hiç bir joker karakter işlemi yapmaması gerektiğini anlatır.

```
$echo '[fo]*' > /tmp/yenidosyam.txt
```

## 5.10 Joker karakter ile ilgili uyarılar, devamı

Bu yaklaşımı kullandığınızda yeni dosyanız beklediğiniz gibi [fo]\* karakter dizimini içerecektir. Alternatif olarak, ters bölü (backslash) kullanarak karakterlerinizin bash için, joker karakter yerine normal karakter anlamına gelmesini sağlayabilirsiniz. (escape characters)

```
$ echo \[fo\]* > /tmp/yenidosyam.txt
```

Yukarıda anlatılan her iki yaklaşımda aynı şekilde çalışacaktır. Ters bölü (backslash) karakterinin bu şekilde kullanımından da bahsettiğimize göre, eğer karakter olarak backslash (\) kullanmak istiyorsanız bunu tek tırnak içinde ya da \\ şeklinde yazabileceğinizi söyleyebiliriz.

## 5.11 Tek Tırnak ile Çift Tırnağın Kıyaslanması

Çift tırnağın da tek tırnakla aynı şekilde çalışacağını bilmelisiniz, fakat şunu da belirtmek gerekir ki çift tırnak tek tırnağa benzer bir şekilde çalışsa da kabuğunu bazı sınırlı genişletme işlemlerine izin verir. Bu yüzden, yukarıda da gösterilen joker karakter olarak da anlamlı ifadeleri komut olarak kullanacaksanız en iyi seçim tek tırnak olacaktır. Eğer joker karakter genişletmesine ilişkin daha ayrıntılı bilgi edinmek istiyorsanız komut satırında man 7 glob yazınız. Tırnak kullanımına ilişkin daha fazla bilgi edinmek istiyorsanız komut satırında man 8 glob yazarak QUOTING başlıklı bölümü okuyunuz.

# 6 Düzenli İfadeler

## 6.1 Düzenli ifade nedir?

Düzenli ifadeler ("regex" veya "regexp" de denir) yazı kalıplarını ifade etmek için kullanılan özel bir biçimdir. Linux sistemlerinde, arama ve değiştirme işlerinde olduğu kadar yazı kalıplarının bulunmasında da düzenli ifadeler kullanılır.

## 6.2 Glob kıyaslaması

Düzenli ifadelere bakarken düzenli ifade sözdiziminin (sentaksının) önceki eğitimdeki (bu bölümün sonundaki Kaynaklar başlığının altındaki Bölüm 1'e bakın) dosya ismi açılımı (globbing) kurallarına benzediğini düşünebilirsiniz. Ancak sakın bunu sizi aldatmasına izin vermeyin; bu iki kavram ancak ve ancak çok yüzeysel olarak birbirlerini andırırlar. Düzenli ifadeler de dosya ismi açılım kuralları da her ne kadar başlangıçta benzer gibi görünseler de aslında çok farklı kurallara sahip ve farklı durumlarda kullanılan iki sistemdir.

## 6.3 Basit bir dizi parçası

Uyarımızı yaptıktan sonra düzenli ifadelerin en basit türüne bakalım, basit dizi parçası. Bunun için grep komutunu kullanacağız. Bu komut bir dosyanın içeriğini tarayıp belli bir düzenli ifade kalıbına uyan kısımları bulur. grep, düzenli ifade kalıbına uyan her satırı basar ve buna uymayan satırları da görmezden gelir:

```
$ grep bash /etc/passwd
root:x:0:0:root:/home/root:/bin/bash
knoppix:x:1000:1000:Knoppix User:/home/knoppix:/bin/bash
```

Yukarıda, grep programına ilk parametre olarak bir düzenli ifade geçildiğini ve ikinci parametre olarak bir dosya ismi verildiğini görebilirsiniz. Grep /etc/passwd dosyasındaki her satırı okur bash olarak verilmiş basit karakter dizi parçasının o satırda mevcut olup olmadığına, başka bir deyişle o satırın kendisine verilen düzenli ifade kalıbı ile eşleşip eşleşmediğine bakar. Eğer bir eşleşme söz konusu olursa ilgili satırı basar, olmazsa bir sonraki satıra geçer

## 6.4 Basit karakter dizisi parçasını anlamak

Genellikle eğer belli bir karakter dizisini arıyorsanız bunu herhangi bir "özel" karakter kullanmadan olduğu gibi yazabilirsiniz. Ancak ve ancak aradığınız karakter dizisinin içinde +, ., \*, [, ], ve karakterlerinden biri ya da birkaçı var ise o zaman bunları tırnak içine almanız ve bunlardan önce tersbölü karakteri basmanız gerekir. Birkaç basit örneğe daha bakalım:

- /tmp (/tmp karakter dizisini arar)
- "[kutu]" ([kutu] karakter dizisini arar)
- "komik" (\*komik\* karakter dizisini arar)
- "ldso" (ld.so karakter dizisini arar)

## 6.5 Metakarakterler

Düzenli ifadeler ile çalışırken yukarıdaki örneklerde incelediğimiz aramalardan çok daha karmaşık aramaları metakarakterleri kullanarak gerçekleştirebilirsiniz. Bu metakarakterlerden biri . (nokta) karakteridir ve bu "herhangi bir karakter" anlamına gelir:

```
$ grep dev.hda /etc/fstab
/dev/hda1 swap swap defaults 0 0
/dev/hda2 /mnt/hda2 auto noauto,user,exec 0 0
```

Bu örnekte gördüğünüz gibi programın /etc/fstab dosyasına bakarak ürettiği çıktıda dev.hda gibi bir karakter dizisi mevcut değildir. Ancak grep programının aradığı dev.hda karakter dizisi değil dev.hda kalıbı idi. Hatırlarsanız . herhangi bir karakter anlamına geliyordu. Gördüğünüz gibi . metakarakterini işlevsel olarak komut satırındaki dosya ismi açılımı mekanizmasındaki ? metakarakterine benzetebiliriz.

## 6.6 [] Kullanımı

Eğer . sembolünü kullanarak yaptığımızdan daha özel bir eşleşme isteseydik [ ve ] sembollerini (köşeli parantez karakterleri) kullanarak belli bir karakter kümesini gösterebilirdik:

```
$ grep dev.hda[10] /etc/fstab
/dev/hda1 swap swap defaults 0 0
```

Gördüğünüz gibi bu sözdizim kuralı da tıpkı dosya ismi açılımlarındaki [] özelliğine benzetilebilir. İşte düzenli ifadelerle ilgili dikkat etmeniz gereken noktalardan biri daha: Sözdizimi olarak benzer görünen ama birebir aynı olmayan bazı kurallar düzenli ifadeleri öğrenmeyi güçleştirebilir.

## 6.7 [^] Kullanımı

Köşeli parantezlerin anlamını [ karakterinden hemen sonra bir karakteri koyarak tersine çevirebilirsiniz. Bu durumda köşeli parantezlerin içindeki karakterler değil de onların dışında geriye kalan diğer karakterlerden herhangi biri yakalanır. Bu arada lütfen dikkat edin: düzenli ifadelerde [] kullanırken dosya açılım kurallarında [!] kullanıyorduk:

```
$ grep dev.hda[^10] /etc/fstab
/dev/hda2 /mnt/hda2 auto noauto,user,exec 0 0
```

## 6.8 Farklılaşan sözdizimi

Köşeli parantezler içindeki sözdiziminin farklı olduğuna dikkatleri çekmekte fayda var. Mesela köşeli parantez içinde bir . karakteri yazarsanız bu gerçekten de nokta karakterini yakalamak istediğinizi gösterir tıpkı yukarıdaki örneklerde 1 ve 2 karakterlerinin eşleşmesinde olduğu gibi. Köşeli parantezlerin dışında ise . nokta karakteri herhangi bir karakterin yerine geçer ve başına karakteri gelmediği sürece özel olarak nokta karakterinin yakalanmasını zorunlu kılmaz. Bundan faydalanarak /etc/fstab dosyasında dev.hda içeren tüm satırları şu komut ile bulabiliriz:

```
$ grep dev.[.]hda /etc/fstab
```

Alternatif olarak şöyle de yazabilirdik:

```
$ grep "dev\\.hda" /etc/fstab
```

Her iki düzenli ifade de /etc/fstab dosyasındaki herhangi bir satır ile eşleşmeyecektir.

## 6.9 ”\*” metakarakteri

Bazı metakarakterler kendi başlarına herhangi bir şey ile eşleşmezler ancak bunun yerine kendilerinden önce gelen karakterin anlamını değiştirirler. Bunlardan biri de \* (yıldız) karakteridir ve kendinden önce gelen karakterin sıfır ya da daha çok tekrar eden şeklini yakalar. Birkaç örneğe bakacak olursak:

- `ab*c` (`abbbbc`'yi yakalar ama `abqc`'yi yakalamaz)
- `ab*c` (`abc`'yi yakalar ama `abbqbbc`'yi yakalamaz)
- `ab*c` (`ac`'yi yakalar ama `cba`'yi yakalamaz)
- `b[cq]*e` (`bqe`'yi yakalar ama `eb`'yi yakalamaz)
- `b[cq]*e` (`bccqqe`'yi yakalar ama `bccc`'yi yakalamaz)
- `b[cq]*e` (`bqqcce`'yi yakalar ama `cqe`'yi yakalamaz)
- `b[cq]*e` (`bbbeee`'yi yakalar)
- `.*` (herhangi bir karakter dizisini yakalar)
- `foo.*` (`foo` ile başlayan herhangi bir karakter dizisini yakalar)

`ac` satırı `ab*c` regex ile eşleşir çünkü \* metakarakteri kendisinden önce gelen ifadenin sıfır defa tekrar etmesine (yani hiç bulunmamasına) izin verir. Şunu da eklemek gerekir ki \* metakarakteri temel olarak \* glob karakterinden daha farklı bir şekilde yorumlanmaktadır.

## 6.10 Satır Başı ve Satır Sonu

Burada ayrıntılı olarak inceleyeceğimiz ve sırasıyla satır başı ve satır sonunu ifade eden son metakarakterler `^` ve `$` metakarakterleridir. Düzenli ifadenizin en başında `^` metakarakterini kullanarak modelinizin satır başında bulunmasını zorlayabilirsiniz. Aşağıdaki örnekte `#` karakteri ile başlayan herhangi bir satırla eşleştirme yapmak için `^` düzenli ifadesini kullanıyoruz:

```
$ grep ^# /etc/apt/sources.list
# See sources.list(5) for more information, especially
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
```

## 6.11 Tam Satır Düzenli İfadeler

`^` ve `$` metakarakterleri tüm satır ile eşleştirme yapmak için birlikte kullanılabilir. Örneğin, aşağıdaki örnekte yer alan düzenli ifade içerisinde herhangi bir sayıda diğer karakterlerin yer aldığı, `#` ile başlayan ve `.` karakteri ile sonlanan bir satırla eşleştirmeyi sağlayacaktır:

```
$ grep '^#.*\.$' /etc/fstab
# CDROMs are managed through the apt-cdrom tool.
```

Yukarıdaki örnekte, `$` karakterinin kabuk tarafından yorumlanmasını engellemek için tüm ifadeyi tek tırnak karakterleri arasında yazdık. Tek tırnak olmasaydı, daha grep yorumlamadan `$` karakteri düzenli ifade içerisinde yok olacaktı.

# 7 FHS ve dosyaları bulmak

## 7.1 Dosya Sistemi Hiyerarşi Standardı

Dosya Sistemi Hiyerarşi Standardı linux sistemi üzerindeki bir dizinin planlamasını belirleyen bir belgedir. D.S.H.S'ı, dağıtım-bağımsız yazılım geliştirmeyi daha basit hale getirmek için ortak bir planlama sağlamak amacıyla ortaya atılmıştır. D.S.H.S aşağıdaki dizinleri belirlemektedir. (Direk olarak D.S.H.S spesifikasyonundan alınmıştır):

- `/` (kök -root- dizini)

- /boot (açılış yükleyiciye -boot loader- ait statik dosyalar)
- /dev (aygıt dosyaları)
- /etc (üzerinde çalışılan -host- makinaya ait sistem konfigürasyonu)
- /lib (paylaşılan temel kütüphaneler ve çekirdek modülleri)
- /mnt (bir dosya sistemini geçici olarak bağlamak için bağlantı noktası)
- /opt (Sonradan eklenebilen -add-on- uygulama yazılım paketleri)
- /sbin (gerekli ikili/çalışabilir -binary- sistem dosyaları)
- /tmp (geçici dosyalar)
- /usr (ikincil hiyerarşi)
- /var (değişken bilgiler)

## 7.2 İki Bağımsız D.S.H.S Kategorisi:

D.S.H.S planlama belirlemesi iki bağımsız dosya kategorisi olduğu fikrini temel alır; paylaşılabılır - paylaşılamaz, ve değişken - statik. Paylaşılabılır bilgi ana makinalar (hosts) tarafından paylaşılabılır ancak paylaşılmayan bilgi verilen ana makinaya özgüdür (konfigürasyon dosyaları gibi). Değişken bilgi düzenlenebilir ve değiştirilebilirken statik bilgi değiştirilemez (fakat bu kural sistem kurulumu ve bakımı için geçerli değildir.)

Aşağıdaki tablo, bu kategoriler içine girebilecek dizin örnekleri ile bu 4 olası kombinasyonu özetlemektedir. Yine bu tablo da D.S.H.S'den alınmıştır.

	shareable	unshareable
static	/usr	/etc
	/opt	/boot
variable	/var/mail	/var/run
	/var/spool/news	/var/lock

## 7.3 /usr dizinindeki ikincil hiyerarşi

/usr altında, root dosya sistemine çok benzeyen ikincil bir hiyerarşi göreceksiniz. Makina açıkken /usr dizinin olması gerek şart olmadığından bu dizin bir ağ üzerinde paylaşılabılır ("paylaşılabılır") ya da bir CD-ROM içinden bağlanabilir ("statik"). Bir çok Linux kurulumunda /usr dizinin paylaşımı kullanılmıyorsa da root daki birincil hiyerarşi ile /usr deki ikincil hiyerarşi arasındaki farkı gözetmenin faydasını algılamak çok önemlidir.

Burada D.S.H.S'si ile ilgili söylenecekler bu kadardır. Dökümanın kendisi oldukça kolay anlaşılır olduğundan daha ayrıntılı bilgi için kullanabilirsiniz. Eğer dökümanı okursanız Linux D.S.H.S'si hakkında daha bir çok şey öğreneceğinizin kesin olduğunu söyleyebiliriz.

## 7.4 Dosyaları bulmak

Linux sistemleri içlerinde yüzlerce dosya barındırmaktadır. Her ne kadar bu dosyaların yerlerini kaybetmeyecek kadar usta olsanız bile, büyük bir ihtimalle bir tanesini bulmak için arasına yardma ihtiyaç duyacaksınız. Linux'te dosyaları bulmak için birkaç tane farklı araç yer almaktadır. Aşağıdaki konu başlıkları bu araçları göstermekte ve işiniz için gerekli olan doğru aracı bulmanıza yardımcı olmaktadır.

## 7.5 PATH

Bir programı komut satırında çalıştırdığınızda, kabuk aslında bir grup dizin arasında yazdığınız komutu aramaktadır. Örneğin, `ls` yazdığınızda, kabuk gerçekte bu komutun `/usr/bin` altında yer aldığını bilemez. Onun yerine dizinlerin tutulduğu ve birbirlerinden iki nokta ile ayrıldığı `PATH` çevre değişkenine başvurur. `PATH`'in içinde yazılı olan değeri görmek için aşağıdaki gibi yazabiliriz:

```
$ echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:
/usr/local/sbin:/usr/local/bin:/usr/games:.
```

`PATH` değişkeninin sahip olduğu bu değere göre (sizdeki değişkenin değeri farklı olabilir, kabuk yazılan `ls` komutu için öncelikle `/usr/local/bin` daha sonra `/usr/bin` altında arama yapacaktır. Büyük bir ihtimalle `ls` `/usr/bin` altında yer almaktadır, bu yüzden kabuk burada komut arama işlemini bitirecektir.

## 7.6 PATH'i düzenlemek

Elemanlar atayarak kendi `PATH` değişkeninizi değiştirebilirsiniz:

```
$ PATH=$PATH:~/bin
$ echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:
/usr/local/sbin:/usr/local/bin:/usr/games:./home/knoppix/bin
```

Ayrıca, her ne kadar mevcut olan `PATH` değişkenini kullanmak kadar kolay olmasa da `PATH` değişkeni içerisinde bazı elemanları silmeniz de mümkündür. Bunun için en iyi yol, olmasını istediğiniz yeni `PATH` değerinin yazmaktır:

```
$ PATH=/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:~/bin
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/knoppix/bin
```

## 7.7 "which" hakkında her şey

Aradığınız programın `PATH` ile verilen dizinlerden birinde olup olmadığını `which` komutu ile bulabilirsiniz. Örneğin, aşağıdaki komut ile Linux sistemimizin sağduyuya sahip olup olmadığını sorgulayabiliriz:

```
$ which sagduyu
```

Aşağıdaki örnekte ise `ls` komutunun nerede bulunduğunu görebiliriz:

```
$ which ls
/usr/bin/ls
"which -a"
```

Son olarak `-a` seçeneğine dikkat etmelisiniz. Bu seçenek kullanıldığında `which` komutu `PATH` değişkeninde tanımlı tüm dizinlerde aradığınız programın olup olmadığını gösterir:

```
$ which -a ls
/bin/ls
whereis
```

Eğer bir programın yerini öğrenmek dışında onunla ilgili daha çok bilgiye erişmek istiyorsanız `whereis` komutunu denemelisiniz:

```
$ whereis ls
ls: /bin /bin/ls /usr/share/man/man1/ls.1.gz
```

Buradan görürüz ki `ls` programı iki dizinde bulunmaktadır, `/bin` and `/usr/bin`. Buna ek olarak `/usr/share/man` dizininde de `ls` ile ilgili bir man sayfası olduğu bilgisini ediniriz. `man ls` yazdığınızda karşınıza gelen sayfa yukarıda belirtilen sayfadır. `whereis` aynı zamanda kaynak arama, alternatif arama yolları tanımlama ve sıradışı girdileri arama gibi özelliklere de sahiptir. Detaylı bilgi için lütfen `whereis` man sayfasını inceleyin.



## 7.8 find

find alet çantanızdaki araçlardan biridir. find komutu ile artık sadece program dosyaları ile sınırlı değilsiniz, her türlü dosyayı çok çeşitli kriterlere göre arayabilirsiniz. Örneğin /usr/share/doc dizini ve alt dizinlerinde README dosyasını bulmak için şu komutu vermeniz yeterlidir:

```
$ find /usr/share/doc -name README
/usr/share/doc/aalib1/README
/usr/share/doc/adduser/examples/README
/usr/share/doc/afio/README
```

## 7.9 find ve joker karakterler

-name parametresine değer geçerken dosya açılımındaki (glob) joker karakterleri kullanabilirsiniz ancak bu durumda onları çift tırnak içine almanız veya önlerine ters bölü (backslash) koymanız gerekir (ki bash tarafından açılmayıp find komutuna oldukları gibi gönderilebilsinler). Örneğin uzantıya sahip README dosyalarını bulmak için şu komutu verebiliriz:

```
$ find /usr/share/doc -name README\*
/usr/share/doc/LANG/fr/xte1/README_IMINITEL.txt.gz
/usr/share/doc/a2ps/README.gz
/usr/share/doc/aalib1/README
```

## 7.10 find ile küçük büyük harf ayrımı yapmadan aramak

Elbette arama yaparken büyük küçük harf ayrımı yapmak istemeyebilirsiniz:

```
$ find /usr/share/doc -name '[Rr][Ee][Aa][Dd][Mm][Ee]*'
```

Veya daha basit olarak:

```
$ find /usr/share/doc -iname readme\*
```

Gördüğünüz gibi -iname seçeneği büyük küçük harf ayrımını ortadan kaldırır.

## 7.11 find ve düzenli ifadeler

Eğer düzenli ifadelere alışkınsanız -regex seçeneğini kullanarak belli bir kalıpla eşleşecek şekilde dosya isimlerini arayabilirsiniz. -iname seçeneğine benzer şekilde -iregex seçeneği ile de büyük küçük harf ayrımı yapılmasını engelleyebilirsiniz.

## 7.12 find ve türler

-type seçeneği ile belli tipte dosyalar üzerinden arama yapabilirsiniz. Bu seçeneğe geçebileceğiniz argümanlar şunlardır: b (blok cihazı), c (karakter cihazı), d (dizin), p (isimlendirilmiş boru), f (normal dosya), l (sembolik bağlantı) ve s (soket). Örneğin /usr/bin dizinindeki vim sözcüğünü içeren sembolik linkleri aramak için:

```
$ find /usr/bin -name '*vim*' -type l
/usr/bin/evim
/usr/bin/gvim
/usr/bin/gvimdiff
/usr/bin/rgvim
/usr/bin/rvim
/usr/bin/vimdiff
```

## 7.13 find ve 'mtime'lar

-mtime seçeneği dosyaları son güncelleme zamanlarına göre aramanızı sağlar. mtime seçeneğine verilen değer 24-saatlik zaman periyodu formatındadır ve genellikle + (verilen zamandan sonra) veya - (verilen zamandan önce) ile kullanılır. Örneğin şu senaryoya bir bakalım:

```
$ ls -l ?
-rw----- 1 root root 0 Jan 7 18:00 a
-rw----- 1 root root 0 Jan 6 18:00 b
-rw----- 1 root root 0 Jan 5 18:00 c
-rw----- 1 root root 0 Jan 4 18:00 d
$ date
Mon Jan 7 18:14:52 EST 2002
```

Son 24 saat içinde oluşturulmuş dosyaları arayabilirsiniz:

```
$ find . -name \? -mtime -1
./a
```

Veya içinde bulunduğunuz 24 saatten önce oluşturulmuş dosyaları arayabilirsiniz:

```
$ find . -name \? -mtime +0
./b
./c
./d
```

### 7.14 -daystart seçeneği

Eğer ek olarak -daystart seçeneğini de kullanırsanız zaman periyodları 24 saat öncesinden değil günün başlama anından başlar. Örneğin dün ve evvelki gün oluşturulmuş dosyaları listelemek için:

```
$ find . -name \? -daystart -mtime +0 -mtime -3
./b
./c
$ ls -l b c
-rw----- 1 root root 0 Jan 6 18:00 b
-rw----- 1 root root 0 Jan 5 18:00 c
```

### 7.15 -size seçeneği

-size seçeneği dosyaları büyüklük kriterine göre aramanızı sağlar. Aksi belirtilmediği sürece -size seçeneğine 512-byte'lık bloklar cinsinden büyüklük verildiği kabul edilir ancak bunun sonuna bir takı eklemek işleri kolaylaştırabilir ve daha doğal kılabilir. Kullanabileceğiniz takılar şunlardır: b (512-byte'lık blok), c (byte), k (kilobyte), ve w (2-byte'lık sözcük). Bunlara ek olarak + öneki ile verilen büyüklükten daha büyük dosyaları ve - öneki ile de daha küçük dosyaları arayabilirsiniz. Örneğin /usr/bin dizini içinde boyu 50 byte'tan küçük olan dosyaları bulmak isterseniz:

```
$ find /usr/bin -type f -size -50c
/usr/bin/bison.yacc
/usr/bin/kdvi
/usr/bin/krdp
/usr/bin/pydoc2.1
/usr/bin/pydoc2.2
/usr/bin/rgrep
/usr/bin/txi2pdf
```

### 7.16 Bulunan dosyaların işlenmesi

Pekçok değişik kriterlere göre arayıp bulduğunuz onca dosya ile ne yapacağınızı merak ediyor olmalısınız! find komutu bulduğu dosyalar üzerinde işlem yapma kabiliyetine de sahiptir ve bunun için -exec seçeneğinden faydalanır. Bu seçenek argüman olarak ; ile sonlandırılan ve çalıştırılabilir bir komut alır ve her {} görüldüğü yere de bulmuş olduğu dosya ismini koyarak ilgili komutu çalıştırır. Bunun anlamının en kolay yolu şu örneğe bakmak:

```
$ find /usr/bin -type f -size -50c -exec ls -l '{}' ';'
-rwxr-xr-x 1 root root 29 2002-04-04 12:29 /usr/bin/bison.yacc
-rwxr-xr-x 1 root root 30 2001-12-01 07:14 /usr/bin/kdvi
-rwxr-xr-x 1 root root 27 2000-09-24 14:46 /usr/bin/krdp
-rwxr-xr-x 1 root root 45 2002-07-29 22:39 /usr/bin/pydoc2.1
```

```
-rwxr-xr-x  1 root    root          48 2002-07-29 23:19 /usr/bin/pydoc2.2
-rwxr-xr-x  1 root    root          31 2001-11-28 17:36 /usr/bin/rgrep
-rwxr-xr-x  1 root    root          36 2002-09-07 10:08 /usr/bin/texi2pdf
```

Gördüğünüz gibi find komutu bir hayli güçlü bir komuttur. UNIX ve Linux'un yıllarca süren geliştirilmesi boyunca bu komut da geliştirilmiştir. find komutunun daha pek çok faydalı seçeneği vardır. İlgili man sayfasında bunları bulabilirsiniz.

## 7.17 locate

Şimdiye dek which, whereis ve find komutlarını ele aldık. find çalışırken biraz vakit geçtiğini farketmişsinizdir çünkü üzerinde her dizine tek tek girip bakmak zorundadır. locate komutu ise bir veritabanını kullanarak işleri biraz hızlandırabilir. Bu komut yol isminin herhangi bir kısmı ile eşleşme yapabilir sadece dosya ismine bakmak zorunda değildir. Örneğin:

```
$ locate bin/ls
/bin/ls
/bin/lspci
/sbin/lsmmod
/sbin/lspci
/sbin/lspnp
/usr/bin/lsattr
/usr/bin/lsdev
/usr/bin/lsof
/usr/bin/lspgpot
/usr/bin/lss16toppm
/usr/sbin/lsof
/usr/sbin/lssusb
```

## 7.18 updatedb kullanımı

Pekçok Linux sistemi periyodik olarak veritabanını güncelleyen bir süreç (process) çalıştırır. Eğer sisteminiz yukarıdaki komuta aşağıdaki gibi bir hata ile cevap verirse o zaman bir veritabanı oluşturmak için updatedb komutunu çalıştırmanız gerekir:

```
$ locate bin/ls
locate: /var/spool/locate/locatedb: No such file or directory
$ su
Password:
# updatedb
```

updatedb komutunun işini tamamlaması uzun bir süre alabilir. Eğer gürültücü bir harddiskiniz varsa tüm dosya sistemi indekslenirken başınız epey ağrıyabilir.

## 7.19 slocate

Pek çok Linux dağıtımında locate komutu yerine artık slocate komutu kullanılmaktadır. Genellikle locate isimli bir sembolik bağlantı olduğu için hangisini kullanayım diye tereddüt etmenize gerek yoktur. slocate "secure locate" yani "güvenli locate" anlamına gelmektedir. Bu komut kullandığı veritabanında dosya ve dizin izinlerini de depolar ve böylece normal kullanıcılar erişim iznine sahip olmadıkları dizine bakamazlar. slocate komutunun kullanım şekli locate komutunda olduğu gibidir ancak çıktı bilgisi komutu çalıştıran kullanıcıya göre değişebilir.

# 8 Süreç (process) Kontrolü

## 8.1 xeyes'ı başlatmak

Süreç kontrolünü öğrenmek için önce bir süreç başlatalım:

```
$ xeyes -center red
```

xeyes penceresinin birden açıldığını ve kırmızı gözlerin fareyi takip ettiğini göreceksiniz. Bu arada bir şey daha farkedeceksiniz: Komut satırına dönmediğinizi.

## 8.2 Süreci durdurmak

Komut satırına geri dönebilmek için Control-C tuş kombinasyonuna basmalısınız (genellikle Ctrl-C veya  $\hat{C}$  şeklinde gösterilir):

```
 $\hat{C}$ 
$
```

bash komut satırına geri döndünüz ancak xeyes penceresi de kayboldu. Aslında işi yapan süreç tamamen sonlandırılmış, öldürülmüş durumda. Control-C ile süreci öldürmek yerine Control-Z ile durdurabiliriz:

```
$ xeyes -center red
 $\hat{Z}$ 
[1]+  Stopped                  xeyes -center red
$
```

Şimdi gene bash komut satırına döndünüz ama bu sefer xeyes penceresi durması gereken yerde duruyor. Ancak fare ile oynarsanız göreceksiniz ki gözler sizi takip etmiyor artık, donup kalmış durumdadır. Eğer xeyes penceresi üzerine başka bir pencereyi sürükleyip sonra çekerseniz göreceksiniz gözler yeniden çizilmiyor bile. Süreç hiçbir şey yapmıyor şu anda yani "Durmuş" (Stopped) halde.

## 8.3 fg ve bg

Süreci "durdurulmuş olmaktan çıkarmak" ve yeniden çalışır hale getirmek için onu bash komutlarından biri olan fg komutu ön plana çekebiliriz:

```
$ fg
xeyes -center red
 $\hat{Z}$ 
[1]+  Stopped                  xeyes -center red
$
```

Şimdi de yine bir bash komutu olan bg ile süreci arkaplanda çalıştıralım:

```
$ bg
[1]+  xeyes -center red &
$
```

Harika! xeyes süreci şu anda arkaplanda çalışıyor ve bizim de karşımıza kullanabileceğimiz bir komut satırı geldi.

## 8.4 "&" Kullanımı

Eğer xeyes'ı doğrudan arkaplanda çalıştırmak isteseydik (Control-Z ve sonra da bg kullanmak yerine), xeyes komutunun sonuna "&" (ampersand) eklememiz yeterli olacaktı:

```
xeyes -center blue &
[2] 4386
```

## 8.5 Birden çok sayıda arkaplan süreci

Şimdi arkaplanda bir kırmızı ve bir de mavi xeyes sürecimiz var. Bunları bash komutlarından biri olan jobs komutu ile listeleyebiliriz:

```
$ jobs -l

[1]-  4205 Running              xeyes -center red &
[2]+  4386 Running              xeyes -center blue &
```

Sol sütundaki sayılar bash kabuğunun bu süreçleri başlatırken onlara atadığı süreç numaralarıdır. İki numaralı süreç (başka bir deyişle job) yanında bir + (artı) sembolü vardır ve bu da sürecin şu anda çalışmaktan olan süreç olduğunu gösterir yani fg komutunu verirsiniz önplanda çalışmaya başlayacak olan süreç budur. Tabi eğer isterseniz numarasının vererek başka bir süreci de önplana çekebilirsiniz. fg 1 komutu kırmızı xeyes sürecini önplana çekecektir. Bir sonraki sütun ise süreç ID'sini yani pid numarasını gösterir karşımızdaki listede (bunun gelmesini de jobs komutuna verdiğimiz -l seçeneğine borçluyuz). Son olarak her iki süreç (job) da "Running" yani çalışıyor durumda ve bunları başlatan komut satırlarını da en sağda görebilirsiniz.

## 8.6 Sinyallere giriş

Süreçleri öldürmek, durdurmak ya da devam ettirmek için, Linux işletim sistemi "sinyal" olarak bilinen özel bir iletişim şekli kullanır. Bir sürece belli bir sinyal yollayarak o süreci sonlandırabilir, durdurabilir, ya da başka şeyler yapabilirsiniz. Aslında Control-C, Control-Z, ya da fg, bg gibi komutları kullandığımızda yaptığınız tam da budur – bash kabuğunun belli bir sürece belli bir sinyal göndermesini sağlamaktasınızdır. Bu sinyaller kill komutuna süreç numarası (pid) verilerek de gönderilebilir:

```
$ kill -s SIGSTOP 4386
$ jobs -l
[1]- 4205 Running          xeyes -center red &
[2]+ 4386 Durduruldu (sinyal) xeyes -center blue
```

Gördüğünüz gibi kill komutu isminin çağrıştırdığı şekilde illa ki bir süreci "öldürmek" zorunda değildir. "-s" seçeneği kullanılarak kill komutunun bir sürece herhangi bir sinyali göndermesi sağlanabilir. Linux süreçlerine SIGINT, SIGSTOP ya da SIGCONT sinyallerinden biri gönderildiğinde sırası ile bunları sonlandırır, durdurur ya da devam ettirir. Bir sürece yollayabileceğiniz başka sinyal türleri de vardır; bunlardan bazıları uygulamanın özelliklerine göre yorumlanacaktır. Belli bir sürecin hangi sinyale karşı nasıl tepki vereceğini öğrenmenin en güzel yolu ilgili süreci başlatan komutun man sayfasına bakıp oradaki SIGNALS bölümünü okumaktır.

## 8.7 SIGTERM ve SIGINT

Eğer bir süreci öldürmek (sonlandırmak) isterseniz pek çok seçeneğiniz vardır. Aksi belirtilmediği sürece kill komutu SIGTERM sinyalini gönderir ve bu Control-C ile yollanabilen SIGINT sinyalinden farklı olmakla birlikte benzer sonuç üretir:

```
$ kill 4205
$ jobs -l
[1]- 4205 Sonlandırıldı      xeyes -center red &
[2]+ 4386 Durduruldu (sinyal) xeyes -center blue
```

## 8.8 Büyük ölüm

Süreçler hem SIGTERM hem de SIGINT sinyallerini dikkate almayıp görmezden gelebilirler; bunu ya bilerek ve isteyerek yaparlar ya da bir şekilde durmuş veya takılmışlardır. Bu gibi durumlarda bir balyoz kullanmak gerekebilir ve devreye SIGKILL sinyali girer. Bir süreç SIGKILL sinyalini görmezden gelemmez:

```
$ kill 4386
$ jobs -l
[2]+ 4386 Durduruldu (sinyal) xeyes -center blue
$ kill -s SIGKILL 4205
$ jobs -l
[2]+ 4205 Süreç durduruldu      xeyes -center blue
```

## 8.9 nohup

Belli bir işi (süreci) başlattığımız terminal o işin kontrol terminali olarak adlandırılır. Bazı kabuklar öntanımlı olarak (bash değil) siz logout komutu ile terminalden çıkınca arkaplandaki işlere SIGHUP sinyalini yollayıp onların sonlanmasına yol açarlar. Eğer siz terminalden logout komutu ile çıktığımızda süreçlerinizin bu şekilde sonlandırılmasını istemiyorsanız o zaman süreci başlatırken nohup seçeneğini kullanmanızda fayda vardır:

```
$ nohup make &
$ exit
```

## 8.10 ps komutu ile süreçleri listelemek

Yukarıda kullandığımız jobs komutu içinde bulunduğumuz mevcut bash oturumunda başlatılan süreçleri listeliyordu. Sisteminizdeki tüm süreçleri görmek için kullanmanız gereken komut ise ps komutudur ve bu komuta a ve x seçeneklerini vermeniz gereklidir.

```
$ ps ax
  PID TTY          STAT TIME  COMMAND
    1 ?            S      0:04  init
    2 ?            SW      0:00  [keventd]
    3 ?            SWN     0:00  [ksoftirqd_CPU0]
    4 ?            SW      0:01  [kswapd]
    5 ?            SW      0:00  [bdflood]
    6 ?            SW      0:00  [kupdated]
```

Bu komutun çıktısı olan listenin ancak küçük bir kısmını yukarı yazdık çünkü genellikle bu liste çok daha uzun olur. Bu liste size makinanın o anda ne yaptığının bir fotoğrafını verir ancak göz atılacak pek çok veri vardır içinde. Eğer ax seçeneklerini kullanmadan ps komutunu çalıştıracak olsaydınız sadece size ait ve belli bir terminalin kontrolü altındaki süreçleri görürdünüz. ps x size tüm süreçleri gösterir, belli bir terminal tarafından kontrol edilmeyenler dahil. Eğer ps a şeklinde kullanmış olsaydınız o zaman da belli terminallere bağlı ve herkese ait süreçleri görürdünüz.

## 8.11 Hem ormanı hem de ağaçları görebilmek

Her süreç ile ilgili farklı bilgileri de listeleyebilirsiniz. -forest (orman) seçeneği süreç hiyerarşisini görmeyi ve sisteminizdeki birçok sürecin arasındaki ilişkiyi algılamayı sağlar. Bir süreç başka bir süreç başlattığında bu yeni sürece "çocuk" (child) süreç denir. -forest seçeneği kullanılarak elde edilen bir ps çıktısında ebeveyn süreçler yani çocuk süreçleri başlatan süreçler solda görünürken çocuk süreçler de sağa doğru dallanacak şekilde listelenirler.

```
$ ps x --forest
  PID TTY          STAT TIME  COMMAND
  318 ?            S      0:00  /bin/sh /etc/X11/xinit/xinitrc
  444 ?            S      0:00  \_ kwrapper ksmserver --restore
2932 ?            S      0:00  kdeinit: kio_uiserver
  495 ?            S      0:00  /usr/bin/kdesud
  493 ?            S      0:00  kdeinit: kcookiejar
  462 ?            S      0:10  kdeinit: kicker
```

## 8.12 "u" ve "l" ps seçenekleri

"u" ve "l" seçenekleri a ve x seçeneklerine eklenebilen seçeneklerdir ve süreçler hakkında daha çok bilgi almanızı sağlarlar:

```
$ ps au
  USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         237  0.0  0.4   3196  1168 tty1      S   10:21   0:00 /bin/bash -login
root         238  0.0  0.4   3196  1168 tty2      S   10:21   0:00 /bin/bash -login
root         239  0.0  0.4   3196  1168 tty3      S   10:21   0:00 /bin/bash -login
root         240  0.0  0.4   3196  1168 tty4      S   10:21   0:00 /bin/bash -login
root        3606  0.0  0.5   3128  1440 tty1      S   11:10   0:00 /bin/bash
knoppix     5054  0.5  0.8   3128  2096 tty0      S   11:30   0:00 /bin/bash
knoppix     5195  0.0  0.5   3508  1532 tty0      R   11:32   0:00 ps au
```

## 8.13 "top" kullanımı

Eğer kendinizi sürekli ps komutunu çalıştırır ve neler olup bittiğini anlamaya çalışır durumda bulursanız ihtiyacınız olan esas komut top komutudur. top komutu sürekli güncellenen bir süreç listesi basmanın yanı sıra özet bilgi vermeyi de ihmal etmez:

```
$ top
11:34:16 up 1:13, 0 users, load average: 0.43, 0.21, 0.12
47 processes: 45 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 3.0% user, 1.8% system, 0.0% nice, 95.2% idle
Mem: 255680K total, 249136K used, 6544K free, 10624K buffers
Swap: 393552K total, 43852K used, 349700K free, 37780K cached
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	COMMAND
309	root	14	0	204M	132M	6644	S	2.5	52.9	2:23	XFree86
5050	knoppix	11	0	11316	10M	9356	R	0.7	4.2	0:01	kdeinit
3849	knoppix	10	0	38532	36M	13612	S	0.5	14.7	0:45	kword
5282	knoppix	13	0	1016	1016	808	R	0.5	0.3	0:00	top
1	root	8	0	52	48	28	S	0.0	0.0	0:04	init
2	root	9	0	0	0	0	SW	0.0	0.0	0:00	keventd
3	root	19	19	0	0	0	SWN	0.0	0.0	0:00	ksoftirqd_CPU0
4	root	9	0	0	0	0	SW	0.0	0.0	0:01	kswapd
5	root	9	0	0	0	0	SW	0.0	0.0	0:00	bdflood

## 8.14 nice

Her sürecin bir öncelik derecesi vardır ve Linux buna göre sürecin çalışma hızını ayarlar. Bir sürecin önceliğini bu sürecin isminin başına nice yazıp başlatmak suretiyle ile ayarlayabilirsiniz:

```
$ nice -10 xmms /cdrom/Demos/Audio/opensource.ogg
```

Öncelik ayarına "nice" dendiği için buna yüksek bir değer verdiğiniz takdirde diğer süreçlere neza-  
ket sergilediğinizi, onlara CPU için öncelik tanıdığınızı hatırlamak zor olmayacaktır. Aksi belirtilmediği  
sürece, süreçler 0 öncelikli olarak başlarlar, yani yukarıda ogg'nin 10 ile başlatılması demek, ogg'nin  
CPU'ya ne kadar ihtiyacı olursa olsun önceliği diğer süreçlere tanınması ve böylece onların normal hızda  
çalıştırılmasına izin vermesi demektir. Süreçlerin "nice" derecesini ps ve top komutlarının yukarıdaki  
çıktılarındaki NI sütununda görebilirsiniz.

## 8.15 renice

Nice komutu, sadece çalıştırdığınız anda, bir komutun önceliğini belirler. Eğer çalışmakta olan bir komu-  
tun önceliğini değiştirmek isterseniz renice kullanmanız gerekir.

```
$ ps l 3849
F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND
000 1000 3849 456 10 0 59556 44208 select S ? 1:02 kword

$ renice 10 3849
3849: eski öncelik 0, yeni öncelik 10

$ ps l 3849
F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND
000 1000 3849 456 17 10 59556 44208 select SN ? 1:03 kword
```

## 9 Yazı işleme

Yönlendirme Bir komutun çıktısını bir dosyaya yönlendirmek için > operatörünü kullanırız, şöyle ki:

```
$ echo "dosya" > kopyasi
```

Çıktıyı yönlendirmeye ek olarak, oldukça güçlü bir kabuk özelliği olan pipe (boru)ları da kullanabiliriz.  
Bağlantıları kullanarak bir komutun çıktısını diğerine girdi olarak vermemiz mümkün. Şu örneğe bir bakın

```
$ echo "merhaba dunyali" | wc
1      2      16
```

— karakteri soldaki komutun çıktısının sağdaki komuta girdi olarak verilmesini sağlamaktadır. Yukar-  
daki örnekte echo komutu "merhaba dunyali" cumlesini sonunda bir satır sonu ile birlikte basar. Normalde  
bu çıktının terminalde görünmesi lazım fakat yaptığımız bağlantı onu wc komutuna yönlendirdi ve böylece  
satır, kelime ve karakter (boşluk ve satır sonu dahil) sayılarını öğrendik. Bir boru örneği İşte bir başka  
örnek:

```
$ ls -s | sort -n
```

Bu durumda, `ls -s` normalde mevcut dizinin listesini her dosyanın boyu dosya isminden önce gelecek şekilde terminal ekranına basacakken biz bu çıktıyı borudan geçirip `sort -n`'ye yönlendirdiğimiz için karşımıza çıktının sayısal olarak sıralanmış şekli gelmektedir. Ev (home) dizininizdeki büyük dosyaları bir bakışta tespit etmek için gerçekten güzel bir yöntem!

Aşağıdaki örnekler biraz daha karışıktır ancak boru mekanizması ile ne kadar güçlü ve becerikli işlemler gerçekleştirilebildiğini göstermeleri bakımından faydalıdır. Şimdiye dek görmediğiniz bazı komutlarla karşılaşabilirsiniz, bunun sizi yavaşlatmasına lütfen izin vermeyin. Bunun yerine boru mekanizmasını yani bir çıktının bir sonraki programa girdi olarak nasıl geçildiğini anlamaya odaklanın. Böylece bunlardan günlük Linux kullanımınızda bolca faydalanabilirsiniz. Sıkıştırılmışları açan boru hattı Normalde sıkıştırılmış bir dosyayı açmak ve arşivi çözmek için şunu yaparsınız:

```
$ bzip2 -d linux-2.4.16.tar.bz2
$ tar xvf linux-2.4.16.tar
```

Bu yöntemin dezavantajı diskinizde sıkıştırılmamış bir ara dosya barındırmasıdır. `tar` yazılımı doğrudan standart girdiden okuyabildiği için (yani buna parametre olarak illa ki diskteki bir dosyayı geçmeniz gerekmediği için) yukarıdaki sonucun aynısına şöyle bir boru hattı kullanarak, daha pratik ve ekonomik bir şekilde erişebiliriz:

```
$ bzip2 -dc linux-2.4.16.tar.bz2 | tar xvf -
```

Vay canına! Sıkıştırılmış arşiv (tarball) dosyamız bir anda açılıverdi ve arada ortaya çıkan geçici bir dosyaya da ihtiyaç duymadık.

## 9.1 Daha uzun bir boru hattı

Bir başka örneğe bakalım:

```
$ cat dosyam.txt | sort | uniq | wc -l
```

`cat` komut ile `myfile.txt` dosyasının içeriğini elde edip bunu `sort` komutuna girdi olarak veriyoruz. `sort` komutu bu girdiyi alınca tüm satırları alfabetik olarak sıralıyor ve ürettiği çıktıyı boru üzerinden `uniq` komutuna aktarıyor ve bu program da tekrar eden satırları iptal ediyor (yani birbirinin aynısı olan satırlardan sadece bir tane görünüyor) ve bu sonucu `wc -l` komutuna veriyor. `wc` komutunu daha önce görmüştük, `-l` seçeneği ile birlikte kullanıldığında bu komut sadece ve sadece satır sayısını verir. Birkaç test dosyası oluşturup bu boru hattının etkilerini inceleyebilirsiniz.

## 9.2 Metin işleme kasırgası başlıyor!

Şimdi standart Linux metin işleme komutlarının üzerinden fırtına gibi geçeceğiz. Bu bölümde bir hayli yoğun bir içerik aktardığımız için her komuta dair detaylı bir sürü örnek veremeyeceğiz bu yüzden de sizden burayı okuduktan sonra ilgili komutların man sayfalarına bakmanızı istiyoruz. Bunları inceleyin ve değişik seçeneklerle oynayın, öğrenmenin en güzel yolu budur. Genel kural olarak bu komutlar kendisine verilen metin dosyalarının içeriğini uygun şekilde işleyip ekrana basarlar, dosyanın kendisine kalıcı olarak müdahale etmezler. Bu fırtına turundan sonra girdi ve çıktı yönlendirmeye biraz daha detaylı bakacağız. Evet, tünelin ucundaki ışığı göreceksiniz :)

## 9.3 echo

`echo` kendisine verilen argümanları terminale basar. `-e` seçeneğini kullanırsanız "escape dizisi" olarak tabir ettiğimiz özel karakter kombinasyonlarının da düzgün olarak yorumlanmasını sağlarsınız, misal: `echo -e "foo`

`nfoo"` komutu ekrana önce `foo` basar, bir satır aşağı iner başka bir deyişle yenisatır karakteri basar ve sonra yine bir kez `foo` basar. `-n` seçeneğini kullanırsanız çıktıya en sonra otomatik olarak eklenen yenisatır karakterinin basılmasını engelleyebilirsiniz.



## 9.4 cat, sort ve uniq

cat: cat komutu bir dosyanın içeriğini terminal ekranına basar. Bir boru hattına ilk girdiyi vermek için faydalı bir komuttur, misal `cat foo.txt — vesaire — vesaire`.

sort: sort komutu komut satırında kendisine belirtilen dosyanın içeriğini alfabetik olarak sıralayıp basar. Elbette sort komutu da boru üzerinden kendisine aktarılan girdiyi kabul edecek şekilde tasarlanmıştır. man sort ile bu komutun ne kadar farklı sıralama işleri için pratik şekilde kullanılabileceğini görebilirsiniz.

uniq: uniq sıralanmış bir dosyayı ya da boru hattı üzerinden kendisine yollanan veriyi alır ve bunun içindeki birbirinin aynısı olan satırları ayıklar.

## 9.5 wc, head, ve tail

wc: wc komut satırında kendisine belirtilen dosyayı (ya da boru hattından gelen veriyi) okur ve bu dosyanın satır, sözcük ve byte sayısını ekrana basar. man wc ile detaylı bilgi edinebilirsiniz.

head: head komutu bir dosyanın ya da boru hattından gelen verinin ilk on satırını basar. -n seçeneği ile kaç satır basılacağını belirleyebilirsiniz.

tail: Bu komut bir dosyanın ya da boru hattından akan verinin son on satırını basar. -n seçeneği ile kaç satır basılacağını belirleyebilirsiniz.

## 9.6 tac, expand ve unexpand

tac: tac da tıpkı cat komutu gibidir, şu farkla ki okuduğunu tersten basar. Yani son okunan satır ilk olarak basılır, dosyanın ilk satırı ise en son basılır.

expand: expand komutu girdideki tab karakterlerini boşluk karakterine dönüştürür. -t seçeneği ile tab duraklarını belirleyebilirsiniz.

unexpand: unexpand girdideki boşlukları tab karakterine dönüştürür. -t seçeneği ile tab duraklarını belirleyebilirsiniz.

## 9.7 cut, nl ve pr

cut: cut komutu belli bir dosyadaki ya da girdideki karakterle ayrılmış alanları çekip çıkarır.

nl: nl girdideki her satırın başına sıra ile artan bir sayı ekler. Özellikle program kaynak kodu türünden dosyalara göz atarken ya da bunların çıktısını alırken faydalıdır.

pr: pr komutu bir dosyayı sayfalara böler, yazıcı çıktılarında faydalıdır.

## 9.8 tr, sed ve awk

tr: tr bir tür karakter dönüştürme aracıdır, girdideki belli karakterleri çıktındaki başka karakterlere tasvir etmek için kullanılır.

sed: sed çok güçlü, veri akışını işlemeye yönelik (stream-oriented) metin editörüdür.

awk: awk kolay kullanımlı, satır tabanlı bir metin işleme dilidir.

## 9.9 od, split, ve fmt

od: od komutu girdiyi alıp sekizlik ya da onaltılık biçimde basar.

split: split komutu büyük bir dosyayı alıp bunu küçük dosyalara ayırır.

fmt: fmt paragraf kenarları toplanacak şekilde paragrafı yeniden biçimlendirir. Bu yetenek birçok text editör içerisinde yer alsa bile bilinmesi gereken komutlardan bir tanesidir.

## 9.10 Paste, join, ve tee

paste: paste iki ya da daha fazla dosyayı girdi olarak alır, giriş dosyalarından gelen ardışıl satırları birbiri ardına ekler ve sonuç satırlarını çıktı yapar. Text tabloları ve kolonları yaratmak için oldukça kullanışlıdır.

join: join de paste komutuna benzese de tek bir satırda birleşecek şeyleri eşleştirmek için, her girdi satırında bir alan kullanır.

tee: tee kendi girdisini hem dosyaya hem de ekrana basar. Bir şeyin log'unu tutmak ama aynı zamanda çıktıyı ekranda da görmek istediğinizde bu komut gayet kullanışlı olabilir

## 9.11 Yönlendirme (redirection) Kasırgası

Bash komut satırlarında kullanılan >'a benzer olarak, bir dosyayı komuta yönlendirmek için < de kullanabilirsiniz. Bir çok komuta girdi olması için komut satırlarında dosya adı belirleyebilirsiniz. Fakat bazı komutlar sadece standart girdiden alınan girişlerle çalışmaktadır.

Bash ve diğer kabuklar "herefile" kavramına destek verirler. Bu yöntem sizin belli bir gözcü değer ile sonlanmış ve komut çağrımından sonra gelen satırlarda bir komuta girdi belirtmenize olanak sağlar. Örneğin:

```
$ sort <<END
> elma
> armut
> muz
> END
armut
elma
muz
```

Bu örnekte, END ile sonlanan ve girdinin bittiğini ifade eden kelimeyi ve bundan önce de elma, armut ve muz kelimelerini yazdık ve sort programı bu girdinin sıralanmış şekli ile geri döndü.

## 9.12 >> Kullanımı

>> kullanımının << ile benzerlik göstereceğinin düşünecek olursanız da aslında bu ikisi birbirinden farklıdır. >> basitçe çıktıyı > nin yaptığı gibi dosyaya yazmak yerine dosyanın sonuna yazar. Örneğin:

```
$ echo merhaba > dosyam
$ echo oradakiler. > dosyam
$ cat dosyam
oradakiler.
```

O da ne! "merhaba" kelimesini kaybettik. Aslında az önce kastettiğimiz:

```
$ echo merhaba > dosyam
$ echo oradakiler. >> dosyam
$ cat dosyam
merhaba
oradakiler.
```

idi. Bu daha iyi değil mi?

# 10 Sistem ve ağ dökümantasyonu

## 10.1 Linux sistem döküman tipleri

Temel olarak bir Linux sisteminde üç çeşit döküman kaynağı vardır: kılavuz sayfaları (manual pages), bilgi sayfaları (info pages) ve /usr/share/doc içinde bulunan uygulamalar ile gelen dökümanlar. Bu bölümde, dışarıdan yardım almadan önce bu üç kaynağı nasıl inceleyeceğimizi ve onlardan nasıl yararlanabileceğimizi göreceğiz.

## 10.2 Kılavuz sayfaları

Kılavuz dosyaları, ya da "man sayfaları" UNIX ve Linux referans dökümanlarının temellerini oluşturur. İdeali herhangi bir komut, konfigürasyon dosyası veya kütüphane dosyası hakkında man sayfasına bakarak bilgi edinebilmenizdir. Pratikte ise Linux özgür bir yazılımdır ve bazı man sayfaları daha yazılmamış veya çok önceden yazılmış olup güncelliğini yitirmiş olabilir. Yine de man dosyaları yardıma ihtiyacınız olduğunda ilk bakılması gereken adreslerdir.

Bir man sayfasına ulaşmak için basitçe man ve istediğiniz başlığı yazın. Ekrana çıkan dökümandan çıkmak için q'ya basmanız gerekir. Mesela ls komutu hakkında bilgi almak için:

```
$man ls
```

yazmanız gerekir.

### 10.3 Kılavuz sayfaları, devamı

Bir man sayfasının planını bilmek ihtiyacınız olan bilgiye hızlıca ulaşmanıza yardımcı olabilir. Bir man sayfasında genellikle aşağıdaki bölümleri bulacaksınız:

NAME	Komutun ismi ve bir satırlık açıklaması
SYNOPSIS	Komutun nasıl kullanılacağı
DESCRIPTION	Komutun fonksiyonallitesi hakkında derinlemesine açıklama
EXAMPLES	Komutun nasıl kullanılacağına yönelik öneriler
SEE ALSO	İlgili başlıklar (genellikle man sayfaları)

### 10.4 Kılavuz sayfa bölümleri

Kılavuz sayfalarını içeren dosyalar, /usr/share/man (bazı eski sistemlerde ise /usr/man) dizininde bulunur. Bu dizindeki kılavuz sayfalarının yapılandırılması aşağıdaki gibidir.

man1	Kullanıcı Programları
man2	Sistem Programları
man3	Kütüphane fonksiyonları
man4	Özel dosyalar
man5	Dosya biçimleri
man6	Oyunlar
man7	Çeşitli, diğer

### 10.5 Çoklu kılavuz sayfaları

Bazı başlıklar birden fazla bölümde bulunur. Bunu göstermek için, bir başlık için varolan tüm kılavuz sayfalarını gösteren `whatis` komutunu kullanalım:

```
$ whatis printf
printf (1)          - format and print data
printf (3)          - formatted output conversion
```

Bu durumda, `man printf` doğrudan bölüm 1 de bulunan sayfaya gidecektir ("User Programs"). Eğer bir C programı yazıyor olsaydık bölüm 3'deki sayfa daha fazla ilgimizi çekerdi ("Library functions"). Özel bir bölümdeki bir kılavuz sayfasını komut satırında belirterek çağırabiliriz. `printf(3)` için şunu yazabiliriz:

```
$ man 3 printf
```

### 10.6 Doğru kılavuz sayfayı bulmak

Kimi zaman verilen başlık için doğru kılavuz sayfayı bulmak zordur. Bu durumda `man -k` kullanarak kılavuz sayfaların NAME bölümünde arama yapabilirsiniz. Bunun bir altdizi araması olduğuna dikkat edin. Dolayısıyla `man -k ls` benzeri bir komut, çok fazla çıktı verecektir. Bir örnekle bunu görelim:

```
$ man -k manual
Gnome (1) [gnomine] - manual page for Gnome gnomine 1.4.0.4
apropos (1)         - search the manual page names and descriptions
c2man (1)           - generate manual pages from C source code
```

### 10.7 apropos ile ilgili herşey

Bu örnek aslında birden fazla şey anlatıyor! Öncelikle, `apropos` komutu `man -k` ile tam olarak aynı işi yapar. (Bir sır olarak, `man -k` komutu verdiğinizde arka tarafta `apropos` komutunun çalıştığını söyleyelim) İkinci önemli nokta ise `makewhatis` komutu. Bu komut Linux sistemindeki tüm kılavuz sayfaları tarar, `whatis` ve `apropos` komutlarının veritabanını oluşturur. Genellikle, bu iş, root tarafından periyodik olarak tekrarlanır ve veritabanlarının güncel tutulması sağlanır:

```
# makewhatis
```

"man" komutu ile ilgili daha fazla bilgi alabilmek için kılavuz sayfasına bakabilirsiniz:

```
$ man man
```

## 10.8 Kılavuz Yolu (MANPATH)

man programı, başlangıç olarak ilgili sayfaları /usr/share/man, /usr/local/man, /usr/X11R6/man, ve bir ihtimal /opt/man dizinlerinde arar. Bazı durumlarda, bu bakılacak yollara yeni bir tane dahe eklemek gerekebilir. Bunun için, /etc/man.conf dosyasını, bir yazı editörüyle açıp, aşağıdaki gibi bir satır eklemek yeterlidir:

```
MANPATH /opt/man
```

Bu ekleme ile, /opt/man/man dizini altındaki kılavuz sayfalarda bulunacaktır. Ama bu yeni kılavuz sayfalarını whatis komutunun veritabanına eklemek için makewhatis komutunu yeniden çalıştırmak gerektiğini unutmayın.

## 10.9 GNU bilgisi

Kılavuz sayfalarının bir eksiği, yardımcı metinleri (hypertext) desteklememeleri. Bu durumda, bir dosyadan diğerine kolaylıkla atlamak mümkün olmuyor. Bunu farkedenden GNU uzmanları, yeni bir dokümantasyon yöntemi geliştirdiler: "info" (bilgi) sayfaları. Birçok GNU programı artık info sayfaları biçiminde kapsamlı bir dokümantasyonla geliyor. info sayfalarını okumaya "info" komutuyla başlayabilirsiniz:

```
$ info
```

info komutunun bu şekilde kullanılması, sistemde ulaşılabilen sayfaların bir indeksini getirir. Bu çıktının içinde ok tuşlarıyla hareket edebilir, yıldız simgesi ile işaretlenmiş olan bağlantıları enter tuşuyla izleyebilir, ya da q tuşu ile çıkabilirsiniz. Buradaki tuşlar Emacs'deki gibidir, dolayısıyla Emacs editörüne alışkınsanız, işlemlerinizi kolayca yapabilir, yönünüzü rahatlıkla belirleyebilirsiniz. Komut satırından özel bir info sayfası da çağırabilirsiniz:

```
$ info diff
```

info sayfaları hakkında daha fazla bilgi elde etmek isterseniz, info sayfasını okumanızı öneririm. Şu ana kadar bahsettiğimiz temel kullanım yöntemleriyle, bu işi basitçe yapabilirsiniz:

```
$ info info
/usr/share/doc
```

Linux sisteminizde son bir yardım kaynağı daha var. Çoğu programlar, farklı formatlardaki ek belgelerle dağıtılıyor. Bu formatlardan bazıları, düzyazı (text), PDF, PostScript, HTML vb. dir. Sisteminizdeki /usr/share/doc dizinine (bazı eski sistemlerde /usr/doc) girdiğinizde, hepsi belli bir uygulamayla gelmiş, uzunca bir izin listesi göreceksiniz. Bu dokümanları incelemek bazı cevherleri ortaya çıkarmanızı da sağlayabilir. Man sayfalarında veya info sayfalarında bulamayacağınız detaylara, ancak bazı eğitimlerde veya ilave teknik dokümanlarda rastlayabilirsiniz. Hızlı bir aramayla, okunabilecek ne kadar çok malzeme olduğunu görebilirsiniz.

```
$ cd /usr/share/doc
$ find . -type f | wc -l
```

## 10.10 Linux Dokümantasyon Projesi

Sistemde bulunan dokümanlara ek olarak, internet üzerinde de Linux konusunda çok iyi kaynaklara ulaşabilirsiniz. Linux Dokümantasyon Projesi'nde, bir grup gönüllünün, bütünleşmiş ve özgür bir Linux dokümantasyon seti oluşturmak için çalışmaya karar vermesiyle başlamış. Bu projenin varoluş amacı, Linux dokümantasyonunun parçalarını, kullanımı ve arama yapılması kolay, ortak bir platformda birleştirmektir. Linux Dokümantasyonu Projesine <http://www.linuxdoc.org/> adresinden ulaşabilirsiniz.

## 10.11 LDP'ye genel bakış

LDP aşağıdaki başlıkları içeriyor:

- Rehberler - geniş kapsamlı kitaplar, örneğin The Linux Programmer's Guide
- HOWTO kaynakları - konuya özel, derinlemesine kaynaklar, örneğin DSL HOWTO

- SSS (Sıkça Sorulan Sorular) - en sık sorulan sorular ve cevapları, örneğin Brief Linux FAQ
- Kılavuz sayfalar - komutlara özel yardım sayfaları (Linux sisteminizde man komutuyla ulaştığınız dosyaların aynılarıdır).

Hangi bölümde araştırma yapmanız gerektiğini bilmiyorsanız, arama kutularını kullanabilirsiniz. Böylece konu başlığına göre arama yaparak bilgiye ulaşırsınız. LDP bunlara ek olarak Linux Gazette ve LinuxFocus gibi bazı bağlantı ve kaynak listesi de sunuyor. Ayrıca ilgili mail listelerine ve haber arşivlerine de LDP üzerinden ulaşabilirsiniz.

## 10.12 Mail Listeleri

Linux geliştiricileri için en önemli ortak çalışma ortamı mail listeleridir. Çoğu zaman projeler, birbirinden çok uzakta yaşayan, neredeyse dünyanın iki ayrı ucundaki kişilerin ortak çalışmasıyla oluşuyor. Mail listeleri, bu geliştiricilere, proje üzerinde çalışan diğer kişilerle iletişim kurma, tartışma ve bilgi alışverişinde bulunma olanağı sağlıyor. Bu mail listelerinden en bilineni çekirdek geliştiricilerinin "Linux Kernel Mailing List" adlı listesidir. Adresi: <http://www.tux.org/lkml/>.

Geliştirme ortamına sağladığı desteğin yanı sıra, mail listeleri, normal kullanıcılar için de, soru sorma ve uzman kişilerden veya diğer kullanıcılardan cevap alma aracıdır. Örneğin, farklı Linux dağıtımları, yeni üyelerine destek amacıyla mail listeleri oluşturmuşlardır. Kullandığınız dağıtımın mail listelerine ulaşmak için, dağıtıcınızın web sayfasını kullanabilirsiniz. Yukarıda verdiğimiz adresten, Linux çekirdek geliştiricilerinin listesinde LKML FAQ (Linux çekirdek geliştiricileri mail listesi, Sıkça Sorulan Sorular) kısmını incerseniz, liste üyelerinin sık sık aynı sorunun sorulmasını pek hoş karşılamadıklarını göreceksiniz. Bu yüzden, sorunuzu listeye göndermeden önce liste arşivlerini incelemeniz daha doğru bir yöntem olacaktır. Dahası, bu yöntem size vakit kazandıracaktır.

## 10.13 Haber grupları

İnternet haber grupları, mail listelerinin bir benzeridir ama e-mail yerine NNTP denen (Network News Transfer Protocol) Ağ Haberleri Aktarma İletişim Kuralı temeline dayanır. Haber gruplarına katılmak için slrn veya pan gibi bir NNTP istemcisi kullanmak gerekiyor. Haber gruplarının ilk akla gelen avantajı, sadece istediğiniz zaman tartışmalara katılıyor olmanız. Böylece sizin dışınızda gelişen tartışmalar elektronik posta kutunuzu doldurmuyor.

Genele hitap eden listeler comp.os.linux ile başlar. Listeye LDP'nin sitesinden, <http://www.linuxdoc.org/linux/#ng> adresiyle ulaşabilirsiniz.

Mail listeleri gibi, haber grupları da arşivlenir. Popüler bir haber grubu arşivi sitesi olarak Deja News örnek gösterilebilir.

## 10.14 Satıcı firmalar ve üçüncü-parti web siteleri

Farklı Linux dağıtımlarının web sitelerinden, güncellenmiş dokümanlar, kurulum talimatları, donanım uyumluluk/uyumsuzluk açıklamaları ve daha birçok konuda destek alınabilir. Örnek siteler:

- Redhat Linux
- Debian Linux
- Gentoo Linux
- SuSE Linux
- Caldera
- Turbolinux

## 10.15 Linux danışmanları

Bazı Linux danışmanları, hem ücretsiz hem de ücretli olarak Linux dokümantasyonu sağlıyorlar. Bu danışmanlardan bazılarını aşağıda görebilirsiniz:

- LinuxCare
- Mission Critical Linux

## 10.16 Yazılım ve donanım sağlayıcıları

Son yıllarda birçok yazılım ve donanım firması, ürünlerine Linux desteğini eklediler. Bu firmaların sitelerinden, hangi donanımların Linux'u desteklediğini, yazılım geliştirme araçlarını, açılmış kaynak kodlarını, bazı donanımların Linux uyumu için hazırlanmış sürücülerini ve daha birçok bilgiyi alabilirsiniz. Bu hareketin örnekleri şöyle sıralanabilir:

- IBM ve Linux
- Compaq ve Linux
- SGI ve Linux
- HP ve Linux
- Sun ve Linux
- Oracle ve Linux

## 10.17 Geliştiricilerin kaynakları

Tüm bunlara ek olarak, bazı yazılım ve donanım firmaları, Linux geliştiricileri ve sistem yöneticileri için çok iyi kaynaklar oluşturmuş durumdadır. Bunların içinde en başarılı olanlardan birisi, yazılım/donanım firması IBM'in developerWorks Linux ortamıdır.

# 11 Linux izin modeli

## 11.1 Bir kullanıcı, bir grup

Bu bölümde, Linux'un izin ve sahiplik modelini inceleyeceğiz. Daha önce gördüğümüz üzere, her dosyanın sahibi olan bir kullanıcı ve bir grup vardır. Linux'un izin modellerinin özünde bu vardır. Dosyaların sahibi olan kullanıcı ve grupları, `ls -l` komutuyla listeleyebilirsiniz:

```
$ ls -l /bin/bash
-rwxr-xr-x  1 root    root      579816 2002-09-12 00:51 /bin/bash
```

Bu örnekte, `/bin/bash` çalıştırılabilir dosyasının sahibinin `root` kullanıcısı ve `root` grubu olduğunu görüyoruz. Linux izin modeli, her bir dosya için belirlenen üç ayrı izin katmanından oluşur. Bu üç katman, dosya sahibinin izinleri, grubun izinleri ve tüm diğer kullanıcıların izinleridir.

## 11.2 "ls -l" incelemesi

`ls -l` komutumuzun çıktısını inceleyelim.

```
$ ls -l /bin/bash
-rwxr-xr-x  1 root    root      579816 2002-09-12 00:51 /bin/bash
```

İlk `-rwxr-xr-x` bölümü, dosya izinlerinin sembolik gösterimidir. Baştaki tire (-) karakteri dosyanın tipini belirtir. Bu örnekteki dosya normal bir dosyadır. Diğer dosya çeşitleri ve onları ifade eden karakterler şöyledir:

'd' dizin 'l' sembolik bağlantı 'c' character special device 'b' block special device 'p' fifo 's' socket Üç adet üçlü grup

```
$ ls -l /bin/bash
-rwxr-xr-x  1 root    root      579816 2002-09-12 00:51 /bin/bash
```

Dosya tipini belirten karakteri takip eden üç adet üçlü harf grubu görüyoruz. İlk üçlü grup dosyanın sahibinin haklarını, ikinci grup dosyanın sahibi olan grubun haklarını, son üçlü grupta diğer tüm kullanıcılar için geçerli olan hakları simgeler.

"rwx" "r-x" "r-x"

Burada, `r` harfi, dosyanın içeriğinin okunmasına izin verildiği anlamına gelir. `w` harfi dosyaya yazı yazılabileceğini (bu aynı zamanda değişiklik yapma ve silme işlemlerini de kapsar), `x` harfi de dosyanın çalıştırılabileceğini gösterir. Tüm bunları bir araya getirdiğimizde görüyoruz ki, örneğimiz olan dosyayı herkes okuyabilir ve çalıştırabilir ama sadece sahibi (`root`) dosyanın değiştirilmesi konusunda hak sahibidir. Dolayısıyla, tüm kullanıcılar bu dosyayı kopyalama hakkına sahipken, sadece `root`'un güncelleme ya da silme hakkı vardır.

### 11.3 Ben kimim?

Bir dosyanın sahibi olan kullanıcıyı ve grubu değiştirmeyi görmeden önce, halihazırda çalışan kullanıcının kim olduğunu öğrenme ve grup üyelikleri hakkında bilgi alma yöntemini inceleyelim. Eğer bilgisayarı açtıktan sonra su komutunu hiç çalıştırmadıysanız, bilgisayarı açarken girdiğiniz kullanıcı olarak devam ediyorsunuz demektir. Eğer sık sık su komutunu çalıştırıyorsanız, o anda hangi kullanıcıyla çalıştığınızı hatırlamayabilirsiniz. Bu durumda whoami komutunu kullanabilirsiniz:

```
# whoami
root
# su knoppix
$ whoami
knoppix
```

### 11.4 Hangi grupların üyesiyim?

Hangi gruplara üye olduğunuz öğrenmek için groups yazmanız yeter:

```
$ groups
knoppix dialout fax voice cdrom floppy
tape sudo audio dip video games users usb
```

Bu örnekteki kullanıcı, knoppix, dialout, fax, voice, cdrom, floppy, tape, sudo, audio, dip, video, games, users, usb gruplarına üyedir. Diğer kullanıcıların grup bilgilerini de öğrenebilirsiniz. Bunun için, groups komutuna argüman olarak kullanıcı adlarını vermelisiniz:

```
$ groups root daemon
root : root
daemon : daemon
```

### 11.5 Kullanıcı ve Grup Haklarını Değiştirmek

Bir dosyanın ya da başka bir dosya sistemi nesnesinin sahip olduğu kullanıcı ya da grubu değiştirmek için sırasıyla chown ve chgrp komutlarını kullanabilirsiniz. Bu komutlarda herbiri bir isim ve bunu takip eden bir ya da daha fazla dosya ismi ile kullanılır.

```
# chown root /etc/passwd
# chgrp users /etc/passwd
```

Ayrıca isterseniz chown komutunun alternatif bir kullanımı ile tek seferde dosya sahibini ve sahip olduğu grubu değiştirebilirsiniz.

```
# chown root.users /etc/passwd
```

superuser olmadığınız sürece chown komutunu kullanmamalısınız ancak dosyanın ait olduğu grubu sahip olduğunuz gruplardan birisi olarak belirlemek amacıyla chgrp komutunu normal kullanıcı durumunda da kullanabilirsiniz.

### 11.6 Sahipliğin Özyineli Olarak Değiştirilmesi

chown ve chgrp komutlarının her ikisi de bir dizin ağacındaki tüm dosya ve dizinlerin sahip oldukları kullanıcı ve grupları özyineli olarak değiştirmek amacıyla -R opsiyonuna sahiptirler. Örneğin:

```
# chown -R knoppix /home/knoppix
```

### 11.7 chmod Komutuna Giriş

chmod ve chown komutlarının herhangi bir dosya sistemi nesnesinin ait oldukları grup ve kullanıcıyı değiştirdiğini söyledik. Ayrıca dosya ve dizinler üzerinde tanımlı ve ls -l listelemesiyle gördüğümüz dosya sistemi nesnelerinin yazma-okuma-çalıştırma izinlerini de chmod komutu ile değiştirebiliyoruz. chmod iki ya da daha fazla parametre almaktadır: izinlerin nasıl değişeceğini belirleyen durum parametresi (mode), bunu izleyen ve izin işleminin üzerinde uygulanacağı dosya ya da dosyaların isimleri. Şöyle ki:

```
$ chmod +x Desktop/KNOPPIX.desktop
```

Yukarıdaki örnekte durum parametremiz +x olarak belirlenmiştir. Tahmin edeceğimiz gibi +x bu söz konusu dosyayı hem kullanıcı hem de grup ya da başka herhangi birisi için çalıştırılabilir hale getirecektir. Eğer bir dosyanın tüm çalıştırılma izinlerini kaldırmak istiyorsak, yapmamız gereken:

```
$ chmod -x Desktop/KNOPPIX.desktop
```

## 11.8 kullanıcı/grup/diğerleri Parçacık Yapısı

Şu ana kadar chown komutunun tüm örneklerinde izinler user, group ve other şeklinde tanımlanan her üç nesne için de uygulandı. Genellikle bu üçlünden bir ya da ikisi için izin belirlenmesi daha uygundur. Bunu yapmak için bu üçlünden birisini temsil eden sembolik karakter ile birlikte izin tipini + ya da - olarak belirtiriz. u karakterini user, g karakterini group ve o karakterini other için kullanırız:

```
$ chmod go-w Desktop/KNOPPIX.desktop
```

Yukarıdaki örnekte KNOPPIX.desktop dosyasına yazma hakkını dosyanın ait olduğu grup ve diğer herhangi bir kullanıcı ya da grup için kaldırdık. Ama dosyanın sahibi olan kullanıcı için bu hakkı değiştirmedik.

## 11.9 İzinleri Sıfırlamak

İzin bitlerini açığı kapamak için hepsinin birden sıfırlayabiliriz. = operatörünü kullanarak, chmod komutuna ilgili dosya için sadece belirlediğimiz hakları vermek ve bunlar dışındaki hakları da vermemek istediğimizi söyleyebiliriz.

```
$ chmod =rx Desktop/KNOPPIX.desktop
```

Yukarıda üçlünün tamamı için KNOPPIX.desktop dosyası üzerinde okuma ve çalıştırma hakkını verdik ama write hakkını vermedik. Üçlünün belli birtanesi için hakları ayarlamak istiyorsanız aşağıdaki örnekte gösterildiği gibi = operatöründen önce bu üçlünden ilgili olanı belirttiğiniz sembolü yazabilirsiniz.

```
$ chmod u=rx Desktop/KNOPPIX.desktop
```

## 11.10 Sayısal Kalıplar

Şu ana kadar chmod komutu ile dosya haklarını düzenlemek için hep sembolik kalıpları kullandık. Oysa bu iş için çok sık kullanılan bir başka yöntem daha var, 4 basamaklı bir "sekizli sistem" sayısı. Bu sayısal izin ifadelerinde her bir basamak bir izin grubunu ifade eder. Örneğin 1777 ifadesinde, 777 rakamları dosyanın sahibi olan kullanıcının, grubun ve diğer kullanıcıların izinlerini ifade eder. Baştaki 1 ise özel izinleri ifade eder, bu özel izinler konusuna bu bölümün sonunda gireceğiz. Aşağıdaki tabloda bu sekizli sistem sayılarının nasıl yorumlandığını görüyorsunuz:

```
kalıp rakam
rwx 7
rw- 6
r-x 5
r-- 4
-wx 3
-w- 2
--x 1
--- 0
```

## 11.11 Sayısal izin ifadelerinde sözdizim yapısı

Sayısal ifadeler çoğunlukla, bir dosya üzerindeki hakların tamamıyla ilgili bir düzenleme yapılacaksa kullanışlıdır. Aşağıdaki örneği inceleyecek olursak:

```
$ chmod 0755 Desktop/KNOPPIX.desktop
$ ls -l Desktop/KNOPPIX.desktop
-rwxr-xr-x  2 knoppix knoppix 124 2003-04-25 09:56 Desktop/KNOPPIX.desktop
```

bu örnekte "-rwxr-xr-x" ifadesine denk gelen 0755 kalıbı kullanılmış. Yani dosyanın sahibi olan kullanıcı tüm haklara sahipken, gruba ve diğer kullanıcılara sadece okuma ve çalıştırma hakları veriliyor.



### 11.12 umask

Bir işlem yeni bir dosya yarattığında, o dosyayla ilgili hakları da, kendisine en uygun olacak şekilde belirler. Genellikle herkesin okuyabileceği ve yazabileceği kalıp olan 0666 belirlenir ve bu kalıp aslında tahmin edilenden ve istenenden daha fazla toleranslıdır. Neyse ki, Linux yeni bir dosya yaratıldığında "umask" dediğimiz yapıya danışır. Dosyaya başlangıçta verilen haklar, umask değerine bakılarak, daha gerçekçi ve güvenilir bir seviyeye düşürülür. Komut satırında umask yazdığınızda o andaki umask ayarınızı görebilirsiniz:

```
$ umask
0022
```

Linux sistemlerde umask standart olarak 0022 olarak ayarlanmıştır. Bu diğer kullanıcıların dosyalarınızı (eğer ulaşabilirlerse) okuyabileceği ama değiştiremeyeceği anlamına gelir. Yeni dosyaların daha güvenli olması için umask ayarını değiştirebilirsiniz:

```
$ umask 0077
```

Bu umask değeri, dosyanın grubunun ve diğer kullanıcıların, yeni dosya üzerinde hiçbir hakka sahip olmamalarına sebep olur. Bu umask değeri gruba ait kullanıcıların ve diğerlerinin yeni yaratılan dosyalar üzerinde öntanımlı olarak herhangi bir izne sahip olmalarını engeller. Peki o halde bu umask nasıl çalışır? Dosyalar üzerinde iş yaparken kullanılan "düzenli" (normal) izinlerden farklı olarak umask dediğimiz değer hangi izinlerin OLMAYACAĞINI belirler. İzin-sayısal karşılık tablomuza bakıp 0077 umask değerinin ne anlama geldiğini çözmeye çalışalım:

```
İzin sayı
rwx 7
rw-6
r-x 5
r--4
-wx 3
-w-2
--x 1
---0
```

Tabloya göre 0077 ifadesinin son iki rakamı rwxrwx izin durumuna karşılık gelmektedir. Şimdi de lütfen umask değerinin hangi izinlerin iptal edileceğine dair bilgi içerdiğini hatırlayın. Bu bilgileri birlikte düşünersek görürüz ki "grup" ve "diğer" izinler kapatılırken "kullanıcı" izinlerine dokunulmayacaktır.

### 11.13 suid ve sgid'ye giriş

Makineyi açtığınızda ve bir oturum açtığınızda yeni bir kabuk süreci başlar. Bunu zaten biliyorduk, ama bununla ilgili bilemeyeceğiniz birşey daha var ki, o da bu yeni kabuk süreci (aslında bash) sizin kullanıcı bilginizle çalışır. Böylece, sizin sahibi olduğunuz dosya ve dizinlere ulaşabilir. Aslında, bizler kullanıcı olarak, bizim adımıza işlemler yapan programlara tam anlamıyla bağlıyız. Çünkü kullanıcı bilgilerimizi verdiğimiz programlar, dosya sisteminde, bizim iznimizin olmadığı herhangi bir nesneye ulaşamayacaktır. Örneğin, normal kullanıcılar passwd dosyasına yazma hakkına sahip değildir, çünkü root dışındaki tüm kullanıcıların yazma hakkı kaldırılmıştır:

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root users 1239 2003-04-25 10:19 /etc/passwd
```

Oysa ki, normal kullanıcıların da, en azından dolaylı olarak, bu dosya üzerinde değişiklik yapmaya ihtiyacı olabilir, örneğin şifrelerini değiştirecekleri zaman. Ama normal kullanıcıları bu dosya üzerinde yazma hakkı yoksa nasıl olacak bu işlem?

### 11.14 suid

Linux izin modelinde "suid" ve "sgid" isimli iki özel bit vardır. Eğer çalıştırılabilir bir dosyanın "suid" bit'i ayarlanmışsa, o dosya, o anda çalıştıran kullanıcı değilse, asıl sahibi olan kullanıcının adıyla çalıştırılmış gibi olur. Şimdi /etc/passwd dosyasıyla ilgili problemimize geri dönelim. passwd çalıştırılabilir dosyasına baktığımızda sahibinin root kullanıcısı olduğunu görürüz:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x  1 root  root      24680 2002-04-07 17:59 /usr/bin/passwd
```

Burda gördüğümüz bir detay daha var. Dosya sahibi olan kullanıcının izinlerini ifade eden üçlüde x yerine s harfi bulunuyor. Bu s harfi, ilgili dosya için suid ve çalıştırılabilirlik bitlerinin ayarlandığını belirtiyor. Bu nedenle, passwd komutu çalıştığında, o anda çalıştıran kullanıcı değil de root kullanıcısı tarafından çalıştırılmış gibi olacaktır ve passwd komutu root haklarıyla çalışınca, /etc/passwd dosyasını da bir problem olmadan düzenleyebilecektir.

### 11.15 suid/sgid uyarıları

suid'nin nasıl çalıştığını gördük, sgid'de aynı şekilde çalışır. Programların, o anda çalışan kullanıcının değilde programın üzerindeki grup haklarına göre çalışmasına dayanır.

suid ve sgid hakkında birkaç önemli ipucu verelim. suid ve sgid bitleri ls -l listelemesinde x harfinin bulunduğu yerde bulunurlar. Eğer hem x hemde suid bitleri aktifse, listede s (küçük harf olarak) görülür. Ama, x biti aktif değil, suid biti aktifse S (büyük harf olarak) görüntülenir.

Bir başka önemli nokta: suid ve sgid bitleri birçok durumda çok kullanışlı gibi görünselerde, bazı dikkatsiz kullanımlarda sistemin güvenliğinde önemli açıklara sebep olabilirler. Bu yüzden ne kadar az suid'li program olursa o kadar iyidir. passwd komutu da, suid'ye kesin olarak ihtiyaç duyan birkaç programdan biridir.

### 11.16 suid ve sgid'nin değiştirilmesi

suid ve sgid bitlerinin ayarlanmaları ve iptal edilmeleri şu ana kadar gördüğümüz izin ayarları ile aynıdır. Örneğin suid bitinin ayarlanması şöyledir:

```
# chmod u+s /usr/bin/program
```

Burada da bir dizin üzerinden sgid bitinin kaldırılmasını görüyoruz. Biraz ileride de sgid bitinin dizinler üzerindeki etkilerini inceleyeceğiz.

```
# chmod g-s /home/knoppix
```

### 11.17 İzinler ve dizinler

Şu ana kadar normal dosyaların izinlerinden bahsettik. Konu klasörler olunca, işler biraz değişiyor. Klasörler de izin konusunda aynı kalıpları kullanırlar, ama yorumlanmaları biraz farklıdır. Bir klasör için, 'read' hakkı o klasörün içeriğinin görüntülenebilmesi, 'write' hakkı klasör içinde dosya yaratılabilmesi, 'execute' hakkı da klasör içindeki alt dizinlere ulaşılabilmesi anlamına gelir. 'execute' biti ayarlanmamışsa, klasör içinde ulaşılabilir olmayan dosya sistemi nesneleri olduğu anlamına gelir. 'read' biti ayarlanmamışsa, klasör içinde görüntülenemez nesneler vardır. Bu nesneler ancak, disk üzerindeki tam yolu bilindiğinde ulaşılabilir.

### 11.18 Dizinler ve sgid

Bir dizinin sgid biti ayarlanmamışsa, bu dizinin grup bilgileri, içinde yaratılan herhangi bir dosya sistemi nesnesine de yansacaktır. Bu özellik, aynı gruba üye olan birkaç kullanıcının kullanacağı bir dizin ağacı oluşturulduğunda, kullanışlı olacaktır. Şöyle ki:

```
# mkdir /home/grupalani
# chgrp benimgrubum /home/grupalani
# chmod g+s /home/grupalani
```

Burada, benimgrubum üyesi olan tüm kullanıcılar /home/grupalani altında dosya veya dizin yaratabilir ve bu yeni dosya ve dizinlerin grubu, otomatik olarak benimgrubum atanacaktır. Yeni nesnelerin, benimgrubum üyesi olan diğer kullanıcılar tarafından okunabilirlik, yazılabilirlik ve çalıştırılabilirlik ayarları, o dosyanın yaratıcısı olan kullanıcının umask değerine bağlıdır.

## 11.19 Dizinler ve silme işlemi

Linux izinleri standart ayarları altında günlük kullanımda çok ideal ayarlar değildir. Normal olarak, bir klasöre ulaşma hakkı olan kişinin o klasör içindeki bir dosyayı silme veya yeniden adlandırma hakkı olması beklenir. Bireysel kullanıcıların izinleri böyledir ve en uygunu da budur.

Ancak birçok kullanıcının eriştiği ve kullandığı klasörler için özellikle /tmp ve /var/tmp, bu uygulamanın bazı zararları olabilir. Tüm kullanıcıların erişme hakkı olması, aynı zamanda silme veya değiştirme hakkı olması anlamına da gelir, yani bir kullanıcı diğerinin dosyalarını, sahibi olmadığı halde, silebilir veya değiştirebilir. Bu nedenle /tmp klasörünü önemli işler için kullanmamakta fayda var, çünkü herhangi bir kullanıcı "rm -rf /tmp/\*" yazarak tüm dosyaları silebilir.

Neyse ki Linux, yapışkan bit özelliğiyle bu problemi de çözüyor. /tmp dizinini yapışkan biti ayarlanmışsa (chmod +t ile), bu dizindeki dosyaları sadece root, dizinin sahibi (genellikle root) ya da o dosyanın sahibi silebilir veya yeniden adlandırabilir. Tüm Linux dağıtımlarında, /tmp dizininin yapışkan biti ayarlanmıştır, ancak görüldüğü gibi sadece /tmp dizininde değil birçok yerde yapışkan bit işe yaramaktadır.

## 11.20 Özel ilk basamak

Bu bölümü sona erdirmeden önce, sayısal ifadelerde bahsi geçen ilk basamaktan bahsedelim. Gördüğünüz gibi ilk basamak yapışkan bitlerin, suid ve sgid bitlerinin ayarlanması için kullanılıyor:

```
suid sgid stickymode basamak
on on on 7
on on off 6
on off on 5
on off off 4
off on on 3
off on off 2
off off on 1
off off off 0
```

Bir gruba (workgroup) ait bir dizinin izinlerin, 4 basamaklı sayısal ifadelerle ayarlanması ile ilgili olarak şu örneğe bakalım:

```
# chmod 1775 /home/groupfiles
```

## 12 vi Editörü

Tüm Linux sistemlerinde, kullanıcıların zevklerine hitap eden çeşitli metin editörleri vardır. Bunların arasında, Unix sistemler için özel olarak tasarlanmış olan vi editörü en çok kullanılan editörlerin başında gelir. Vi, başlangıçta karmaşık görünse de hızı ve verimi ile her kullanıcının işini büyük ölçüde kolaylaştıracak bir editördür. En çok kullanılan komut takımlarını öğrendikten sonra vi'dan vazgeçmenin ne kadar güç olduğunu göreceksiniz.

### 12.1 Vi'a başlangıç

PC editörlerin büyük çoğunluğunda klavye, editör komutlarını almak ve basılan tuşları ekrana göndermek için kullanılan iki gruba ayrılmıştır. Kullanıcı aynı anda hem komut işletip hem de yazı yazabilir. Bu tip editörler geniş bir klavye haritasına sahip oldukları için, örneğin fonksiyon tuşlarını veya klavyenin en sağındaki sayısal tuşları da kullanıyor olabilirler. vi'ın tasarım aşamasında bu durumdan kaçınılmış, standart dışı klavyesi olan bilgisayarların da var olabileceği düşünülerek klavye üzerinde kullanılması gereken tuşların sayısını mümkün olduğunca indirilmesine çalışılmıştır. Burada karşımıza vi editörünün çalışma aşamasında üç ayrı işlev geliyor. Birincisi, bilgisayara komutların girdisi sırasında kullanılan komut modu, ikincisi yazı yazarken kullanılan yazı modu ve satır modu.

Komut modunda klavye üzerinde görevi olan tüm tuşlar, bilgisayar komut vermek için kullanılıyor. Yazı modunda ise diğer editörlerdekine benzer şekilde yazı yazmak mümkün oluyor. Klavye modunu değiştirdiğinizde klavye tuşlarının işlevleri de hemen değişiyor. vi editörünü ilk çalıştırdığınız anda komut moduna girersiniz. Bu anda her tuşa ait bir komut çalıştırılmaya hazırdır (örneğin 'j' bir karakter aşağı, 'k' bir karakter yukarı gider). Bu anda kullanıcı yazı moduna geçmek isterse 'i' tuşuna basabilir. Yazı moduna iken klavyeden girilen her karakter ekranda görünür. Bundan sonrası daktilo kullanmaya benzer. Tekrar

komut moduna geçmek için ESC tuşu yardımcı olur. Bir dosya yazarken veya düzeltmede bulunurken her iki yazım stili arasında sürekli gidip geldiğinizi farkedebilirsiniz. Bunlara ek olarak dosya işlemleri veya eşleme yaparken iki mod da kullanılmaz. Linux kullanıcısı vi üzerinde çalışırken en alt satırda vi mesajlarını görür ve gerektiği zaman satır modunda komutları girer. Satır moduna geçmek için ':' (iki nokta üstüste) karakteri kullanılır. Bu tuşa basıldığında ekranın en altında bir satır açılır ve vi sizden bir komut girmenizi bekler. vi komutunu yazıp enter'a bastıktan sonra tekrar editör komut moduna geçersiniz.

## 12.2 Dosya açma, kaydetme ve çıkma

Bir vi oturumunu 4 ayrı şekilde açabilirsiniz. Yeni ve boş bir dosya için:

```
$vi
```

varolan bir dosya için:

```
$vi dosya_adi
```

imlecin, açılacak dosyada n numaralı satırda durması için:

```
$vi +n dosya_adi
```

imlecin, aranan bir kalıbın ilk bulunduğu yerde durması için:

```
$vi +/"aranan kalıp" dosya_adi
```

vi'a ilk girdiğiniz anda komut modunda olursunuz. Yazı moduna geçmek için 'i' veya 'a' tuşuna basın. Bu andan itibaren zamanınızın büyük bir kısmını geçireceğiniz yazı moduna atlısınız. Bıkana kadar yazıp önce ESC tuşuna, ardından iki kere 'Z' tuşuna basın. Bu komut dizisi önce dosyayı kaydedecek, sonra da vi editöründen çıkmanızı ve kabuğa dönmenizi sağlayacaktır. vi' a yeni başlayanların yaptıkları en sık hatalardan biri yazı modundan komut moduna geçerken ESC tuşuna basmamalarıdır. 'w' komutu 'ZZ' komutuna benzer çalışır, hafızada halihazırda bulunan dosyayı, editörden çıkmadan diske kaydeder. Eğer isimsiz bir dosyada üzerinde çalışıyorsanız (örneğin vi dosya\_adi yerine sadece vi yazarak editöre girmişseniz) dosyayı sabit diske yazmak için 'w' satır komutunu kullanmalısınız. Bunun için:

```
:w benimdosyam
```

yazıp satır moduna geçerek komutu girmelisiniz.

Vi editöründen çıkmak için :q yazın.

## 12.3 Vi'da yazma

Şimdi vi yardımıyla bir dosya üzerinde denemeler yapalım. Aşağıdaki komutları kabuk satırında iken yazın:

```
$ vi deneme
```

Ekran silinecek ve solda ~ karakterlerinin bulunduğu bir görüntüyle karşılaşacaksınız. Yazı moduna geçmek için i tuşuna basın ve birkaç satır metin girin. Tekrar komut satırına dönmek için ESC tuşuna basın. Şimdi klavyenizdeki yön tuşlarıyla metin içinde gezinebilir yada : tuşu ile satır moduna geçebilirsiniz.

## 12.4 Silme ve kopyalama

Yazı modunda iken backspace tuşu yardımıyla, komut modunda iken ise 'x' komutu ile istediğiniz kadar harfi silmeniz mümkündür. 'dd' komutu ise imlecin üzerinde bulunduğu satırı siler ve alttaki tüm satırlar bir satır yukarı alır. İlgili komutları kısaca listelemek istersek: dd bulunduğu satırın tümünü siler n dd n satır siler D satırı imleçten itibaren siler dw bir kelime siler dG dosyanın sonuna kadar siler p son silinen karakteri imleçten sonra ekle P son silinen karakteri imleçten önce ekle u son değişikliği geri al x imlecin bulunduğu karakteri sil X imleçten önceki karakteri sil nX imleçten öncek n karakteri sil Silinen karakter veya karakter gruplarının hafızaya kaydedildiğini söylemiştik. Bu hafıza geçici bir tampon vazifesi görür. 'yy' komutu üzerinde bulunduğu satırı kopyalar. Eğer komuttan önce sayı girilirse o kadar satır kopyalanır. Ardından kullanıcı metin üzerinde istediği yere gelip 'p' tuşuna basarak hafızadaki satırları imlecin bulunduğu satırın altından itibaren basar. 'yy' komutu yerine 'Y' de tercih edilebilir.

## 12.5 Arama ve eşleştirme

vi'da çalışırken arama yapmak isteyebilirsiniz. Eğer üzerinde çalıştığımız metin uzun ise belirli bir kelime veya kelime gruplarını aramak çok zaman alabilir. Komut modunda iken '/' karakterine basarak arama/tarama moduna geçin. Ekranın en alt kısmında bir satır açılacaktır. Aramak istediğiniz kelimeyi yazın ve ENTER tuşuna basın. vi editörü imlecin bulunduğu yerden itibaren girilen anahtar sözcüğü metin içinde tarayacaktır. Eğer aranan kelime bulunmuşsa satır imlecini bunun üzerine konumlandırır. '/' komutu dosyanın sonuna doğru arama yaparken '?' komutu başa doğru gider. Aynı kelimeyi tekrar aratmak için 'n' tuşuna basın. Arama işlemi, '/' komutu yardımıyla yapılmışsa ileri doğru, '?' komutu yardımıyla yapılmışsa geriye doğru tekrar başlayacaktır. Metindeki bazı kelimeleri değiştirmek istediğinizde ise 's' komutu yardımınıza koşacaktır:

```
: [x,y]s/eski_ifade/yeni_ifade/tercih
```

komutu x ve y numaralı satırlara arasında, eski\_ifade kalıbını, yeni\_ifade kalıbıyla değiştirecektir. Tercih kısmına gireceğiniz 'c' karakteri her değiştirme işleminden önce kullanıcıdan onay isterken, 'i' karakteri, arama ve değiştirme sırasında büyük-küçük harf ayrımı yapılmamasını sağlar.

## 12.6 Diğer dosyaların metne eklenmesi

İmlecin bulunduğu yerden itibaren ikinci.txt dosyasını halihazırda çalıştığımız metne kopyalamak için:

```
:r ikinci.txt
```

## 12.7 Kabuk komutlarının çalıştırılması

vi altında iken kabuk komutlarını, hatta kabuğun kendisini bile çalıştırmak mümkündür. ':' ile komut ismini girebilirsiniz. Aşağıdaki komut vi'dan çıkmadan ls komutunu çalıştıracak ve sonucu ekrana verecektir.

```
:!ls -la
```

Bunun yanısıra komut çıktısını ekran yerine üzerinde çalıştığımız metine aktarmak için aşağıdaki vi komutunu kullanın:

```
:r! ls -al
```

# 13 Linux hesap yönetimi

## 13.1 /etc/passwd dosyası

Bu bölümde Linux'un hesap yönetimi mekanizmasını inceleyeceğiz. Sistemdeki tüm kullanıcıların bilgilerinin bulunduğu /etc/passwd dosyasıyla başlayalım. "less /etc/passwd" yazarak, kendi sisteminizdeki dosyayı görüntüleyebilirsiniz. Bu dosyadaki her bir satır, bir kullanıcı hesabını belirtir. /etc/passwd dosyasından bir örnek satır ele alalım:

```
knoppix:x:1000:1000:Knoppix User:/home/knoppix:/bin/bash
```

Gördüğünüz gibi, bu satırda çok fazla bilgi yokmuş gibi duruyor. Aslında /etc/passwd dosyasındaki her satır : ile ayrılmış birkaç parçadan oluşur.

İlk kısım kullanıcı adını belirtir (knoppix), ikinci bölümde bir x harfi görüyoruz. Eski Linux sistemlerinde, bu kısımda şifrelenmiş bir halde parolalar bulunurdu ve bu parolalar kullanıcı girişinde onaylama için kullanılırdı. Ama yeni Linux sistemlerinin neredeyse tamamında, şifreler başka bir dosyada tutuluyor.

Üçüncü alandaki (1000) ifadesi, kullanıcının numarasını, dördüncü alandaki (1000) ifadesi de, bu kullanıcının bağlı olduğu grubu belirtir. Biraz sonra (1000) numaralı grubun nerede tanımlandığını da göreceğiz.

Beşinci bölüm, bu hesabın düzyazı olarak bir açıklamasını içerir, bu örnekte, açıklama kullanıcının tam adı olarak verilmiş. Altıncı bölümde, kullanıcının home dizininin yeri belirtilir. Son olarak, yedinci kısımda o kullanıcı için standart olarak açılacak olan kabuğu ifade eder. Bu kabuk kullanıcı bilgisayara girdiğinde otomatik olarak açılan kabuktur.

### 13.2 /etc/passwd hakkında ipuçları

Eğer dikkat ettiyseniz, /etc/passwd dosyasında, sizin bilgisayara girerken kullandığınız hesap dışında birçok hesap daha olduğunu görmüşsünüzdür. Bunun sebebi, farklı Linux bileşenlerinin, güvenliği arttırmak amacıyla, kendilerine ait kullanıcı hesapları bulunmasıdır. Bu sistem hesaplarının kullanıcı numaraları 100'den küçüktür ve birçoğunda standart kabuk alanında /bin/false ifadesi yer alır. /bin/false programı bir hata mesajı verip çıkmaktan başka hiçbir şey yapmadığı için, bu sistem hesaplarının normal kullanıcılar tarafından kullanılmasını engeller. Bu hesaplar sadece sistem içi kullanım içindir.

### 13.3 /etc/shadow dosyası

Kullanıcı hesaplarına ait bilgilerin /etc/passwd dosyasında tutulduğunu gördük. Linux sisteminde bu dosya ile ortak çalışan bir dosya daha vardır: /etc/shadow dosyası. Bu dosya /etc/passwd dosyası gibi herkes tarafından okunabilir bir dosya değildir. Bu dosyanın içeriği, sadece root tarafından görüntülenebilen, şifrelenmiş parola bilgilerinden ibarettir. Bu dosyadan alınmış bir satırı inceleyelim:

```
knoppix:$1$inZ2aQ4f$Us100a1moVCgbwXsx.a.E.:12167:0:99999:7:::
```

Yine her bir satır ayrı bir kullanıcı hesabıyla ilgili bilgileri içeriyor ve her bir bölüm : ile ayrılmış durumda. İlk kısım bu parola bilgisinin hangi kullanıcıya ait olduğunu belirtir. İkinci kısımda ise kullanıcının, parolası şifrelenmiş bir halde bulunur. Diğer alanların açıklamalarını da aşağıdaki tabloda bulabilirsiniz:

- 3. Alan 1/1/1970'ten, son şifre değişimine kadar geçen gün sayısı.
- 4. Alan Parola değiştirebilme izni verilene kadar geçen gün sayısı (serbest değişiklik için 0'dır)
- 5. Alan Sistemin, kullanıcıdan parolasını değiştirmesini isteyeceği zamana kadarki gün sayısı (-1: asla)
- 6. Alan Kullanıcının, parolanın geçersiz kalacağı günden, kaç gün önce uyarılacağı (-1: uyarı yok)
- 7. Alan Parolanın iptalinden, hesabın sistem tarafından otomatik iptal edilmesine kadar geçen gün sayısı (-1: iptal yok)
- 8. Alan Hesabın iptal edildiği gün sayısı (-1: hesap hala aktif)
- 9. Alan Daha sonra kullanılmak üzere ayrılmıştır.

### 13.4 /etc/group dosyası

Şimdi de, /etc/group dosyasına gözatalım. Linux sistemindeki tüm grup bilgileri bu dosyada tutulur. Örnek bir satıra bakalım:

```
knoppix:x:1000:
```

/etc/group dosyasındaki satırların anlamlarını inceleyelim. İlk kısım grubun adını barındırıyor. İkinci alanda ise bir parola görüntüsü olarak x karakteri bulunur. Üçüncü kısımda, bu gruba ait olan grup numarasını verir. Bu örnekte boş olarak gözüken dördüncü bölümde de bu gruba üye olan kullanıcıların isimleri vardır.

/etc/passwd dosyası ile ilgili olarak verdiğimiz örnekte, grup numarası olarak 1000 vardı. Bu, knoppix kullanıcısının, knoppix grubunun bir üyesi olduğu anlamına geliyor. Ama /etc/group dosyasında, knoppix grubunun bilgilerinin arasında üye olarak knoppix kullanıcısının belirtilmediğine dikkat edin.

### 13.5 Gruplara ilişkin notlar

Bazı sistemlerde, tüm yeni kullanıcı hesapları için yeni bir grup açılır ve yeni kullanıcı bu grupla ilişkilendirilir. Hatta o kullanıcı ve grubun numaraları dahi aynı verilir. Bazı sistemlerde ise, tüm yeni kullanıcı hesapları, bir standart "users" (kullanıcılar) grubuna üye yapılır. Bu fark tamamen sahip olduğunuz sistemde, sizin tercihinize kalmıştır. Her kullanıcı için birde grup oluşturulması, kendi dosyaları üzerindeki erişim haklarını daha kolay ayarlayabilmesi ve güvendiği kişileri kendi grubuna üye yapabilmesi kolaylıklarını da yanında getirir.

### 13.6 Elle kullanıcı ve grup yaratma

Şimdi kendi kullanıcı ve grubunuzu yaratma işleminin nasıl yapılacağına bakalım. Bunu öğrenmenin en iyi yolu, sistemde elle yeni bir kullanıcı yaratmaktır. Başlamadan önce EDITOR değişkeninizin, en çok kullandığınız editör olarak ayarlanıp ayarlanmadığını kontrol edelim:

```
# echo $EDITOR
vim
```

Eğer ayarlanmamışsa, aşağıdaki komutla EDITOR değişkeninizi ayarlayabilirsiniz:

```
# export EDITOR=/usr/bin/emacs
```

Şimdi şunu yazın:

```
# vipw
```

Şimdi /etc/passwd dosyasını, seçtiğiniz editörle açılmış durumda göreceksiniz. Sistemdeki passwd ve group dosyalarını düzenlemek istendiğinde vipw ve vigr komutlarının önemini kavramak gerekir. Bu komutlar, sistem için kritik olan passwd ve group dosyalarını açarken ekstra tedbirli olurlar ve işlem sonrasında doğru biçimde kapatılmasını (kilitlenmesini) sağlarlar. Böylece hayati önemi olan bu dosyalar zarar görmez.

### 13.7 /etc/passwd dosyasını düzenlemek

Yukarıdaki komutları yazdıktan sonra /etc/passwd dosyanız karşınıza geldi. Şimdi alttaki satırı ekleyin:

```
deneme:x:3000:3000:Deneme kullanicisi:/home/deneme:/bin/false
```

Bu satır sayesinde, sisteme "deneme" adlı ve numarası 3000 olan bir kullanıcı eklemiş olduk. Bu kullanıcıyı henüz yaratmadığımız ama numarası 3000 olan gruba üye yaptık. İstersek bu kullanıcıyı "users" grubuna da ekleyebilirdik. Bunun için grup numarası olarak 3000 değil, o grubun numarasını vermemiz gerekirdi. Yeni kullanıcı için "Deneme kullanicisi" bilgisini ekledik, home dizinini /home/deneme olarak belirledik ve standart kabuğunu da /bin/false olarak ayarladık. Kabuk ayarını güvenlik amacıyla /bin/false olarak seçtik. Eğer bir test kullanıcısı değilse normal bir kullanıcı yaratıyor olsaydı, standart kabuk olarak /bin/bash girecektik. Şimdi değişiklikleri kaydedelim ve çıkalım.

### 13.8 /etc/shadow dosyasını düzenlemek

Şimdi, bu yeni kullanıcımız için /etc/shadow dosyasına yeni bir giriş yapmamız gerekiyor. Bunun için vipw -s yazınca, daha önceden seçtiğimiz editör, /etc/shadow dosyasıyla birlikte karşımıza gelecektir. Şimdi, varolan kullanıcılardan (parolası olan ve standart sistem kullanıcılarından daha uzun olan) bir tanesinin bilgilerini içeren satırı kopyalayın:

```
knoppix:$1$inZ2aQ4f$Us100a1moVCgbwXsx.a.E.:12167:0:99999:7:::
```

Buradaki kullanıcı adını, yeni oluşturduğunuz kullanıcı adı ile değiştirin ve diğer tüm bilgilerin aynı kaldığına (şifrelenmiş parolanın görüntüsü dahil) dikkat edin.

```
deneme:$1$inZ2aQ4f$Us100a1moVCgbwXsx.a.E.:12167:0:99999:7:::
```

Şimdi kaydedip çıkın.

### 13.9 Parola belirleme

Tekrar komut satırına dönmüş durumdasınız. Şimdi, yeni kullanıcımız için bir parola belirleme zamanı:

```
# passwd deneme
Enter new UNIX password: (deneme kullanicisi icin bir parola girin)
Retype new UNIX password: (deneme kullanicisi icin belirlediginiz parolayı
tekrar girin)
```

### 13.10 /etc/group dosyasını düzenlemek

/etc/passwd ve /etc/shadow dosyalarını düzenledik. Şimdi /etc/group dosyasında yapılması gereken ayarları inceleyelim. Bunun için komut satırına şunu yazalım:

```
# vigr
```

/etc/group dosyanız, editörle birlikte açılmış ve düzenlemelerinize hazır olarak karşınızda duruyor. Eğer yeni yaratılan kullanıcıyı, varolan gruplardan birine üye yaparsanız, bu dosyaya yeni bir gruba ilgili bilgiler girmenize gerek yoktur. Ama yeni "deneme" kullanıcısı için bir grup yaratacaksanız aşağıdaki satırı dosyaya eklemeniz gerekiyor:

```
deneme:x:3000:
```

şimdi kaydedip çıkın.

### 13.11 Home dizin yaratma

İşlemimiz bitmek üzere. Şimdi de yeni kullanıcımızın home dizinini yaratmak için aşağıdaki komutları yazalım:

```
# cd /home
# mkdir deneme
# chown deneme.deneme deneme
# chmod o-rwx deneme
```

Artık kullanıcımızın home dizini olması gereken yerde ve hesap kullanıma hazır. Eğer bu hesabı kullanacaksanız, vipw ile deneme kullanıcısının standart kabuk bilgisini /bin/bash olarak değiştirmeniz gerekiyor. (Hatırlarsanız, güvenlik için bu bilgi /bin/false olarak ayarlanmıştı)

### 13.12 Hesap yönetim araçları

Bir kullanıcının yaratılmasını ve ilgili ayarların nasıl yapılacağını gördük. Yeri geldikçe Linux'ta vakit kazandırıcı çeşitli hesap yönetim işlerini göreceğiz. Ayrıca unutmayın ki, bir komut hakkında bilgi aradığınızda, kılavuz sayfaları yardımınıza koşacaktır.

**newgrp:** Standart ayarlar gereği, bir kullanıcının yarattığı dosya üzerindeki haklar ayarlanırken o kullanıcının /etc/passwd dosyasındaki grup bilgileri de dikkate alınır. Eğer o kullanıcı başka gruplara da üye ise, newgrp thisgroup yazarak, o anda bağlı olduğu grubu thisgroup olarak değiştirebilir. Böylece yeni yarattığı bir dosyanın izinleri belirlenirken bu thisgroup grubu dikkate alınır.

**chage:** Bu komut /etc/shadow dosyasındaki parolanın eskimesi ve değiştirilmeye zorlanması ile ilgili sürelerin görüntülenmesi ve değiştirilmesi için kullanılır.

**gpasswd:** Genel amaçlı bir grup yönetim aracıdır.

**groupadd/groupdel/groupmod:** /etc/group dosyasına grup ekleme, silme ve düzenleme yapmak için kullanılırlar.

**useradd/userdel/usermod:** /etc/passwd dosyasına kullanıcı ekleme, silme ve düzenleme yapmak için kullanılırlar. Bu komutların ayrıca çok yararlı başka fonksiyonları da vardır. Bunun için kılavuz sayfalarına bakabilirsiniz.

**pwconv/grpconv:** passwd ve group dosyalarını, yeni nesil gölgeli parola biçimine dönüştürmekte kullanılırlar. Ancak artık Linux sistemlerinin tamamında gölgelenmiş parola yöntemi kullanıldığından, bu komutlara hiçbir zaman ihtiyacınız olmayacaktır.

**pwunconv/grpunconv:** passwd, shadow ve group dosyalarını eski tip, gölgelenmemiş parola biçimine dönüştürmeye yararlar. Bu komutlara da hiçbir zaman ihtiyacınız olmayacaktır.

## 14 Kullanıcı ortamını ayarlama

### 14.1 "fortune" programına giriş

Kullandığımız kabuğun pek çok seçeneği vardır ve bunları kendi keyfinize göre şekillendirebilirsiniz. Ancak şimdiye dek bu değişiklikleri kalıcı hale getirip her login oluşunuzda otomatik olarak gerçekleşecek şekilde nasıl ayarlayabileceğinize değinmedik. Şimdiye dek bunları her seferinden komut satırından elle girmeniz



gerekiyordu. Bu bölümde ise başlangıç dosyalarını düzenleyerek ortamınızı nasıl özelleştirebileceğinize değineceğiz.

Başlangıç olarak giriş anında ilginç mesajlarla karşılaşma işini ayarlayalım. Örnek bir mesaj görmek için fortune komutunu çalıştırın:

```
$ fortune
Don't Worry, Be Happy.
-- Meher Baba
```

```
.bash
_profile Şimdi öyle bir ayarlama yapalım ki fortune komutu her login olduğumuzda bir kere çalışsın. En sevdiğiniz metin düzenleyiciyi kullanarak home dizininizdeki .bash_profile dosyasını açın. Eğer böyle bir dosya mevcut değilse yaratmaktan çekinmeyin. En başa şu satırı yazın:
```

```
fortune
```

Şimdi kabuktan çıkın ve tekrar girin. xdm, gdm ve kdm gibi bir görüntü yöneticisi kullanmıyorsanız sisteme giriş yapar yapmaz eğlenceli bir mesaj ile karşılaşacaksınız:

```
login:
Password:
You will have good luck and overcome many hardships.
$
```

login kabuğu bash başlatıldığında home dizininizdeki .bash\_profile dosyasını okur ve oradaki her komutu komut satırından girilmişçesine çalıştırır. Buna "dosyayı okuma" ("sourcing") denir.

Bash, nasıl başlatıldığına bağlı olarak farklı davranır. Eğer bir "login" kabuğu olarak başlatılırsa yukarıdaki gibi davranır yani önce sistem genelinde tanımlanmış olan /etc/profile dosyasında yazılanları sonra da size özel ~/.bash

\_profile dosyasını okur ve buradaki komutları uygular.

Bash kabuğunu login kabuğu olarak çalıştırmanın iki yöntemi vardır. Birinci yöntem siz sisteme giriş yaptığınızda (login) uygulanan yöntemdir yani -bash isimli süreç çalıştırılır. Bunu süreç listesinde de görebilirsiniz:

```
$ ps u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
knoppix   5065  0.1  0.8  3136  2128 ttty0    S      11:04   0:00 /bin/bash
```

Göreceğiniz liste muhtemelen çok daha uzun olacaktır ancak en azından bir tane COMMAND sütunu görmelisiniz ve bunun altında da kabuğunuzun ismi yazmalı, bizim örneğimizde /bin/bash yazdığı gibi.

## 14.2 –login’i anlamak

Bash kabuğuna "login" kabuğu olarak çalışmasını söylemenin bir yolu daha vardır: –login komut satırı seçeneğini kullanarak. Bu seçenek bazen terminal öykünücüleri (xterm gibi) tarafından açtıkları bash oturumlarının açılış esnasında login mekanizmasını çalıştırmak için kullanılır.

Sisteme girdiğinizde, kabuğunuzun birkaç kopyası daha çalışıyor olacaktır. Bu kabuklar –login ile başlatılmadıkça veya sürecin adında bir tire (dash) olmadıkça, gerçek anlamda sisteme giriş kabukları olmayacaklardır. Eğer bir kabuk sizden bir bilgi isteme veya bir bilgi bekleme durumuna geçerse interaktif kabuk olarak adlandırılır. Eğer kabuk, sisteme giriş kabuğu değil ama interaktif olarak açılırsa, /etc/profile ve ~/.bash\_profile dosyalarını tanımayacak, kaynak olarak ~/.bashrc dosyasına bakacaktır.

interactive login profiler yes yes sourceignore yes no ignoresource no yes sourceignore no no ignoreignore

## 14.3 İnteraktifliğin test edilmesi

rsh ve scp gibi komutlar çalıştırıldığında, kabuğunuz gerçekten interaktif olmadığı halde ~/.bashrc dosyasını kaynak olarak seçebilir. Bu önemlidir, çünkü fortune komutundaki gibi görevi sadece ekrana birşey yazmak olan komutlarla interaktif olmayan bu kabuk oturumlarını kolaylıkla kirletebilirsiniz. Bu yüzden, başlangıç dosyasındaki bir yazıyı ekrana yazdırmadan önce, halihazırda çalışmakta olan kabuğun PS1 değişkenini kontrol ederek, gerçekten interaktif bir kabuk olup olmadığını denetlemek faydalı olacaktır:

```
if [ -n "$PS1" ]; then
fortune
fi
```

#### 14.4 /etc/profile ve /etc/skel

Sistem yöneticisi olarak /etc/profile dosyasıyla çok yakın ilişkileriniz olacaktır. Her ne kadar ilk girişte tüm kullanıcılar tarafından kullanılıyor olsa da, bu dosyayı çalışır konumda tutmak önemlidir. Ayrıca yeni yaratılmış kullanıcı hesaplarının sisteme ilk girişlerinde herşeyin doğru çalışmasını sağlamak için de, bu dosya güçlü bir araçtır.

Ama bazen, yeni kullanıcılar için standartlar belirlemek isterken, bir yandan da değişiklikler yapma hakkını o kullanıcılara vermek istersiniz. Bu durumda /etc/skel dizini devreye girer. Yeni bir kullanıcı hesabı açmak istediğinizde kullandığınız useradd komutu, /etc/skel dizini altındaki tüm dosyaları, yeni yaratılan kullanıcının home dizinine kopyalar. Dolayısıyla, /etc/skel dizininin altına yardımcı .bash\_profile ve .bashrc dosyaları koyarak, yeni kullanıcılarınıza iyi bir başlangıç sunabilirsiniz.

#### 14.5 export

Kabuk değişkenleri, tüm yeni açılan kabuklarda aynı şekilde kullanılmak amacıyla işaretlenebilirler. Bu işlem ihraç (export) için işaretlenme olarak adlandırılır. Kendi kabuğunuzda da ihraç için işaretlenmiş değişkenlerin listesini görebilirsiniz.

```
$ export
declare -x HOME="/home/knoppix"
declare -x HZ="100"
declare -x LANG="tr_TR"
declare -x LANGUAGE="tr"
declare -x LOGNAME="knoppix"
declare -x MAIL="/var/mail/knoppix"
declare -x OLDPWD
declare -x PATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/usr/local/sbin:/usr/local/bin:/usr/games:."
declare -x PWD="/home/knoppix"
declare -x SHELL="/bin/bash"
declare -x SHLVL="2"
declare -x TERM="xterm"
declare -x USER="knoppix"
```

#### 14.6 Değişkenlerin ihraç için işaretlenmesi

Eğer bir değişken ihraç için işaretlenmemişse, yeni açılan kabuklar bu değişkenin ayarlarını göremeyecektir. Ancak, bir değişken export built-in'ine (ihraç yerleşğine) geçirilerek kolaylıkla işaretlenebilir:

```
$ FOO=foo
$ BAR=bar
$ export BAR
$ echo $FOO $BAR
foo bar
$ bash
$ echo $FOO $BAR
bar
```

Bu örnekte FOO ve BAR değişkenleri ayarlanmış durumdadır ama sadece BAR değişkeni ihraç için işaretlendi. Yeni bir kabuk oturumu açıldığında, FOO değeri kaybolacaktır. Bu yeni kabuk oturumu kapatıldığında, orjinal kabuğun hala hem FOO hemde BAR değerlerini koruduğu görülecektir:

```
$ exit
$ echo $FOO $BAR
foo bar
```

## 14.7 Export ve set -x

Bu yapı sayesinde, ~/.bash\_profile veya /etc/profile'da yaratılan ve ayarlanan değişkenler ihraç için işaretlendiğinde, tekrar ayarlanmalarına gerek kalmaz. Ama bazı değişkenler ihraç için işaretlenemezler, sistemin kalıcılığı için, bunların ~/.bashrc dosyasına ve profile konması gerekir. Bu tercihler set yerleşik özelliğiyle (built-in) ayarlanır.:

```
$ set -x
```

-x, kabuğun çalıştırdığı tüm komutları aynı zamanda ekrana da yazmasını sağlar:

```
$ echo $FOO
+ echo foo
foo
```

Bunlar bazı beklenmedik benzerlikleri ve kabuk davranışlarını anlamamızda yardımcı olur. -x tercihini iptal etmek için set +x yapmak yeterlidir. set yerleşikliğiyle ilgili tüm tercihleri bash kılavuz sayfasında bulabilirsiniz.

## 14.8 "set" ile değişkenlerin ayarlanması

set yerleşikliği tercihe bağlı olarak, değişkenlerin ayarlarının yapılması içinde kullanılabilir. Ancak bu tip kullanımlarda zorunlu değildir. Örneğin set FOO=foo kabuk komutu FOO=foo ile tamamen aynı işi yapar. Bir değişkenin ayarlarının iptal edilmesi de unset yerleşikliği ile yapılabilir:

```
$ FOO=bar
$ echo $FOO
bar
$ unset FOO
$ echo $FOO
```

## 14.9 "Unset" ve "FOO=" farkı

Aradaki farkı söylemek her ne kadar zor olsa da, bu komutlar bir değişkenin değerini sıfırlamakla aynı şey değildir. Bunu anlamamanın bir yolu, set yerleşikliğini, halihazırdaki tüm değişkenleri listelemesi için parametresiz olarak çağırmasıdır:

```
$ FOO=bar
$ set | grep ^FOO
FOO=bar
$ FOO=
$ set | grep ^FOO
FOO=
$ unset FOO
$ set | grep ^FOO
```

set yerleşikliğini parametresiz olarak çağırarak export yerleşikliğini kullanmak hemen hemen aynıdır. Aradaki temel fark, set yerleşikliği sadece ihraç için işaretlenmiş olanları değil, tüm değişkenleri listeler.

## 14.10 Komutların çalışmalarını etkilemek için ihraç işlemi

Çevre değişkenlerini ayarlayarak, komutların çalışmalarını etkilemek mümkündür. Yeni kabuk oturumları gibi, kabuk penceresi üzerinden çalıştırdığınız programlar da, sadece ihraç için işaretlenmiş değişkenleri görebilecektir. Örneğin man komutu, yazı dosyasında bir sayfadan diğerine geçerken kullanacağı program için PAGER değişkenine bakar.

```
$ PAGER=less
$ export PAGER
$ man man
```

Pager değişkeninin less olarak ayarlanması, man komutunun ekranda dosyanın sadece bir sayfasını görüntüleyeceğini, bir sonraki sayfaya geçmek için space tuşunu bekleyeceğini gösterir. Ama PAGER değişkenini cat olarak ayarlarsak, tüm yazı dosyasının tamamı, ara vermeksizin, bir kerede görüntülenecektir.

```
$ PAGER=cat
$ man man
```

## 14.11 "env" kullanımı

Biraz önceki işlemde, eğer PAGER değişkenini tekrar less olarak değiştirmeyi unutursanız, man (ve daha birçok komut) görüntüleyecekleri yazı dosyalarını, bir kerede ara vermeden gösterecektir. PAGER değişkeninin sadece bir kereliğine cat olarak ayarlanmasını isterseniz, env komutunu kullanmalısınız:

```
$ PAGER=less
$ env PAGER=cat man man
$ echo $PAGER
less
```

Bu durumda PAGER değişkeni cat değeriyle birlikte ihraç edilecektir, ancak kabuk oturumundaki PAGER değeri değişmeden kalacaktır.

## 15 Midnight Commander (MC)

Midnight Commander (MC), Norton Commander benzeri bir dosya yöneticisi ve dizin görüntüleyicisidir. Yeni başlayanların çok seveceği birçok özellik ve ipucu ile, MC en hızlı ve en kullanışlı araç kabul edilir. Ashında MC'nin kılavuz sayfası 2500 satıra yakındır ama biz burada en önemli kullanım alanlarını inceleyeceğiz. Hemen hemen tüm Linux dağıtımlarıyla birlikte gelen, dosya taşıma, kopyalama veya silme işlemlerini, hem yerel makinede, hemde ftp ile network üzerinden en hızlı gerçekleştirmenizi sağlayacak olan MC'yi çalıştırmak için komut satırına:

```
$mc
```

yazmak yeterli. MC'deyken Ctrl-o seçeneği ile tekrar komut satırına dönebilir, yine ctrl-o ile mc'ye geçebilirsiniz. Üstte F9 tuşu ile ulaşabileceğiniz menü çubuğu görünür. Ok tuşlarıyla bu menü çubuğunda gezinebilir, enter tuşu ile de o seçeneğe girebilirsiniz. Ekranın altında da, MC'de sıkça kullanılan fonksiyon tuşlarıyla ulaşabileceğiniz tercihler bulunur. Bazen bu tercihler siz ilgili tuşa bastığınızda çalışmayabilir. Bu, fonksiyon tuş takımının farklı özellikler için ayarlandığı anlamına gelir. Bu durumda ilgili değişiklikleri yapmanız gerekecektir. Ayarlar

Menü çubuğundan "Options" sekmesini seçip "Configuration" kısmına girdiğinizde yapabileceğiniz ayarları görebilirsiniz. Tab tuşuyla tercihler arasında gezip space tuşuyla da tercihi aktif hale getirebilir veya aktif olan bir tercihi iptal edebilirsiniz. Buradaki ayarlardan bir tanesi "lynx-like motion" dur. Bu tercih aktif olduğunda bir dizinin içine girmek için sağ ok tuşu, bir önceki dizine dönmek içinde sol ok tuşu kullanılabilir. Böylece enter tuşuna bağlı kalmaktan kurtulmuş olursunuz.

### 15.1 Dizin panelleri arasında gezinme

Görüntünüz temel olarak iki ayrı parçaya bölünmüş gibidir. Sağda ve solda bulunan bu paneller bulunduğunuz dizinlerinin içeriklerini görüntülemenizi sağlar. Bir panelden diğerine tab tuşuyla geçebilirsiniz. Aktif olan panel, işaretlenmiş olandır ve herhangi bir operasyonda kaynak olarak bu işaretli dosya alınacaktır. Kopyalanacak işaretli dosya diğer paneldeki dizine kopyalanacaktır. Tab tuşuyla paneller arasında gezinip, yukarıda anlattığımız gibi, ilgili hedef dizini bulduktan sonra tekrar tab'la eski dosyaya dönüp F5 ile kopyalama işlemini gerçekleştirebilirsiniz. Tüm kopyalama işlemleri için kullanıcıdan doğrulama istenecektir, o yüzden bir hata olduğunda geri dönüş yapılması kolaydır. Ayrıca insert tuşu yardımıyla birden fazla dosyayı seçebilir, - veya + tuşlarına basarak, joker karakterler yardımıyla toplu dosya transferleri gerçekleştirebilirsiniz. F6 yeniden adlandırma / taşıma, F5 kopyalama, F8 silme içindir. Yeni bir dizin yaratmak içinde F7'yi kullanabilirsiniz. Dosya düzenleme

F4 ile işaretli dosyanın içerisine girebilir ve ilgili haklara sahipseniz dosya üzerinde düzenleme yapabilirsiniz. F4 size mcedit programını çağıracaktır. Kullanımı çok basit olan bu programla, dosya üzerinde ilgili değişiklikleri yaptıktan sonra F2 ile kaydedip F10 ile mcedit'ten çıkabilirsiniz.

### 15.2 Fare desteği

Midnight Commander, fare desteği de sağlar. Dosyaları işaretlemek, bir dizinin içine girmek, veya bir menüyü açmak için fare ile tıklamanız yeterlidir. Çalıştırılabilir dosyaları da iki kere tıklayarak çalıştırabilirsiniz. Ama yine de klavyenin çok daha kullanışlı olacağını göreceksiniz.

### 15.3 FTP işlemleri

Sol ya da sağ panelin menüsünden FTP bağlantısını seçerek, açılan pencereye FTP adresini girdiğinizde, panellerden birinin FTP üzerinden ulaşabileceğiniz bir alanı göstermesini ve yerel makineniz ile bu FTP alanı arasında dosya transferleri yapılmasını sağlayabilirsiniz.

## 16 Linux Dosya Sistemi

### 16.1 Blok Aygıtlar

Bu bölümde linux dosya sistemini ayrıntılı inceleyeceğimizden bir sistem yöneticisinin bilmesi gereken tüm detaylardan haberdar olacaksınız. Başlangıç olarak blok aygıtlar ile giriş yapacağız. Bir linux sisteminde en çok bilinen blok aygıtlardan birtanesi de IDE sürücüsüdür. Sistemde /dev/hda olarak temsil edilmektedir. Eğer sisteminiz SCSI sürücü ile çalışıyorsa bu durumda ilk sabit sürücünüz /dev/sda olacaktır.

Yukarıda belirtilen blok aygıtlar disk için soyut bir arayüz teşkil eder. Kullanıcı programları sürücünün IDE ya da SCSI olması ile ilgilenmeksizin disk ile iletişimde bu blok aygıtları kullanırlar. Program temelde disk üzerindeki alanı, bitişik desteler şeklinde ve rastgele-erişilebilir 512lik byte blokları şeklinde adresler (böler).

### 16.2 Tüm Diskler ve Bölümler

Linux altında, mkfs olarak adlandırılan özel bir komuta belirli bir blok aygıtı komut satırı parametresi olarak vererek dosya sistemleri yaratırız.

Teorik olarak tüm diski /dev/hda veya /dev/sda gibi belli bir dosya sistemine ev sahipliği yapacak şekilde tek bir blok aygıt olarak kullanmak mümkün olsa da (bir tanesi tüm diski temsil edecek şekilde) pratikte bu kullanım neredeyse hiç tercih edilmez. Bunun yerine tüm blok aygıtlar, bölüm (partition) denilen daha küçük ve yönetimi daha kolay parçalara ayrılırlar. Bölümler, fdisk denilen ve her disk üzerinde saklanan bölüm tablosunu yaratmaya ve düzenlemeye yarayan bir araç ile oluşturulmaktadır. Bölüm tablosu tüm diskin nasıl bölüneceğini tam olarak belirler.

### 16.3 fdisk Kullanımı

fdisk kendisine verilen seçenekleri yardımıyla kullanılabilir. Bu seçenekleri ve fdisk hakkında ayrıntılı bilgiyi "man fdisk" yazarak alabilirsiniz.

Fdiski çalıştırmak için bir aygıt ismini de vermeniz gerekir Şimdi gelin hem fdisk'i kullanalım hem de bir bölümü önce silip sonra yeniden yaratalım:

```
# fdisk /dev/hda
```

```
The number of cylinders for this disk is set to 4865.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

ğimiz işlemi yapabiliriz. Tüm komutları listelemek için "m" yazıp enter'a basarsak aşağıdaki gibi bir liste göreceğiz

```
Command (m for help): m
Command action
a  toggle a bootable flag
b  edit bsd disklabel
c  toggle the dos compatibility flag
d  delete a partition
l  list known partition types
m  print this menu
n  add a new partition
o  create a new empty DOS partition table
p  print the partition table
q  quit without saving changes
```

```

s   create a new empty Sun disklabel
t   change a partition's system id
u   change display/entry units
v   verify the partition table
w   write table to disk and exit
x   extra functionality (experts only)

```

Command (m for help):

fdisk ile belli bir aygıt üzerinde yer alan bölümler hakkında doğrudan bilgi almak için -l parametresi aygıt adı ile birlikte kullanılır. Yani:

```
# fdisk -l /dev/hda
```

```

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

```

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1	*	1	1023	8217216	7	HPFS/NTFS
/dev/hda2		1024	4865	30860865	f	Win95 Ext'd (LBA)
/dev/hda5		1024	2316	10385991	b	Win95 FAT32
/dev/hda6		2317	2397	650601	82	Linux swap
/dev/hda7		2398	4513	16996738+	83	Linux
/dev/hda8		4514	4865	2827408+	83	Linux

Bu örnekte görüyoruz ki sistemimizde hda6, hda7 ve hda8 adında ve linux ile ilişkili 3 adet bölüm yer almaktadır. Bu bölümlerden hda6 ile belirtilen linux swap dosya sistemini, hda7 root denilen ana dizinlerin yer aldığı dosya sistemini ve hda8 de üçüncü bir dosya sistemini temsil etmektedir. Daha da yukarıda da windows ile ilişkili dosya sistemlerini görmekteyiz. Tabloda her bölümün başladığı ve bittiği silindir numaraları da ayrıntılı olarak görülmekte ve her bölümde yer alan bir dosya sistemi için de id numarası verilmektedir. Linux ile ilişkili dosya sistemleri 83 nolu id ile belirlenir.

Son olarak yine aynı amaca yönelik ve daha görsel bir menü sunan "cfdisk" aracını da kullanabilirsiniz. Bu araç size bir menü sağlamak ve yapmak istediğiniz tüm işlemleri bu menü ile kolaylıkla yapmanıza olanak sağlamaktadır. Sisteminizde bu program varsa sadece "cfdisk" yazarak çalıştırabilirsiniz:

```
# cfdisk
```

PC'lerin ilk zamanlarında bölüm yapıcı programlar sadece dört bölüm oluşturmaya (bunlara birincil -primary- deniyordu) izin veriyorlardı. Bu çok sınırlayıcıydı, o yüzden uzatılmış(extended) bölüm diye bir yöntem ile bu sınırlama aşıldı. Uzatılmış bölüm, birincil bölüme çok benzer ve yine dört bölümlük sınırlamanın içinde kalırlar. Fakat uzatılmış bölümlerin içine istediğiniz kadar mantıksal (logical) bölüm koyabilirsiniz, böylece dört bölüm sınırı da aşılmış olur.

## 16.4 fdisk Kullanımı, Devamı

```

Command (m for help): p
Disk /dev/hda: 240 heads, 63 sectors, 2184 cylinders
Units = cylinders of 15120 * 512 bytes
Device Boot Start End Blocks Id System
/dev/hda1 1 14 105808+ 83 Linux
/dev/hda2 15 49 264600 82 Linux swap
/dev/hda3 50 70 158760 83 Linux
/dev/hda4 71 2184 15981840 5 Extended
/dev/hda5 71 209 1050808+ 83 Linux
/dev/hda6 210 348 1050808+ 83 Linux
/dev/hda7 349 626 2101648+ 83 Linux
/dev/hda8 627 904 2101648+ 83 Linux
/dev/hda9 905 2184 9676768+ 83 Linux
Command (m for help):

```

Bu örneğimizde, hda1'den hda3'e kadar birincil bölümleri görmekteyiz. hda4 ise hda5'ten hda9'a kadar mantıksal sürücülerini içeren uzatılmış bir bölümdür. O halde demek ki hda4 bölümünü tek başına bir dosya

sistemi barındıracak şekilde kullanamıyoruz. Görüldüğü gibi bu uzatılmış bölüm hda5'ten hda9'a kadar olan bölümler için bir taşıyıcı vazifesi görmektedir. Ayrıca görüldüğü gibi her bölüm bir "id" ya da "bölüm tipi" denilen bir değere sahiptir. Yeni bir bölüm yarattığınız zaman bu "bölüm tipi" nin düzgün ayarlandığından emin olmanız gerekmektedir. Linux dosya sistemi barındıracak olan bölümler için bölüm tipi olarak '83' değeri doğru bir seçim olacaktır. Linux swap dosya sistemini barındıracak bölümler için de bu değer '82' olmalıdır.

fdisk komutunda t parametresini kullanarak bölüm tipini ayarlayabilirsiniz. Linux çekirdeği açılış sırasında disk üzerindeki dosya sistemlerini ve swap aygıtlarını otomatik tanımak için bölüm tipi ayarlarını kullanmaktadır.

## 16.5 Fdisk ve Ötesi

fdisk ile yapabileceğimiz yeni disk yaratmak (n komutu ile), değişiklikleri diske yazmak (w komutu ile) ve daha birçok işlemin olduğunu söyleyebiliriz. Hatırlarsanız m yazarak yardım alabiliyorduk. Eğer fdisk kullanımını konusunda tecrübesizseniz size, içerisinde önemli sayılabilecek bilgi içermeyen ayrı bir disk üzerinde bu programı kullanarak kavramaya çalışmanızı öneririz. Bir defa bölümleri yaratıp diske yazdığımızda, yeni bölüm blok aygıtlarınız kullanıma hazır demektir. Bu yeni blok aygıtları yeni linux dosya sistemlerini saklamak amacıyla kullanacağız.

## 16.6 Dosya Sistemlerini Yaratmak

Bir blok aygıt dosyaları barındırmak amacıyla kullanılmadan önce, bu aygıt üzerinde yeni bir dosya sistemi yaratmamız gerekmektedir. Bu işlem için "mkfs" komutunu kullanırız. Kullanacağımız mkfs, yaratmak istediğimiz dosya sisteminin tipine göre değişmektedir. Bu örnekte, boş ve kullanılmayan blok aygıt olan /dev/hda8 üzerinde ext2 dosya sistemi yaratmak için mke2fs kullanıyoruz:

```
# mke2fs /dev/hda8
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
354112 inodes, 706852 blocks
35342 blocks (5.00%) reserved for the super user
First data block=0
22 block groups
32768 blocks per group, 32768 fragments per group
16096 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 30 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

Yukarıdaki örnekte az önce de söylediğimiz gibi mke2fs komutu /dev/hda8 üzerinde ext2 dosya sistemi yarattı.

## 16.7 Dosya sistemlerini mount etmek

Bir dosya sistemini oluşturduktan sonra onu mount edebiliriz.

```
# mount /dev/hda8 /mnt/usrDeneme
```

Bir dosya sistemini mount etmek için, disk bölümü blok aygıtını ilk argüman olarak ve bir mount (bağlama) noktasını ikinci argüman olarak belirtmek gerekir. Yeni dosya sistemi artık sözkonusu bağlantı noktasından erişilebilir olacaktır. Bu aynı zamanda /mnt dizini altındaki dosyaları saklama etkisi de yapabilir. Sözkonusu dosya sistemi umount işlemine tabi tutulduğunda artık buradaki eski dosyalar görünmezler. umount komutunu kullandıktan sonra /mnt altındaki yeni bağlanma noktasına eklenen yeni dosya ve dizinler artık yeni dosya sisteminde yer alır.

## 16.8 Takas bölümü oluşturmak ve kullanmak

Takas aygıtı olarak kullanılmak üzere bir disk bölümü oluşturduysak, bunu mkswap komutu ile düzenleyebiliriz. Doğal olarak bu komutun argümanı bölüm blok aygıtı olacaktır.

Not: Burada mkswap PATH değişkeninin gösterdiği dizlerin altında öntanımlı olarak yer almayabilir. Bunun için /sbin dizini de PATH değişkenine eklemek ya da komutu /sbin/mkswap şeklinde kullanmak gerekecektir.

```
# mkswap /dev/hdc6
```

Geleneksel dosya sistemlerinin aksine takas bölümleri mount edilmez. Bunun yerine swapon komutu ile aktif hale getirilirler.

```
# swapon /dev/hdc6
```

Normalde, Linux sisteminizin başlangıç betikleri, otomatik olarak takas bölümlerini etkin hale getirecektir. Bu yüzden swapon komutu takas bölümlerini o an kullanmak istediğinizde kullanılır. Mevcut takas aygıtlarını görmek için, cat /proc/swaps komutunu vermeniz gerekir.

## 16.9 Mount edilmiş dosya sistemlerini görmek

Hangi dosya sistemlerinin mount edildiğini görmek için mount komutunu tek başına çalıştırın.

```
$ mount
/dev/hda7 on / type xfs (rw)
none on /proc type proc (rw)
none on /proc/bus/usb type usbdevfs (rw)
none on /dev type devfs (rw)
none on /dev/pts type devpts (rw,mode=0620)
none on /mnt/cdrom type supermount
(ro,dev=/dev/hdc,fs=auto,--,iocharset=iso8859-1,
codepage=850,umask=0)
none on /mnt/floppy type supermount
(rw,sync,dev=/dev/fd0,fs=auto,--,iocharset=iso8859-1,
codepage=850,umask=0)
/dev/hda5 on /mnt/win_d type vfat
(rw,iocharset=iso8859-1,codepage=850,umask=0)
automount(pid1505) on /misc type autofs
(rw,fd=7,pgrp=1505,minproto=2,maxproto=4)
automount(pid1526) on /net type autofs
(rw,fd=7,pgrp=1526,minproto=2,maxproto=4)
/dev/hda1 on /mnt/win_c type ntfs (ro,iocharset=iso8859-1,umask=0)
```

Benzer bilgilere cat /proc/mounts yazarak da erişmeniz mümkün. Bazı Linux sistemlerinde devfs sistemi kullanıldığından, bu komutun çıktısının ilk satırında "root" bölüm blok aygıtının uzun bir yolu bulunmaktadır. "Root" dosya sistemi, kernel tarafından açılış sırasında otomatik olarak mount edilmektedir. Devfs aygıt-yönetim dosya sistemini kullanan yeni sistemlerde /dev altında artık bölüm ve disk blok aygıtlarının eskiden kullanılan isimleri bulunmamaktadır. Mesela /dev/ide/host0/bus1/target0/lun0/part7 /dev/hdc7 nin asıl adıdır ve /dev/hdc7 asıl blok aygıtına bir sembolik linktir. Sisteminizin devfs kullanıp kullanmadığını /dev/.devfsd dosyasının olup olmadığına bakarak görebilirsiniz. Eğer varsa devfs aktif demektir.

## 16.10 Mount Seçenekleri

mount seçeneklerini kullanarak mount edilecek dosya sisteminin çeşitli özelliklerini özelleştirmek mümkündür. Mesela "ro" seçeneğini kullanarak bir dosya sistemini sadece okunabilir (read only) halde yaratabilirsiniz.

```
# mount /dev/hda8 /mnt/usrDeneme -o ro
```

/dev/hda8 in /mnt altına bu seçenek ile mount edilmesi sonucunda /mnt altında hiç bir dosya değiştiremez, sadece okunabilir. Eğer dosya sisteminiz zaten okuma/yazma modunda mount edilmişse ve bunu sadece okunabilir moda değiştirmek istiyorsanız sistemi önce umount sonra da tekrar mount etmek yerine "remount" seçeneğini kullanabilirsiniz.



```
# mount /mnt/usrDeneme -o remount,ro
```

Dikkat ediniz burada bölüm blok aygıtı belirtmedik. Çünkü dosya sistemi zaten mount edilmiş haldedir ve mount komutu /mnt/usrDeneme nin /dev/hda8 ile ilişkili olduğunu zaten bilmektedir. Dosya sistemini tekrar yazılabilir hale getirmek için tekrar remount kullanıyoruz:

```
# mount /mnt/usrDeneme -o remount,rw
```

Şunu belirtmek gerekir ki eğer herhangi bir süreç /mnt altındaki dosya ya da klasörlerden bir tanesini kullanıyorsa remount komutları düzgün olarak çalışmayacaktır. Yani bir mount edilen bir dosya içerisinde sadece ls komutunu çalıştırıyor olsak bile bu durumda iken umount çalışmayacaktır:

```
# cd usrDeneme/
# ls
buBolumHakkinda.txt deneme/
# umount /mnt/usrDeneme/
umount: /mnt/usrDeneme: device is busy
```

Linux altında tüm mount seçenekleri hakkında ayrıntılı bilgi almak istiyorsanız man mount yazmanız yeterlidir.

## 16.11 fstab, Giriş

Şu ana kadar el yordamıyla mount işlemin nasıl yapabileceğimizi gördük. Genel olarak belli bir kurala göre düzenli bir şekilde mount etmeniz gereken bir dosya sistemi olduğunda el yordamıyla mount etmek biraz karmaşık ve zor bir işlem haline gelecektir. Ayrıca bazı dosya sistemleri için de, mesela /var dosya sistemi, zaten el yordamıyla mount işlemi imkansızdır. Sonuçta el yordamıyla mount yerine eğer varsa böyle dosya sistemlerini açılış sırasında otomatik bir şekilde mount etmek gerekir. Bunu sisteme yaptırmak için yapmamız gereken tek şey /etc/fstab dosyasına uygun satırları eklemek olacaktır. Bir dosya sisteminin açılış sırasında otomatik olarak mount edilmesini tercih etmeseniz bile, /etc/fstab dosyasına yazılacak satırlar ile bu işlemin çok kolay bir şekilde yapılabileceğini söylemeliyiz.

## 16.12 Örnek bir fstab Dosyası

Gelin örnek bir /etc/fstab dosyasını inceleyelim:

```
# cat /etc/fstab
# <fs> <mountpoint> <type> <opts> <dump/pass>
/dev/hda7 / xfs defaults 1 1
none /dev/pts devpts mode=0620 0 0
none /mnt/cdrom supermount dev=/dev/hdc,fs=auto,ro,--,iocharset=iso8859-1,
    codepage=850,umask=0 0 0
none /mnt/floppy supermount dev=/dev/fd0,fs=auto,--,iocharset=iso8859-1,
    sync,codepage=850,umask=0 0 0
/dev/hda8 /mnt/usrDeneme ext3 noauto,noatime 1 2
/dev/hda1 /mnt/win_c ntfs iocharset=iso8859-1,ro,umask=0 0 0
/dev/hda5 /mnt/win_d vfat iocharset=iso8859-1,codepage=850,umask=0 0 0
/dev/hda6 swap swap defaults 0 0
# /proc should always be enabled
proc /proc proc defaults 0 0
```

Yukarıdaki /etc/fstab dosyasının içinde aktif haldeki (başında # işareti olmayan) her satır bir bölüm blok aygıtı, bir bağlanma noktası, bir dosya sistemi çeşidi, dosya sistemi bağlanırken kullanılan dosya sistemi seçenekleri ve iki adet sayısal alan içermektedir.

Birinci sayısal alan dump back up komutuna yedeklenecek dosya sistemlerini bildirir. Eğer sisteminizi yedeklemeyi düşünmüyorsanız tabi ki bu sayıyı göz ardı edebilirsiniz. Son sayı ise dosya sistemi bütünlüğünü kontrol eden fsck tarafından kullanılmaktadır ve bu programa açılış sırasında dosya sisteminin hangi sırada kontrol edilmesi gerektiğini anlatır. fsck birkaç noktada tekrar değineceğiz.

/dev/hda8 yazılı satırı incelediğimizde ext3 dosya sistemine sahip olduğunu ve /mnt/usrDeneme altından mount edildiğini görmekteyiz. Aynı satırda /dev/hda8 in mount seçeneklerine baktığımızda şunları görmekteyiz: noauto, açılış sırasında bu dosya sisteminin otomatik olarak mount edilmemesi gerektiğini söyler. Bu seçenek olmadan /dev/hda8 açılış sırasında /mnt/usrDeneme altına otomatik olarak mount edilecektir.

noatime seçeneği ile de disk üzerindeki atime (son erişim zamanı) kaydının tutulmaması sağlanmaktadır. Bu bilgi genelde gerekli değildir ve bu özelliğin kaldırılması dosya sistemi üzerinde performans artışına neden olmaktadır. nodiratime mount seçeneği ile de izinler üzerindeki atime güncellemesini de iptal edebilirsiniz.

### 16.13 Örnek bir fstab, Devamı

```
# <fs> <mountpoint> <type> <opts> <dump/pass>
/dev/hda7 / xfs defaults 1 1
none /dev/pts devpts mode=0620 0 0
none /mnt/cdrom supermount dev=/dev/hdc,fs=auto,ro,--,iocharset=iso8859-1,
    codepage=850,umask=0 0 0
none /mnt/floppy supermount dev=/dev/fd0,fs=auto,--,iocharset=iso8859-1,
    sync,codepage=850,umask=0 0 0
/dev/hda8 /mnt/usbDeneme ext3 noauto,noatime 1 2
/dev/hda1 /mnt/win_c ntfs iocharset=iso8859-1,ro,umask=0 0 0
/dev/hda5 /mnt/win_d vfat iocharset=iso8859-1,codepage=850,umask=0 0 0
/dev/hda6 swap swap defaults 0 0
# /proc should always be enabled
proc /proc proc defaults 0 0
```

Şimdi de /proc yazılı satıra bir göz atalım. Buradaki default seçeneğine dikkat edin. Eğer bir dosya sistemin standart mount seçenekleri ile mount etmek isterseniz defaults seçeneğini kullanmalısınız. /etc/fstab dosyasında birçok alan yer aldığı için seçenek alanımızı boş bırakamayız. Ayrıca /dev/hda6 satırına da göz atalım. Bu satır /dev/hda6'yı swap aygıt olarak belirlemektedir. Swap aygıtlar diğer dosya sistemleri gibi mount edilmediğinden, mount noktası kısmına swap yazıldığını görmekteyiz. /etc/fstab dosyası sayesinde /dev/hda6 swap aygıtımız açılış sırasında otomatik olarak aktif hale gelecektir.

Aşağıdaki gibi yazmak yerine, yukarıda olduğu gibi bir /etc/fstab içerisinde /dev/cdrom kullanarak CD-ROM sürücüyü mount etmek daha kolay hale gelecektir.

```
# mount -t iso9660 /dev/cdrom /mnt/cdrom -o ro
```

artık şöyle yazılabilir:

```
# mount /dev/cdrom
```

/etc/fstab dosyasını kullanmak "user" seçeneğinin vereceği avantajdan yararlanmamızı sağlar. "user" mount seçeneği, sisteme ilgili dosya sisteminin herhangi bir kullanıcı tarafından mount edilebileceğini bildirir. Ayrıca users ile de tüm kullanıcıların ilgili aygıtı mount edebileceğini belirleyebiliriz. CD-ROM gibi taşınabilir aygıtlar için bu özellik çok kullanışlıdır. Bu mount seçeneği olmasaydı sadece root kullanıcı CD-ROM sürücüyü mount edebilirdi.

### 16.14 Dosya sistemlerini Unmount Etmek

Genelde sistem yeniden başlatıldığında ya da kapatılırken mount edilmiş tüm dosya sistemleri otomatik olarak unmount edilir. Bir dosya sistemi unmount edileceği zaman geçici bellekte bekleyen herhangi bir dosya sistemi bilgisi diske yazılır.

Ayrıca el yordamı ile de istenirse unmount işlemi gerçekleştirilebilir. Bir dosya sistemini unmount etmeden önce bu sistem üzerinde herhangi bir süreç tarafında açılmış ya da kullanılan dosya olmadığından emin olmalıyız. Bu durum sağlandığında aygıt adını ya da bağlama noktasını parametre olarak girerek unmount komutunu kullanabiliriz.

```
# umount /mnt
```

veya

```
# umount /dev/hdc6
```

umount sonrasında, /mnt altında önceden mount edilmiş sistemin kapsadığı dosyalar yeniden belirecektir.

## 16.15 fsck'ya Giriş

Eğer sistemimiz bir şekilde kilitlenir ya da bozulursa, sistem düzgün bir şekilde dosya sistemlerini unmount etmeyi başaramayacaktır. Bu durumda dosya sistemi güvensiz bir durum içerisine girmiş olur. Sistem tekrar başlatıldığında fsck bu dosya sistemlerinin düzgün bir şekilde unmount edilmediğini saptayacak ve /etc/fstab içerisinde listelenen dosya sistemlerinin bütünlüğünü kontrol etmek isteyecektir.

Şunu önemle belirtmek gerekir ki, fsck tarafından kontrol edilecek bir dosya sisteminin /etc/fstab dosyasında yer alan satırında, pass alanında (son alan) sıfır olmayan bir sayı olmalıdır. Tipik olarak kök dosya sisteminin passno değeri 1 sayıdır ki bu kök dizinin ilk olarak kontrol edilecek dosya sistemi olacağını belirler. Kontrol edilecek diğer tüm dosya sistemlerinin de passno alanına benzer şekilde 2 ya da daha büyük bir değer verilmelidir.

Bazen sistemin yeniden başlatılması sırasında fsck'nın kısmen zarar görmüş bir diski tam olarak onarmadığını görürüz. Böyle durumlarda sistemi tek kullanıcı çalışır hale getirerek fsck'yı el yordamıyla çalıştırmak gerekir. Bu kullanımda kontrol edilecek blok aygıtı fsck'ya parametre olarak göndeririz. fsck dosya sistemi tamiratını gerçekleştirirken kısmi dosya sistemi zararlarını düzeltip düzeltmeyeceğini soracaktır. Genel olarak bu sorulara y (yes) cevabı vererek fsck'nın sistemi onarmasını sağlamalıyız.

## 16.16 fsck ile İlgili Problemler

Disk üzerinde yer alan bilgilerin bütünlüğünün kontrol edilebilmesi için fsck bütün diski taramaktadır. Bu problemlerden bir tanesidir. Özellikle çok büyük dosya sistemlerinde fsck'nın çalışmasının bir saatten daha da uzun sürmesi beklenen bir durumdur.

Bu problemi çözmek için journaling denilen yeni bir dosya sistemi geliştirilmiştir. Journaling dosya sistemi disk üzerindeki dosya sistemi içerisinde yapılan değişiklikleri disk üzerinde günlük dosyasına kaydeder(log file). Sistemde meydana gelen bir arıza durumunda dosya sistemi sürücüsü bu günlük dosyasını araştırır. Bu dosya son değişikliklerin tam bilgisini tuttuğu için sadece bu kısımların hatalar için kontrol edilmesi yeterlidir. Bu yeni tasarım sayesinde dosya sisteminin büyüklüğüne bağlı olmaksızın, fsck'nın kontrol etme işlemi sadece birkaç saniye sürmektedir. Bu yüzden Linux camiasında journaling dosya sistemlerinin popüleritesi artar hale gelmiştir.

Şimdi gelin Linux'un değişik dosya sistemlerine bir göz atalım.

## 16.17 ext2 Dosya Sistemi

Uzun yıllardır ext2 dosya sistemi linux için standart haline gelmiş durumdadır. Bu dosya sistemi birçok uygulama için iyi bir performans sergilese de yukarıda bahsettiğimiz journaling yeteneğine sahip değildir. fsck'nın çalışması çok uzun zaman alacağından büyük dosya sistemleri için ext2'nin bu özelliği uygunsuzluk yaratır. Buna ek olarak her ext2 dosya sisteminin taşıyabileceği inode sayısı sınırlı olduğundan, ext2 dosya sisteminin bazı yapısal kısıtlamaları vardır. Tüm bu bilgilerle ext2 için oldukça sağlam ve verimli ancak non-journalised bir dosya sistemidir diyebiliriz.

- Yer aldığı çekirdekler: 2.0+
- journaling yeteneği: yok
- mkfs komutu: mke2fs
- mkfs örneği: mke2fs /dev/hdc7
- ilgili komutlar: debugfs, tune2fs, chattr
- performans ile ilgili komut seçenekleri: noatime, nodiratime

## 16.18 ext3 Dosya Sistemi

Bu dosya sistemi diski ext2 ile aynı şekilde biçimlendirir ancak journaling yeteneğini de kazandırır. Aslında tüm Linux dosya sistemleri içerisinde ext3 en kapsamlı journaling desteğine sahip olanıdır. Sadece döküman özelliklerini (metadata) kaydetmek yerine ayrıca tüm döküman özellikleri artı bilgilerin de journaling kaydını tutar. Bu özel journaling yeteneği sadece fsck'nın çalışmasını kısaltmakla kalmaz ayrıca bilgi bütünlüğünün sağlanmasına da yardımcı olmaktadır. Bu yüzden bilgi bütünlüğünün en önemli önceliğe sahip olduğu sistemler için en iyi dosya sistemi ext3 olacaktır. Fakat bazı durumlar için bu bilgi bütünlüğü performansı etkilemektedir. Buna ek olarak ext3 dosya sistemi diski ext2 ile aynı şekilde

biçimlendirdiği için yukarıda bahsedilen ext2 sınırlandırmalarından nasibini almaktadır. Eğer oldukça sağlam genel amaçlı ve journalised yeteneğine sahip bir dosya sistemine ihtiyaç duyuyorsanız ext3 iyi bir seçim olacaktır.

- Yer aldığı çekirdek: 2.4.16+
- journaling: metadata, ordered data writes, full metadata+data
- mkfs komutu: mke2fs -j
- mkfs örneği: mke2fs -j /dev/hdc7
- ilgili komutlar: debugfs, tune2fs, chattr
- performans ile ilgili mount seçenekleri: noatime, nodiratime
- diğer mount seçenekleri:
  - \* data=writeback (disable journaling)
  - \* data=ordered (the default, metadata journaling and data is written out to disk with metadata)
  - \* data=journal (bilgi ve dosya özellikleri bütünlüğü için tam data journaling. Yazma performansını yarıya düşürür)
- ext3 kaynakları:
  - \* Andrew Morton's ext3 page
  - \* Andrew Morton's excellent ext3 usage documentation (recommended)
  - \* Advanced filesystem implementor's guide, part 7: Introducing ext3
  - \* Advanced filesystem implementor's guide, part 8: Surprises in ext3

### 16.19 ReiserFS Dosya Sistemi

ReiserFS dosya sistemi küçük dosya performansı, çok iyi genel performans, ve kolay ölçeklendirilebilirlik (bölümlenebilir) amacıyla tasarlanmış ve diğerlerine nazaran oldukça yeni bir dosya sistemidir. ReiserFS dosya sistemi uzun süren fsck'dan kurtulmak amacıyla dosya özellikleri (metadata) günlüğü kullanır. Ancak günlük gerçekleştirmesi (implementation) sistem kilitlenmesi durumunda son değişikliklerin yapıldığı bilginin bozulmasına izin verir. Genel olarak ReiserFS çok iyi bir performans sergilemektedir. Ancak bazı dosya sistemi yükleri için garip performans davranışları sergilemektedir. Buna ek olarak ReiserFS'in fsck aracı henüz başlangıç aşamasında olduğundan bozulmuş bir dosya sisteminden bilgi kurtarmakta zorlanabilirsiniz. ReiserFS'in yeni olmasından kaynaklanan bu konular yüzünden bu dosya sisteminin gelişimi halen devam etmektedir. Hızı ve ölçeklendirilebilirliği yüzünden birçok kişi tarafından tercih edilmektedir.

- Yer aldığı çekirdekler: 2.4.0+ (2.4.16+ önerilen)
- günlük tutma(journaling): dosya özellikleri (metadata)
- mkfs komutu: mkreiserfs
- mkfs örneği: mkreiserfs /dev/hdc7
- performans ile ilgili mount seçenekleri: noatime, nodiratime, notail
- ReiserFS Kaynakları:
  - \* The home of ReiserFS
  - \* Advanced filesystem implementor's guide, Part 1: Journalling and ReiserFS
  - \* Advanced filesystem implementor's guide, Part 2: Using ReiserFS and Linux 2.4

## 16.20 XFS Dosya Sistemi

XFS dosya sistemi SGI tarafından Linux'a port edilmiş, günlük tutan bir dosya sistemidir. XFS henüz mevcut çekirdeğe yerleştirilmemiştir. XFS hakkında geniş bilgi almak için <http://oss.sgi.com/projects/xfs> sayfasını ziyaret edebilirsiniz. XFS'e giriş için Advanced filesystem implementor's guide, Part 9: Introducing XFS dokümanını okuyabilirsiniz.

## 16.21 JFS Dosya Sistemi

JFS yüksek performanslı günlük tutan ve Linux'e IBM tarafından port edilmiş bir dosya sistemidir. JFS IBM enterprise sunucuları tarafından kullanılır ve yüksek performans için dizayn edilmiştir. JFS de mevcut kernel içerisine henüz yerleştirilmemiştir. JFS projesi web sitesinde bu dosya sistemi hakkında daha ayrıntılı bilgi bulabilirsiniz.

## 16.22 VFAT Dosya Sistemi

VFAT dosya sistemi Linux dosyalarını depolayabileceğiniz gerçek bir dosya sistemi değildir. Daha ziyade DOS ve Windows FAT tabanlı işletim sistemleri ile dosya paylaşımına ve değişimine olanak sağlayan DOS uyumlu bir dosya sistemi sürücüsüdür. VFAT dosya sistemi sürücüsü standart Linux çekirdeği içerisinde yer almaktadır.

# 17 Sistemi Başlatmak

Bu bölümde Linux sistem açılış işleminden bahsedilmektedir. Burada boot yükleyicisi kavramını, açılışta çekirdek ayarlarını nasıl yapacağımızı ve açılış log dosyalarını nasıl inceleyeceğimizi göreceğiz.

## 17.1 MBR

Sistemin yüklendiği dağıtıma bakmaksızın açılış işlemi tüm Linux makinalarda birbirine benzemektedir. Aşağıdaki örnek hardiske bir göz atalım:

```
+-----+
|      MBR      |
+-----+
| Partition 1:  |
| Linux root (/)|
| containing   |
| kernel and   |
| system.      |
+-----+
| Partition 2:  |
| Linux swap    |
+-----+
| Partition 3:  |
| Windows 3.0   |
| (last booted |
| in 1992)      |
+-----+
```

İlk olarak bilgisayarın BIOS'u harddiskinizin ilk birkaç sektörünü okumaktadır. Bu sektörler içerisinde "Master Boot Record" ya da "MBR" denilen çok küçük bir program yer almaktadır. MBR Linux çekirdeğinin (yukarıdaki örnekte bölüm 1) yerinin bilgisini saklamaktadır. Böylelikle çekirdeği belleğe yükler ve başlatır.

## 17.2 Çekirdek Açılış İşlemi

Bundan sonra göreceğimiz (her ne kadar çok hızlı bir şekilde geçse de ) aşağıdakine benzer bir satır olacaktır:

```
Linux version 2.4.16 (root@time.flatmonk.org)
(gcc version 2.95.3 20010315 (release))
```

Bu satır çekirdek çalışmaya başladığı zaman ekrana bastığı ilk satırdır. İlk kısım çekirdeğin sürümünü, bundan sonra gelen satır çekirdeği yazan kişinin kimliğini (genellikle root), sonraki kısım derleyiciyi ve sonra da yazıldığı saat dilimini (timestamp) belirtmektedir.

Bundan sonra gelen satırlarda ise sistemde yer alan donanıma bağlı olarak işlemci, disk kontrol edici, PCI veriyolu, diskler, ses kartları, seri portlar, disket sürücüler, USB aygıtlar, ağ adaptörleri ve muhtemel diğer donanımlarla ilgili bunların durumlarını gösteren raporlar yer alacaktır.

### 17.3 /sbin/init Programı

Çekirdek yüklemesini tamamladığı zaman init denilen bir program çağırılmaktadır. Bu program sistem kapatılıncaya kadar çalışmaya devam eder. Aşağıda da göreceğiniz gibi programa verilen process ID her zaman 1 değerini alır.

```
$ ps --pid 1
PID TTY TIME CMD
1 ? 00:00:04 init.system
```

init programı bundan sonra birtakım betikler (script) çalıştırarak dağıtımın geri kalan kısmını da başlatır. Bu betikler geleneksel olarak /etc/rc.d/init.d dosyasında yer alır ve sistemin sunucu adını belirleyen, dosya sistemini hatalara karşı denetleyen, genel dosya sistemlerini mount eden, ağ uygulamalarını başlatan, yazıcı servislerini başlatan ve bunun gibi işlemleri gerçekleştirecek bazı servisleri çalıştırmalar. Betiklerin çalışması tamamlandığında, init getty adında ve giriş ekranını getiren bir program çalıştırır.

### 17.4 LILO'yu İnceleyelim

Şu ana kadar açılış işlemine genel bir bakış yapmış olduk. Şimdi ilk kısmı, MBR ve çekirdeğin yüklenmesini biraz daha ayrıntılı inceleyelim. MBR'nin bakımı "boot loader" sorumluluğundadır. x86 tabanlı Linux sistemleri için en popüler iki boot loader LILO (Linux LOader) ve GRUB (GRand Unified Bootloader) dir. Bu ikisi içerisinde LILO daha eski ve daha çok bilinen boot loader dir. Sistemde LILO'nun hazır olduğu zaman bu durum açılışta "LILO boot:" şeklinde bildirilir. Şunu belirtmek gerekir ki, genelde sistem duraklamadan konfigürasyon ayarları yapıldığından, bu satırı görebilmek için shift tuşuna basmanız gerekebilir.

LILO komut satırında bundan daha fazla birşey yazmayacaktır. Ancak eğer < tab > tuşuna basarsanız karşınıza mevcut çekirdeklerin ya da diğer işletim sistemlerinin yer aldığı bir liste gelecektir. Bunlardan bir tanesini yazarak ya da < enter > tuşuna bakarak istediğinizi başlatabilirsiniz. Alternatif olarak sadece < enter > tuşuna basarsınız ve öntanımlı olarak listedeki ilk seçenek başlatılmış olur.

### 17.5 LILO Kullanımı

Nadir olarak açılış sırasında çekirdeğe bir seçenek göndermek isteyebiliriz. En çok bilinen bu seçeneklerden bazıları root=alternatif bir root dosya sistemi belirlemek için, init=alternatif bir init programı belirlemek için (mesela konfigürasyonu bozulmuş bir sistem için init=/bin/sh) ve mem=sistemdeki hafıza miktarını belirlemek için (örneğin Linux'un sadece 128MB'yi otomatik olarak algıladığı durum için mem=512M). Bu seçenekleri LILO açılış komutuna gönderebilirsiniz.

```
LILO boot: linux root=/dev/hdb2 init=/bin/sh mem=512M
```

Eğer düzenli olarak komut satırı seçenekleri kullanmak istiyorsanız, bunları /etc/lilo.conf dosyasına yazabilirsiniz. Bu dosyanın formatı lilo.conf (5) kılavuz sayfasında açıklanmaktadır.

### 17.6 LILO Hakkında Önemli Not

GRUB'ı incelemeye geçmeden önce LILO hakkında bilinmesi gereken birşey daha olduğunu söyleyelim. /etc/lilo.conf dosyasında değişiklik yaptığınız zaman ya da yeni bir çekirdek yüklediğiniz zaman lilo'yu çalıştırmanız gerekmektedir. LILO programı MBR'yi yeniden yazarak yeni çekirdeğin tam yerini de içerecek şekilde yaptığınız değişiklikleri geçerli hale getirir.

Buradaki örnekte LILO'yu açıklamalı (verbose) mod ile çalıştırıyoruz:

```
# lilo -v
LIL0 version 21.4-4, Copyright (C) 1992-1998 Werner Almesberger
'lba32' extensions Copyright (C) 1999,2000 John Coffman
Reading boot sector from /dev/hda
Merging with /boot/boot.b
Mapping message file /boot/message
Boot image: /boot/vmlinuz-2.2.16-22
Added linux *
/boot/boot.0300 exists - no backup copy made.
Writing boot sector.
```

## 17.7 GRUB'ı İnceleyelim

GRUB boot loader LILO'dan sonra gelen ve gelecek nesil boot loader olarak görebileceğimiz bir uygulamadır. LILO'nun komut satırı sistemi yerine menü arayüzü sağlamaktadır. Ancak sistem yöneticileri için bu farklar çok önemli değildir. GRUB LILO'nun destek verdiğinden daha fazla işletim sistemine destek vermektedir. Açılış menüsünde şifre tabanlı bazı güvenlik ayarları sunmaktadır ve yönetimi de kolaydır.

GRUB genellikle grub-install komutu ile yüklenmektedir. Bir kez kurulduğunda, GRUB'ın menüsü /boot/grub/menu.lst dosyası düzenlenerek yönetilebilir. GRUB kurulumu ve yönetimi bu dökümanın faaliyet alanında olmadığından bu işlemler için GRUB info sayfalarını okumanızı öneririz.

## 17.8 GRUB Kullanımı

Çekirdeğe parametre vermek için boot menüsünde iken 'e' tuşuna basmanız gerekir. Bu size yüklemek istediğiniz çekirdek adını girmenize ya da çekirdeğe gönderilecek parametreleri girmenize olanak sağlar. Düzenleme işlemini bitirdiğinizde, < enter > tuşuna ve sonra da b tuşuna basarak yaptığınız yeni değişiklikler ile birlikte sistemi başlatabilirsiniz.

LILO ile GRUB arasındaki bahsedilmesi gereken önemli bir değişiklik ise GRUB'un yeni bir çekirdek yüklendiğinde ya da kendisi konfigüre edildiğinde tekrar kurulmaya ihtiyaç duymamasıdır. Bunun nedeni GRUB'ın Linux dosya sistemiyle interaktif çalışması, LILO'nun ise sadece yüklenecek olan çekirdeğin yerini tutmasıdır. Bu tek özellik yeni bir çekirdek yükledikten sonra lilo yazmayı unutan sistem yöneticilerinin boşuna iş yapmış hissiyle sinirlenmelerine engel olmaktadır

## 17.9 dmesg

Çekirdek ve init betiklerinden (script) gelen açılış mesajları genelde çok çabuk geçmektedir. Bir hata mesajını farketmeniz bile siz daha mesajı algılayamadan o kaybolacaktır. Bu durumda neyin yanlış gittiğini anlayabilmek için (ayrıca nasıl düzelteceğinize dair fikir sahibi olmak için ) açılış işlemi tamamlandıktan sonra bakabileceğiniz iki yer vardır.

Eğer hata çekirdek yüklenirken ya da donanım aygıtları araştırılırken meydana gelmiş ise dmesg komutu ile çekirdeğin log dosyasının bir kopyasını ekrana getirebilirsiniz.

```
# dmesg | head -1
Linux version 2.4.16 (root@time.flatmonk.org)
(gcc version 2.95.3 20010315 (release))
```

Burada gördüğünüz gibi çekirdeğin yüklenmesi sırasında yazdığı ilk satır ortaya çıktı. Gerçekten de eğer dmesg komutunu çıktısını bir sayfa görüntüleyiciye koyarsanız çekirdeğin açılışta yazdığı bütün mesajlar ve aynı anda çekirdeğin konsola yazdığı bütün mesajları görebilirsiniz.

## 17.10 /var/log/messages

Bilgi için bakılacak ikinci yer ise /var/log/messages dosyası olacaktır. Bu dosya kütüphanelerden, diğer arka plan süreçlerden ve çekirdekten gelen bilgileri giriş olarak alan "syslog" arka plan süreci (daemon) tarafından kaydedilmektedir. messages dosyasındaki her satır tarihlendirilmiştir. Bu dosya açılışta init betiklerinin çalışması sırasında meydana gelen hataları incelemek için iyi bir dosyadır. Örneğin kernel'dan gelen son birkaç mesajı görmek için:

```
# grep kernel /var/log/messages | tail -3
May 1 21:17:29 localhost kernel: hdc: ATAPI 52X CD-ROM drive, 128kB Cache, DMA
May 1 21:17:29 localhost kernel: Uniform CD-ROM driver Revision: 3.12
May 1 21:17:29 localhost kernel: SCSI subsystem driver Revision: 1.00
```

### 17.11 Tek-kullanıcı Modu

Çekirdeğe açılış sırasında parametre gönderilebileceğini artık biliyoruz. En sık kullanılan parametrelere birtanesi sistemi single-user modunda açmaya yarayan s parametresidir. Bu mod sadece root dosya sisteminin mount edilmesini, en az sayıda init betiği (script) çalışmasını ve bir login ekranından ziyade bir kabuk başlatılmasını sağlar. Ayrıca ağ ayarları konfigüre edilmez ve böylece harici etkilerin işinizi engellemesi ihtimali ortadan kalkar.

### 17.12 Tek-kullanıcı (Single-user) Modunu Kullanmak

O halde böyle bir mod ile sistemde nasıl çalışacağız? Bu soruya cevap verebilmek için Linux ile Windows arasındaki çok önemli bir farka değinmek gerekiyor. Windows normalde konsolda yerleşmiş ve aynı anda tek bir kullanıcının kullanacağı şekilde tasarlanmıştır. Yani efektif olarak her zaman single-user modda çalışmaktadır. Diğer yandan Linux daha ziyade uzaktaki kullanıcılara ağ uygulamaları sunmak ya da kabuk ve X oturumları sağlamak amacıyla kullanılır. Durum böyleyken sistemde yapılacak olan dosya sistemi yaratma ve düzenleme işlemleri, sistemin CD'den güncellenmesi, kurulum ve yedekleme işlemleri ve buna benzer bakım işlemleri sırasında bu ekstra değişkenler tercih edilmeyecektir. Böyle durumlarda single-user moda geçmek gerekir.

### 17.13 Çalışma Seviyelerini Değiştirmek

Aslında single-user moda geçmek için sistemin yeniden başlatılması gerekmemektedir. init programı sistemin o andaki modunu ya da "çalışma seviyesini" yönetebilmektedir. Linux için standart çalışma seviyeleri aşağıdaki şekilde sıralandırılabilir:

- 0: Bilgisayarı kapat
- 1: ya da s: tek-kullanıcı modu
- 2: çok-kullanıcı, ağ yok
- 3: çok kullanıcı, text konsol
- 4: çok-kullanıcı, grafik konsol
- 5: 4 ile aynı
- 6: Bilgisayarı yeniden başlat

Bu çalışma seviyeleri dağıtıma göre değişebilir bu yüzden kullandığınız dağıtımın dökümanını incelemek gerekir. single-user moda geçmek için, init'e çalışma seviyesini değiştirmesi gerektiğini belirten telinit komutunu kullanmalısınız.

```
# telinit 1
```

Yukarıdaki tabloda da görebileceğiniz gibi aynı yöntemle sistemi yeniden başlatabilir ya da kapatabilirsiniz. telinit 0 sistemi kapatacak, telinit 6 ise yeniden başlatacaktır. telinit komutu ile çalışma seviyesinin değiştirmeye kalktığınızda init scriplerinden bazıları çalışmaya başlayacak belli servisleri açacak ya da kapatacaklardır.

### 17.14 Sistemi Düzgün Kapatmak

Dikkat ederseniz eğer sistemde o anda başka kullanıcılarda varsa sistemi bu şekilde kapatmak o kullanıcılar için çok zararlı olabilir. shutdown komutu bunun için çalışma seviyelerini daha düzenli bir şekilde değiştirecek bir metod sunar. kill komutunu diğer süreçlere sinyal gönderme özelliğine benzer bir şekilde shutdown komutu da sistemi kapatmak yeniden başlatmak ya da tek-kullanıcı moduna geçmek için kullanılabilir. Örneğin 5 dakika içinde tek kullanıcı moduna geçmek için:

```
# shutdown 5
Broadcast message from root (pts/2) (Tue Jan 15 19:40:02 2002):
The system is going DOWN to maintenance mode in 5 minutes!
```

Eğer Control+c tuşuna basarsanız tek-kullanıcı moduna geçmeyi sağlayan bu komutu kaldırabilirsiniz. Bu gördüğünüz mesaj sistemde o anda çalışan tüm kullanıcıların terminallerinde belirecektir. Böylece bu kullanıcılar yaptıkları işi kaydedecek ve düzgün bir şekilde sistemden çıkabilecek zamanı bulabileceklerdir. (tabi bu 5 dakikanın yeterli olup olmayacağı ayrı bir tartışma konusu.



## 17.15 Sistemi Aniden Kapatmak

Eğer sistemde çalışan tek kullanıcı siz iseniz belli bir süre belirtmek yerine "now" parametresini kullanabilirsiniz. Örneğin sistemi hemen o anda yeniden başlatmak için:

```
# shutdown -r now
```

Bu örnekte -r "reboot after shutdown" bir sonraki örnekte göreceğimiz -h "halt after shutdown" anlamına gelir. Bu durumda artık Control+c tuşuna basma şansınız olmayacaktır çünkü artık kapatma işlemine başlanmıştır. Son olarak bahsedeceğimiz -h parametresi ise şöyle kullanılır:

```
# shutdown -h 1
Broadcast message from root (pts/2) (Tue Jan 15 19:50:58 2002):
The system is going DOWN for system halt in 1 minute!
```

## 17.16 Öntanımlı Çalışma Seviyesi

Bu ana kadar gördüklerimizle Linux sistemi için init programının önemini kavradığımızı tahmin ediyorum. /etc/inittab dosyasını düzenleyerek init'i konfigüre edebilirsiniz. Bununla ilgili açıklama inittab(5) kılavuz sayfasında açıklanmaktadır. Biz bu dosyada sadece bir noktaya değineceğiz:

```
# grep ^id: /etc/inittab
id:3:initdefault:
```

Buna göre benim sistemimde çalışma seviyesi 3 öntanımlı çalışma seviyesidir. Eğer sisteminize açılır açılmaz grafik ekranda giriş yapmak istiyorsanız bu değeri değiştirebilirsiniz (genelde 4 ya da 5). Bunu yapabilmek için sadece dosyadaki ilgili satırı değiştirmeniz yeterlidir. Ancak unutmayınız ki eğer bu değeri geçersiz bir sayı ile değiştirirseniz daha önce anlattığımız init="/bin/sh" a başvurmanız gerekecektir.

## 18 Çalışma Seviyeleri

### 18.1 Tek-kullanıcı Modu

Bundan önceki bölümde de söylediğimiz gibi çekirdeğe açılış sırasında parametre gönderilebileceğini artık biliyoruz. En sık kullanılan parametrelerden bir tanesi sistemi single-user modunda açmaya yarayan "s" parametresidir. Bu mod sadece root dosya sisteminin mount edilmesini, en az sayıda init scripti çalışmasını ve bir login ekranından ziyade bir kabuk başlatılmasını sağlar. Ayrıca ağ ayarları konfigüre edilmez ve böylece harici etkilerin işinizi engellemesi ihtimali ortadan kalkar.

### 18.2 Çalışma Seviyeleri

Aslında single-user moda geçmek için sistemin yeniden başlatılması gerekmemektedir. init programı sistemin o andaki modunu ya da "çalışma seviyesini" yönetebilmektedir. Linux için standart çalışma seviyeleri aşağıdaki şekilde sıralandırılabilir:

- 0: Bilgisayarı kapat
- 1 ya da s: tek-kullanıcı modu
- 2: çok-kullanıcı, ağ yok
- 3: çok kullanıcı, text konsol
- 4: çok-kullanıcı, grafik konsol
- 5: 4 ile aynı
- 6: Bilgisayarı yeniden başlat

### 18.3 elinit Komutu

single-user moda geçmek için, init'e çalışma seviyesini değiştirmesi gerektiğini belirten telinit komutunu kullanmalısınız.

```
# telinit 1
```

Yukarıdaki tabloda da görebileceğiniz gibi aynı yöntemle sistemi yeniden başlatabilir ya da kapatabilirsiniz. telinit 0 sistemi kapatacak, telinit 6 ise yeniden başlatacaktır. telinit komutu ile çalışma seviyesinin değiştirmeye kalktığınızda init betiklerinden bazıları çalışmaya başlayacak belli servisleri açacak ya da kapatacaklardır.

### 18.4 Çalışma Seviyesi Etiketi

Dikkat ederseniz eğer sistemde o anda başka kullanıcılarda varsa sistemi bu şekilde kapatmak o kullanıcılar için çok zararlı olabilir. shutdown komutu bunun için çalışma seviyelerini daha düzenli bir şekilde değiştirecek bir metod sunar. kill komutunu diğer süreçlere sinyal gönderme özelliğine benzer bir şekilde shutdown komutu da sistemi kaptmak yeniden başlatmak ya da tek-kullanıcı moduna geçmek için kullanılabilir. Örneğin 5 dakika içinde tek kullanıcı moduna geçmek için:

```
# shutdown 5
Broadcast message from root (pts/2) (Tue Jan 15 19:40:02 2002):
The system is going DOWN to maintenance mode in 5 minutes!
```

Eğer Control+c tuşuna basarsanız tek-kullanıcı moduna geçmeyi sağlayan bu komutu kaldırabilirsiniz. Bu gördüğünüz mesaj sistemde o anda çalışan tüm kullanıcıların terminallerinde belirecektir. Böylece bu kullanıcılar yaptıkları işi kaydedecek ve düzgün bir şekilde sistemden çıkabilecek zamanı bulabileceklerdir. (tabi bu 5 dakikanın yeterli olup olmayacağı ayrı bir tartışma konusu.

### 18.5 "now" ve "halt"

Eğer sistemde çalışan tek kullanıcı siz iseniz belli bir süre belirtmek yerine "now" parametresini kullanabilirsiniz. Örneğin sistemi hemen o anda yeniden başlatmak için:

```
# shutdown -r now
```

Bu örnekte -r "reboot after shutdown" bir sonraki örnekte göreceğimiz -h "halt after shutdown" anlamına gelir. Bu durumda artık control+c tuşuna basma şansınız olmayacaktır çünkü artık kapatma işlemine başlanmıştır. Son olarak bahsedeceğimiz -h parametresi ise şöyle kullanılır:

```
# shutdown -h 1
Broadcast message from root (pts/2) (Tue Jan 15 19:50:58 2002):
The system is going DOWN for system halt in 1 minute!
```

### 18.6 init Konfigürasyonu

Bu ana gördüklerimizle Linux sistemi için init programının önemini kavradığımızı tahmin ediyorum. /etc/inittab dosyasını düzenleyerek init'i konfigüre edebilirsiniz. Bununla ilgili açıklama inittab(5) kılavuz sayfasında açıklanmaktadır. Biz bu dosyada sadece bir noktaya değineceğiz:

```
# grep ^id: /etc/inittab
id:3:initdefault:
```

Buna göre benim sistemimde çalışma seviyesi 3 öntanımlı çalışma seviyesidir. Eğer sisteminize açılır açılmaz grafik ekranda giriş yapmak istiyorsanız bu değeri değiştirebilirsiniz (genelde 4 ya da 5). Bunu yapabilmek için sadece dosyadaki ilgili satırı değiştirmeniz yeterlidir. Ancak unutmamalıyız ki eğer bu değeri geçersiz bir sayı ile değiştirirseniz daha önce anlattığımız init=/bin/sh' a başvurmanız gerekecektir.

## 19 Dosya Sistemi Kotaları

### 19.1 Kotalara Giriş

Kotalar size disk kullanımı için kullanıcı ya da grup olarak izlemeniz gereken yolu belirleyen bir Linux özelliğidir. Bir kullanıcı ya da grubun diski haksız bir biçimde kullanmasını ya da doldurmasını engellemek açısından gayet kullanışlı bir özelliktir. Kotalar sadece root kullanıcısı tarafından aktif hale getirilebilir ya da kullanılabilir. Bu bölümde kotaları nasıl ayarlayacağınızı ya da efektif bir şekilde kullanacağınızı anlatmaya çalışacağız.

### 19.2 Çekirdek Desteği

Kotalar dosya sisteminin bir özelliği olduğundan çekirdeğin desteğine ihtiyaç duyarlar. Öncelikle çekirdeğinizde kota desteğinin olup olmadığını belirlemeniz gerekmektedir. Bunun için grep komutunu kullanabilirsiniz:

```
# cd /usr/src/linux
# grep -i quota .config
CONFIG_QUOTA=y
CONFIG_XFS_QUOTA=y
```

Eğer bu komut daha az bir bilgiyle dönerse (mesela CONFIG\_QUOTA ayarlanmamıştır şeklinde) çekirdeği kota desteği verecek şekilde yeniden derlemeniz gerekmektedir. Bu aslında zor bir işlem olmasa da bu kursun konusu dışında olduğundan burada söz etmeyeceğiz.

### 19.3 Dosya Sistemi Desteği

Kota yönetimini ayrıntılı incelemeden önce şunu belirtmeliyiz ki kota desteği 2.4.x çekirdek serisinde henüz tamamlanmamıştır. ext2 ve ext3 dosya sistemlerinde kotalar ile ilgili problemlere rastlanmakta ve ReiserFS dosya sisteminde desteklenmediği görülmektedir. Bu eğitimde örnekler XFS dosya sistemi temel alınarak uygulanmaktadır. Çünkü bu dosya sisteminin kotaları düzgün bir şekilde desteklediği bilinmektedir.

### 19.4 Kotaların Konfigürasyonu

Sisteminizde kotaları ayarlamaya başlamak için, kotalar aktif hale getirilmiş bir şekilde etkilenen dosya sistemlerini /etc/fstab dosyası içinde düzenlemelisiniz. Buradaki örnekte ext3 dosya sistemimiz kullanıcı ve grup kotaları aktif hale getirilmiş bir şekilde mount edilmektedir:

```
# grep quota /etc/fstab
/dev/hda8 /mnt/usrDeneme ext3 usrquota,grpquota,noauto,noatime 1 2
# mount /mnt/usrDeneme
```

### 19.5 Kotaların Konfigürasyonu, Devamı

Unutmayınız, usrquota ve grpquota seçenekleri kotaları aktif hale getirmek için gerek şart değildir. Kotaların aktif hale getirildiğinde quotaon komutunu kullanarak emin olabilirsiniz.

```
# quotaon /mnt/usrDeneme/
```

İleride ihtiyaç duyabileceğinizi tahmin ederek, kotaların aktif halini bozmak için de quotaoff komutunu kullanabileceğinizi söyleyebiliriz:

```
# quotaoff /mnt/usrDeneme/
```

Ama şu an için bu eğitimde yer alan örnekleri deneyecekseniz kotaların aktif hale gelmiş olduğundan emin olunuz.

## 19.6 quota Komutu

quota komutu o anda mount edilmiş tüm dosya sistemleri için kullanıcının disk kullanımını ve limitlerini gösterir. -v seçimi kotaların kullanılabildiği ama kullanıcı için alan ayrılmadığı dosya sistemlerini gösterir.

```
# quota -v
Disk quotas for user root (uid 0):
Filesystem blocks quota limit grace files quota limit grace
/dev/hdc1 0 0 0 3 0 0
```

Birinci kolon, bloklar kolonu, listelenen her dosya sisteminde o anda root kullanıcının ne kadar miktarda disk alanı kullandığını gösterir. Ondan sonra gelen kolonlar, kota ve limit kolonları, disk alanı için o anda yer alan limitleri temsil eder. Daha sonra kota ve limit arasındaki farkı ve grace kolonunun anlamını açıklayacağız. Dosyalar kolunu root kullanıcının ilgili dosya sisteminde ne kadar dosyaya sahip olduğunu gösterir. Ondan sonra gelen kota ve limit kolonları dosyalar üzerindeki limitleri temsil eder.

## 19.7 Kotaları Görüntülemek

Herhangi bir kullanıcı önceki örnekte de gördüğümüz gibi kendine ait kotalarla ilgili raporu görüntüleyebilir. Ancak diğer kullanıcı ve grupların kotalarında sadece root kullanıcısı bakabilir. Diyelim ki /usr/users altına mount edilmiş bir /dev/hdc1 dosya sistemimiz, ve ali ve veli adında da iki tane kullanıcımız olsun. Öncelikle ali kullanıcısının disk kullanımına ve limitine bakalım:

```
# quota -v ali
Disk quotas for user ali (uid 1003):
Filesystem blocks quota limit grace files quota limit grace
/dev/hdc1 4100 0 0 6 0 0
```

Bu örnekte görüyoruz ki ali'nin kotaları sıfıra ayarlanmıştır ki bu da hiç bir limiti olmadığı anlamına gelir.

## 19.8 edquota

Diyelim ki ali isimli kullanıcıya kota vermek istiyoruz. Bu durumda kullanmamız gereken komut edquota olacaktır. Kotaları düzenlemeye başlamadan önce /usr/users üzerinde ne kadar alanımız olduğunu görelim:

```
# df /usr/users
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/hdc1 610048 4276 605772 1% /usr/users
```

Bu 600MB lık alanın kısmen çok büyük bir dosya sistemi olmadığını sanırım siz de görebiliyorsunuz. Bu sebeple ali kullanıcısına belli bir alandan daha fazlasını kullanamayacağı bir kota vermek mantıklı bir yaklaşım olacaktır. edquota komutunu çalıştırdığınız zaman komut satırında belirlediğiniz her kullanıcı ve grup için geçici bir dosya yaratılacaktır.

## 19.9 edquota, Devamı

edquota komutu sizi, bu geçici dosya aracılığıyla kotaları ekleyebileceğiniz ve/veya düzenleyebileceğiniz belli bir editör içerisinde çalıştırır.

```
# edquota ali
Disk quotas for user ali (uid 1003):
Filesystem blocks soft hard inodes soft hard
/dev/hdc1 4100 0 0 6 0 0
```

Yukarıdaki kota komutunun çıktısına benzer olarak bu geçici dosyadaki "blocks" ve "inodes" kolonları ali kullanıcısının kullandığı disk alanını ve dosya sayısını belirtmektedir. Blok ve inode sayılarını değiştiremezsiniz. Bunu yapmaya çalıştığınız zaman sistem size engel olacaktır. soft ve hard kolonları gördüğümüz gibi ali kullanıcısının değerleri şu anda limitsiz kotalarını göstermektedir. (0 demek limit yok demektir. )

## 19.10 edquota'yı Anlamak

soft kısıtlaması dosya sistemi üzerinde ali kullanıcısının kendisine tahsis ettiği kullanılacak maksimum disk alanının miktarını belirler. (bir diğer deyişle sahip olduğu kotasını belirler). Eger ali kullanıcısı kendisine verilen bu soft kısıtlamasını aşacak kadar fazla alan kullanırsa, e-mail ile bu kota hakkında bir uyarı alacaktır. Hard kısıtlaması sistemdeki bir kullanıcının aşamayacağı sınırı belirler. Eger ali kullanıcısı bu sınırı da aşacak bir işlem yapmaya kalkarsa "disk kotası aşıldı" (disk quota exceeded) şeklinde bir mesaj alacak ve yapmak istediği işlem tamamlanmayacaktır.

## 19.11 Değişiklikler Yapmak

O halde burada ali kullanıcısını soft ve hard limitlerini değiştirerek dosyayı kaydedebiliriz.

```
Disk quotas for user ali (uid 1003):
Filesystem blocks  soft  hard inodes soft hard
/dev/hdc1  4100 10000 11500      6 2000 2500
```

quota komutunu da çalıştırarak yaptığımız değişiklikleri gözlemleyebiliriz:

```
# quota ali
Disk quotas for user ali (uid 1003):
Filesystem blocks quota limit grace files quota limit grace
/dev/hdc1 4100 10000 11500 6 2000 2500
```

## 19.12 Kotaları Kopyalamak

Hatırlarsanız bu dosya sisteminde bir de veli kullanıcısını olduğunu kabul etmiştik. Eğer bu kullanıcıya ali kullanıcısı ile aynı kotayı vermek istiyorsak edquota ile -p parametresini kullanarak bu komuta bundan sonra gelen tüm kullanıcılar için ali kullanıcısını prototip olarak kullanmasını söyleyebiliriz. Grup kullanıcılarını kotalarını ayarlamak için bu yöntem çok kullanışlı olacaktır.

```
# edquota -p ali veli
# quota veli
Disk quotas for user veli (uid 1003):
Filesystem blocks quota limit grace files quota limit grace
/dev/hdc1      0 10000 11500      1 2000 2500
```

## 19.13 Grup Kısıtlamaları

edquota komutunu kullanarak dosyaların disk üzerinde kullandığı alanı belli bir grubu temel alarak kısıtlayabiliriz. Mesela users grubunun kotasını düzenlemek için:

```
# edquota -g users
Disk quotas for group users (gid 100): Filesystem blocks soft hard inodes
soft hard /dev/hdc1 4100 500000 510000 7 100000 125000
```

Şimdi de users grubunun düzenlenmiş yeni kotasını görmek için de:

```
# quota -g users
Disk quotas for group users (gid 100):
Filesystem blocks quota limit grace files quota limit grace
/dev/hdc1  4100 500000 510000      7 100000 125000
```

## 19.14 repquota Komutu

Sistemde çok fazla sayıda kullanıcımız varsa, quota komutunu kullanarak her kullanıcının kotasını görüntülemek bizi canımızdan bezdirebilir. repquota komutu dosya sistemindeki kotaları güzel bir şekilde raporlamaya yarar. Örneğin, /usr/users üzerindeki tüm kullanıcı ve grupların kotaların görmek için:

```
# repquota -ug /usr/users

*** Report for user quotas on device /dev/hdc1
Block grace time: 7days; Inode grace time: 7days
```

```

Block limits File limits
User used soft hard grace used soft hard grace
-----
root -- 0 0 0 3 0 0
john -- 0 10000 11500 1 2000 2500
jane -- 4100 10000 11500 6 2000 2500
*** Report for group quotas on device /dev/hdc1
Block grace time: 7days; Inode grace time: 7days
Block limits File limits
Group used soft hard grace used soft hard grace
-----
root -- 0 0 0 3 0 0
users -- 4100 500000 510000 7 100000 125000

```

## 19.15 repquota Seçenekleri

Burada bahsetmekte yarar göreceğimiz bir kaç tane daha repquota seçeneği vardır. repquota -a kota ayarı aktif olan mevcut olan-mount edilmiş okunabilir-yazılabilir dosya sistemlerini raporlamaya yarayacaktır. repquota -n isimlerde uid'leri gid'leri çözmeyecektir. Bu uzun listeleri görüntülerken zaman kazandıracaktır.

## 19.16 Kotaları Görüntülemek

Eğer sistem yöneticisi iseniz, sistemdeki kotaların geçilmediğinden emin olmak için bunları görüntülemenin bir yolunu arayacaksınız. Bunun için kolay yöntemlerden bir tanesi warnquota komutunu kullanmaktır. Bu komut sistemde kotalarını geçen kullanıcılara kendilerini uyaran bir mail gönderir. Benzer olarak warnquota cron-job gibi çalışmaktadır.

Bir kullanıcı kendi "soft" limitini aştığı zaman quota komutunun çıktısında "grace" kolonunda kotanın aşılma mühleti, yani bu dosya sisteminde ne kadar zamandır kotanın aşılmış olduğu belirecektir.

```

Disk quotas for user jane (uid 1003):
Filesystem blocks quota limit grace files quota limit grace
/dev/hdc1 10800* 10000 11500 7days 7 2000 2500

```

Öntanımlı olarak, bloklar ve inode lar için bu mühlet 7 gündür.

## 19.17 Mühleti Belirlemek

equota komutu kullanarak bir dosya sistemi için kota aşma mühletini belirleyebilirsiniz.

```
# edquota -t
```

Bu yazım sizi aşağıdaki gibi bir geçici dosyanın editörü içerisine atacaktır:

```

Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem Block grace period Inode grace period
/dev/hdc1 7days 7days

```

Dosya içindeki yazı oldukça açıklayıcı niteliktedir. Kotalarını aşan kullanıcılarınıza uyarı gönderildikten sonra bunu farketmeleri ve gerekli dosyaları silmeleri için yeterli zamanı bırakmayı unutmamalısınız.

## 19.18 Kotaları Açılışta Kontrol Etmek

Kotaları açılış sırasında da kontrol edebilirsiniz. quotacheck komutunu çalıştıran bir betik (script) dosyasını kullanarak bunu yapabilirsiniz. Quota mini howto belgesinde bunun için örnek bir betik yer almaktadır. quotacheck komutu aynı zamanda bozulmuş kota dosyalarını onarma yeteneğine de sahiptir. Bu konuda daha fazla bilgi almak için quotacheck (8) kılavuz sayfasını inceleyebilirsiniz.

Bundan önce quotaon ve quotaoff komutlarından bahsetmiştik. Açılış betiğinizin içerisine quotaon ekleyerek kotaların aktif hale gelmesinin sağlayabilirsiniz. Kotaların desteklendiği bütün dosya sistemlerinde aktif hale gelmesini isterseniz -a seçeneğini kullanınız.

```
# quotaon -a
```

## 20 Sistem Logları

### 20.1 syslogd'ye Giriş

syslog sorumlusu (daemon), sistemde çalışan programlardan gelen mesajlar ile olgun bir istemci-sunucu mekanizması sunar. Syslog bir program sorumlusundan (daemon) ya da programdan gelen mesajları alır, önceliğine ve tipine göre kategorize eder ve yönetici konfigürasyonu kurallarına göre raporlar. Logları yönetebilmek açısından bu yöntem oldukça sağlamdır ve bütünlük arz etmektedir.

### 20.2 Logları Okumak

Şimdi gelin syslog tarafından kaydedilmiş bir log dosyasını içeriğini inceleyelim. Sonra da syslog konfigürasyonunun nasıl yapılacağına dönebiliriz. FHS log dosyalarının /var/log içerisinde yer alması gerektiğini belirlemektedir. Burada tail komutunu kullanarak son on mesajın görüntüleyeceğiz:

```
# cd /var/log
# tail messages
Jan 12 20:17:39 bilbo init: Entering runlevel: 3
Jan 12 20:17:40 bilbo /usr/sbin/named[337]: starting BIND 9.1.3
Jan 12 20:17:40 bilbo /usr/sbin/named[337]: using 1 CPU
Jan 12 20:17:41 bilbo /usr/sbin/named[350]: loading configuration from
        '/etc/bind/named.
Jan 12 20:17:41 bilbo /usr/sbin/named[350]: no IPv6 interfaces found
Jan 12 20:17:41 bilbo /usr/sbin/named[350]: listening on IPv4 interface lo,
        127.0.0.1#53
Jan 12 20:17:41 bilbo /usr/sbin/named[350]: listening on IPv4 interface eth0,
        10.0.0.1#
Jan 12 20:17:41 bilbo /usr/sbin/named[350]: running
Jan 12 20:41:58 bilbo gnome-name-server[11288]: starting
Jan 12 20:41:58 bilbo gnome-name-server[11288]: name server starting
```

Yazı işleme konusundan hatırlayacaksınız ki tail komutu bir dosyadaki son satırları görüntüler. Burada gördüğümüz gibi bilbo isimli nameserver sistemde en sona yakın zamanda çalışmaya başlamıştır. Eğer IPv6 konuşlandırılacak olsaydı named daemon'ının hiç bir IPv6 arayüzü bulamadığını farkedecektik. Ayrıca tüm bunlara ek olarak yine bu dosyada bir kullanıcının gnome-name-server'dan hemen önce GNOME çalıştırdığını anlamaktayız.

### 20.3 Logları Kuyruktan Takip Etmek

Tecrübeli bir sistem yöneticisi bir log dosyasının çıkışını gözlemlemek için tail -f komutunu kullanacaktır.

```
# tail -f /var/log/messages
```

Yukarıdaki kullanım sayesinde bir servis çalıştırırken ya da durdururken kendisinden gelen mesajları da görme şansımız olacaktır. Debug işlemi için bu çok kullanışlı bir yöntemdir. Hatta bazı sistem yöneticileri ortaya çıkan mesajları o anda hemen görebilmek için bir terminalde sürekli bu komutu çalışır halde kullanmaktadırlar. Mesela bir terminalde yukarıdaki komutu yazıp başka bir terminalde de httpd servisini kapatalım:

```
# service httpd stop
Shutting down httpd2:          [ OK ]
#
```

Şimdi ilk terminale dönersek en alt satıra eklenen şu mesajı da göreceğiz:

```
.....
...
May  1 23:11:00 localhost CROND[3827]: (mail) CMD
        (/usr/bin/python -S /var/lib/mailman/cron/qranner)
May  1 23:11:25 localhost httpd: httpd shutdown succeeded
```

## 20.4 Log Dosyalarını grep ile Kullanmak

log dosyalarını incelemek için bir başka yararlı yöntem de grep komutunu kullanmak olacaktır. Yukarıda bahsedilen durum için "httpd" ile birlikte grep kullanarak durumu ıceleyebiliriz:

```
# grep httpd /var/log/messages
```

Ya da httpd ile ilgili olarak gelen son 3 mesajı da şu şekilde görebiliriz:

```
[root@localhost erman]# grep httpd /var/log/messages| tail -3
May  1 23:11:25 localhost httpd: httpd shutdown succeeded
May  1 23:14:15 localhost httpd2: httpd2: Could not determine the server's
        fully qualified domain name, using 127.0.0.1 for ServerName
May  1 23:14:18 localhost httpd: httpd2 startup succeeded
```

## 20.5 Log Dosyalarının Özeti

Aşağıda genelde /var/log altında yer alan syslog tarafından kullanılan ve belirlenen log dosyaları özetlenmiştir.

- messages: Genel sistem programları ve daemonlar tarafından üretilen bilgi ve hata mesajları
- secure: Ekstra güvenlik amacıyla messages dosyasındaki mesajlardan ayrı tutulan ve şifre doğrulama ile ilgili hata ve bilgi mesajları
- maillog: Mail ile ilişkili hata ve bilgi mesajları
- cron: Cron ile ilgili hata ve bilgi mesajları
- spooler: UUCP ve haberlerle ilgili hata ve bilgi mesajları

## 20.6 syslog.conf

Şimdi artık syslog konfigürasyon dosyası olan /etc/syslog.conf dosyasını inceleyebiliriz. (Not: Eğer syslog.conf dosyanız yoksa bile bilgi edinmek için bu kısmı yine de okuyun ama başka bir syslog da kullanıyor olabilirsiniz.) Bu dosyaya göz gezdirdiğimizde yukarıda bahsedilen ortak log dosyalarının her biri için bir satır olduğunu göreceksiniz. Ayrıca bunun dışında başka satırlar da olacaktır. Aşağıda belirlendiği gibi dosyanın formatı kullanıcı.öncelik şeklindedir.

## 20.7 syslog.conf, Devamı

Kullanıcı Alanı: Mesajı üreten alt sistemi belirler. Bu alan ile ilgili geçerli anahtar sözcükler auth, authpriv, cron, daemon, kern, lpr, mail, news, syslog, user, uucp ve local0 dan local7'e kadar olan anahtar sözcüklerdir. Öncelik Alanı: Mesajın minimum seyrekliğini belirler. Bunu anlamı ilgili mesaj en az belirlenen seyreklikte ya da daha fazla sayıda bu kuralla eşlecektir. Öncelikle ilgili anahtar sözcükler debug, info, notice, warning, err, crit, alert ve emerg anahtar sözcükleridir.

Aksiyon Alanı: Aksiyon alanında ya bir dosya adı, ya tty (/dev/console gibi), ya başında @ işareti ile belirlenmiş başka bir makina adı ya da mesajın o anda sistemde olan herkese gönderileceği anlamına gelen \* işareti olmalıdır. En çok kullanılan aksiyon ise bir dosya adıdır.

## 20.8 Yeniden Yükleme ve Ek Bilgiler

Umarız bu konfigürasyon syslog sisteminin esnekliğini ve gücünü anlamana yardımcı olmuştur. Değişiklik yapmadan önce daha ayrıntılı bilgi için syslog.conf (5) kılavuz dosyasını okumanızı tavsiye ederiz. Ayrıca syslogd(8) kılavuz dosyası da bir çok ayrıntıdan bahsetmektedir.

Unutmayınız ki konfigürasyon dosyasında yaptığımız değişikliklerin etkin olması için syslog arka plan sürecini (daemon) bir şekilde bilgilendirmeniz gerekmektedir. Ona bir SIGHUP göndermek bu iş için yeterli olacaktır. Ve bunu kolaylıkla yapmak için de killall komutu kullanabilirsiniz:

```
# killall -SIGHUP syslogd
```



## 20.9 Güvenlik Notu

Şunu belirtmek gerekir ki syslogd tarafından yazılan log dosyaları eğer mevcut değilse otomatik olarak yaratılırlar. Sizin o andaki umask ayarınızdan bağımsız olarak dosya herkesin okuyabildiği (world-readable) şekilde yaratılacaktır. Eğer güvenlik konusu sizi ilgilendiriyorsa dosyaları sadece root kullanıcısını okuyabileceği ve yazabileceği şekilde chmod ile izinlerini değiştirmelisiniz. Buna ek olarak aşağıda anlatılacak olan logrotate programı uygun izinlerle yeni log dosyaları oluşturacak şekilde konfigüre edilebilir. syslog daemon'ı her zaman için mevcut olan bir log dosyasının o andaki özelliklerini muhafaza edeceğinden, dosya yaratıldıktan sonra bir daha bunu düşünmenize gerek kalmayacaktır.

## 20.10 logrotate

/var/log altında yer alan log dosyaları sürekli büyüyecektir ve sistemi doldurma potansiyelleri vardır. logrotate gibi logları otomatik arşivleyen ve yöneten bir programın kullanılması tavsiye edilen bir durumdur. logrotate günlük bir cron-job olarak çalışır ve log dosyalarını çevirme, sıkıştırma, silme ya da mail atma özellikleri katmak için konfigüre edilebilir.

Örneğin öntanımlı bir logrotate konfigürasyon dosyası logları 4'er haftalık backlogs halinde tutarak haftalık olarak çevirebilir (dosyaya bir seri numarası ekleyerek bunu yapar) ve bu backlogları yer kazanmak amacıyla sıkıştırır. Buna ek olarak program syslogd'ye, artık log dosyalarının boş olduğunu ve bu dosyaların sonuna uygun ekleme yapmasını sağlamak amacıyla bir SIGHUP göndermesini sağlayacak şekilde konfigüre edilebilir.

logrotate hakkında daha fazla bilgi almak için, içerisinde programın açıklaması ve konfigürasyon dosyasının sözdiziminin (syntax) yer aldığı logrotate(8) kılavuz sayfasını inceleyebilirsiniz.

## 20.11 İleri Başlık: klogd

syslog konusunu kapatmadan önce hevesli okuyucular için bir kaç ayrıntıdan bahsedilebilir. Bu ipuçları syslog ile ilgili başlıkları incelerken rahat kavramanıza yardımcı olabilir.

Öncelikle şunu belirtmek gerekir ki; syslog arka plan süreci (daemon) aslında içerisinde klogd adında başka bir arka plan sürecin de yer aldığı syslogd paketinde yer almaktadır. klogd'nin işi çekirdekten bilgi ve hata mesajlarını almak, kategorize etme ve raporlama işini yapması için syslogd'ye göndermektir. klogd tarafından alınan mesajlar sizin dmesg komutuyla elde edeceklerinizle tamamen aynıdır. Fark ise; dmesg o anda buffer da yer alan uyarıyı aynen yazarken klogd bu mesajları syslogd'ye gönderir. Böylece buffer boşaltılsa bile artık bu mesajlar kaybolmayacaktır.

## 20.12 İleri Başlık: Alternatif Loglayıcılar

İkinci olarak syslogd paketine alternatif olacak başka log programları olduğunu belirtmeliyiz. Alternatifler daha verimli ve daha kolay konfigüre edilebilmek ve daha fazla özellik içermek amacıyla ortaya çıkmıştır. Eğer syslogd'nin sizin için yeterince iyi olmadığını düşünürseniz en popüler olanlardan Syslog-ng ve Metalogunda yer aldığı bu alternatifleri deneyebilirsiniz.

Üçüncü olarak logger komutunu kullanarak mesajları kendi betiğiniz içerisinde loglayabilirsiniz. logger(1) kılavuz sayfasında daha fazla bilgi bulabilirsiniz.

Open Source Enterprise Centre  
GNU/Linux Sertifikasyon Programı

Linux 102 Ders Notları

# İçindekiler

<b>1</b>	<b>Paylaşılan Kütüphaneler</b>	<b>1</b>
1.1	Paylaşılan Kütüphanelere Giriş . . . . .	1
1.2	Dinamik olarak bağlanmış bağımlılıklar . . . . .	1
1.3	Dinamik Yükleyici . . . . .	1
1.4	ldconfig Hakkında İpuçları . . . . .	2
1.5	LD_LIBRARY_PATH . . . . .	2
<b>2</b>	<b>UygulamalarıKaynaktan Derlemek, Giriş</b>	<b>2</b>
2.1	Dosyalarıİndirmek (downloading) . . . . .	2
2.2	Dosya Arşivleme ve Sıkıştırmaya İlişkin Kısa Açıklama . . . . .	2
2.3	Paketi Açmak . . . . .	3
2.4	Arşivleri Listelemek . . . . .	4
2.5	bzip2 ile Sıkıştırılmış Arşivleri Açmak . . . . .	4
2.6	bzip2 Boruhatları(devamı) . . . . .	5
2.7	Kaynaklarıİncelemek . . . . .	5
2.8	Konfigürasyon . . . . .	5
2.9	configure Kullanmak . . . . .	5
2.10	-prefix Seçeneği . . . . .	5
2.11	-prefix Kullanmak . . . . .	5
2.12	Peki ya FHS ne diyor? . . . . .	5
2.13	Configure . . . . .	6
2.14	config.cache . . . . .	6
2.15	configure ve Makefile Dosyaları . . . . .	6
2.16	Makefile Giriş . . . . .	6
2.17	make ProgramınıÇağırarak . . . . .	6
2.18	Kurulum . . . . .	7
2.19	make install . . . . .	7
2.20	Program bir defa yüklendikten sonra . . . . .	7
2.21	İşte bu kadar! . . . . .	7
2.22	OlasıProblemler . . . . .	7
2.23	Eksik Kütüphaneler . . . . .	8
2.24	Diğer Problemler . . . . .	8
2.25	Diğer Problemler (devamı) . . . . .	8
<b>3</b>	<b>Paket Yönetimi Kavramları, Paket Yönetimi Avantajları</b>	<b>8</b>
3.1	Paket Yönetiminin Dezavantajları . . . . .	8
<b>4</b>	<b>rpm the (R)ed Hat (P)ackage (M)anager, rpm'e Giriş</b>	<b>8</b>
4.1	Bir rpm Yükleme . . . . .	9
4.2	Bir rpm'i yeniden Yükleme . . . . .	9
4.3	Bir rpm i Zorlama ile Yükleme . . . . .	9
4.4	-nodeps Seçeneği İle Yükleme ve Kaldırma . . . . .	10
4.5	Paketleri Güncelleme . . . . .	10
4.6	rpm -q İle Sorgulama Yapma . . . . .	11
4.7	rpm -ql ile DosyalarıListelemek . . . . .	11
4.8	rpm -qp İle Paketleri Sorgulamak . . . . .	11
4.9	Tüm Kurulu Paketleri Sorgulamak . . . . .	12
4.10	Bir Dosyanın Sahibini Bulma . . . . .	12
4.11	BağımlılıklarıGöstermek . . . . .	12
4.12	Bir Paketin Bütünlüğünü Sağlama . . . . .	13
4.13	Kurulmuş Bir Paketin Kontrolünü Sağlama . . . . .	13
4.14	rpm'i Konfigüre Etme . . . . .	14

<b>5</b>	<b>Debian Paket Yönetimi, apt-get Giriş</b>	<b>14</b>
5.1	Taklit Kurulum . . . . .	14
5.2	Paket Kaynak Listesi: apt-setup . . . . .	14
5.3	apt-get'ten dselect'e . . . . .	15
5.4	dselect'e Başlarken . . . . .	15
5.5	dselect Select Modunu Kullanmak . . . . .	16
5.6	Paket Durumu . . . . .	16
5.7	Kurulum ve Konfigürasyon . . . . .	17
5.8	dpkg İle Kurulmuş bir Paketin Durumunu Ele Almak . . . . .	17
5.9	Bir dosya ile .deb Arasındaki Bağlantı . . . . .	17
5.10	Yüklenecek Paketleri Bulmak . . . . .	18
5.11	Dpkg İle Bir Paketi Konfigure Etmek . . . . .	18
5.12	Ek Debian Paket Yönetim Araçları . . . . .	19
<b>6</b>	<b>Huzurlarınızda çekirdek... Linux!</b>	<b>19</b>
6.1	Donanım ile arabirim oluşturmak . . . . .	19
6.2	CPU Soyutlama . . . . .	19
6.3	IO(giriş/çıkış)'larıSoyutlama . . . . .	19
6.4	Ağ Merkezi . . . . .	19
6.5	Ağın Faydaları . . . . .	19
6.6	Açılış işlemleri . . . . .	20
6.7	Ve huzurlarınızda... modüller . . . . .	20
6.8	Modüller nerede durur ? . . . . .	20
6.9	Modüller – her işlem için kullanılamaz . . . . .	20
<b>7</b>	<b>Çekirdek kaynaklarınıbulup bilgisayarınıza indirmek</b>	<b>20</b>
7.1	Hangi çekirdek kaynaklarınıkullanmalı? . . . . .	20
7.2	Çekirdeği kaynağından edinmek . . . . .	21
7.3	Çekirdek kodu paketini açmak . . . . .	21
<b>8</b>	<b>Çekirdeği Düzenlemek</b>	<b>21</b>
8.1	Yeni yöntem . . . . .	21
8.2	Konfigürasyon ipuçları . . . . .	21
<b>9</b>	<b>Çekirdeği derlemek ve kurmak</b>	<b>23</b>
9.1	make bzImage . . . . .	23
9.2	Modülleri derlemek . . . . .	23
<b>10</b>	<b>LILLO'da başlangıç konfigürasyonu</b>	<b>23</b>
10.1	LILLO'yu konfigüre etmek . . . . .	24
10.2	LILLO kodu . . . . .	24
10.3	LILLO konfigürasyonunun nedenleri ve niçinleri . . . . .	24
<b>11</b>	<b>PCI aygıtları</b>	<b>25</b>
11.1	Mevcut PCI aygıtlarınıincelemek . . . . .	25
11.2	PCI donanım kaynakları . . . . .	25
<b>12</b>	<b>Linux USB</b>	<b>26</b>
12.1	USB'yi etkin hale getirmek . . . . .	26
12.2	Son birkaç adım . . . . .	26
12.3	usbdevfs'yi bağlamak (mount) . . . . .	26
<b>13</b>	<b>Diff, Patch</b>	<b>27</b>
13.1	Diff . . . . .	27
<b>14</b>	<b>Çekirdeği yamamak (patch)</b>	<b>28</b>
<b>15</b>	<b>Kaynaklar</b>	<b>29</b>

<b>16 TCP/IP Ağ İşlemlerine Giriş</b>	<b>29</b>
16.1 Çözüm: Ethernet üzerinden TCP/IP	30
16.2 IP adreslerine giriş	30
16.3 IP adresleri ile Ethernet arayüzünü eşleştirmek	31
16.4 ifconfig -a kullanımı	31
16.5 TCP/IP çalışıyor!	32
16.6 İsim çözümleme sınırları	32
16.7 DNS kullanımı	32
16.8 Dışarıya bağlanmak	33
16.9 Ev ödevi	33
<b>17 Internet servisleri ve inetd'ye giriş</b>	<b>33</b>
17.1 inetd'yi ayarlamak: /etc/services	33
17.2 inetd'yi ayarlamak: /etc/inetd.conf	34
17.3 Servisleri iptal etmek	34
17.4 inetd'yi bir başlangıç betiği ile başlatmak/durdurmak	34
17.5 inetd'yi manuel olarak durdurmak/başlatmak	35
17.6 TCP wrappers'a giriş	35
17.7 TCP wrappers ile kayıt tutma	35
17.8 TCP wrappers kullanarak erişimi yerel kullanıcılar ile kısıtlamak	35
17.9 TCP wrappers ile sadece bilinen kullanıcılara izin vermek	36
17.10 xinetd: geliştirilmiş (extended) inetd	36
17.11 xinetd ayarlaması	36
<b>18 Güvenliğe Genel Bakış - Giriş</b>	<b>37</b>
18.1 Dosya izinleri ve log dosyaları	37
18.2 root kullanıcısının diğer dosyaları ile ilgili izinler	37
18.3 Kullanıcı dosyalarının dosya izinleri	37
18.4 SUID/SGID programlarını bulmak	38
18.5 ulimit ile kullanıcıların limitlerini ayarlamak	38
18.6 ulimit ile CPU zamanını sınırlamak	38
18.7 Kullanıcı limitleri, devam	39
18.8 Gizli girişlerin engellenmesi (intrusion prevention)	39
18.9 Kullanılmayan ağ servislerini kapatmak (süpersunucu)	39
18.10 Kullanılmayan ağ servislerini kapatmak (tek başına çalışan sunucular)	40
18.11 Değişiklikleri test etmek	40
18.12 Bakım yaparken kullanıcıların girişini engelleme	40
18.13 iptables (ipchains) konusuna giriş	41
18.14 iptables ve Linux paket filtresi	41
18.15 Gizli girişlerin tespiti - syslog dosyaları	41
18.16 Gizli giriş tespiti - tripwire	41
18.17 Gizli giriş tespiti - portsentry	42
18.18 Genel tavsiyeler: Yazılımı Güncel Tutmak	42
18.19 Genel tavsiyeler: yüksek kaliteli parolalar	42
18.20 Genel tavsiyeler: Güvenliğinizi Test Etmek	43
<b>19 Yazdırma İşlemlerine Giriş</b>	<b>43</b>
19.1 Bir yazıcı spooler daemon kurulumu (lpd)	43
19.2 Temel yazıcı ayarları (/etc/printcap)	43
19.3 Spool dizinlerinin yaratılması	44
19.4 Yazdırma spooler istemcilerini kullanmak	44
19.5 Uzaktaki bir LPD sunucusuna yazdırmak	45
19.6 Uzaktaki bir MS Windows ya da Samba sunucusuna yazdırmak	45
19.7 Magicfilter	46
19.8 printcap'i Magicfilter'i gösterecek şekilde ayarlamak	46
19.9 Magicfilter alternatifi olarak Apsfilter	46
19.10 Kaynaklar	47
19.11 Her sistem yöneticisinin düzenli olarak takip etmesi gereken en önemli siteler:	47
19.12 Etkileşimli, güvenli kabuk oturumu	47

19.13Güvenli kabuk . . . . .	47
19.14ssh kullanımı . . . . .	48
19.15sshd başlatmak . . . . .	48
19.16Güvenli kopya . . . . .	48
19.17Güvenli kabuk doğrulama ve yetkilendirme tercihleri . . . . .	48
<b>20 NFS</b>	<b>49</b>
20.1 NFS'e giriş . . . . .	49
20.2 Temel NFS bilgileri . . . . .	49
20.3 NFS'in özellikleri . . . . .	49
20.4 Linux'ta NFS, sürüm 3 . . . . .	49
20.5 NFS'te güvenlik . . . . .	49
<b>21 NFS kurulumu</b>	<b>49</b>
21.1 /etc/exports üzerindeki düzenlemeler . . . . .	50
21.2 İhraç kısıtlamalarıile çalışma . . . . .	50
21.3 Bir başka /etc/exports örneği . . . . .	50
21.4 NFS3 sunucusunu çalıştırmak . . . . .	51
21.5 İhraç tercihlerini değiştirme . . . . .	51
21.6 NFS istemcilerini yapılandırma . . . . .	51
21.7 NFS istemci servislerinin çalıştırılması . . . . .	52
21.8 İhraç edilen NFS dosya sistemlerini bağlama . . . . .	52
21.9 İhraç edilen dizinlerin içinde yapılan bağlama işlemleri . . . . .	52

# 1 Paylaşılan Kütüphaneler

## 1.1 Paylaşılan Kütüphanelere Giriş

Linux sistemlerinde çalıştırılabilir iki çeşit program tipi vardır. Birinci çeşit programlar statik olarak bağlanmış çalıştırılabilir programlar olarak adlandırılır. Statik çalıştırılabilirler (static executables) ihtiyaçları olan tüm fonksiyonları barındırırlar. Bir başka deyişle bunlar bütün halledirler. Bu yüzden statik çalıştırılabilirler kullanılmak için harici bir kütüphaneye ihtiyaç duymazlar. İkinci çeşit ise dinamik çalıştırılabilirler olarak adlandırılır. Bu ikinci tip programlara az sonra değineceğiz.

Statik Çalıştırılabilirler ile Dinamik Çalıştırılabilirlerin Kıyaslanması İlgili çalıştırılabilir programın statik olup olmadığını ldd komutunu kullanarak anlayabiliriz.

```
# ldd /sbin/lilo
not a dynamic executable
```

ldd tarafından bildirilen "dinamik çalıştırılabilir değildir" bilgisi bize lilo programının statik bağlantılı olduğunu anlatmaktadır.

Dinamik çalıştırılabilirler çalışmaları için gerekli fonksiyonları almak için paylaşılmış kütüphanelere ihtiyaç duyan, bir anlamda eksik programlardır. Bu yüzden aynı programa ilişkin statik çalıştırılabilir versiyonun boyutu dinamik çalıştırılabilir versiyonuna göre daha büyük olacaktır.

## 1.2 Dinamik olarak bağlanmış bağımlılıklar

ln komutunun ihtiyaç duyduğu paylaşılmış kütüphanelerin listesini almak için ldd komutunu kullanırız.

```
# ldd /bin/ln
libc.so.6 => /lib/libc.so.6 (0x40021000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Gördüğümüz gibi, ln çalışmak için libc.so.6 ve ld-linux.so.2 kütüphanelerine ihtiyaç duyar. Bir kural olarak, az önce de bahsettiğimiz gibi dinamik olarak bağlanmış programlar statik olarak bağlanmışlara göre çok daha az yer tutarlar. Öte yandan bir Linux sistemindeki hemen hemen bütün çalıştırılabilir programlar dinamik bağlantılıdır.

## 1.3 Dinamik Yükleyici

Öyleyse, eğer dinamik çalışabilirler, çalışmaları için ihtiyaç duydukları herşeye sahip değillerse Linux'un hangi parçası gerektiği zaman bu programlarla birlikte gerekli kütüphaneleri yükleme görevini üstlenir ? Bunun cevabı dinamik yükleyici (dynamic loader) adı verilen bir kütüphane; yani ld-linux.so.2 dir (ln komutunun bağımlı olduğu kütüphaneler arasında gördüğümüz gibi)

Şimdi de dinamik yükleyicinin sistemdeki gerekli paylaşılmış kütüphaneleri nasıl bulduğuna bir bakalım.

```
ld.so.conf
```

Dinamik yükleyici paylaşılan kütüphaneleri /etc/ld.so.conf ve /etc/ld.so.cache dosyaları sayesinde bulmaktadır. Eğer /etc/ld.so.conf dosyasına cat komutu ile bakacak olursanız muhtemelen aşağıdaki gibi bir liste ile karşılaşacaksınız:

```
$ cat /etc/ld.so.conf
/lib
/usr/lib
/usr/X11R6/lib
/usr/i486-linuxlibc1/lib
```

ld.so.conf dosyası dinamik yükleyicinin paylaşılan kütüphaneler için bakacağı bütün dizinlerin listesini içermektedir (otomatik olarak yüklenen /lib ve /usr/lib'in yanında)

```
ld.so.cache
```

Ancak dinamik yükleyicinin bu bilgiyi görebilmesi için bu bilginin ld.so.cache dosyası içerisine dönüştürülmesi gerekir. Bu işlem ldconfig komutu çalıştırılarak yapılmaktadır.

```
# ldconfig
```

ldconfig tamamlandıktan sonra /etc/ld.so.conf ile yaptığınız tüm değişiklikleri en son haliyle gösteren /etc/ld.so.cache dosyasına sahipsiniz demektir. Bundan sonra dinamik yükleyici paylaşılan kütüphaneleri ararken artık sizin /etc/ld.so.conf dosyasına eklediğiniz yeni dizinlere de bakacaktır.

## 1.4 ldconfig Hakkında İpuçları

ldconfig komutunun görebildiği tüm paylaşılan kütüphaneleri görüntülemek için şu şekilde yazmak gerekir:

```
# ldconfig -p | less
```

Paylaşılan kütüphane yollarını konfigüre etmek için kullanışlı bir yöntem daha vardır. Bazen dinamik yükleyiciden paylaşılan kütüphaneleri ararken /etc/ld.so.conf içerisinde yazılı yollara bakmadan önce sizin belirlediğiniz özel bir dizin içerisine bakmasını isteyebilirsiniz. Bu yeni yüklenmiş kütüphaneler, çalışmayan eski bir uygulamayı kullanmak için gerekebilir.

## 1.5 LD\_LIBRARY\_PATH

Dinamik yükleyiciye önce belirli bir dizine bakmasını söylemek için LD\_LIBRARY\_PATH değişkenini bu istediğiniz dizinler ile set etmeniz gerekir. Birden fazla sayıda yolu ; ile ayırabilirsiniz. Örneğin:

```
# export LD_LIBRARY_PATH="/usr/lib/old:/opt/lib"
```

LD\_LIBRARY\_PATH değişkeni export edildikten sonra , o andaki kabukta başlatılan çalıştırılabilirler eğer mümkünse /usr/lib/old ya da /opt/lib altındaki kütüphaneleri kullanacaklar, olmazsa /etc/ld.so.conf içerisinde belirtilen yollarda gerekli kütüphaneleri arayacaklardır.

Linux paylaşılan kütüphaneleri ile ilgili genel bilgilerimizi burada noktıyoruz. Bu konuda ayrıntılı bilgiyi man ldconfig ya da man ld.so komutları ile öğrenebiliriz.

# 2 Uygulamaları Kaynaktan Derlemek, Giriş

Diyeelim ki elinizde sisteminize yüklemek istediğiniz bir uygulama var. Bu uygulamanın en yeni sürümünü kurmak istiyorsunuz. Ancak bu sürüm henüz rpm paket formatı haline gelmemiş olsun. Uygulamayı sadece kaynak kod halinde bulabildiğinizi düşünelim. Bu durumda yapmanız gereken şey bu uygulamayı kaynağından derlemek olacaktır. Bu bölümde bu işlemin nasıl yapılacağını göreceğiz.

## 2.1 Dosyaları İndirmek (downloading)

İlk işiniz derlemek için ihtiyacınız olan kaynak dosyaları bulmak ve indirmek olacaktır. Bu dosyaları muhtemelen tar.gz, tar.Z, tar.bz2 ya da tgz uzantısı halinde arşivlenmiş ve sıkıştırılmış olarak bulacaksınız. Arşivi, kullandığınız tarayıcı ya da ftp uygulaması ile indirmelisiniz. Eğer uygulama ile ilgili bir web sayfası varsa, kurulum bilgileri hakkında döküman bulabilmeniz açısından bu siteyi ziyaret etmekte fayda vardır. Yükleyeceğiniz program o anda sisteminizde olan ya da olmayan başka programların varlığını gerektirebilir. Eğer bu başka programların sisteminizde olmadığından eminseniz öncelikle bunları temin etmeli, paket ya da kaynaklarının kullanarak sisteminize yüklemelisiniz. Bundan sonra asıl uygulamayı kurmak için artık hazırsınız demektir.

## 2.2 Dosya Arşivleme ve Sıkıştırmaya İlişkin Kısa Açıklama

tar komutu yardımıyla dosyalarınızı arşivleyebilir, oluşturulmuş bir arşiv içerisine yeni dosyalar ekleyebilir ve arşivleri açabilirsiniz. Aşağıdaki örnekte önce /home/knoppix altında "temp" adında bir dizin ve bu dizin içerisine file1, file2 adında iki dosya ile dir1 adında başka bir dizin ve en son olarak da bu dir1 dizini içerisine de file1 adında bir dosya açalım. Bundan sonra bu temp dizinini arşiv haline getirmek için şunu yazalım:

```
# tar -cf arxiv.tar ./temp
# ls
arxiv.tar  Desktop  temp  tmp
```



Burada -c parametresi arşiv yaratılması için kullanılır. -f parametresi de kendisinden sonra gelecek olan ismi yeni arşivin ismi olarak algılar. Şimdi yine /home/knoppix altında temp2 diye bir dizin oluşturalım ve bu dizinin içerisine arşivi açalım:

```
# mkdir temp2
# ls
arsiv.tar  Desktop  temp  temp2  tmp
# cd temp2
# tar -xf ../arsiv.tar
# ls
temp
# cd temp
# ls
dir1  file1  file2
```

Bu örnekte de gördüğümüz gibi arşivimizi aynı şekliyle -x parametresi ile açmış olduk. Şimdi diyelim ki arşivimize yeni bir dosya eklememiz gerekiyor. Bunun için ilk akla gelen şey arşivi açıp yeni dosyamızı ekleyip sonra yeniden arşivlemekse buna hiç gerek yok. Çünkü -r parametresi yardımıyla arşivi açmadan yeni bir dosya eklememiz mümkün:

```
# touch ekle.txt
# tar -rf arsiv.tar ekle.txt
```

Şimdi de arşivimizde hangi dosyaların bulunduğuna -t parametresi yardımıyla bir göz atalım:

```
# tar -tf arsiv.tar
./temp/
./temp/dir1/
./temp/dir1/file1
./temp/file2
./temp/file1
ekle.txt
```

Dosya sıkıştırmaya yarayan araçlardan en yaygın olanlardan birtanesi de gzip'dir. gzip ile bir dosyayı sıkıştırabilir ve gunzip ile sıkıştırılmış dosyayı açabilirsiniz:

```
# gzip ekle.txt
# ls
arsiv.tar  Desktop  ekle.txt.gz  temp  temp2  temp3  tmp
# gunzip ekle.txt.gz
# ls
arsiv.tar  Desktop  ekle.txt  temp  temp2  temp3  tmp
#
```

Sonuç itibarıyla bir dizin içerisinde yer alan dosyaları en iyi şekilde paketlemek için bunları önce arşivlemek sonra da sıkıştırmak gerekir. Az önce oluşturduğumuz /temp dizini bu şekilde uygun bir tar.gz paketi haline getirelim:

```
# tar -cf arsiv.tar ./temp
# ls
arsiv.tar  Desktop  temp  temp2  temp3  tmp
# gzip arsiv.tar
# ls
arsiv.tar.gz  Desktop  temp  temp2  temp3  tmp
#
```

## 2.3 Paketi Açmak

Kaynak arşivi açmak nispeten daha kolaydır. Eğer arşivinizin uzantısı tar.gz, tar.Z ya da .tgz biçiminde ise arşivi açmak için şu komutu kullanabilirsiniz:

```
$tar -xzf archivename.tar.gz
```

x: açmak için

v: görünür durumda açmak için (açılan dosyalar ekranda yazılır halde)

f: Bundan sonra açılacak arşiv dosya ismi yazılacağı anlamına geliyor

Bundan sonra bütün kaynak dosyaları tek bir dizin içerisinde yeralacak şekilde arşiv dosyası açılacaktır. Bu sayede, bir arşivi açtığınız zaman, o anda çalıştığınız dizin içerisine bu arşivde yer alan dosyalar dolmayacak, onun yerine daha düzenli bir biçimde ayrı bir dizin içerisine yer alacaklardır.

## 2.4 Arşivleri Listelemek

Çalışmalarınız sırasında, açıldığı zaman çalıştığınız dizin içerisine yüzlerce dosya dolduracak bir arşivi açmanız gerekecektir. Bir çok arşiv dosyası böyle açılmadığı halde bazen bu durumla karşılaşabilirsiniz. Eğer arşivinizi açıldığında içeriğinde yer alan dosyaların hepsinin tek dizin içerisinde toplanacağından emin olmak istiyorsanız, arşivin içeriğini aşağıdaki komut ile görebilirsiniz:

```
$ tar -tzvf archivename.tar.gz | more
```

t: text listeleme için. Herhangi bir açma işlemi yapılmayacağı anlamına geliyor.

Eğer arşiv listesinin sol tarafında ortak bir dizin belirtilmemişse, arşivinizi yeni açacağınız bir dizin içerisine taşımanız ve açma işlemini burada gerçekleştirmeniz gerekmektedir. Aksi takdirde işlem çok karışık bir hal alacaktır.

## 2.5 bzip2 ile Sıkıştırılmış Arşivleri Açmak

Arşiv dosyanız .tar.bz2 formatında olabilir. Bu uzantıya sahip dosyalar bzip2 ile sıkıştırılmış dosyalardır. Bzip2 genelde gzip den daha iyi bir sıkıştırma yapmaktadır. bzip2 nin tek dezavantajı sıkıştırma ve açma işlemini daha yavaş gerçekleştirmesi ve gzip'in kullandığından daha fazla bellek alanı kullanmasıdır. Ancak yeni nesil bilgisayarlarda bu ayrıntı artık önemsizdir. Zaten bzip2'nin zaman geçtikçe daha popüler olarak kullanılması da bu gerçeği ispatlamaktadır.

bzip2'nin artan ününden dolayı birçok linux dağıtımı artık yamalı halde tar versiyonlarını barındırmaktadır. Bu versiyonların kullanımında y veya i seçeneği ile tar uygulamasına dosyanın bzip2 ile sıkıştırıldığını ve açmak için de yine bzip2 kullanacağını belirtebiliriz. Sahip olduğunuz tar uygulamasının yamalı versiyonu olup olmadığını anlamak için şunu yazın:

```
$ tar -tyvf archive.tar.bz2 | more
```

ya da

```
$tar -tivf archive.tar.bz2 | more
```

Bu komutlardan herhangi birisi çalışmazsa (ve tar yanlış parametre kullandığınızı söylüyorsa) bile hala yapabileceğimiz şeyler vardır. Okumaya devam ediniz :)

bzip2 Boruhatları Sizin tar uygulamanız kısayollar ile bzip2 dosyalarını açamıyorsa ne yapmak gerekir? Neyse ki sisteminiz GNU versiyonu tar uygulaması içermese bile bu işi yapabilmek için nerdeyse tüm Unix sistemlerinde kullanılabilen kolay bir yol mevcuttur. bzip2 dosyasının içeriğini görüntülemek için bir boruhattı yaratabiliriz:

```
# ls
# cat ../arsiv.tar.bz2 | bzip2 -d | tar -t | more
./temp/
./temp/dir1/
./temp/dir1/file1
./temp/file2
./temp/file1
# cat ../arsiv.tar.bz2 | bzip2 -d | tar -x | more
# ls
temp
#
```

## 2.6 bzip2 Boruhatları (devamı)

Bundan önceki iki örnekte, arşiv dosyamızın içeriğini görüntüleyen ve onu açan standart bir Unix boru hattı yarattık. tar uygulaması, kendisine stdin üzerinden gelen dosyayı açarak disk üzerine değil onun yerine more programına gönderdi.

Eğer bu boru hattı yöntemini kullandığınızda sistem bzip2 diye bir uygulamayı bulamadığını bildiriyorsa muhtemelen bzip2 sisteminizde yüklenmemiştir. Uygulama ile ilgili kaynakları [redhat.com](http://redhat.com)<sup>1</sup> ve [debian.org](http://debian.org)<sup>2</sup> adreslerinde bulabilirsiniz. bzip2 nin sisteme kurulmasından sonra (bu dökümanı takip ederek) ilk önce kurmak istediğiniz uygulamayı açıp kurma işlemini tamamlayabileceksiniz.

## 2.7 Kaynakları İncelemek

Kaynaklarınızı açtıktan sonra açma işlemini yaptığınızı dizine girmek ve buradaki dosyaları kontrol etmek isteyeceksiniz. Genelde kurulumla ilgili bir dosyayı aramak akıllıca olacaktır. Genel olarak bu bilgiler ana kaynak dizini içerisinde yer alan README ya da INSTALL dosyaları içerisinde yazılıdır. Pratikte sizin çalıştığınız platforma özel kurulum bilgilerini README.platform ya da INSTALL.platform (burada platform kullandığınız işletim sisteminin belirtiyor) dosyalarında bulursunuz.

## 2.8 Konfigürasyon

Modern kaynakların birçoğu ana kaynak dizini içerisinde bir konfigürasyon script dosyası barındırır. Bu script özel olarak kaynakları kurmak için tasarlanmıştır. Böylece sisteminiz üzerinde bu kaynaklar düzgün bir şekilde derlenir. Çalıştırıldığı zaman, script sisteminizi inceler yeteneklerini hesaplar ve kaynakların kurulumu ve yüklenmesine ilişkin komutları barındıran Makefile dosyaları oluşturur.

Bu script her zaman "configure" olarak adlandırılır. Eğer ana kaynak dizini içerisinde bir configure dosyası bulursanız iyi bir ihtimalle bu dosya sizin kullanmanız için oraya koyulmuştur. Eğer configure skriptini bulamıyorsanız, kaynak dosyalarınız muhtemelen çeşitli sistemler üzerinde çalışacak şekilde dizayn edilmiş standart Makefile dosyası ile gelmiştir. Bu durumda bundan sonra gelecek olan konfigürasyon adımlarını geçip dökümanı "make" hakkında konuştuğumuz yerden itibaren takip edebilirsiniz.

## 2.9 configure Kullanmak

configure skriptini çalıştırmadan önce bu uygulama hakkında fikir sahibi olmakta yarar var. ./configure -help yazarak programınız için mevcut tüm konfigürasyon seçeneklerini görüntüleyebilirsiniz. Özellikle -help çıktısının en üstünde olanlar ve gördüğünüz diğer seçeneklerin birçoğu neredeyse her configure skripti içerisinde yer alan standart seçeneklerdir. Sonlarda yer alan seçenekler ise sizin derlemek istediğiniz pakete özgü seçeneklerdir. Bu seçenekleri incelemek ve hangilerini kullanıp kullanmayacağımıza karar vermek gerekir.

## 2.10 -prefix Seçeneği

GNU autoconfig tabanlı birçok configure skripti, programı nereye kuracağımızı kontrol etmenize olanak sağlayan -prefix seçeneğine sahiptir. Öntanımlı olarak kaynakların çoğu /usr/local dizini içerisine yüklenir. Bunun anlamı binary'ler /usr/local/bin, kılavuz sayfaları /usr/local/man vb. şeklinde dosyaların yerleştirileceğidir. Normalde bizim istediğimiz de budur; /usr/local içerisinde derlediğimiz programlar saklanacaktır.

## 2.11 -prefix Kullanmak

Eğer kaynakları başka bir yere yüklemek isterseniz, (mesela /usr içerisine) -prefix=/usr seçeneğini configure skriptine göndermek gerekir. Aynı şekilde /opt dizini içerisine kurulum yapmak isterseniz de yine configure skriptine -prefix=/opt seçeneğini göndermemiz gerekir.

## 2.12 Peki ya FHS ne diyor?

Bazen bir program, kurulumda öntanımlı olarak dosyalarını disk üzerinde standart olmayan bir yere koymak isteyebilir. Yani özel olarak bir kaynak arşivi Linux Dosyasistemi Hiyerarşisine uygun olmayan

<sup>1</sup><http://sources.redhat.com/bzip2>

<sup>2</sup><http://packages.debian.org/stable/utils/bzip2>

kurulum yollarını kullanıyor olabilir. Neyse ki configure scripti ile kullandığımız prefix seçeneği sadece kurulum yolunu değiştirmemize izin vermekle kalmaz aynı zamanda kılavuz sayfaları gibi çeşitli sistem parçalarının da kurulum yolunu değiştirmemize olanak sağlar. Birçok kaynak arşivi henüz FHS uyumlu olmadığından, prefix seçeneğinin bu özelliği çok kullanışlı olmaktadır. Muhtemelen her zaman kaynak paketinizi FHS uyumlu yapabilmek için `--mandir=/usr/share/man` ve `--infodir=/usr/share/info` seçeneklerini configure scriptine göndermeniz gerekmektedir.

## 2.13 Configure

Değişik konfigürasyon seçeneklerine bakıp kullanacaklarımızı seçtikten sonra, configure programını çalıştırmanın zamanı gelmiştir. Configure programını çalıştırırken, komut satırından bazı seçenekler girmek durumunda kalabileceğinizi (genellikle öntanımlı seçenekler çalışır fakat bu tam istediğiniz sonucu elde edeceğiniz anlamına gelmez) unutmayınız.

configure komutunu çalıştırmak için

```
$ ./configure <seçenekler>
```

```
$ ./configure
```

veya

```
$ ./configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-thr
```

şeklinde yazabilirsiniz. İhtiyaç duyduğunuz seçenekler, konfigüre edeceğiniz belli paketlere bağlıdır. Configure, çalıştırdığınızda öncelikle bir süre için sisteminizde hangi araçlar ve hangi seçenekler olduğunu algılar ve bunları ekrana basar.

## 2.14 config.cache

Konfigürasyon işlemi sona erdiğinde configure betiği, bütün konfigürasyon verisini config.cache dosyasında depolar. Bu dosya configure betiği ile aynı yerde durur. Sistemde bir değişiklik yaptıktan sonra eğer ./configure komutunu tekrar çalıştırmanız gerekirse önce `rm config.cache` dosyasını silmeniz fayda var, aksi takdirde configure öncelikle bu bilgileri kullanacaktır.

## 2.15 configure ve Makefile Dosyaları

configure betiği çalışmasını tamamladıktan sonra kaynakları derlemenin zamanı gelmiştir. make adında bir program kullanılarak bu adım gerçekleştirilebilir. Eğer yazılım paketinizde configure betiği varsa ve bunu çalıştırdıysanız, bu betik Makefile denilen ve sisteminiz için özelleştirilmiş dosyaları yaratacaktır. Bu dosyalar make programına kaynakları nasıl kuracağını, binary'leri, kılavuz dosyalarını ve destek dosyalarını nasıl yükleyeceğini anlatır.

## 2.16 Makefile Giriş

Makefile dosyaları genel olarak makefile ya da Makefile olarak adlandırılır. Kaynak dosyaların bulunduğu dizinde normalde bir tane makefile olacaktır. Hazır yaratılmış Makefile dosyaları kullandığınız program gibi hedeflerin tam olarak nasıl kurulacağı gibi kuralları içerir. make programı tüm komutların çalışması gereken sırayı ve çalışma şekillerini ortaya koyar.

## 2.17 make Programını Çağırarak

make programını çağırmak çok kolay bir işlemdir. Çalıştığımız dizin içerisindeyken komut satırında sadece "make" yazmanız yeterlidir. Bundan sonra make programı çalıştığımız dizinde yer alan makefile ya da Makefile olarak adlandırılan dosyayı bulacak ve yorumlayacaktır. Eğer sadece make yazarsanız, program öntanımlı hedefi kuracaktır. Program geliştiriciler normalde kendi makefile dosyalarını oluştururlar ve böylece öntanımlı hedef tüm kaynağı derleyecektir.

```
$ make
```

Bazı makefile dosyalarında öntanımlı hedef yoktur ve derleme işleminin başlaması için sizin bir tane belirtmeniz gerekir.

```
$ make all
```

Bu komutlardan bir tanesini yazdıktan sonra bilgisayar programınızı object koda çevirirken biraz zaman geçecektir. Hiçbir hata olmadığını varsayarsak, bu aşamadan sonra derlenmiş programınızı artık sisteminize kurmaya hazırsınız demektir.

## 2.18 Kurulum

Program derlendikten sonra bir önemli adım daha kalmış demektir: kurulum. Program derlenmiş olsa bile kullanıma hazır değildir. Programın tüm bileşenleri kaynak dizinden sistem üzerinde kullanılacağı doğru dizinler içerisine kopyalanmalıdır. Mesela tüm binary dosyaları `/usr/local/bin` ve tüm kılavuz sayfaları `/usr/local/man` vb. dizinleri içerisine yüklenmelidir.

Yazılımı kurmak için root kullanıcısı olmanız gerekir. Bunun için sisteme en baştan root olarak girebilir ya da ayrı bir terminalde su yazarak root kullanıcısı olabilirsiniz. Bundan sonra o andaki kabuk oturumundan "exit" yazarak ya da control-D ye basarak çıkana kadar root haklarına sahipsiniz demektir. Eğer zaten root kullanıcısı iseniz, kurulum işlemine devam edebilirsiniz.

## 2.19 make install

Kaynakları kurmak için ana dizinde sadece aşağıdaki komutu yazmanız yeterlidir:

```
# make install
```

"make install" yazmakla, make programına "install" hedefini gerçekleştirmesini anlatmış oluruz. Bu hedef geleneksel olarak en son yaratılan kaynak dosyaların sisteminiz üzerinde doğru yerlere kopyalanması için kullanılır. Böylece program kullanılabilir hale gelecektir. Eğer herhangi bir -prefix seçeneği belirlemezseniz, büyük bir ihtimalle bir kaç dosya ve dizin `/usr/local` dizin ağacı içerisine kopyalanacaktır. Programınızın büyüklüğüne bağlı olarak kurulum hedefi tamamlanması için bir kaç saniyeden birkaç dakikaya kadar zaman geçebilir.

Ayrıca dosyaları kolaylıkla kopyalamak için, make install kurulan dosyaların doğru izin ve sahiplik haklarına sahip olduğunu da gözetecektir. make install başarıyla tamamlandıktan sonra program artık yüklenmiştir ve kullanıma hazırdır (ya da neredeyse hazırdır.)

## 2.20 Program bir defa yüklendikten sonra

Program yüklendikten sonra ne yapmak gerekiyor? Tabi ki onu çalıştırmak gerekmektedir. Henüz yüklediğiniz programın nasıl çalıştırılacağı konusunda bilgi sahibi değilseniz aşağıdaki gibi yazarak programın kılavuz sayfasını inceleyebilirsiniz:

```
$ man [program_adı]
```

Programınızın ancak bazı gerekli ayar adımlarının geçilmesinden sonra çalışması da olasıdır. Örneğin eğer web sunucusu yüklediyseniz, onu sistem açılışı sırasında da otomatik olarak çalışacak şekilde ayarlamanız ya da scriptlerin localhost altında çalışması için konfigürasyon dosyası içerisinde birtakım ayarların yapılması gerekir. Ayrıca yine bazı programlar için de öncelikle `/etc` altında yer alan programla ilgili konfigürasyon dosyasını da ayarlamanız gerekebilir.

## 2.21 İşte bu kadar!

Artık belirli bir yazılım paketini kendi kaynağından yüklemiş bulunuyorsunuz. Aşağıdaki şekilde yazarak programınızı çalıştırabilirsiniz:

```
$ program_adı
```

Tebrikler!

## 2.22 Olası Problemler

configure, make hatta make install betiklerinin bir hata ile dönmesi çok karşılaşılan bir durumdur. Aşağıdaki bölümlerde bu bilinen ortak problemlerden bahsedeceğiz.

## 2.23 Eksik Kütüphaneler

Bazen belirli bir kütüphanenin eksik olması yüzünden configure betiğinin çalışmadığını görebilirsiniz. Kurulum işlemine devam edebilmek için, o andaki kurulum işlemini geçici olarak bir kenara bırakıp programın ihtiyacı olan bu kütüphaneler için bazı kaynak ya da binary paketleri bulmak gerekecektir. Doğru kütüphane yüklendikten sonra configure ve make betikleri işlemlerini tamamlayabileceklerdir.

## 2.24 Diğer Problemler

Bazen nasıl düzeltileceğini bilmediğiniz problemler ile karşılaşabilirsiniz. Unix/linux konusunda tecrübeniz arttıkça configure ve make işlemleri sırasında karşılaşacağınız anlaşılması daha zor gibi görünen problemlere tanı koyabileceksiniz.

Bazen yüklenen bir kütüphanenin çok eski ya da çok yeni olmasından kaynaklanan hatalar olabilir. Ya da program geliştiricilerinin yazdıkları programı sizin sisteminizde çalışmayacağını öngörememiş olmasından dolayı bu tür hatalarla karşılaşabilirsiniz.

## 2.25 Diğer Problemler (devamı)

Bu tür problemler ile karşılaştığınızda en iyi nereden yardım alabileceğinizi belirleyiniz. Eğer bu sizin programı kaynağından derlemek konusunda ilk denemeniz ise, önce derlemesi sorunsuz olacak başka bir program ile bu denemeyi yapmakta yarar vardır. Bu diğer basit programı bir defa derlediğinizde asıl programın derlenmesi sırasında meydana gelen problemin hangi noktada olduğu konusunda fikir sahibi olabilirsiniz.

# 3 Paket Yönetimi Kavramları, Paket Yönetimi Avantajları

Sisteminize bir yazılımı yüklemek için uygulamaları kaynaktan derlemenin yanında başka bir yöntem daha vardır. Yazılım paketlerini yüklemek, güncellemek ya da kaldırmak amacıyla tüm linux dağıtımları belli şekillerde paket yönetimleri sunmaktadır. Paket yönetimi, uygulamaların kaynaktan derlenmesine kıyasla bazı avantajlar sunar:

\* Kurulum ve kaldırma kolaylığı \* Mevcut kurulmuş paketleri güncelleme kolaylığı \* Konfigürasyon dosyalarının korunması \* Yüklenen dosyalarının takibinin basitleştirilmesi

## 3.1 Paket Yönetiminin Dezavantajları

En popüler paket yönetim araçlarını kullanmaya başlamadan önce paket yönetiminden hoşlanmayan linux kullanıcılarının da olduğunu belirtmekte fayda var. Bu kullanıcılara göre:

\* Belli bir sistem için oluşturulan Binary'ler daha iyi performans göstermektedir. \* Yüklenen bir paketin ihtiyaç duyduğu kütüphanelerden arındırılmış olması ciddi bir sorundur \* Paketlerin yaratılması oldukça zordur \* Paket veritabanında meydana gelecek bir bozulma sisteme zarar verebilir

Sözkonusu olumsuzluklar olsa da bir çok kullanıcıya göre paket yönetiminin avantajları dezavantajlarına baskın gelmektedir. Ayrıca yukarıda bahsedilen her kusur için de uygun bir karşıt fikir de mevcuttur. Çoklu paketler farklı sistemler için optimize edilebilirler ve paket yönetici araçları da bağımlı kütüphaneleri belirleyecek şekilde geliştirilebilir, veritabanları diğer dosyalara bağlı olarak yeniden oluşturulabilir ve bir paketin yaratılması için başta harcanan efor paketin daha sonra güncellenmesi ve kaldırılabilmesi kolaylığı sayesinde hafifletilebilir.

# 4 rpm the (R)ed Hat (P)ackage (M)anager, rpm'e Giriş

Red Hat'in 1995 de rpm'i ortaya koyması Linux dağıtımları için çok büyük bir adım oldu. Bu sadece Red Hat linux üzerinde paket yönetimini sağlamakla kalmayıp sahip olduğu GPL lisansı sayesinde açık kaynak paketleme açısından bir standart haline geldi. rpm için çeşitli grafik arayüzleri ve kullanımı kolaylaştıran araçlar olsa da öntanımlı olarak komut satırı üzerinde bir arayüze sahiptir. Bu bölümde örnek olarak Xsnow programını kullanarak rpm için en çok bilinen komut satırı işlemlerini inceleyeceğiz.

Eğer bundan sonrasını takip etmek istiyorsanız birçok rpm tabanlı dağıtım ile çalışabilen aşağıdaki rpm'i indiriniz:

`xsnow-1.41-1.i386.rpm`

Not: Bu paketi indirmek isterseniz kullanabileceğiniz kaynaklardan bir tanesi de rpmfind.net<sup>3</sup> olabilir.

Not: Eğer burada kullanılan çeşitli rpm terimleri aklınızı karıştırıyorsa şunu hatırlayınız: "bir rpm" bir rpm paketini temsil ederken "rpm" daima bir programı anlatmaktadır.

## 4.1 Bir rpm Yüklemek

Şimdi Xsnow rpm'i rpm -i kullanarak yükleyebiliriz. Tabi bu işlem için öncelikle root kullanıcısı olmak gerekir:

```
$ su
Password:
# rpm -i xsnow-1.41-1.i386.rpm
```

Eğer bu komut sonunda hiç bir çıkış üretilmiyorsa komutumuz çalışmış demektir. X masaüstünde eğlenmek istiyorsanız Xsnow uygulamasını çalıştırmanız gerekir. Kişisel olarak genelde bir rpm yüklediğimiz görsel bir geri besleme bekleriz. Bunun için -h (işlemi gösteren işaret) ve -v (görünür mod) seçeneklerini kullanmamız gerekir:

Bunun için önce rpm2i kaldıralım ve sonra -v ve -h seçenekleri ile yeniden yükleyelim:

```
# rpm -iv xsnow-1.41-1.i386.rpm
warning: xsnow-1.41-1.i386.rpm: V3 DSA signature: NOKEY, key ID b1f6e46c
Preparing packages for installation...
xsnow-1.41-1
# rpm -e xsnow
# rpm -ivh xsnow-1.41-1.i386.rpm xsnow
warning: xsnow-1.41-1.i386.rpm: V3 DSA signature: NOKEY, key ID b1f6e46c
Preparing...
1:xsnow
```

## 4.2 Bir rpm'i yeniden Yüklemek

Eğer aşağıdaki komutu tekrar çalıştırırsanız, aşağıdaki gibi bir mesaj ile karşılaşacaksınız:

```
# rpm -ivh xsnow-1.41-1.i386.rpm
warning: xsnow-1.41-1.i386.rpm: V3 DSA signature: NOKEY, key ID b1f6e46c
Preparing...
package xsnow-1.41-1 is already installed
```

Bir rpm tekrar yüklemek isteyeceğiniz durumlar olabilir. Örneğin /usr/X11R6/bin/xsnow binary'sini yanlışlıkla sildiğinizde bu durum ile karşı karşıya kalırsınız. Bu durumda rpm'i rpm -e ile kaldırıp tekrar yüklemelisiniz.

```
# rpm -e xsnow
```

Dikkat ediniz; birazdan göreceğiniz örnekte rpm den gelen bilgi mesajı paketin sistemden silinmesine engel teşkil etmez. Çünkü bu bilgiye göre paket zaten sistemde bulunmamaktadır.

```
# rpm -e xsnow
error: package xsnow is not installed

# rpm -ivh xsnow-1.41-1.i386.rpm
warning: xsnow-1.41-1.i386.rpm: V3 DSA signature: NOKEY, key ID b1f6e46c
Preparing...
1:xsnow
```

## 4.3 Bir rpm i Zorlama ile Yüklemek

Bazen sistemde bir programa bağlı olarak çalışan başka programlar varken, o programı sistemden kaldırmak zorluk yaratabilir. Örneğin Xsnow a bağlı olarak çalışan bir x-amusements rpm'i yüklediğinizi varsayalım. Bu durumda rpm -e ile Xsnow'u sistemden kaldıramazsınız.

<sup>3</sup><http://www.rpmfind.net>

```
# rpm -e xsnow
error: removing these packages would break dependencies:
/usr/X11R6/bin/xsnow is needed by x-amusements-1.0-1
```

Bu durumda Xsnow uygulamasını `-force` seçeneği ile yeniden yükleyebilirsiniz. Şimdi zaten yüklü bir paketi `-force` kullanmadan yüklemeye çalışalım:

```
# rpm -ivh xsnow-1.41-1.i386.rpm
warning: xsnow-1.41-1.i386.rpm: V3 DSA signature: NOKEY, key ID b1f6e46c
Preparing... ##### [100%]
package xsnow-1.41-1 is already installed
```

Şimdi de `-force` kullanalım:

```
# rpm -ivh --force xsnow-1.41-1.i386.rpm
warning: xsnow-1.41-1.i386.rpm: V3 DSA signature: NOKEY, key ID b1f6e46c
Preparing... ##### [100%]
1:xsnow ##### [100%]
```

#### 4.4 `-nodeps` Seçeneği İle Yüklemek ve Kaldırmak

`-force` kullanımına alternatif olarak bağımlılıkları kontrol etmeden de rpm'i `-nodeps` seçeneği ile sistemden kaldırabilirsiniz. Bu kullanım da nadiren işe yarayabilir.

```
# rpm -e --nodeps xsnow
# rpm -ivh xsnow-1.41-1.i386.rpm xsnow
```

Bir rpm'i yüklerken de `-nodeps` seçeneğini kullanabilirsiniz. Aslında düzgün kurulum ve kullanım açısından `-nodeps` seçeneğinin kullanımı pek tavsiye edilmese de bazen gerekebilir.

```
# rpm -ivh --nodeps xsnow-1.41-1.i386.rpm xsnow
```

#### 4.5 Paketleri Güncellemek

Diyelim ki elinizde Xsnow 1.42 versiyonu için yeni bir rpm olsun. Bu durumda mevcut kurulumunuzu güncellemek isteyeceksiniz. Eğer rpm `-ivh -force` seçeneğini kullanırsanız kurulumu gerçekleştirirsiniz. Ancak bu durumda da rpm in kendi veritabanında her iki versiyonun da kurulu olduğu bilgisini bulacak ve paketi ya iki defa kuracak ya da bununla ilgili bir çıkışta hatası verecektir.

```
# rpm -ivh xsnow-1.42-8.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
Preparing... ##### [100%]
file /usr/X11R6/bin/xsnow from install of xsnow-1.42-8 conflicts with
file from package xsnow-1.41-1
file /usr/X11R6/man/man1/xsnow.1x.gz from install of xsnow-1.42-8
conflicts with file from package xsnow-1.41-1
```

Bunun yerine kurulumunuz güncellemek için rpm `-U` seçeneğini kullanmanız gerekir.

```
# rpm -Uvh xsnow-1.42-8.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
Preparing... ##### [100%]
1:xsnow ##### [100%]
```

Burada küçük bir ipucu verelim: Genelde nadiren rpm `-i` seçeneğini kullanırız. Çünkü rpm `-U` zaten sistemde paket yoksa yine de kurulumu gerçekleştirir. Ama `-i` seçeneğinden farklı olarak da eğer paket sistemde varsa varolan sürümü günceller. Bu kullanım özellikle komut satırında bazıları kurulmuş bazıları da henüz kurulu olmayan birden fazla paket yazdığımız zaman daha da kullanışlı olacaktır. Diyelim ki sistemimizde sadece xsnow-1.41 versiyonu yüklü olsun. Şimdi hem xsnow-1.42 hem de xfisht tank rpm'lerini yüklemek için `-U` seçeneğini kullanalım.

```
# rpm -Uvh xsnow-1.42-8.i386.rpm xfishtank-2.1tp-1.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
Preparing... ##### [100%]
1:xfishtank ##### [ 50%]
2:xsnow ##### [100%]
```



## 4.6 rpm -q İle Sorgulama Yapmak

Örneklerde de farkedeceğiniz gibi rpm kurulumunda paketler için tam isim ancak rpm kaldırılması sırasında sadece isim yazılmaktadır. Bunun sebebi daha önce de bahsettiğimiz gibi rpm'in kendi içerisinde o anda kurulu olan paketlere ilişkin bir veritabanına sahip olmasıdır. Bu yüzden kurulu paketler için sadece isim kullanmak yeterlidir. Mesela rpm'e hangi Xsnow versiyonunun yüklü olduğunu soralım:

```
# rpm -q xsnow
xsnow-1.42-8
```

Aslında rpm paketlerle ilgili olarak sadece isim ve sürüm bilgisinden daha fazlasına sahiptir. rpm -qi kullanarak Xsnow hakkında daha fazla şey öğrenebiliriz:

```
# rpm -qi xsnow
Name       : xsnow                      Relocations: (not relocateable)
Version    : 1.42                      Vendor: Red Hat, Inc.
Release    : 8                        Build Date: Mon 26 Aug 2002 11:51:09 PM EEST
Install date: Thu 22 May 2003 12:37:02 PM EEST Build Host: daffy.perf.redhat.com
Group      : Amusements/Graphics       Source RPM: xsnow-1.42-8.src.rpm
Size       : 111829                    License: MIT
Signature  : DSA/SHA1, Wed 04 Sep 2002 12:45:26 AM EEST, Key ID 219180cddb42a60e
Packager   : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL        : http://www.euronet.nl/~rja/Xsnow/
Summary    : An X Window System based dose of Christmas cheer.
Description :
The Xsnow toy provides a continual gentle snowfall, trees, and Santa
Claus flying his sleigh around the screen. Xsnow is only for the X
Window System, though; consoles just get coal.
```

## 4.7 rpm -ql ile Dosyaları Listelemek

rpm tarafından kullanılan veri tabanı çok fazla bilgi içermektedir. Yukarıda da gördüğümüz gibi burada yüklenen paketin sürümünü ve bununla ilişkili diğer bilgileri gözlemleyebilmekteyiz. rpm ayrıca rpm -ql kullanımı ile verilen bir kurulu pakete ait dosyaları da listeleyebilir:

```
# rpm -ql xsnow
/usr/X11R6/bin/xsnow
/usr/X11R6/man/man1/xsnow.1x.gz
/usr/share/doc/xsnow-1.42
/usr/share/doc/xsnow-1.42/README
/usr/share/pixmaps/xsnow.png
```

Bu komutun çıkışı sadece konfigürasyon dosyaları ve dökümantasyon dosyaları ile sınırlandırmak isterseniz -c ve -d seçeneklerini ekleyebilirsiniz. Bu kullanım uzun dosya listelerine sahip olan rpm ler için oldukça kullanışlıdır. Biz bu kullanımı Xsnow uygulamamız için çalıştıralım:

```
# rpm -qld xsnow
/usr/X11R6/man/man1/xsnow.1x.gz
/usr/share/doc/xsnow-1.42/README
```

## 4.8 rpm -qp İle Paketleri Sorgulamak

Yüklemeden önce eğer rpm hakkında mevcut bilgiyi rpm -qi ile öğrenebilseydiniz bu paketi kurup kurmayacağınıza karar verebilirdiniz. Ancak -qi ile sadece yüklü paketler hakkında ayrıntılı bilgi alabilmekteyiz. Aslında rpm -qp bir rpm için veritabanını değil rpm'in kendisini sorgulamaktadır. Bu ana kadar gördüğümüz tüm sorgulamalar kurulu paketlere uygulanabildiği gibi benzer şekilde -p seçeneği ile rpm dosyalarına da uygulanabilir. Burada az önce yaptığımız tüm örnekleri bu sefer -p seçeneği kullanarak tekrar deneyelim:

```
# rpm -qp xsnow-1.42-8.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
xsnow-1.42-8
```

```
# rpm -qpd xsnow-1.42-8.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
/usr/X11R6/man/man1/xsnow.1x.gz
/usr/share/doc/xsnow-1.42/README
```

## 4.9 Tüm Kurulu Paketleri Sorgulamak

-a seçeneği kullanarak sisteminize yüklenmiş tüm paketleri sorgulayabilirsiniz. eğer çıkışı sort üzerinde bir pager uygulamasına boru ile gönderirseniz sisteminizde hangi paketlerin kurulu olduğunu güzel bir şekilde görüntüleyebilirsiniz. Örneğin:

```
# rpm -qa | sort | less
```

[burada çıkışı yazmadık]

Şimdi sistemimizde kaç tane rpm'in yüklü olduğuna bakalım:

```
# rpm -qa | wc -l
641
```

Şimdi de tüm bu rpm'lerin içerisinde kaç tane dosya olduğuna bakalım:

```
# rpm -qa | wc -l
97322
```

Bir ipucu: rpm -qa kullanarak çoklu sistemler için kullanım kolaylığı sağlanabilir. Eğer sıralı çıkışı bir makinadaki dosyaya yönlendirip aynı işi başka makina için de yaparsak, diff programını kullanarak hangi paketlerin farklı olduğunu gözlemleyebiliriz.

## 4.10 Bir Dosyanın Sahibini Bulmak

Bazen verilen bir dosyanın hangi rpm'e ait olduğunu bulmak gerekebilir. Teorik olarak /usr/X11R6/bin/xsnow dosyasına hangi rpm'in sahip olduğunu aşağıdaki gibi bir komut ile ortaya çıkarabilirsiniz:

rpm -qf ile bir dosyanın hangi rpm'e ait olduğunu sorgulayabilirsiniz.

```
# rpm -qf /usr/X11R6/bin/xsnow
xsnow-1.42-8
```

## 4.11 Bağımlılıkları Göstermek

-nodeps benzeri seçenekleri kullanmadığınız sürece rpm normalde bağımlılıkları bozacak işlemleri yapmanıza izin vermeyecektir. Örneğin sisteminizde öncelikle x kütüphaneleri olmadan xsnow yükleyemezsiniz. Bir defa Xsnow yüklediğinizde de öncelikle Xsnow'u kaldırmadan x kütüphanelerini de kaldıramazsınız. Bazen sorun olsa da bu bize rpm'in gücünü göstermektedir. Bunun anlamı bir rpm yüklediğinizde onun direk çalışmasını beklersiniz. rpm zaten sisteminizdeki bağımlılıkları belirlediğinden bu bağımlılıklar yüzünden fazladan iş yapmak istemezsiniz. Bazen bağımlılıkları çözmeye çalışırken, bir paketi -R seçeneği ile kullanmak yararlı olabilir. Böylece sistemde ne olması gerektiğini tam olarak belirleyebilirsiniz.

Örneğin xsnow paketi C kütüphanesine, math kütüphanesine, X kütüphanelerine ve belirli rpm versiyonlarına bağımlıdır.

```
# rpm -qpr xsnow-1.42-8.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
libc.so.6
libc.so.6(GLIBC_2.0)
libm.so.6
libX11.so.6
libXext.so.6
libXpm.so.4
```

Ayrıca -p seçeneğini es geçerek aynı bilgi için kurulu veritabanını da sorgulayabilirsiniz.

```
# rpm -qR xsnow
```

## 4.12 Bir Paketin Bütünlüğünü Sağlamak

Bir web sayfasından ya da ftp sitesinde paket indirdiğimizde güvenlik amacıyla bu paketi kurmadan önce bütünlüğünü kontrol etmek isteyebiliriz. Tüm rpm'ler MD5 sum imzalıdır. Ayrıca bazı yazarlar paketlerinin güvenliğini artırmak amacıyla PGP ve GPG imzaları da kullanırlar. Bir paketin imzasını kontrol etmek için `--checksig` seçeneğini kullanabilirsiniz:

```
# rpm --checksig xsnow-1.41-1.i386.rpm
xsnow-1.41-1.i386.rpm: md5 GPG NOT OK
```

Bir dakika! Bu çıkışa göre GPG imzası geçerli değildir. Neyin yanlış olduğunu anlamak için çıkışı açıklamalı durumda inceleyelim:

```
# rpm --checksig -v xsnow-1.41-1.i386.rpm
xsnow-1.41-1.i386.rpm: MD5
sum OK: 8ebe63b1dbe86ccd9eaf736a7aa56fd8 gpg: Signature made Thu 10 May
2001 01:16:27 AM EDT using DSA key ID B1F6E46C gpg: Can't check
signature: public key not found
```

O halde problemimiz yazarımızın public anahtarını alamamamız. Paketin yazarının web sitesinden anahtarı aldıktan sonra (`rpm -qi` çıkışında web adresi gösterilmektedir) bu key dosyasını import edersiniz. Bir anahtar import edildiğinde sizin sisteminizde keyring denilen ve içerisinde tüm anahtarların tutulduğu bir dosya içerisine yazılır. Böylece kullandığımız, indirdiğiniz döküman ve dosyalar bu anahtarlara göre kontrol edilerek güvenlik sağlanmaktadır.

```
# gpg --import dan.asc
# rpm -K xsnow-1.41-1.i386.rpm
xsnow-1.41-1.i386.rpm: md5 gpg OK
```

Not: Bu kullanım sisteminizde mevcut rpm versiyonuna göre değişebilir.

## 4.13 Kurulmuş Bir Paketin Kontrolünü Sağlamak

Bir paketin bütünlüğünü kontrol etmeye benzer olarak, `rpm -V` ile yüklenmiş dosyalarımızın da bütünlüğünü kontrol edebilirsiniz.

```
# rpm -V xsnow
```

Normalde hiç bir sorun olmadığını anlatmak amacıyla bu komut hiç bir çıkış üretmez. Şimdi işleri biraz daha karıştırıp tekrar deneyelim:

```
# rm /usr/X11R6/man/man1/xsnow.1x.gz
rm: remove regular file '/usr/X11R6/man/man1/xsnow.1x.gz'? Y
# cp /bin/sh /usr/X11R6/bin/xsnow
cp: overwrite '/usr/X11R6/bin/xsnow'? y
[root@localhost erman]# rpm -V xsnow
S.5...T /usr/X11R6/bin/xsnow
missing d /usr/X11R6/man/man1/xsnow.1x.gz
```

Bu çıkış bize Xsnow binary'si için yapılan MD5 sum, dosya boyutu ve mtime testlerinin başarısız olduğunu göstermektedir. Ve bütün kılavuz sayfaları eksik durumdadır! Şimdi bu bozulmuş kurulumu onaralım ve uygulamamızı çalıştıralım:

```
# rpm -e xsnow
[root@localhost erman]# rpm -ivh xsnow-1.42-8.i386.rpm
warning: xsnow-1.42-8.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
Preparing...
##### [100%]
1:xsnow
##### [100%]
[root@localhost erman]# xsnow
Xsnow-1.42, December 14th 2001 by Rick Jansen (rja@euronet.nl)
WWW: http://www.euronet.nl/~rja/Xsnow/
```

<— Şimdi masauztüne bakınız:) —>

#### 4.14 rpm’i Konfigüre Etmek

rpm nadir olarak konfigüre edilmeye ihtiyaç duyar. Eski versiyonlarda run-time operasyonları etkilemek için /etc/rpmrc içerisinde değişiklikler yapabiliyordunuz. Yeni versiyonlarda bu dosya /usr/lib/rpm/rpmrc içerisine taşınmıştır ve sistem yöneticileri tarafından değiştirilebilir değildir. Daha çok çeşitli platformlar için uyumluluk bilgilerini ve bayrakları listeler. Eğer rpm’i konfigüre etmek istiyorsanız /etc/rpm/macros dosyasını değiştirerek bunu yapabilirsiniz. Bu nadiren yapılan bir işlem olduğu için burada ayrıntılı olarak bahsetmeyeceğiz. Aşağıdaki komut ile doğru olan dökümantasyon dosyasını bulabilirsiniz:

```
# rpm -qld rpm | grep macros
```

### 5 Debian Paket Yönetimi, apt-get Giriş

Debian paket yönetimi çeşitli farklı araçlardan meydana gelmektedir. Komut satırı araçlarından olan apt-get ile yeni paketleri en kolay şekilde yükleyebilirsiniz. Örneğin frozen-bubble programını yüklemek için root kullanıcısı olarak aşağıdakini yapınız:

```
apt-get install frozen-bubble
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  frozen-bubble-data
The following NEW packages will be installed:
  frozen-bubble
1 packages upgraded, 1 newly installed, 0 to remove and 214 not upgraded.
Need to get 5225kB of archives. After unpacking 291kB will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.de.debian.org/testing/main frozen-bubble-data 1.0.0-3 [5155kB]
8% [1 frozen-bubble-data 423935/5155kB 8%] 13.1kB/s 6m5s
```

Bu çıkışı gözden geçirdiğimizde frozen-bubble programının yüklendiğini, web’den getirildiğini, açıldığını ve kurulduğunu görebilirsiniz.

#### 5.1 Taklit Kurulum

Eğer apt-get kuracağımız paketlerin başka paketlere bağlı olduğunu tespit ederse bu paketleri de otomatik olarak bulup kuracaktır. Son örnekte sadece frozen-bubble kurulmuştur. Çünkü frozen-bubble’ın bağlı olduğu tüm paketler zaten sistemde mevcut idi. Fakat bazen apt-get’in bulması gereken paket sayısı çok fazla olabilir ve kurulumu başlatmadan önce nelerin yüklenmesi gerektiğini görmekte fayda vardır. Bunu tam olarak -s seçeneği ile yapabilirsiniz. Eğer sistemimize bir icq istemcisi olan ickle isimli programı yüklemek istersek, önce hangi paketlerin yüklenmesi gerektiğini öğrenmek için:

```
# apt-get -s install ickle
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  ickle
0 packages upgraded, 1 newly installed, 0 to remove and 214 not upgraded.
Inst ickle (0.3.2-1 Debian:testing)
Conf ickle (0.3.2-1 Debian:testing)
```

Bundan sonra da eğer isterseniz apt-get, paketlerin yükleneceği ve konfigure edileceği (ya da kurulacağı) liste sırasına göre işleme devam eder.

#### 5.2 Paket Kaynak Listesi: apt-setup

apt-get sizin için paketleri otomatik olarak getirdiğinden, henüz kurmadığı paketleri nereden bulması gerektiğini bilmelidir. Bu bilgi /etc/apt/sources.list içerisinde bulunmaktadır.

```
# cat /etc/apt/sources.list
# See sources.list(5) for more information, especially
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.

# Security updates for "stable"
deb http://security.debian.org stable/updates main contrib non-free
deb http://security.debian.org testing/updates main contrib non-free

# Stable
deb http://ftp.de.debian.org/pub/debian stable main contrib non-free
deb http://ftp.de.debian.org/pub/debian-non-US stable/non-US main contrib non-free
....
```

<listenin devamını yazmadık>

Bu listeyi eliniz ile değiştirebilirsiniz bile (bkz: source.list kılavuz sayfası) bu iş için daha interaktif bir araç kullanabilirsiniz.

```
# apt-setup
```

Bu araç sizin debian paketlerini bulabileceğiniz CDROMlar, web siteleri ve ftp siteleri yerler konusunda işlem yapmanıza olanak sağlar. İşiniz bittiğinde yaptığınız değişiklikleri /etc/apt/source.list dosyasına kaydeder ve sorduğunuz zaman apt-get yine bu listedeki kaynaklarda paketleri aramaktadır.

### 5.3 apt-get'ten dselect'e

apt-get aracı apt-get kılavuz sayfasında da bulabileceğiniz bir çok komut satırı seçeneğine sahiptir. Öntanımlı olan seçenekler genel olarak gayet iyi çalışmaktadır ancak eğer aynı seçeneği her zaman kullandığınızı düşünüyorsanız /etc/apt/apt.conf dosyasına bir ayar eklemek isteyebilirsiniz. Bu dosyanın yazım kuralları apt.conf kılavuz sayfasında ayrıntılı olarak anlatılmaktadır.

```
# man apt.conf
```

Dediğimiz gibi apt-get bizim kullandığımız install komutunun yanında bir çok komuta sahiptir. Bunlardan bir tanesi de Debian sisteminizde yüklü her debian paketi için statü ayarına uyan apt-get dselect-upgr komutudur.

### 5.4 dselect'e Başlarken

dselect denilen araç bir Debian sisteminde paketleri yönetmek için kullanılan bir arayüzdür. Dselect menüsü kullanarak sistem yöneticisi aşağıdaki işlemleri gerçekleştirebilir:

- Mevcut paket versiyonlarının listesini güncellemek
- Mevcut ve kurulmuş paketlerin durumlarını görüntülemek
- Paket seçimlerini değiştirmek ve bağımlılıkları yönetmek
- Yeni paketler yüklemek ve paketleri yeni versiyonları ile güncellemek

Her paket için durum (statü) /var/lib/dbkg/status dosyasında yer alsa da bu dosyayı güncellemek için en iyi yol yine daha interaktif bir araç olan dselect'i kullanmaktır.

```
# dselect
```

Debian 'dselect' package handling frontend.

```
0. [A]ccess      Choose the access method to use.
1. [U]pdate      Update list of available packages, if possible.
2. [S]elect      Request which packages you want on your system.
3. [I]ninstall   Install and upgrade wanted packages.
4. [C]onfig      Configure any packages that are unconfigured.
* 5. [R]emove     Remove unwanted software.
6. [Q]uit        Quit dselect.
```

Debian 'dselect' package handling frontend.

- 0. [A]ccess      Choose the access method to use.
- 1. [U]pdate      Update list of available packages, if possible.
- 2. [S]elect      Request which packages you want on your system.
- 3. [I]ninstall    Install and upgrade wanted packages.
- 4. [C]onfig      Configure any packages that are unconfigured.
- \* 5. [R]emove      Remove unwanted software.
- 6. [Q]uit        Quit dselect.

Move around with ^P and ^N, cursor keys, initial letters, or digits;  
Press <enter> to confirm selection. ^L redraws screen.

Version 1.10.9 (i386).

Copyright (C) 1994-1996 Ian Jackson.

Copyright (C) 2000,2001 Wichert Akkerman.

This is free software; see the GNU General Public Licence version 2  
or later for copying conditions. There is NO warranty. See  
dselect --licence for details.

Move around with ^P and ^N, cursor keys, initial letters, or digits;  
Press <enter> to confirm selection. ^L redraws screen.

Version 1.10.9 (i386).

Copyright (C) 1994-1996 Ian Jackson.

Copyright (C) 2000,2001 Wichert Akkerman.

This is free software; see the GNU General Public Licence version 2  
or later for copying conditions. There is NO warranty. See  
dselect --licence for details.

## 5.5 dselect Select Modunu Kullanmak

Her paketin durumunu görüntülemek ve değiştirmek için Select seçeneğini seçebilirsiniz. Bundan sonra ekran dolusu yardım görüntülenecektir. Bu yardımı okuduktan sonra <enter> tuşuna basınız. Daha sonra aşağıdakine benzer şekilde bir paket listesi göreceksiniz:

```

EIOM Pri Section Package      Inst.ver   Avail.ver  Description
- All packages -
--- Obsolete and local packages present on system ---
----- Obsolete/local Required packages -----
----- Obsolete/local Required packages in section base -----
*** Req base   adduser      3.50      <none>
*** Req base   base-files    3.0.8     <none>
*** Req base   base-passwd   3.5.3     <none>
*** Req base   bash          2.05b-7   <none>
*** Req base   bsduutils     2.11z-1   <none>
*** Req base   console-tool  0.2.3dbs-32 <none>

```

All packages

The line you have highlighted represents many packages; if you ask to  
install, remove, hold, &c it you will affect all the packages which  
match the criterion shown.

If you move the highlight to a line for a particular package you will  
see information about that package displayed here. You can use 'o' and  
'O' to change the sort order and give yourself the opportunity to mark  
packages in different kinds of groups.

## 5.6 Paket Durumu

Her paketin durumu kapalı bir başlık olan EIOM altında yer almaktadır. Bizi ilgilendiren kolon her paketin aşağıdakilerden bir tanesi ile işaretlendiği M karakteri altındadır:

İşareti değiştirmek için istediğiniz koda göre bir tuşa basınız (eşit, dash ya da altçizgi). Eğer işareti \* (asteriks) karakterine değiştirmek istiyorsanız + tuşuna basmalısınız. Bunu yaptıktan sonra büyük harf Q kullanarak yaptığımız değişiklikleri kaydedip Select ekranından çıkabilirsiniz. Eğer dselect içerisinde

yardım almak isterseniz herhangi bir zamanda ? tuşuna basarak yardım alabilirsiniz. Sonra da tekrar space tuşuna basarak yardım ekranından çıkabilirsiniz.

## 5.7 Kurulum ve Konfigürasyon

Debian paketlerin durum ayarlarına bağlı olarak siz apt-get dselect-upgrade yazana kadar paketleri yüklemeyi ya da kaldırmaz. Bu komut aslında sizin için bir seferde birkaç tane adımı gerçekleştirir: Kurulum, Kaldırma ve Konfigürasyon. Kurulum ve Kaldırma adımları sırasında soru sormak için program durmaz. Fakat konfigürasyon adımı paketlerin sizin istediğiniz yolla konfigüre edilmesini sağlamak amacıyla bir takım sorular sorulmaktadır. Bu adımları çalıştırmak için başka yollar da vardır. Örneğin her adımı ayrı olarak dselect menüsünden seçebilirsiniz. Bazı paketler kendi konfigürasyon adımları için debconf denilen bir sistem kullanırlar. Böylece kendi kurulumları ile ilgili soruları text terminali, grafik arayüz veya web sayfası gibi çeşitli ortamlarda sorabilirler.

## 5.8 dpkg İle Kurulmuş bir Paketin Durumunu Ele Almak

Uzun paket listeleri için şu ana kadar görmüş olduğumuz Debian paket yönetim araçları birden fazla adımdan oluşan operasyonlar için kullanılabilecek en iyi araçlardır. Ancak yine de bu araçların paket yönetimi konusunda eksik olduğu yönler vardır. Bu durumlar için dpkg kullanmak isteyeceksiniz. Örneğin bir paketin tüm durum ve açıklamalarını görmek için -s seçeneğini kullanınız:

```
#dpkg -s ickle
Package: ickle
Status: deinstall ok config-files
Priority: optional
Section: net
Installed-Size: 1364
Maintainer: Leo Costela <costelaaa@ig.com.br>
Version: 0.3.2-1
Config-Version: 0.3.2-1
Depends: ickle-common (= 0.3.2-1), libc6 (>= 2.2.4-4),
        libglib1.2 (>= 1.2.0), libgtk1.2 (>= 1.2.10-4),
        libgtkmm1.2, libicq2000 (= 0.3.2-1), libsigc++0,
        libstdc++2.10-glibc2.2 (>= 1:2.95.4-0.010810),
        xlibs (>> 4.1.0)
Suggests: ickle-control (= 0.3.2-1)
Description: An ICQ2000 client for GTK+
 Ickle is an ICQ client which is able to communicate with ICQ servers
 using the new ICQ2000 protocol. This offers much better reliability
 than the older ICQ99 protocol. ickle has a constantly improving
 interface, now supporting history browsing, and much more.
.
Ickle currently supports:
 * Sending/Receiving messages, URLs and SMS messages
 * Retrieving away messages
 * Fetching user info from server
 * Fetching/setting away messages
 * Custom away messages
 * Authorisation Requests
 * "User Added You" messages
.
This version is compiled without gnome support, for gnome support
install the package 'ickle-gnome'.
```

## 5.9 Bir dosya ile .deb Arasındaki Bağlantı

.deb paketi dosyaları içerdiğinden dolayı, paketin içerisinde yer alan dosyaları listelemenin bir yolu olduğunu düşünebilirsiniz. Evet böyle bir şey yapmanız mümkündür ve bunun için -L seçeneğini kullanmanız gerekmektedir:

```
# dpkg -L frozen-bubble
/.
```

```
/usr
/usr/lib
/usr/lib/menu
/usr/lib/menu/frozen-bubble
/usr/lib/perl5
/usr/lib/perl5/auto
/usr/lib/perl5/auto/fb_c_stuff
/usr/lib/perl5/auto/fb_c_stuff/fb_c_stuff.bs
/usr/lib/perl5/auto/fb_c_stuff/fb_c_stuff.so
/usr/games
/usr/games/frozen-bubble
/usr/games/frozen-bubble-editor
/usr/share
/usr/share/doc
/usr/share/doc/frozen-bubble
/usr/share/doc/frozen-bubble/README
/usr/share/doc/frozen-bubble/AUTHORS
/usr/share/doc/frozen-bubble/copyright
/usr/share/doc/frozen-bubble/changelog.gz
/usr/share/doc/frozen-bubble/changelog.Debian.gz
/usr/share/man
/usr/share/man/man6
/usr/share/man/man6/frozen-bubble.6.gz
/usr/share/man/man6/frozen-bubble-editor.6.gz
/usr/share/gnome
/usr/share/gnome/apps
/usr/share/gnome/apps/Games
/usr/share/gnome/apps/Games/frozen-bubble.desktop
/usr/share/perl5
/usr/share/perl5/fbsyms.pm
/usr/share/perl5/FBLE.pm
/usr/share/perl5/fb_c_stuff.pm
/usr/share/perl5/fb_stuff.pm
/usr/share/pixmaps
/usr/share/pixmaps/frozen-bubble-icon-48x48.png
/usr/share/pixmaps/frozen-bubble-icon-16x16.png
/usr/share/pixmaps/frozen-bubble-icon-32x32.png
/usr/share/pixmaps/frozen-bubble.xpm
```

Bunun tersine belirli bir dosyanın hangi pakete ait olduğunu anlamak için de `-S` seçeneğini kullanabilirsiniz.

```
# dpkg -S /usr/share/pixmaps/frozen-bubble.xpm
frozen-bubble: /usr/share/pixmaps/frozen-bubble.xpm
```

Gördüğünüz gibi kolonun solunda paketin ismi belirtilmektedir.

## 5.10 Yüklenecek Paketleri Bulmak

Genellikle `apt-get` ihtiyacınız olan bir Debian paketini zaten biliyor olacaktır. Eğer bu olmazsa bu debian paketleri listesi içerisinde ya da web den bu paketleri bulabilecek durumda olmalısınız. Eğer `.deb` dosyasını bulur ve indirirseniz `-i` seçeneğiyle kurabilirsiniz:

```
# dpkg -i /tmp/dl/xsnow_1.40-6_i386.deb
```

eğer `.deb` olarak aradığınız dosyayı bulamayıp yerine `.rpm` ya da başka bir paket formatında bulduysanız `alien` programını kullanabilirsiniz. `alien` programı paketleri çeşitli formatlardan `.deb` formatına çevirmektedir.

## 5.11 Dpkg İle Bir Paketi Konfigure Etmek

Bu paketlerden bir tanesini konfigure etmek için `dpkg --configure` komutu kullanılmalıdır. Bu kullanım yüklenmiş bir programın dosyaları ya da programa ait ayar dosyaları zarar gördüğü zaman işe yarayabilir.



## 5.12 Ek Debian Paket Yönetim Araçları

Burada anlattıklarımıza ilave olarak, daha birçok Debian paket yönetimi araçları bulabilirsiniz. Bunun için Debian anasayfasına, Debian kurulum rehberine, Debian paket listelerine ve Alien anasayfasına bakabilirsiniz.

## 6 Huzurlarınızda çekirdek... Linux!

Genel olarak Linux ismi, Linux dağıtımı adı altında geçen, programların birarada çalışmasından oluşan dağıtımları tanımlamak için kullanılır. Halbuki teknik olarak Linux sadece çekirdek' (kernel'in ismidir. Herne kadar Linux diye adlandırıldığımız diğer parçalar (kabuk, derleyiciler vs.) sistemin çalışması ve tam bir işletim sistemi oluşturması için aynı derecede gerekli olsalar da teknik olarak çekirdekten ayrılırlar. Yine de insanlar "Linux" kelimesini "Linux tabanlı dağıtım" olarak kullanmaya devam edeceklerdir. Yine de herkesin hemfikir olacağı nokta Linux çekirdeğinin her Linux İşletim sisteminin kalbini oluşturduğudur.

### 6.1 Donanım ile arabirim oluşturmak

Linux çekirdeğinin temel görevi sisteminizde donanıma erişim için direkt bir arabirim sağlamaktır. Çekirdek ham donanım ve uygulama programları arasında bir soyutlama katmanı sağlar. Böylelikle programlar ana kartınızın veya disk kontrol ünitesinin detaylarını bilmek durumunda kalmadan çok daha üst seviyede diskten okuyup ona bilgi yazabilirler mesela.

### 6.2 CPU Soyutlama

Linux çekirdeği aynı zamanda sisteminizdeki işlemciye de bir soyutlama katmanı sağlar - bu sayede birçok program eş zamanlı çalışıyormuş izlenimi verir. Herbir programın gerektiği kadar işlemci zamanı ve diğer kaynaklardan yararlanmasını çekirdek ayarlar.

Şu anda bir Linux sistem üzerinde çalışıyorsanız, kullandığımız çekirdek ya UP(uniprocessor (tek işlemcili)) ya da SMP(symmetric multiprocessor(simetrik çok işlemcili)) bir çekirdektir. Eğer bir SMP ana karta sahipseniz ve UP çekirdek kullanıyorsanız, Linux'unuz ekstra işlemcilerden haberdar olmayacaktır. Bunu düzeltmek için kendi donanımınıza uygun bir SMP çekirdek derlemeniz gerekir. Halihazırda SMP çekirdekler, küçük bir performans kaybıyla da olsa UP sistemlerde de çalışacaklardır.

### 6.3 IO(giriş/çıkış)'ları Soyutlama

Çekirdek aynı zamanda her türlü dosya girdi çıktılarını soyutlama işini de üstlenir. Her bir programınızın bu işleri kendi başına yaptığını bir düşünün. Disk kontrolörlerinizi değiştirdiğiniz takdirde bütün programlarınız çakılıp kalacaklardı. Bu yüzden Linux çekirdeği de programların kullanabileceği veri depolama ve erişimi için soyutlama için Unix modellerini kullanmaktadır. Bu şekilde kullandığınız veritabanı, veriyi IDE ya da SCSI RAID dizisinde ya da ağ üzerinden erişilen bir dosya sisteminde tutup tutmadığınızla ilgilenmeyecek, sadece o veri ile işini yapacaktır.

### 6.4 Ağ Merkezi

Linux'un en ünlü ve iddialı olduğu alanlardan biri oldukça güçlü ağ yapısı, özellikle de TCP/IP desteğidir. Eğer şimdiye kadar TCP/IP desteğinin Linux çekirdeğinde yer aldığını tahmin ettiyseniz, haklısınız. Çekirdek, standartlarla uyumlu, yüksek seviyeli bir arabirim ile ağ üzerinden veri gönderilmesini ve alınmasını destekler. Arka planda ise Linux çekirdeği sizin ethernet kartınız ya da pppd daemon ile düşük seviyeli bağlantıları halleder ve Internet iletişimini sağlar. Bu konunun detayları bir sonraki dersin konusudur.

### 6.5 Ağın Faydaları

Linux'un en iyi özelliklerinden biri çekirdekte yeralan birçok yararlı özelliktir. Mesela bir Linux çekirdeğini, bütün evinizdeki ağın modem üstünden Internet'e çıkış kapısı olması için konfigüre edebilirsiniz -buna IP Masquerading veya IP NAT denir. Bunun dışında Linux çekirdeği ağdaki diğer Unix makinalarla dosya paylaşabilecek şekilde de konfigüre edilebilir. Çekirdekte buna benzer bir çok faydalı özellikli vardır, bunları çekirdek özelliklerini incelerken daha detaylı olarak da göreceksiniz.

## 6.6 Açılış işlemleri

Sanırsanız, şu an Linux'un açılışta hangi aşamalardan geçtiğini gözden geçirmek için tam da zamanı. Linux tabanlı sisteminizi açtığınızda çekirdek imajı (tek bir ikilik dosyadır) diskten hafızaya Lilo ya da Grub gibi bir yükleyici tarafından yüklenir ve ardından çekirdek sistemin kontrolünü alır. Yaptığı ilk şeylerden biri mevcut donanımları gözden geçirmek ve desteklemek üzere konfigüre edildiği donanımları tanımdır. Donanımlar doğru olarak tanınıp işler hale getirildiklerinde, çekirdek normal kullanıcı alanında çalışan programları (process(işlemler) diye de geçer bunlar) çalıştırır. İlk çalıştırılan program /sbin/init'dir. Init ise /etc/inittab altında tanımlanan diğer programları çalıştırır. Saniyeler içinde sisteminiz ayağa kalkmış ve çalışır halde, kullanımınıza hazırdır. Herne kadar hemen hiçbir zaman çekirdeğe doğrudan erişim sağlamasanız da, Linux çekirdeği bütün normal programların "üstünde" aralıksız çalışmakta, bu programların ve kütüphanelerin ihtiyaç duydukları sanallaştırma ve soyutlama işlemlerini çeşitli katmanlarda gerçekleştirmektedir.

## 6.7 Ve huzurlarınızda... modüller

Yakın tarihli bütün Linux çekirdekleri, çekirdek modüllerini destekler. Çekirdek modülleri oldukça hoş şeylerdir -basitçe çekirdeğin bir parçası olan, göreceli olarak çok daha küçük dosyalardan oluşurlar. Çekirdek belli bir fonksiyonu gerçekleştirmek için bir modüle ihtiyaç duyduğunda o modülü diskten otomatik olarak yükler ve kendisine entegre eder, böylelikle dinamik olarak yetkinliğini arttırabilir. Eğer bir çekirdek modülünün özellikleri bir süre kullanılmazsa, çekirdek onu hafızadan çıkartabilir ve böylelikle autocleaning denen bir işlemi gerçekleştirerek gereksiz kaynak kullanımını azaltır. Çekirdek modülleri olmasa, çalışmakta olan çekirdeğimize (ki kendisi tek bir dosyadan oluşur) bütün gereken özellikleri yüklememiz gerekecekti. Normalde ise temel ihtiyaçları sağlayacak bir çekirdek derleyip, ilerde ihtiyaç duyabileceğimiz modülleri ekleriz. Eğer gerekirse uygun modül çekirdeğe daha sonra dinamik olarak eklenebilir. Bu aynı zamanda RAM ve diğer kaynakların da korunması anlamına gelir, çünkü bir modül ancak gerek duyulduğunda yüklenir ve atıl hale geldiğinde hafızadan çıkartılır.

## 6.8 Modüller nerede durur ?

Çekirdek modülleri genelde /lib/modules/x.y.z (x.y.z modüllerin uyumlu olduğu çekirdek sürümünü belirtir) altında dururlar; her modülün isminin sonunda ".o" uzantısı vardır, bu sayede bu dosyalar makina dili komutları içeren dosyalar olarak tanımlanırlar. Tahmin edebileceğiniz gibi her modül, belli bir çekirdek fonksiyonunu tanımlar. Bir modül FAT dosyasistemi desteği sağlayabilirken bir başkası ISA ethernet desteği sağlayabilir.

## 6.9 Modüller – her işlem için kullanılamaz

Herşeyi bir modül haline getiremeyeceğinizi belirtmekte fayda var. Modüller disk üzerinde durdukları için sizin başlatılabilir linux çekirdeğinizin kendi içinde mevcut disk kontrolörünüzün desteğine sahip olması gerekir. Eğer bu destek yoksa (veya bu desteği modüller olarak derlediyseniz) çekirdeğiniz bu modülleri yüklemeyi beceremeyecektir (çünkü modüllerin yüklü olduğu diske erişim yeteneğinden yoksundur) ve bu yüzden de sisteminiz başlatılamayacaktır.

# 7 Çekirdek kaynaklarını bulup bilgisayarınıza indirmek

Bu döküman yazıldığı sıralarda en yeni çekirdek 2.4.18'di. 2.4.18 çekirdek 2.4 tutarlı çekirdek serisinin bir parçasıdır. Bu seri gerçek dünyada üretim amaçlı kullanılacak sistemler için hazırlanmaktadır. Bu seri ile eş zamanlı olarak geliştirilen 2.5 serisi de kullanıma açıktır, fakat bunların üretim sistemlerinde kullanılması tavsiye edilmemektedir. 2.5'deki 5 sayısı tek bir sayıdır, ve çekirdek terminolojisinde bu serinin çekirdek geliştiricilerine ve deneme kullanıcılarına yönelik deneysel bir sürüm olduğunu belirtir. 2.5 sürüm çekirdekler, yeteri kadar güvenilir hale gelip, üretim sistemlerinde kullanılabilir hale ulaştıklarında 2.6 sürüm numarasını alacaklardır (çift sayıya dikkat).

## 7.1 Hangi çekirdek kaynaklarını kullanmalı ?

Eğer mevcut çekirdeğinizin yeni bir sürümünü derlemek istiyorsanız (mesela SMP desteği eklemek isteyebilirsiniz) en iyi yöntem dağıtımınızın çekirdek kaynak kodu paketini kurmaktır. Bunu kurduktan sonra /usr/src/linux dizini altında bazı dosyaların hazır olduğunu göreceksiniz. Bunun dışında tamamen yeni

bir çekirdek kurmayı da isteyebilirsiniz. Genelde en doğru yaklaşım dağıtımınızın yeni ya da güncellenmiş kaynak kodu paketinden kurulum yapmaktır. Bu paket sizin Linux sisteminizde optimal olarak çalışmak üzere yamanmış ve konfigüre edilmiş çekirdek kaynak kodlarını içerecektir.

## 7.2 Çekirdeği kaynağından edinmek

Eğer macera meraklısıysanız, çekirdek kaynak kodunu <http://www.kernel.org/pbu/linux/kernel> adresinden "genel bir" kaynak kodu indirerek edinebilirsiniz. Bu dizinde Linus ya da Marcello tarafından yayınlanmış resmi kaynak kodlarını bulabilirsiniz. Burada önemli nokta dağıtımınızda bulunan bütün özellikleri bu çekirdeğin desteklemeyebileceği. Bu sebeple tam olarak ne yaptığınızdan emin olmadan böyle bir "genel" çekirdek derlememenizde fayda var.

Kernel.org'da değişik dizinler altında düzenlenmiş çekirdek kaynak kodlarını bulacaksınız. Kernel (v2.2, v2.4 vs.) Her dizinin içinde "linux-x.y.z.tar.gz" ve "linux-x.y.z.tar.bz2" isimli dosyalar bulunacaktır. Bunlar temel kaynak kodu dosyalarıdır. Bunların dışında "patch-x.y.z.gz" ve "patch-x.y.z.bz2" dosyaları da vardır. Bunlar ise önceki sürümleri güncellemeye yarayan yama dosyalarıdır. Eğer yeni bir çekirdek derleyecekseniz "linux" diye başlayan dosyalardan birini almanız gerekir.

## 7.3 Çekirdek kodu paketini açmak

kernel.org'dan yeni bir çekirdek kodu paketi aldıysanız önce bunu açmak gerekecektir. Bunun için önce /usr/src dizinine gidi. Eğer burada bir "linux" dizini varsa onu "linux.old" dizini haline getirin (mv linux linux.old tabi bunu root olarak yapıyorsunuz) Şimdi yeni çekirdeği açmanın zamanı. /usr/src altındayken tar zxvf /kernelin/bulundugu/yer/linux-x.y.z.tar.gz ya da cat /kernelin/bulundugu/yer/linux-x.y.z.tar.bz2 — bzip2 -d — tar xvf -. Buradaki fark dosyaların bzip2 ya da gzip ile sıkıştırılmasından kaynaklanmaktadır. Yeni linux dağıtımlarında tar jxvf /kernelin/bulundugu/yer/linux-x.y.z.tar.bz2 -C /usr/src/linux/ komutu da tar.bz2 dosyasının açılmasını sağlayacaktır.

Bu komutu verdikten sonra çekirdek kaynak kodları sözkonusu dizine açılacaktır. Yalnız unutmanız gereken nokta bu dosyaların açık halinin muhtemelen 50MB'dan fazla tutacağıdır. Buna göre yer açmanızda fayda var harddiskinizde.

## 8 Çekirdeği Düzenlemek

Çekirdeği derlemeden önce bazı düzenlemeler yapmamız gerekecek. Bu düzenlemeler, yeni çekirdeğimizde hangi özelliklerin olacağını ve hangilerinin olmayacağını belirleyecek. Bu sırada hangi özelliklerin ana çekirdek dosyasında yer alacağı (ve boot işlemi sırasında yükleneceği) hangilerinin ise daha sonradan eklenebilir modüller olarak derleneceğine de karar vereceğiz.

Eski zamanlarda çekirdeği düzenlemek ve derlemek ciddi bir problemdi. Zira config betiğini çaşırtmayı gerektiriyordu. Herne kadar hala bu imkan mümkün olsa da biz pek tavsiye etmiyoruz, tabi yüzlerce soruya teker teker cevap vermek isterseniz siz bilirsiniz.

### 8.1 Yeni yöntem

/usr/src/linux dizinine girdikten sonra "make config" yerine "make menuconfig" ya da "make xconfig" komutunu çalıştırabilirsiniz. make menuconfig güzel bir konsol tabanlı ve renkli menu sistemi sunmakta, make xconfig ise benzer bir sistemi x tabanlı bir arabirim ile kullanmanıza izin vermekte.

make menuconfig kullandığınızda dikkat etmeniz gereken noktalardan biri, solunda < > bulunan seçeneklerin modül olarak derlenebileceği.

Üzerinde durduğunuz bir seçeneği space tuşuna basarak çekirdeğe dahil edebilir "<\*>", çekirdekten çıkartabilir "< >" ya da modül olarak derlemek üzere seçebilirsiniz "<M>". Bunun dışında y tuşu ile ekleme, n tuşu ile çıkartma ve m tuşu ile modül yapmayı direkt seçmeniz de mümkün. Birçok çekirdek seçeneğinin detaylı açıklamalara sahip olması da bir başka şanstır, bu bilgilere erişmek için de h tuşuna basmanız gerekir.

### 8.2 Konfigürasyon ipuçları

Malesef hepsini burada açıklayamayacağımız kadar çok konfigürasyon seçenekleri mevcuttur. Bunun yerine size önemli kategorilerin açıklamalarını yapmayı uygun bulduk.

Code maturity level options (Kod olgunluğu seviye seçenekleri)

Şimdi make menuconfig ya da make xconfig seçeneği ile çekirdek düzenleme ortamına girelim.

Kod olgunluğu seviye seçenekleri: Bu kategoride tek bir seçenek vardır. "Prompt for development and/or incomplete code/drivers" (geliştirme ve/veya tamamlanmamış seçenekleri seçebilmek ister misiniz ?) Eğer bu seçeneği seçerseniz, deneysel kabul edilen bir çok özellik (ReiserFS?, devfs vs.) kendileri ile ilgili kategoriler altında erişilebilir olacaklardır. Eğer bu seçenek seçilmediyse göreceğiniz bütün özellikler "kararlı" yapı altında kullanılmakta olan özellikler olacaktır. Bunu seçmeniz, seçim şanslarınızı arttıracığından iyi olur.

Modüller ve İşlemci ile alakalı seçenekler: Loadable module support (Yüklenbilir modül desteği)

Bu konfigürasyon kategorisinin altında çekirdeğin modül desteği ile alakalı üç seçenek vardır. Genelde üçünün de seçili olması gerekir.

Processor type and features (işlemci cinsi ve özellikleri): Bu bölüm çeşitli işlemci konfigürasyon seçeneklerini içermektedir. Burada "Symmetric multiprocessor support" özellikle önemlidir ve birden fazla işlemciye sahip makinalar için elzemdir. Aksi takdirde sisteminizdeki ilk CPU algılanacaktır. "MTTR Support" seçeneği genellikle seçilir , çünkü modern sistemlerde X konusunda performans artışı sağlamaktadır.

General and parallel port options: (Genel ve paralel port seçenekleri)

General setup (Genel düzen): Bu bölümde Networking ve PCI Support seçenekleri genel olarak seçili olmalıdır. "Kernel support for ELF binaries" (Elf çalışabilir programları için çekirdek desteği) de. a.out ve MISC binary seçenekleri de tavsiye edileir fakat bunların modül olarak derlenmesi daha iyi olacaktır. "System V IPC" ve "Sysctl support" seçeneklerini de seçmeye özen gösterin. Bu seçenekler hakkında daha fazla bilgi edinmek isterseniz help seçeneği ile detaylı bilgiler alabileceğiniz daha önce belirtmiştik.

"Paralel port" seçeneği özellikle yazıcı kullananlar için gerekecektir. Tam bir yazıcı desteği için "Character Devices" (karakter aygıtları) bölümünden "Paralel Printer Support" (Paralel yazıcı desteği) kısmını da seçmek gerekir.

RAID ve LVM Multi-device support (RAID and LVM)(Çoklu aygıt seçimi): Bu bölümde Linux'un yazılımsal RAID ve mantıksal bölüm yöneticisi özellikleri seçilir. Yazılımsal RAID, disklerinizin birbirleri ile yedekli olarak çalışmasını ve bu sayede performans ve güven artışı sağlamayı amaçlar. Yazılımsal RAID konusunda detaylı bilgilere sondaki kaynaklar bölümünden ulaşabilirsiniz.

Networking and related devices(Ağ ve ilgili aygıtlar) Networking options (Ağ seçenekleri): Bu bölümde her türlü ağ seçenekleri mevcuttur. Linux sisteminizi tipik bir ağa bağlayacaksanız, "Packet socket" "Unix domain sockets" ve "TCP/IP Networking" seçeneklerini seçmeniz gerekmektedir.

Bu temel seçenekler dışında bir çok seçenek daha, mesela "Network packet filtering" (Ağ paket filtreleme, ip tables kullanmak, firewall kurmak için) bulunmaktadır.

Çekirdeğinizin desteklemesini istediğini ethernet kartlarını da belirtmeniz gerekmektedir. Aradığınız kart muhtemelen "Ethernet (10 or 100MBit) alt kategorisinin altında bulunmaktadır.

IDE support(IDE desteği) ATA/IDE/MFM/RLL support: Bu bölümde IDE sürücüler, CD ve DVD Rom'lar ve diğer benzer çevrebirimler kullananlara özgü seçenekler yer almaktadır. Eğer sisteminizde IDE diskleri varsa "Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support" seçeneğini seçin.

Bunun dışında "Include IDE/ATA-2 DISK support"(IDE/ATA2 disk desteğini kat) ve mevcut ana kartınıza uygun chipset seçeneklerini seçin ve bunu çekirdeğe ekleyin, modül olarak derlemeyin ki sisteminiz boot edebilsin). Eğer IDE CD Rom'a sahipseniz, "Include IDE/ATAPI CD-ROM support" seçeneğini de eklemeyi unutmayın. Anakartınızın chipset desteğini seçmeden de IDE disklere muhtemelen ulaşabileceksiniz fakat bu durumda ana kartın performans avantajlarından yararlanmamış olduğunuzdan daha yavaş bir erişimle karşı karşıya kalabilirsiniz.

"Enable PCI DMA by default if available" (eğer varsa PCI DMA desteğini aç) seçeneği de heme hemen bütün sistemler için tavsiye edilen bir özelliktir. DMA olmadan IDE çevrebirimleri PIO modunda ve duruma göre potansiyellerinin on onbeş katı daha yavaş çalışabilirler.

Belli bir disk için DMA'nın etkin hale geldiğini hdparm -d 1 /dev/hdx ile root olarak kontrol edebilirsiniz. Burada /dev/hdx DMA'ı etkin hale getirmek istediğiniz diskin adıdır.

SCSI support(SCSI desteği) Bu kategori SCSI diskler ve çevrebirimlerle ilgili bütün seçenekleri barındırır. Eğer SCSI tabanlı bir sisteminiz varsa "SCSI support", "SCSI disk support", "SCSI CD-ROM support" ve "SCSI tape support" seçeneklerinizi ihtiyaçlarınıza göre seçebilirsiniz.

IDE disklerde olduğu gibi eğer sistemin yüklü olduğu disk SCSI bir disk ise bu destekleri yüklenbilir modüller olarak değil ana çekirdek dosyasının içine koymanızda fayda var.

Bunlardan başka SCSI'nin gerektiği gibi çalışabilmesi için "SCSI low-level drivers" bölümüne gidip, SCSI kartınızın seçili olduğundan emin olun, yine bu seçenek de çekirdeğin içinde direkt yer almalıdır, modül olarak değil.

Miscellaneous character devices (Çeşitli karakter aygıtları): Character devices: Bu bölüm çeşitli çekirdek aygıt sürücülerini içerir. "Virtual terminal" ve "Support for console on virtual terminal" seçeneklerinin

seçili olduğundan emin olun. Bunlar sistem açıldıktan sonra standart text tabanlı terminallerin sizi karşılaması için gereklidirler.

Büyük ihtimalle "Unix98? PTY support" u da seçmenizde fayda var. Bunlar dışındaki her şeyi seçimi size kalmıştır. Eğer paralel yazıcı kullanmak istiyorsanız "Paralel printer support" seçeneğini seçmeniz gerektiğini tekrar hatırlatalım.

"Enhanced real-time clock support" tavsiye edilir, "/dev/agpgart (AGP desteği)" ve "Direct Rendering Manager" da X altında performans sağlayan Linux 3D hızlandırıcılı kartlar için gereklidir. X'i bu hızlı modda çalıştırmak için çekirdek derleme dışında da bazı ayarlar yapmak gerekecektir.

File systems and console drivers(Dosya sistemleri ve konsol sürücüler): File systems: Bu kategorideki seçenekler tahmin edeceğiniz gibi dosya sistemlerini içermektedir. Root (/) dosya sisteminiz için gerekli olan dosya sistemi desteğini (ext3, reiserfs, jfs, xfs vs.) çekirdeğe dahil etmeniz gerekmektedir. Bundan başka "/proc file system support" seçeneği de bir çok dağıtım için elzemdir.

Genel olarak "/dev/pts file system support for Unix98? PTYs" seçeneğini seçmeniz tavsiye edilir, tabi "/dev file system support" seçeneğini seçmiyorsanız. O takdirde "/dev/pts" seçeneğini seçilmemiş bırakmanız gerekir.

Console drivers: Tipik olarak herkez "VGA text console" (x86 sistemler için elzemdir) seçecektir. İsterseniz "Video mode selection support" de seçebilirsiniz. "Frame buffer support" seçtiğiniz takdirde ise text tabanlı ekran yerine grafik ekranda yazıları görebilirsiniz. Fakat bu seçeneklerini bazıları X ile karışıklıklara sebep olmaktadır bu yüzden düz VGA konsola bağlı kalmakta, en azından başlangıçta fayda vardır.

## 9 Çekirdeği derlemek ve kurmak

Çekirdeği bir kez konfigüre ettikten sonra artık derlemeniz gerekir. Fakat bunu yapmak için önce bağımlılıkları öğrenmelisiniz. Bunu basitçe /usr/src/linux dizini içinde make dep komutunu çalıştırarak yapabilirsiniz.

### 9.1 make bzImage

Artık asıl çekirdek imajını derlemenin zamanı geldi. Yine aynı dizinde make bzImage yazıp entera basın. Makinanızın özelliklerine ve seçtiğiniz seçeneklere göre yarım saat kadar bir süre sonra (veya çok daha uzun) /usr/src/linux/arch/i386/boot (x86 PC çekirdeği için) altında bzImage dosyasını oluşturulmuş bulacaksınız.

Bu imajı nasıl sisteme kuracağınıza da değineceğiz ama şimdilik modüllere bir bakalım.

### 9.2 Modülleri derlemek

bzImage'ı oluşturduğumuza göre, modülleri derlemenin zamanı geldi demektir. Eğer çekirdek seçeneklerinden hiçbirinde modül seçeneğini işaretlememişseniz bile bu basamağı atlamayın, modülleri bzImage'ın hemen ardından derlemek en doğru harekettir. Zaten eğer modülleri etkin hale getirmediyseniz bu aşama oldukça hızlı geçecektir.

```
make modules && make modules_install
```

yazın ve entera basın.

Bu komutla birlikte modüller derlenecek ve /usr/lib/kernelsürümü dizinine kurulacaktır.

Tebrikler sıfırdan bir çekirdek derlediniz.

## 10 LILO'da başlangıç konfigürasyonu

Artık LILO'yu yeni çekirdeği yükleyecek şekilde konfigüre etmenin zamanı geldi. LILO en çok bilinen ve kullanılan linux başlangıç yükleyicisidir, bütün popüler Linux dağıtımlarında da bulunur.

İsterseniz öncelikle /etc/lilo.conf dosyasına bakarak başlayalım. Bu dosyada bir yerde image=/vmlinuz gibi bir satır olacaktır. Bu satır LILO'ya çekirdek için nereye bakması gerektiğini belirtir.

## 10.1 LILO'yu konfigüre etmek

LILO'yu yeni çekirdeği başlatacak şekilde konfigüre etmek için iki seçeneğiniz var. İlki mevcut çekirdek dosyasının üstüne yenisini yazmak –ki çok tavsiye edilmez, herşeyin yolunda gideceğinden bu kadar emin misiniz ? Daha güvenilir seçenek ise LILO'yu istenirse eski çekirdekle sistemi açabilecek şekilde konfigüre etmek. Şimdi bu yöntemi nasıl uygulayacağımızı göreceğiz.

## 10.2 LILO kodu

/etc/lilo.conf dosyanız aşağıdaki gibi olabilir

```
boot=/dev/hda
delay=20
vga=normal
root=/dev/hda1
read-only
image=vmlinuz
label=linux
```

lilo.conf'a yeni bir başlangıç seçeneği eklemek için şunları yapın. Önce /usr/src/linux/arch/i386/-boot/bzImage dosyasını /vmlinuz2 gibi bir isimle root'a (/) kopyalayın.

Ardından lilo.conf dosyanızın son üç satırını kopyalayıp tekrar yapıştırın. Nerdeyse bitti :)

Şimdi lilo.conf dosyanızın aşağıdaki gibi görünmesi lazım

```
boot=/dev/hda
delay=20
vga=normal
root=/dev/hda1
read-only
image=vmlinuz
label=linux
image=vmlinuz
label=linux
```

Önce ilk "image=" satırını "image=/vmlinuz2" şeklinde değiştirin ve ikinci "label=" satırını da "label=oldlinux" şeklinde değiştirmeyi unutmayın. Tabi eğer dosyanın yukarılarında bir yerlerde "delay=20" gibi bir satır yoksa böyle bir satır da eklemeniz gerekecek. Eğer böyle bir satır varsa bekleme süresini minimum 20 yapın.

Sonuç olarak nihai lilo.conf dosyanız aşağıdaki gibi görünecek;

```
boot=/dev/hda
delay=20
vga=normal
root=/dev/hda1
read-only
image=vmlinuz2
label=linux
image=vmlinuz
label=oldlinux
```

Bütün bu değişiklikleri yapıp text editörünüzden çıktıktan sonra lilo komutunu verip diskinizdeki konfigürasyonun güncellenmesini sağlayın. Eğer bunu yapmazsanız yaptığımız değişiklikler bir işe yaramayacaktır. Ayrıca bunu yapınca boot map dosyası da güncellenecektir.

## 10.3 LILO konfigürasyonunun nedenleri ve niçinleri

Gelelim yaptığımız değişikliklerin açıklanmasına. Bu lilo.conf dosyası iki ayrı çekirdekle açılış yapmak üzere düznelendi. Orjinal çekirdeğiniz /vmlinuz ile açılış yapabileceğiniz gibi yeni derlediğiniz /vmlinuz2 çekirdeği ile de açılış yapabilirsiniz. Otomatik olarak yeni çekirdek seçilecektir, çünkü onun image/label satırı dha yukarda tanımlanmıştır. Eğer eski çekirdeği boot etmek isterseniz açılış sırasında shift tuşuna basın. Bu size istediğini etiketi girip kernel seçme şansını verecektir. Burada oldlinux yazıp boot edebilir, ya da tab'a basıp seçeneklerinizi gözden geçirebilirsiniz.

## 11 PCI aygıtları

Bu bölüm PCI aygıtları ile çalışmanın ince detaylarından bahsedecektir. Linux altında PCI aygıt desteğini sağlamak basitçe "General Setup" bölümü altında "PCI Support" seçeneğini seçmekle olur. Bunun yanında "PCI device name database" seçeneğini seçmeniz de tavsiye edilir. Bu sayede PCI aygıtlarının aygıt numarası ve sayısal bilgileri yerine gerçek ingilizce isimlerini görebilirsiniz.

### 11.1 Mevcut PCI aygıtlarını incelemek

Bilgisayarınıza takılı olan PCI aygıtlarını incelemek için `cat /proc/pci` yazabilirsiniz. Çıkan bilginin çok işinize yarayacağı, hatta anlaşılabilir olacağı bile kesin değil:) Bunun yerine `lspci -v` yazarsanız çok daha anlaşılabilir bir çıktı ile karşılaşacaksınız. `lspci` komutu `pciutils` paketinin bir parçasıdır bu programlara <http://atrey.karlin.mff.cuni.cz/~mj/pciutils.html> adresinden ulaşabilirsiniz. Genellikle de hemen her dağıtımda bulunurlar.

`lspci -v` komutu ile bilgisayarınızda bulunduğunu bile bilmediğiniz donanımlarla karşılaşabilirsiniz. Genellikle bunlar bilgisayarınızın ana kartının üstünde bulunan pci tabanlı çevrebirimleridir. Bunlar genellikle BIOS vasıtası ile aktif ya da pasif hale getirilebilirler. `Pciutils` paketi aynı zamanda `setpci` komutu ile çeşitli PCI donanımlarının zamanlamasına müdahale edebilecek bir program sağlar. PCI donanım zamanlamaları ve etkileri konusunda IBM Developerworks'ün "Linux Hardware Stability Guide" makalesinin ikinci bölümünü okuyabilirsiniz.

### 11.2 PCI donanım kaynakları

Görevlerini yerine getirebilmek için sisteminizdeki PCI aygıtları değişik donanım kaynaklarınızdan faydalanırlar, kesmeler bunlara bir örnektir. Birçok PCI aygıtı donanımsal kesmeleri (interruptlar) işlenecek verileri olduğunu sisteme haber vermek için kullanırlar. Çeşitli donanım aygıtlarınız için hangi kesmelerin kullanıldığını görmek için `/proc/interrupts` dosyasına bakmanız yeterlidir. Bunun için `cat /proc/interrupts` komutunu vermeniz gerekir. Çıktısı şu şekilde olacaktır.

```
> cat /proc/interrupts
CPU0
 0:   1848603 XT-PIC timer
 1:    97489 XT-PIC keyboard
 2:         0 XT-PIC cascade
 5:    12148 XT-PIC eth0
 9:   1397985 XT-PIC nvidia
10:         1 XT-PIC soundblaster
11:    9921 XT-PIC bttv
12:   540231 XT-PIC PS/2 Mouse
14:   25550 XT-PIC ide0
15:         7 XT-PIC ide1
NMI:         0
LOC:   1848535
ERR:         0
MIS:         0
>
```

İlk kolon bir IRQ numarası verir, ikinci kolon ise kernel tarafından o kesmeden kaç tane üretildiğini gösterir. Son kolon ise sözkonusu donanımın kısa ismini içerir.

Donanımlarınızın kullanmakta olduğu IO portlarının listesini de şöyle görebilirsiniz

```
> cat /proc/ioports
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
```

```

0213-0213 : isapnp read
0220-022f : soundblaster
02f8-02ff : serial(set)
0300-0303 : MPU-401 UART
0376-0376 : ide1
03c0-03df : vga+
03e8-03ef : serial(set)
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0a79-0a79 : isapnp write
0cf8-0cff : PCI conf1
4000-403f : Intel Corp. 82371AB PIIX4 ACPI
5000-501f : Intel Corp. 82371AB PIIX4 ACPI
e000-e01f : Intel Corp. 82371AB PIIX4 USB
e400-e43f : 3Com Corporation 3c905 100BaseTX [Boomerang]
e400-e43f : 00:0b.0
f000-f00f : Intel Corp. 82371AB PIIX4 IDE
f000-f007 : ide0
f008-f00f : ide1

```

## 12 Linux USB

Çekirdeği konfigüre ederken büyük ihtiamlle "USB support" bölümü dikkatinizi çekmiştir. Burada Universal Serial Bus (Evrensel Seri Veriyolu) denilen USB'nin özellikleri bulunmaktadır. Nispeten yeni bir veriyolu olan USB özellikle çevrebirim alanında gittikçe yaygınlaşmaktadır, şimdiden USB klavye, fare, modem, printer, scanner hatta depolama cihazları çıkmıştır ve gün geçtikçe de daha yaygın kullanılmaktadırlar.

Linux üzerinde USB desteği de sadece bir kaç yıllık olduğundan, birçok Linux kullanıcısı şimdiye kadar pek USB cihaz kullanmamış ya da Linux altında USB desteğinin tam nasıl çalıştığını bilmiyor olabilir.

### 12.1 USB'yi etkin hale getirmek

Linux USB desteğini sağlamak için önce "USB Support" bölümüne gidip "Support for USB" seçeneğini seçin. Bundan sonra yapmanız gereken uygun USB sisteminizi seçmek. Seçenekleriniz "EHCI", "UHCI", "UHCI(alternate driver)" ve "OHCI". Linux kullananların problem yaşadığı temel nokta burasıdır.

UHCI, OHCI, EHCI eee, hani bunun kolay olması gerekiyordu ;)

EHCI ve arkadaşlarının ne olduğunu anlamak için öncelikle USB kullanacak her anakartın belli bir USB sistemi olduğunu anlamak gerekir. İşte bu sistemlerin isimleri EHCI, UHCI ve OHCI'dir.

EHCI sürücüsü, yeni model hızlı USB 2.0 sistemlerde yer alır.

OHCI sürücüsü ise, PC olmayan sistemleri de desteklemek amacıyla geliştirilmiştir, SIS ve ALI chipsetlerini de destekler.

UHCI sürücüsü ise basitçe geri kalan cihazlar içindir

Tedbirli davranmak için EHCI ve UHCI'den birini seçip (çok da farketmez) OHCI desteğini de çekirdeğe eklemekte fayda vardır.

### 12.2 Son birkaç adım

USB desteğini etkin hale getirip gerekli HCI sürücüsünü sağladıktan sonra yapılması gereken bir iki küçük ayar daha var. "Preliminary USB device filesystem"i etkin hale getirmeniz ve Linux ile kullanacağınız herhangi bir USB sürücüsünü sağlamanız ger ekir. Mesela USB oyun kontrollerini etkin hale getirmek için "USB Human Interface Device (full HID) support"u seçmeniz gerekebilir.

### 12.3 usbdevfs'yi bağlamak (mount)

USB desteği olan yeni çekirdeğinizi boot ettikten sonra USB aygıt dosya sistemini /proc/bus/usb altında bağlamanız gerekir

```
mount -t usbdevfs none /proc/bus/usb
```

Sistem boot ettiğinde otomatik olarak USB sistemi bağlayabilmek içinse fstabın içine /proc satırından sonra



```
none /proc/bus/usb usbdevfs defaults 0 0
```

satırını eklemek gerekir.

## 13 Diff, Patch

Unix’de herşeyin bir dosya olarak ele alındığını ve bu şekilde işlem gördüğünü daha önce anlatmıştık. Bu sebeple özellikle text işleme komutları oldukça gelişmiş durumdadır. Bir dosya içinden istenen kelimeleri ayıklama, onları sıralama vs. gibi işlemleri zaten daha önceki derselerde yapmıştık. Bir başka önemli işlem de bir dosyaya yapılan değişiklikleri bir değişikliktir dosyasında toplayıp, önceki dosya ve değişiklikleri kullanarak dosyanın son haline ulaşmak.

Bu iş için kullandığımız komutlar diff ve patch komutları.Şimdi bunları teker teker ele alıyoruz.

### 13.1 Diff

Önce bir dosya yarÖrnek olarak 2.4.18 kernelimize preemptive patchini uygulayacağız.atalım. Bu dosya bizim evimizde vereceğimiz bir partinin isim listesi olsun.

```
birinci.txt
```

ve bunun içinde şunları yazalım

```
Ali
Berkkan
Ebru
Tanya
Osman
Mehmet
Kemal
```

Ve kaydedelim. Şimdi ise aynı dosyayı değiştirip, partiye katılamayacaklarını belirtmiş olan Osman, Mehmet ve Ebru’yu çıkartalım. Yerine Mine ve Ela’yı koyalım, partimizde daha çok kız olsun ;) Dosyanın son hali

```
Ali
Berkkan
Tanya
Kemal
Ela
Mine
```

olacak. Bu halini de son.txt diye kaydedelim ve şu komutu çalıştıralım.

```
diff birinci.txt son.txt
```

```
3d2
< Ebru
5,6d3
< Osman
< Mehmet
7a5,6
> Ela
> Mine
```

Belki ilk bakışta çok anlamlı gelmeyebilir bu çıktı ama aslında çok da anlamlı ed komutlarıdır bunlar. Açıklamak gerekirse şu anlamlara gelmektedirler

birinci.txt dosyasının üçüncü satırı ikinci dosyada yok onu sil birinci.txt dosyasının beşinci ve altıncı satırları olan Osman ve Mehmet ikinci dosyada yok onları sil ikinci dosyanın beşinci ve altıncı satırları ilk dosyada yok, onları yedinci satırdan itibaren ekle (Kemal’in ardına)

şimdi çok popüler birisi olduğunuzu ve bu listenin binlerce kişi uzunluğunda olduğunu düşünün. Ya ikinci bir listeyi de herkese göndereceksiniz, yada bu birkaç satırlık diff dosyası ile bu işi kolayca halledeceksiniz.

Tabi bunun için sözkonusu değişiklikleri bir dosya haline getirmemiz gerekmekte, bu da ilk derste öğrendiğimiz yönlendirme komutları ile gerçekleşmekte:

```
$ diff birinci.txt son.txt > degisiklik.txt
```

Artık elimizde iki dosya var birinci.txt ve degisiklik.txt dosyası.  
şimdi yapmamız gereken

```
$ patch birinci.txt degisiklik.txt
patching file birinci.txt
```

Herşey yolunda gittiye şu an birinci.txt dosyası ve son.txt dosyalarının aynı olması lazım. Bakalım aynı mıymışlar ?

```
$ diff birinci.txt son.txt
```

Herhalde diff gibi bir komutumuz varken, binlerce satırlık iki dosyanın aynı olup olmadığını gözümüzle kontrol edecek değildik :) Gördüğünüz gibi diff komutu hiçbir çıktı vermemiş, dolayısıyla bu iki dosyanın aynı olduğunu onaylamıştır.

Basitçe iki dosya için bu şekilde kullanılan patch ve diff komutları yüzlerce dosya için de kullanılabilir. Örnek olarak 2.4.18 kernelimize preemptive patchini uygulayacağız.

## 14 Çekirdeği yamamak (patch)

Linux çekirdeği, belli bir ekip tarafından geliştirilmekte olan bir yazılım projesidir. Bu ekibin dahilinde ve haricinde birçok kişi Linux çekirdek kodu ile ilgilenmektedir. Bu doğrultuda insanlar zaman zaman kendileri için geliştirdikleri özelliklere çekirdeğe eklemekte ve bundan diğer kişileri haberdar etmektedirler. Eğer bu değişiklikler, yeteri kadar faydalı iseler bir süre sonra zaten çekirdeğin bir parçası haline alırlar.

Diyelim ki bu türden bir çekirdek yaması biliyorsunuz, (örneğimiz de Linux'un daha etkin çalışmasını sağlayan pre-emptive yaması olsun) ve bunun dahil olduğu bir çekirdek derlemek istiyorsunuz. Normalde make xconfig çalıştırdığımızda Processor Type and Features altında bulunması gereken Preemptible Kernel seçeneği göreceğiniz gibi standart çekirdek kodunda bulunmamakta.

Bunun için yapmanız gereken sözkonusu yama dosyasını temin etmek ve bunu mevcut çekirdek kodunuzun bulunduğu /usr/src/linux altına kopyalamak. Bunun ardından ise o dizinin altında şu komutu veriyorsunuz

```
patch -p1 < preempt-kernel-rml-2.4.18-5.patch
patching file CREDITS
patching file Documentation/Configure.help
patching file Documentation/preempt-locking.txt
patching file MAINTAINERS
patching file arch/arm/config.in
patching file arch/arm/kernel/entry-armv.S
patching file arch/arm/tools/getconstants.c
patching file arch/i386/config.in
patching file arch/i386/kernel/entry.S
patching file arch/i386/kernel/i387.c
patching file arch/i386/kernel/smp.c
patching file arch/i386/kernel/traps.c
patching file arch/i386/lib/dec_and_lock.c
patching file arch/sh/config.in
patching file arch/sh/kernel/entry.S
patching file arch/sh/kernel/irq.c
patching file drivers/ieee1394/csr.c
patching file drivers/sound/sound_core.c
patching file fs/adfs/map.c
patching file fs/exec.c
patching file fs/fat/cache.c
patching file fs/nls/nls_base.c
patching file include/asm-arm/dma.h
patching file include/asm-arm/hardirq.h
patching file include/asm-arm/pgalloc.h
patching file include/asm-arm/smplock.h
patching file include/asm-arm/softirq.h
patching file include/asm-i386/hardirq.h
```

```

patching file include/asm-i386/highmem.h
patching file include/asm-i386/hw_irq.h
patching file include/asm-i386/i387.h
patching file include/asm-i386/pgalloc.h
patching file include/asm-i386/smplock.h
patching file include/asm-i386/softirq.h
patching file include/asm-i386/spinlock.h
patching file include/asm-sh/hardirq.h
patching file include/asm-sh/smplock.h
patching file include/asm-sh/softirq.h
patching file include/linux/brlock.h
patching file include/linux/dcache.h
patching file include/linux/fs_struct.h
patching file include/linux/sched.h
patching file include/linux/smp.h
patching file include/linux/smp_lock.h
patching file include/linux/spinlock.h
patching file include/linux/tqueue.h
patching file kernel/exit.c
patching file kernel/fork.c
patching file kernel/ksyms.c
patching file kernel/sched.c
patching file lib/dec_and_lock.c
patching file mm/slab.c
patching file net/core/dev.c
patching file net/core/skbuff.c
patching file net/socket.c
patching file net/sunrpc/pmap_clnt.c

```

Gördüğünüz gibi patch komutu, sözkonusu dosyayı alıp, o dosya doğrultusunda gerekli yerlerde gerekli değişiklikleri yaptı.

Şimdi tekrar make xconfig yapıyoruz ve görüyoruz ki biraz önce olmayan Preemptible Kernel seçeneği artık var.

Bu şekilde bir yada daha fazla yama ile çekirdeğimizi yamayabilir istediğimiz gibi yeni özellikler ekleyebiliriz. Burada unutulmaması gereken yamanın sürümü ile çekirdek kodunun sürümünün aynı olması gerektiği, aksi takdirde problemlerle karşılaşma ihtimalimiz yüksek.

## 15 Kaynaklar

Linux Kernel How-to kernel kurulum bilgilerini edinmek için iyi bir kaynaktır. LILO, Linux Crash Rescue How-TO dosyaları da sizlere yardımcı olabilir.

Linux Kernel'leri [www.kernel.org](http://www.kernel.org) adresinde bulunmaktadır

[www.linuxdoc.org](http://www.linuxdoc.org) adresinden Linux konulu hemen her türlü bilgiye ulaşabilirsiniz. Burada "Guides" başlığı altındaki belgeler özellikle oldukça faydalı olacaktır. Eric S. Raymond'un Unix ve Internet Fundamentals How-To belgesi de bilgilendirici olabilir.

Türkçe içerik olarak da yukarda sayılan bir çok belgenin türkçelerini [belgeler.org](http://belgeler.org) adresinde bulabileceğiniz gibi, [ileriseviye.org](http://ileriseviye.org) ve [fazlamesai.net](http://fazlamesai.net) gibi kaynaklardan da faydalanabilirsiniz.

## 16 TCP/IP Ağ İşlemlerine Giriş

Birkaç GNU/Linux yüklü makinadan oluşan Ethernet tabanlı basit bir YAA (Yerel Alan Ağı - LAN [Local Area Network]) kurmak çok sık karşılaşılan görece olarak basit bir iştir. Genellikle bilmeniz gereken tek şey Linux yüklü bilgisayarlarınızda bir tür Ethernet ağ kartı bulunup bulunmadığıdır. Daha sonra merkezi bir Ethernet hub ya da switch cihazı ve uygun kablolama ile makinalar birbirine fiziksel olarak bağlanır. Eğer tüm makinaların çekirdeğinde ilgili Ethernet kartları ve TCP/IP desteği aktif halde ise artık makinalarınız yerel ağ üzerinden haberleşmek için her şeye sahip demektir.

Ancak tek başına Ethernet konusunun pek eğlenceli olduğu söylenemez. Bu aşamada tüm donanımınız ve çekirdeğiniz yerel ağ üzerinden iletişim kurmaya hazır olmakla birlikte bu durum tek başına yeterli değildir. Linux uygulamalarının büyük çoğunluğu ağ üzerinde ham Ethernet paketlerini (ya da çerçevelerini) kullanarak veri değiş tokuşu yapmaz. Bunun yerine, uygulamaların büyük bir kısmı TCP/IP

protokolü olarak isimlendirilen ve daha yüksek ve soyut seviyeli bir protokol kullanır. Mutlaka TCP/IP sözünü daha önce duymuşsunuzdur. Bu bir protokol süitidir ve bünyesindeki protokoller bugün Internet diye bildiğimiz devasa uluslararası ağ üzerindeki iletişimin temel kurallarını belirler (TCP/IP'nin açılımı: Transmission Control Protocol / Internet Protocol).

## 16.1 Çözüm: Ethernet üzerinden TCP/IP

Yapılması gereken Ethernet uyumlu ağımızdaki bilgisayarları TCP/IP protokolü ile iletişim kurabilecekleri şekilde ayarlamaktır. Bunun nasıl çalıştığını anlamak için önce Ethernet'i biraz anlamalıyız. Ethernet ile birbirlerine bağlı bilgisayarlardan oluşan bir ağda, her makinanın üzerindeki Ethernet kartının kendine özgü ve diğerlerinden farklı bir adresi vardır. Kartın fiziksel olarak içine üretilirken gömülen bu adres şuna benzer:

00:01:02:CB:57:3C

Ethernet sistemi CSMA/CD (Carrier Sense and Multiple Access with Collision Detection - Taşıyıcı Algılama ve Çarpışma Tespitli Çoklu Erişim) isimli bir yöntem kullanır. CSMA/CD sistemi kısaca şu demektir: tüm cihazlar tek bir fiziksel ortam üzerinden haberleşirler, belirli bir anda sadece tek bir cihaz yayın yapabilir ve tüm cihazlar eşzamanlı (simultane) olarak yapılan yayını dinleyebilirler. Eğer 2 farklı cihaz aynı anda yayın yapmaya kalkarsa iletişim çarpışması oluşur, bu tespit edilir ve her iki cihaz da çok kısa bir süre beklerler (bu süre belli bir aralıktaki gelişigüzel tayin edilir) ve tekrar yayın yapmaya çalışırlar. Böylece tek bir seferde tek bir cihazın belli bir uzunluktaki mesajını ağ ortamındaki bilgisayarlara iletmesi pratik olarak sağlanır, bir cihazın ürettiği ve ağ ortamında yayınladığı mesajın kime ait olduğu da mesaja eklenen ve yukarıda bahsi geçen Ethernet adresi sayesinde belirlenmiş olur.

## 16.2 IP adreslerine giriş

Bu donanım adresleri yerel ağdaki makinaları birbirlerinden ayırt etmeek için kullanılır. Donanım adresini kullanan bir makina bir başka makinaya bir Ethernet paketi yollayabilir. Ancak TCP/IP tabanlı iletişim yöntemleri farklı bir adresleme şekli kullanır ve bunlara IP adresleri denir. IP adreslerinin görüntüsü şuna benzer:

192.168.1.1

Her bir çevresel ağ cihazı için, çekirdekte, ilgili bir arabirimin varolması gerekmektedir. Örneğin Linux'da, ethernet arabirimleri, eth0 ve eth1 şeklinde isimlendirilmişlerdir. PPP arabirimleri ppp0 ve ppp1 olarak ve FDDI arabirimleri ise fddi0 ve fddi1 şeklinde isimlendirilmişlerdir. Bu arabirim isimleri, bir ayar komutunda fiziksel bir cihazı belirtmek istediğiniz zaman kullanılmakta ve bunun dışında bir kullanım alanları bulunmamaktadır.

TCP/IP ağı tarafından kullanılmadan önce, her bir arabirime dünyanın geri kalanıyla haberleşildiğinde kimlik numarasıymış gibi kabul edilen, bir IP adresi atanmalıdır. Bu adres, arabirim isminden farklıdır, eğer bir arabirimi kapı olarak düşünürseniz, adresi de o kapının üzerine yapıştırılmış isim plakası (ya da kapı numarası) olarak düşünebilirsiniz.

Donanımın bir kısmı tarafından ki bu donanım parçası En Büyük Aktarım Birimi (MTU - Maximum Transfer Unit) olarak bilinir, işlenen datagramların boyutlarını kısıtlamak gibi bir ayarın yapılması da mümkündür.

IP ağ protokolü, adresleri, 32 bitlik numaralar olarak görür. Her bir makineye, bulunduğu ağ ortamında tek olan bir numara atanmak zorundadır. Internet Protokolünün genelde 4. sürümü kullanılmaktadır. Yerini almak üzere tasarlanmış olan bir de 6. sürümü vardır. IPV6 değişik bir adresleme şeması kullanır ve daha büyük bir adresleme alanına sahiptir. Linux'da IPV6 desteği bulunmaktadır, ancak henüz bu kitabın kapsamına alınmamıştır. Linux çekirdeğindeki IPV6 desteği iyidir, ancak ağ ortamında kullanılmasını sağlayacak uygulamaların IPV6'ya göre değiştirilmesi gerekmektedir.

Eğer sadece, başka bir dış ağ ile ilişkisi bulunmayan, yerel bir ağda çalışıyorsanız, bu numaraları kendi istediğiniz şekilde atayabilirsiniz. Bazı IP adres aralıkları, belirli tipte özel ağlara ayrılmıştır. Ancak Internet üzerindeki siteler için, bu numaralar, NIC (Network Information Center - Ağ Bilgi Merkezi) olarak bilinen, merkezi bir otorite tarafından atanmaktadır. Çoğunlukla, IP adresleriniz sizin yerinize IP adresinizi alan bir sağlayıcı tarafından atanır. Bunun yerine, ağımız için kullanacağımız IP adresini doğrudan NIC'den hostmaster@internic.net adresine yazarak bir eposta ile ya da <http://www.internic.net/> adresindeki formu doldurarak alabilirsiniz.

IP adresleri, okunulabilirliği arttırmak amacıyla, öktet denilen, 4 adet 8 bitlik (1 baytlık - 0 dan 255 e kadar) numaraya ayrılmıştır. Örneğin quark.physics.groucho.edu adresindeki makinenin (Grouche Üniversitesinde, fizik bölümündeki quark isimli makinenin) IP adresi 0x954C0C04 (onaltılık sistemde) olsun.

Bu adres 149.76.12.4 olarak farklı bir gösterimde de yazılabilir. Bu ikinci gösterim, genelde noktalı dördümlü gösterim olarak bilinir.

Bu tip gösterimin başka bir nedeni ise, IP adreslerinin ilk öktetlerinin ağ numarasını, geri kalanların ise makine numarasını göstermeleridir. NIC'e IP adresi almak için başvurduğunuzda kullanmayı düşündüğünüz her makine için size bir IP verilmez, onun yerine size bir ağ numarası atanır ve bu aralıktaki tüm geçerli IP'leri tercihinize göre ağınızdaki makinelere atamanıza izin verilir.

Makine kısmının sayısı, ağın büyüklüğüne bağlıdır. Farklı ihtiyaçları karşılamak üzere, IP adreslerinde farklı noktalardan ayrılan, farklı ağ sınıfları tanımlanmıştır. Bu sınıflar şu şekilde tanımlanmışlardır:

**A Sınıfı:** A sınıfı ağlar 1.0.0.0'dan 127.0.0.0'a kadar olan ağları kapsar. Sadece ilk öktet, ağ numarasını belirler, geri kalanlar ise her bir ağ içinde kullanılabilecek (atanabilecek) makine sayısını belirler ki bu sınıfta, dolayısıyla, 24 bit makine kısmına ayrılmıştır. Bu da kabaca, ağ başına 1.6 milyon makine anlamına gelmektedir.

**B Sınıfı:** B sınıfı ağlar, 128.0.0.0'dan 191.255.0.0'a kadar olan ağları içermektedir. Ağ numarası, ilk iki öktet ile belirlenir. Bu sınıf ile 65,024 adet makineden oluşan 16,320 adet ağ tanımlanabilir.

**C Sınıfı:** C sınıfı ağlar ise 192.0.0.0 ile 223.255.255.0 arasında kapsar. İlk 3 öktet ağ numarasını belirler ve geri kalanlar da her bir ağ içinde bulunabilecek makine sayısını belirler. Dolayısıyla her biri 254 makineli yaklaşık 2 milyon ağ tanımlanabilir.

**D, E, ve F Sınıfları:** 224.0.0.0 ile 254.0.0.0 arasında kalan adresler, ya deneysel amaçlıdır ya da belirli amaçlar için ayrılmışlardır ve herhangi bir ağ tanımlamazlar. Bir internet üzerinde, paketlerin, bir seferde bir çok noktaya iletilmesi hizmetini veren IP çokluyayımı (IP Multicast) için bu aralıktaki adresler atanır.

Yukarıdaki 149.76.12.4 IP numaralı quark'ın adresi, B sınıfı olan 149.76.0.0 numaralı ağdaki 12.4 numaralı makineyi belirtir.

Bir önceki listede, tüm olası numaraların kapsanmadığını fark edebilirsiniz. Bunun nedeni 0 ve 255 numaralı öktetlerin özel amaçlar için ayrılmış olmasındandır. Bir adreste, host (konak) bilgisayar tarafındaki bitlerin tamamı 0 ise o adres ağı belirler, eğer bu bitlerin tamamı 1 ise bu adres yayın (broadcast, ağın tümüne yayın) adresi olarak bilinir. Buna göre 149.76.255.255 geçerli bir makine adresi belirtmez onun yerine 149.76.0.0 ağındaki tüm makineleri belirtir.

0.0.0.0 ve 127.0.0.0 ağı ise sizin kendi makinenizdeki IP trafiğini rahatlatmak amaçlı, sizin kendi makinenizi ifade eder. Birincisine öntanımlı rota (default route) diğerine de geridönüş (loopback) adresi denir.

Yani bilgisayarınız içinde kalan, yerel ağa çıkmanızı gerektirmeyen durumlarda 127.0.0.1 IP adresini kullanarak yine kendi bilgisayarınız içinde, programların haberleşmelerini sağlayabilirsiniz. Genelde, 127.0.0.1 adresi, kısa devreymiş gibi davranan, geridönüş arabirimi (loopback interface) olarak adlandırılan ve sizin kendi makinenizde bulunan özel bir arabirime ayrılmıştır.

### 16.3 IP adresleri ile Ethernet arayüzünü eşleştirmek

Ethernet tabanlı yerel ağın (LAN) TCP/IP ile çalışabilmesi için her makinenin Ethernet kart adresine (donanım adresine) bir IP adresi karşılık getirmeniz, başka bir deyişle bunları birbirleri ile eşleştirmeniz gerekir. Bunu Linux ortamında yapmanın kolay bir yolu vardır.

Aslında eğer halihazırda Linux ortamında Ethernet kullanıyorsanız mevcut Linux dağıtımınızın başlangıç betiklerinde şuna benzer bir komut vardır:

```
ifconfig eth0 192.168.1.1 broadcast 192.168.1.255 netmask 255.255.255.0
```

Yukarıdaki ifconfig komutu eth0'ı (dolayısı ile eth0'ın donanım adresini) 192.168.1.1 numaralı IP adresi ile eşleştirir. Bu işlevin yanı sıra aynı komut satırı IP ile ilgili diğer bilgileri de sisteme söyler: broadcast adresi (192.168.1.255) ve ağ maskesi (netmask) (255.255.255.0). Bu komutu çalıştırıldığında eth0 arayüzünüz aktive olmuştur ve bir IP adresine sahiptir.

### 16.4 ifconfig -a kullanımı

Halihazırda çalışmakta olan tüm ağ cihazlarınızı ifconfig -a komutunun çıktısında görebilirsiniz:

```
eth0
```

```
Link encap:Ethernet inet addr:192.168.1.1
```

```
HWaddr 00:01:02:CB:57:3C
```

```
Bcast:192.168.1.255
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500
```

```
Mask:255.255.255.0 Metric:1
```

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0
dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:100 RX
bytes:0 (0.0 b) TX bytes:0 (0.0 b) Interrupt:5 Base address:0xc400
Link encap:Local Loopb
ack inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:1065 errors:0 dropped:0 overruns:0 frame:0 TX packets:1065 errors:0
dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:2
62542 (256.3 Kb) TX bytes:262542 (256.3 Kb)
```

```
lo
```

Yukarıda ayarları yapılmış bir eth0 arayüzünün yanısıra bir lo (localhost) arayüzü görmektesiniz. lo isimli arayüz özel bir sanal ağ arayüzüdür ve herhangi bir ağ bağlantısı olmaması halinde de TCP/IP uygulamalarının bilgisayarda yerel olarak çalışmalarını sağlamak için geliştirilmiştir.

## 16.5 TCP/IP çalışıyor!

Bir kez tüm ağ arayüzleriniz aktive edilip IP adresleri ile eşleştirildi mi Ethernet tabanlı ağınız artık TCP/IP trafiğini taşımak için kullanılabilir. Sisteminizdeki bilgisayarlar birbirlerine IP adreslerini kullanarak erişebilirler ve bilgisayarlarınıza ping, telnet, ssh gibi komutlarla erişebilirsiniz.

## 16.6 İsim çözümleme sınırları

Şu aşamada her ne kadar bilgisayarlarınıza ping 192.168.1.1 gibi komutlarla erişseniz de onlara tanıdık isimlerle hitap edemezsiniz. Söz gelimi ping mybox yazarsanız çalışmaz. Bunun için yapmanız gereken her Linux yüklü bilgisayarda /etc/hosts isimli bir dosya oluşturmak ve bu dosyanın içinde IP adreslerine karşılık istediğiniz isimleri yazmaktır.

Yani mesela üç bilgisayardan oluşan bir ağ var ise /etc/hosts dosyası şunun gibi olacaktır:

```
127.0.0.1    localhost
192.168.1.1  mybox mybox.gentoo.org
192.168.1.2  testbox testbox.gentoo.org
192.168.1.3  mailbox mailbox.gentoo.org
```

Söz konusu /etc/hosts dosyasında zorunlu olarak "localhost" 127.0.0.1 eşleştirmesi olduğuna dikkat edin. Ayrıca yerel ağdaki tüm bilgisayarların isimleri (hostname) kısa ve uzun formatta tanımlanmıştır. Bu /etc/hosts dosyasını diğer bilgisayarlara da kopyaladıktan sonra artık bilgisayara IP adresleri yerine tanımladığımız isimlerle hitap edebiliriz, mesela artık ping mybox gibi bir komutu çalıştırabiliriz!

## 16.7 DNS kullanımı

Küçük çaplı yerel ağlarda yukarıdaki yöntem pratik gibi görünebilir ama bilgisayar sayısı arttıkça isim-adres eşleştirmesinin bu şekilde yapılması çok baş ağrıtır. Bu tür durumlarda "IP-makina ismi" eşleştirmesini tek bir bilgisayarda depolayıp bir "DNS sunucusu" çalıştırmak çok daha pratik olacaktır (DNS - domain name service server - alan adı servisi sunucusu). Böylece her makinayı öyle bir ayarlayabiliriz ki bir isme karşılık hangi IP adresinin karşılık geldiğini tek bir merkezi makineye sorup öğrenebilir. Bunu yapmanın yolu her makinada /etc/resolv.conf isimli bir dosya oluşturmaktır. Bu dosyanın içeriği şunun gibidir:

```
domain gentoo.org nameserver 192.168.1.1 nameserver 192.168.1.2
```

Yukarıdaki `/etc/resolv.conf` ayar dosyasına göre uzun formatta verilmeyen her isim (`testbox.gentoo.org` yerine `testbox` demek gibi) yerel ağdaki bir makinanın ismi olarak kabul edilmelidir. Yine yukarıdaki dosyadaki bilgiler `192.168.1.1` adresli bir makina da bir DNS sunucusunun çalıştığını ve `192.168.1.2` adresli makina da yedek DNS sunucusunun beklemekte olduğunu söylemektedir. Aslında ağa bağlı hemen hemen tüm Linux PC'ler `resolv.conf` dosyasında öntanımlı bir isim sunucu içerirler (ağa aktif olarak bağlı olmasalar bile). Bunun sebebi ISS (Internet Servis Sağlayıcı) şirketinin sunduğu DNS sunucuyu kullanmak durumunda olmalarıdır.

Böylece kullanıcılar Web tarayıcılarını açıp, bir IP adresi yerine `ibm.com` gibi akılda kolay kalan bir isim yazıp web sitesine bağlanabilirler.

## 16.8 Dışarıya bağlanmak

Madem az önce Internet'e bağlanmaktan bahsettik 3 sistemden oluşan basit ağımızı 'dışarıya nasıl bağlayabiliriz' konusuna gelelim. Tipik olarak bir tür 'router' satın alabilir ve bunun bir ucunu bizim Ethernet tabanlı yerel ağımıza diğer ucunu da DSL, kablo modem, T1 ya da normal telefon hattına bağlayabiliriz. Bu router'a belli bir IP adresi verilir ve böylece bu cihaz ağımızdaki sistemlerle iletişim kurabilir. Daha sonra ağımızdaki her bilgisayarı öntanımlı olarak bu router'ı Internet'e çıkış kapısı (gateway) kabul edecek şekilde düzenleyebiliriz. Yani sistemdeki makinalardan birinin adresinin içermeyen her veri paketi router'a yönlendirilir ve sonra ağın dışına çıkması gereken bu paketin yönlendirilmesi ile artık router ilgilenir. Genellikle kullandığımız dağıtımın başlangıç betikleri (initialization scripts) sizin için öntanımlı olarak router ayarlarını yaparlar. Bunu gerçekleştirmek için sisteminiz çalıştırılırken aşağıdakine benzer bir komut kullanılır:

```
route add -net default gw 192.168.1.80 netmask 0.0.0.0 metric 1
```

Yukarıdaki komuta göre öntanımlı router adresi `192.168.1.80`'dir. Sisteminizde ayarları yapılmış tüm routerlarla ilgili bilgileri edinmek için `route -n` komutunu verebilirsiniz. Hedef adresi olan `"0.0.0.0"` yol öntanımlı yoldur.

## 16.9 Ev ödevi

Şu ana dek Linux ağ işlemlerine dair en temel kavramlardan bahsettik. Maalesef uygun IP adresini seçmek, ağ maskeleri, broadcast adresleri, vs. gibi her konuya değinecek yerimiz ve vaktimiz yok. Bununla birlikte Linux ve ağ işlemleri hakkında en çok ve en detaylı dokümantasyon üretilmiş konudur. Buradaki temel bilgileri kavradıktan [Linuxdoc.org](http://linuxdoc.org) sitesindeki Guides kısmında yer alan Linux Network Administrator Guide'nin özellikle 2 ile 6 arasındaki bölümlerini okumanız sizin için çok faydalı olacaktır!

## 17 Internet servisleri ve inetd'ye giriş

Tek bir Linux sistemi düzinelerce hatta yüzlerce ağ servisi sunabilir. Örneğin telnet programını kullandığınızda uzaktaki bir sistemin telnet hizmetine erişirsiniz. Ya da mesela ftp programını kullandığınızda uzaktaki bir sistemin ftp hizmetine erişirsiniz. Bu servisleri sunmak isteyen uzaktaki sunucu sistem ya tek tek her bir servisle ilgili yazılımı çalıştırır ve bağlantı kurulmasını bekler (örn. `/usr/sbin/in.telnetd` ve `/usr/sbin/in.ftpd` programları) ya da `inetd` isimli bir yazılım çalıştırır. `inetd` yazılımı gelen bağlantı kurma taleplerini dinler ve uygun servisle ilgili yazılımı çalıştırır. Bu yüzden de `inetd` aynı zamanda "Internet süpersunucusu" olarak da bilinir. Tipik bir Linux kurulumunda `inetd` gelen bağlantıların çoğu ile başa çıkmak üzere ayarlanmıştır. Sadece birkaç program (`sshd` ve `lpd` gibi) `inetd`'den bağımsız olarak ağdan gelen ve kendilerini ilgilendiren veri paketleri ile ilgilenirler.

### 17.1 inetd'yi ayarlamak: `/etc/services`

Az önce `inetd`'nin gelen bağlantı taleplerini türlerine göre sınıflandırdığını söylemiştik. Her bağlantı talebi temelde bir TCP/IP paketidir ve başlık kısmında bazı tanımlayıcı bilgiler barındırır. Bizi en çok ilgilendiren kısımlar kaynak adresi, hedef adresi, protokol ve port numarası bilgilerini içeren kısımlardır. Bilgisayar gelen bağlantı talepleri `inetd` tarafından port numarası ve protokole göre (genellikle TCP ya da UDP, `inetd` tarafından sunulabilen tüm servislerin bir listesi için bkz. `/etc/protocols`) sınıflandırılır. Her satırın formatı aşağıdaki gibidir:

```
service-ismi port-numarası/protocol-ismi aliases # yorumlar
```

Örnek bir `/etc/services` dosyasında bizi ilgilendiren satırlara bir göz atalım:

```
# grep ^[^\#] /etc/services | head -5

tcpmux echo echo discard discard 1/tcp 7/tcp 7/udp 9/tcp 9/udp # TCP port service multiplexer

sink null sink null
```

Genellikle `/etc/services` halihazırda faydalı tüm servis isimlerini ve port numaraları bilgilerini içerir.

Eğer kendi servislerinizi ve port numaralarınızı ekleyecekseniz öncelikle atamaları yapılmış port numaralarını bilmeniz gerekir.

## 17.2 inetd'yi ayarlamak: `/etc/inetd.conf`

inetd yazılımının asıl konfigürasyon dosyası `/etc/inetd.conf`'dir ve şu formata sahiptir:

```
service-name socket-type protocol wait-flag user server-program
```

Servisler `inetd.conf` içinde port numarası ile değil servis ismi ile anıldıkları için bunların önceden `/etc/services` dosyasında tanımlanmış olması gerekir.

Örnek bir `/etc/inetd.conf` dosyasında bizi ilgilendiren satırlara bir göz atalım (örn. telnet ve ftp):

```
# grep ^telnet /etc/inetd.conf

telnet stream tcp nowait root /usr/sbin/in.telnetd

# grep ^ftp /etc/inetd.conf

ftp stream tcp nowait root /usr/sbin/in.ftpd -l -a
```

Her iki servis için de kullanılan TCP protokolüdür, servislerle ilgili yazılımlar (`in.telnetd` veya `in.ftpd`) root kullanıcısı olarak çalıştırılacaktır. `/etc/inetd.conf` dosyasındaki bilgi giriş alanlarının detaylı açıklaması için `inetd(8)` man sayfasına bakabilirsiniz.

## 17.3 Servisleri iptal etmek

inetd ayar dosyasında bulunan bir servisi iptal etmek çok basittir. Sadece ilgili satırın başına bir diyez (#) sembolü koymak sureti ile o satırı yorum satırı haline getirin. Satırı silmemeniz menfaatiniz icabıdır çünkü daha sonra o servisi yine etkin hale getirmek isteyebilirsiniz. Örneğin bazı sistem yöneticileri güvenlik sebeplerinden ötürü (iletişim şifreli olarak gerçekleştirilmediği için) telnet servisini iptal ederler:

```
# vi /etc/inetd.conf [istenmeyen satırın başına # karakterini koyun ve kaydedip çıkın]
# grep ^.telnet /etc/inetd.conf
# telnet stream tcp nowait root /usr/sbin/in.telnetd
```

## 17.4 inetd'yi bir başlangıç betiği ile başlatmak/durdurmak

Yukarıda bahsettiğimiz türden değişiklikleri `/etc/inetd.conf` dosyası üzerinde yaptıktan sonra bunların geçerli hale gelebilmesi için `inetd` servisini yeniden başlatmanız gerekmektedir. Pek çok Linux dağıtımı bünyesindeki `/etc/init.d` ya da `/etc/rc.d/init.d` dizininde ilgili betik dosyasını barındırır:

```
# /etc/rc.d/init.d/inet stop
Stopping INET services: # /etc/rc.d/init.d/inet start Starting INET services:
```

In fact, you can usually use "restart" as a shortcut:

```
[ OK ]
```

```
[ OK ]
```

```
# /etc/rc.d/init.d/inet restart Stopping INET services: Starting INET services:
```

```
[ [ OK OK ] ]
```



## 17.5 inetd'yi manuel olarak durdurmak/başlatmak

Eğer yukarıda bahsi geçen betik sizin için işe yaramazsa babadan kalma yöntemi kullanabilirsiniz, bu daha da kolaydır. inetd yazılımını durdurmak için killall komutu işinizi görür:

```
# killall inetd
```

Başlatmak için komut satırından komutu yazmanız yeterli olacaktır. Otomatik olarak arkaplanda çalışmaya başlar:

```
# /usr/sbin/inetd
```

Bunların dışında bir de inetd'yi durdurmadan ayar dosyasını yeniden okumasını sağlamanın kestirme bir yolu vardır: prosese HUP sinyalini göndermek:

```
# killall -HUP inetd
```

Bu aşamada sisteminize telnet ya da ftp programları ile bağlanamamanız gerekiyor çünkü telnet ve ftp servisleri iptal edilmiş durumda. Emin olmak için telnet localhost komutunu deneyin. (Eğer bu sisteme telnet ya da ftp ile erişmeniz gerekiyorsa yapmanız gereken az önce # ile iptal ettiğiniz satırların başındaki # karakterini kaldırmak ve inetd'yi yeniden aktive etmektir):

```
# telnet localhost
telnet: Unable to connect to remote host: Connection refused
```

## 17.6 TCP wrappers'a giriş

tcp\_wrappers paketi tcpd isimli küçük bir daemon olup inetd tarafından ilgili servis yerine çağrılır. tcpd programı her gelen bağlantı talebinin kaynak adresini bir log dosyasına kaydeder ve sadece güvenilir sistemlerden gelen bağlantılara izin verecek şekilde filtreleme yapabilir. tcpd, kullanmak için yapmanız gereken inetd ayar dosyasına aşağıdaki gibi bir satır yazmaktır:

```
ftp      stream  tcp  nowait  root  /usr/sbin/tcpd  in.ftpd  -l -a
telnet   stream  tcp  nowait  root  /usr/sbin/tcpd  in.telnetd
```

## 17.7 TCP wrappers ile kayıt tutma

Öntanımlı olarak bağlantılar sınırlandırılmamıştır ancak bunların kaydı tutulur. Örneğin yukarıdaki değişikliği yaptıktan sonra etkin olması için inetd'yi yeniden başlatalım ve kısa bir inceleme kaydı tutulan bağlantıları görelim:

```
# telnet localhost login: (press <ctrl-d> to abort)
# tail -1 /var/log/secure
Feb 12 23:33:05 firewall in.telnetd[440]: connect from 127.0.0.1
```

telnet ile bağlanma talebi gördüğünüz gibi tcpd tarafından log dosyasına kaydedilmiştir ve buradan da tcpd yazılımının çalışmakta olduğunu anlayabiliriz. tcpd sürekli bir kayıt hizmeti sunduğundan tek tek servislerin log tutma işi ile uğraşmasına gerek kalmamış olur. Bu bakımdan tcpd'nin yaptığı iş, bağlantıları yakalayıp ilgili servisleri başlatan inetd'nin yaptığı işe benzer. Linux'un (UNIX'in) basitliği olağanüstü değil mi?

## 17.8 TCP wrappers kullanarak erişimi yerel kullanıcılar ile kısıtlamak

tcpd programı ile ilgili olarak iki ayar dosyası vardır: /etc/hosts.allow ve /etc/hosts.deny. Bu dosyalar aşağıdaki şekle benzerler:

```
daemon_list : client_list [ : shell_command ]
```

Erişim izni şu sıraya göre verilir ya da reddedilir.

- Arama ilk bulunan eşleşmede durdurulur.
- /etc/hosts.allow dosyasında eşleşen bir kullanıcı bulunduğunda erişim izni verilir.

- /etc/hosts.deny dosyasında eşleşen bir kullanıcı bulunduğunda erişim reddedilir. Eğer bir eşleşme bulunmaz ise izin verilir.

Örneğin sadece kendi makinamızdaki kullanıcılara izin vermek istersek /etc/hosts.deny dosyasında bir erişim politikası oluştururuz (kaynağı localhost olanlar haricindeki tüm bağlantıları reddet):

```
in.telnetd: ALL EXCEPT LOCAL
```

## 17.9 TCP wrappers ile sadece bilinen kullanıcılara izin vermek

Bu aşamada inetd'yi yeniden yüklemeye gerek yoktur çünkü her telnet talebinde tcpd yeniden çağrılır. Hemen bir deneme yapabiliriz:

```
# telnet box.yourdomain.com
Trying 10.0.0.1... Connected to box.yourdomain.com. Escape character is '^]'.
Connection closed by foreign host.
```

Al sana! Reddedildin! (Reddedilmenin başarı anlamına geldiği ender anlardan biridir, tadını çıkarın ;-). Ağ içindeki kullanıcılara izin vermek için istisnai durumu /etc/hosts.allow dosyası içine yazmamız yeterlidir:

```
in.telnetd: .yourdomain.com
```

Bu noktada artık sisteme telnet ile yeniden girebilir hale geldik. Şimdiye kadar anlattıklarımız tcp-wrappers yazılımının sadece birkaç yüzeysel özelliğine işaret ediyordu. tcp-wrappers ile ilgili çok daha detaylı bilgi mevcuttur ve bunları tcpd(8) ile hosts\_access(5) man sayfalarında bulabilirsiniz.

## 17.10 xinetd: geliştirilmiş (extended) inetd

inetd klasik ve geleneksel Internet süpersunucusu olmakla birlikte son dönemlerde bu sistem yeniden yazılmış ve güvenlikle de ilgili olmak üzere daha özellikli hale getirilmiştir. Red Hat ya da Debian gibi pek çok modern Linux dağıtımında xinetd programı inetd'nin yerini alır. Bu gelişmiş yazılımın özellikleri arasında şunlar da vardır:

- Erişim kontrolü (TCP wrappers hazır olarak gelir)
- Gelişmiş kayıt tutma yeteneği (bağlantı süreleri, başarısız bağlantılar, vs.)
- Başka bir makinadan servislerin yönlendirilmesi
- IPv6 desteği
- Tek bir ayar dosyası yerine parçalı ayar dosya yapısı

## 17.11 xinetd ayarlaması

xinetd ayar dosyası ismi /etc/xinetd.conf'dur. Genellikle bu dosya sadece birkaç satır barındırır ve birkaç ayar içerir, detaylı ayarlamalar başka dosyalara havale edilir:

```
# cat /etc/xinetd.conf
```

```
defaults {
```

```
instances log_type = 60 = SYSLOG authpriv
```

```
log_on_success = HOST PID log_on_failure = HOST RECORD includedir /etc/xinetd.d }
```

Yukarıda gördüğümüz son satır xinetd'ye detaylı ayarları /etc/xinetd.d dizinindeki dosya parçacıklarından okumasını söyler. rectory. telnet ile ilgili bölüme bakalım:

```
# cat /etc/xinetd.d/telnet
```

```
service telnet {
```

```
flags socket_type wait user server = REUSE = stream = no = root = /usr/sbin/in.telnetd
```

```
log_on_failure += USERID }
```

Gördüğünüz gibi xinetd'yi ayarlamak zor değildir ve inetd'ye kıyasla daha kolaydır. xinetd ile ilgili detaylı bilgilere xinetd(8), xinetd.conf(5) ve xinetd.log(5) man sayfalarından erişebilirsiniz. Web üzerinde de inetd, tcp.wrappers ve xinetd ile ilgili bir hayli detaylı bilgi bulabilirsiniz. Lütfen bu dokümanın Kaynaklar kısmında verilen kaynakları incelemeyi ihmal etmeyin.

## 18 Güvenliğe Genel Bakış - Giriş

Tamamen güvenli bir sistem kurmak imkansızdır. Ancak sabırlı ve dikkatli şekilde çalışarak kuracağınız Linux sunucularınızı cracker'ların, script-kiddie'lerin ve diğer kötü adamların çoğundan koruyabilirsiniz. Bazı standart ve önemli noktalara dikkat ederek sisteminiz üzerinde denemeler yapmaya çalışan şüpheli (ve genellikle amatör) şahısları caydırabilirsiniz. Ancak unutmayın ki sadece burada anlatılan adımları takip ederek %100 güvenli bir sistem kurmanız mümkün değildir. Bunun yerine konu ile ilgili ana başlıklara değinecek ve güvenlik konusunun temellerine giriş yapmanız için basit örnekler vereceğiz. Linux sistem güvenliği kabaca ikiye ayrılabilir: dahili güvenlik ve harici güvenlik. Dahili güvenlik ile kast edilen sistemdeki kullanıcıların istemeden ya da bilinçli olarak sisteme zarar vermelerini engellemektir. Harici güvenlik ile kast edilen ise açık olarak izin verilip yetkilendirilmiş kullanıcılar haricindeki kullanıcıların sisteme izinsiz girişlerini engellemektir. Bu bölümde önce dahili sonra da harici güvenlik konularına kısaca değineceğiz ve genel yöntemler ile ipuçları ile bölümü tamamlayacağız.

### 18.1 Dosya izinleri ve log dosyaları

Dahili güvenlik ağdaki kullanıcılarınıza ne kadar güvenebileceğinize bağlı olarak bir hayli karmaşık bir görev haline alabilir. Buradaki yordamlar kısaca bir kullanıcının hassas verilere erişmesi ve sistem kaynaklarını sömürmesini engellemeye yönelik olacaktır. Dosya izinleri ile ilgili olarak, bunları üç durumda değiştirmek isteyebilirsiniz: Öncelikle /var/log dizinindeki dosyaları herkesin okuyabilmesi gerekmez. Sistemde root kullanıcısı dışında kimsenin log dosyalarına burnunu sokması için geçerli bir sebebi yoktur. syslog ile ilgili olarak LPI 101 Bölüm 4'e ve logrotate(8) man sayfasına bakabilirsiniz.

### 18.2 root kullanıcısının diğer dosyaları ile ilgili izinler

root kullanıcısının nokta ile başlayan dosyalarının da normal kullanıcılar tarafından okunmasının engellenmesi gerekir. ls -la komutu ile root kullanıcısının ev (home) dizinindeki dosyaların izinlerini ve düzgün şekilde korunup korunmadıklarını mutlaka kontrol edin. İsterseniz tüm dizini sadece root tarafından okunur hale de getirebilirsiniz:

```
# cd
# pwd
/root
# chmod 700 .
```

### 18.3 Kullanıcı dosyalarının dosya izinleri

Ve son olarak da kullanıcı dosyalarına bakalım. Genellikle bu dosyalar ilk yaratıldıklarında öntanımlı olarak herkes tarafından okunabilir şekilde yaratılırlar. Kullanıcıların çoğu böyle bir şeyi bilmez, beklemes ve zaten bunun iyi bir uygulama olduğu da söylenemez. Bunu engellemek için /etc/profile dosyasındaki umask aşağıdaki gibi bir şey yapabilirsiniz:

```
if [ "$UID" = 0 ]; then # root user; set world-readable by default so that
                        # installed files can be read by normal users.
umask 022 else # make user files secure unless they explicitly open them
                # for reading by other users
umask 077
fi
```

umask'la ve ayarlanması ile ilgili detaylı bilgi için umask(2) ve bash(1) man sayfalarına başvurabilirsiniz. umask(2) man sayfasının aynı isimli C fonksiyonundan bahsettiğine dikkat edin, bu sizi şaşırtmasın çünkü anlatılan bilgiler doğrudan bash içinde kullanılan umask komutu için de geçerlidir. Ayrıca LPI 101 Bölüm 3'te de bununla ilgili detaylar vardır.

## 18.4 SUID/SGID programlarını bulmak

İzinsiz olarak ve gizlice root erişimi sağlamaya çalışan kötü niyetli bir kullanıcının ilk yapacağı şey sistemde SUID ya da SGID biti ayarlanmış programlarını bulmaya çalışmak olacaktır. LPI 101 Bölüm 3'te bahsettiğimiz gibi bu bitler ilgili programların onlara sahip olan kullanıcı ya da grubun izinleri kapsamında çalışmasına yol açarlar. Elbette bu durum bazı programların düzgün olarak çalışması için gereklidir. Ancak sorun şu ki programdaki bir hata onu çalıştıran kullanıcıya gereğinden fazla hak tanınmasına yol açabilir. SUID ya da SGID biti ayarlanmış programları dikkatli şekilde incelemeli ve buna gerçekten ihtiyaçları olup olmadığına karar vermelisiniz. Sisteminizde hiç ihtiyaç duymadığınız SUID/SGID programları bile bulunuyor olabilir. Bu tür programları aramak için find komutundan faydalanabiliriz. İsterseniz gelin /usr dizinini bir tarayalım:

```
# cd /usr
# find . -type f -perm +6000 -xdev -exec ls {} \;

-rwsr-sr-x 1 root root 593972 11-09 12:47 ./bin/gpg
-r-xr-sr-x 1 root man 38460 01-27 22:13 ./bin/man
-rwsr-xr-x 1 root root 15576 09-29 22:51 ./bin/rcp
-rwsr-xr-x 1 root root 8256 09-29 22:51 ./bin/rsh
-rwsr-xr-x 1 root root 29520 01-17 19:42 ./bin/chfn
-rwsr-xr-x 1 root root 27500 01-17 19:42 ./bin/chsh
-rwsr-xr-x 1 lp root 8812 01-15 23:21 ./bin/lppasswd
-rwsr-x-- 1 root cron 10476 01-15 22:16 ./bin/crontab
```

Karşımıza çıkan listede dikkatle incelenmesi gereken bir aday var: lppasswd isimli dosya CUPS yazdırma sisteminin bir parçasıdır. Kendi sisteminizde yazıcı servisi sunmuyorsanız bu programa da ihtiyacınız yoktur. lppasswd programında herhangi bir hata ya da açık olmayabilir fakat kullanmadığımız bir programı sistemde bulundurarak neden risk alalım ki? Benzer şekilde kullanmadığımız tüm servisleri iptal etmekte fayda vardır. İhtiyacımız olduğu anda nasıl olsa kolayca bunları yeniden etkin hale getirebilirsiniz.

## 18.5 ulimit ile kullanıcıların limitlerini ayarlamak

bash kabuğundaki ulimit komutu belli bir kullanıcının hangi kaynağa ne ölçüde erişebileceğini belirlemenizi ve kısıtlamanızı sağlar. Bir kez bir limitin değeri azaltıldı mı o proses çalışırken bunu bir daha artırmanın yolu yoktur. Ayrıca o proses ile ilgili limitlere o prosesin tüm çocuk prosesleri de maruz kalır.

Bu kısaca şu demektir: /etc/profile dosyasında yapacağınız ulimit ayarları tüm kullanıcıları etkileyecektir (kullanıcıların bash ya da /etc/profile dosyasını okuyan bir kabuk yazılımını çalıştırdıklarını var sayıyoruz). Mevcut limit durumunu öğrenmek için ulimit -a komutunu kullanabiliriz:

```
# ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
file size (blocks, -f) unlimited
max locked memory (kbytes, -l) unlimited
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
stack size (kbytes, -s) unlimited
cpu time (seconds, -t) unlimited
max user processes (-u) 3071
virtual memory (kbytes, -v) unlimited
```

Bu limitleri sistemin güvenliğini artıracak ve kullanıcıları rahatsız etmeyecek şekilde ayarlamak çok zor olabilir bu yüzden ulimit ile ayarlama yaparken çok dikkatli olmalısınız.

## 18.6 ulimit ile CPU zamanını sınırlamak

Bir ulimit örnek uygulaması olarak bir prosesin CPU zamanını 1 saniye ile kısıtlamayı ve sonra süreci döngüye sokarak zaman aşımına uğratmayı deneyelim. Yeni bir bash süreci başlatmayı unutmayın aksi takdirde bu denemenin sonucunda sistemden çıkarsınız!

```
# time bash
# ulimit -t 1
# while true; do true; done
Killed

real  0m28.941s
user  0m1.990s
sys   0m0.017s
```

Yukarıdaki örnekte "user" zamanı artı "sys" zamanı proses tarafından kullanılan toplam CPU zamanını verir. bash prosesi 2 saniye işaretin gelince Linux çekirdeği 1 saniye limitinin geçildiğini tespit eder ve proses sonlandırılır. Harika değil mi?

Not: Bir saniyelik limit sadece bir örnektir. Sakın kullanıcılarınıza böyle bir şey yapmayın! Birkaç saatlik limit bile sorun yaratabilir çünkü X grafik sunucusu gerçekten CPU zamanını çok kullanır. Gerçek bir uygulama için CPU zamanı haricinde bir kaynağı kısıtlamayı düşünün.

## 18.7 Kullanıcı limitleri, devam

Eşzamanlı (simultane) loginler ya da disk kullanım miktarı gibi kaynakları da kısıtlamak isteyebilirsiniz ancak bunlar ulimit ile ilgili konular değildir. Bu tür kısıtlamalar için aşağıdaki yazılım paketlerini inceleyebilirsiniz:

- Clobberd yazılımı kullanıcı aktivitesini gözetler ve zaman, ağ kullanımı gibi değerleri ölçer.
- Idled yazılımı çok uzun süre iş yapmayan, boş duran kullanıcıları ya da uzun süredir sisteme bağlı olan kullanıcıları sistemden atabilir.
- Aynı zamanda kullanıcıların belli bir miktardan fazla login olmasını da engelleyebilir.
- LPI 101 Bölüm 4'te dosya sistemi kotaları ile kısıtlama anlatılmıştır.

## 18.8 Gizli girişlerin engellenmesi (intrusion prevention)

Harici güvenlik iki kategoride ele alınabilir: Gizli girişlerin engellenmesi ve gizli girişlerin tespit edilmesi. Gizli giriş engelleme tedbirleri yetkisiz kullanıcıların sisteme girişlerini engellemek içindir. Eğer bu tedbirler işe yaramazsa gizli giriş tespit sistemleri ile izinsiz girişin ne zaman gerçekleştirildiği ve hasar miktarı tespit edilebilir. Tam bir Linux kurulu sistem oldukça karmaşık bir sistemdir, kurulmuş olan her şeyin kaydını tutmak hele de her paketin güvenliği ile uğraşmak kolay bir iş değildir. Paket sayısı az ise bunları kontrol etmek de kolay olacaktır. Gizli giriş önlemenin ilk kuralı ihtiyaç duymayacağınız paketleri kurmamak ya da kaldırmaktır. Paket yönetim sistemlerine genel bakış için LPI 101'in 4. bölümüne bakabilirsiniz.

## 18.9 Kullanılmayan ağ servislerini kapatmak (süpersunucu)

Kullanılmayan ağ servislerini kapatmak gizli girişleri engellemek için atılacak ilk adımlardan biridir. Örneğin inetd ya xinetd gibi bir Internet süpersunucusu çalıştırıyorsanız genellikle in.rshd, in.rlogind ve in.telnetd gibi servisler öntanımlı olarak açıktır. Bu ağ servislerinin hemen hemen tamamı ssh gibi daha güvenli alternatiflere sahiptirler. inetd servislerini iptal etmek için /etc/inetd.conf dosyasındaki ilgili servis satırlarının başına "#" sembolü yazmak sureti ile bunları yorum satırı haline getirmek ve inetd'yi yeniden başlatmak yeterlidir.

xinetd kullanıyorsanız servisleri iptal etmek için benzer şeyleri /etc/xinetd.d içindeki ilgili dosya üzerinde yapmanız yeterli olacaktır. Örneğin telnet servisini kapatmak için ya /etc/xinetd.d/telnet dosyasının içeriğinin tamamını yorum satırı haline getirin veya basitçe dosyayı silin. Sonra xinetd'yi yeniden başlatın. Eğer inetd ile birlikte tcpd kullanıyorsanız ya da süpersunucu olarak xinetd kullanıyorsanız gelen bağlantı taleplerini sadece güvenilir bilgisayarlarla sınırlama imkanınız vardır. tcpd ile ilgili olarak bu kılavuzun önceki bölümlerine bakabilirsiniz.

xinetd için ise xinetd.conf(5) man sayfasında "only\_from" ifadesini aratın.

## 18.10 Kullanılmayan ağ servislerini kapatmak (tek başına çalışan sunucular)

Bazı ağ servis sunucuları inetd ya da xinetd tarafından başlatılmaz kendi başlarına çalışırlar. Birkaç örnek vermek gerekirse: atd, lpd, sshd, nfsd ve benzerleri. Aslında inetd ve xinetd de kendi başlarına çalışan sunuculardır ve ilgili ayar dosyalarındaki satırların hepsini yorum satırı haline getirdiyseniz bunları tamamen kapatmayı da düşünebilirsiniz. Kendi başına çalışan sunucular genellikle init sistemi tarafından bilgisayar açılırken ya da runlevel değişirken devreye sokulurlar.

init sisteminin belli bir sunucuyu başlatmasını engellemek için ilgili runlevel dizinindeki ilgili dizinin başlatma betiğine yönlendirilmiş sembolik bağlantıyı bulun ve bunu silin. runlevel dizinleri /etc/rc3.d veya /etc/rc.d/rc3.d gibi isimlere sahiptir (runlevel 3 için). Tabii bu arada diğer runlevel'larla ilgili dizinleri de kontrol etmek isteyebilirsiniz. Durdurmak istediğiniz servislerle ilgili sembolik bağlantıları sildikten sonra halihazırda aktif olan sunucuyu da durdurmanız gerekmektedir. Bunu yapmanın en iyi yolu ilgili servisin init betiğini kullanmaktır ki bu da genellikle /etc/init.d veya /etc/rc.d/init.d dizininde bulunur. Örneğin sshd yazılımı durdurmak için şuna benzer bir komut vermeniz yeterli olacaktır:

```
# /etc/init.d/sshd stop
* Stopping sshd... [ ok ]
```

## 18.11 Değişiklikleri test etmek

inetd ya da xinetd ayarlarınızı düzenledikten ve gerekli kısıtlamaları yaptıktan sonra veya bir sunucuyu init betiği ile durdurduktan sonra işlerin yolunda gidip gitmediğini anlamak için sisteminizi test etmelisiniz. TCP portlarını telnet istemcisi ile ve servis ismini ya da port numarasını vererek test edebilirsiniz. Örneğin rlogin servisinin iptal edildiğinden emin olmak için:

```
# grep ^login /etc/services
login 513/tcp

# telnet localhost 513
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
```

Standart telnet istemcisine ek olarak sisteminizin "açıklığı"nı test etmek için faydalanabileceğiniz birkaç becerikli yazılım daha vardır. netcat ve nmap isimli yazılımlar bunlar arasında en meşhur olanlardır. ncat bilgisayar ağlarının İsviçre Çakısı olarak bilinir, temel olarak yaptığı içi ağ bağlantıları kurup TCP ya da UDP protokollerini kullanarak veri okuyup yazmaktır. nmap ise ağı keşfetmek veya güvenliği denetlemek için kullanılan bir araçtır. Esas olarak nmap belli bir adresteki (ya da adreslerdeki) bilgisayarların üzerindeki açık portları tarayıp bunları tespit etmeye yarar. Bu faydalı programlarla ilgili web adreslerini en sondaki Kaynaklar bölümünde bulabilirsiniz.

## 18.12 Bakım yaparken kullanıcıların girişini engelleme

Yukarıdaki yöntemlere ek olarak loginleri yani sisteme girişleri engellemenin çok genel bir yöntemi vardır: /etc/nologin isimli bir dosya oluşturmak. Bu yöntem sadece kısa süreli sistem bakım işlemlerinde kullanıcıların sisteme girişini engellemek için kullanılır. root kullanıcısı sisteme girebilir fakat diğer (normal) kullanıcılar giremez. Örneğin:

```
# cat > /etc/nologin ===== System is currently undergoing maintenance until
2:00. Please come back later. ===
# telnet localhost login: agriffis

Password:
Login incorrect
===== System is currently undergoing maintenance until 2:00.
Please come back later. ===
```

Bakım işlemleriniz bitince söz konusu dosyayı silmeyi sakın unutmayın, aksi takdirde hiçkimse sisteme giremez. (Yok hayır, ben böyle bir dalgınlık sonucu insanlara saç baş yoldurtmadım, kesinlikle bunu yapan ben değilim ;-)

### 18.13 iptables (ipchains) konusuna giriş

iptables ve ipchains komutları çalışmakta olan bir Linux çekirdeğinin ağ paket filtreleme kurallarını incelemek ve ayarlamak için kullanılır. ipchains komutu 2.2.x sürümü çekirdekler için kullanılıyordu ve her ne kadar 2.4.x serisi çekirdekler üzerinde kullanılabilse de iptables komutu alıştırıcıdır. Paket filtreleme kuralları hem firewall hem de router aktivitelerini ayarlamak için kullanılabilir. Mevcut kuralları incelemek için iptables komutunu -L parametresi ile kullanabilirsiniz:

```
# iptables -L
Chain INPUT (policy ACCEPT) target prot opt source
Chain FORWARD (policy ACCEPT) target prot opt source
Chain OUTPUT (policy ACCEPT) target prot opt source

destination

destination

destination
```

Bu örnekte gördüğümüz sonuna kadar açık bir sistedir, herhangi bir routing ve firewall işlevi yoktur.

### 18.14 iptables ve Linux paket filtresi

Linux paket filtresini etkin olarak kullanmak sağlam bir TCP/IP bilgisini ve bunun Linux çekirdeğinde nasıl uygulandığını bilmeyi gerektirir. netfilter ana sayfası (kılavuzun sonundaki kaynaklar kısmına bakın) daha çok bilgi edinmek için uygun bir başlangıç noktasıdır. Kendi kural setinizi sağlıklı oluşturabilecek seviyeye gelene dek size yardımcı olabilecek birçok hazır betik vardır. Bunlardan en kapsamlı olanının ismi gShield isimli bir betik dosyasıdır (kaynaklar bölümüne bakın). Bünyesinde birçok açıklama satırı barındıran bu ayar dosyasını kullanarak en çok kullanılan paket filtreleme kurallarını sisteminizde aktive edebilirsiniz.

### 18.15 Gizli girişlerin tespiti - syslog dosyaları

Gizli girişlerin tespit konusu, sistemlerini özel gizli giriş tespit sistemlerine (IDS - Intrusion Detection System) emanet etmiş sistem yöneticileri tarafından genellikle ihmal edilir. Maalesef böyle bir durum, kötü niyetli biri söz konusu özel sistemde bir açık bulup bunu geçebilirse karşısına bir hayli kolay girilebilir bir sistem çıkacak demektir ve sistem yöneticisi uzunca bir süre bu durumda haberdar olmayabilir.

Gizli giriş tespiti yapmanın temel yolu sistem kayıtlarına dikkat etmekten geçer yani syslog dosyalarına. Bu dosyalar genellikle /var/log dizininde dururlar ancak isimleri kullandığınız dağıtıma göre bazı değişiklikler gösterebilir.

```
# less /var/log/messages

Feb 17 21:21:38 [kernel] Vendor: SONY Model: CD-RW CRX140E Rev: 1.0n
Feb 17 21:21:39 [kernel] eth0: generic NE2100 found at 0xe800,
Version 0x031243, DMA 3 (autodetected), IRQ 11 (autodetected).
Feb 17 21:21:39 [kernel] ne.c:v1.10 9/23/94 Donald Becker (becker@scyld.com)
Feb 17 21:22:11 [kernel] NVRM: AGPGART: VIA MVP3 chipset Feb 17 21:22:11
[kernel] NVRM: AGPGART: allocated 16 pages Feb 17 22:20:05 [PAM_pwd]
authentication failure; (uid=1000) -> root for su service
Feb 17 22:20:06 [su] pam_authenticate: Authentication failure
Feb 17 22:20:06 [su] - pts/3 chouser-root
```

Bu mesajların tamamının anlamını kavrayacak hale gelmeniz biraz vakit alabilir ancak en önemlileri kendilerini zaten belli eder. Örneğin yukarıdaki log dosyasının sonuna doğru "chouser" isimli bir kullanıcının su komutu ile root haklarına erişmeye çalıştığını ve başarısız olduğunu görüyoruz.

### 18.16 Gizli giriş tespiti - tripwire

Bazı özel yazılımlar mevcut dosya sisteminin bir tür fotoğrafını çekip bunu daha sonraki dosya sistemi yapısı ile hızlıca kıyaslayabilirler ve değişiklik olup olmadığına bakabilirler. Eğer sistemin normal işleyişi sırasında hangi dosyaların ya da dizinlerin değişmesinin kabul edilebilir hangilerinin kabul edilemez

olduğunu belirtirseniz bu yazılımlar sizi değişiklikler konusunda hızlıca uyarırlar: bazı hassas dosyalardaki küçük ve önemsiz gibi görünen değişiklikler sisteme gizlice girilip dosyalara müdahale edildiğine işaret edebilir. Tripwire bu tür yazılımlardan en meşhur olanıdır (bağlantı için kılavuzun sonundaki kaynaklar kısmına bakın). Bir kere kurduktan sonra tripwire ayar dosyasını kendi sisteminize ve ihtiyaçlarınıza göre ayarlamanız ve yazılımı düzenli olarak çalıştırıp (genellikle günde bir kez) sistemi olası şüpheli dosya değişimlerine karşı test etmeniz gerekir.

## 18.17 Gizli giriş tespiti - portsentry

Psionic Technologies tarafından geliştirilmiş olan PortSentry paketi gizli giriş önleme ile gizli giriş tespit etme kategorilerine giren melez bir uygulamadır. Çalıştırıldığı zaman bilgisayarınızın ağ trafiğini gözetler ve "şüpheli" kabul ettiği herhangi bir ağ trafiği ile karşılaşırsa bu olayın kaydını tutar ve aynı kaynaktan aynı tür olayın gerçekleşmesini engeller. Bu yazılımla ilgili bağlantılar da Kaynaklar bölümünde bulunabilir. Bir kez yazılımı kurup çalıştırdığımızda PortSentry'nin şüpheli eylemlere karşı neler yaptığını syslog dosyanızda görebilirsiniz:

```
# tail /var/log/messages
Oct 15 00:21:24 mycroft portsentry[603]: attackalert: SYN/Normal scan from
    host: 302.174.40.34/302.174.40.34 to TCP port: 111
Oct 15 00:21:24 mycroft portsentry[603]: attackalert: Host 302.174.40.34
    "ALL: 302.174.40.34" has been blocked via wrappers with string:
Oct 15 00:21:24 mycroft portsentry[603]: attackalert: Host 302.174.40.34 has
    been blocked via dropped route using command: "/sbin/route add -host
    302.174.40.34 reject"
Oct 15 00:21:24 mycroft portsentry[603]: attackalert: SYN/Normal scan from
    host: 302.174.40.34/302.174.40.34 to TCP port: 111
Oct 15 00:21:24 mycroft portsentry[603]: attackalert:
    Host: 302.174.40.34/302.174.40.34 is already blocked Ignoring
Oct 15 00:33:59 mycroft portsentry[603]: attackalert: SYN/Normal scan from
    host: 302.106.103.19/302.106.103.19 to TCP port: 111
Oct 15 00:33:59 mycroft portsentry[603]: attackalert: Host 302.106.103.19 has
    been blocked via wrappers with string: "ALL: 302.106.103.19"
Oct 15 00:33:59 mycroft portsentry[603]: attackalert: Host 302.106.103.19 has
    been blocked via dropped route using command: "/sbin/route add -host
    302.106.103.19 reject"
```

## 18.18 Genel tavsiyeler: Yazılımı Güncel Tutmak

Ne kadar test edilmiş olursa olsun hemen her yazılımın süreç içinde güvenlikle ilgili hataları tespit edilebilir. Bunlar tespit edildikten kısa bir süre sonra söz konusu yazılımların tamir edilmiş halleri de yayınlanır ve sistem yöneticisinin bunları güncellemesi gerekir. Az önce okumuş olduğunuz tavsiye güvenlik uzmanları tarafından en çok tekrarlanan ve amatör sistem yöneticileri tarafından da en çok görmezden gelinen tavsiyedir. Bir musibet bin nasihate bedeldir ama bu şekilde bir şeyler öğrenmek canınızı yakabilir: Bilgisayarlarımız bir yıldır bilinen ve güvenlik yamaları çoktan yayınlanmış (ve sizin tarafınızdan maalesef uygulanmamış) bir açık yüzünden zarar görürse üzülürsünüz.

"Açık kaynaklı yazılımlar mı yoksa kapalı kaynak kodlu yazılımlar mı daha güvenlidir?" tartışması ise uzun zamandı süren çok sıcak bir tartışmadır. Bu konuda söylenebilecek en iyi şey belki de şudur: düzgün olarak yönetildikleri sürece ve güvenlik yamaları zamanında uygulandığı sürece her iki sistem de güvenli kabul edilebilir. Pek çok web sitesi düzenli olarak tespit edilmiş açıkları ve bunları gideren yamaları yayımlar. Bunlardan en önemlileri ve güvenilir olanları CERT ve SecurityFocus' BugTraq listesidir. Ayrıca freshmeat.net gibi yazılım güncelleme haberlerini insanlara sunan siteler de yazılımlarla ilgili açıkları ve çözümleri duyururlar. Bir başka önemli kaynak da kullandığınız yazılımların resmi web siteleridir. Bahsi geçen önemli sitelerin adresleri kaynaklar bölümünde verilmiştir. Bu arada CERT ve Bugtraq sitelerini ziyaret edip aşına olursanız sizin için iyi olur.

## 18.19 Genel tavsiyeler: yüksek kaliteli parolalar

Sıradan ve sıkıcı bir tavsiye gibi görünse de kendiniz ve kullanıcılarınız için yüksek kaliteli parolalar seçmeniz, bu konuda kullanıcılarınızı eğitmeniz güvenli bir sistem kurmanın en temel aşamalarından biridir. Asla ve kat'a kolay tahmin edilebilen sözcükler, yakınlarınızın isimleri, telefon numaralarınız, doğumgünü tarihiniz, evde beslediğiniz hayvanın ismi, sevgilinizin adı, vb.gibi kolay tahmin edilebilir



bilgileri kullanmayın. Parolanızda mümkünse aynı zamanda hem harfler hem rakamlar hem de noktalama işaretleri bulunsun.

## 18.20 Genel tavsiyeler : Güvenliğinizi Test Etmek

Sisteminizin güvenliğini test etmek önemlidir ancak başarılı geçen bir testin ardından rahatlamak gibi bir lüksünüz yoktur. Belli bir teste göre sisteminizin güvenli görünmesi hiçbir açık barındırmadığı anlamına gelmez. Yeterli zamanı, azmi ve araştırma yeteneği olan kötü niyetli bir insanın sisteminize giremeyeceği garanti edilemez.

Yukarıdaki bölümlerde ağ güvenliği için nmap ve netcat yazılımlarından bahsetmiştik. Bunların yanı sıra zayıf parolaları tespit etmek de önemlidir. Bu konularla ilgili faydalı araçlar kılavuzun sonundaki Kaynaklar bölümünde yer almaktadır.

## 19 Yazdırma İşlemlerine Giriş

Bu bölümde Linux üzerinde de kullanılan ve bazen Berkeley LPD olarak da adlandırılan klasik UNIX yazdırma sisteminin kurulumundan ve kullanılmasından bahsedeceğiz. Linux üzerindeki yazdırma servislerine dair başka sistemler de mevcuttur ve bunları Kaynaklar kısmında görebilirsiniz. Bir yazıcının fiziksel olarak kurulması bu kılavuzun kapsamı dışındadır. Yazıcıyı bilgisayara bir kez fiziksel olarak bağladıktan sonra bir yazıcı "spooler" daemon'ı kurmak isteyeceksiniz ki ağdaki bilgisayarlar ve yazıcının bağlı olduğu bilgisayar yazıcıya yazdırılabilecek belgeleri gönderebilsinler.

### 19.1 Bir yazıcı spooler daemon kurulumu (lpd)

En iyi LPD yazıcı "spooler" yazılımlarından biri LPRng'dir. Kurulum yöntemi dağıtımınıza göre biraz değişiklik gösterebilir. Red Hat ya da Debian yazılım paketlerinin kurulumu için LPI 102 1. bölüme bakabilirsiniz. Bir kez kurulduktan sonra yazıcı spooler daemon'ı (resmi ismi ile Line Printer Daemon) komut satırından çalıştırılabilir. Normal bir kullanıcı olarak sisteme girin ve şunu deneyin:

```
$ /usr/sbin/lpd -help
--X option form illegal
usage: lpd [-FV] [-D dbg] [-L log]
Options
-D dbg - set debug level and flags
Example: -D10,remote=5
set debug level to 10, remote flag = 5
-F - run in foreground, log to STDERR
Example: -D10,remote=5
-L logfile - append log information to logfile
-V - show version info
```

Daemon kurulduğuna göre şimdi her seferinde otomatik olarak başlatılacağından emin olmalısınız. Dağıtımınızın LPRng paketi bunun sizin için çoktan ayarlamış olabilir ancak eğer yapmadı ise LPI 101 4. bölüm size runlevel'lar ve lpd gibi daemon yazılımlarının otomatik olarak başlatılması konusunda size bilgi verecektir.

### 19.2 Temel yazıcı ayarları (/etc/printcap)

Yazdırma spooler daemon'ı bir tür boruhattı gibi çalışır. Pek çok farklı noktadan kendisine gelen yazdırma işlerini kabul eder ve bunları uygun yazıcılara yönlendirir. Yazıcı meşgul iken bu işler "spool"a yani kuyruğa girerler ve yazdırılmak için sıralarını beklerler. Makinaya doğrudan bağlı bir yazıcıya yazdırırken bu boru hattının her iki ucu da /etc/printcap (ya da /etc/lprng/printcap) dosyasındaki ayarlarla belirlenir. printcap dosyasındaki her girdi (bu dosyanın ismi "printer capabilities" sözcüklerinin kısaltmasından gelir) bir yazdırma kuyruğunu gösterir:

```
$ more /etc/printcap
lp|Generic dot-matrix printer entry:\
:lp=/dev/lp0:\
:sd=/var/spool/lpd/lp:\
:pl#66:\
```

```
:pw#80:\
:pc#150:\
:mx#0:\
:sh:
```

Son girdinin en sonunda bir tersbölü ( ) bulunmadığına dikkat edin. Sizin sisteminizde başka girdiler de olabilir ve bunlar daha karmaşık olabilir ancak genel olarak bu dosyanın formatı yukarıda gördüğünüz gibidir. Önce girdinin ismi gelir, lp ardından da bu kuyruğun detaylı bir tanımlaması. lp=/dev/lp0 şeklindeki anahtar sözcük, değer ikilisi bu kuyruk-taki yazdırma işlerinin hangi cihaza yönlendirileceğini gösterirken sd anahtar sözcüğü de yazdırılacak işlerin hangi dizinde barındırılacağını belirtir. Anahtar sözcük, değer çiftlerinin geriye kalanları /dev/lp0 dosyasına fiziksel olarak nasıl bir yazıcı bağlı olduğunu belirtir.

### 19.3 Spool dizinlerinin yaratılması

Eğer bir girdi yaratırsanız buna paralel olarak ilgili spool dizinin doğru izinler verilmiş halde mevcut olduğundan emin olmanız gerekir. Genellikle lp kullanıcısı olarak çalışan yazıcı daemon'ının bu spool dizinine erişmesini isteyeceksiniz. root kullanıcısı olarak şu komutları vermeniz gerekir:

```
# mkdir -p /var/spool/lpd/lp
# chown lp /var/spool/lpd/lp
# chmod 700 /var/spool/lpd/lp
# checkpc -f
# /etc/init.d/lprng restart
```

LPRng printcap'ı kontrol edebilmeniz için faydalı bir araç içerir. Aynı zamanda, eğer siz unutursanız spool dizinini de ayarlar:

```
# checkpc -f
```

Son olarak lpd'yi tekrar başlatın. Ne zaman printcap'ı değiştirdiyseniz değişikliklerin geçerli olması için bunu yapmanız gerekir. lpd yerine lprng kullanmanız gerekebilir:

```
# /etc/init.d/lprng restart
```

Eski Berkeley yazdırma sistemi checkpc aracını içermez bu yüzden birden fazla yazıcınız var ise bunlara bastırılacak belgeler yollayarak printcap ve spool dizinlerinin doğru olup olmadığını kontrol etmeniz gerekir.

### 19.4 Yazdırma spooler istemcilerini kullanmak

Yazdırma spooler istemcisi sunucu daemon ile iletişim kurabilecek pek çok istemci ile birlikte gelir. En çok kullanılanı lpr programıdır ve yaptığı şey bir yazdırma kuyruğuna ilgili belgeyi yollayıp yazdırılmasını sağlamaktır. Bunu denemek için önce küçük bir metin dosyası oluşturun ve sonra da:

```
$ lpr sample.txt
```

Eğer çalıştı ise ekranda herhangi bir mesaj görmezsiniz fakat yazıcınız çalışmaya başlar ve dosyanızın kağıda basıldığını görebilirsiniz. Eğer çok düzgün görünmüyorsa üzülmeyin biraz sonra gerekli filtreleri nasıl kuracağımızı ve her türlü formatı nasıl yazıcıya gönderebileceğimizi inceleyeceğiz. Yazıcı kuyruğundaki yazdırma "iş"lerinin listesini lpq komutu ile öğrenebilirsiniz. -P seçeneği hangi kuyruğun gösterileceğini belirler eğer bunu es geçerseniz öntanımlı kuyruk listelenecektir:

```
$ lpq -Plp
Printer: lp@localhost 'Generic dot-matrix printer entry'
Queue: 1 printable job
Server: pid 1671 active
Unspooler: pid 1672 active
Rank Owner/ID Class Job Files Size Time
active chouser@localhost+670 A 670 sample.txt 8 21:57:30
```

Eğer bir yazdırma işini iptal etmek istiyorsanız lprm komutunu kullanmalısınız. Bu komut bir yazdırma işi çok uzun sürüyorsa ya da bir kullanıcı yanlışlıkla aynı belgeyi bir kereden daha çok göndermişse işe yarar. Tek yapmanız gereken yukarıdaki listeden job id'sini tespit etmek:

```
$ lprm chouser@localhost+670
Printer lp@localhost:
checking perms 'chouser@localhost+670'
dequeued 'chouser@localhost+670'
```

lpc ile yazıcı kuyruğunda pek çok şey yapabilirsiniz. Detaylar için lpc man sayfasına bakınız.

## 19.5 Uzaktaki bir LPD sunucusuna yazdırmak

Doğrudan kendi bilgisayarınıza bir yazıcı bağlı olmasa bile yine de lpd'yi kullanarak ağ üzerindeki bir bilgisayara bağlı bulunan yazıcıya belgelerinizi gönderebilirsiniz. İstemci makinada /etc/printcap dosyasına yerel bir yazıcı gibi görünen ama aslında yazdırma işlerini uzaktaki makineye yönlendiren bir girdi oluşturabilirsiniz:

```
farawaylp|Remote printer entry:\
:rm=faraway:\
:rp=lp:\
:sd=/var/spool/lpd/farawaylp:\
:mx#0:\
:sh:
```

Burada ağ üzerinde bulunan ve kendisine bir yazıcı bağlı olan makinenin ismi faraway'dir ve bu makinadaki yazıcının ismi de lp'dir. spool dizini /var/spool/lpd/farawaylp'dir ve uzaktaki makinenin spool dizinine gönderilmek üzere sırada bekleyen yazdırma işleri bu dizinde barındırılacaklardır. Tabii bu spool dizinini uygun izinleri de vererek yaratmanız gerektiğini unutmayın:

```
# mkdir -p /var/spool/lpd/farawaylp
# chown lp /var/spool/lpd/farawaylp
# chmod 700 /var/spool/lpd/farawaylp
# checkpc -f
# /etc/init.d/lprng restart
```

Yerel olarak uzaktaki yazıcıya farawaylp ismini verdiğimiz göre ona yönlendirilmiş işleri de şu şekilde görüntüleyebiliriz:

```
$ lpr -Pfarawaylp sample.txt
```

## 19.6 Uzaktaki bir MS Windows ya da Samba sunucusuna yazdırmak

Samba sayesinde uzaktaki bir Microsoft Windows yazıcı sunucusuna belge yollamak zor bir iş olmaktan çıkmıştır. Önce yerel printcap girdimizi oluşturalım:

```
smb|Remote windows printer:\
:if=/usr/bin/smbprint:\
:lp=/dev/null:\
:sd=/var/spool/lpd/smb:\
:mx#0:
```

Buradaki yeni anahtar if'dir yani girdi filtresi (input filter). Bunu smbprint betiğine yönlendirmek yazma işinin lp cihazı yerine uzaktaki bir Windows sunucusuna yönlendirmesine yol açacaktır. Bu durumda yine de yazdırma daemon'ın kilit işlemleri için kullanacağı bir cihaz listelemeliyiz: /dev/null. Ancak tabii ki hiçbir yazdırma işi bu cihaza gönderilmeyecektir. Spool dizinini yaratmayı unutmayın!

```
# mkdir -p /var/spool/lpd/smb
# chown lp /var/spool/lpd/smb
# chmod 700 /var/spool/lpd/smb
# checkpc -f
# /etc/init.d/lprng restart
```

Favori metin editörünüz ile ismi yukarıdaki gibi belirlenmiş spool dizininde bir .config dosyası oluşturun (örneğimizde: /var/spool/lpd/smb/.config):

```
server="WindowsServerName"
service="PrinterName"
password= " "
user=" "
```

Bu değerleri Windows çalıştıran bilgisayarı ve oradaki yazıcıyı gösterecek şekilde ayarlayın, ve işlem tamam:

```
$ lpr -Psmbl sample.txt
```

smbprint betiği genellikle Samba ile birlikte gelir ancak her dağıtımda bulunmayabilir. Eğer bunu sisteminizde bulamazsanız Samba HOWTO belgesinde bulabilirsiniz.

## 19.7 Magicfilter

Şimdiye dek sadece metin dosyaları ile ilgilendik ve bu da çok heyecan verici sayılmaz. Genellikle elinizin altındaki herhangi bir yazıcı tek bir grafik formatını basabilecek kapasitededir ancak basılmasını arzuladığımız pek çok farklı format mevcuttur: PostScript, gif, jpeg ve diğerleri. Magicfilter isimli program bir tür girdi filtresi olarak çalışır tıpkı smbprint'in yaptığı gibi. Dosya formatlarını dönüştürmez sadece basılacak belgenin tipinin belirlenmesini sağlayacak bir altyapı sunar ve sonra gerekli dönüştürme yazılımının çalıştırılmasını sağlar. Dönüştürme yazılımları ayrıca kurulmalıdır. Bunlardan en önemlisi ise Ghostscript yazılımıdır; bu yazılım Postscript formatındaki dosyaları yazıcının yerel formatına dönüştürebilir.

## 19.8 printcap'ı Magicfilter'i gösterecek şekilde ayarlamak

Bu araçlar kurulduktan sonra yapılması gereken printcap dosyasını bir kez daha ayarlamaktır. if anahtarı Magicfilter'i gösterecek şekilde yazıcınızı ayarlayabilirsiniz:

```
lp|The EPSON Stylus Color 777 sitting under my desk:\
:if=/usr/share/magicfilter/StylusColor-777@720dpi-filter:\
:gqfilter:\
:lp=/dev/usb/lp0:\
:sd=/var/spool/lpd/lp:\
:pl#66:\
:pw#80:\
:pc#150:\
:mx#0:\
:sh:
```

/usr/share/magicfilter içinde düzinelerce farklı yazıcı için ayarlar mevcuttur, yazıcınız için doğru olanı seçtiğinizden emin olun. Bunlardan her biri bir metin dosyasıdır ve yazıcının tam ismi en tepededir, bu şekilde hangi dosyaya ihtiyacınız olduğunu tespit edebilirsiniz. Ayrıca eklenmiş olan bir gqfilter seçeneği de uzaktaki bir istemciden gelen belgelerin yine filtreye yönlendirilmesini sağlar. Bu sadece LPRng ile çalışır. Daha önce /var/spool/lpd/lp yazıcı spool dizini daha önce ayarlanmış olduğu için tek yapılması gereken printcap söz dizimini kontrol etmek ve sunucuyu yeniden başlatmak:

```
# checkpc -f
# /etc/init.d/lprng restart
```

Artık Postscript dosyaları da dahil olmak üzere çok çeşitli belgeleri yazdırabilirsiniz. Başka bir deyişle favori web tarayıcınızın menüsündeki "Print" seçeneği artık düzgün şekilde çalışıyor olmalı.

## 19.9 Magicfilter alternatifi olarak Apsfilter

Apsfilter, Magicfilter'in sunduğu özelliklerin çoğunu sunar fakat aynı zamanda spool dizinler, printcap girdileri gibi şeyleri de kendisi ayarlar.

Ghostscript'in yine de kurulu olması gereklidir, ardından Apsfilter elkitabındaki detaylı açıklamaları okuyarak gerekli ayarlamaları yapabilirsiniz.

## 19.10 Kaynaklar

xinetd anasayfasından ayarlamalarla ilgil detaylı bilgi edinebilirsiniz. Kendi servis isimleriniz ve port numaralarınızı /etc/services dosyasına eklerken önceden ataması yapılmış olan portlar ve servislerle çakışmalarına dikkat edin. netfilter anasayfası iptables ile ilgili detaylı bilgi edinmek için birincil kaynaktır. Kendi kural setinizi kendi sisteminize göre hassas olarak kurabilecek kadar deneyim sahibi olana dek gShield gibi hazır bir iptables betiğini denemenizi tavsiye olunur.

İlk akla gelen güvenlik araçları arasında şunlar sayılabilir: Tripwire en popüler gizli giriş tespit araçlarından biridir.

Psionic Technologies'den gelen PortSentry ise hem tespit etme hem de önlemeye yöneliktir. (LinuxWorld'deki "How to stop crackers with PortSentry" başlıklı makale bu yazılımla ilgili detaylı bilgi verir)

Bunlara ek olarak Wietse Venema'nın TCP Wrappers yazılımına aşina olmanız menfaatiniz icabıdır, bu yazılım sisteminizdeki bağlantıları gözetlemenizi ve kontrol etmenizi sağlar.

Ağınız dibine dek açık mı? Bu tip analizler için şu iki temel yazılımdan faydalanabilirsiniz: netcat basit bir UNIX programıdır ve her türlü ağ bağlantısını istediğiniz TCP ya da UDP protokolü üzerinden kurmanızı sağlar.

nmap ise ağı keşfetmek ve güvenlik denetimi yapmak için kullanılan bir araçtır. nmap açık olan portları tarar.

Parola test etme yazılımları sisteminizdeki parolaların kalitesi konusunda size bilgi verir. Bu amaçla geliştirilmiş en meşhur yazılımlardan biri John the Ripper'dır. Daha detaylı bir test edici olarak SAINT yazılımına göz atabilirsiniz.

## 19.11 Her sistem yöneticisinin düzenli olarak takip etmesi gereken en önemli siteler:

CERT devlet tarafından finanse edilen ve Carnegie Mellon University tarafından işletilen bir merkezdir. İlgili odağı Internet güvenlik problemleridir. SecurityFocus tarafından barındırılan BugTraq da güvenlikle ilgili problemlerin düzenli olarak duyurulduğu en önemli listelerden biridir. Özel olarak güvenlik yönetimi ile ilgilenmiyorsanız bile bulisteye üye olmanız menfaatiniz icabıdır. Yapılan duyurular arasında sizin sisteminiz ile bağlantılı olanlar ilginizi çekebilir.

Linux güvenliği ile ilgili tavsiye edebilecek diğer sitelere gelince: Linux Security HOWTO, O'Reilly's Security Pageve tabii ki developerWorks'dan Security Zone.

Yazdırma konusu ile ilgili olarak LPRng anasayfası pekçok bilgi içerir. Tabii Printing HOWTO da değerli bir belgedir. Bu konu ile ilgili bir başka önemli kaynak da LinuxPrinting.org sitesidir. Spesifik yazıcılarla ilgili olarak Serial HOWTO belgesine danışabilirsiniz. USB kılavuzu da USB yazıcılarla ilgili bilgi verir.

Heterojen ağlarda Samba büyük bir yardımcıdır. Bu tür bir ortamda yazıcı servisleri ile uğraşacaksanız Samba ana sayfasını ve Samba HOWTO belgesini de okumakta fayda vardır. Burada iki filtreleme programı ele alınmıştır:

Magicfilter ve Apsfilter. Unutmayın ki bu programların bir dönüştürücüye ihtiyacı vardır (tercihen Ghostscript).

## 19.12 Etkileşimli, güvenli kabuk oturumu

Eskiden, ağ üzerinden etkileşimli bir oturum açmak istediğinizde telnet veya rsh kullanırdınız. Ancak ağ işlemlerinin popülerite kazanmasıyla bu araçların eskisi kadar uygun olmadığı ortaya çıktı. Çünkü telnet ve rsh kesinlikle güvensizdi. Sunucu ve istemci arasındaki bilgileri şifrelenmeden gönderiliyordu ve böylece ağı girebilen herhangi birisi tarafından kolaylıkla okunabiliyordu. Sadece bu kadarla kalmıyor, kimlik doğrulama işlemi için sunucuya yollana parola düz metin halinde gönderiliyordu, bu da ağ üzerinde data yakalayabilen herkes için sizin hesabınıza girmenin çok kolay olduğu anlamına geliyordu. Ayrıca bir ağ yoklayıcısı ile, tüm telnet oturumunuz tekrar oluşturulabilmesi ve sizin ekranda gördüğünüz herşeyin görüntülenebilmesi mümkündü. Sonuç olarak, ağların güvenli ve yoklanamaz olacağı varsayımıyla oluşturulmuş olan bu araçların, bugünkü çok dağılmış ve halka açık ağlar üzerinde kullanılması çok uygun olmadığı açık.

## 19.13 Güvenli kabuk

Daha iyi ve güvenli bir çözüme ihtiyaç doğmuştu. Bu çözüm güvenli kabuk (secure shell), bir diğer adıyla ssh oldu. Günümüzde çok popüler olan bu araç tüm Linux dağıtımlarında openssh paketi olarak bulun-

yor. Ssh'in, güvenli olmayan diğer araçlardan farkı, istemci ile sunucu arasındaki tüm iletişimi güçlü bir şifreleme mekanizmasıyla şifrelemesidir. Böylece istemci ve sunucu arasındaki iletişimi görüntülemek çok zor (hatta imkansız) hale geliyor. Bu da, güvenli kabuk ismini hak etmiş olmasını sağlıyor. Ssh çok özellikli bir güvenlik yapısına sahiptir, bir şekilde ağa girilse ve sunucu-istemci iletişimi görüntülense bile parola doğrulama için şifrelemenin yanında bazı anahtar değişimi stratejileriyle, parolanın çalınmasını imkansız kılar. İnternetin bu kadar popüler olduğu günümüzde, Linux sistemlerin kullanıldığı ağlarda, güvenliği üstü düzeye çıkarmak için ssh uygun bir araçtır. Güvenliğin önemini kavramış sistem yöneticileri, ağlarında telnet ve rsh kullanılmasını uygun bulmaz, hatta bazen yasaklarlar. Çünkü ssh aynı işi yapabilen ve çok daha güvenli bir alternatiftir.

## 19.14 ssh kullanımı

Çoğunlukla, tüm dağıtımlardaki openssh paketleri herhangi bir ayarlamaya ihtiyaç duyulmaksızın kullanılabilirler. openssh kurulunca birkaç ikilik dosyanız olacak. Bir tanesi sshd çalıştıran bir güvenli kabuk sunucusuna bağlanmanızı sağlayacak olan güvenli kabuk istemcisi: ssh. Ssh kullanmak ve bir oturum açmak için:

```
$ ssh knoppix@remotebox
```

yazarsınız. Burada remotebox makinesine knoppix kullanıcısıyla bağlanma komutu vermiş olduk. Telnet'teki gibi bir parola sorulacak, parolayı girdikten sonrada uzaktaki makinede sizin için yeni bir oturum açılmış olacak.

## 19.15 sshd başlatmak

Eğer makinenize ssh ile ulaşılmasına izin vermek isterseniz sshd sunucusunu çalıştırmanız gerekir. Bunun için openssh paketiyle gelen rc-script'i kullanmalısınız. Şöyle ki:

```
# /etc/init.d/sshd start
```

veya

```
# /etc/rc.d/init.d/sshd start
```

Eğer gerekirse, sshd yapılandırmasını /etc/ssh/sshd\_config dosyası üzerinden değiştirebilirsiniz. Daha fazla bilgi almak ve diğer tercihleri görmek için sshd kılavuz sayfasını kullanabilirsiniz:

```
man sshd
```

## 19.16 Güvenli kopya

Openssh paketi ayrıca çok kullanışlı bir araç daha içerir. Güvenli kopya (secure copy), scp, olarak adlandırılan bu araçla ağınız üzerinde kendi makinenizden başka makineye "veya ters yönde güvenli kopyalama işlemleri yapabilirsiniz. Örneğin, /foo.txt dosyasını remotebox makinesindeki home dizinine kopyalamak istersem:

```
$ scp ~/foo.txt knoppix@remotebox:
```

Parola girme ve doğrulama işleminden sora kopyalama yapılacaktır. Tersinde, örneğin remotebox makinesinin /tmp dizinindeki bar.txt dosyasını, yerel makinemde, halihazırda çalıştığım dizine kopyalamak istersem:

```
$ scp knoppix@remotebox:/tmp/bar.txt
```

yazarım.

## 19.17 Güvenli kabuk doğrulama ve yetkilendirme tercihleri

Openssh'in farklı doğrulama yöntemleri vardır. Doğru kullanılmak şartıyla, uzak sistemlere parola girmeksizin de ulaşabilirsiniz. Bu ayarlar yukarıda bahsettiğimiz /etc/ssh/sshd\_config dosyası üzerinden yapılabilir.

## 20 NFS

### 20.1 NFS'e giriş

Ağ Dosya Sistemi (The Network File System - NFS), yerel bir ağda, birbirine bağlı UNIX ve Linux sistemlerin, aralarında dosya paylaşımını sağlayan bir teknolojidir. NFS, Linux ve UNIX dünyasında uzun süredir bilinen ve yaygın olarak kullanılan bir sistemdir. NFS'in temel kullanımı, kullanıcıların, ağ üzerinde herhangi bir makinede oturum açtıklarında kendi home dizinlerine ulaşmalarını sağlamaktır. NFS sayesinde uzak makinelerin dosya sistemi ve ağaç yapılarının, yerel olarak oturum açılan makineye bağlanması (mount) ve tam uyumluluğu sağlanabiliyor. Bu yeterlilik ve basitlik, NFS'i, Linux dünyasında en popüler ve kullanışlı ağ dosya paylaşım sistemi haline getirmiştir.

### 20.2 Temel NFS bilgileri

NFS üzerinden dosya paylaşımı için önce bir NFS sunucu kurulması gerekir. Bu sunucu dosya sistemlerini ihraç etme özelliğine sahip olacak. Bir dosya sisteminin ihraç edilmesi, ağ üzerindeki diğer sistemler tarafından kullanılabilir hale gelmesi anlamındadır. Böylece, NFS istemci olarak kurulmuş bir sistemden, bu ihraç edilen dosya sistemini standart "mount" komutuyla bağlayabilirsiniz. Bağlama işlemi tamamlanınca, uzaktaki sistem, yerel makineye bağlanmış herhangi bir sistem gibi (örneğin /mnt/cdrom) kullanılabilir. Bu yapıda, yerel sisteme bağlanan dosya sisteminin bir disk'ten değil, sunucudan alındığına dikkat edin. Bu durum Linux sistemlerinin zaten uyumlu olduğu bir uygulamadır, bu yüzden çoğu zaman sorunsuz çalışır.

### 20.3 NFS'in özellikleri

NFS dosya paylaşım sisteminin bazı ilginç özellikleri vardır. Bu özelliklerden ilki NFS'in "başına buyruk" tasarımıdır. İstemcilerin sunucuya bağlantıları, kendi başına olduğundan, NFS sunucusundaki herhangi bir nedenle kapanıp-açılma durumu, istemcilerin bağlantısında bir kopmaya sebep olmaz. Sadece geçici olarak duraklatılan uzak NFS dosya bağlantıları, sunucu tekrar eski haline geldiğinde, kaldığı yerden devam edecektir. Yine bu başına buyruk özellik sayesinde, bir sunucu, çok sayıda istemciyi, ek bir yük olmaksızın, rahatlıkla kaldırabilir. Fazla istemci olmasının getireceği tek yük, ağ üzerinde dosya transferleri sırasında ortaya çıkan yoğunluk olacaktır. Özetle şunu söyleyebiliriz ki, NFS'in performansı, ağ üzerindeki istemci sayısına değil, veri transferi yoğunluğuna bağlıdır.

### 20.4 Linux'ta NFS, sürüm 3

NFS kurulacağı zaman sürüm 2 yerine sürüm 3'ü tercih etmenizi öneririz. Sürüm 2'nin, dosya kitleme ve bazı süreçlerde takılma konusunda olumsuz bir şöhreti vardı. Ama bu sorunlar sürüm 3'te çözüldü. Bu sürüm sağlam ve başarılı kabul ediliyor. Linux 2.2.18 ve üzeri çekirdekler NFS 3 sunucu ve istemcilerini destekliyorlar. Dolayısıyla NFS 3 varken, artık NFS 2 kullanmak için herhangi bir sebep yok.

### 20.5 NFS'te güvenlik

NFS sürüm 2 ve sürüm 3 te güvenlik nedeniyle bazı kısıtlamalar olduğunu belirtelim. NFS, özel olarak, güvenli bir ağ üzerinde çalışmak üzere tasarlanmış bir sistemdir. Yerel ağınızda güvenli olarak NFS kullanmak istiyorsanız, sunucu makinesinin root parolasının sadece ve sadece sistem yöneticilerinde olması gerekir. Eğer sıradan bir kullanıcı, root parolasına sahipse, NFS güvenlik duvarını kolayca aşabilir ve sunucu üzerinde, normalde ulaşamayacağı dosyalara ulaşarak sistem güvenliğini tehdit edebilir. Bu yüzden NFS sistemi plansız, geliş güzel kurulmamalıdır. Yerel ağınızda NFS kullanmaya karar vermeniz çok iyi ama önce sunucunuz için bir kalkan (firewall) kurmalısınız. Ayrıca yerel ağınız dışından NFS sunucunuza ulaşımın mümkün olmadığından emin olun. Sonrasında da, yerel ağınızın güvenli olduğundan emin olup, bu güvenlik kontrollerini sık sık yapmalı ve yeri geldikçe geliştirmelisiniz. Artık güvenli olarak NFS kullanmaya hazırsınız.

## 21 NFS kurulumu

NFS 3 kurulumunun ilk adımı bir NFS sunucusu kurmaktır. Sunucu olarak seçilen makinenin çekirdeğinde NFS sunucu desteğini aktif hale getirmek gerekir. NFS 3 için 2.2.18 ve üzeri bir çekirdeğe (2.4 ve üzeri önerilir) sahip olmalıyız. Eğer çekirdeğinizi kendiniz derliyorsanız, /usr/src/linux dizinine gidip

```
# make menuconfig
```

komutunu çalıştırmalıyız. Burada "File systems section" seçeneğini, ardından da "Network File Systems" seçeneğini seçtikten sonra, aşağıdaki tercihleri aktif hale getirin:

```
<*> NFS file system support
[*]   Provide NFSv3 client support
<*> NFS server support
[*]   Provide NFSv3 server support
```

## 21.1 /etc/exports üzerindeki düzenlemeler

Çekirdeğinizi derleyip kurduktan sonra, bilgisayarınızı yeniden başlattığınızda sisteminiz NFS 3 sunucu ve istemci özelliklerine sahip olacaktır. Çekirdekte NFS desteğini sağladıktan sonra, sıra /etc/exports dosyasını düzenlemeye geldi. /etc/exports dosyası üzerindeki düzenlemeler, hangi dosya sistemlerinin ihraç edilebileceğini, bu dosya sistemlerine kimlerin ulaşabileceğini ve bu kullanıcıların okuma/yazma veya sadece okuma haklarıyla ulaşabileceklerini belirler. Ayrıca NFS'in çalışmasıyla ilgili bazı diğer ayarları da buradan kontrol edebiliriz.

/etc/exports dosyasını incelemeden önce, varolan bir problemten bahsedelim. Linux çekirdeğindeki NFS sistemleri, her bir dosya sisteminde tek bir dizinin ihraç edilmesine izin verir. Yani aynı ext3 dosya sisteminde (örneğin /dev/hda6 da) bulunan /usr ve /home dizinlerini birlikte ihraç edilmeleri için /etc/exports dosyasına yazamazsınız. Eğer her iki klasörü de /etc/exports dosyasına eklerseniz, bu dosya tekrar okunduğunda (örneğin NFS sunucu çalışırken exportfs -ra yazarsanız) aşağıdaki hatayı görürsünüz:

```
sidekick:/home: Invalid argument (Geçersiz değişken)
```

## 21.2 İhraç kısıtlamaları ile çalışma

Şimdi bu problemi nasıl çözeceğimize, yani hem /usr hemde /home klasörlerini aynı anda nasıl ihraç edeceğimize bakalım. Her iki dizinde aynı dosya sistemindeyse birlikte ihraç edemeyeceğimizi söyledik. O zaman / (kök) dizinini ihraç ederiz. Böylece NFS istemcileriniz artık hem /home hemde /usr dizinlerini NFS üzerinden bağlayabilirler. Ayrıca NFS sunucusundaki /etc/exports dosyası da her bir dosya sistemi için tek bir dizin ihraç edildiği için hata vermez. Linux NFS'in bu kurulumunu da gördükten sonra /etc/exports dosyasına bakabiliriz.

/etc/exports dosyası

/etc/exports dosyasının yapısını anlamamanın en iyi yolu bir örneğini incelemektir. Bizim NFS sunucusundaki /etc/exports dosyasının içeriği şöyle:

```
# /etc/exports: NFS file systems being exported.
/ 192.168.1.9(rw,no_root_squash)
/mnt/backup 192.168.1.9(rw,no_root_squash)
See exports(5).
```

Görüldüğü gibi dosyanın ilk satırı bir açıklama. İkinci satırda kök ("/") dizinin ihraç edildiği belirtiliyor. Şunu unutmayın ki, kök dizin ihraç edildikten sonra aynı dosya sisteminde ki bir başka dizin ihraç edilemez. Örneğin NFS sunucunun /mnt/cdrom klasörüne bağlanmış bir CD sürücü varsa, bu sürücü /etc/exports'ta ayrıca bağlanmadığı sürece ulaşamazdır. Bunun için dosyanın üçüncü satırına bakalım. Burada kök dizinden başka bir dosya sistemine ait olan ve sistemin yedeğinin tutulduğu /mnt/-backup dizini ihraç ediliyor. Her satırdaki "192.168.1.9(rw,no\_root\_squash)" bilgisi de, nfsd'ye, sadece 192.168.1.9 IP adresli istemcilerin, ihraç edilen dizinlere ulaşabileceğini ve bu istemcilerin okuyabildikleri gibi yazma haklarına da sahip olacaklarını söyler. Ayrıca bir NFS istemcisinde super kullanıcı hesabıyla dosya sistemine root haklarıyla ulaşmasına izin veriliyor.

## 21.3 Bir başka /etc/exports örneği

Buradaki /etc/exports dosyası da bir önceki örnekteki dosya sistemini ihraç etmek için oluşturulmuş. Ancak bir fark, buradaki sunucu, LAN üzerindeki (192.168.1.1 den 192.168.1.254 e kadar) tüm makinelere hizmet veriyor.



```
# /etc/exports: NFS file systems being exported.
/ 192.168.1.1/24(rw,no_root_squash)
/mnt/backup 192.168.1.1/24(rw,no_root_squash)
See exports(5).
```

Bu `/etc/exports` örneğinde, `/24` maskesi kullanılarak, verilen IP adresindeki son 8 bit maskeleniyor. Burada IP adresi ifadesiyle `"` arasında boşluk olmamasına dikkat edin. Bu araya koyacağınız bir boşluk karakteri dosyanın yanlış yorumlanmasına sebep olur. `"rw"` ve `"no_root_squash"` dışında da bazı izin tercihleri vardır. Bunun için `"man exports"` yazarak tüm tercihleri görebilirsiniz.

## 21.4 NFS3 sunucusunu çalıştırmak

`/etc/exports` dosyası düzenlendikten sonra, NFS sunucunuzu çalıştırmaya hazırsınız demektir. Çoğu dağıtımda `nfs` çalıştırma komut dosyaları hazır geliyor. Yazmanız gereken:

```
/etc/init.d/nfs start
```

veya

```
/etc/rc.d/init.d/nfs start
```

NFS çalıştırıldıktan sonra, `rpcinfo` yazdığınızda aşağıdakine benzer bir çıktı almanız gerekiyor.

```
# rpcinfo -p
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100024 1 udp 32768 status
100024 1 tcp 32768 status
391002 2 tcp 32769 sgi_fam
100011 1 udp 792 rquotad
100011 2 udp 792 rquotad
100011 1 tcp 795 rquotad
100011 2 tcp 795 rquotad
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100021 1 udp 32770 nlockmgr
100021 3 udp 32770 nlockmgr
100021 4 udp 32770 nlockmgr
100005 1 udp 32771 mountd
100005 1 tcp 32770 mountd
100005 2 udp 32771 mountd
100005 2 tcp 32770 mountd
100005 3 udp 32771 mountd
100005 3 tcp 32770 mountd
100009 1 udp 821 yppasswdd
100004 2 udp 734 ypserv
100004 1 udp 734 ypserv
100004 2 tcp 737 ypserv
100004 1 tcp 737 ypserv
```

## 21.5 İhraç tercihlerini değiştirme

Eğer NFS servisi çalışırken bazı değişiklikler yaparsanız, bu değişikliklerin aktif olması için

```
exportfs -ra
```

yazmanız gerekir. Böylece NFS sunucunuz yeniden düzenlenmiş ve çalışır hale gelmiştir. Artık NFS istemcilerini yapılandırarak, ihraç ettiğiniz dosya sistemlerini bağlamalarını sağlayabiliriz.

## 21.6 NFS istemcilerini yapılandırma

NFS 3 istemcileri için çekirdek yapılandırılması NFS sunucu ile çok benzer. Tek fark, sadece aşağıdaki tercihlerin seçilmiş olması yeterlidir.

```
<*> NFS file system support [*] Provide NFSv3 client support
```

## 21.7 NFS istemci servislerinin çalıştırılması

NFS istemci servislerini başlatmak için, sunucudaki gibi bazı hazır komut dosyalarını, "nfslock" veya "nfsmount", çalıştırmak yeterlidir. Bu komutlar NFS istemcinin tek ihtiyacı olan rpc.statd servisini başlatacak. Bu servis, dosya kilitleme sisteminin doğru çalışmasını sağlar. Tüm istemci servisleri kurulduktan sonra, rpcinfo yazarsanız, aşağıdakine benzer bir çıktı alırsınız:

```
# rpcinfo
  program vers proto port
    100000    2  tcp   111 portmapper
    100000    2  udp   111 portmapper
    100024    1  udp  32768 status
    100024    1  tcp  32768 status
```

Bu kontrolü uzaktaki bir sistemden yapmak isterseniz, komutu rpcinfo -p makine\_adi parametreleriyle çalıştırmak gerekir:

```
# rpcinfo -p sidekick
  program vers proto port
    100000    2  tcp   111 portmapper
    100000    2  udp   111 portmapper
    100024    1  udp  32768 status
    100024    1  tcp  32768 status
```

## 21.8 İhraç edilen NFS dosya sistemlerini bağlama

Sunucu ve istemciler kurulduktan sonra ve NFS sunucusu, istemcinizden gelen bağlantılara izin vermek üzere yapılandırıldıysa, bir sonraki aşamaya, ihraç edilen bir dosya sistemini istemci makineye bağlama işlemine geçebilirsiniz. Aşağıdaki örnekte, inventor makinesi NFS sunucusu, sidekick ise (IP adresi 192.168.1.9) NFS istemcisi olarak kurulmuş. Inverntor'ın /etc/exports dosyasında aşağıdaki satır yer alıyor.

```
/ 192.168.1.1/24(rw,no_root_squash)
```

Böylece, 192.168.1 ağındaki tüm makinelere bağlantı izni verilmiş. Şimdi sidekick makinesine root olarak girin ve aşağıdakileri yazın:

```
# mount inventor:/ /mnt/nfs
```

Inverntor makinesinin kök dizini, sidekick makinesinde /mnt/nfs dizinine bağlanmış oldu.

```
cd /mnt/nfs
```

yazarak, inventor makinesinin dizinlerinde gezebilir ve dosyalarına bakabilirsiniz. Şunu tekrar hatırlatalım ki, inventor makinesinin kök ("/") dizini başka bir dosya sisteminde olsaydı, /mnt/nfs/home dizini hiçbirşey içermeyecekti. Bu durumda, buradaki bilgiye ulaşmak için yeni bir bağlama işlemine (aynı zamanda inventor makinesinin /etc/exports dosyasında yeni bir satıra) ihtiyaç duyacaktık.

## 21.9 İhraç edilen dizinlerin içinde yapılan bağlama işlemleri

/etc/exports dosyasındaki / 192.168.1.1/24(rw,no\_root\_squash) satırının kök dizini içindeki dizinlere de bağlama işlemi uygulamamıza izin verdiğini unutmayalım.

Örneğin, inventor makinesinde kök ("/") dizin ile /usr dizinleri aynı dosya sistemindeyse ve siz / yerine /usr dizinini bağlamak isterseniz, o zaman:

```
# mount inventor:/usr /mnt/usr
```

yazarsınız. Böylece inventor'un /usr dizin ağacı, sizin önceden varolan /mnt/usr dizininize bağlanmış oldu. Bunun için /etc/exports dosyasında bir değişikliğe ihtiyaç olmadığına dikkat edin. Kök dizininin ihraç edilmesi, onun altındaki dizinlere bağlama işlemi yapılmasını özgür bırakmıştı.