İSTANBUL BİLGİ ÜNİVERSİTESİ
Bilgisayar Bilimleri Bölümü

Open Source Enterprise Centre GNU/Linux Sertifikasyon Programı

Linux 101 Ders Notları

Son Güncelleme: 9 Şubat 2007

İçindekiler

1 Linux, Knoppix ve bu eğitim hakkında

1.1 Neden Linux ?

Bu eğitime geldiğinize 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 kullanıcılı (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 kullanıcılı 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 sunucu 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 gönderek 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'ınız varsa Knoppix harddiskinizi kullanmaz bile. Dahası Knoppix ile yaptığınız 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 yaptığmız 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 seçebillir 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 tanıştıracak, 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 yansıtıyor 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 kullanıp 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 bulunduğunuz 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

Göreli yollar biraz karmaşık görülebilir. Burada sonuçta varılacak hedef dizinin görülmediği 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 dizininine götürecektir. root kullanıcılar icin home dizini /root ve diğer kullanıcılar için ise /home/kullanıcı_adı seklindedir. Peki ya acaba kendi home dizinimiz içerisinde bir dosya belirlemek istersek ne yapmak gerekir? Mesela bir dosya argümanını program komutuna parametre olarak gondermek isteyelim. Eğer dosya kendi home dizinimiz içerisinde yer alıyorsa, yazılabilecek komut aşağıdaki şekilde düşünülebilir:

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

Fakat yukarıdaki örmekte 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 anlamanın temeli inode mantığını anlamayı gerektirir. Örnek olarak, her dizinde gördüğünüz . .. bağlantılarını düşünelim. Gerçekte .. 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

Gördüğünüz gibi, /usr/local/bin/.. ile /usr/local'ın inode numaraları aynı! Açığa çıkan bu bilgiden varabileceğimiz 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 ust dizinleri yaratmasını sağlayabilirsiniz, aşağıda görüldüği 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 gördüğünüz 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 dizin girişleri ve inode'lar arasındaki ilişkiden bahsederken bağlantı terimini kullanmıştık. Aslında Linux altında iki tip bağlantı çeşidi vardır. Daha önce tartıştığımız bağlantılar katı bağlantılardı. 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 ilişkilenmiş katı bağlantılar olmasına rağmen bu doğrudur. "root" kullanıcısı bile olsanız, kendinize ait dizinlere ilişkilenmiş yeni katı bağlantılar yaratamazsınız. Katı bağlantıların ikinci kısıtlaması da dosya sistemleri arasında geçiş yapılamamasıdır. Bunun anlamı, / ve /usr dizinleri farklı dosya sistemlerinda 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 değilde ismi ile ilişkilenirler. 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 bağıl 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 bağıl 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. Bağıl yol ve tam yolların her ikisinin de çok faydalı olduğu durumlar vardır. Sonuçta kendi uygulamanıza en uygun olan tipi seçmeniz gerekmektedir. Genellikle bağıl 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 beliritim /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 ayrım 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)
- "ldȯo" (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 . metakarakteri işlevsel olarak komut satırındaki dosya ismi açılımı mekanizmasındaki ? metakarakterine benzemektedir.

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 benzemektedir. 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 $\hat{k}$arakteri 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 $\hat{[]}$ 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, especialy
# 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 içerisinden 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; paylaşılabilir - paylaşılamaz, ve değişken - statik. Paylaşılabilir bilgi ana makinalar (hosts) tarafından paylaşılabilir 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 paylaşılabilir ("paylaşılabilir") ya da bir CD-ROM içinden bağlanabilir ("statik"). Bir çok Linux kurulumunda /usr dizinin paylaşımı kullanılmasa 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 arasıra yardıma 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 içerisinden 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/xtel/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 alışıksanız -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/krdb
/usr/bin/pydoc2.1
/usr/bin/pydoc2.2
/usr/bin/rgrep
/usr/bin/texi2pdf
```

7.16 Bulunan dosyaların işlenmesi

Pekçok değişik kritere 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 {} gördüğü yere de bulmuş olduğu dosya ismini koyarak ilgili komutu çalıştırır. Bunun anlamanın 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/krdb
-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 zorundadur. 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/lsmod
/sbin/lspci
/sbin/lspnp
/usr/bin/lsattr
/usr/bin/lsdev
/usr/bin/lsof
/usr/bin/lspgpot
/usr/bin/lss16toppm
/usr/sbin/lsof
/usr/sbin/lsusb
```

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 uptadedb 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 izninine 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 Ĉ şeklinde gösterilir):

```
^C
$
```

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

```
$ xeyes -center red
^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 durumdalar. 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
^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 verirseniz 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, durdumak 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 kullandığınızda 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 gelemez:

```
$ 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) başlattığınız 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 çıktığınızda 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 [bdflush]
    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 görmenizi ve sisteminizdeki birçok sürecin arasındaki ilişkiyi algılamanızı 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 ttyp1    S    11:10   0:00 /bin/bash
knoppix   5054  0.5  0.8   3128 2096 ttyp0    S    11:30   0:00 /bin/bash
knoppix   5195  0.0  0.5   3508 1532 ttyp0    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 bdflush
```

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 nezaket 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 tanıması 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 komutun 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. Yukardaki örnekte echo komutu "merhaba dunyali" cumlesini sonunda bir satır sonu ile birlikte basar. Normalde bu çıktının terminalde gözükmesi 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 önekler 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 Nomalde 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 barındımasıdır. 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 seçeğini 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 çıktıdaki 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 olasanız 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 fonksiyonalitesi 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 bölümününde arama yapabilirsiniz. Bunun bir altdizi aramasi 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 icin makewhatis komutunu yeniden çalıştırmak gerektiğini unutmayın.

10.9 GNU bilgisi

Kılavuz sayfalarının bir eksiği, yardımlı metinleri (hypertext) desteklememeleri. Bu durumda, bir dosyadan diğerine kolaylıkla atlamak mümkün olmuyor. Bunu farkeden 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 sekilde 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 ileçıkabilirsiniz. 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 sayfasini 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, farkli 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 dizin listesi göreceksiniz. Bu dokümanlari incelemek bazı cevherleri ortaya çıkarmanızı da sağlayabilir. Man sayfalarında veya info sayfalarında bulamayacağınız detaylara, ancak bazı eğitmenlerde 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 Dökümantasyon Projesi

Sistemde bulunan dokümanlara ek olarak, internet üzerinde de Linux konusunda çok iyi kaynaklara ulaşabilirsiniz. Linux Dokümantasyon Projesi'de, 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 Dökümantasyonu 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 yanısıra, 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 incelerseniz, 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

Internet 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. Populer 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 ekledi. 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 durumdalar. 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, herbir 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 biraraya 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 calışan kullanıcının kim olduğunu öğrenme ve grup üyelikleri hakkında bilgi alma yöntemini inceleyelim. Eğer bilgisayarı açtıktan sonra su komutunu hiç çalıştırmadıysanız, bilgisayarı acarken girdiginiz kullanici 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 edeceğiniz 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 üçlüden bir ya da ikisi için izin belirlenmesi daha uygundur. Bunu yapmak için bu üçlüden 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 üçlüden 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 herbir 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 düşünürsek 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 değilde, asıl sahibi olan kullanıcının adıyla çalıştırılıyormuş 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 çalıştırılıyormuş 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 altdizinlere 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 ayarlanmışsa, bu dizinin grup bilgileri, içinde yaratılan herhangi bir dosya sistemi nesnesine de yansıyacaktır. Bu özellik, aynı gruba üye olan birkaç kullanıcının kullanacağı bir dizin ağacı oluşturulacağında, 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 çalıştırabilirlik ayarları, o dosyanın yaratıcısı olan kullanıcının umask değerine bağlıdır.

11.19 Dizinler ve silme işlemi

Linux dizinleri standart ayarları aslında 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 dizinleri 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 farkedeksiniz. 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 atlarsınız. 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 ndd 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 çalıştığınız 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 çalıştığınız metne kopyalamak icin:

```
: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 herbir 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 tanıtıldığını 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, /ets/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 yeralır. /bin/false programı bir hata mesajı verip çıkmaktan başka hiçbirşey 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 dosyayla 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$Us10Oa1moVCgbwXsx.a.E.:12167:0:99999:7:::
```

Yine herbir satır ayrı bir kullanıcı hesabıyla ilgili bilgileri içeriyor ve herbir 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 gunden, 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. ikinci 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 kullancı 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 değilde, o grubun numarasını vermemiz gerekirdi. Yeni kullanıcımız 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 değilde normal bir kullanıcı yaratıyor olsaydık, 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$Us10Oa1moVCgbwXsx.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$Us10Oa1moVCgbwXsx.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 yapacaksanız, bu dosyaya yeni bir grupla 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 dikakte 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

Kullandığınız 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 ttyp0    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 profilerc yes yes sourceignore yes no ignoresource no yes sourceignore no no ignore-ignore

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 yerleşiğine) 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 durumdalar 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 yerleşiğiyle ilgili tüm tercihleri bash kılavuz sayfasında bulabilirsiniz.

14.8 "set" ile değişkenlerin ayarlanması

set yerleşiği 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 yerleşiği 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 anlamanın bir yolu, set yerleşiğini, halihazırdaki tüm değişkenleri listelemesei için parametresiz olarak çağırmaktır:

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

set yerleşiğini parametresiz olarak çağırmakla export yerleşiğini kullanmak hemen hemen aynıdır. Aradaki temel fark, set yerleşiği 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. Aslında 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 menuyu 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 sürücünüzün IDE ya da SCSI olması ile ilgilenmeksizin disk ile iletişiminde 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 yeralmaktadır. 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 bülümün 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 menu sağlamakta 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 hda5ten hda9'a kadar mantıksal sürücüleri 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 kullanımı konusunda tecrübesizseniz size, içerisinde önemli sayılabilecek bilgi içermeyen ayrı bir disk üzerinde bu programı kullanarak kavramaya çalışmanızı öneririz. Bir defa bölümleri yaratıp diske yazdığınızda, 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 yeralır.

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 değiskeninin gösteridiği 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 sitemini 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 değiştirelemez, 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 komutnu 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 sitemleri 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 dizinler 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/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
```

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 edebileceğiniz 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
```

unmount 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 sayısıdır 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 onaramadığını görürüz. Böyle durumlarda sistemi tek kullanıcı çalışır hale getirerek fsck'yı el yordamıya 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 değişikleri disk üzerinde günlük dosyasına kaydeder(log file). Sistemde meydana gelen bir arıza durumunda dosya sistemi sürücüsü bu günlük dosyasını araştırır. Bu dosya son değişiklerin 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 camiasinda journaling dosya sistemlerinin popülaritesi 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 bütülüğü 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 gerçeklemesi (implementation) sistem kilitlenmesi durumunda son değişiklerin 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. ResiserFS'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 dökümanını 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 sektorünü 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 çağırmaktadır. 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 çalıştırırlar. 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) dır. Bu ikisi içerisinde LILO daha eski ve daha çok bilinen boot loader dır. Sistemde LILO'nun hazır olduğu zaman bu durum açılışta "LILO boot:" şeklinde bildirilir. Şunu belirtmek gerekir ki, genelde sistem duraklamadan konfigurasyon ayarları yaptığından, 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 söylemeliyiz. /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 değişikleri geçerli hale getirir.

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

```
# lilo -v
LILO 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 menusunde 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 farketseniz 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 paramereler-den 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, telinint 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 de. 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 (genelde4 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, telinint 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 kulanıcılar 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 de. Bu gördüğünüz mesaj sistemde o anda çalışan tüm kullanıcıların terminallerinde belirecektir . Böylece bu kullanıclar 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 (genelde4 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 şeklide 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 getirldiğinde 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 oldugu 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 kullancının kotasını görüntülemek bizi canımızdan bezdirebilir. repquota komutu dosya sitemindeki 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 arzetmektedir.

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 konuşlandıracak olsaydık 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/qrunner)
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 iceleyebiliriz:

```
# 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 ilişkli 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/sylog.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 belirtilen 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: Aksion 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 anlamanıza 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 yaptığınız 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 değillerse 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 deamon'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 incelereken rahat kavramanıza yardımcı olabilir.

Öncelikle şunu belirtmek gerekir ki; syslog arka plan süreci (deamon) 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 kylogd 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.