

Önemli LINUX Kavram ve Komutları

05

- Önemli LINUX Kavramları
 - Standart Giriş ve Standart Çıkış
 - Çekirdek: Kernel
 - Dosya Sistemleri
 - Süreçler
 - Link Kavramı ve ln Komutu
 - “Pipe” Kavramı
- Biraz Nefes Alalım
 - Kullanışlı LINUX Komutları

Önemli LINUX Kavramları

Standart Giriş ve Standart Çıkış

“Standart Giriş” ve “Standart Çıkış”, LINUX işletim sisteminin çok önemli iki kavramıdır.

LINUX konsol komutlarının yüzde doksanı, işlevlerini standart giriş biriminden okuyacakları veriler üzerinde yerine getirip, varsa sonuçlarını standart çıkış birimine gönderir. Bir başka deyişle, LINUX komutlarının yüzde doksanı, görevlerini klavyeden okuyacakları veriler üzerinde yerine getirip, varsa sonuçlarını ekrana gönderir.

Standart giriş birimine LINUX terminolojisinde “**STDIN**”, standart çıkış birimine de “**STDOUT**” denir.

Örneğin **wc** komutu (*word count*) standart girişten okuyacağı verilerdeki karakter, sözcük ve satırları sayar; bu sonuçları da standart çıkış birimine görüntüler.

```
caayfer@notebook.lojman.bilkent.edu.tr: /home/caayfer - Shell - Konsolle <4>
[caayfer@notebook caayfer]$ wc
Birçok insan için bilgisayar apandisit gibidir; sorun çıkarıncaya
kadar varlığını bile farketmezler.
Edward A. Feigenbaum, 1983
← 6      16     131
[caayfer@notebook caayfer]$
```

wc komutunun çıktısı olan satırda 3 sayı göreceksiniz. Birincisi, programa girdi olarak verilen dosyadaki satırların sayısı; ikincisi sözcüklerin sayısı; sonuncusu da karakterlerin sayısıdır. **wc** komutunun biraz daha ayrıntılı olarak anlatımını birkaç sayfa ileride bulacaksınız.



Bir programa standart girişten girilen verilerin sonunu belirtmek için Ctrl-D tuşuna basılır. Bir başka deyişle klavyenin "dosya sonu" Ctrl-D ile belirtilir. Bu nedenle yukardaki örneği denerken **wc** programına karakterlerini, sözcüklerini ve satırlarını sayması için girdiğiniz satırlar bitince imleç satır başındayken Ctrl-D tuşuna bir kez basmalısınız. Sadece bir kez basmanız önemlidir. İki kez basarsanız kabuk programınıza da standart girişin sonuna geldiğinizi belirtmiş olursunuz ki bu kabuk programınızın işini bitirdiğini zannedip sona ermesine ve telnet penceresinin kapanmasına yol açar. Eğer yanlışlıkla basılan Ctrl-D tuşunun **bash** kabuğunu sonlandırmamasını; onun yerine açıkça **exit** komutunun kullanılması gerekmesini istiyorsanız örneğin

```
export IGNOREEOF=8
```

gibi bir komutla **bash** kabuğunun ancak peşpeşe sekiz tane Ctrl-D basılması durumunda kendini öldürmesini belirtebilirsiniz.

Elbette bu komutu kişisel dizininizdeki **.bashrc** dosyasına yerleştirerek sisteme her bağlanışınızda, bir başka deyişle sizin için **bash** kabuğunun her başlatılışında bu komutun otomatik olarak çalışmasını sağlayabilirsiniz. Eğer **IGNOREEOF** ortam değişkeninin tüm kullanıcılar için "8" değerini almasını istiyorsanız "**export IGNOREEOF=8**" komutunu **/etc/bashrc** dosyasına yerleştirebilirsiniz.

Giriş ve Çıkışı Yönlendirmek

Çalıştırıldığında ürettiği çıktıları standart çıktıya (ekrana) yazan bir komut veya uygulama programının çıktılarını saklamak isterseniz standart çıktıyı bir disk dosyasına yönlendirebilirsiniz. Bunun için programı çalıştıran komut satırının sonuna küçük bir ekleme yapmanız gerekecektir:

```
ls -al /tmp > dosya_adi
```

Yukardaki komutta “**ls -al /tmp**” komutu çalıştırılmadan önce standart çıktı, diskteki çalışma dizininde yer alacak **dosya_adi** isimli dosyaya yönlendirilecektir. Bu durumda **ls** komutunun üreteceği tüm çıktılar **dosya_adi** isimli dosyaya kaydedilecektir. Çalışma dizininde **dosya_adi** isimli bir dosya zaten varsa ve erişim yetkileri uygunsa **ls** programının çıktısı bu dosyadaki kayıtların üzerine kaydedilecektir, yani dosyanın eski içeriği kaybolacaktır.

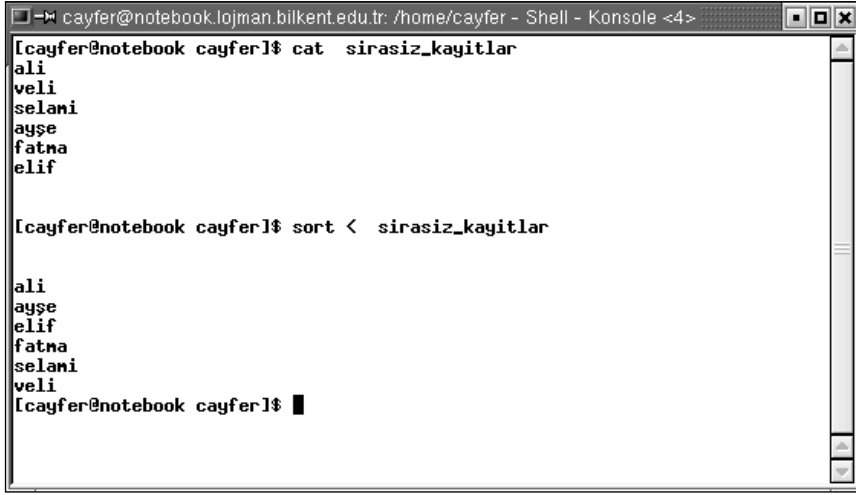
Bazı durumlarda, standart çıktının yönlendirileceği dosyada bulunan eski kayıtları bozmadan yenilerini bunların arkasına eklemek isteyebilirsiniz. Bu durumda yönlendirmeyi “>>” ile yapmanız yeterli olacaktır.

```
ls -al /tmp >> dosya_adi
```

Çalışmak için gereksinim duyduğu verileri standart girişten (klavyeden) okumak üzere yazılmış bir programın, söz konusu verileri diskteki bir dosyadan almasını sağlamak için standart giriş yönlendirmesi yapmalısınız. Örneğin,

```
sort < sirasiz_kayitlar
```

komutu, sıralanacak satırları diskteki “**sirasiz_kayitlar**” isimli dosyadan alacaktır. Verilerini standart girişten okuyup, çıktısını standart çıkış birimine gönderen programlara **filtre** programlar denir. **sort** komutu bu filtre programlara güzel bir örnek oluşturur çünkü program standart girişten gelen kayıtları sıralayıp standart çıkışa gönderir.



```
caayfer@notebook.loyman.bilkent.edu.tr: /home/caayfer - Shell - Konsolle <4>
[caayfer@notebook caayfer]$ cat sirasiz_kayitlar
ali
veli
selami
ayşe
fatma
elif

[caayfer@notebook caayfer]$ sort < sirasiz_kayitlar

ali
ayşe
elif
fatma
selami
veli
[caayfer@notebook caayfer]$ █
```

Giriş yönlendirme ve çıkış yönlendirmeyi birlikte kullanabilirsiniz. Örneğin **sirasiz** isimli dosyadaki satırları sıralayıp, sıralanmış satırları **sirali** isimli bir dosyaya kaydetmek için aşağıdaki komutlardan birini kullanabilirsiniz:

```
sort < /tmp/sirasiz > /tmp/sirali
sort > /tmp/sirali < /tmp/sirasiz
```

STDERR

UNIX programcılarının geleneksel yaklaşımlarından biri, yazdıkları programların hata mesajlarının standart çıktıda değil, “standart hata”da (yani STDERR’de) belirecek şekilde kod geliştirmeleridir. Aksi belirtilmedikçe STDERR, “ekran” ortamıdır; işte bu yüzden hata mesajlarını da ekranda görürsünüz.

STDOUT ve STDERR ayırımı, gerektiğinde hata mesajlarının ayrı bir ortama yönlendirilmesini sağlar. Örneğin normal bir kullanıcı olarak (yani “**root**” olmadan)

```
du -s /
```

komutunu verirseniz, ekranda

```
[cayfer@cayfer /]$ du -s /
du: cannot change to directory `bcc/mail': Permission denied
du: cannot change to directory `bcc/bcc2': Permission denied
du: cannot change to directory `etc/cups/ssl': Permission denied
du: `etc/cups/certs': Permission denied
du: cannot change to directory `etc/skel/tmp': Permission denied
du: cannot change to directory `etc/uucp': Permission denied
```

gibi hata mesajları alırsınız. Bu mesajlar aslında STDOUT'a değil, STDERR'e gönderilmektedir. Nitekim **du** komutunu, çıktısını **/tmp/du_raporu** diye bir dosyaya yönlendirmek üzere

```
du -s / > /tmp/du_raporu
```

şeklinde verseniz bile bu hata mesajları **du_raporu** dosyasına değil, gene ekrana görüntülenecektir. **/tmp/du_raporu** dosyasına yalnızca diskte kapladığı alan başarıyla hesaplanabilen dizinlere ilişkin satırlar yönlendirilecektir.

Böylece; özellikle geri planda çalışan işlerin hata mesajlarıyla terminal penceresini kirletmelerini önlemek ve kaybolmasın diye hata mesajlarını ayrı bir dosyada biriktirmek için STDERR'i bir dosyaya yönlendirmek mümkündür:

```
du -s / > /tmp/rapor 2> /tmp/hatalar &
```

Yukarıdaki komuttaki ">" karakteri STDOUT'u yönlendirmek istediğinizi; "2>" karakterleri ise STDERR'i yönlendirmek istediğinizi; en sondaki "&" ise programı geri planda çalıştırmak istediğinizi belirtmektedir.

Standart Giriş ve Standart Çıkış kavramlarının yanı sıra, LINUX işletim sisteminde, iyi kavranması gereken birkaç önemli kavram daha var. Bu kavramlar, ilk okuduğunuzda çok karışık ya da anlaşılmaz gelebilir; ama LINUX sistem yöneticisi olma yolunda ilerlemeyi düşünüyorsanız lütfen dikkatlice okuyunuz, gerek duyarsanız başka kaynaklara başvurunuz ama bu kavramları anlamadan geçmeyiniz.

Bu kavramlar:

- Çekirdek... İngilizce, daha doğrusu UNIX'çesiyle "kernel",
- Dosya Sistemleri; UNIX'çesiyle "file systems",
- Süreçler; UNIX'çesiyle "process",

Kim Korkar LINUX'tan?

- Baęlantılar; UNIX'çesiyle "links",
- "pipe" kavramı.

Çekirdek: Kernel

Modern işletim sistemleri, tüm işlevlerini yerine getirecek modülleriyle birlikte belleęe yüklenmezler. Bunun en önemli nedeni bellekten tasarruf edebilmektir. İşletim sistemi ne kadar az bellek kullanırsa uygulama programlarına o kadar fazla boş bellek kalacaktır. Bu nedenle modern işletim sistemleri, bellekte kalması kaçınılmaz ve "kernel" adı verilen "çekirdek" modüller ve bunun etrafında gerektiğçe belleęe yüklenen yan modüller topluluęu olarak geliştirilir.

LINUX çekirdeğinin temel görevleri bellek ve süreçleri denetlemek ve donanım birimlerini yönetmektir. Örneğın, çekirdek, bilgisayara takılı olan IDE disk sürücülerin tüm teknik ayrıntılarının farkındadır. Dönme hızından yazıcı kafanın nasıl hareket ettirileceğine kadar ince ayrıntıları bilir. Benzeri şekilde ses arabirimlerini, Ethernet arabirimlerini tanıyıp bunları denetleyebilen yazılımlar birer çekirdek modülüdür. UNIX ve dolayısıyla LINUX çekirdeklerinin en önemli özellięi, tanıdıkları bu çevre birimlerini dięer işletim sistemi modüllerine ve uygulama programlarına birer "özel dosyaymış" gibi göstermeleridir. UNIX'in 30 yılı aşkın bir süredir bilgisayar dünyasında başarıyla yer alabilmesinin en önemli nedeni çekirdeğın bu özelliğidir.

Çekirdek için IDE elektronik arabirim standardında üretilmiş, 7200 rpm hızında dönen, 6 plakası ve 12 okuma-yazma kafası olan bir disk, çekirdek dışında kalan modüller için `/dev` dizini altında yer alan `hda` isimli bir özel dosyadır (*node*). Benzer şekilde bilgisayarın birinci disket sürücüsü `/dev/fd0`, birinci Ethernet kartı `/dev/eth0`'dır.

Sistemin açılışı sırasında ilk yapılan işlerden biri çekirdeğın belleęe yüklenmesidir. Çekirdek belleęe yüklenince sistemin denetimini üzerine alacak, açılışı tamamlamak üzere gereken modülleri kendisi yükleyecek; işi biten ya da bekleyebilecek modülleri bellekten atacaktır.

Bazı işletim sistemi işlevlerini yerine getiren yazılım modülleri çekirdekte yer almalıdır. Örneğın, belleğın yetmemesi durumunda disk takas alanlarının bellek gibi kullanılmasını sağlayan modüller her zaman gerçek bellekte (RAM) bulunmalıdır. Eđer bu modüller bellek yetmiyor diye diske atılırsa

bir daha onları gerçek belleğe geri almak mümkün olmaz. Benzeri şekilde diskin nasıl kullanıldığını bilen modül, bellekte yer açmak için diske atılırsa, geri yüklenmesi olanaksız olacaktır.

İşletim sisteminin bazı modülleri de performans açısından her zaman bellekte yer almalıdır. Sistemde yüksek öncelikte çalışan ve hiçbir zaman sanal bellek işlemlerine tabi tutulmayan süreçler genellikle çekirdeğin bir parçası olacak şekilde geliştirilmiştir. Örneğin disk kotası denetimleri, IP paket filtreleme işleri, IP paket yönlendirme işleri olabildiğince yüksek performans gerektiren işler oldukları için çekirdekte yer alan programlarla yapılırlar.

Ancak, her işi çekirdekte halletmek de olası değildir. Başta LINUX olmak üzere, modern işletim sistemleri, tüm modülleriyle birlikte belleğe yerleştirilemeyecek kadar büyüktür. Modern LINUX sürümleri bu sorunu “Loadable Kernel Modules”; yani “yüklenbilir çekirdek modülleri” kavramıyla çözmüştür.

Kendi çekirdeğinizi oluşturmanız için şimdilik bir neden görememekle birlikte LINUX'un kaynak kodlarını yeniden derleyerek kendinize özel bir çekirdek oluşturmanızın mümkün olduğunu belirtmeden geçemeyeceğiz.

Dosya Sistemleri

Microsoft'un MS-DOS/Windows işletim sistemlerinde dosya ve dizinlerin yerini belirtirken A:, C: gibi sürücü isimleri kullanılması tercih edilmiştir. Bir başka deyişle, kullanıcılar kullandıkları bilgisayara bağlı disklerin sayısını ve isimlerini bilmek ve ilgilendikleri dizin ya da dosyaların bu fiziksel ortamlardan hangisinde bulunduğunu bilmek ve gerektiğinde belirtmek zordur. LINUX'ta durum biraz, hatta oldukça farklıdır.

Şimdilik, LINUX bilgisayarınızın tek başına (bir bilgisayar ağına bağlı olmaksızın) çalışan bir bilgisayar olduğunu varsayalım. Bilgisayarınızın iki disk, bir disket, bir de CD-ROM sürücüsü olsun.

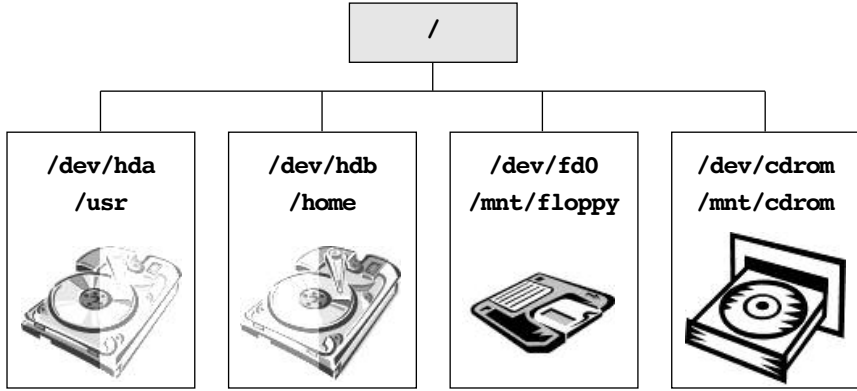
Disklerinizden ilki en az 2 parçaya (partition) bölünmüştür. (Bir bölüm LINUX işletim sistemi için, ikincisi de takas alanı için; hatırladınız mı?) İkinci diskiniz ise büyük olasılıkla tek parçadır.

Kim Korkar LINUX'tan?

Yukarıdaki varsayımlarımıza göre bilgisayarınızın beş diski varmış gibi düşünebilirsiniz. (İki parçaya ayrılmış birinci disk ve tek parça olan ikinci disk, disket sürücü ve CD-ROM sürücü.) MS-DOS ya da Windows kullanıyor olsaydınız, bu disklere A:, C:, D:, E: ve F: isimleriyle erişirdiniz.

Şimdi sıkı durun: LINUX kullanıcılarının, disklerin ne şekilde ayrılmış olduğundan, hatta bilgisayarda kaç disk sürücüsü bulunduğu haberi olması bile gerekmemektedir. LINUX'ta tüm diskler ve disk bölümleri (*partition*), disket sürücüler ve CD-ROM sürücüler, **/dev** dizinin altında birer **alt dizin** olarak yer alır.

Şematik olarak göstermek gerekirse:



Her disk ve disk parçası üzerinde diğerlerinden bağımsız bir dosya sistemi (*file system*) bulunmalıdır. Dosya sistemi, doğrudan erişimli bir veri saklama biriminde (disk, disket, CD-ROM gibi) dolu ve boş alanların yönetimini, dizin ve dosyaların yaratılmasını, silinmesini ve en önemlisi bunlara hızlı erişimi sağlayan bir veri yapısı bulundurur. Bu veri yapıları disklere formatlama (bir türlü “biçemleme” demeye alışamadık; zorla değil ya...) sırasında kaydedilir. Özet olarak, dosyaları oluşturan disk alanı bloklarının diskin fiziksel olarak nerelerinde yer aldığı, bu blokların kime ait olduklarının ve erişim haklarının saklandığı veri yapılarına “dosya sistemi” (*file system*) denir.

LINUX'ta disklerinizde kullanabileceğiniz birden fazla dosya sistemi seçeneği vardır. Biz, ext3 adı verilen sistemi kullanmanızı öneririz.

LINUX'ta dosya sistemleri, diskler (daha doğrusu disk bölümleri) üzerinde, formatlama işleminden sonra **mke2fs**, **mkreiserfs** gibi komutlarla yaratılır. (Merak etmeyin; bu işi sistem kurulumu sırasında farketmeden yaptınız bile.) Hatırlarsanız LINUX'unuzu kurarken "root" (/) dizininin hangi disk bölümüne bağlanacağını belirtmişsiniz. (/dev/hda6 gibi) İşte hiyerarşik LINUX dosya yapısının en tepe noktası bu "root" dizinidir. Sisteminizdeki tüm disk bölümleri, diskler, disket sürücüler, CD-ROM sürücüler, hatta başka bilgisayarlar üzerinde erişebileceğiniz disk/dizinler hep bu "root" dizinin altında "alt dizinler" olarak görünür.

Örneğin, /home dizini (kullanıcıların kişisel dizinlerinin yer aldığı dizin) bilgisayarınızın ikinci diski üzerinde olabileceği gibi / disk bölümünde yer alan gerçek bir alt dizin de olabilir. Özellikle merak edip bakmadıkça bir dizinin hangi diskte yer aldığını göremezsiniz. Yani **ls** komutu size bir dizinin hangi disk bölümünde ya da sürücüde yer aldığını söylemez.

Bilgisayarınızda disk bölümlerinin ve disket sürücü gibi çevre birimlerinin hangi dizinler altında erişilebilir olduğunu görmek için **mount** komutunu kullanabilirsiniz.

Örneğin, tek diskli ve bu diskte hem LINUX hem Windows barındıran bilgisayarda verilen **mount** komutu aşağıdaki gibi bir rapor üretir:

```

cayfer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsol <4>
[cayfer@notebook cayfer]$ mount
/dev/hda6 on / type ext3 (rw,noatime)
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,node=0620)
none on /dev/shm type tmpfs (rw)
/nnt/cdrom on /nnt/cdrom type supernount (ro,dev=/dev/hdc,fs=iso9660,--,iocharset=iso8859-1)
/nnt/floppy on /nnt/floppy type supernount (rw,sync,dev=/dev/fd0,fs=vfat,--,iocharset=iso8859-1,unask=0,codepage=850)
/dev/hda1 on /nnt/windows type vfat (rw,iocharset=iso8859-1,unask=0,codepage=850)
[cayfer@notebook cayfer]$ █

```

Kim Korkar LINUX'tan?

Aynı komutu daha karmaşık disk yapısı olan, örneğin birden fazla diski olan ve bu diskleri de ayrıca **bölümlendirilmiş** olan bir bilgisayarda verirseniz alacağınız rapor aşağıdakine benzer şekilde olacaktır:

```
caayfer@caayfer.bilkent.edu.tr: /home/caayfer - Shell - Konsole
[caayfer@caayfer caayfer]# mount
/dev/hda1 on / type ext3 (rw)
none on /proc type proc (rw)
none on /proc/bus/usb type usbdevfs (rw)
none on /dev type devfs (rw)
/dev/hdc2 on /depo type ext3 (rw)
none on /dev/pts type devpts (rw,node=0620)
/dev/hda8 on /home type ext3 (rw)
none on /mnt/floppy type supernount (rw, sync, dev=/dev/fd0, fs=auto, --, iocharset=iso8859-1, codepage=850, unask=0)
/dev/hda5 on /var type ext3 (rw)
/dev/hda7 on /var/spool/nail type ext3 (rw)
none on /mnt/cdrom2 type supernount (ro, dev=/dev/hdd, fs=auto, --, iocharset=iso8859-1, codepage=850, unask=0)
none on /mnt/cdrom type supernount (ro, noexec, nosuid, nodev, dev=/dev/scd0, fs=iso9660, --, iocharset=iso8859-1, codepage=850, unask=0)
[caayfer@caayfer caayfer]#
```

Evet, biliyoruz! Feci bir görüntü ama zamanla alışılıyor. LINUX sistem yöneticisi olmak kolay değil. Şaka bir tarafa; bu işlerin çoğunu KDE veya GNOME altında çalıştırabileceğiniz yönetim araçlarıyla da yapabilirsiniz. Ancak LINUX'u konsol komutlarıyla kullanabilecek şekilde öğrenmek sizin için çok daha yararlı olacaktır. LINUX dışında UNIX makinelerin başına geçtiğinizde de iş yapabilmemiz için KDE'siz, GNOME'suz yaşamayı öğrenmelisiniz.

Yararlı bilgiler içeren satırları griye boyadık. Diğer satırlar yararsız değil elbette ama şimdilik o satırları anlamaya çalışmak için erken.

Bu listeyi dikkatlice incelediğinizde, bilgisayarda sadece iki fiziksel disk bulunduğunu (sadece **/dev/hda** ve **/dev/hdc** serisi disklerin adı geçiyor) ve bu disklerin birinci IDE kanalının ilk diski (Primary master, **hda**) ve ikinci IDE kanalının da gene ilk diski (Secondary Master, **hdc**) olduğunu göreceksiniz. Bu arada şunu da belirtmek gerekir: Bu bilgisayarda başka diskler de takılı olabilirdi; örneğin **/dev/hdd** diye bir disk de olabilirdi.

```
/dev/hda1 on / type ext3 (rw)
```

satırını, bilgisayarın birinci IDE kanalının ilk diskinin (**hda**) bir numaralı bölümünün (**hda1**) bu sistemin root dizini olarak kullanıldığını gösteriyor. Ayrıca bu disk bölümünün okunabilir ve yazılabilir (**rw**) durumda olduğunu ve **ext3** dosya sistemi formatında yaratıldığını gösteriyor.

```
/dev/hdc2 on /depo type ext3 (rw)
```

satırını, bilgisayarın ikinci IDE kanalının ilk diskinin (**hdc**) iki numaralı bölümünün (**hdc2**) bu sistemin “/” dizini altındaki **depo** isimli bir dizine bağlandığını gösteriyor. Yani bu diske bakmak isteyenler **/depo** dizinine bakmalılar. Ayrıca bu disk bölümü üzerindeki dosya sisteminin **ext3** olduğu ve şu anda okunabilir/yazılabilir durumda olduğunu gösteriyor. Tabii yetkisi olanlar için...

```
none on /mnt/cdrom type supermount (rw,dev=/dev/cdrom,fs=iso9660)
```

satırını, bilgisayarın CD sürücüsünün (**/dev/cdrom**) sistemin **/mnt** dizinin altında **cdrom** isimli bir dizine (**/mnt/cdrom**) bağlandığını gösteriyor.

En baştaki “**none**”, CD sürücüde şu anda bir CD bulunmadığını belli ediyor. Ayrıca bu dizine **iso9660** dosya sistemine (standart CD dosya sistemidir) sahip CD’lerin takılabileceği ve bu CD’lerin oku/yaz kullanılabilceğini gösteriyor; yani **/mnt/cdrom**, yazılabilir CD’leri de destekliyor. “**supermount**” sözcüğü bu sürücüye bir CD takıldığında LINUX çekirdeğinin bu CD’yi belirtilen dizine otomatik olarak bağlayacağını gösteriyor. Sürücüye bir CD takıp içine bakmak isteyen birisi **/mnt/cdrom** dizinine bakmalıdır.

Benzer bir mantık disket sürücüler için de kullanılır.

```
none on /mnt/floppy type supermount (rw,dev=/dev/floppy,fs=vfat)
```

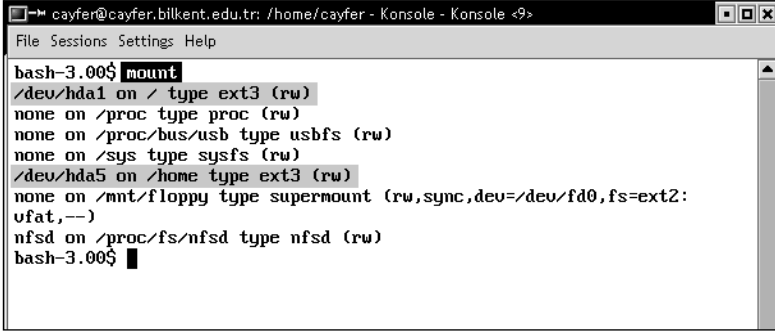
satırını, bilgisayarın disket sürücüsünün (**/dev/floppy**) sistemin **/mnt** dizini altındaki **floppy** isimli bir dizine (**/mnt/floppy**) bağlandığını gösteriyor. Yani disket takıp içine bakmak isteyen birisi **/mnt/floppy** dizinine bakmalıdır. Ayrıca bu dizine **vfat** dosya sistemine (standart MS-DOS disket dosya sistemidir) sahip disketlerin de takılabileceğini gösteriyor. “**supermount**” sözcüğü bu sürücüye bir disket takıldığında LINUX çekirdeğinin bu disketi belirtilen dizine otomatik olarak bağlamaya çalışacağını gösteriyor.

Kim Korkar LINUX'tan?

Gene yukardaki rapora göre **hda** diskinin sekiz numaralı bölümü (**hda8**) **/home** dizinine, beş numaralı bölümü **/var** dizinine; yedi numaralı bölümü de **/var/spool/mail** dizinine bağlanmıştır.

Üzerinde bir dosya sistemi olan bir disk birimine veya bölümüne okuma veya yazma amacıyla ulaşabilmeniz için, o dosya yapısının, “/” dosya yapınızda bir yerlerdeki bir alt dizine “mount edilmiş” (iliştirilmiş) olması gerekmektedir. (“/” dizini, bilgisayarın açılması sırasında otomatik olarak iliştirilmektedir. Eğer bu / dizini, bilgisayarın açılması aşamasında bağlanamazsa, o bilgisayar zaten açılmaz.)

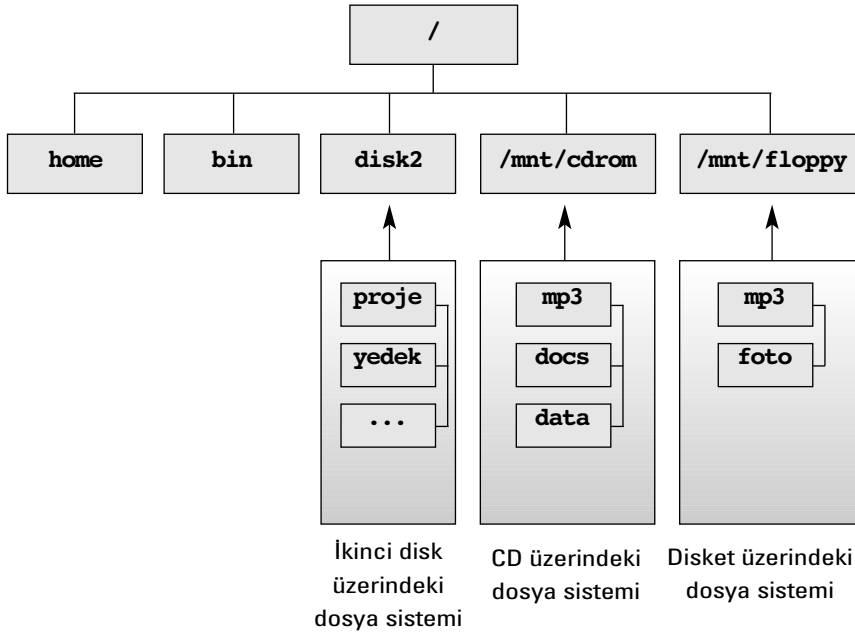
Bir bilgisayarda “mount edilmiş” disk ve disk bölümlerini daha kısa ve anlaşılır şekilde görebilmek için “df” komutunu da kullanabilirsiniz.



```
bash-3.00$ df
/dev/hda1 on / type ext3 (rw)
none on /proc type proc (rw)
none on /proc/bus/usb type usbfs (rw)
none on /sys type sysfs (rw)
/dev/hda5 on /home type ext3 (rw)
none on /mnt/floppy type supermount (rw, sync, dev=/dev/fd0, fs=ext2:
ufat, --)
nfsd on /proc/fs/nfsd type nfsd (rw)
bash-3.00$
```

Sistemin açılışı sırasında çeşitli dizinlere otomatik olarak iliştirilmesi istenen disk bölümleri **/etc/fstab** dosyasında belirtilir. Şimdilik bu dosya ile ilgili bir şey söylemek istemiyoruz; biraz erken. Sisteminize kurulum sırasında yerleştirilen **/etc/fstab** dosyası daha uzun süre işinizi görecektir. Bu dosyanın içeriğini merak ediyorsanız “**more /etc/fstab**” komutuyla görebilirsiniz.

LINUX'taki dosya yapılarını ters duran bir ağaca benzetirsek, dosya sistemlerini iliştirme işlemini, bir ağacı, bir başka ağacın dallarından birine iliştirmek (monte etmek) gibi düşünebilirsiniz.



LINUX işletim sisteminde **mount** komutu yalnızca bağlanmış diskleri listelemek için kullanılmaz. Sistem çalışırken disklerin bağlantısını çözmek, yeni disk ya da disk bölümlerini bağlamak için de kullanılır.

“mount” etmek derken disklerin bilgisayara fiziksel olarak takılıp çıkarılmasından söz etmiyoruz! O işi bilgisayarı kapatmadan yaparsanız başınız derde girer.



mount komutunu bu şekilde kullanabilmek için root kullanıcı yetkilerine sahip olmanız gerekecektir; yani eğer root kullanıcı değilseniz, **mount** komutunu yalnızca parametresiz olarak kullanmanıza izin verilecektir. **mount** komutu hakkında daha ayrıntılı bilgiyi sistem yönetimi ile ilgili bölümlerde bulabilirsiniz.



Disket sürücülerini ve CD-ROM sürücülerini de küçük birer disk sürücü olarak düşünmelisiniz; bu nedenle, bu sürücülerini kullanabilmeniz için önce “/” dizini altında bir yereye iliştilmeniz gerekir. Disketler ve CD’ler, takılıp çıkarılabilir birimler olduklarından, bilgisayar açılırken otomatik olarak iliştilmezler (mount edilmezler). Ancak LINUX çekirdeği bu sürücülere bir disket ya da CD takıldığında ve yeni takılan birimi kullanan bir komut verdiğinizde bu birimi otomatik olarak iliştilir (supermount). Bu otomatik iliştilme işleminin başarılı olabilmesi için üzerinde anlamlı bir dosya yapısı olan disket ya da CD takmalısınız. Yani formatsız bir disket ya da boş bir CD takarsanız doğal olarak otomatik iliştilme işlemi başarılı olamayacaktır. Supermount kavramının bir önemi de root yetkileri olmayan kullanıcılarının da sisteme disket ve CD takıp çıkarabilmelerini sağlamaktır. Biliyorsunuz, iliştilme amacıyla mount komutunu yalnızca root kullanabilir.

İş biten disket ve CD’ler (bazen de diskler) **umount** edilmelidir; yani, bu birimlere takılı medyalar üzerindeki dosya sistemlerinin, “/” dosya sistemiyle bağlantısı kesilmelidir. Bir dosya yapısının bağlı bulunduğu dizinle ilişkisini kesmek için:

```
umount /cdrom  
umount /dev/hdc1  
umount /var/spool/mail
```

gibi komutlar kullanılır.

Dikkat ederseniz hangi dosya sisteminin “umount” edileceğini, isterseniz sürücü adıyla (**/dev/hdc1** gibi), isterseniz de bağlı bulunduğu dizin adıyla belirtebiliyorsunuz.

umount komutunu kullanabilmeniz için root kullanıcı olmanız gerekir. Normal kullanıcıların CD-ROM sürücü ve disket sürücülerini çözebilmeleri için **eject** komutunu kullanmaları gerekir.

```
eject cdrom  
eject floppy  
eject
```

Parametresiz kullanıldığında, varsa CDROM sürücüdeki CD unmount edilir ve kapağı açılır.

Siz hala sormadınız; bari konuyu biz açalım: **mount** komutunun verdiği listede sistemin kurulması sırasında ayırdığınız takas alanının (örneğin **/dev/hda5**) nereye iliştiirildiğini göremiyorsunuz. Bu son derece normal çünkü takas alanı root dahil hiç kimsenin erişemeyeceği bir alandır. Takas alanının yönetimi tamamen LINUX çekirdeğine aittir. Sistem yöneticisinin takas alanıyla ilgili tek denetim şansı takas işlemlerini durdurmak ve başlatmaktır. (**swapon** ve **swapoff** komutları). Ancak bu işi yapmaya da pek gerek olmaz.



fsck

LINUX dosya sistemleri (ister ext2, ister reiserfs olsun) oldukça karmaşık veri yapılarıdır. Özellikle güç kaybı ya da reset düğmesine basılması nedeniyle kapanan sistemlerde bu dosya yapısının bozulma, daha doğrusu tutarlılığının kaybolması olasılığı vardır. Bir dosya sistemi içindeki tutarlılık bozulduğunda, sistem kullanılmaya devam edilirse sorun gittikçe büyür. Bu nedenle LINUX, sistemin düzgün kapatılıp kapatılmadığını izlemek için birtakım mekanizmalar kullanır. Düzgün olmayan bir kapatmadan sonraki ilk açılışın hemen başında dosya sistemleri üzerinde otomatik olarak tutarlılık testi başlatılır ve bu test bir şekilde başarıyla geçilmeden sistem tam olarak açılmaz.

Sorun çıkarma potansiyeli olan dosya sistemleri **fsck** programıyla test edilir. Test sırasında rastlanan sorunların çoğu **fsck** tarafından otomatik olarak düzeltilir. **fsck** yapacağı bir düzeltmenin tehlikeli olabileceğini, yani düzeltmenin başarılı olmama olasılığını hissederse yapacağı değişiklik için izin ister. Genellikle bu izinleri vermek zorundasınızdır.

Dosya sistemlerinin tutarlı olması çok önemli olduğu için LINUX her dosya sistemini belli bir sayıda; örneğin 20 mount edişte bir **fsck** ile kontrol eder. Bu nedenle sisteminizin bazı açılışları alıştığınız süreden uzun sürer. Bir dosya sisteminin **fsck** ile kontrol edilme süresi o diskin büyüklüğüne bağlıdır. 20 GByte ve oldukça dolu bir disk için bu süre 5 dakika kadar sürebilir. Sürenin uzun olması nedeniyle, birden fazla dosya sistemi kontrol edilecekse **fsck** kendi kendinin gereği kadar kopyasını paralel olarak başlatır.

Tipik LINUX bilgisayarların yılda, bilemediniz 3-5 kez kapatılması gerekebileceğini düşünürseniz, bu 20 seferde bir otomatik **fsck** başlatılmasının kullanıcıları ve sistem yöneticisini pek de rahatsız etmeyeceğini belirtmeliyiz.

Süreçler

LINUX işletim sisteminin çok kullanıcı ve çok işli bir işletim sistemi olduğunu şimdiye kadar birkaç kez vurgulamıştık. Burada bir daha açıklamak gerekirse; LINUX işletim sisteminin denetimindeki bir bilgisayar aynı anda birden fazla kullanıcıya hizmet edebilir. Her kullanıcı için birden fazla program çalıştırabilir; ve bu arada geri planda çeşitli servisler çalıştırılarak ağ üzerinden gelen isteklere de yanıt verilebilir. (Veritabanı sunucusu, web sunucusu gibi)

Bir LINUX bilgisayarı üzerinde çalışan işler:

- Kullanıcı programları,
- Servis (sunucu) programları ve
- Sistemin kendi gereksinimi için çalıştırılan programlardan oluşur.

İşletim sistemi, yöneticisinin ve kullanıcılarının belirttiği işler dışında kendi işlerini de bir sürü programı aynı anda çalıştırarak yürütür. Örneğin, **at** komutuyla belli bir tarih ve saatte başlatılması gereken işleri izleyen **atd** programı gibi; bilgisayar ağı üzerinden gelebilecek TCP/IP isteklerini değerlendiren **inetd** programı gibi, belirli aralıklarla disklere yapılan kayıt işlemlerinin fiziksel olarak disklere kaydedilmesi işini düzenleyen (*flushing disk buffers*) **update** programı gibi sistem yazılımları geri planda çalışır. Kimi sürekli çalışır, kimi de gerektiği zaman çalışır, işi bitince durur. Tipik bir LINUX bilgisayarda, kullanıcı programları dışında çok sayıda sistem programı sürekli çalışıyor durumdadır.

Bir LINUX bilgisayarda, belirli bir anda, merkezi işlem birimini (ya da birimlerini) ve belleği paylaşarak birlikte çalışan programlara genel anlamda **süreç** (*process*) adı verilir. Süreç kavramı program kavramından biraz daha değişik. Bir program birden fazla süreçten meydana gelebilir; hatta tek bir süreç olarak çalışmakta olan bir program gerek gördüğünde kendisinin bir kopyasını çıkarıp onu da yeni bir süreç olarak çalıştırmaya başlayabilir. Bunun en yaygın örneği web sunumu işini yapan **httpd** (*apache*) ve veritabanı

yönetim/sunum işini yapan **mysqld** yazılımlarıdır. Sistem yöneticisi, sistemin açılışı sırasında bu programlardan birer tane başlatır. Bu programlar, ağ üzerinden istek geldikçe kendi kendilerinin kopyalarını çıkarıp onları da çalıştırmaya başlarlar. (LINUX dilinde “spawn ve fork”)

Süreçlerin Merkezi İşlem Birimi (MİB) zamanını paylaşmaları işletim sisteminin çekirdeği tarafından koordine edilir.

MİB paylaşımına ilişkin önemli bir terim de “zaman dilimi” (*time slice*) kavramıdır. Her süreç, MİB’ni belirli ve kısa bir süre (tipik olarak 10-100 milisaniyelik zaman dilimleri) için kullanabilir. Zaman dilimini dolduran süreçler beklemeye alınıp, MİB, sırada bekleyen bir başka sürece tahsis edilir. Bu şekilde tüm süreçler aynı anda çalışıyormuş gibi bir etki elde edilir. Bu süreçlerin birden fazla kullanıcıya ait olmaları durumunda da, MİB kullanıcılar arasında paylaştırılmış olur. Eğer bilgisayarda birden fazla MİB varsa, LINUX çekirdeği işleri bu MİB’lerine otomatik olarak dağıtabilir. LINUX’un çok kullanıcı olma özelliğinin altında yatan temel mekanizma budur.

Herhangi bir anda, bilgisayarda çalışan süreçlerin neler olduğunu görmek isterseniz,

ps ax

komutunu kullanabilirsiniz.

```

cayfer@notebook.ijman.bilkent.edu.tr: /home/cayfer/public_html - Shell - Konsolle
[cayfer@notebook public_html]$ ps ax
  PID TTY          STAT       TIME COMMAND
    1 ?           S           0:04  init
    2 ?           SM          0:00  [keventd]
    3 ?           SMN         0:00  [ksoftirqd_CPU0]
    4 ?           SM          0:04  [kswapd]
    5 ?           SM          0:00  [bdflush]
    6 ?           SM          0:00  [kupdated]
    7 ?           SM<         0:00  [ndrecoveryd]
   11 ?           SM          0:02  [kjournald]
   98 ?           S           0:00  devfsd /dev
  184 ?           SM          0:00  [khud]
   841 ?          SM          0:00  /sbin/cardmgr
   856 ?          SM          0:00  portmap
   870 ?           S           0:00  syslogd -n 0
   878 ?          SM          0:00  klogd -2
   938 ?           S           0:00  gpm -t ps/2 -n /dev/psaux
   955 ?          SM          0:00  yperv
  1055 ?          S           0:14  xfs -port -1 -daemon -droppriv -user xfs
  1111 ?          SM          0:00  rpc.yxfrd
  1126 ?          S           0:00  /usr/sbin/atd
  1147 ?          SM          0:00  saslauthd -a pam -T

```

Kim Korkar LINUX'tan?

ps ax komutuyla alınan çalışan süreçler listesi genellikle çok uzundur. (Tipik olarak 300 satır kadar.) Yalnızca içinde çalışmakta olduğunuz kabukla ilgili süreçleri görmek için **ps** komutunu parametresiz olarak kullanabilirsiniz. Kendinize ait süreçleri görmek isterseniz (farklı kabuklar içinde çalışıyor olabilirler) kullanmanız gereken komut:

ps u cayfer

gibi olmalıdır.

Her sürecin PID (Process ID) denilen kendine özgü bir numarası vardır. Bir süreçle bilgi alışverişinde bulunmak ya da o sürece mesaj göndermek isteyen diğer süreçler, bu numaraları kullanırlar. Örneğin 4261 numaralı süreci kesmek (LINUX terminolojisinde “öldürmek”) için bu sürece “kendini öldür” anlamında bir mesaj göndermek gerekir. LINUX'ta temel bazı haberleşme işleri için numarayla ve harf dizileriyle kodlanmış mesajlar vardır. Örneğin bir süreci öldürmek istiyorsanız o sürece “kendini öldür” anlamında “9” mesajını göndermeniz yeterli olacaktır. Tabii ki, bu sürecin sizin sözünüzü dinleyip intihar etmesi için sizin ya root olmanız ya da süreci başlatan kullanıcı, yani sürecin sahibi olmanız gerekir.

Çalışan süreçlerin listesini daha ayrıntılı bir şekilde görmek isterseniz

ps alx

komutunu kullanabilirsiniz.

```
l[cayfer@notebook cayfer]$: ps alx
 F  UID  PID  PPID  PRI  NI   VSZ  RSS  MCHAN  STAT  TTY      TIME  COMMAND
100  0    1    0    8    0  1288  84  do_sel  S    ?        0:04  init
040  0    2    1    9    0    0    0  contex  SM   ?        0:00  [keventd]
040  0    3    1   19   19    0    0  ksofti  SMN  ?        0:00  [ksoftirqd_CPU0]
040  0    4    1    9    0    0    0  ksuapd  SM   ?        0:04  [ksuapd]
040  0    5    1    9    0    0    0  bdf_lus SM   ?        0:00  [bdf flush]
040  0    6    1    9    0    0    0  kupdat  SM   ?        0:00  [kupdate]
040  0    7    1   -1  -20    0    0  nd_thr  SM<  ?        0:00  [ndmthr]
040  0   11    1    9    0    0    0  end     SM   ?        0:02  [end]
140  0   98    1    9    0  1720  192  devfsd  S    ?        0:00  devfsd /dev
040  0  184    1    9    0    0    0  end     SM   ?        0:00  [khubd]
040  0  841    1    9    0  1412  4    do_sel  S    ?        0:00  /sbin/cardmgr
140  32 856    1    9    0  1416  4    do_pol  S    ?        0:00  portmap
140  0  870    1    9    0  1360  176  do_sel  S    ?        0:00  syslogd -n 0
140  0  878    1    9    0  1936  4    do_sys  S    ?        0:00  klogd -2
140  0  938    1    9    0  1336  60  nanosl  S    ?        0:00  gpm -t ps/2 -n /dev/psa
140  0  955    1    9    0  1412  4    do_pol  S    ?        0:00  upserv
140  414 1055   1    9    0  6972  2540  do_sel  S    ?        0:14  xfs -port -1 -daemon -d
140  0 1111   1    9    0  1468  4    do_pol  S    ?        0:00  rpc.yppxfrd
040  2 1126   1    9    0  1312  108  nanosl  S    ?        0:00  /usr/sbin/atd
040  0 1147   1    9    0  1500  4    wait_f  S    ?        0:00  saslauthd -a pam -T
040  25 1179   1    9    0  9988  464  rt_sig  S    ?        0:00  named -u named
040  25 1183  1179   9    0  9988  464  do_pol  S    ?        0:00  named -u named
140  25 1188  1183   9    0  9988  464  rt_sig  S    ?        0:00  named -u named
040  25 1189  1183   9    0  9988  464  nanosl  S    ?        0:00  named -u named
040  25 1190  1183   9    0  9988  464  do_sel  S    ?        0:00  named -u named
140  0 1203   1    9    0  2660  4    do_sel  S    ?        0:00  /usr/sbin/sshd
140  0 1238   1    9    0  2068  4    do_sel  S    ?        0:00  xinetd -stayalive -reus
```

Bu listedeki önemli bilgiler şunlardır:

ps Komutu Rapor Ayrıntıları	
Bilgi Alanı	Açıklama
UID	Sürecin sahibinin sayısal kullanıcı kodu.
PID	(Process ID) Süreç tanımlama numarası.
PPID	Süreci bir başka süreç başlattıysa o sürecin numarası. (Parent Process ID)
TTY	(Teletype : Çok eskilerden kalan bir alışkanlık.) Sürecin hangi terminalden başlatıldığını belirtir. ?: Herhangi bir terminalden değil, sistemin kendisinin başlattığı süreçleri gösterir. ptsN: (pts1, pts2 gibi) Ağ üzerinden bağlanmış kullanıcılar tarafından başlatılmış süreçleri gösterir. ttyN: (tty1, tty2 gibi) Seri arabirim üzerinden (modemle bağlantı gibi) bağlı kullanıcılar tarafından başlatılmış süreçleri gösterir.
STAT	(<i>Status</i>) Sürecin bulunduğu duruma ilişkin bir kod. R (<i>Runnable</i>): Çalışabilir durumda, sırasını bekliyor. S (<i>Sleeping</i>): Uyuyor. Bir şeylerin olmasını bekliyor Z (<i>Zombie</i>): Bu süreç ile bağlantılı tüm diğer süreçler bitmiş veya ölmüş; bunun da bitmiş olması gerekirdi ama bir nedenle ölememiş. ps listesinde hâlâ görünüyor olması zararsızdır.
TIME	Sürecin ne kadar zamandır çalıştığını gösterir.
COMMAND	Süreci başlatan komut satırıdır. (Varsa)

Şeytanlar (Daemons)

LINUX süreçleri arasında “**daemon**” (“deymın” diye okunur) sözcüğüyle tanımlanan bir özel süreç çeşidi vardır. Bunların normal süreçlerden tek farkı hiçbir şekilde konsola ya da ekrana mesaj göndermemeleridir. Daemon'lar geri planda sessizce çalışırlar. Eğer başlarına kayda değer bir iş gelirse bunu **/var/log/syslog** dosyasına ya da kendi log dosyalarına kaydederler. Genellikle sistemin açılışıyla birlikte başlatılıp sistem kapanıncaya kadar sürekli çalışırlar. Bir kural olmamakla birlikte daemon tipi süreçlere ilişkin programların isimleri genellikle “**d**” harfiyle biter. (**httpd**, **ftpd**, **named** gibi)



“Daemon”, İngilizce'de “şeytan”, “zebani” anlamında kullanılan bir sözcüktür. Geri planda sessizce ama sürekli çalışan programlara daemon adı verilmesinin ilginç bir nedeni var:

Büyük fizikçi Maxwell, gazların dinamiğini kolay anlatmak için iki bölümlü hayali bir kutu tasarlamış. Kutunun iki bölümü arasında ancak bir gaz molekülü geçebilecek kadar bir delik olduğunu ve ancak yeteri kadar kinetik enerjiye sahip moleküllerin bu delikten diğer tarafa geçebileceğini söylemiş. Moleküllerin kinetik enerjileri de sıcaklıklarıyla doğru orantılı olduğu için bir süre sonra kutunun iki bölümünün ısısının eşitleneceğini böyle açıklamış. Kavramı dramatize etmek için de deliğin başında bir zebaninin oturduğunu ve delikten geçmeye çalışan tüm moleküllerin enerji düzeylerini kontrol edip ancak yeteri kadar hızlı olanların geçmesine izin verdiğini anlatmış. (Maxwell'in şeytanı hakkında daha ayrıntılı bilgiyi Bilim ve Teknik dergisinin Haziran 2003 sayısında bulabilirsiniz.)

MULTICS işletim sistemini geliştiren ekipte yer alan ve asıl mesleği fizik olan Fernando J. Corbato da arka planda sessizce çalışıp diğer süreçleri denetleyen programlara, Maxwell'in zebanisinden esinlenerek “daemon” adını vermiş.

Daemon'lara klasik örnekler olarak **httpd** ve **named** gösterilebilir. **httpd** web sunucu yazılımı, **named** ise “www.pusula.com” gibi alfabetik internet adreslerinin sayısal IP karşılıklarını bulan DNS yazılımıdır.

Bazen daemon tipi süreçler ölebilirler. Bunun nedeni genellikle programın bir hatası ya da içinden çıkılmaz bir sorun yüzünden sistem yöneticisi tara-

fından özellikle öldürülmeleridir. Böyle durumlarda daemon programın yeniden başlatılması çoğunlukla sorunu çözer.

Süreç Öldürmek

Diyelim ki başlattığınız bir iş kontrolden çıktı ve istediğiniz ya da beklediğiniz gibi davranmıyor ya da yanlış iş başlattığınızı farketmişsiniz. Doğal olarak bu işi hemen kesmek istiyorsunuz. İlk denemeniz gereken Ctrl-C tuşudur. Olmazsa Ctrl-D tuşu... (Fazladan basacağınız Ctrl-D kabuk programınızı sona erdirip terminal ekranınızın kapanmasına neden olabilir.) Ctrl-C ve Ctrl-D tuşlarıyla bir süreci öldürebilmeniz için, o sürecin çalıştırıldığı ekranın açık olması ve daha önemlisi o sürecin sahibi olmanız gerekir.

Öldürmek istediğiniz bir programı Ctrl-C veya Ctrl-D tuşlarıyla öldüremiyorsanız o programa ilişkin sürecin numarasını öğrenip, sürece “öl” mesajını göndermeniz gerekir.

Bunun için:

1. Süreç size ait değilse root kullanıcı olun. Bunun için herhangi bir telnet penceresinden,
su -
komutunu verip ardından sorulan şifreyi girin.
2. Uygun bir **ps** komutuyla (“**ps ax**”) çalışmakta olan süreçlerin bir listesini alın.
3. Bu listeye bakarak sorun çıkararak sürecin numarasını öğrenin. (Diyelim ki 5443)
4. “**kill 5443**” komutuyla 5443 numaralı sürece “*kendini öldür*” mesajını gönderin.
5. Tekrar aynı **ps** komutunu kullanarak sürecin listeden kaybolup kaybolmadığını kontrol edin.
6. Eğer sorun yaratan süreç hala direniyorsa,

kill -9 5443

komutuyla biraz daha sert bir emir olan “*kendini koşulsuz öldür (geber)*” mesajını gönderin.

Kim Korkar LINUX'tan?

Gene olmadı diyelim.

Süreci hâlâ öldüremiyorsanız, kabuk programınızı öldürmeyi deneyiniz. Hâlâ direniyorsa, başı bozuk süreci öylece bırakmayı da düşünebilirsiniz. Bu sürecin sisteme ne kadar yük getirdiğini **top** komutuyla görebilirsiniz. Eğer bu süreç performans açısından ya da bir başka şekilde sorun çıkarmıyorsa bırakın ortalıkta zombi gibi sürsün. Yok eğer sorun çıkarıyorsa sistemini zi düzgün olarak kapatıp tekrar açın.



Sisteminizde denetimden çıkmış, öldüremediğiniz süreçler varsa ve bunlar diğer işleri bozuyorsa sisteminizi kapatıp açmaktan başka seçeneğiniz kalmamış demektir. Ancak böyle bir durumda bile bilgisayarınızı elektrik anahtarından kapatmayı veya reset düğmesine basmayı aklınızdan dahi geçirmemelisiniz! LINUX altında çalışırken böyle bir durumla karşılaşma olasılığınızın çok çok düşük olduğunu belirtmek isteriz.



Bazı programlar bellekte birden fazla kopya olarak yer alır ve çalışırlar. Bunun en çok rastlanan örneklerinden biri çok sayıda pencerede çalışan Mozilla ya da Netscape web tarayıcılarıdır. Bu yazılıma ait süreçlerin hepsini birden öldürmeniz gerekirse, teker teker süreç numaralarını bulup öldürmektense

```
killall -9 mozilla-bin
```

komutunu kullanabilirsiniz. Dikkat ederseniz **killall** komutunda süreç numarası değil süreç adı belirtiliyor. Web tarayıcınızın çalışırken başlatmış olduğu süreçlerin isimlerini bilmiyorsanız **ps ax** komutu işinize yarayacaktır.

Link Kavramı ve 1n Komutu

Şimdi biraz mistik bir konudan söz edeceğiz. LINUX işletim sistemi altında bazı dosyalar aslında buldukları yerde olmayabilirler. Evet, yanlış okudunuz! Diskin üzerinde yer alan bazı dosyalar aslında orada olmayabilir; hatta bir dosyanın sistemde tek bir kopyası olmasına rağmen, bu dosya birden fazla dizinde, üstelik farklı isimlerle yer alabilir. Kavraması ve kullanması zor bir kavram fakat bir kez mecbur kalıp da kullandınız mı hoşunuza gideceğine emin olabilirsiniz.

Galiba en iyisi bir örnekle anlatmak:

Farzedin ki, bilgisayarınıza **matlab** isimli yeni bir uygulama programı yüklemeniz gerekiyor. Ancak, programın bir gereği olarak, program paketine ilişkin dosyaların **/usr/local/matlab** diye bir dizinin altına yer alması gerekiyor. Eh! Olabilir. Ancak bir sorun var! **/usr** diskinde, yeni programa ilişkin dosyalar için yeterli boş yer yok ve burada silebileceğiniz gereksiz dosyalar da yok!

İşte mistik **ln** kavramı, bu problemi LINUX'un şanına yaraşır bir yöntemle çözmenizi sağlar.

Yeteri kadar boş yeri olan disk bölümlerinden birinde, örneğin **/home** dizinin bulunduğu disk bölümünde, yeni yükleyeceğiniz program için bir dizin yaratınız: (**/home** dizini altında yeni dizin yaratabilmek için root kullanıcı olmanız gerekecektir.)

```
mkdir /home/matlab
```

Sonra, bu dizini, **/usr/local** altında yer alıyormuş gibi gösterebilmek için,

```
ln -s /home/matlab /usr/local/matlab
```

komutunu veriniz.

Böylece, gerçekte **/home** altında yer alan **matlab** dizini, aynı zamanda **/usr/local** altında da varmış gibi olacaktır. Bu dizini kullanırken isterseniz **/home/matlab**, isterseniz **/usr/local/matlab** dizin adreslerini kullanabilirsiniz. Böylece **matlab** dizinini **/usr/local** altında görmek isteyen **matlab** yazılımını kandırmış oldunuz.

Link kavramının çok işe yarayabileceği, bir öncekine benzeyen bir senaryo daha anlatabiliriz.

Diyelim ki, elinizde **mhsb2002** isimli bir dosya var ve muhasebe departmanının kullandığı muhasebe programı bu dosyayı mutlaka bu isimde görmek istiyor. Öte yandan yeni satın aldığınız bir mali analiz programı, aynı muhasebe verilerini **acct2002** adıyla görmek istiyor.

Söz konusu dosyanın adı **mhsb2002** olduğu zaman muhasebe departmanının sorunu yok ama siz mali analiz programını çalıştıramıyorsunuz. Analiz çalışmaları için dosyanın adını değiştirseniz, siz çalışabiliyorsunuz ama bu sefer muhasebe departmanındaki program kullanılamıyor. Dosyanın adını **mhsb2002** olarak tutup, kendi analiz çalışmalarınız için **acct2002** adlı bir kopyasını çıkarmak da düşünülebilir ama çok kullanıcı ortamında siz analizler üzerinde çalışırken öte taraftan muhasebe personeli yeni kayıtlar girip sizin analizlerinizi eskimiş kayıtlar üzerinde yapmanıza neden oluyorlar. İşte böyle bir durumda link kavramı ve **ln** komutu gene sizi kurtaracaktır.

```
ln ./mhsb2002 ./acct2002
```

Bu komutla **mhsb2002** dosyasını **acct2002** isimli bir dosyaya bağladığınızda (aslında sadece tek bir asıl kopya var; o da **mhsb2002**. **acct2002** isimli bir dosya ise aslında yok, **acct2002** asıl dosyanın bir başka adı), bu sayede **mhsb2002** dosyasında yapılan her değişiklik **acct2002** diye tanınan dosyada da aynen gözlenebilecektir. İşin bir başka yararlı tarafı da, **acct2002** isimli dosyanın diskte hiç yer kaplamayacak olmasıdır.

Bu örnekler arasında, dikkatinizi çekmiş olduğunu umduğumuz önemli bir fark var. İlk örnekte, yani matlab örneğinde, **ln** komutunda **-s** diye bir parametre kullandık; oysa ikinci muhasebe örneğinde kullanmadık!

- Eğer **ln** komutuyla birbirlerine bağlanacak olan dosya sistemi elamanları birer dizinse; **-s** parametresini kullanmak zorundasınız.
- Eğer **ln** komutuyla birbirlerine bağlanacak olanlar birer dosyaysa ama farklı dosya sistemlerindeyse (örneğin, farklı disklerdeyse), gene **-s** parametresini kullanmak zorundasınız.
- **ln** komutuyla, bir dizini ve bir dosyayı birbirlerine bağlayamazsınız. Bağlanacak olanların ikisi de dizin, ya da ikisi de dosya olmalıdır.

Aynı dosya sisteminde yer alan ve **"-s"** kullanılmadan bağlanmış olan dosyalardan birini silmeniz diğerini etkilemez. Asıl dosyayı silseniz bile, LINUX bağlantıyı fark edip dosyayı diskten gerçekten silmeyecektir. LINUX her dosya için yapılmış bağlantıları sayar ve her silme işleminde bağlantı sayısını bir azaltır. Gerçek silme işi bu bağlantı sayısı sıfırlanınca yapılır.

Farklı dosya sistemlerinde yer alan bağlantılar için bu bağlantı sayma işine güvenmeyiniz. Farklı dosya sisteminde bağlantısı olan bir dosyayı silerseniz başınız derde girer. Asıl dosya silinir ve diğer sistemde gerçekte var olmayan bir dosyayı gösteren bir bağlantınız kalır.



Bir dosyanın gerçekten var olan bir dosya mı, yoksa sadece bir bağlantı mı (link) olduğunu anlamak için **ls** komutunu **-l** seçeneği ile kullanmanız gerekir. İçinde bağlantılı dosyalar bulunan bir dizinde **ls -l** komutunu vererek, alacağınız listede bağlantılı dosyaları ve hangi dosyaya bağlantılı olduklarını açıkça görebilirsiniz.

```

cayfer@notebook lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsol - <4>
[cayfer@notebook cayfer]# ls -al /etc/rc*
lrwxrwxrwx 1 root root      7 Ara 29 17:52 /etc/rc -> rc.d/rc
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc0.d -> rc.d/rc0.d
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc1.d 1 -> rc.d/rc1.d
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc2.d 1 -> rc.d/rc2.d
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc3.d -> rc.d/rc3.d
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc4.d -> rc.d/rc4.d
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc5.d -> rc.d/rc5.d
lrwxrwxrwx 1 root root     10 Ara 29 17:52 /etc/rc6.d -> rc.d/rc6.d
lrwxrwxrwx 1 root root     13 Ara 29 17:52 /etc/rc.local -> rc.d/rc.local
-rwxr-xr-x 1 root news    4565 Eyl 13 17:41 /etc/rc.news
lrwxrwxrwx 1 root root     15 Ara 29 17:52 /etc/rc.sysinit -> rc.d/rc.sysinit

```

Bu örnek listeye göre, aslında **/etc/rc0.d** diye bir dosya bulunmadığı, bu isimde **/etc** dizini altındaki **rc.d** dizinin altındaki **rc0.d** dosyasına bir bağlantı yapılmış olduğu anlaşılmaktadır. (**/etc/rc0.d** → **rc.d/rc0.d**)

Dikkat ederseniz, **ls -l** komutunun verdiği listede, gerçek bir dosya (dizin) değil de, bağlantı olan dosyalara (dizinlere) ait satırların başında bir **l** harfi bulunmaktadır.

İpin ucunu kaçırmayacağınıza eminseniz, bağlantılara bağlantı yapabilirsiniz.

“Pipe” Kavramı

“Pipe” (boru) kavramı, daha önce açıklamış olduğumuz “Giriş/Çıkış Yönlendirme” kavramıyla kolayca karıştırılan, bu yüzden de dikkatle ele alınması gereken bir kavramdır. Kısaca bir tekrarlamak gerekirse; “çıkış yönlendirme (>”, çalıştırılan bir programın, standart çıktı birimine yazacağı satırların bir dosyaya yönlendirilmesi işlemidir. Aynı mantıkla, verilerini standart giriş biriminden okuyan programlar için “giriş yönlendirme (<)”; verilerin bir dosyadan okunmasını sağlayan işlemidir.

“**Piping**” işlemiyse, gene bir çeşit yönlendirmedir; ancak, **bir programın standart çıktısı, bir başka programa standart girdi olarak** yönlendirilir.

“**Pipe**” kurmak için, aynı komut satırında en az iki program birden başlatmalı ve bu iki programa ilişkin komutların arasına “ ” karakterini yerleştirmeniz gerekir.

Şimdi **grep** ve **more** komutlarını birlikte kullanarak (biliyorsunuz; pipe kurmak için en az iki komut gerekiyor) pipe kavramının kullanımına birkaç örnek verelim: (**grep** programını şimdilik öylesine bir komut olarak düşünün lütfen. **grep** hakkında söyleyecek o kadar çok şey var ki, ona da dört-beş sayfa ayırdık.)

```
grep ayfer * | more
```

Bu komut satırında **grep** ve **more** programları aynı anda başlatılıyor. “**ayfer**” ve “*****” **grep** programına parametre olarak gönderiliyor. **grep** programının çıktısı ise **more** programına girdi olarak yönlendiriliyor.

“**grep ayfer ***” komutu çalışma dizini içindeki dosyalarda “**ayfer**” karakter dizisini arar ve bulduğu dosyalarla bu dosyalar içinde “**ayfer**” sözcüğü geçen satırları standart çıktı birimine listeler. Eğer bu liste çok uzunsa, terminal pencerenizin içinde akar gider ve siz pek bir şey göremezsiniz.

grep programının çok uzun olabilecek çıktısını **more** programına girdi olarak yönlendirdiğinizde **more** programı standart girişten gelen satırları içinde bulunduğu terminal ekranına sayfa sayfa listeleyecektir. Ekran her dolduğunda sol alt köşede **-- more --** işareti görünecek ve listelemeye devam edilmesi için sizin bir tuşa basmanız beklenecektir.

Bir başka örnek:

```
ps ax | grep in.named
```

- **ps** ve **grep** programlarını birlikte başlatır,
- **ps** programının **ax** parametresiyle kullanıldığında oldukça uzun olabilecek çıktısını **grep** programına girdi olarak gönderir,
- **grep** kendisine gönderilen satırlar arasında, içinde “**in.named**” sözcüğü geçenleri bulur ve sadece bu satırları listeler.

Şimdi de sıkı bir pipe örneği...

```
echo Sistemde `who | wc -l` kullanıcı var
```

Bu komut satırında birkaç kademeli bir işlem yapılması istenmektedir. Ayrıca bu komutta kullanılan tırnak işaretinin karakter dizilerini sınırlayan tırnaktan olmadığına, Q-Türkçe klavyede virgülle aynı tuşta yer alan “ters tırnak” işareti olduğuna dikkatinizi çekeriz.

who ve **wc** programları birlikte başlatılacak ve **who** programının çıktısı standart girişindeki satır, kelime ve karakterleri sayan **wc** programına gönderilecektir. (**-l** seçeneği yalnızca satırların sayılmasını sağlıyor.) **wc** programının çıktısıysa, (**who** komutunun listelediği satırların sayısı) tırnaklar arasına yerleştirilerek; örneğin üç kullanıcı varsa, “Sistemde 3 kullanıcı var” dizisine dönüşecektir. Bu dizi de **echo** programına girdi olarak transfer edilecektir. **echo** programı ise parametrelerini aynen ekrana gönderir. Böylece ekranda “**Sistemde 3 kullanıcı var**” dizisinin görünmesi sağlanır.

Bu örnekteki komutu, kişisel dizinizdeki **.bashrc** dosyasına eklerseniz, sisteme her bağlandığınızda, sistemde siz dahil, kaç kişinin çalıştığını öğrenmiş olursunuz.

Biraz Nefes Alalım

LINUX işletim sisteminde yüzlerce komut var! Bunların bir kısmını belki de hiç kullanmayacaksınız. Bir kısmını ise çok sık kullanacaksınız. Genellikle hangi komutlardan yararlanacağınız tamamen bilgisayar ne amaçla kullanıldığınıza bağlıdır.

Önceki bölümlerde biraz fazla teknik ayrıntıya kaçtığımızın farkındayız; bu yüzden bu bölümde biraz nefes almak amacıyla, çok önemli olmayan, fakat kullanımı da hoş olan birkaç komuttan söz etmek istiyoruz. Bu komutlardan söz ederken kullanacağımız genel form doğal olarak

```
komut [ -seçenekler ] [ parametre ] [ parametre ] ...
```

olacaktır. Tüm LINUX konsol komutları zaten bu formdadır.

Bu formdaki [] karakterleri, aralarında yer alan seçenek ve/veya parametrelerin isteğe bağlı olduğunu (*optional*) göstermektedir. Örneğin;

Kim Korkar LINUX'tan?

cp [-ri] dosya1 dosya2

formunda verilen bir komutta **-r** veya **-i** veya **-ri** isteğe bağlı parametreler; **dosya1** ve **dosya2** ise zorunlu parametreler olarak anlaşılmalıdır.

Kullanışlı LINUX Komutları

cal [ay] [yıl] (*calendar*)

Parametresiz kullanırsanız, içinde bulunduğunuz ay için bir takvim yaprağı listelenir.

```
cayfer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsolle <5>
[cayfer@notebook cayfer]$ cal
Şubat 2003
Pa Pz Sa Çr Pr Cu Ct
          1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28

[cayfer@notebook cayfer]$
```

Parametre olarak herhangi bir yıl girerseniz, o yıl için 12 aylık bir takvim listelenir.

```
cayfer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsolle <4>
[cayfer@notebook cayfer]$ cal 2003
2003
          Ocak                Şubat                Mart
Pa Pz Sa Çr Pr Cu Ct   Pa Pz Sa Çr Pr Cu Ct   Pa Pz Sa Çr Pr Cu Ct
          1 2 3 4       1 2 3 4 5 6 7 8       1 2 3 4 5 6 7 8
 5 6 7 8 9 10 11       9 10 11 12 13 14 15       9 10 11 12 13 14 15
12 13 14 15 16 17 18   16 17 18 19 20 21 22       16 17 18 19 20 21 22
19 20 21 22 23 24 25   23 24 25 26 27 28       23 24 25 26 27 28 29
26 27 28 29 30 31     30 31

          Nisan                Mayıs                Haziran
Pa Pz Sa Çr Pr Cu Ct   Pa Pz Sa Çr Pr Cu Ct   Pa Pz Sa Çr Pr Cu Ct
          1 2 3 4 5       1 2 3       1 2 3 4 5 6 7
 6 7 8 9 10 11 12     4 5 6 7 8 9 10       8 9 10 11 12 13 14
13 14 15 16 17 18 19   11 12 13 14 15 16 17       15 16 17 18 19 20 21
20 21 22 23 24 25 26   18 19 20 21 22 23 24       22 23 24 25 26 27 28
27 28 29 30           25 26 27 28 29 30 31       29 30

          Temmuz                Ağustos                Eylül
Pa Pz Sa Çr Pr Cu Ct   Pa Pz Sa Çr Pr Cu Ct   Pa Pz Sa Çr Pr Cu Ct
          1 2 3 4 5       1 2       1 2 3 4 5 6
```

sleep n

n parametresi olarak verilen saniye kadar bekler. Herhalde aklınıza ilk olarak böyle bir komutun ne işe yarayacağı sorusu gelmiştir. İlk bakışta pek işe yaramazmış gibi görünen bu komut, kabuk programları yazmaya başladığınızda (*shell programming*) işinize yarayacaktır.

watch [-n saniye] komut

“komut” komutunu, **n** parametresi olarak verilen saniye aralıklarla sürekli olarak çalıştırır. Örneğin,

watch -n 5 who

who komutunu her 5 saniyede bir tekrarlar. Sisteme bağlanmasını beklediğiniz birisi varsa onu beklerken bu komut çok işinize yarayacaktır. **watch** programı Ctrl-C ile kesilinceye kadar çalışır. LINUX'ta programları durdurmanın tek yolu Ctrl-C değildir. Süreçlerle ilgili bölümde çalışmakta olan programları durdurmanın ve davranışlarını denetlemenin, bir başka deyişle çalışan süreçlere sinyal göndermenin yöntemini öğrenmişsiniz.

wc [-lwc] [dosya] (word count)

Parametresi olan dosyadaki satır, sözcük ve karakterleri sayar. Eğer parametre olarak bir dosya adı belirtilmezse, standart girişteki satırlar için bu sayım işini yapar. Sayım sonuçlarını standart çıkışa yazar.

- l seçeneği verilirse, yalnızca satırları,
- w seçeneği verilirse, yalnızca sözcükleri,
- c seçeneği verilirse, yalnızca karakterleri sayar.

Hem satırları, hem de sözcükleri birlikte saydırmak isterseniz, **-lw** seçeneğini kullanabilirsiniz.

```
cayfer@notebook.igjman.bilkent.edu.tr: /home/cayfer - Shell - Konsol <5>
[cayfer@notebook cayfer]$ wc /etc/terncap
16432  77749 703515 /etc/terncap
[cayfer@notebook cayfer]$
[cayfer@notebook cayfer]$ wc -l /etc/terncap
16432 /etc/terncap
[cayfer@notebook cayfer]$
[cayfer@notebook cayfer]$ wc -lw /etc/terncap
16432  77749 /etc/terncap
[cayfer@notebook cayfer]$
[cayfer@notebook cayfer]$ wc -lw < /etc/terncap
16432  77749
[cayfer@notebook cayfer]$ █
```

tail [-n] [-f]

Bir dosyanın son **n** satırını görüntüler. Eğer **n** belirtilmezse son 10 satır görüntülenir. Kullanışlı değil mi? Hele bir de **-f** parametresini öğrenince bu komutu çok seveceksiniz...

-f parametresiyle birlikte kullanıldığında dosya sonuna geldiğinde **tail** programı işini bitirmez ve dosyaya yeni satırlar eklendikçe onları da göstermeye devam eder.

En iyisi durumu bir örnekle anlatmak:

Diyelim ki sisteminiz web servisi verecek şekilde kurulmuş. Eğer çıldırmadıysanız web servisini **apache** isimli programla veriyor olmalısınız. **apache** hem performans hem güvenlik hem de güvenilirlik açısından dünyanın en gelişmiş web sunucu yazılımıdır ve tüm LINUX dağıtımlarında standart olarak bulunur.



Bir anımızı anlatmadan geçemeyeceğiz... Birkaç yıl önce (2000 falan) Web sunucu güvenliği konularını tartışmak için bir liste açılmıştı. Listenin açıldığını duyuran şahıs "burada güvenlikle ilgili tartışmalar yapıp bilgi ve deneyim paylaşacağız" demişti. Birkaç saat sonra aynı listeye birisi "Microsoft IIS güvenliği de tartışılacak mı?" diye bir soru gönderdi. Yanıt çok ilginçti : "IIS'in güvenlikle ne ilgisi var ki?"

Neyse, bu kadar çamur atma yeter. (Şimdilik) Örneğimize dönelim... **apache** web sunucusu (aksi belirtilmedikçe) sunduğu web sayfalarını ziyaret edenlerin log dosyasını tutar. (log yerine uygun bir Türkçe sözcük bulamadık.) Bu dosyada hangi saatte hangi dosyaların hangi istemciler tarafından istendiğinin

kaydı tutulur. Bu bilgiler birçok web sitesi işleticisi için çok değerlidir. Log kayıtları üzerinde ayrıntılı istatistik çalışması yaparak en çok hangi sayfaların ziyaret edildiğini, ziyaretçilerin bu sayfalarda ne kadar zaman geçirdiğini, en çok hangi ülkeden ziyaretçi geldiğini falan öğrenebilirler. Apache log dosyalarının analizi için bir sürü hazır ve özgür program bulabilirsiniz. Bunların en popülerlerinden biri “**analog**” isimli pakettir. (*www.analog.cx*) Neyse...

Apache'nin log kayıtlarını dosyaya eklendikçe görmek isterseniz;

tail -f /var/log/httpd/access_log

komutunu kullanabilirsiniz. Bu komut **/var/log/httpd/access_log** dosyasının son 10 satırını gösterecek fakat **-f** parametresi verildiği için dosya bitince durmayacaktır. Dosyaya yeni satırlar eklendikçe onları da listelemeye devam edecektir. Böylece web sitenizin ziyaretçilerini gerçek zamanda izleyebileceksiniz. (Bu komutu deneyebilmeniz için sisteminizde bir web sunucusu kurulu ve çalışıyor olmalıdır; üstelik birilerinin de web sitenizi ziyaret ediyor olması gerekir ki log dosyasına yeni kayıt düşülsün.)

more

Uzun metin dosyalarını; örneğin log dosyalarını, ekrana sayfa sayfa listelemek için kullanılır. Dosyanın ekrana veya terminal penceresine sığıldığı kadarı görüntüledikten sonra sol alt köşede

--More-- (12%)

gibi bir satır görünür. Bu işaret dosyanın yüzde 12'sine geldiğinizi ve daha görüntülenecek satırlar olduğunu belirtir.

Boşluk tuşuna basarak dosyada sayfa sayfa ilerleyebilirsiniz. Satır satır ilerlemek için Enter tuşunu, geri dönmek içinse **b** tuşunu kullanabilirsiniz.

Bölme işaretiyle (/) aynı **vi**'da olduğu gibi arama da yapabilirsiniz. Örneğin **--More--** işaretinin karşısına **/Ayfer** yazarsanız dosya içinde ilk rastlanan “**Ayfer**” dizisine kadar ilerlersiniz. Son aramayı tekrarlamak için **n** tuşuna basmanız yeterlidir.

less

more komutunun neredeyse aynısıdır; ancak ekrana enine sığmayan satırlar

Kim Korkar LINUX'tan?

için sağa sola ok tuşlarıyla yatay kaydırma yapmanız da mümkündür. Aynı **more** komutunda olduğu gibi bölme işaretiyle dosyanın içinde istediğiniz karakter dizisini arayabilirsiniz.

top

Bilgisayarınız yavaşladığında, bilgisayarınızı en fazla meşgul eden işleri görmek için kullanabileceğiniz bir komuttur. Hem merkezi işlem biriminin kullanım oranlarını hem de takas alanı kullanım oranlarını görebilirsiniz.

```
cayfer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsolle <5>
10:42am up 1 day, 10:35, 7 users, load average: 1.39, 1.35, 1.24
135 processes: 131 sleeping, 4 running, 0 zombie, 0 stopped
CPU states: 98.8% user, 1.1% system, 0.0% nice, 0.0% idle
Mem: 255428K av, 242892K used, 12536K free, 0K shrd, 54112K buff
Swap: 248936K av, 10352K used, 238584K free, 80452K cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT  %CPU %MEM    TIME COMMAND
 2616 news       17   0   116   24   16 R    96.9  0.0  2041m expireover
 4583 root        9    0  84256 17M 2584 R    1.1  7.0   0:22 X
 5041 cayfer   13   0  1084 1084  812 R    0.9  0.4   0:00 top
 4695 cayfer   9    0 12388 12M 10824 S    0.1  4.8   0:03 kdeinit
 4703 cayfer   9    0 14884 14M 12644 S    0.1  5.8   0:03 kdeinit
 4719 cayfer   9    0 13100 12M 11480 S    0.1  5.1   0:02 kdeinit
 4724 cayfer   9    0 13084 12M 11452 S    0.1  5.1   0:01 kdeinit
 4899 cayfer   9    0 14228 13M 11368 S    0.1  5.5   0:05 ksnapshot
   1 root        8    0   132   84   68 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 SWM  0.0  0.0   0:00 ksoftirqd_CPU0
   4 root        9    0    0    0    0 SW   0.0  0.0   0:03 kswapd
   5 root        9    0    0    0    0 SW   0.0  0.0   0:00 bdflush
   6 root        9    0    0    0    0 SW   0.0  0.0   0:00 kupdated
   7 root       -1 -20    0    0    0 SW<  0.0  0.0   0:00 ndrecoveryd
  11 root        9    0    0    0    0 SW   0.0  0.0   0:02 kjournald
  98 root        9    0   904  884  704 S    0.0  0.3   0:00 devfsd
 184 root        9    0    0    0    0 SW   0.0  0.0   0:00 khubb
 841 root        9    0   480  384  316 S    0.0  0.1   0:00 cardmgr
 856 rpc        9    0   484  480  400 S    0.0  0.1   0:00 portnap
 870 root        9    0   552  548  440 S    0.0  0.2   0:00 syslogd
 878 root        9    0  1092 1088  364 S    0.0  0.4   0:00 klogd
 938 root        9    0   428  424  364 S    0.0  0.1   0:00 gpm
 955 root        9    0   532  532  448 S    0.0  0.2   0:00 ypserv
1055 xfs         9    0  5836 5832 1148 S    0.0  2.2   0:11 xfs
```

En üst satırdaki

10:42am up 1 day, 10:35, 7 users, load average: 1.39, 1.35, 1.24

satırı sistemin 1 gün , 10 saat ve 35 dakikadır çalışmakta olduğunu göstermektedir.

Gene aynı satırda o anda 7 kullanıcının sistemi kullandığı (7 telnet bağlantısı yapmış tek kişi de olabilir) belirtiliyor. Bu kullanıcı sayısı sistemde kabuk programı çalıştıran kullanıcı sayısıdır; yani bu makinedeki web sitesini ziyaret edenler ya da bu makinedeki veritabanı sunucusundan yararlananların sayısını içermez.

Aynı satırda ortalama yük oranının son bir dakika içinde 1.39, son beş dakika içinde 1.35, son 15 dakika içindeyse 1.24 olduğunu görüyorsunuz. Bu ortalama yük oranları göreceli sayılardır ve tipik olarak birden küçüktür. Zaman zaman iki, üç, hatta 9'a kadar çıkması da normaldir. Yük oranlarınız genellikle 1 civarında değerlerde dolaşıyorsa sisteminizi tam ve ideal kapasiteyle kullanıyorsunuz demektir. Yük oranları genellikle iki veya üçten büyük değerlerde dolaşıyorsa makinenize biraz fazla yükleniyorsunuz demektir; yani artık merkezi işlem biriminizi daha hızlı biriyle değiştirmenin zamanı gelmiş demektir. Ancak üst modele geçmeye karar vermeden önce, **top** komutunun ürettiği rapordaki

```
Mem: 255428K av, 242040K used, 13388K free, 0K shrd, 54324 Kbuff
Swap: 248936K av, 10352K used, 238584K free 80480K cached
```

satırlarını da gözlemenizi öneririz. Eğer takas alanının kullanım ortalaması yüksekse (yukardaki örnekte görüntünün alındığı anda takas alanının pek kullanılmadığı görülmektedir) merkezi işlem birimini değiştirmektense öncelikle belleğinizi arttırmak daha yararlı olabilir.

top, ekranını her birkaç saniyede bir güncelleştirir. Gerek bu güncelleştirme sıklığını, gerekse rapor düzenini değiştirmek için **top** programına klavyeden bir tuşa basarak verebileceğiniz komutlardan bazıları aşağıdaki tabloda listelenmiştir:

top Komutları	
Tuş	Açıklama
q	top programını durdurur.
P (Büyük P)	İşleri merkezi işlem birimi kullanım oranlarına göre listeler.
M	İşleri bellek kullanım oranlarına göre listeler.
T	İşleri çalışmakta oldukları sürelere göre listeler.
s	Ekranın güncelleşme sıklığını değiştirmek için kullanılır.
boşluk tuşu	Ekranı güncelleştirmek için kullanılır.

which

Kabuk programınızın komut satırından bir LINUX komutu yazdığınızda, bu komuta ilişkin bir program dosyasının diskten belleğe yüklenip çalışmaya başlayacağını biliyorsunuz. Peki bu program dosyasının hangi dizinde bulunduğunu bilebiliyor musunuz? Hatırlatmakta yarar var; sisteme verdiğiniz komutlara ilişkin program dosyaları **PATH** ortam değişkeninizde belirtilen dizinlerde ve belirtildiği sırayla aranacaktır. **PATH** ortam değişkeninizdeki dizinleri görmek için

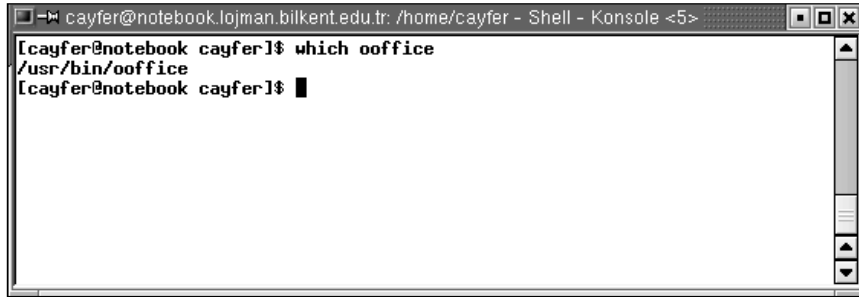
echo \$PATH

komutunu kullanabilirsiniz.



```
cafer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsolle <5>
[cafer@notebook cafer]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/ganes:/home/cayfer/bin
[cafer@notebook cafer]$
```

which komutuna, parametre olarak vereceğiniz bir komutun hangi dizindeki program dosyası kullanılarak çalıştırılacağını öğrenebilirsiniz.



```
cafer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsolle <5>
[cafer@notebook cafer]$ which ooffice
/usr/bin/office
[cafer@notebook cafer]$
```

Yukarıdaki örnekte, telnet ekranınızda **ooffice** komutunu vermeniz durumunda, **/usr/bin** dizinindeki **ooffice** program dosyasının belleğe yüklenerek çalıştırılacağı görülüyor.

zip

PC dünyasının yakından tanıdığınız dosya sıkıştırma programıdır.

Genel formu:

zip [-r] zip_dosyası dosya1 dosya2 ... dosyaN

olan bu komutun ürettiği sıkıştırılmış dosyalar MS-DOS serisi işletim sistemi için Phil Katz tarafından geliştirilmiş olan PKZIP programının ürettiği dosyalarla uyumludur.

```

cayfer@notebook.1ojman.bilkent.edu.tr: /home/cayfer/public_html - Shell - Konsolle
[cayfer@notebook cayfer]# cd public_html
[cayfer@notebook public_html]# zip html.zip *html
  adding: ColorCalc.html (deflated 44%)
  adding: colorcodes.html (deflated 90%)
  adding: cope3.html (deflated 86%)
  adding: cope.html (deflated 91%)
  adding: CTutorial.html (deflated 74%)
  adding: deneme.html (stored 0%)
  adding: index.html (deflated 62%)
  adding: kn.html (deflated 95%)
  adding: l.html (deflated 63%)
  adding: liste.html (deflated 95%)
  adding: localhome.html (deflated 68%)
  adding: MS02-065.asp.html (deflated 69%)
  adding: mysql-trk.html (deflated 51%)
  adding: netscape-test.html (deflated 41%)
  adding: perl.html/ (stored 0%)
  adding: photo.html (deflated 27%)
  adding: procmail.html (deflated 70%)
  adding: Project2.html (deflated 13%)
[cayfer@notebook public_html]#

```

-r seçeneğiyle birlikte kullanıldığında sıkıştırılacak dosyalar arasında dizinlerin de bulunması durumunda, o dizinleri ve varsa, alt dizinleri de sıkıştırarak “**zip_dosyası**” içine yerleştirilecektir. Komutun tüm seçenekleri hakkında açıklayıcı bilgi için **zip** komutunu hiç parametre kullanmadan verebilirsiniz.

unzip

Adından da anlaşılacağı gibi **zip** komutuyla sıkıştırılmış dosyaları açan programdır.

gzip

Dosya sıkıştırma programları arasında LINUX dünyasında en popüler olanı, **gzip** programıdır. GNU organizasyonu tarafından genel kamu lisansı ile bilgisayar kullanıcılarına armağan edilmiştir.

Kim Korkar LINUX'tan?

gzip programının kullanma mantığı zip'e göre biraz farklıdır. **gzip** dosyaları teker teker ve kendi üzerlerine sıkıştırır. Örneğin:

```
gzip mail2002.log
```

diye bir komut verilirse, **mail2002.log** dosyası sıkıştırılarak **mail2002.log.gz** dosyasına dönüştürülür.

```
gzip *log
```

diye bir komut verildiğinde çalışma dizininde adı "**log**" ile biten beş tane dosya varsa, iş bittiğinde çalışma dizininde adı **.gz** ile biten beş tane sıkıştırılmış dosya oluşur.

Bir dizindeki dosyaları ve alt dizinlerini birlikte sıkıştırıp tek bir sıkıştırılmış dosya elde etmek istediğinizde kullanacağınız komut **gzip** değil, **tar** komutu olmalıdır. **tar** komutunu ileride ayrı bir bölümde anlatacağız.

```
gunzip
```

gzip komutuyla sıkıştırılmış dosyaları açmak için kullanılır. **gzip** komutunun dosyaları teker teker sıkıştırması gibi **gunzip** de sıkıştırılmış dosyaları teker teker açar.

Daha önce **gzip** ile sıkıştırılmış **mail2002.log.gz** dosyasını **gunzip** ile açmak istediğinizde;

```
gunzip mail2002.log.gz
```

komutunu vermeniz yeterlidir.

```
bzip2
```

bzip2 son yıllarda hızla yaygınlaşan bir dosya sıkıştırma programıdır. Sıkıştırma oranı konusunda GNU Lisansı ile dağıtılan **gzip**'den daha başarılı olduğu söylenir. **gzip** ve **gunzip** gibi kullanılır.

```
bzip2 büyük_dosya
```

```
bunzip2 büyük_dosya.bz2
```

BUNLARI BİLİYOR MUYDUNUZ?

Linux'un Desteklediği Donanım Platformları

LINUX işletim sistemi; Intel, AMD, Cyrix, Digital Alpha, SUN Sparc, Apple Macintosh, RS6000, Crusoe, PowerPC, Motorola 68K, Atari, NEC Alpha, IBM S/390, IBM S/370, VAX, MIPS, Playstation, XBox, Fujitsu AP1000+ serisi ve daha birçok merkezi işlem birimine uyarlanmış ve başarıyla kullanılmaktadır.

Tüm bu gerçeklere rağmen LINUX'un geleceği konusunda şüpheleri olanlara diyebilecek bir şey bulamıyoruz.

Kim Korkar LINUX'tan?