Kabuk Programlama
Shell Programming

İlginizi çekmiyorsa bu bölümü atlayabilirsiniz.

Bu bölümdeki amacım, okuyuculara kabuk programlamayı öğretmek değil, sadece bu kavramın nasıl bir şey olduğu konusunda fikir vermek. Aslında oldukça karmaşık ve deneyim isteyen kabuk programlama, tipik UNIX kullanıcılarının pek ilgisini çekmez. Programcılık temeli olan okuyucularaysa oldukça ilginç gelebilir; ancak bu kitapta anlatılanlar kabuk programlamayı öğrenmek için kesinlikle yeterli değildir.

Güzel güzel kabuk programlarından bahsederken konuyu dağıtıp süreç kavramına ve onunla ilgili komutlara daldık. Aslında birbirleriyle yakın ilişkisi olan bu iki kavramı da başka nasıl anlatacağımı bilemedim. Neyse, kabuğumuza dönelim..

Kabuk programları, (**sh**, **csh** ve **tsch**) aslında oldukça gelişmiş birer programlama dilini çözümleyebilecek yeteneğe sahiptir. Genel amaçlı işler için pek kullanışlı olmamakla birlikte ileri düzeydeki UNIX kullanıcıları ve sistem yöneticilerinin oldukça sık kullanılacakları özelliklere sahiptirler.

csh ve sh kabuk programlama dilleri birbirlerinden oldukça farklıdır. Bu kitapta vereceğim örnekler genellikle csh kabuğuna göre olacaktır.

Kabuk programlama hakkında daha fazla ayrıntıya girmek isteyen okuyucular için

The UNIX C Shell Field Guide	**UNIX POWER TOOLS**
Gail & Paul Anderson	J. Peek, Tim O'Reilly & M. Loukides
Prentice-Hall, 1986	O'Reilly & Associates, 1993
ISBN 0-13-937468-X 025	ISBN 0-553-35402-7

isimli kitapları hararetle tavsiye ederim.

MS-DOS'daki **batch** dosyalarını hatırlarsınız. Sık tekrarlanacak komut dizilerini, uzantısı BAT olan bir dosyaya yazıp, sanki bir programmış gibi sadece bu dosyanın adını vererek komut dizisini çalıştırırdık. (Nedense geçmiş zaman kullandım... UNIX işletim sisteminin tadını bir kez alan kullanıcılar için MS-DOS, gerçekten de gerilerde kalmış bir işletim sistemi gibi oluyor.)

kabuk programlama'nın mantığı da aynı **batch** dosyalar gibidir. Tek farkla ki; kabukların koşula bağlı komut çalıştırma, karşılaştırmalar, klavyeden kullanıcıların bilgi girme olanakları gibi özelliklerinin çok, ama gerçekten çok daha gelişmiş olmaları sayesinde karmaşık işlerin yapılmasına olanak sağlarlar.

MS-DOS'ta, bilgisayar açıldığında otomatik olarak çalıştırılması gereken programlarla ilgili komut satırlarının yer aldığı AUTOEXEC.BAT diye bir dosya vardır. UNIX'de de aynı amaca yönelik; ama bu sefer, çok daha fazla sayıda dosya vardır.

Örneğin BSD UNIX'lerde, **/etc** dizinin altında **rc, rc.boot, rc.local** gibi isimleri olan dosyalar; sistem açılışının çeşitli aşamalarında otomatik olarak çalıştırılması istenen programlarla ilgili komut satırlarını içerirler. (System V UNIX'lerde bu dosyalar **/etc/rc**'nin altında yer alan dosyalardır.) Ancak, bu satırlar, basit birer komutlar dizisi yerine, **sh** veya **csh kabuk programlarıdır**. Aynı şekilde, kullanıcıların sisteme **login** veya **logout** ettiklerinde otomatik olarak çalıştırılacak kabuk programları da her kullanıcının kendi **home** dizininde, **.login, .cshrc** ve **.logout** isimli dosyalarda yer alır.

Şimdi isterseniz birkaç örnek kabuk programına göz atalım:

İlk Kabuk Programımız

İlk örneğimiz, sistemde yaratmış olabileceğimiz bazı gereksiz dosyaları, arada sırada temizlemek için kullanacağımız bir kabuk programı yazmakla ilgili...

Öncelikle aşağıdaki UNIX komutlarını **temizle** isimli bir dosyaya kaydediniz. Bu iş için **vi** komutunu kullanabilirsiniz.

```
df
cd  ~
/usr/bin/rm *tmp
cd proglar
/usr/bin/rm  *.o
cd ..
/usr/bin/rm core
df
```

Bir sonraki adımda, bu dosyanın bir program dosyası gibi çalıştırılacağını belirtmemiz gerekir. (Dosya erişim yetkilerini hatırlayınız...)

Bunu yapabilmek için

```
% chmod 755 temizle
```

komutunu veriniz. (**rwxr-xr-x** yetki kalıbı).

Kabuk programımızdaki komutlara şimdi birer birer göz atalım:

`df`	Disklerin ne kadarının kullanıldığını gösteren bir UNIX komutudur. *(Disk Free).* Sistemdeki dosya sistemlerinin (disk bölümlerinin) toplam kapasitelerini ve ne kadarının kullanılmış olduğunu rapor eder. Bu komutu, disklerdeki temizlik öncesi boş yer durumunu görmek için koyduk.
`cd ~`	Kullanıcının **home** dizinine geçmek için. Zaten başkalarına ait dosyaları silemeyiz.
`/usr/bin/rm *tmp`	Adı **tmp** ile biten dosyalarısilmek için. Bu komutta MS-DOS kullanıcılarına garip gelecek iki nokta var. Birincisi ***.tmp** yerine ***tmp** kullanılmış olması! Biliyorsunuz, UNIX'de dosya uzantısı diye bir kavram yok ve noktanın da özel bir anlamı yok. O nedenle, nokta kullansaydık, sadece adının son 4 karakteri **.tmp** olan dosyaları silmiş olurduk. **sirali-tmp** gibi bir ismi olan dosya silinmeden kalırdı. İkincisi de, **rm** komutunun sadece **rm** şeklinde değil, **/usr/bin/rm** şeklinde kullanılmış olması. Silme komutunu sadece **rm** şeklinde kullanmış olsaydık, büyük olasılıkla, **alias** komutuyla **"rm -i"** olarak değiştirilmiş olan **rm** komutu çalışacak ve silinecek tüm dosyalar için birer kere **"Emin misiniz?"** anlamında **"Are you sure?"** sorusu ile karşılaşacaktık.
`cd proglar` `/usr/bin/rm *.o`	**proglar** alt dizinine geçmek ve adının sonu **.o** olan dosyaları silmek için. (Derleyicilerin ürettiği *'object'* dosyalar)
`cd ..`	Bir üst düzeydeki dizine geçmek için (**home** dizinimize geri dönüyoruz).
`/usr/bin/rm core`	hatalı yazılmış programlar ya da doğru yazılmış programları hatalı kullanmamızdan dolayı oluşabilecek **core** isimli dosyaları silmek için.
`du`	Temizlik sonrası disk kullanım *(disk usage)* durumunu görmek için.

Aslında bütün bu silme işlerini tek bir **rm** komutuyla, yapabilirdik; ama o zaman kabuk programlamaya fazla kısa bir örnek vermiş olurduk (!).

```
/usr/bin/rm  ~/*tmp  ~/proglar/*.o  ~/core
```

Şimdi biraz daha karmaşık bir kabuk programına göz atalım.

İkinci Kabuk Programımız

Bu **csh** kabuk programı (adı **merhaba** olabilir) çalıştırıldığında; parametresi olarak verilen kullanıcının sistemde olup olmadığına bakacak; kullanıcı sistemdeyse, **talk** programını başlatarak onunla doğrudan görüşmemizi sağlayacak; yok eğer o kullanıcı sistemde değilse, **mail** programını başlatıp ona bir elektronik mesaj göndermemizi sağlayacaktır.

```
#!/bin/csh
# Ornek bir csh kabuk programi
#
#  9 Mayis 1995 - Ugur Ayfer
#
set w = (`who | grep $argv[1]` )
if ($#w == 0) then
        echo "$argv[1] sistemde degil... mektup gonderiniz..."
        mail $argv[1]
else
        echo "$argv[1]  sistemde... Gorusebilirsiniz...."
        talk $argv[1]
endif
```

İşte bu kabuk programı biraz çetrefilli...

ile başlayan satırlar kabuk tarafından dikkate alınmaz. O nedenle programınız hakkında açıklamalar yapmak için kullanabilirsiniz.

İlk satırdaki **#!/bin/csh** özel bir kalıptır. Bu kalıp, kabuk programınının çalıştırılması sırasında **csh** kabuğunun kullanılması gerektiğini belirtir. Eğer bir başka kabuk çalışıyorsa; yeni bir **csh** kabuğu başlatılır ve program bitince bu yeni **csh** öldürülür.

`set w = (`who | grep $argv[1]` )` komutu biraz karışık.. Bu komut önce **who** programını çalıştırıyor. Bu komutla, sistemde çalışan kullanıcıların listesi üretiliyor. Bu liste ekrana görüntülenmek yerine **grep** programına girdi olarak gönderiliyor *(piping).* Bir filtre olarak görev yapan **grep** programı, kendisine gönderilen satırlar arasında sadece içinde $argv[1]; (yani **merhaba** komutunu kullanırken vereceğimiz parametre) geçen satırları geçiriyor.

Bu liste (içinde ilgilendiğimiz kullanıcının adı geçen satırlar) **w** isimli bir kabuk değişkene atanıyor.

Eğer bu listenin uzunluğu sıfırsa (`if ($#w == 0) then`) ; listede ilgilendiğimiz şahsın adı geçen bir satır yok demektir; yani aradığımız şahıs sistemde değildir. Bu durumda ekrana

```
..... sistemde degil... mektup gonderiniz...
```

mesajını yazıp **mail** programını başlatacağız. Tahmin edeceğiniz gibi, **echo** komutu, parametresini standart çıktı birimine aynen tekrarlar.

Eğer, w listesinin uzunluğu sıfırdan büüyükse, ilgilendiğimiz kullanıcı en az bir iş yapıyor demektir. Bu durumda, (**else**) `..... sistemde... Gorusebilirsiniz...` mesajını yazıp **talk** programını başlatacağız.

Yazdığımız bu merhaba programını kullanabilmek için bir kullanıcın adını parametre olarak vermeliyiz; örneğin :

```
% merhaba maslan
maslan sistemde degil... mektup gonderiniz...
```

mail programı başlatılır...

```
% merhaba reyyan
reyyan sistemde... Gorusebilirsiniz...
```

talk programı başlatılır...

İşte şimdi UNIX'ce konuşmaya başladık...

Sergio Aragones

Daha önce **find** komutundan bahsederken daha kolay kullanılan bir **ff** kabuk programından söz etmiş ve bu kabuk programının listesini vermiştim. Şimdi bu **ff** programını tekrar bir gözden geçirelim.

Önce problemi bir kez daha tanımlayalım :

a) **ff** adlı bir kabuk programı yaratacağız.
b) Bu program tek parametre ile kullanılırsa, çalışma dizinimizde ve alt dizinlerinde, parametrede verilen dosyayı arayan **find** komutunu çalıştıracağız..
c) Eğer program iki parametre ile kullanılırsa, ikinci parametredeki dosyayı birinci parametredeki dizinden başlayarak arayacak bir **find** komutu çalıştıracağız.
d) Programın parametresiz ya da ikiden fazla parametre ile kullanılmasına izin vermeyeceğiz.

```
#!/bin/sh
 case $# in
   1) find . -name "$1" -print;;
   2) find "$1" -name "$2" -print;;
   *) echo "Error. Usage: ff [path] name"
      echo "              ff [path]  \"name*\""
      echo "              ff [path]  \"*name\""
 esac
```

Bu programın birinci satırındaki **#!/bin/sh** özel bir kalıptır. Bir kabuk programı bu kalıpla başladığı zaman; satırların yorumlanması sırasında mutlaka **sh** kabuk programının kullanılacağını belirtir. Kullanıcı **csh** kabuğunu kullanıyor olsa bile, bu satırı görünce UNIX, **sh** kabul programını başlatır, kabuk programı bitince de **sh** kabuğunu öldürür.

İkinci satır olan **case $# in** yapısal programlama dillerinin tamamında bulunan **case** deyimlerinin aynısıdır. **$#** sembolu komut verildiğinde kullanılmış olan parametre sayısıdır.

`1) find . -name "$1" -print;;` satırı, parametre sayısının 1 olması durumunda çalıştırılacak komutu tanımlamaktadır. Bu satırdaki **"$1",** birinci parametrenin aynen buraya yerleştirileceği anlamındadır. Örneğin programımızı **ff aranan** şeklinde çalıştırmışsak, kabuk programımız bunun yerine, **find . -name aranan -print** komutunu çalıştıracaktır

`2) find "$1" -name "$2" -print;;` satırı, parametre sayısının iki olması durumunda çalıştırılacak komutu tanımlamaktadır. Bu satırdaki **"$1"** ve **"$2"** birinci ve ikinci parametrelerin yerleştirileceği pozisyonları göstermektedir. Örneğin programımızı **ff basla aranan** şeklinde çalıştırmışsak, kabuk programımız bunun yerine, **find basla -name aranan -print** komutunu çalıştıracaktır.

```
*) echo "Error. Usage: ff [path] name"
   echo "              ff [path]  \"name*\""
   echo "              ff [path]  \"*name\""
```

satırlarıysa, parametre sayısının 1 ve 2 dışında bir değerde olması durumunda çalıştırılacak komutları tanımlamaktadır. Böyle bir durumda ekrana

```
Error. Usage: ff [path] name
              ff [path]  \"name*\""
              ff [path]  \"*name\""
```

mesajları verilecek ve kullanıcı; hatasından dolayı uyarılmasının yanısıra, UNIX geleneklerine uygun bir desende programın doğru kullanım kalıbı konusunda da aydınlatılacaktır.

En sondaki **esac** kelimesiyse, **case** deyiminin sonunu belirlemektedir.

Kendi geliştirdiğimiz bu komutun kullanımına ilişkin birkaç örnek vermek gerekirse...

`ff kitaplar`	Çalışma dizini ve altındaki dizinlerde **kitaplar** adlı dosya veya dizini ara.
`ff /usr kitaplar`	/usr dizini ve altındaki dizinlerde **kitaplar** adlı dosya veya dizini ara.

`ff "kitap*"`	Çalışma dizini ve altındaki dizinlerde, adı **kitap** karakterleriyle başlayan dosya veya dizinleri ara.
`ff /etc "*kitap*"`	/etc dizini ve altındaki dizinlerde, adının içinde **kitap** karakteri geçen dosya veya dizinleri ara.

Aynı işi yapmak üzere bir csh kabuk programı yazacak olsaydık

```
#!/bin/csh
if  ("$#argv" == 1) then
    find . -name "$argv[1]" -print
endif
if ("$#argv" == 2) then
    find "$argv[1]" -name "$argv[2]" -print
endif
if ("$#argv" < 1 || "$#argv" > 2) then
    echo "Error. Usage: ff [path]  name"
    echo '              ff [path]  "name*" '
    echo '              ff [path]  "*name" '
endif
```

Bu programı yazarken kullanıdığım bazı önemli kavramlar :

`"$#argv"` Komutu verirken kullanılan parametrelerin sayısı
`"$argv[1]"` Komut satırındaki ilk parametre

Sanırım kabuk programlama hakkında bu kadar tanıtma yeter.

Çevreyi Tanıyalım

İyi bir bilgisayar kullanıcısı elinin altındaki kaynakları tanımalı, o kaynakların kuvvetli ve zayıf taraflarının yanısıra kullanım alanlarını iyi bilmelidir. UNIX için bu tanıma süreci, PC'lere göre oldukça uzun sürer. Sanıldığının aksine, bu gecikme UNIX'in zorluğundan değil, büyüklüğünden kaynaklanmaktadır. Ehhh, tabi, büyük olunca da biraz da karmaşık oluyor ama genede öğrenilemeyecek kadar değil.

Hangi UNIX bilgisayarında olursa olsun; bir terminalin başına geçip, **login** etmeyi başarıp, **'ne var, ne yok!'** anlamında bir "**ls /**" çektiğinizde aşağı yukarı aynı listeyle karşılaşırsınız.

```
abc:/home/ayfer> ls /
 Mail/          etc/           lost+found/     sys/
 bin@           export/        mnt/            tmp/
 boot           home/          pcfs/           usr/
 cdrom/         kadb*          quotas          var/
 dev/           lib@           sbin/           vmunix*
abc:/home/ayfer>
```

* Bu örnek liste, BSD UNIX (SUNOS 4.1.1) işletim sistemiyle çalışan, DTK marka bir SPARC iş istasyonundan alınmıştır.

Bu listedeki bir takım dizinler, kullanım amacı açısından tüm UNIX'lerde standarttır. Şimdi bu dizinlere teker teker bir göz atalım...

Dizin	Kullanım Amacı
Mail/	Kullanıcılara gelen elektronik posta mesajlarının toplandığı dizin. Normal olarak kullanıcıların bu dizine doğrudan hiç bir erişim hakkı bulunmaz. Kullanıcılara gelen mesajların, mail yazılımı tarafından kendi home dizinlerine dağıtımı bu dizinden yapılır.
bin@	UNIX komutlarının büyük bir çoğunluğunu oluşturan programların yer aldığı dizin.
boot	Pek standart bir dosya değil. Bu sisteme özgü olsa gerek.

cdrom/	CD-ROM sürücüsü kullanıldığında, sürücüye takılı olan CD üzerindeki dosya sisteminin **mount** edileceği boş bir dizin. **(mount point)**

dev/	UNIX bilgisayarına bağlı olan ve bağlanabilecek tüm donanım unsurlarının işletim sisteminin çekirdek modülü (**kernel**) ile bağlantısının kurulmasını sağlayan özel **dosyamsı** kayıtların yer aldığı dizin. (Bu dizinin altındakiler, ne birer dosya ne de birer dizindir, o yüzden "dosyamsı" sözcüğünü kullandım).
etc/	Sistem yöneticisinin eli ayağı olan dosyaların bulunduğu dizindir. Kullanıcıların ve şifrelerinin tanıtıldığı dosya (**passwd**), bilgisayar ağıyla ilgili tanıtım kayıtlarının bulunduğu dosyalar (**hosts**, **defaultdomain** vs) sistemin açılışı sırasında çalıştırılacak olan kabuk programları (**rc***), yazıcı tanımları (**printcap**), terminal bağlantıları ile ilgili kontrol dosyaları (**termcap**, **ttytab** vs), sistemdeki disklerin **mount** edilmeleri ile ilgili tanıtım dosyası (**fstab**), kullanıcılara yapılacak duyurunun yer aldığı dosya (**motd**) hep bu dizindedir. Bu dizinin başına bir kaza gelirse, o sistem bir daha kolay kolay ayağa kalkamaz.
export/	Bu bilgisayarın disklerinden yararlanarak işletim sistemini yükleyen başka disksiz bilgisayarlar bulunduğu durumlarda kullanılan disk sahalarıdır.
home/	Kullanıcıların **home** dizinlerinin bulunduğu dizindir.
kadb*	**"adb like standalone kernel debugger"** (ne demekse...)
lib@	Standart UNIX kütüphanelerinin bulunduğu dizine (**/usr/lib**) bir bağlantı (**ln** komutunu hatırlayınız). (Bağlantı *(link)* olduğunu **@** karakterinden anlıyoruz.)

lost+found/	Bilgisayarın kurallara uygun bir şekilde törenle (!) kapatılmadığı veya disklerde bir arıza olduğunda yapılan kontrollerde (**fsck : file system check**) gerçek adı ve yeri bulunamayan dosya parçalarının toplandığı dizin. Buraya düşen dosyalar pek kolay kurtarılamazlar. (MS-DOS'daki FILE0001.CHK gibi).
mnt/	Çeşitli disk/disket/CDROM gibi çevre birimlerinin **mount** edilmesi için genel amaçlı bir boş dizin. (**mount point**)
pcfs/	MS-DOS formatlı disketlerin **mount** edilmesi için boş bir dizin.
quotas	Kullanıcıların disk kullanma kotalarının tanıtıldığı dosya.
sbin/	Marka ve donanıma bağımlı bazı sistem komutlarına ait dosyaların bulunduğu dizin.
sys/	Marka ve donanıma bağımlı UNIX modüllerinde değişiklik yapmak gerektiğinde kullanılacak dosya ve dizinlerin bulunduğu dizin.
tmp/	Uygulama programları ve kullanıcılar tarafından yaratılan geçici dosyalar için ayrılmış bir dizin. Bu dizinin içindeki dosya ve alt dizinler, sistemin her açılışında otomatik olarak silinir.
usr/	Uygulama programları, derleyiciler, standart yazılım kütüphaneleri gibi ortak kullanımda olan programların yerleştirildiği dizindir.
var/	Sistemde meydana gelen önemli olaylarla ilgili **log** dosyalarının (**syslog**, **messages** gibi) saklandığı dizin .
vmunix*	UNIX işletim sisteminin çekirdek programı ***(kernel)***. Sistem açıldığında belleğe ilk olarak bu program yüklenir.

/dev Dizini

Yukarıda listelenen dizinler arasında **/dev** özel açıklamalar gerektirmektedir. Bu dizinde yer alan dosyalar aslında tam anlamıyla birer dosya değildir. Dizin altında isimleri bulunmakla birlikte, diskte **hiç yer harcamazlar**. Bu özelliklerinden dolayı **/dev** dizini altında yer alan kayıtlara dosya değil; **düğüm** ***(node)*** adı verilir.

Bu düğümlerin her birinin birer **Major** ve birer **Minor** numaraları vardır. Bir UNIX komutu, ya da uygulama programı, **/dev** dizininde yer alan bir isim aracılığıyla bir donanım unsuruna ulaşmak istediğinde (örneğin, teybe bir kayıt işlemi için **/dev/rst0** düğümüne ulaştığında), UNIX, bu major-minor numaralar aracılığı ile çekirdek programın hangi modülünün harekete geçirileceğini anlayacak ve kontrolu, o donanım unsurunu tüm özellikleriyle tanıyıp denetleyebilen bir programa geçirecektir ***(device driver)***.

BSD

Tipik bir UNIX bilgisayarının **/dev** dizininde yüzlerce düğüm yer alır. Ben bu kitapta bunlardan sadece bir kaç tanesinden söz etmek istiyorum. Hem hepsini anlatmaya imkan ve gerek yok; hem de bu **/dev** dizininin yapısı gerek kullanılan UNIX'in tipine, gerekse bilgisayarın üreticisinin tercihlerine bağlı olarak büyük farklılıklar göstermektedir; ancak kullanım mantığı temelde hepsinde aynıdır. **/dev** dizininden söz ederken SUN Micro Systems firmasınca geliştirilen, BSD uyumlu SunOS 4.1.x UNIX'de kullanıldığı şekliyle söz edeceğim.

/dev Dizininde Yer Alan Bazı Düğümler

Düğüm	Tanımladığı Donanım Unsuru
/dev/console	Bilgisayarın ana ekranı (ya da ana terminali). Sistemde ortaya çıkan donanım sorunları ve diğer önemli olaylara ilişkin mesajlar bu donanım birimine gönderilir.
/dev/mem	Sistemin ana belleği
/dev/kbd	Sistem konsolunun klavyesi
/dev/mouse	Sistem konsolunun **mouse** birimi

/dev/null	Hiç bir yere bağlı olmayan, "kara delik" gibi bir düğüm. Buraya her şeyi kopyalayabilirsiniz. Hiç bir zaman dolmaz; ama buraya kopyalananlar da hiç bir zaman geri gelmez. Bir programın çıktılarını merak etmiyor ve hiç bir şekilde gerek duymuyorsanız, programı çalıştırırken, standart çıktı birimini bu düğüme yönlendirebilirsiniz. **komut > /dev/null** gibi
/dev/rst0	Sistemdeki ilk teyp birimi (**işi bitince kaseti başa saracak şekilde kullanım için**) (**r** : *rewind*)
/dev/rst1	Sistemdeki ikinci teyp birimi (**işi bitince kaseti başa saracak şekilde kullanım için**)
/dev/nrst0	Gene sistemdeki ilk teyp birimi; ancak bu kez; işi bitince kaseti başa sarmayacak şekilde kullanılması söz konusu. (**n** : *no rewind*). Bir kasete peşpeşe dosyalar kaydedileceği zaman ya da bir kasette peşpeşe kayıtlı dosyalar okunacağı zaman bu düğüm kullanılmalıdır.
/dev/sd0a	Sistemdeki ilk diskin **a** adı verilmiş bölümü ***(a partition)***
/dev/sd0b	Sistemdeki ilk diskin **b** adı verilmiş bölümü ***(b partition)***
/dev/sd1h	Sistemdeki **ikinci** diskin **h** adı verilmiş bölümü ***(h partition)***
/dev/sr0	Sistemdeki ilk CD-ROM sürücü okuyucusu
/dev/fd0	Sistemdeki ilk disket sürücü birimi (üzerinde UNIX veya MSDOS formatlı disket takılıyken).
/dev/rfd0	Sistemdeki ilk disket sürücü (üzerinde **formatsız** disket takılıyken) ***(r : raw device)***
/dev/ttya	Sistemdeki birinci seri arabirim (RS232)
/dev/ttyb	Sistemdeki ikinci seri arabirim (RS232)
/dev/bpp0	Sistemdeki ilk paralel yazıcı arabirimi (Centronics)

/dev/ttyp0	Ethernet üzerinden bağlanan terminal emülatörleri için sanal seri arabirimlerden ilki *(**pseudo tty port**).*
/dev/ttyp1	Ethernet üzerinden bağlanan terminal emülatörleri için sanal seri arabirimlerden **ikincisi** *(pseudo tty port).*
/dev/le0	Sistemin ilk Ethernet Arabirimi
/dev/le1	Sistemin ikinci Ethernet Arabirimi

SVR4

Yukarıdaki **/dev** düğümleri sadece birer örnektir. Her UNIX bilgisayarında aynen bulunmaları gerekmez. Örneğin, SVR4 UNIX'lerde diskleri tanımlayan düğümler

```
/dev/dsk/c0t0d0s3
```

gibi isimlerle anılırlar.

Gerek duydukça, sizin sisteminizde tanımlı olan /dev düğüm isimlerini ve nasıl kullanılacaklarını sistem yöneticinizden öğrenebilirsiniz. Eğer sistem yöneticisi sizseniz, bilgisayarınızın dökümantasyonunda yeterli açıklamaları bulacağınıza inanıyorum.

mount Komutu Üzerine Çeşitlemeler

Aslında, **mount**, daha çok sistem yöneticilerinin kullandığı bir komut olmakla birlikte, normal kullanıcıları da yakından ilgilendirmektedir.

Şöyle kısa bir hatırlatma yapmak gerekirse; **mount** komutu, dosya *sistemlerini (file systems)* birbirlerine bağlamakta kullanılır. Örneğin, bir bilgisayardaki ikinci disk sürücüsünü **/** altında bir dosya sistemine iliştirmek için; bir CD-ROM sürücüsüne takılı olan CD'yi okuyabilmek amacıyla **/dev/sr0** düğümünü **/** altında bir yerlere **mount** etmek için kullanılır.

BSD

mount komutunu detaylı olarak açıklamaya başlamadan önce, SunOS 4.1.x UNIX işletim sisteminin bakış açısıyla disklerin yapılarından söz etmek istiyorum. Aslında tüm UNIX'ler diskleri SunOS'inkine benzer bir yapıda görmek isterler.

UNIX bilgisayarlarında kullanılan diskler genellikle ***SCSI*** *(Small Computer Standard Interface)* arabirimine sahiptir. Bu arabirimin bir özelliği olarak her diskin 0 ile 7 arasında bir adresi olmalıdır. (Sizin bilgisayarınızda bu 8 adresten hangilerinin diskler için kullanılabileceğini sistem dökümantasyonuna bakarak öğrenebilirsiniz.)

Birden fazla diski olan bilgisayarlarda, disklerden bir tanesi **Sistem Diski** olarak tanımlanmalıdır. **Sistem diski** olarak seçilen disk, işletim sisteminizin özelliklerine bağlı olarak bir kaç bölüme (**partition**) ayrılmış olmalıdır. Örneğin **SunOS 4.1.1** de, sistem diski en az 3, en fazla 7 bölüme ayrılabilir. Her bir bölümün bir bölüm adı (ya da numarası) olmalıdır. Varsa, diğer disklerin bölümlere ayrılıp ayrılmaması, sistem yöneticisinin tercihine bırakılmıştır.

Disklerdeki bu bölümlerin, üzerlerinde bulundukları disklerin SCSI adresleri ve bölüm numaralarına göre verilmiş birer ismi olmalıdır ve bu isimler **/dev** dizininde birer **düğüm** *(node)* olarak yer almalıdır. Örneğin

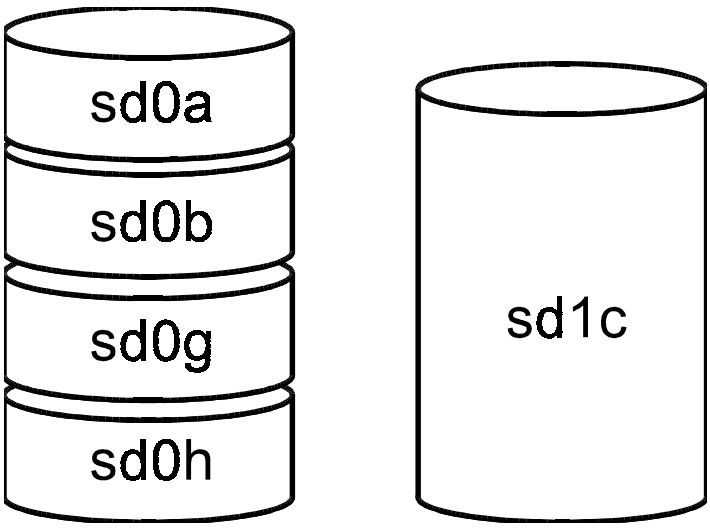
BSD

BSD UNIX'de DİSK İSİMLENDİRME SİSTEMİ	
/dev/sd0a	SCSI adresi, sistemdeki sıfırıncı (ilk) diske karşılık gelen diskin **a** isimli bölümü
/dev/sd1c	SCSI adresi, sistemdeki bir numaralı (ikinci) diske karşılık gelen diskin **c** isimli bölümü
/dev/sd3h	SCSI adresi, sistemdeki 3 numaralı diske karşılık gelen diskin **h** isimli bölümü

SVR4

SVR4 UNIX'de DİSK İSİMLENDİRME SİSTEMİ	
/dev/dsk/c0t0d0s0	Sistemdeki ilk disk kontrol birimine bağlı olan (**c0**), SCSI adresi 0 olan (**t0**), bu adresteki ilk sürücü olan (**d0**) diskin ilk bölümü (**s0**). c : channel t : target d : drive s : slice
/dev/dsk/c0t1d0s3	Bu da günün bilmecesi...

Şimdi, bu disk yapılandırma mantığı çerçevesinde, bilgisayarımızda iki adet disk birimi olduğunu ve aşağıdaki şekilde bölümlendirildiklerini varsayalım :



Bu durumda, disk bölümlerini BSD UNIX altında, **/dev** isimleri şöyle olur :

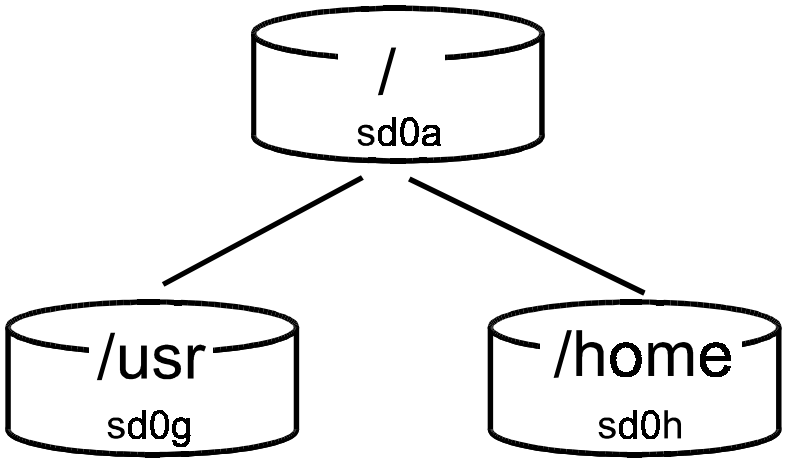
Sıfırıncı disk : **/dev/sd0a** Birinci disk :

```
/dev/sd1c
/dev/sd0b
/dev/sd0g
/dev/sd0h
```

Bu disk bölümlerinin UNIX altında kullanılabilmeleri için birer **mount** noktasına **mount** edilmeleri gerekir. Sistemin açılışı sırasında (**/etc/fstab**) dosyasında belirtildiği şekilde **mount** işlemleri otomatik olarak yapılır. SunOS 4.1.1 altında, genellikle

`/dev/sd0a` `/`
`/dev/sd0g` `/usr`
`/dev/sd0h` `/home` mount noktalarına **mount** edilirler.

`/dev/sd0b` özel bir şekilde **swap** alanı olarak kullanılır; o nedenle **mount** edilmez. (*swap alanı* : ana belleğin yetmediği durumlarda, sistemi çok yavaşlatma pahasına da olsa; belleğin uzantısıymış gibi kullanılan disk alanı).



Şimdi, **root** kullanıcının (**mount** komutunu sadece **root** kullanıcı kullanabilir) bilgisayarımızın ikinci diskini de devreye sokabilmek için neler yapması gerektiğini bir gözden geçirelim.

İkinci diskimiz (**sd1c**) ayrı bir dosya sistemine ***(file system)*** sahip olmalı; yani UNIX kurallarına uygun olarak formatlanmış ve **mkfs** *(make file system)* veya **newfs** *(new file system)* komutu kullanılarak üzerinde bir dosya sistemi yaratılmış olmalıdır. Bu formatlama ve dosya sistemi yaratma işi **sadece bir kez**, diskin bilgisayara ilk takıldığı zaman yapılmalıdır. Her iki işlem de (**format** ve **mkfs**) disk üzerindeki kayıtları tamamen silen işlemlerdir.

Eğer diskimiz hazırsa, **root** kullanıcı, bu diski, **/** dosya sisteminde hangi dizin altında görmek istediğine karar vermelidir. Örneğimizde bu dizin **/disk2** olsun. (Pekala **/usr/disk2** veya **/home/ugur2** de olabilirdi...)

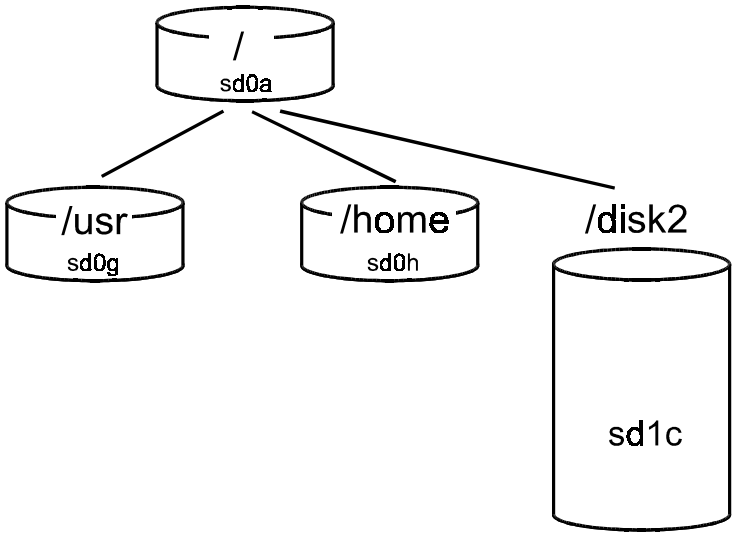
Sonra, **/** dizini (birinci diskin a bölümü : /dev/sd0a) altında **disk2** isimli **boş** bir dizin bulunduğuna emin olmalıdır. Eğer bu isimde bir dizin yoksa, **mkdir /disk2** komutu iş görecektir. (Hatırlatayım; bu işleri ancak **root** kullanıcı yapabilir).

Son olarak

mount noktası
(mount point)

```
# mount /dev/sd1c /disk2
```

komutuyla **mount** işlemini gerçekleştirir. Artık ikinci disk birimi bilgisayarın dosya yapısına entegre oldu ve tüm kullanıcılara **/** dizininin altında **disk2** isimli bir dizin olarak görünüyor. Normal koşullarda kullanıcıların, sistemde gördükleri dizinlerin birer disk birimi mi, yoksa disk bölümü mü, yoksa basit birer dizin mi olduklarını bilmeleri gerekmez. Ancak, sistem yöneticisinin tüm bu olup bitenlerden haberi olmalıdır.



Eğer kullandığınız bilgisayardaki disklerin (veya disk bölümlerinin nasıl bir düzen içinde ve nerelere **mount** edildiklerini öğrenmek isterseniz, **mount** komutunu parametresiz olarak kullanabilirsiniz.

```
abc:/home/ayfer> mount
/dev/sd0a on / type 4.2 (rw,quota)
/dev/sd0g on /usr type 4.2 (rw)
/dev/sd0h on /home type 4.2 (rw,quota)
abc:/home/ayfer>
```

Bu listeden öğrendiklerimiz şunlar :

a) Sistemde sadece bir disk var (ikinci disk varsa bile, **mount** edilmemiş). Çünkü disk isimlerinin hepsi **sd0** ile başlıyor.

b) İlk diskin **a** bölümü **/** olarak, **g** bölümü **/usr** olarak, **h** bölümüyse **/home** olarak **mount** edilmiş.

c) Tüm disk bölümlerindeki dosya yapılarının tipi 4.2 imiş. (SunOS tarafından dosya yapısı sürüm numarasıyla *(version)* ilgili olarak kullanılan özel bir kod).

d) Disk bölümlerinin hepsi okuma ve yazmaya açıkmış (**rw**). (Bu dosya sistemindeki dosya ve dizinlerin kullanıcı yetki kalıplarında belirtilen yetkiler saklı kalmak kaydıyla).

e) **/** ve **/home** dizinlerinin kullanımında kullanıcılara **kota** uygulanıyormuş. Yani, kullanıcılar kendilerine ayrılmış olan kotadan daha büyük disk alanı işgal edecek şekilde dosya açamayacaklar.

Bir bilgisayardaki disklerin ve disk bölümlerinin, sistemin açılışı sırasında otomatik olarak nerelere **mount** edileceğini sistem yöneticileri **/etc/fstab** dosyasında belirtirler. Böylece sistemin her açılışında, dosya sistemlerini tek tek **mount** etmekten kurtulurlar.

Tipik bir **/etc/fstab** dosyasında

```
/dev/sd0a  /      4.2 rw 1 1
/dev/sd0h  /home  4.2 rw 1 3
/dev/sd0g  /usr   4.2 rw 1 2
```

satırları bulunur.

SVR4 BSD UNIX'lerdeki **/etc/fstab** dosyasının görevini, SVR4 UNIX'lerde **/etc/vfstab** dosyası üstlenmiştir. **/etc/vfstab** dosyasındaki satır desenleri biraz farklı olmakla beraber, **/etc/fstab** ile aynı mantıkta düzenlenmişlerdir.

Bu açıklamalardan sonra, günlük hayatta rastlanan **mount** uygulamalarına örnekler vermek istiyorum.

- Bir disk bölümünü **salt oku** *(read only)* olarak **mount** etmek için :

BSD
```
# mount -r /dev/sd1c /home/disk2
```

SVR4
```
# mount -o ro /dev/dsk/c0t1d0s2 /home/disk2
```

mount komutunu kullanabilmek için **root** kullanıcı olmanız gerekir.

- **xyz** isimli bir başka bilgisayarın (doğal olarak, bizim bilgisayarımızla aynı bilgisayar ağında bulunmak kaydıyla) bir dizinini, kendi bilgisayarımızdaki bir dizine **mount** etmek için :

BSD
```
# mount -t nfs xyz:/home /home2
```

SVR4
```
# mount -F nfs xyx:/home /home2
```

- Üzerinde UNIX dosya sistemi olan bir disketi **mount** etmek için :

BSD
```
# mount /dev/fd0 /mnt
```

SVR4
```
# mount /dev/diskette /mnt
```

- Aynı disketin yazmaya karşı koruma deliği açıksa (disket yazmaya karşı korumalıysa) :

BSD
```
# mount -r /dev/fd0 /mnt
```

SVR4
```
# mount -o ro /dev/diskette /mnt
```

- MS-DOS formatlı bir disketi **mount** etmek için :

  BSD `# mount -t pcfs /dev/fd0 /pcfs`

  SVR4 `# mount -F pcfs /dev/diskette /pcfs`

- MS-DOS formatlı ve yazmaya karşı korumalı bir disketi **mount** etmek için :

  BSD `# mount -r -t pcfs /dev/fd0 /pcfs`

  SVR4 `# mount -F pcfs -o ro /dev/diskette /pcfs`

- **ISO 9660** standardında kaydedilmiş bir CD yi **mount** etmek için (**-r** : CD ler her zaman yazmaya karşı korumalıdır) :

  BSD `# mount -r /dev/sr0 /cdrom`

  SVR4 `# mount -o ro /dev/dsk/c0t6d0s2 /cdrom`

- **High Sierra File System** standardında kaydedilmiş bir CD yi **mount** etmek için :

  BSD `# mount -r -t hsfs /dev/sr0 /cdrom`

  SVR4 `# mount -F hsfs -r ro /dev/dsk/c0t6d0s2 /cdrom`

- **xyz** isimli bir başka bilgisayarın (doğal olarak, bizim bilgisayarımızla aynı bilgisayar ağında bulunmak kaydıyla) **/cdrom** dizinine **mount** edilmiş bir CD-ROM sürücüsünü, kendi bilgisayarımızdaki **/cdrom** dizinine **mount** etmek için :

  BSD `# mount -t nfs xyz:/cdrom /cdrom`

  SVR4 `# mount -F nfs xyz:/cdrom /cdrom`

- **mount** edilmiş bir dosya sistemini, bilgisayarın **/** dosya sisteminden ayırmak istediğinizde, (bu iş genellikle disket ve CD ler için anlamlıdır) :

BSD SVR4

```
# umount /dev/xxx                                    unmount
```

veya

```
# umount /mmm
```

komutu kullanılmalıdır. Burada **xxx** sürücünün /dev dizinindeki adı; **mmm** ise **mount noktasının** adıdır. Bir örnek vermek gerekirse, önceden

```
# mount -r -t hsfs /dev/sr0  /cdrom
```

komutuyla **mount** edilmiş bir CD yi **umount** etmek için

```
# umount /dev/sr0   veya
# umount /cdrom
```

komutlarını kullanabilirsiniz.

mount/umount komutlarının kullanımında dikkat edilmesi gereken noktalar :

mount ve **umount** komutlar□, genellikle tüm UNIX'lerde sadece **root** kullanıcı tarafından kullanılabilirler.

Bir donanım unsurunun **mount** edilebilmesi için, üzerinde geçerli bir dosya sistemi bulunmalıdır. Bu yüzden sadece

- formatlı diskler,
- CD'ler,
- formatlı disketler ve
- formatlı Magneto Optik diskler

mount edilebilir.

Teyp birimlerinin kullanılmasında **mount** komutunun kullanılması söz konusu değildir; çünkü teyp kasetleri üzerinde dosya sistemi bulunmaz.

Çalışma diziniz bir **mount** noktasında, ya da onun altında bir yerlerdeyse; o **mount** noktasına bağlı sürücüyü **umount** edemezsiniz. Örneğin, bir CD'yi **/cdrom** dizinine **mount** ettiyseniz ve bu CD üzerindeki işlerinizi daha kolay yapmak için **cd /cdrom/xyz** komutunu verdiyseniz (çalışma dizininiz **/cdrom/xyz** ise), **umount /cdrom** komutunu kullanamazsınız. (Kullanmak

istediğinizde **"Device Busy"** hata mesajıyla uyrarılırsınız; bindiğiniz dalı kesmenize izin verilmez.)

mount edilmiş CD, disket gibi takılıp çıkarılabilen birimleri **umount** etmeden kesinlikle yuvalarından çıkarmayınız. Zaten bu yüzden, bir çok UNIX bilgisayarında CD ve disket sürücülerin üzerinde CD ve disketi çıkarmak için kullanılan düğme bulunmaz; bulunsa bile içinde **mount** edilmiş medya bulunuyorsa, düğme çalışmaz. Bu tip sürücülerin içindeki medyayı çıkarabilmek için

```
# eject cdrom
# eject floppy
```

gibi komutlar kullanmanız gerekir. Bu komutlar, kullandığınız bilgisayara göre değişebilir.

Çalışan bir sistemde diskleri gerekmedikçe **umount** etmeyiniz.

Teypleri **mount** etmek gibi bir kavramın olmadığını aklınızdan çıkarmayınız. Elinde bir kasetle dolaşarak, **"Yahu bu teybi nasıl mount edeceğim?"** diye soran cahil bir UNIX kullanıcısı durumuna sakın düşmeyiniz.

İçinde dosya ve alt dizinler olan bir dizini, **mount noktası** olarak kullanırsanız (yani oraya CD, disket, disk gibi bir medya mount ederseniz), UNIX bu isteğinize karşı koymadan **mount** işlemini gerçekleştirecektir. Ama, medya **mount** edilmiş olarak kaldığı sürece, dizinin, eskiden altında bulunan dosya ve alt dizinlere ulaşamazsınız.

Senaryolar

Şimdi isterseniz MS-DOS formatlı bir disketin içindeki , adı ***.DAT** kalıbına uyan dosyaları, UNIX bilgisayarımızda **/home/ayfer** dizinine kopyalamak için neler yapılması gerektiğine bir göz atalım. (Senaryolar **SunOS 4.x.x** sahnesinde oynanacak şekilde hazırlanmıştır.)

Öncelikle **root** kullanıcı olmanız gerekir. Eğitim amaçlı bu senaryoya göre, **root** şifresini bildiğinizi varsayalım.

Sonra, disketi nereye **mount** edeceğinize karar vermalisiniz. Bence **/disket** dizini uygun.. Sonra **/disket** diye bir dizininin *(mount point)* bulunup bulunmadığını kontrol etmelisiniz.

```
# ls /disket
```

Olumsuz bir mesajla karşılaşmazsanız ve dizin boşsa devam edebilirsiniz. Eğer böyle bir dizin olmadığına ilişkin bir mesaj alırsanız

```
# mkdir /disket
```
komutuyla, dizini yaratabilirsiniz.

Ardından

```
# mount -t pcfs /dev/fd0 /disket
```

komutuyla disketi **mount** ediniz. Eğer disket yazmaya karşı korumalıysa, komutu

```
# mount -r -t pcfs /dev/fd0 /disket
```

şeklinde vermelisiniz. **mount** işlemi başarılı olursa, herhangi bir mesaj almadan sistem hazır işaretini *(prompt)* görürsünüz.

Şimdi kopyalama işine başlayabilirsiniz.

```
# cp  /disket/*.dat  /home/ayfer
```

Kopyalama bittiğinde,

```
# umount /disket   veya
# umount /dev/fd0
```

komutuyla disketi sistemden ayırıp,

```
# eject floppy
```
komutuyla yuvasından çıkarmalısınız.

Bir de kısaca, CD kullanımına örnek vereyim. Bu örnekteki en önemli detay, CD kayıt tipiyle ilgili. Bir kaç paragraf önce **ISO 9660** ve **High Sierra File System** adlı CD kayıt sistemlerinden söz ettim. Bir CD nin hangi sisteme göre kaydedildiğini her zaman kolayca anlayamazsınız Böyle durumlarda en kötü olasılıkla **mount** komutunun iki formunu da deneyerek işinizi görebilirsiniz.

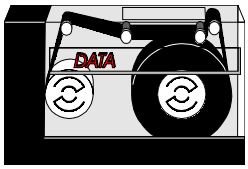
```
# ls /cdrom                  /cdrom dizini var mı?
# mount -r /dev/sr0 /cdrom
# cd /cdrom                  çalışma dizinini
                             /cdrom yaptık.
# cp *.dat  /home/ayfer      Kopyalama işleri...
# umount /cdrom              CD'yi umount etmek
                             için...
Device Busy                  Bindiğiniz dalı
                             kesemezsiniz..
# cd /home/ayfer       Çalışma dizinini değiştirdik
# umount /cdrom
# eject cdrom                CD'yi çıkarmak için
```

SunOS 5.x.x (SOLARIS 2.x) Kullanıcılarına...

Eğer bir SUN iş istasyonu kullanıyorsanız ve işletim sistemi sürümünüz SunOS 5.x.x ise (nam-ı diğer SOLARIS 2), CD ve disket **mount** işlerinde hayat sizin daha kolay (bazı durumlarda da daha zor) demektir.

Eğer **vold** daemon'u *(**volume manager daemon**)* çalışıyorsa (**"ps -e"** komutuyla çalışıp çalışmadığını öğrenebilirsiniz) CD veya disket **mount** etmek istediğinizde, CD veya disketi **yuvasına takmanız yeterli olacaktır**. Taktığınız medyanın formatı uygunsa otomatik olarak, önceden belirlenmiş **mount** noktalarından ilgili olanına **mount** edilecek ve kullanıma hazır olacaktır. İşi biten CD ve disketler için **eject** komutunu kullandığınızda, **umount** işi de otomatik olarak yapılacaktır. Ancak bu kolaylıkların kullanılabilmesi için **vold** programının arka planda çalışır durumda olması gerektiğini unutmayınız.

Teyp Kullanımı



UNIX dünyasının, disklerden sonraki en önemli manyetik çevre birimi teyp sürücüleridir. UNIX uygulamalarında, program ve veri dosyaları genellikle oldukça büyük olduğundan disket sürücüler pek kullanılmazlar. Hatta bir çok UNIX bilgisayarında disket sürücü bulunmaz bile.

Teyp sürücülerinin belli başlı üç kullanım alanı vardır. Doğal olarak ilk akla geleni **yedekleme**dir. Kullandığınız bilgisayar ne olursa olsun, yedeklemenin **çok önemli** olduğunu tekrarlamaya gerek yok sanırım.

İkinci önemli kullanım alanı, PC lerdeki disketler gibi yazılım ve **veri dağıtım ortamı**dır. Örneğin bir UNIX program paketi satın aldığınızda genellikle size bir teyp kaseti gönderilir. (Son yıllarda yazılım dağıtımı amacıyla CD kullanımı hızla yaygınlaşmaktadır, ama QIC standardındaki teyp kasetleri hala çok önemli).

Üçüncü kullanım alanıysa, bilgisayarın sistem diskinde bir sorun olduğunda ve **bilgisayarı işletim sistemi dağıtım teybinden açmak** gerektiğinde görülür. UNIX işletim sisteminin disketlere sığdırılması pek mümkün olmadığından, bilgisayarınızla birlikte ya bir teyp kaseti *(UNIX Boot Tape),* ya da bir *CD (UNIX Boot CD)* gelir. İşte işletim sistemi dağıtım teybiniz (ya da CD niz) budur; ÇOK İYİ KORUYUNUZ.

İçinde bulunduğumuz yıllarda (1995•) UNIX dünyasında kullanılan teyp birimlerinin en yaygın tipleri

QIC Teypler	150 - 525 Mega Byte
8 mm EXABYTE	2.5 - 7 Giga Byte
4 mm DAT teypler	2 - 16 Giga Byte

olarak sıralanabilir.

• *Bu kitabın uzun yıllar piyasada dolaşacağını umduğum için tarih belirttim; bakarsınız umduğum olur!*

QIC teyp sürücüler, yaklaşık VHS video kaset büyüklüğünde kasetler kullanırlar. Kaset kapasitelerinin mütevazi olması yanısıra; QIC en yaygın teyp standardıdır.

8 mm teyp sürücüleri, şekil olarak aynı 8 mm video kasetlerine benzeyen kasetler kullanırlar. Bazı 8 mm video kasetler bu sürücülerde kullanılabilmekle birlikte (daha ucuz olmalarından dolayı) bunu pek tavsiye etmem; veri kaydı için özel yapılmış kasetleri kullanmanız daha sağlıklı olacaktır.

4 mm teyplerin kullandığı kasetler neredeyse otellerde kullanılan küçücük kibrit kutularına benzer. Bu kasetleri görünce içine Giga Byte'larca kayıt sığdırılabildiğine inanmak oldukça güçtür. Kayıt teknolojisi açısından en sağlıklı ve güvenilir teypler olarak tanıtılmaktadırlar (ancak siz siz olun, bilgisayarlarla ilgili hiçbir konuda hiçbir şeye fazla güvenmeyin).

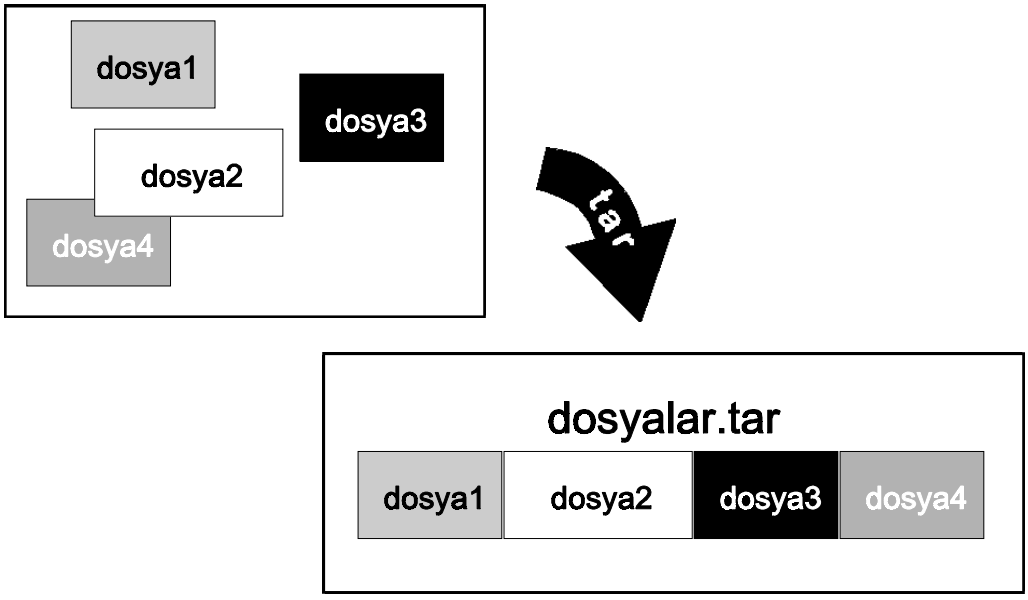
Hangi tipi kullanılırsa kullanılsın, UNIX açısından teyp teyptir. Hepsi için kullanılan komutlar aynıdır. (**tar**, **cpio**, **mt, dump**, **ufsdump** gibi komutlar).

> MS-DOS teyp kullanıcılarını çatlatacak bir haberim var. UNIX'de kullanılacak **teyplerin formatlanması gerekmemektedir**.

UNIX tar Dosyaları *(tape archive files)*

UNIX işletim sisteminde yedeklenecek veya teybe çekilip bir yerlere gönderilecek dosyalar genellikle önce bir **tar** dosyasına dönüştürülüp, sonra teybe çekilir. Bu işlem teybe dosya çekebilmek için uygulanması gereken bir kural değildir; sadece iyi yerleşmiş bir UNIX geleneğidir.

tar dosyaları, birden fazla dosyanın peşpeşe eklenerek tek bir dosyaya kopyalanması yoluyla elde edilir.



tar dosyalarının isimlerinin sonuna **.tar** eklenmesi bir UNIX geleneğidir. Böylece dosyayı alan kişi, o dosyanın bir **tar** paketi olduğunu; bir başka deyişle, **tar** formatı altında birleştirilmiş dosyalar içerdiğini anlayacaktır.

tar Komutu *tape archiver*

Çok sık kullanılan, bu nedenle de iyi öğrenilmesi gereken bir komuttur. Genel formu (basitçe) :

```
% tar [cxt][v]f tar-dosyası [dosyalar]
```

Bu karmaşık genel formu çözmeye çalışmayın; çeşitli seçenekleri birer örnekle açıklamaya çalışacağım. Ancak; **c**, **x, t** ve **v** harfleriyle belirtilen işlem kodlarını önce bir tanıyalım.

c **tar** dosyası yarat *(**create**)* anlamında
x **tar** dosyasını aç *(**extract**)* anlamında
t **tar** dosyasının içindeki dosyaların isimlerini listele *(**table of contents**)*

v **c, x** veya **t** emrini yerine getirirken, olup biteni ekrana listele demektir *(**verbose**).* Bu seçeneği kullanmazsanız **tar** komutu sessizce çalışır ve hangi dosyaları işlediğini ekrana listelemez. Bu seçeneği her zaman kullanmanızı öneririm.

Şimdi örneklere geçelim... Diyelim ki, **home** dizinizde

dosya1 (10 KByte)
dosya2 (12 KByte)
dosya3 (20 Kbyte)

isimli 3 dosya var.

```
% tar  cvf  dosyalar.tar  dosya*                create
% tar  cvf  dosyalar.tar  dosya?
% tar  cvf  dosyalar.tar  dosya1  dosya2  dosya3
```

komutlarından herhangi biriyle bu üç dosyayı birleştirip **dosyalar.tar** isimli dördüncü bir dosya oluşturabilirsiniz.

```
abc:/home/ayfer> ls -l dos*
-rw-------  1 ayfer      10240 May 12 16:53 dosya1
-rw-------  1 ayfer      12288 May 12 16:53 dosya2
-rw-------  1 ayfer      20480 May 12 16:53 dosya3
abc:/home/ayfer> tar cvf dosyalar.tar dosya*
a dosya1 20 blocks
a dosya2 24 blocks
a dosya3 40 blocks
abc:/home/ayfer>
```

Bu komuttan sonra **home** dizininizdeki adı **dos** ile başlayan dosyaları listelediğinizde

Yeni yaratılmış olan tar dosyası

```
abc:/home/ayfer> ls -l dos*
-rw-------  1 ayfer      10240 May 12 16:53 dosya1
-rw-------  1 ayfer      12288 May 12 16:53 dosya2
-rw-------  1 ayfer      20480 May 12 16:53 dosya3
-rw-r--r--  1 ayfer      49152 May 12 16:53 dosyalar.tar
abc:/home/ayfer>
```

Diskinizde bulunan bir **tar** dosyasının içinde yer alan dosyaların isim listesini görmek istediğinizde

```
% tar tvf dosyalar.tar                    table of contents
```

komutunu vermeniz yeterlidir.

```
abc:/home/ayfer> tar tvf dosyalar.tar
rw-------8700/33  10240 May 12 16:53 1995 dosya1
rw-------8700/33  12288 May 12 16:53 1995 dosya2
rw-------8700/33  20480 May 12 16:53 1995 dosya3
abc:/home/ayfer>
```

Bir **tar** dosyasını bir başka bilgisayara taşıyıp, orada yeniden açmak istediğinizdeyse

```
% tar xvf dosyalar.tar                    extract
```

komutunu kullanmalısınız.

Başka bilgisayar, başka dizin...

```
xyz:/home/hasan> tar xvf dosyalar.tar
x dosya1, 10240 bytes, 20 tape blocks
x dosya2, 12288 bytes, 24 tape blocks
x dosya3, 20480 bytes, 40 tape blocks
xyz:/home/hasan> ls -l dos*
-rw-------  1 ayfer      10240 May 12 16:53 dosya1
-rw-------  1 ayfer      12288 May 12 16:53 dosya2
-rw-------  1 ayfer      20480 May 12 16:53 dosya3
xyz:/home/hasan>
```

"Peki...Teyp bunun neresinde?"

dediğinizi duyar gibi oluyorum. **tar** dosyası adı yerine bir teyp sürücüsünün **/dev** dizinindeki adını verirseniz, **tar** komutu teyp üzerinde çalışacaktır. Örneğin

```
% tar cvf /dev/rst0 dosya*                    create
```

komutunu verirseniz, **dosya1**, **dosya2** ve **dosya3** dosyalarının birleştirilmesiyle elde edilecek **tar** dosyası teybe kaydedilecektir. Dikkat ederseniz, bu durumda **tar** dosyasıyla ilgili bir isim vermiyoruz: **dosyalar.tar** veya benzeri bir isim vermedik. **UNIX işletim sisteminde teypte yer alan tar dosyalarının isimleri olmaz.** Teyp sürücüsünün adı ve teyp kasetinin okuyucu kafa karşısındaki pozisyonu teyp dosyalarını tanımlamak için yeterlidir.

Kasetinin başında bir **tar** dosyası bulunan teypden, bu **tar** dosyasının içindeki dosyaları çalışma dizininize indirmek istediğinizde

```
% tar xvf /dev/rst0                           extract
```

komutunu kullanabilirsiniz.

Aynı mantıkla, teybin başındaki **tar** dosyasının içindekilerin listesini görmek istiyorsanız

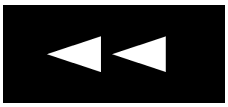
```
% tar tvf /dev/rst0                       table of contents
```

komutu işinizi görecektir.

Örnekler hep BSD UNIX için verilmiştir. SVR4 UNIX kullanılması durumunda değişecek tek şey, teyp sürücüsünün **/dev** dizinindeki adı olacaktır.

```
% tar cvf /dev/rmt/0 dosya*
% tar tvf /dev/rmt/1            gibi...
```

Tahmin etmiş olacağınız gibi teyp kasetinin, okuyucu kafa karşısındaki pozisyonu son derece önemlidir. **tar** komutunda **/dev** adı olarak

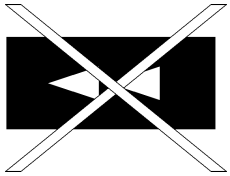


İşi bitince başa saran teyp sürücüleri

BSD	SVR4	Sürücü
/dev/rst0	/dev/rmt/0	Birinci teyp sürücü
/dev/rst1	/dev/rmt/1	İkinci teyp sürücü

kullandığınız sürece, **tar** komutu işini bitirdiğinde, teyp sürücüsü otomatik olarak kasetin başına saracaktır. Kaseti teyp sürücüsünden çıkarıp geri takmanız, aynı şekilde kasetin başla sarılmasına neden olacaktır.

Kasetinizde ardarda birden fazla **tar** dosyası varsa ve siz bu iki kaydı da ayrı ayrı diskinize indirmek istiyorsanız, birinci **tar** komutunuzda kullanacağınız **/dev** adında kasetin başa sarılmamasını istediğinizi belirtmek zorundasınız.



İşi bitince başa sarmayan teyp sürücüleri

BSD	SVR4	Sürücü
/dev/nrst0	/dev/rmt/0n	Birinci teyp sürücü
/dev/nrst1	/dev/rmt/1n	İkinci teyp sürücü

```
% tar xvf /dev/nrst0          /dev adındaki n harfine dikkat!
```

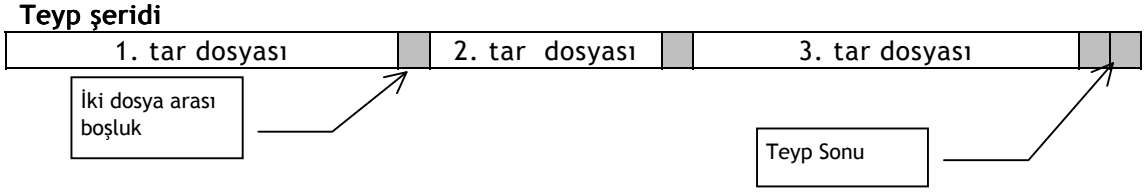
komutunu kullandığınızda (**nrst0 : *no rewind SCSI tape 0***), **tar** komutu işini bitirince, teyp kaseti başa sarılmayacaktır. Böylece ikinci **tar** dosyasını da diske indirmeniz mümkün olacaktır. Ancak, burada küçük bir sorun var. Teybin okuyucu kafası şu anda ikinci **tar** dosyasının başında değil de, iki **tar** dosyası arasındaki boşlukta duruyor. O nedenle ikinci **tar** dosyasını işlemek için vereceğiniz **tar** komutu bir hata mesajına neden olacaktır.

```
tar: /dev/rst0: I/O error
```

Bu hata mesajını aldığınızda okuyucu kafa bu boşluğu atlamış olacağından son komutu tekrarlarsanız, ikinci **tar** dosyasını işlemeye başlamış olursunuz; tabii ikinci **tar** dosyası gerçekten varsa... Eğer tekrar aynı hata komutunu alırsanız, kasetin gerisi boş demektir. Daha fazla uğraşmanız anlamsızdır.

Şimdi bir kaç senaryo oynayalım:

1. Teybinizin içinde 3 adet **tar** dosyası var ve siz sadece **üçüncüyü** diske indirmek istiyorsunuz. **Ne yapmalısınız?**



Kaseti teybe taktığınızda otomatik olarak başa sarıldığını dikkate alarak, önce ilk **tar** dosyasını atlamalısınız.

Dosya atlamanın en kolay yolu, **tar** komutunu **t** seçeneği ile kullanmak olduğundan

```
% tar tvf /dev/nrst0
```

komutuyla ilk dosyanın içindekilerin listesini alınız. Bu liste anlamsız olmakla birlikte teybin çalıştığını göstermesi açısından yararlıdır. Bu **tar** komutunda boş bulunup **/dev/rst0** kullanırsanız listeyi boşuna almış olursunuz. Çünkü **tar** komutu istediğiniz listeyi (**t** seçeneği) verdikten sonra kaseti tekrar başa saracaktır.

Şimdi okuyucu kafa birinci ve ikinci dosyalar arasındaki boşlukta duruyor olmalı. Anlamsız da olsa bir **tar** komutuyla bu boşluğu atlamalısınız (biliyorsunuz, hata mesajı gelecek).

```
% tar tvf /dev/nrst0
tar: /dev/rst0: I/O error
```

Şimdi teybin okuyucu kafası ikinci **tar** dosyasının başında... Bu dosyayı ve arkasındaki boşluğu da atlamak için

```
% !!        İkinci tar dosyasını atlamak için
% !!        İkinci ve üçüncü tar dosyalarının
            arasındaki boşluğu atlamak için
```

komutlarını ardarda verebilirsiniz. (**!!** komutu ancak **csh** kabuk programını kullanıyorsanız ve **history** değişkeninin bir değeri varsa anlamlıdır; eğer **sh** kabuğu kullanıyorsanız son komutu paşa paşa tekrar yazmak zorundasınız).

Şimdi kafa üçüncü **tar** dosyasının başına gelmiş olmalı. Artık normal **tar** komutunuzu (***extract*** seçeneği ile) verebilirsiniz.

```
% tar xvf /dev/rst0
```

***tar** komutu bittiğinde işimiz de bitmiş olacağından, artık kaset başa sarılabilir.*

2. **home** dizinizdeki adı **a** ile başlayan dosyaları ilk **tar** dosyası olarak, adı **b** ile başlayan dosyaları ikinci ve adı **c** ile başlayan dosyaları da üçüncü **tar** dosyası olarak teybe kaydetmek istiyorsunuz. **Ne yapmalısınız?**

Teyp şeridi

a ile başlayan dosyaların **tar** dosyası		**b** ile başalayan dosyalar		**c** ile başlayan dosyaların **tar** dosyası	

İlk önce kasetin başına adı **a** ile başlayan dosyaların **tar** dosyasını yaratmalısınız.

```
% tar cvf /dev/nrst0 a*
```

Hemen ardından ikinci ve üçüncü **tar** dosyalarını yaratabilirsiniz.. Dosyalar arasındaki boşlukların yaratılması sizin sorumluluğunuzda olmayacaktır.

```
% tar cvf /dev/nrst0 b*
% tar cvf /dev/nrst0 c*
```

3. **home** dizinizdeki adı **a, b** ve **c** ile başlayan dosyaları tek bir **tar** dosyası olarak teybe kaydetmek istiyorsunuz. **Ne yapmalısınız?**

Teyp şeridi

a, b ve c. ile başlayan dosyalar		

Bu işi tek komutta yapmalısınız. (Tek bir **tar** dosyası elde edebilmek için)..

```
% tar cvf /dev/nrst0 a* b* c*
```

4. Kasetinizde adı **a, b** ve **c** ile başlayan dosyalardan oluşan tek bir **tar** dosyası var. Bu dosyalar arasından **sadece** adı **b1** ile başlayanları **indirmek** istiyorsunuz. **Ne yapmalısınız?**

Teyp şeridi

a, b ve **c** ile başlayan dosyalar		

tar komutunu **x** seçeneği ile kulllanacağınız kesin; ancak dosya adlarıyla ilgili tercihinizi belirtirken önemli bir püf noktasına dikkat etmelisiniz. İçgüdüleriniz

```
% tar xvf /dev/rst0  b1*
```
HATALI !

şeklinde bir komut yazmanızı söylüyor, değilmi?

Maalesef içgüdünüz sizi yanıltıyor ! Vermeniz gereken komut

```
% tar xvf /dev/rst0 "b1*"
```
DOĞRU !

Hatanın ne olduğunu ilk bakışta görememeniz oldukça normal. Şimdi kabuklar hakkında öğrendiklerinizi bir tazeleyelim.

Kabuk programları, klavyeden girilen (ya da kabuk programlarından gelen) komut satırlarını irdeleyip, varsa parametreleri çözüp yerlerine yerleştirmeye çalışacaktır. Kabuk programı **hatalı** komuttaki **b1*** karakterlerini görünce, komutun verildiği andaki çalışma dizininde bulunan dosyalar arasında (**Dikkat!** teypteki değil, çalışma dizininde bulunan dosyalar arasında) adı **b1** ile başlayanları bulup onların isimlerini komut satırında **b1*** in yerine yerleştirip, komutu öyle çalıştıracaktır.

Eğer çalışma dizininde adı **b1** ile başlayan dosya yoksa, komut satırınız

```
tar xvf /dev/rst0
```

(dosya adına ilişkin bir parametre yok) olarak yazılmış kabul edilip **tar** komutu çalıştırılacak ve çok doğal olarak kasetteki tüm dosyaları indirecektir.

Bu sorunu halletmek için, kabuk programına **b1*** karakterlerini çözümlememesi gerektiğini bildirmek gerekir. Bunun için **tar** komutunda **b1*** karakterlerini tırnak içine almalıyız. Böylece **"b1*"** kalıbı kabuk programı tarafından değil, **tar** programı tarafından çözümlenecektir ve teypteki dosyalar arasında yanlızca adı bu kalıba uyan dosyalar diske indirilmiş olacaktır.

tar Komutunu kullanırken dikkat edilmesi gereken noktalar

tar komutunu kullanırken çok tekrarlanan bazı hatalara dikkatinizi çekmek istiyorum.

1. **tar** komutu, **tar** dosyası yaratırken **dosya ve dizin ayırımı** yapmaz. Parametre olarak verilen dosya kalıbına uyan her şey **tar** dosyasının içine kopyalanır. Dizinler ve alt dizinleri buna dahildir.

2. **"tar cvf /dev/nrst0 *"** komutu (çalışma dizinindeki her şeyi teybe **tar** dosyası olarak çek anlamında) aslında her şeyi çekmeyecektir. Komutun bu şekilde kullanılması durumunda adı **.** (nokta) ile başlayan dosyalar **tar** dosyasına dahil edilmeyecektir. Adları noktayla başlayan dosyaları da kasete çekmek istiyorsanız;

```
"tar cvf /dev/nrst0 .login .cshrc .X* *"
```

gibi bir komut kullanmanız gerekir.

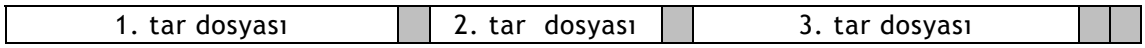
```
"tar cvf /dev/nrst0 .* *"
```

şeklindeki bir komut tehlikeli olabilir. Bulunduğunuz dizinde, normal dosyalar ve dizinler yanısıra yer alan **.** ve **..** isimli iki özel dizin vardır. Bunlardan .. bir üst düzey dizini gösterdiği için; verdiğiniz komut, bir üst düsey dizini de teybe çekmek istediğiniz şeklinde yorumlanabilir.

3. Bir çok UNIX kitabında açıklamasını göreceğiniz **-r** parametresi, ilk bakışta anlaşılacağı gibi teybin sonuna yeni bir **tar** dosyası eklemek için değil, teyp yazma kafasının üzerinde bulunduğu **tar** dosyasının sonuna eklemeler yapmak için kullanılır. Teyplerde bu parametrenin kullanılması, eğer varsa, söz konusu

tar dosyasının ardından gelen dosyaların tamamen kaybedilmeleriyle sonuçlanacaktır.

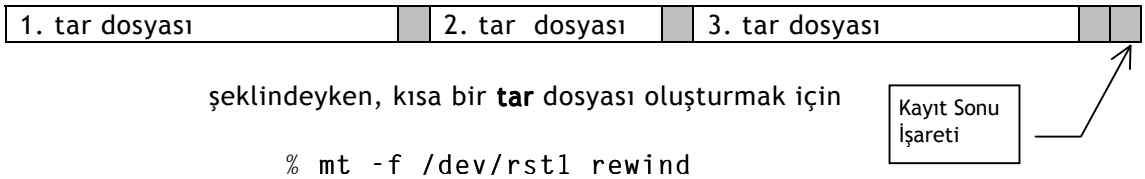
4. Sık tekrarlanan bir başka hatayı ise bir örnekle açıklamak istiyorum. Diyelim ki elimizde aşağıdaki şekilde kaydedilmiş bir teyp kaseti var :



Kasetin sonuna ekleme yapmayı planlarken yanlışlıkla, **tar cvf** komutunu kaset teybin başındayken verdiniz. Anında hatanızı farkettiniz ve hemen işinizi kestiniz. (Ctrl-C) ya da hemen teyp birimini kapattınız. Birinci **tar** dosyasının bozulduğuna hiç şüphe yok ama 2. ve 3. **tar** dosyalarını **kurtardığınızı sanıyorsanız YANILIYORSUNUZ**. Yaptığınız hata kasetin tümündeki kayıtların kaybolmasına yol açtı.

Aynı mantıkla, birinci **tar** dosyasından daha kısa bile olsa, birinci dosya üzerine yapacağınız bir kayıt, kasetin bu kayıttan sonrasını kullanılamaz duruma getirecektir. Görsel olarak anlatmak gerekirse...

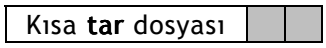
Kasetinizdeki kayıt yapısı



şeklindeyken, kısa bir **tar** dosyası oluşturmak için

```
% mt -f /dev/rst1 rewind
% tar cvf /dev/nrst1 kisa-dosyalar
```

komutlarını verirseniz yeni kaset yapınız



şekline dönüşür. (**Üzgünüm...**)

5. **tar** programı, daha önce bir **tar** dosyası içine paketlenmiş olan dosyaları geri çıkarırken (***extract*** ederken) bu dosyaları alındıkları dizinlere yerleştirmeye çalışır. Örneğin

```
% tar cvf /dev/nrst1 /home/ayfer/*
```

şeklinde bir komutla **tar**'lanmış dosyaları bir başka bilgisayarda

```
% tar xvf /dev/nrst0
```

komutuyla geri indirmek istediğinizde, dosyalar yeni bilgisayarda **/home/ayfer** diye bir dizinin altına indirilecek; böyle bir dizin yoksa yaratılmaya çalışılacaktır. Bu tip sıkıntıları önlemek için **tar** programıyla dosya çekerken

```
% cd /home/ayfer
% tar cvf /dev/nrst1 ./*
```

şeklinde göreceli dizin tanımları kullanmanızı öneririm. **./*** kalıbı; "bulunduğum dizindeki herşey" anlamına gelmektedir.

6. **tar** programı, dosya ***extract*** ederken diskte aynı isimde bir dosya olsa bile **uyarmadan** üzerine yeni dosyayı indirir.

7. **tar** programı, **tar** dosyası yaratırken bağlantılı dosyaları ***(link)*** kopyalamaz. Eğer bu tip dosyaların kopyalanmasını özellikle istiyorsanız, komutunuzda **-h** seçeneğini belirtmeniz gerekir. **tar cvfh /dev/nrst1 ./*** gibi... (SVR4 UNIX'de **-L**).

8. **tar** programı, sadece teyp kullanımıyla sınırlı değildir; disketlerle birlikte de aynı rahatlıkla kullanılabilir. Ancak, normal **tar** programları medya dolduğunda (teyp ya da diskette boş yer kalmadığında) hata mesajı vererek dururlar ***(Abort).*** Bir medya dolduğunda diğer bir boş medya isteyen **tar** programlarının adı genellikle **bar**'dır. SVR4 UNIX'lerde **tar** komutları birden fazla medya üzerine kayıt yapabilecek şekilde geliştirilmiştir. Sizin sisteminizde hangisini kullanacağınıza karar vermek için sistem yöneticinize danışınız.

9. Bazı **tar** programları, **tar** dosyası yaratırken bir yandan da veri sıkıştırması yapabilirler. Bu özellik standart olmadığından, bir başka bilgisayara taşımak üzere **tar** dosyası (**tar** teybi) yaratırken **tar** programlarının bu sıkıştırma özelliğini kullanmayınız. Eğer mutlaka sıkıştırma yapmanız gerekiyorsa, dosyalarınızı önce diskte bir dosyaya **tar**'layın; sonra bu dosyayı standart UNIX **compress** komutuyla sıkıştırın, ondan sonra bu dosyayı tekrar **tar** komutuyla teybe (ya da diskete) kaydedin. Bu yöntemi kullandığınızı teyp ya da disketin etiketi üzerinde açıklayıcı bir not olarak iliştirip diğer bilgisayarda kullanılmak üzere yollayın. Örnek olarak :

```
% tar cvf dosyalar.tar dosya*
% compress dosyalar.tar
% tar cvf /dev/fd0 dosyalar.tar.Z
```

compress programı sıkıştırdığı dosyanın adının sonuna **.Z** ekler.

Etkileyici Bir UNIX Gösterisi

Şimdi MS-DOS kullanıcılarını **çatlatacak** bir UNIX gösterisi yapmak; bir başka deyişle **hava atmak** istiyorum. Gösterinin komutun nasıl çalıştığını anlamazsanız fazla dert etmeyin ama günün birinde bu komutun sırrını çözecek kadar UNIX öğrenmeye de azmedin lütfen. Aslında olay basit, ama komutu karmaşık. UNIX alışkanlığınız arttıkça bu tip komutları sizin de kullanacağınıza eminim.

Yapmak istediğimiz iş şu :

Bilgisayar ağı üzerindeki **abc** bilgisayarının önünde oturarak, dosyalarımızı **xyz** makinasının üzerindeki **rst1** teyp ünitesinde takılı olan kasete kopyalayacağız.

```
abc:/> tar cvfb - 20 dosya* | rsh xyz dd of=/dev/rst0 obs=20b
```

Neler olup bittiğini merak ettiyseniz, açıklayayım:

abc bilgisayarında **tar** programını başlatıyoruz, adı **dosya*** kalıbına uyan dosyaları bir **tar** dosyasına dönüştürüyoruz; ancak bu **tar** dosyasını fiziksel bir sürücü yerine standart çıktıya yönlendiriyoruz (**tar** dosyası adı olarak **-** işareti). Aynı anda; adı **xyz** olan uzaktaki bilgisayarda **dd** *(**device to device copy**)* programını başlatıyoruz (**rsh** : *remote shell* komutu). **abc** makinasının standart çıktısındaki kayıtları (**tar** dosyamızı), **xyz** makinasında çalışmakta olan **dd** programına girdi olarak ***pipe*** edip, **dd** programının çıktı dosyası olarak tanımlanmış olan **/dev/rst0** sürücüsüne kaydedilmesini sağlıyoruz. Bu arada, bilgisayarlar arası transferi hızlandırmak amacıyla da kayıtlarımızı 20'şer 20'şer blokluyoruz.

dd komutunu merak ettiyseniz, bir **/dev** biriminden bir başka **/dev** birimine veri kopyalamak için kullanılan programdır. Örneğin teypten teybe, disketten teybe kopyalamalarda kullanılabilir. Daha fazla bilgi almak için **man dd** komutunu kullanınız.

dump (BSD UNIX) ve **ufsdump** (SVR4) genellikle sadece sistem yöneticilerini ilgilendiren ve diskleri teyplere yedeklemek için kullanılan komutlardır. Bu komutları kitabın "Sistem Yöneticine" başlıklı bölümlerinde anlatacağım.

mt Komutu *(magnetic tape controls)*

Teyp sürücülerindeki kasetlerin okuma/yazma kafası karşısındaki pozisyonunu ayarlamak için kullanılan bir komuttur.

```
% mt -f /dev/nrst0  hareket-kodu
```

hareket-kodu olarak

```
rewind      (kısaca rew de kullanılabilir)
fsf         (forward space file)
fsf n
eof         (end of file)
retension
erase
```

anahtar sözcükleri kullanılabilir.

mt hareket-kodu	Görevi
rewind	Kaseti başa sarar
fsf	Kaseti bir dosya veya dosyalar arası boşluk kadar ileri sarar
fsf n	Kaseti **n** tane dosya ve dosyalar arası boşluk kadar ileri sarar. (Örneğin, kasetinizi 3 dosya ileri atlatmak istiyorsanız **fsf 6** kullanmalısınız. (dosya+boşluk+dosya+boşluk+dosya+boşluk)
eof	Kasetin sonuna kadar ileri sarar. Kaset sonlarına dosya eklemek istediğinizde önce bu komutla kaseti sona sarmalısınız.
retension	Kaseti sona kadar ileri ve sonra tekrar başa sarar; böylece kasetin manyetik şeridinin gerginliği düzenlenmiş olur.
erase	Bir kasetin içindeki tüm kayıtları siler.

Eğer **mt** komutunu **/dev/rst0** gibi ***"no rewind device (nrst0)"*** belirtmeden kullanarak kaseti 4 dosya ileri ya da kaset sonuna sararsanız pek anlamlı bir iş yapmış sayılmazsınız. Çünkü, **rst0** şeklinde verilen **/dev** adları, teyp sürücüsünün işi bittiğinde (örneğin 4 dosya ileri sardıktan sonra) kaseti başa saracaktır.

Eğer kullanacağınız **mt** hareket-kodu **rewind**, **erase** ya da **retension** ise sorun yok. İş bitince zaten kaset başa sarılı durumda kalacaktır.

cpio Komutu *(copy input to output)*

tar kadar yaygın olarak kullanılmamakla birlikte, önemli teyp programlarındandır. Kullanılmaları **tar**'a göre daha fazla dikkat ve uzmanlık gerektirir. En önemli özelliği, teybe kopyalanacak dosya ve dizinlerin isimlerini standart giriş biriminden kabul etmesidir. Bu sayede, teybe kopyalanacak dosyaların isimleri başka programlar tarafından üretilen bir isim listesinden alınabilir. Çok karışık geldiyse bu komutla ilgili bölümü atlayabilirsiniz.

Meraklı okuyucularla devam edelim....

Aşağıdaki komut satırı, **find** ve **cpio** programlarını birlikte kullanarak son 10 gün içinde değişikliğe uğramış olan dosyaları teybe çekecektir. (İster inanın ister inanmayın).

```
% find . -type f -mtime -10 -print | cpio -o > /dev/nrst0
```

Komut satırının mantığı şu :

`.` (nokta)	Aramaya, bulunduğum dizinden başla
`-type f`	Tipi "dosya *(file)*" olan kayıtlarla ilgileniyorum (dizinleri dikkate alma)
`-mtime -10`	Son 10 gün içinde değişmiş olanları ayıkla
`-print`	Bulduklarının adlarını listele
`| cpio`	Bu listeyi ekrana değil de **cpio** programına girdi olarak gönder (*pipe* kuruyoruz)
`-o /dev/nrst0`	cpio programı çıkışını **nrst0** teybine yazsın.

cpio programıyla elde edilen teyp kayıtları sadece **cpio** programı ile geri yüklenebilirler.

Yukarıdaki komutla yedeklenmiş dosyaları teypden geri yüklemeniz gerekirse

```
% cpio -idmuv < /dev/rst0
```

Eğer sadece bir dosyayı indirmek isterseniz (tabii ki daha önce **cpio** ile teybe kaydedilmiş olmak şartıyla)

```
% cpio -idmuv istenen_dosya < /dev/rst0
```

cpio komutuyla birlikte kullandığım garip **idmuv** seçeneklerinin ne olduğunu merak ettiyseniz **man** komutu size yardımcı olabilir.

Aman Ha! Sakın Ha! Elinizde bir teyp kasetiyle ortalıkta dolaşıp

"Yahu! Bu kaseti nasıl **mount** edeceğim?"

diye dolaşmayın. Rezil olursunuz! UNIX'de teypler "**mount**" edilmez! (Teyplerle ilgili olarak kullanılan "**mt**" komutu "mount" değil, *"magnetic tape control"* anlamındadır.) UNIX'de bir çevre biriminin "mount" edilebilmesi için, çevre birimi üzerinde önceden bir dosya yapısı *(file system)* oluşturulmuş olması gerekir; bu da şimdilik yalnızca disk, disket, CDROM ve Magneto Optik diskler gibi **doğrudan erişimli** çevre birimlerinde *(Direct Access Storage Device)* mümkündür. Teypler; şerit yapılarından dolayı sıradan erişimli çevre birimleridir *(Sequential Access Storage Device).*

Kullanışlı UNIX Komutları

UNIX işletim sisteminde, **/bin** ve-veya **/usr/bin**, **/usr/sbin**, **/usr/5bin** gibi dizinlerinde yüzlerce program bulunur. Bunların önemli bir kısmının ne işe yaradığını bile anlamak meseledir. Ancak zamanla; çıraklık yapa yapa ve başınızı vura vura bu programlarının çoğunu anlayacak, öğrenecek ve kullanacaksınız. Başlangıçta bilmeniz gerekebilecek bazı komutları seçip açıklamak istiyorum. Bu seçimimde kullandığım kriter basit : seyrek de olsa kendi kullandığım komutlardan, herhangi bir sırayı dikkate almaksızın söz edeceğim. Bu arada, bazı tekrarlar olacak ama sizi rahatsız edeceğini sanmıyorum.

% cat *(catenate)*

En temel kopyalama programı. Standart girişi standart çıkışa kopyalar.

Eğer parametresiz olarak başlatırsanız garip bir duruma düşersiniz. **cat** yazıp Enter tuşuna basar basmaz imleç bir alt satıra iner ve oradan başlayarak klavyeden her yazdığınız satırı geri ekrana tekrarlar. Aslında program tanımına uygun davranıyor; yani standart girişi (klavyeyi) standart çıkışa (ekrana) kopyalıyor. Bu durum tamamen yararsız olduğundan, kurtulmak için, imleç satır başındayken Ctrl-D tuşuna basmanız gerekir. (Sadece bir kere; aksi takdirde, fazladan basacağınız Ctrl-D karakterleri kabuk programınız tarafından "dosya sonu" olarak yorumlanabilir ve daha fazla komut vermek istemediğiniz sonucuna varılarak kabuk programınız öldürülür; bir başka deyişle istemeden **logout** etmiş olursunuz.)

Parametreli olarak kullanıldığında, parametresinde verilen dosya (ya da dosyaları) standart çıkış birimi olan ekrana listeler. (MS-DOS'daki TYPE komutu gibi).

Buraya kadar olan kısmı zaten biliyordunuz. Peki aşağıdaki **cat** komutu formlarına ne dersiniz?

```
% cat dosya1 dosya2 dosya3 > genel_dosya
```
dosya1, dosya2 ve dosya3 dosyalarını bu sırayla peşpeşe ekleyip bir genel_dosya dosyası oluşturur.

```
% cat dosya1 dosya2 dosya3 > genel_dosya &
```
Aynı işi arka planda çalışarak yapar.

```
% cat buyuk_dosya > /dev/null
```
buyuk_dosya isimli dosyayı "kara deliğe" yani "hiç bir yere" kopyalar. Böyle bir komutla, dosyanın başından sonuna kadar okunabilir olup olmadığını denemiş olursunuz.

```
% cat liste > /dev/bpp0
```

liste isimli programı doğrudan yazıcı arabirimine kopyalar. Yazıcı ile ilgili **daemon**'lar çalışmıyor olsa bile bu yöntemle yazıcıdan döküm alabilirsiniz. Ancak, **root** kullanıcı olmanız gerekir.

```
% cat < /dev/ttyb
```

İkinci seri arabirime bağlı olan terminalin klavyesinden yazılan her şeyi sizin ekranınıza kopyalar. Bu komut ancak mesai arkadaşlarını deli etmek isteyen **root** kullanıcılar tarafından verilebilir.

```
% compress dosya_adi
```

Kısa dönemde gerekli olmayacak ya da teybe çekilecek disk dosyalarının daha az yer işgal etmelerini sağlamak amacıyla kullanılan sıkıştırma programıdır.

```
% compress  buyuk_dosya
```

komutunu verdiğinizde, **buyuk_dosya** isimli dosya sıkıştırılır ve **buyuk_dosya.Z** isimli daha küçük bir dosyaya dönüştürülür. (MS-DOS dünyasındaki PKZIP gibi). Küçültmenin ne oranda olacağı tamamen dosyanın içeriğine bağlıdır. ASCII text içeren dosyalarda sıkıştırma oranı oldukça yüksek olabilir. MS-DOS'daki PKZIP programından farklı olarak **compress** programı, dosyaları **teker teker ve kendi üzerlerine** sıkıştırır.

```
% uncompress dosya_adi
```

Daha önce sıkıştırılmış olan dosyaları geri açan programdır.

```
% uncompress  buyuk_dosya.Z
```

```
% tail [ -n ] dosya
```

Bir ASCII text dosyasının en son **n** satırını listelemek için kullanılır. Eğer **-n** belirtilmemişse son 10 satır listelenir.

tail komutunun çok hoş bir özelliği daha vardır. Eğer komutu **-f** parametresiyle kullanırsanız, (**tail -f uzayan_dosya**) dosyanın sonuna gelindiğinde program durmaz, eklenecek yeni kayıtları beklemeye başlar. Böylece, başka programlar tarafından sonuna devamlı kayıt eklenen dosyalara eklenen satırları ekranınızda sürekli olarak gözleyebilirsiniz. Gözleminiz sona erince, programı Ctrl-C ile durdurabilirsiniz.

```
% head [ -n ] dosya
```

Bir ASCII text dosyasının ilk **n** satırını listelemek için kullanılır. Eğer **-n** belirtilmemişse ilk 10 satır listelenir.

```
% sort [ -dbfru +f -g ] dosya
```

dosya isimli dosyanın içindeki satırları alfabetik sıraya dizer, sıralı dosyayı standart çıktı birimine (ekrana) gönderir.

-d parametresi *(dictionary sort)* kullanılırsa, sıralama sözlük sırasında yapılır. (Yanlızca rakam, harf ve boşluklar dikkate alınır; noktalama işaretleri ve özel karakterler dikkate alınmaz)

-b parametresi kullanılırsa satır başlarındaki boşluklar dikkate alınmaz.

-f parametresi kullanılırsa büyük harfler küçük harflere dönüştürülerek sıralama yapılır. Örneğin; **Ayfer** ile **ayfer** eşdeğer kabul edilerek sıralama yapılır.

-r parametresi kullanılırsa, sıralama küçükten büyüğe değil, büyükten küçüğe doğru yapılır ***(reverse order)***

-u parametresi kullanılırsa birbirinin aynı olan satırlara rastlandığında sadece bir tanesi dikkate alınır. Bu parametreyi aşağıdaki örnekteki gibi kullanarak bir dosyadaki mükerrer kayıtları ayıklayabilirsiniz.

```
% sort -u dosya1 > dosya2
```

+*f* parametresi, sıralamada kullanılacak anahtar bilginin, satırın ***f*** numaralı bilgi sahasında başladığını gösterir. (**Dikkat** ! Bilgi saha sıra numaraları sıfırdan başlar)

-g parametresi ise sıralamada kullanılacak anahtar bilginin, satırın ***g*** numaralı sahasında sona erdiğini belirtir. (**Dikkat !** Saha sıra numaraları sıfırdan başlar)

Satırlardaki sahalar boşluk karakterleri ve TAB karakterleriyle belirlenir. Eğer boşluk ve TAB dışında bir ayırıcı karakter kullanmak isterseniz, bu karakteri **-tx** parametresini kullanarak (ayırıcı karakter = x) belirtebilirsiniz.

sort komutunun kullanımı üzerine bir dizi örnek vermek istiyorum. Bu örneklerin tümünde aşağıdaki **sirasiz** isimli test veri dosyasının kullanıldığı varsayılmıştır.

Dosyanın orijinal, **sırasız** hali			**sort sirasiz > sirali** komutundan sonra...		
sarf	kalem	200	mobilya	masa	12
mobilya	masa	12	mobilya	sandalye	8
mutfak	kahve	23	mobilya	sehpa	4
mobilya	sandalye	8	mutfak	bardak	70
sarf	silgi	123	mutfak	kahve	23
mutfak	seker	340	mutfak	seker	340
mobilya	sehpa	4	sarf	kalem	200
mutfak	bardak	70	sarf	silgi	123

sort -r sirasiz > sirali komutundan sonra			**sort +1 -2 sirasiz > sirali** komutundan sonra		
sarf	silgi	123	mutfak	bardak	70
sarf	kalem	200	mutfak	kahve	23
mutfak	seker	340	sarf	kalem	200
mutfak	kahve	23	mobilya	masa	12
mutfak	bardak	70	mobilya	sandalye	8
mobilya	sehpa	4	mobilya	sehpa	4
mobilya	sandalye	8	mutfak	seker	340
mobilya	masa	12	sarf	silgi	123

% cmp dosya1 dosya2 *(compare files)*

dosya1 ve **dosya2** isimli dosyaları karşılaştırır; dosyalar arasında bir fark varsa, bu farkların bulunduğu satır ve karakter numarasını verip durur. Örneğin

dosya1	dosya2
Elma, insanlar tarafindan cok sevilen bir meyvadir. Uzun yillardir ben de elma yemeyi bir aliskanlik haline getirdik.	Elma, insanlar tarafindan cok sevilen bir meyvadir. Uzun yillardir biz de elma yemeyi bir aliskanlik haline getirdik.

dosyalarına

```
% cmp dosya1 dosya2
```

komutu uygulanırsa

```
dosya1 dosya2 differ: char 69, line 3
```

yanıtını alırız. (Her satırın sonundaki satır başı karakterini de bir karakter saymayı unutmayın.

cmp komutundan daha yetenekli olan bir de **diff** komutu vardır. Bu **diff** komutunu öğrenmek de sizin ödeviniz olsun. Sınavda sorarım haaaa!

```
% crypt < normal_dosya > kriptolu_dosya
```

Diskteki **normal_dosya** isimli dosyayı okur, klavyeden bir **anahtar sözcük** girmenizi ister ve bu anahtar sözcüğü kullanarak şifreli bir dosya olan **kriptolu_dosya** dosyasını yaratır. Eğer **normal_dosya** isimli dosyayı silerseniz, geriye kalan şifreli dosyanın içeriğini sizden başka hiç kimse bir daha göremez. (**root** kullanıcı dahi göremez). Ancak, programa verdiğiniz anahtar sözcüğü kesinlikle unutmamanız gerekir. Eğer bu sözcüğü bir unutursanız, dosyanızın şifresini çözebilmek için size hiç, ama hiç kimse yardım edemez.

```
% tr [-ds] [dizi_1] [dizi_2] < dosya1 > dosya2
                                                    translate
```

Standart girişteki tüm karakterleri tarayarak **dizi_1** kalıbına uyanları **dizi_2** kalıbındakilerle değiştirir.

Örneğin

```
% tr 123 abc < dosya1 > dosya2
```

komutu verildiğinde, **dosya1** dosyası taranarak, bu dosyada rastlanan tüm **1**'ler **a** ile; tüm **2**'ler **b** ile ve tüm **3**'ler **c** ile değiştirilerek **dosya2** dosyası elde edilir.

Aynı komutu

`% tr [1-3] [a-c] < dosya1 > dosya2`şeklinde de verebilirdik.

Komutun **-s** parametresi kullanılırsa, tekrar eden karakterden oluşan diziler tek karaktere dönüştürülür.

Örneğin, **tekrarli** isimli dosyanın içinde `"Imdaaaaaaaat!!!!!!"` dizisi varsa

```
% tr -s < tekrarli
```

komutu ekrana (standart çıktıya) **"Imdat!"** sözcüğünü gönderir.

% file dosya(lar)

Parametresi olarak verilen dosyaların ne tip dosyalar olduğunu belirtir.

Acemi kullanıcılar sık sık bir dosyanın ne içerdiğini görmek için

```
% cat dosya_adi
```

komutunu verip, dosyayı ekrana listelemek isterler. Eğer söz konusu dosya bir ASCII text dosyası değilse, dosyanın içinde yer alan kod dizilerinden birinin terminali kilitleme olasılığı yüksektir.

Bu komutun önemi işte bu gibi durumlara düşmemek için, bir dosya ya da dosya grubunun ne tip kayıtlar içerdiğini anlamak için kullanılabilmesindedir. Bu komutun **ASCII** ya da **text** olarak nitelendirdiği dosyaların içine korkmadan **cat**, **head**, **tail** veya **more** komutlarıyla bakabilirsiniz.

Örneğin

```
% file a* b*
```

komutu, adı **a** veya **b** harfiyle başlayan dosyaların tiplerini ekrana listeler.

```
abc:/home/ayfer> cd /etc
abc:/etc> file a* b* c* d*
adm:            symbolic link to ../var/adm
aliases:        ascii text
aliases.dir:    empty
aliases.pag:    binary Computer Graphics Metafile
arp:            symbolic link to ../usr/etc/arp
autoreply.data: c-shell commands
cron:           symbolic link to ../usr/etc/cron
domainname:     ascii text
dp:             directory
dp.conf:        English text
dp.start:       executable shell script
dumpdates:      ascii text
abc:/etc>
```

% du [dizin adi] *(disk usage)*

Parametresi olarak belirtilen dizinde (dizin belirtilmezse çalışma dizini kabul edilir) ve onun alt dizinlerinin diskte harcadıkları alanların büyüklükleri listelenir. Bu liste **blok** cinsinden verilir ve **1 blok = 512 Byte'**dır.

```
abc:/home/ayfer> du
146     ./Mail
1       ./.elm
1086    ./burkey
1       ./.wastebasket
1473    ./docs
233     ./denemeler
91      ./kitap/bolumler
134     ./kitap
734     ./Humor
3899    .
abc:/home/ayfer>
```

Bu örneğe göre **Mail** dizinindeki dosyaların toplam uzunluğu 73 Kbyte, **burkey** dizini 543 Kbyte ve tüm dizinlerin toplamı da yaklaşık 1.95 Mbyte dır.

% df *(disk free)*

Komutun verildiği anda **mount** edilmiş olan tüm dosya sistemlerinin toplam kapasitelerini, ne kadarlarının kullanıldığını ve boş yer miktarını **Kbyte** cinsinden listeler. SVR4 UNIX kullanıcıları, aşağıdakine benzer bir liste alabilmek için, komutu **"df -k"** şeklinde kullanmak zorunda kalabilirler.

```
abc:/home/ayfer> df
Filesystem    kbytes    used  avail capacity  Mounted on
/dev/sd0a      16327   12417   2278    84%    /
/dev/sd0g     413767  367489   4902    99%    /usr
/dev/sd0h     529020  419638  56480    88%    /home
abc:/home/ayfer>
```

% tty *(teletype)*

Terminalinizin sisteme hangi arabirimden bağlı olduğunu merak ederseniz kullanmanız gereken komuttur.

```
abc:/home/ayfer> tty
/dev/ttya
abc:/home/ayfer>
```

% bc *(high Precision calculator)*

Oldukça kullanışlı bir hesap makinası programıdır. Tipik kullanıma bir örnek

```
abc:/home/ayfer> bc
12+19
31
37 - 19
18
9 * 6
54
84/7
12
sqrt(64)
8
quit
abc:/home/ayfer>
```

% split [-n] cok_buyuk_dosya

Çok büyük dosyaları daha küçük parçalara ayırmak için kullanılır. Eğer **-n** parametresiyle bir sayı verilmezse **çok-büyük-dosya** 1000'er satırlık parçalara bölünecek ve **xaa, xab, xac,...., xaz, xba, xbb, ...** diye isimler altında gereği kadar dosya yaratılacaktır. Satır başı karakterleri *(CR : Carriage Return)* ile ayrılmış bilgi grupları satır kabul edilecektir. Eğer dosyanın içinde hiç satır başı karakteri yoksa veya makul sıklıkta satır başı karakteri yoksa, parçalama pek başarılı olmayabilir.

Bir dosyayı 100'er satırlık parçalara bölmek isterseniz, kullanacağınız komut

```
% split -100 dosya
```

olmalıdır.

```
% join dosya1 dosya2 > yeni_dosya
```

İki dosyayı **yanyana** birleştirmek için kullanılır. Sanırım bir örnekle açıklaması çok daha kolay olacak :

dosya1	dosya2
isimler	telefonlar
ali	312-111 323 333
veli	212-777 666 666
selami	266-345 345 345

`% join dosya1 dosya2 > dosya3` komutundan sonra

dosya3
isimler telefonlar
ali 312-111 323 333
veli 212-777 666 666
selami 266-345 345 345

```
% touch dosya
```

Bazen, içinde hiç bir şey olmayan boş bir dosya yaratmanız gerekebilir. Böyle bir durumda **touch** komutundan yararlanabilirsiniz.

Eğer parametre olarak verdiğiniz dosya, **diskinizde zaten varsa**, o zaman bu dosyanın en son erişildiği tarih ve saat yenilenecektir. Böylece, dosyanızı sistem yöneticisinin **"Son 10 gündür ellenmemiş dosyaları sil"** şeklinde vereceği genel temizlik komutlarından kurtarmış olursunuz.

Eğer parametre olarak belirttiğiniz dosya yoksa, bu isimde bir boş dosya (sıfır byte uzunluğunda) yaratılır.

X-Windows *X11R5 & X11R6*

Bilgisayar dünyasına ilk olarak Apple marka MacIntosh bilgisayarlarıyla kazandırılmış olan Grafik Kullanıcı Arabirimi *(GUI : Graphical User Interface)* kavramı MS-Windows ile kişisel bilgisayar (PC) dünyasına; X-Windows ile de UNIX dünyasına atladı.

Bir fare ve grafik bir ekranla bilgisayarların kullanımı çok kolaylaştı. Bilgisayarda yapılması istenen işle ilgili olan ikonu *gösterip tıklamak* yeterli bir hale geldi.

X-Windows yazılım paketi MIT *(Massachusetts Institute of Technology)* tarafından geliştirilinceye kadar, UNIX bilgisayarlarının ekranları 24 satır ve her satırda 80 kolondan oluşan zavallı bir görünümün dışına çıkamıyorlardı.

X-Windows; X11R5 *(X11 Release 5)* sürümüyle birlikte grafik ekran sürebilen tüm UNIX bilgisayarlarında kullanılan bir Grafik Kullanıcı Arabirim standardına dönüştü. Bu standart etrafında çeşitli Pencere Yönetim *(Window Manager)* yazılımları üretildiyse de hepsinin temelinde X11 yazılımı yatmaktadır. Bu pencere yönetici yazılımlarına örnek olarak **Openlook**, **Motif**, **VUE** ve **twm** (*tab window manager*; X11R5 in standart pencere yöneticisi) gösterilebilir.

Eğer UNIX bilgisayarınızın grafik ekranı ve uygun bir X11 yazılım paketi varsa hayat sizin çok kolay ve zevkli bir hale gelecektir. Ekranınızda bir sürü pencere açıp, her pencerede farklı bir uygulama başlatıp, UNIX'in çok iş düzenini desteklemesi sayesinde bütün uygulamalarınızı aynı anda yürütebilirsiniz. Hele bilgisayarınız bir ağda yer alıyorsa, pencerelerinizden bir kaçında başka bilgisayarlara bağlantı kurarak (**rlogin** veya **telnet**) bir kaç bilgisayarı aynı anda kullanabilirsiniz. (MS-Windows kullanıcılarının çatladığını duyar gibiyim...)

Bilgisayarınızdaki X olanakları hakkında sistem yönetinizden bilgi alabilirsiniz.