UNIX'de Erişim Yetkileri

UNIX işletim sistemi, kendisini ve denetlediği kaynakları, acemi veya kötü niyetli kullanıcılara karşı korumak zorundadır. Öte yandan, kullanıcıların kayıtlarını da birbirlerine karşı korumak gerekmektedir. Bir üniversitenin bilgisayarındaki öğrenci işleri müdürlüğünün kayıtlarına herkesin erişebildiğini hayal edebiliyor musunuz?

UNIX işletim sistemi, oldukça kuvvetli bir güvenlik sistemine sahiptir ve bu güvenlik sisteminin temelinde, kullanıcıların sisteme tanıtımı sırasında yapılan düzenlemeler yatar. Sistemin yönetiminden sorumlu olan kişi(ler), kullanıcıları kullanım konularına göre sınıflandırır(lar). Örneğin; öğrenci işleri, kütüphane, satın alma, mühendislik fakültesi, edebiyat fakültesi gibi... Bu sınıflara **kullanıcı grupları** *(user group)* adı verilir ve her kullanıcı grubunun bir numarası olur.

Sonra, sıra her bir kullanıcı için bir isim ve kullanıcı numarası vermeye ve bu kullanıcıların ait oldukları grupları belirlemeye gelir.

Özetlemek gerekirse, UNIX işletim sistemi ile çalışan bir bilgisayarı kullanmak istiyorsanız, önce sistem yöneticisine bir uğrayıp sizin için bir **kullanıcı hesabı** *(user account)* açmasını istemelisiniz. Sistem yöneticisi sizin için sistemde daha önce kullanılmamış ve sizin kimliğinizi hatırlatan bir kullanıcı ismi ve sadece size ait olan bir kullanıcı numarası seçecektir. Ait olduğunuz **grubun** numarasına ve sizin kullanıcı dizininizin bulunacağı disk ve dizine de karar verecek ve sisteme sizi tanıtacaktır. Bu işlemler tamamlandığında UNIX bilgisayarının herhangi bir terminalinden **login** edebilirsiniz. Sisteme ilk bağlanışınızda sizden şifre sorulmayacaktır (sistem yöneticiniz, farklı bir düzenleme yaparak yeni kullanıcılara geçici birer şifre vermiş olabilir).

Sisteme girer girmez **passwd** komutu ile şifrenizi tanıtmalı veya değiştirmelisiniz. Başkalarından gizleyecek kayıtlarınız olmayacaksa bile gizli bir şifreniz olmalıdır; aksi takdirde başkaları sizin kullanıcı adınızla sisteme girebilir ve istemeden de olsa kayıtlarınızı bozabilir.

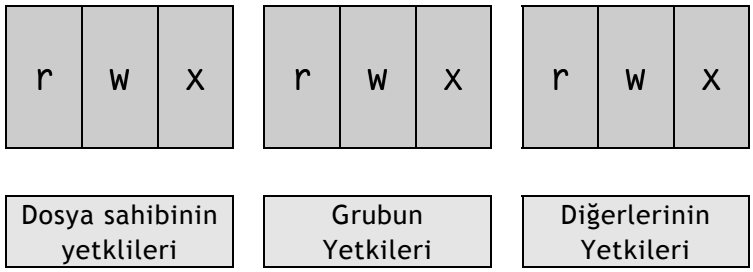
Şimdi dönelim dosya ve dizinlerin erişim haklarına...

Hatırlarsanız, **ls -l** komutu ile bir dizinde yer alan dosyaların (ve dizinlerin) listesini aldığınızda,

```
-rwxr-xr-x  1 root    239783 Feb  09 13:34 kadb
```

benzeri satırlar görmekteydiniz. Bu satırlardaki erişim yetkileri ile ilgili olan **rwxr-xr-x** e benzeyen kod dizileriyle gösterilmektedir. Bu dokuz karakterden oluşan dizi aslında üçer karakterlik üç parçadan oluşmaktadır. (Bu örnekte **rwx**, **r-x** ve **r-x**).

İlk üç karakter dosyanın sahibinin yetkilerini, ikinci üçlü, dosyanın sahibiyle aynı kullanıcı grubunda yer alan kullanıcıların yetkilerini, son üçlü ise **diğer** kullanıcıların bu dosya üzerindeki yetkilerini tanımlamaktadır.



Her üçlü aynı kalıptadır. Her üçlünün ilk pozisyonunda bir **r** harfinin varlığı, ilgili kullanıcının dosyayı okuma yetkisinin bulunduğunu gösterir. Bu pozisyonda bir **eksi** işareti varsa, söz konusu kullanıcı tipi için okuma yetkisi olmadığı anlaşılır.

Bu mantıkla,

r : Okuma yetkisi, ***(read access)***
w : Yazma yetkisi, ***(write access)***
x : Dosya bir program dosyası ise, programı çalıştırma yetkisini gösterir. ***(execute access)***

Bir kaç örnek sanırım konuya daha fazla açıklık getirecektir :

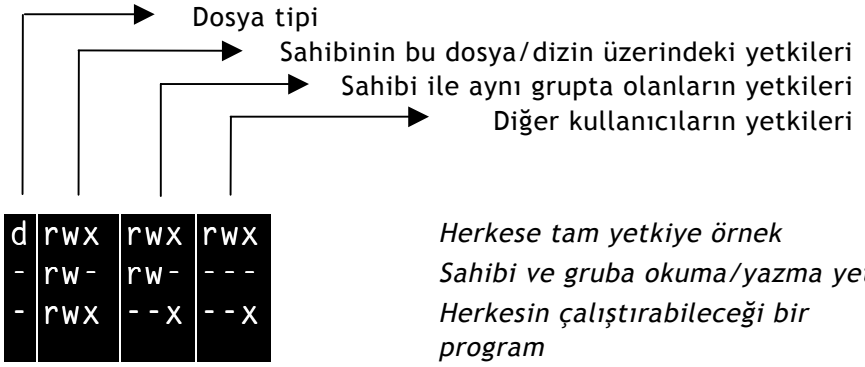
Dosya Yetki Kodu	Anlamı
rwxrwxrwx	Bu dosyayı herkes okuyabilir, Herkes bu dosyaya kayıt yapabilir, dosyanın adını değiştirebilir; hatta dosyayı silebilir, Eğer bu bir program dosyasıysa, herkes bu programı çalıştırabilir.
rwxr-xr-x	Bu dosyayı herkes okuyabilir ve program dosyasıysa çalıştırabilir; ancak, sadece sahibi bu dosyada bir değişiklik yapabilir.
rwx------	Bu dosya üzerinde sahibi istediği tüm işlemleri yapabilir; ancak dosya, diğer kullanıcılara tamamen kapalıdır.
rw-r--r--	Bu dosya bir program dosyası değil, çünkü hiç kimsenin çalıştırma *(execute)* yetkisi yok! Sahibi dosyayı okuyup yazabilir ancak diğer kullanıcılar sadece okuyabilir. (Aslında, henüz çalıştırma yetkileri düzenlenmemiş bir program dosyası olabilir.)
rw-rw----	Bu dosyada bir program dosyası değil. Dosyanın sahibi ve kendisiyle aynı grupta olan kullanıcıların okuyup yazma yetkileri var, ancak diğer kullanıcıların hiç bir şekilde erişmeleri mümkün değil.
rwx--x--x	Sahibi dışında kalan kullanıcılar, bu program dosyasını sadece çalıştırabilirler.

Dizinler için de **rwxrwxrwx** yetki kodları söz konusudur. Dosya yetki kodlarına çok benzemekle beraber, detaylarda bazı önemli farklılıklar vardır. Bu farkları daha sonra açıklayacağım.

Diskinizdeki dosya ve dizinlerin bazılarının yetki kodlarında **r**, **w** ve **x** harflerinden farklı olarak **s**, **S** ve **t** gibi kodlar da görebilirsiniz. Şimdilik bunlara pek aldırmayın.

Bir kez daha özetlemek gerekirse :

% **ls -l** komutu verdiğinizde alacağınız dosya-dizin listesinde göreceğiniz yetki kalıpları aşağıdaki şemaya göre yorumlanmalıdır.



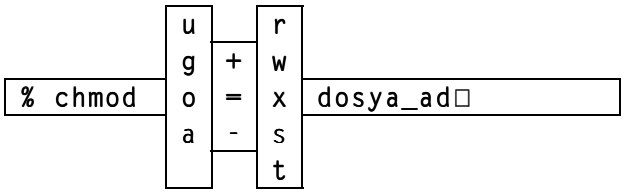
Doğal olarak dosya ve dizinler üzerindeki yetkileri değiştirmek mümkündür; ancak erişim yetkilerini değiştirmeye yetkili olmanız gerekmektedir. Bu yetki sadece dosyanın veya dizinin sahibinde ve **root** isimli kullanıcıda vardır.

Dosya ve dizinlerin erişim yetkilerini değiştirmek için

```
% chmod                                    (change mode)
```

komutu kullanılır.

Bu komut iki değişik formda kullanılabilir. Kullanımı göreceli olarak kolay olan formu :



Bu formda,

u : dosyanın sahibi *(user)*
g : dosyanın sahibiyle aynı grupta olanlar *(group)*
o : diğer kullanıcılar *(others)*
a : herkes *(all)*

+ : yetki ekleme
= : yetki eşitleme
- : yetki çıkarma

r : okuma yetkisi *(read)*
w : yazma yetkisi *(write)*
x : Çalıştırma *(execute)*
s : **suid** biti (daha sonra anlatılacaktır)
t : **sticky** bit (daha sonra anlatılacaktır)

Birkaç örnek vermek gerekirse :

`chmod a+x adres` **adres** isimli program dosyasına, herkes için çalıştırma yetkisi verir.

`chmod o-w mhsb.z` **mhsb.z** dosyasından, diğer kullanıcıların yazma yetkisini kaldırır.

`chmod go=rx adres` **adres** dosyasının grup ve diğerleri için erişim yetkisini **r-x** kalıbına eşitler.

chmod komutunun bir diğer formu da (UNIX ustaları tarafından genellikle tercih edilen form), yetkilerin sayısal olarak gösterildiği formdur. Yetki tanım grupları aşağıdaki tabloya göre sayısal birer değerle eşleştirilir :

4	2	1
r	w	x
Owner		

4	2	1
r	w	x
Group		

4	2	1
r	w	x
Others		

Diyelimki **adresler** dosyasının erişim yetkilerinin **rw-r-xr-x** olmasını istiyorsunuz. Bu yetki kalıbını üçer üçer ayrılmış olarak düşünüp (**rwx r-x r-x**), yukarıdaki tabloya göre verilmek istenen yetkilere karşılık gelen sayıları üçlü gruplar halinde toplayınız ve elde edeceğiniz üç tane sayıyı yanyana getirip 3 haneli bir sayı elde ediniz. Bir başka deyişle :

4	2	1		4		1		4		1
r	w	x		r	-	x		r	-	x
7				5				5		
755										

chmod komutunda bu sayıyı kullanarak dosya ya da dizinlerinizin erişim yetkilerini tanımlayabilirsiniz;

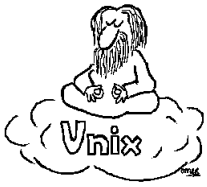
```
% chmod  755  adresler
```

Bir dosya veya dizinin erişim yetkilerini **SADECE** dosyanın sahibi ve **root** kullanıcı değiştirebilir.

Bir **chmod** komutu ile birden fazla dosyanın erişim yetkilerini aynı anda değiştirebilirsiniz:

```
% chmod 755 dosya1 dosya2 dosya3 ...
```

Dosyalar için **r**, **w** ve **x** yetkileri yeteri kadar açık anlamlıdır; ancak dizinler için bu yetkilerin biraz daha karmaşık anlamları vardır. Şöyleki :



Bir dizin için **r** *(read)* yetkiniz varsa :
o dizindeki dosyaların isimlerini **ls** komutu ile görebilirsiniz. (Eğer x *(execute)* yetkiniz yoksa, bazı UNIX'lerde **ls** komutunu kullanabilmenize karşın, **ls -l** komutunu kullanamazsınız).

Bir dizin için **w** *(write)* yetkiniz varsa :
o dizindeki dosyaların yerleşiminde değişiklikler yapabilirsiniz. Örneğin dosyaların adını değiştirebilirsiniz veya dosyaları silebilirsiniz. Eğer bir dizine **w** yetkiniz varsa fakat o dizin içindeki bir dosyaya **w** yetkiniz yoksa, o dosyanın içeriğini değiştiremezsiniz AMA O DOSYAYI SİLEBİLİRSİNİZ veya ADINI DEĞİŞTİREBİLİRSİNİZ).

Bir dizin için **x** *(execute)* yetkiniz varsa :
çalışma dizinizi bu dizin olarak değiştirebilirsiniz. (**cd** komutunu bu dizin için kullanabilirsiniz). Bir dizini çalışma dizini olarak kullanmak için **r** *(read)* yetkisi yeterli değildir; **x** yetkisi de gerekir.
İçinde gizli bilgiler olmayan ama gene de diğer kullanıcılar tarafından kurcalanmasını istemediğiniz dizinler için en uygun yetki düzenlemesi **rwxr-xr-x** olarak kabul edilebilir.

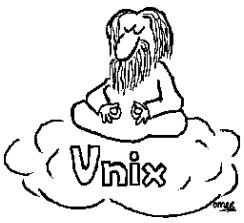
Eğer bir dizininizi sizden başka kimsenin kullanmasını ve içine bakmasını istemiyorsanız

```
% chmod go-rwx dizin_adi
```

komutunu kullanabilirsiniz

SUID Biti ve SUID Programlar

Bir kaç sayfa önce, **chmod** komutundan söz ederken **suid bit** kavramından bahsetmiştim. Bir program dosyasının **SUID** bitini set etmek (yani **chmod +s prog** gibi bir komut vermek), bu **prog** programını çalıştıran kullanıcıların, program çalıştığı sürece ve sadece bu program ile ilgili dosyalar açısından, program dosyasının sahibinin yetkilerine sahip olmalarını sağlar. Biraz karışık oldu ama, sanırım şu örnek açıklayıcı olacaktır.



Şifresini değiştirmek isteyen bir kullanıcı **passwd** komutunu kullanacaktır. Bu program kullanıldığında, şifre değişikliği, sahibi **root** olan **/etc/passwd** dosyasında bir kayıt değişikliği yapılmasını gerektirecektir. Ancak bu dosya, UNIX sisteminin en önemli dosyalarından birisi olduğu için çok iyi korunmakta ve sahibi (yani **root**) dışında kimsenin bu dosyaya **w** *(write)* yetkisi bulunmamaktadır. İşte **SUID** kavramı bu soruna bir çözüm getirmektedir. **passwd** programının yer aldığı **/usr/bin/passwd** dosyasının **SUID** biti set olduğu için, **passwd** komutunu veren kullanıcılar bu program çalıştığı sürece ve **/etc/passwd** dosyasına erişim söz konusu olduğunda geçici olarak **root** yetkilerine sahip olacaklardır.

Sistem yöneticisine : SUID programlar önemli birer emniyet gediği olabilirler. Bir programa SUID özelliği vermeden önce dikkatlice düşününüz. Eğer, SUID özelliği vermek istediğiniz program, kullanıcıya bir şekilde UNIX komutu verme olanağı sağlıyorsa; bu programa kesinlikle SUID özelliği vermeyiniz.

"SUID" özelliğine sahip dosyalar, ayrıntılı **ls** listelerinde bir **s** harfiyle gösterilir.

```
-rwsr-xr-x  2 bin        512 Feb  24 09:56 passwd*   gibi.
```

STICKY Bit

Eski UNIX uyarlamalarında; disklerin ortalama erişim sürelerinin ve veri transfer hızlarının düşük olduğu zamanlarda; program dosyalarının disklerden belleğe yüklenebilmeleri için geçen süreler kullanıcıları rahatsız etmekteydi. Bu yüzden, sık sık kullanılan komutları oluşturan programların disk dosyalarına **"sticky"** özelliği verilirdi. Bu özellik sayesinde, bu tip programlar, bir kez belleğe yüklendikten sonra, programın çalışması sona erdiğinde bile bellekten atılmazlardı; böylece, komutun bir sonraki kullanımı için program bellekte hazır olurdu. **"sticky"** özelliğine sahip dosyalar, ayrıntılı **ls** listelerinde bir **t** harfiyle gösterilir.

```
-rwxr-xr-t  2 bin        512 Feb  24 09:56 ls*   gibi.
```

Artık, günümüz UNIX'lerinde STICKY BIT kavramı kullanılmamaktadır. Öte yandan, bu bit'in, asıl tarifine hiç de benzemeyen amaçlarla kullanan UNIX uygulamalarının bulunduğu da bir gerçek; ancak, bu özel durumlar konumuzun oldukça dışında kalıyor.

Dosyaların/Dizinlerin Sahibini Değiştirme

```
# chown                                    (change owner)
```

Bu komutu sadece root kullanıcı kullanabilir!

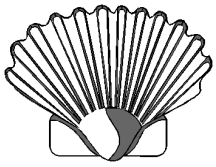
Erişim yetkileriyle ilgili olarak, zaman zaman dosya ve dizinlerin sahiplerinin değiştirilmesi gerekebilmektedir. Örneğin, **root** kullanıcı bir nedenle, bir kullanıcı dizininde bir dosya ya da dizin yaratırsa ve bu yeni yaratılan dosya/dizinin o kullanıcı tarafından tam yetkiyle kullanılmasını isterse, bunu sağlamanın en kolay yolu, bu yeni yaratılan dosya/dizinin sahibini o kullanıcı yapmaktır :

```
# whoami                        (ben kimim?)
root                            (root' muşum)
# mkdir /home/ayfer/yeni        (yeni isimli dizini yarat)
# ls -l /home/ayfer
.
.
drwxr--r--  1 root    512 Feb  12 13:34 yeni
.
```

```
.
# chown ayfer /home/ayfer/yeni     (sahibini değiştir)
# ls -l /home/ayfer
.
.
drwxr--r--  1 ayfer  512 Feb  12 13:35 yeni

.
.
```

csh ve sh kabukları



Bir UNIX bilgisayarına **login** ettiğinizde karşınıza bir **kabuk programı** çıkacaktır. Bu andan itibaren vereceğiniz tüm UNIX komut satırlarını irdeleyen, verdiğiniz komuta uygun programı diskten belleğe yükleyen, çalıştıran ve varsa, verdiğiniz parametreleri bu programa aktaran işte bu kabuk programıdır.

> Bir kullanıcı **login** ettiğinde, kendisi için hangi kabuk programının çalıştırılacağına sistem yöneticisi karar verir. Sistem yöneticisi, kullanıcı tanıtımı sırasında bu bilgiyi, **/etc/passwd** dosyasında, söz konusu kullanıcıya ait satırın sonundaki parametrede belirtir. Ancak bu seçim mutlak değildir. Kullanıcılar istedikleri kabuk programını kullanabilirler. Hatta, çok pencereli bir ortamda (*windowed environment : X-Windows, Motif* gibi) çalışıyorlarsa, farklı pencerelerde farklı kabuklar bile kullanabilirler. Kabuk değiştirmek için gereken tek işlem, istenen kabuk programını çalıştırmaktan ibarettir. Bir kabuk programıyla işiniz bittiğinde **exit** komutuyla o kabuk programınızdan çıkabilirsiniz.

Şimdi bir komutun irdelenmesi ve yerine getirilmesi aşamalarını bir örnekle izleyelim.

UNIX işletim sistemine verilen komutun

```
% cp  ./a*  /home/ugur
```

olduğunu varsayalım.

Kullanmakta olduğunuz **csh** veya **sh**, RETURN tuşuna basıldığında, **c** harfi ile başlayıp **ugur** sözcüğü ile biten satırınızı irdeleyecek, **cp** harflerinin bir komut adı olduğunu kabul edecektir. Komut satırının geri kalanını da bu komutun parametreleri olarak çözümlemeye çalışacaktır.

./a* parametresini ayırırken ***** *(asterisk)* karakterini görünce şöyle bir duralayıp çalışma dizininde (**.**) bulunan ve adı **a** harfi ile başlayan dosyaların isimlerini bulacak ve sanki herbiri klavyeden yazılmışcasına komut satırına yerleştirecektir. Sonra da **/home/ugur** karakterlerini son parametre olarak değerlendirecektir. Sonuçta, ilk verdiğimiz komut

```
% cp ./abc  ./abd ./abe  /home/ugur
```

şekline dönüşecektir. (tabii ki çalışma dizininizde **abc**, **abd** ve **abe** isimli dosyalar olduğunu varsayarak).

Daha sonra, **cp** komutuna ait program dosyası **PATH (path** de olabilir**)** isimli **kabuk** değişkeninde (**shell variable**) yer alan dizinlerde aranacak ve bulunursa belleğe yüklenerek çalıştırılacaktır. (**PATH** ve **path** kabuk değişkenleri daha sonra anlatılacaktır). Komut satırının geri kalan kısmıysa (**./abc ./abd ./abe /home/ugur**) **cp** komutuna parametre olarak gönderilecektir.

UNIX işletim sisteminde dikkat edilmesi gereken önemli bir nokta : Dosya isim kalıpları *(wildcards)* kullanılan komutlarda; kalıpların açılması işlemi komut çalıştırılmadan önce yapılır ve bu açılmış halleri, ilgili komut programına parametre olarak aktarılır.

Eğer, bu kalıpların açılmadan, parametre olarak programa geçirilmesini istiyorsanız, söz konusu kalıbı **" "** (çift tırnakl arasına yerleştirmelisiniz.

Dosya İsim Kalıpları

Hatırlayacağınız gibi, MS-DOS işletim sisteminde, bir anda birden fazla dosyanın adını kullanmak gerektiğinde, söz konusu dosyaların isimlerinin ortak karakterlerinden yararlanarak kalıplar yazmak mümkündür. Örneğin

```
COPY *.DAT  A:
COPY TA*.* \ESKI
DIR K??.DOC                    gibi.
```

UNIX işletim sisteminde de bu tür kolaylıkların; hatta daha fazlasının olması doğaldır, değil mi?

UNIX *Wildcard* karakterleri

*****	Her türlü karaktere uyar. Tüm dosyaları seçen bir kalıp kullanmanız gerekirse tek bir ***** karakteri yeterli olacaktır. MS-DOS'daki gibi ***.*** yazarsanız, adının içinde en az bir nokta olan dosyaları seçmiş olursunuz; yani adının içinde nokta olmayan dosyaları seçmemiş olursunuz.
?	Herhangi bir **tek** karaktere uyar. Örneğin, **???** dizisi, adı üç harfe kadar olan tüm dosyalara uyacaktır.

[a,b,c]	**a** veya **b** veya **c** karakterlerinin herbirine uyar.
[0-9]	0'dan 9'a kadar rakamlara uyar.

Örnekler

`% cat kitap[1-3] > hepsi`	**kitap1**, **kitap2** ve **kitap3** dosyalarını peşpeşe ekleyerek **hepsi** isimli dosyaya kopyalar.

`% chmod a=x *.[o,sh]`
veya
`% chmod a=x *[.o,.sh]`
	Adının son karakterleri **.o** veya **.sh** olan tüm dosyaların erişim yetkilerini herkes için **--x** olarak değiştirir.

*** kalıp** karakteri, adı **.** (nokta) ile başlayan dosyalar hariç tüm dosya isimlerine uyar. Örneğin **cp * /home/ayfer/sakla** komutu, çalışma dizinindeki, adı nokta ile başlayanlar dışındaki tüm dosyaları **/sakla** dizinine kopyalayacaktır. Eğer adı nokta ile başlayan dosyaları da **(.login, .cshrc** gibi) kopyalamak istiyorsanız

```
% cp  .*  *  /home/ayfer/sakla    veya
% cp  .l*  .c*  *  /home/ayfer/sakla
```

komutunu kullanmalısınız.

Kabuk programlarının, dosya adı kalıbı kullanılan komut satırlarını irdelemesi biraz özeldir. Şöyleki, verdiğiniz komut satırındaki *, ?, [ve] karakterler, komut işletilmeye başlamadan çalışma dizininizde veya belirtilen dizinde bulunan dosyalarla eşleştirilir, kalıba uyan dosyaların isimleri komut satırına yerleştirilir ve program ondan sonra çalıştırılır. Bir örnek vermek gerekirse; yazacağınız **cp *.dat /home/ayfer** gibi bir komut satırı, kabuk programı tarafından

```
% cp ocak.dat subat.dat mart.dat /home/ayfer
```

şeklinde bir satıra dönüştürülüp kopyalama programı ondan sonra başlatılır. (Tabii ki, çalışma dizininizde ocak.dat vs isimli dosyaların bulunması şartıyla.) Komutlarınızı verirken bu davranışı dikkate almanız gerekir. Eğer, dosya isim kalıpları tırnak içinde yer alıyorsa, o zaman bu açma işlemi yapılmaz.

Shell Değişkenleri

Bir UNIX bilgisayarına **login** ettiğinizde, bir **shell** çalışma seansı başlatmış olursunuz. Bu seans boyunca kullanacağınız bir takım programlar, çalışmalarını düzenleyen bazı değişkenlerin (**kabuk değişkenleri**) belirli değerlere sahip olmasını isteyebilirler. Örneğin, SUN iş istasyonlarında, **openwin** isimli program, **OPENWINHOME** isimli bir değişkende, **openwin** programının yüklenmiş olduğu dizin adının (**/usr/openwin** gibi) bulunmasını ister.

Kullandığınız kabuk programına göre, uygun bir komutla kabuk değişkenleri yaratıp, bunlara birer değer verebilirsiniz. Örneğin,

C-Shell için

```
% setenv OPENWINHOME  /usr/openwin                csh için
```

Bourne shell (sh) için

```
$ OPENWINHOME=/usr/openwin                       sh için
$ export OPENWINHOME
```

path ve PATH Değişkenleri

UNIX işletim sisteminde büyük ve küçük harf ayrımının önemli olduğunu daha önce vurgulayarak belirtmiştim. **path** ve **PATH** değişkenleri de bu nedenle farklı değişkenlerdir ancak görevleri aynıdır. **path** değişkeni c-shell için; **PATH** ise Bourne shell için anlamlıdır. Ancak csh, **path** değişkenine bir değer verildiğinde, **PATH** değişkeninde de uygun değer değişikliğini otomatik olarak yapar.

path ve **PATH** değişkenlerinin görevi basittir. Bir komut verildiğinde, komutu oluşturan program dosyasının hangi dizinlerde aranacağını belirtir.

```
% set path = ( /bin /usr/bin ~/bin . )        csh için
```

veya bunun eşdeğeri olan

```
$ PATH=/bin:/usr/bin:~/bin:.                     sh için
```

değer atamaları yapılmışsa, UNIX işletim sistemine bir komut verildiğinde, program dosyası önce **/bin** dizininde aranır. Eğer bulunamazsa, **/usr/bin** dizinine bakılır. Eğer orada da bulunamazsa kullanıcı dizinin altındaki (**~** : *home directory*) **bin** dizinine bakılır. En son olarak da çalışma dizinine bakılır (**.**).

MS-DOS işletim sisteminde durum biraz farklıdır. MS-DOS'da, bir komut öncelikle çalışma dizininde aranır, bulunamazsa PATH değişkeninde belirtilmiş olan dizinlerde aranır.

Diğer Önemli Shell Değişkenleri

PRINTER : Bir dosyayı yazıcıya göndermek için kullanılan **lpr** komutunda, **-P** parametresiyle yazıcı adı belirtmediğinde, varsayılacak yazıcı adını belirler.

```
% setenv PRINTER laser              csh için

$ set PRINTER=laser                 sh için
$ export PRINTER
```

TERM : Kullandığınız terminalin tipini belirler. Bu değişkene vermeniz gereken değeri sistem yöneticinizden öğrenmelisiniz.

```
% setenv TERM vt100                 csh için

$ set TERM=vt100                    sh için
$ export TERM
```

MANPATH : UNIX komutları hakkında bilgi almanız gerektiğinde kullanacağınız **man** komutunun, komutlar hakkındaki kullanım kılavuzu sayfalarını bulacağı dizin veya dizinleri belirtir. Bu değişken tanımlı değilse, **man** komutu, kılavuz sayfalarını **/usr/man** dizininde arar.

```
% setenv MANPATH /usr/man:/use/lang/man
```

.login ve .logout Dosyaları

Eğer kullanıcı dizininizde **.login** isimli bir dosya varsa, sisteme **login** ettiğinizde bu dosyadaki UNIX komutları sanki klavyeden birer birer girilmiş gibi çalıştırılacaktır. (Size özgü bir AUTOEXEC.BAT dosyası gibi).

Aynı mantıkla, sistemden çıkmak için **logout** komutunu verdiğinizde de; varsa, **.logout** dosyasındaki komutlar çalıştırılacaktır.

Bu dosyalar sizin kullanıcı dizininizde *(home directory)* yer aldığı için sadece size özgü dosyalardır. Bu dosyalarda istediğiniz değişiklikleri yapabilirsiniz. Yapacağınız değişiklikler başkalarını etkilemeyecektir.

Tipik bir **.login** dosyası

```
set TERM=vt100
set path=/usr/bin:/bin:/usr/lang:
set PRINTER=laser
```

Tipik bir **.logout** dosyası

```
/bin/rm  ~/tmp/*              geçici dosyaları siliyor
clear
echo "Gule Gule..."
```

.cshrc Dosyası

Sizin için bir **csh** çalışmaya başlarsa ve sizin kullanıcı dizininizde **.cshrc** isimli bir dosya varsa, **csh** kabuk programı ilk olarak bu dosyadaki UNIX komutlarını çalıştıracaktır. Her ne kadar kesin bir kural değilse de, geleneksel olarak bu dosyanın içine sadece **csh** kabuk programını ilgilendiren UNIX komutları yerleştirilir.

Şimdi, iyi hazırlanmış bir **.cshrc** dosyasını örnek olarak inceleyelim; ancak bu dosyayla ilgili notlar size pek açıklayıcı gelmezse hiç üzülmeyin ve bu bölümü atlayın. Ama, bu kitabı bitirmenize yakın bu bölümü tekrar bir gözden geçirmenizi öneririm.

```
#########################################################
# Ornek bir .cshrc dosyas□                              #
#########################################################
#
set lpath=( /home/ayfer/bin )
set path= ( /usr/bin /usr/local/bin /bin $lpath )
set cdpath = (~/src ~/bin ~ )
set noclobber
set history=30
set ignoreoef
#
alias dir ls
alias copy 'cp -i'
alias ll 'ls -l'
alias mroe more
alias h history
#
umask 022
#########################################################
```

##.....##	Bu karakterle başlayan satırlar açıklama satırı *(comment)* olarak kabul edilir ve **csh** tarafından dikkate alınmaz.

`set lpath=( .. )`	**lpath** isimli bir kabuk değişkeni tanımlanmış ve değer olarak kullanıcının kişisel programlarını yerleştirdiği dizinlerin isimleri verilmiş. *(local path)* .

```
set path=( .. $lpath )
```

path isimli shell değişkeni tanımlanıyor. Yani, bir komut verildiğinde, komutla ilgili program önce **/usr/bin** dizininde; burada bulunamazsa **/usr/local/bin** dizininde, orada da bulunamazsa **/bin** dizininde; orada da bulunamazsa **lpath** isimli kabuk değişkeninde tanımlanmış olan dizinlerde aranacaktır. (Daha önce tanımlanmış bir değişkenin değerinin kullanılabilmesi için değişkenin adının başına bir **$** işareti yerleştirmek gerekmektedir.)

```
set cdpath=(~/src .. )
```

Bu kabuk değişkeni, yalnızca **csh** kabuğunda kullanılabilir. Hayatı oldukça kolaylaştıran bir csh özelliğidir.

Diyelim ki, kullanıcı dizininiz altında, geliştirdiğiniz programların kaynaklarının yer aldığı **src** isimli bir dizin var ve bu dizinin altında da **proje1** ve **proje2** isimli iki dizin var. Bir dizinden diğerine geçmek için, normal olarak **cd home/ayfer/src/proje1** gibi uzun **cd** komutları yazmak gerekecektir. Eğer **cdpath** değişkeniniz uygun şekilde tanımlı ise, **proje2** dizinine geçmek için, herhangi bir dizindeyken **cd proje2** komutunu vermeniz yeterli olacaktır. Bir başka deyişle, dosyalar için **path** değişkeninin sağladığını, **cdpath** değişkeni dizinler için sağlar.

```
set noclobber
```

Bir programın çıktısını **>** operatörünü kullanarak bir dosyaya yönlendirdiğinizde, eski bir dosyanın üzerine kayıt yapmanız durumunda uyarılmanızı sağlar. Örneğin, kullanıcı dizininizdeki dosyaların bir listesini bir dosyada saklamak istediğinizde **ls -l > dosya_listesi** benzeri bir komut vermeniz gerekecektir. Eğer **set noclobber** komutu verilmişse (**noclobber** özelliği açılmışsa) ve **dosya_listesi** isimli dosya önceden varsa, kullanıcı uyarılacaktır.

```
set history=30
```

history, csh'e özgü bir değişkendir. Kullanıcının verdiği son 30 komutun bellekte saklanmasını, ve **!** işareti yardımı ile eski komutların tekrarlanabilmesini sağlar. Vermiş olduğunuz son UNIX komutunu tekrarlamak isterseniz, klavyeden **!!** girmeniz yeterlidir. (MS-DOS'daki F3 tuşu gibi.)

Sondan bir önceki komutu tekrarlamak isterseniz
!-2, 7 adım önceki komutu tekrarlamak için **!-7** komutlarını kullanmanız gerekir.

Bitmedi...
Daha önce verilmiş komutlar arasında, **c** harfi ile başlayan en son komutu tekrarlamak için **!c** komutunu kullanabilirsiniz. Daha kesin tanımlamalar gerekirse, **!ca** gibi daha uzun diziler kullanabilirsiniz.

Hala bitmedi...
Son vermiş olduğunuz 30 komutu görmek için (daha doğrusu **history** isimli c-shell değişkeninde belirtilmiş sayı kadar) **history** komutunu kullanabilirsiniz. (Lütfen, "history komutu" ile "history değişkenini" karıştırmayınız; bu ikisi ilgili olmakla birlikte farklı şeylerdir). **history** komutunu verdiğinizde, daha önce vermiş olduğunuz komutlar, birer sıra numarasıyla ekrana listelenir. Bu listedeki komutlardan birini tekrarlamak istediğinizde **!** işareti ve hemen yanına tekrarlamak istediğiniz komutun sıra numarasını yazmanız yeterlidir. (**!14** gibi..)

```
set ignoreoef
```

Ctrl-D tuşunun **logout** anlamını kaldırır.

Bir programa bilgi girişini bitirmek için genellikle **Ctrl-D** tuşuna (daha doğrusu tuşlarına) basılır. Ancak yanlışlıkla birden fazla **Ctrl-D** basarsanız, ilki programınıza bilgi girişini sona erdirir, ikincisiyse kullandığınız kabuk programını durdurur. Eğer bu kabuk, yegane kabuk programınızsa, sistemden **logout** etmenize neden olur.

. Bu tip hataları önlemek için **set ignoreeof** komutuyla Ctrl-D'nin bu anlamı kaldırılır. Artık **logout** edebilmek için açıkça **logout** komutunu vermeniz gerekecektir.

```
alias dir ls
alias copy 'cp -i'
alias ll 'ls -l'
alias mroe more
alias h history
alias ls 'ls -F'
```

Daha önce UNIX'de kendi komutlarınızı yaratmanın mümkün olduğunu söylemiştim. İşte bu amaçla kullanılan **alias** komutuna bir kaç örnek...

alias dir ls komutu, MS-DOS alışkanlıklarından kolay vazgeçemeyen kullanıcılar için yararlı olabilir. Bu komutu verdiğinizde, artık, dosya listesi almak için isterseniz **dir** isterseniz **ls** komutunu kullanabilirsiniz.

Aynı mantıkla **cp -i** yerine **copy** komutunu kullanabilirsiniz.

Dikkat : Yeni **copy** komutunun tanıtımı iki bölümden ('**cp**' ve '**-i**') oluşuyor. Bu iki bölümün birarada değerlendirilmesi için tırnak içine alınmaları gerekmiştir.

ll *(long ls)*, oldukça sık kullanılan '**ls -l**' komutu için bir kısaltma olarak tanıtılmıştır.

mroe, klavyeden hızlı bir şekilde **more** yazmak istediğinizde yapabileceğiniz bir hatayı baştan düzeltmek için tanımlanmış bir komuttur. Yani, yanlışlıkla **mroe** yazdığınızda bile, bu komut **more** ile eşdeğer kabul edilecektir.

h ise, uzun uzun **history** yazmaktan kurtulmak için tanımlanmıştır.

ls komutuysa **-F** parametresi ile birlikte çalışacak şekilde değiştirilmiştir. Hatırlarsanız **-F** parametresi kullanıldığında, **ls** komutu, dizin isimlerinin sonuna bir **/** işareti; çalıştırılabilecek program dosyalarının sonunaysa birer ***** işareti yerleştirmekteydi...


```
umask 022
```

Bu komut, yeni UNIX kullanıcıları için biraz karmaşık gelebilir. Eğer size de karışık gelirse bu bölümü atlayabilirsiniz. **.cshrc** dosyanızda bu komutun bulunması yararlıdır; bence nedenini anlamasanız da bu komutu **.cshrc** dosyanıza koyunuz.

Hatırlarsanız, dosya ve dizinler için erişim yetkileri söz konusuydu. Her dosya ve dizin yaratışınızdan sonra bu yeni yarattığınız dosya veya dizinin erişim yetkilerini **chmod** komutuyla değiştirmek yerine, bu yeni yaratılanlar için sizin seçeceğiniz bir erişim yetki kalıbının

otomatik olarak kullanılmasını sağlamak için **umask** komutunu kullanılır. **umask** komutunun parametresini oluşturan (parametresiz kullanırsanız, o anda geçerli olan **umask** değerini öğrenirsiniz) 3 haneli sayının yorumlanması biraz gariptir. **umask** parametresi, verilen yetkileri değil, kaldırılan yetkileri belirtir.

Sanırım bir örnekle anlatmak daha kolay olacak :

Eğer standart yetki kalıbı, **umask** komutuyla değiştirilmemişse yeni yaratılan dosyalara **rw-rw-rw-**; dizinlereyse **rwxrwxrwx** erişim yetkileri verilir.

Şimdi; **umask 022** komutunun verildiğini varsayalım. 022 sayısını 0 2 2 şeklinde 3 ayrı sayı olarak düşünün ve her sayıyı üçer haneli ikilik sayılara *(binary)* çevirin.
0 2 2 : 000 010 010 gibi... Bu diziyi **rw- rw- rw-** ve **rwx rwx rwx** yetki kodları ile alt alta yazın.

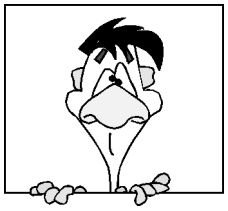
Dosyalar için	Dizinler için
rw- rw- rw-	rwx rwx rwx
000 010 010	000 010 010

Bu düzenlemede 0'ların altına gelen yetkilere dokunulmamakta, ancak 1'lerin altına gelen yetkiler kaldırılmaktadır

Yani, eğer yaratılan bir dosyaysa, erişim yetkileri **rw-r--r--**; bir dizinse, **rwxr-xr-x** olarak belirlenecektir.

Dosyalar için	Dizinler için
rw- rw- rw-	rwx rwx rwx
000 010 010	000 010 010
rw- r-- r--	rwx r-x r-x

Kararsız kullanıcılara önerim, umask olarak 022 kullanmalarıdır.



vi

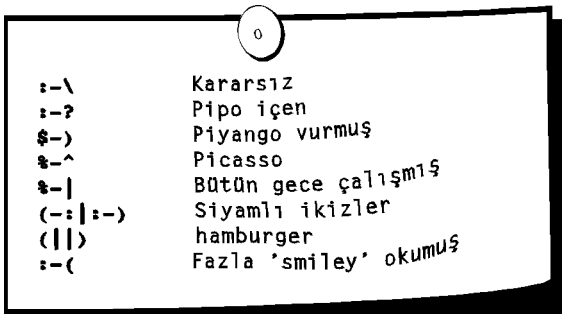
visual Editor

UNIX işletim sistemiyle birlikte kullanılabilecek pek çok editör vardır; **emacs**, **edit**, **ed**, **ex** gibi.... Ancak bunlar arasında, tüm UNIX uyarlamalarında standart olarak bulunan editör **vi**'dır. Bu nedenle, her UNIX kullanıcısının, az da olsa **vi** editörünü kullanmayı bilmesi gerekmektedir. Bu bölüm, kullanıcılara, metinlerini ve programlarını yazabilecek ve bunlar üzerinde değişiklikler yapabilecek kadar **vi** öğretmek için hazırlanmıştır. Bu bölümde anlatılan her komut ve işlem dizisini ezbere bilmek gerekmemekle birlikte, tamamını bir kere okumak ve örnekleri tekrarlamak yararlı olacaktır.

MS-DOS dünyasının editörlerine alışkın bir kullanıcı için, **vi**, başlangıçta çok itici gelecektir. Ancak; bu iticiliğin arkasında her türlü UNIX bilgisayarının her türlü ekranında çalışabilme özelliği olduğunu unutmayınız.

Beğenseniz de beğenmeseniz de, vi öğrenmelisiniz!

vi bir kelime işlemci değildir! Sadece bir editördür. UNIX altında kelime işlem uygulaması yapmak isterseniz **latex**, **nroff** ve **troff** gibi yazılımlar kullanmanız gerekecektir; ama, bu yazılımların kullanımının temelinde gene **vi** bulunacaktır.

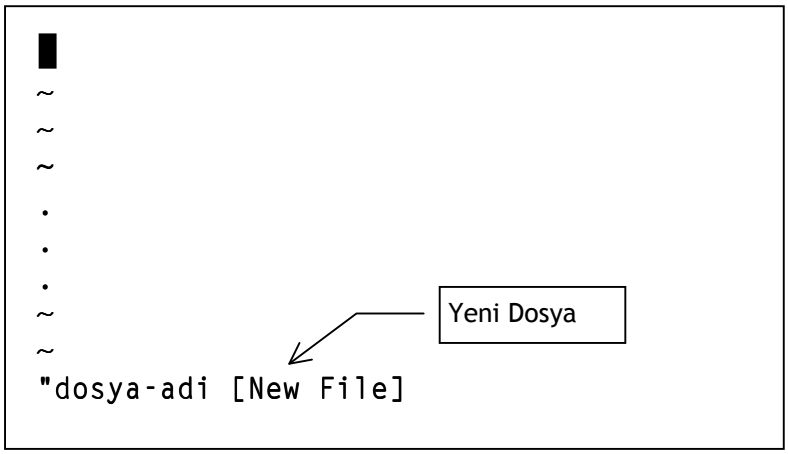


Dosya Açma / Yaratma

```
% vi   dosya-adi
```

komutuyla **dosya-adı** adlı dosyayı edit etmeye başlayabilirsiniz

Eğer bulunduğunuz dizinde bu isimli bir dosya yoksa, ekranda şöyle bir görüntü ile karşılaşaksınız:



Bu ekranda ▉ işareti imleci *(cursor)* temsil etmektedir. **~** işaretleriyse ekranda aynen görünmekte ve bu satırlarda herhangi bir kayıt bulunmadığını (boş satır bile bulunmadığını) göstermektedir.

Boşluk(lar)dan oluşan satırla, içinde bir şey olmayan satır farklıdır!

Edit etmek istediğiniz dosya bulunduğunuz dizinde olmak zorunda değildir.

```
% cd /home/ayfer
% vi /home/reyyan/deneme      geçerli bir komut dizisidir.
```

İlk vi Denemesi : Bir dosya yaratalım...

```
% vi deneme
```

komutunu verip editörü başlatınız. Çalışma dizininizde **deneme** isminde bir dosya yoksa, ekranda :

```
~
~
~
.
.
~
~
"deneme" [New File]
```

göreceksiniz.

EKLEME DURUMU

İmleç sol üst köşede iken **i** (*insert* komutu) tuşuna bir kez basınız. (Küçük **i** tuşuna basmaya dikkat ediniz). Bu andan itibaren basacağınız her tuş, dosyanın içine kaydedilecektir. Örnek olarak şu satırları giriniz :

```
Bu ilk deneme dosyamizdir.
Enter (veya RETURN) tusuna basarak alt satira geciniz.
Bu da ucuncu satir.

Bir satir bos biraktiktan sonra devam edebilirsiniz.
Son satir
~
.
.
~
"dosya-adi [New File]
```

Şimdi dosyayı saklamayı görelim.

KOMUT DURUMU

Dosyayı saklamak için, editöre sakla komutunu verebilmek için **ekleme durumundan** (*insert mode* : hatırlarsanız en başta bir küçük **i** harfi ile bu duruma geçmiştik) çıkıp **komut durumuna** geçmemiz gerekecektir. **Ekleme durumundan** çıkmak için bir kez **Esc** tuşuna basmamız yeterlidir. (Fazla **Esc** basmanın bir zararı olmaz. Eğer varsa, ekranınızın düdüğü, her fazla basış için bir kez ötecektir; o kadar.)

Şimdi artık komut durumuna geçtiğimize göre **:wq** (iki nokta üstüste - **w** harfi - **q** harfi) tuşlarına basarak **w** *(write)* **q** *(quit)* komutlarını verebiliriz. (*"Komutlar"* dedim; çünkü **w** ve **q** aslında ayrı ayrı iki komuttur).

Ekleme veya Komut durumlarının hangisinde bulunduğunuza dair ekranda bir işaret aramayın. Yoktur! Ancak, hangi durumda bulunduğunuzdan emin olmak istiyorsanız, **bir kaç kez Esc** tuşuna basıp komut durumuna geçiniz. Gerekirse **i** komutu ile komut durumundan ekleme durumuna geçebilirsiniz.

Tekrar deneme çalışmamıza dönersek; son bastığımız **Esc** tuşundan ötürü komut durumunda bulunduğumuzu biliyoruz. Komut vermek amacıyla **:** (iki nokta üstüste) tuşuna bastığımızda, imleç, ekranın en alt sol köşesine inecek; bastığımız **:** karakterini burada gösterecek ve komut bekleyecektir. komut olarak **wq** harflerini giriniz (**w** *(write)* ve **q** *(quit)* komutları).

Dosyayı kaydetme işleminin başarılı olduğunu şu mesajdan anlayabilirsiniz :

```
"deneme" [New file] 7 lines   135 characters
```

q *(quit)* komutundan ötürü, **vi**'dan tamamen çıkmış ve UNIX komut düzeyine dönmüş olmalısınız. Şimdi, UNIX'deki dosya isimlerini listeleme komutuyla bu yeni yarattığımız dosyanın adını görebilmelisiniz. (İpucu : **ls** komutu.)

Dosya açarken karşılaşabileceğiniz sorunlar :

- Var olduğunu bildiğiniz bir dosya adı vermiş olmanıza rağmen, ekranın en altında **[New File]** mesajını görebilirsiniz. Bu durum, büyük olasılıkla dosyanın adının **vi** başlatma komutunda hatalı yazılmış olmasından kaynaklanmaktadır.

UNIX işletim sisteminde deneme ile Deneme farklı dosya adlarıdır!

Eğer hata bu değilse, büyük olasılıkla söz konusu dosya, belirtilen dizinde bulunmamaktadır (**vi** komutunda dizin belirtilmemiş ise, bulunduğunuz dizinde bu dosya yoktur.)

- Aşağıdaki mesajlardan birini görürseniz :

```
[open mode]  veya

Visual needs addressable cursor or upline capability
```

veya

```
I don't know what kind of terminal you are on - all I have is
'unknown'. Using [open mode]
```

Kabuk değişkenlerinizden **TERM** ya hiç tanımlı değil, ya da hatalı tanımlanmıştır. Bu durumda lütfen hemen **:q** komutlarıyla (iki nokta üstüste ve ardından küçük **q** ve ardından RETURN) **vi** programından çıkıp, sistem yöneticinize başvurunuz. Eğer sistem yöneticisi sizseniz, **:q** komutu ile çıkıp

```
% setenv TERM vt100          (csh için)

$ set TERM=vt100             (sh için)
$ export TERM
```

komutları ile **TERM** değişkenini tanımlamayı deneyiniz. Eğer başarılı olmazsa, **vt100** yerine **sun** ve **vt102** kelimelerini deneyiniz. Hala başarılı olamıyorsanız bir bilene danışmalısınız.

- Aşağıdaki mesajlardan birini görürseniz :

```
[Read Only]           veya
File is Read only     veya
Permission denied
```

söz konusu dosyaya ya da bulunduğu dizine yazma yetkiniz yok demektir. Normal olarak yapabileceğiniz bir şey olmadığından **vi** seansını bir an önce durdurup, dosyanın sahibi ile (veya sistem yöneticisi ile) görüşmeniz gerekmektedir.

- Aşağıdaki mesajlardan birini görürseniz :

```
Bad file number            veya
Block special file         veya
Directory                  veya
Executable                 veya
Non-ascii file             veya
file non-ASCII
```

söz konusu dosya, ya normal bir dosya değildir ya da üzerinde edit işlemi yapılamayacak bir dosyadır; hemen **:q** komutu ile çıkınız.

-
```
File System full
```

Mesajını görürseniz, çalıştığınız disk veya disk kotanız dolmuş demektir; gereksiz dosyaları silerek yer açmanız gerekmektedir. Bu ilk defa başınıza geliyorsa sizden daha iyi UNIX bilen birinden yardım isteyiniz.

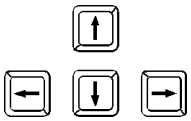
Disk Kotası :

UNIX işletim sisteminde, kullanıcıların disklerde kullanabilecekleri alanların toplam büyüklüğünü sınırlama (kota koyma) olanağı vardır. Böyle bir sınırlamanın olup olmayacağına; olacaksa her kullanıcı için kotanın kaç megabyte olacağına sistem yöneticisi karar verir.

Disk kotanızı doldurduğunuzda bir mesajla uyarılırsınız ve genellikle kotanızı biraz aşmanıza izin verilir; ancak bir kaç gün içinde tekrar kota limitlerinizin altına inmezseniz, sistem yöneticisi dosyalarınızdan bir kısmını silecektir.

vi editörü, ekranın tamamını kullanan **tam-ekran** *(full screen)* bir editördür. Bu nedenle temel işlevler için en iyi hakim olunması gereken komutlar, imleçi ekranda dolaştırma komutlarıdır.

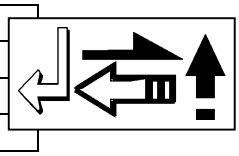
İmleç Dolaştırma Komutları



Bu komutların verilmesi sırasında, editör **komut** konumunda bulunmalıdır. (Yani; editör yeni başlatılmış olmalı veya ekleme konumundan çıkmak için **Esc** tuşuna basılmış olmalıdır. Hatırlayacaksınız, fazla **Esc** basmanın bir zararı yoktur; dolayısıyla bulunduğunuz konumla ilgili bir şüpheniz varsa, hiç çekinmeden iki kere **Esc** tuşuna basınız.)

Eğer kullandığınız terminalin özellikleri sisteme doğru tanıtıldıysa ve klavyenizde **ok tuşları** varsa, en temel hareketleri (birer karakter sağa, sola, yukarı ve aşağı) bu ok tuşları ile yapabilmelisiniz. Eğer terminal tanımlarınız hatalıysa; eksikse veya klavyenizde bu tuşlar yoksa, imleci gezdirme komutları şunlardır :

h	sola bir karakter
j	aşağı bir satır
k	yukarı bir satır
l	sağa bir karakter (küçük le harfi)



Tabii bu arada **RETURN** (veya **ENTER**) tuşu ile **BackSpace** tuşunun da sırasıyla satır başı ve bir geri anlamına geldiğini hatırlatmakta fayda var.

Bir çok **vi** komutu gibi, imleç hareket komutlarının da başına bir sayı koyarak, bu sayı kadar sağa, sola, aşağı ve yukarı hareket sağlanabilir.

3h	3 karakter sola
2k	2 satır yukarı, gibi.

Birerli adımlar dışında hareket sağlayan ve çok kullanılan komutlardan bazılarıysa:

0	(sıfır) imlecin bulunduğu satırın başına
$	imlecin bulunduğu satırın sonuna
w	bir sonraki sözcüğün başına
b	sözcüğün başına (ya da bir önceki)

Ekleme Komutları *(Inserting Text)*

Yazı yazarken en çok yapılan işlerden biri, eski bir metnin başına, sonuna ve araya satır/kelime/harf eklemektir. **vi** editöründe ekleme konumuna geçmenin yöntemlerinden birini (**i** komutu) daha önce belirtmiştim. Yine de tekrarlamak istiyorum :

i	imlecin, üzerinde bulunduğu karakterin hemen **solundan** başlayarak, **Esc** tuşuna basıncaya kadar basılan her karakteri metne ekler. Varsa, eski metin sağa doğru itelenir.

Diğer ekleme komutları :

a	imlecin, üzerinde bulunduğu karakterin hemen **sağından** başlayarak, **Esc** tuşuna basıncaya kadar, basılan her karakteri metne ekler. Varsa, eski metin sağa doğru itelenir.

A	imlecin, üzerinde bulunduğu **satırın sonundan** başlayarak, **Esc** tuşuna basıncaya kadar, basılan her karakteri metne ekler. Varsa, eski metin aşağı doğru itelenir.

Bu komutların büyük veya küçük harflerle verilmesi farklı ve önemlidir. Dikkatli olunuz..
.

Yazı Silme *(Deleting Text)*

Daha önce yazılmış metin parçalarını **silmek** için kullanılan komutlar :

x	imlecin üzerinde bulunduğu TEK karakteri sil
3x	imlecin üzerinde bulunduğu karakter dahil, sağa doğru 3 karakter sil
dw	imlecin bulunduğu yerden kelime sonuna kadar sil. Eğer imleç sözcüğün başındaysa, sözcüğü sil.
2dw	imlecin bulunduğu yerden başlayarak 2 sözcük sil
dd	imlecin bulunduğu satırı sil
2dd	imlecin bulunduğu satır dahil, aşağı doğru iki satır sil
D	imlecin bulunduğu yerden satır sonuna kadar sil (**d$** komutuna eşdeğerdir)
d$	Satırın sonuna kadar sil (**D** komutuna eşdeğerdir)

Yanlışlıkla bir silme işlemi yaparsanız ...

Eğer hatanızı hemen farkederseniz, **u** komutuyla *(undo)* son silme işlemini geri döndürebilirsiniz (**Aslında sadece son silme işlemini değil, son değişikliği geri döndürür**). Eğer son sildiğiniz metin parçasını geri getirmek isterseniz, hemen farketmeseniz bile, bu işlemi **p** komutu yardımı ile geri çevirebilirsiniz. Ancak bu geri getirme, metnin silindiği yere değil, imlecin **p** komutu verildiği andaki yerine yapılır. (Hiç yoktan iyi, değil mi?).

Metin bloklarının yerini değiştirme *(Moving Text)*

vi editörüyle metin bloklarının yerini değiştirmek istediğinizde kullanacağınız yöntem **kes-yapıştır** yöntemidir. Yerini değiştireceğiniz metin bloğunu önce bulunduğu yerden, uygun bir komutla (örneğin tek satır için **dd** komutu gibi) silmeli (kesme işlemi); daha sonra **p** komutuyla *(put)* yeni yerine yapıştırmalısınız. **Eğer yapmak istediğiniz işlem, bir metin bloğunu silmekse, kesme işleminden sonra başka bir yere yapıştırmamanız yeterlidir**. Bu işlemleri bir örnekle anlatmak daha kolay. Aşağıdaki örnek metindeki, **'istediğinizde, once.....'** kelimeleri ile başlayan ikinci satırı, **'Ayni Windows ...'** kelimeleri ile başlayan satırın arkasına taşımak istediğimizi varsayalım : Önce imleci bu satırın üzerine getirmeliyiz (**dd** komutuna hazırlık)

```
vi editoru ile metin parcalarinin yerini degistirmek

istediginizde, once eski yerinden silmeli, daha

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

Oldukca kolay, degil mi ?
```

dd komutunu verdiğimizde, bu satır ekrandan kaybolacak ve geçici bir bellek sahasına alınacaktır.

```
vi editoru ile metin parcalarinin yerini degistirmek

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

Oldukca kolay, degil mi ?
```

Daha sonra imleci bir aşağı satıra indirerek ....
j (Tabii, eğer çalışıyorsa, aşağı ok tuşunu da kullanabilirsiniz.)

```
vi editoru ile metin parcalarinin yerini degistirmek

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

Oldukca kolay, degil mi ?
```

p (küçük p) komutu ile yeni yerine yapıştırabiliriz.

```
vi editoru ile metin parcalarinin yerini degistirmek

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

istediginizde, once eski yerinden silmeli, daha

Oldukca kolay, degil mi ?
```

Bu noktada size bir problem sunmak istiyorum. Metnin herhangi bir yerinde, iki harfin yerini değiştirmenin en kısa yolu nedir ?

x ve **p** komutlarını bir arada kullanarak (**xp** şeklinde) bu işi yapabilirsiniz. Lütfen deneyiniz !

Metin Bloklarını Kopyalama *(Copying Text)*

vi editörü ile metin bloklarını kopyalamak istediğinizde kullanacağınız yöntem **kopyala-yapıştır** yöntemidir. Kopyalayacağınız metin bloğunu önce bulunduğu yerde, **y** *(yank)* komutuyla (örneğin, tüm satır için **yy** komutu veya **Y** gibi) geçici belleğe aktarmalı ve daha sonra kopyalanacağı yere **p** komutu *(put)* ile yapıştırmalısınız.

Bu işlemleri bir örnekle anlatmak daha kolay olacak galiba... Aşağıdaki örnek metindeki, **'istediğinizde, once...'** kelimeleri ile başlayan ikinci satırı, **'Ayni Windows ...'** kelimeleri ile başlayan satırın arkasına kopyalamak istediğimizi varsayalım :

Önce imleci bu satırın üzerine getirmeliyiz (**yy** komutuna hazırlık)

```
vi ile metin parcalari baska yerlere kopyalanmak

istendiginde, once eski yerinde bellege alinmali, daha

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

Oldukca kolay, degil mi ?
```

yy komutunu verdiğimizde, bu satır **geçici bir bellek sahasına kopyalanacaktır**. Ekranda bir değişiklik olmayacaktır.

```
vi ile metin parcalari baska yerlere kopyalanmak

istendiginde, once eski yerinde bellege alinmali, daha

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

Oldukca kolay, degil mi ?
```

yy komutundan sonra ekranda bir değişiklik olmaz

Daha sonra imleci iki aşağı satıra indirerek ....

jj (Tabii, eğer çalışıyorsa, aşağı ok tuşunu da kullanabilirsiniz.)
(Veya **2j**)

```
vi ile metin parcalari baska yerlere kopyalanmak

istendiginde, once eski yerinde bellege alinmali, daha

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

Oldukca kolay, degil mi ?
```

p (küçük p) komutu ile yeni yerine yapıştırabiliriz.

```
vi ile metin parcalari baska yerlere kopyalanmak

istendiginde, once eski yerinde bellege alinmali, daha

sonra yeni yerine yapistirmaliyiz.

Ayni Windows cut & paste gibi...

istediginizde, once eski yerinde bellege alinmali, daha

Oldukca kolay, degil mi ?
```

Kopyalama ve yer değiştirmede kullanılan **p** (küçük p) komutu yerine **P** (büyük p) komutu kullanılırsa, yapıştırma, imlecin bulunduğu satırın altına değil, ÜSTÜNE yapılır.

Son Komutu Tekrarlama

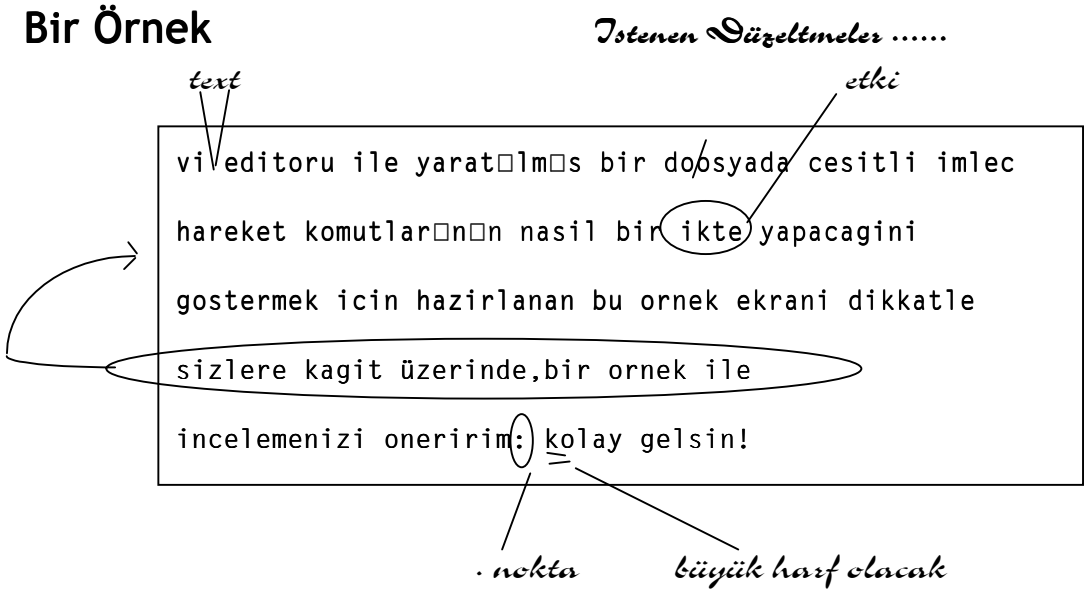
Bir nedenle, son **değişiklik** komutunuzu başka yerlerde de tekrarlamanız gerekirse, **.** (nokta) komutu ile bunu yapabilirsiniz.

Metin Eklemenin / Değiştirmenin Bir Kaç Değişik Yolu

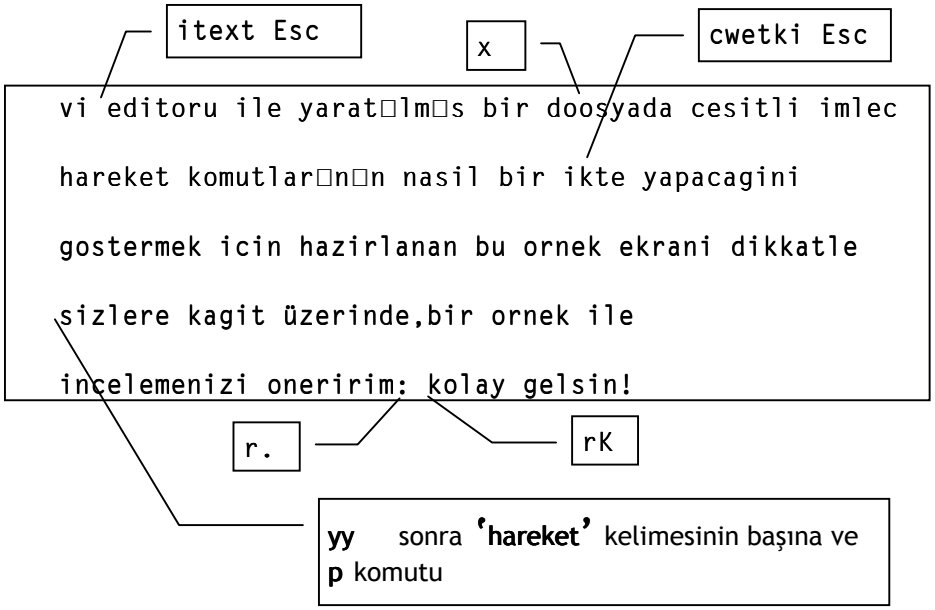
I	*(Insert)*	(Büyük **i**) İmlecin bulunduğu satırın BAŞINA eklemeye başla. (**Esc**'e kadar)
o	*(Open Line)*	(Küçük O) İmlecin bulunduğu satırın ALTINA bir boş satır aç ve oraya eklemeye başla.
O	*(Open above)*	(Büyük O) İmlecin bulunduğu satırın ÜSTÜNE bir boş satır aç ve eklemeye başla.
s	*(Substitute Char)*	İmlecin bulunduğu yerdeki KARAKTERİ sil ve yerine yeni metni eklemeye başla . (**Esc**'e kadar)
S	*(Substitute Line)*	İmlecin bulunduğu SATIRI sil ve yerine yeni metni eklemeye başla.
r	*(Replace Char)*	İmlecin bulunduğu karakteri bir sonra basılacak karakterle değiştir.
R	*(Replace Text)*	İmlecin bulunduğu noktadan itibaren, yeni metni eski metnin ÜZERİNE yerleştir. (**Esc**'e kadar)
J	*(Join)*	(Büyük **j**) İmlecin bulunduğu satırla arkasındaki satırı BİRLEŞTİR.
cw	*(Change Word)*	İmlecin buluduğu sözcüğü, yeni girilecek sözcükle değiştir..

Bir sonraki sayfada, daha önceden hatalı olarak dökümü yapılmış bir metin üzerinde kalemle yapılmış düzeltmelerin dosyaya işlenmesini canlandıran bir örnek bulacaksınız. Dikkatle incelemenizi öneririm.

Bir Örnek



..... ve bu düzeltmeleri yapmak için gereken işlemler



Düzeltilmiş Metin

```
vi text editoru ile yarat□lm□s bir dosyada cesitli imlec

hareket komutlar□n□n nasil bir etki yapacagini

sizlere kagit üzerinde,bir ornek ile

gostermek icin hazirlanan bu ornek ekrani dikkatle

incelemenizi oneririm. Kolay gelsin!
```

Bir çok **vi** komutu, başına bir **çarpan** yerleştirerek birden fazla kez tekrarlanabilir. (Aynı cebirdeki gibi...)

Örneğin,

Satır silmek için kullanılan **dd** komutu, **5dd** şeklinde verildiğinde '5 satır sil' anlamına gelir.

Bir satır aşağı inmek için kullanılan **j** komutu yerine **5j** komutu verilirse, bilin bakalım kaç satır aşağı inilir ? (Tabii aşağıda o kadar satır varsa !)

Komutların bu özelliğini daha önceki örneklerde farketmiş olmalıydınız. Eğer bu **çarpan** özelliğini şu ana kadar farketmediyseniz, pek dikkatli okumuyorsunuz demektir.

Metnin İçinde Dolaşma

Şu ana kadar hep bir kaç satırdan oluşan dosyalarda çalıştık. Hayat her zaman bu kadar kolay değildir. Özellikle elektronik posta için metin yazarken ya da program geliştirirken, sık sık metin içinde sayfa sayfa (bir başka deyişle ekran-ekran) ileri-geri gitmek gerekir. Bu işlemler için kullanılan komutları burada kısaca bir tablo halinde sıralayacağım. Birer kere denemenizi öneririm. Üzerinde çalışmak için uzun bir dosya hazırlamak yerine, sisteminizin **/etc** dizinindeki **termcap** dosyasını kendi **home** dizininize kopyalayıp bu dosya üzerinde çalışabilirsiniz. (**cp /etc/termcap ~**)

vi KOMUTU	İŞLEVİ
CTRL F *(Forward)*	Bir ekran İLERİYE
CTRL B *(Backward)*	Bir ekran GERİYE
CTRL D *(Down)*	Yarım Ekran İLERİYE
CTRL U *(Up)*	Yarım Ekran GERİYE
CTRL R *(Redraw)*	Ekranı yeniden düzenle (Çalışırken bir başka kullanıcıdan mesaj gelirse, ekranınız bozulacaktır. Bu komut, ekranı silip yeniden oluşturarak düzenlenmiş olur)
CTRL Y	Ekranı bir satır aşağı kaydır, imleç yerinde kalsın.
CTRL E	Ekranı bir satır yukarı kaydır, imleç yerinde kalsın.
z RETURN	(Küçük z) İmlecin bulunduğu satır ekranın EN ÜSTÜNE gelecek şekilde ekranı düzenle
z.	(Küçük z ve nokta) İmlecin bulunduğu satır EKRANIN ORTASINA gelecek şekilde ekranı düzenle
z-	(Küçük z ve eksi) İmlecin bulunduğu satır EKRANIN EN ALTINA gelecek şekilde ekranı düzenle
H *(Home)*	EKRANIN EN ÜST satırına git.
M *(Mid Screen)*	EKRANIN ORTA satırına git
L *(Lower Screen)*	EKRANIN EN ALT satırına git
RETURN	Bir sonraki satırın ilk karakterine git.

Yukarıda açıklanan komutların bazı özel kullanımları vardır. Bunlar az kullanılan özellikler olup, burada sadece bir fikir vermek amacı birkaç örnek vereceğim.

200z RETURN	200 üncü satırı ekranın en üstüne getir.
4H	Ekranın en üst satırının 4 altındaki satıra git.
5L	Ekranın en alt satırının 5 üstündeki satıra git.

Metnin İçinde Arayarak Dolaşma *(Text Search)*

Diyelimki program yazıyorsunuz ve değişiklik yapmak istediğiniz satırın yeri hakkında bir fikriniz yok. Tek hatırladığınız, değiştirmek istediğiniz satırda '**kayit_sayısı++**' diye bir karakter dizisi var. Bu diziyi içeren bir satır bulmak için

```
/kayit    RETURN
```

tuşlarına basarak bir komut yazmanız yeterli olacaktır. İmleç, içinde bu dizi geçen ilk satırda duracaktır.

Arama, imlecin bulunduğu yerden ileriye doğru yapılır. Eğer geriye doğru arama yapmak isterseniz **/** yerine **?** karakterini kullanmanız gerekir.

Diyelim ki, programın ilk rastladığı **kayit** sözcüğü, sizin ilgilendiğiniz değil ve aramaya devam etmek istiyorsunuz. Bu durumda **n** *(next)* tuşuna basmanız yeterlidir. Tüm arama komutunu yeniden yazmanız gerekmez. Eğer aramanın yönünü değiştirmek isterseniz **n** yerine **N** tuşuna basınız. Eğer aramanın halen hangi yönde olduğunu hatırlamıyor, fakat aramayı ileriye doğru yöneltmek istiyorsanız **/** tuşuna, tam tersi içinse **?** tuşuna basabilirsiniz.

Bulup Değiştirme *(Find & Replace)*

Diyelim ki bulunduğunuz satırda **Unix** olarak yazılmış bir sözcük var ve UNIX geleneklerine uymayan bu sözcüğü **UNIX** olarak değiştirmek istiyorsunuz. Bu işi yapmak için imleci **n** harfinin üzerine götürerek **3rNIX** komutunu vermeniz yeterli olacaktır. Bu yöntemde bulma görevini siz; değiştirme görevini ise **vi** üstlenmiş oluyor. Her iki işi de **vi**'ın yapmasını istiyorsanız değişikliğin yapılmasını istediğiniz satırın üzerine gelip

```
:s/Unix/UNIX                veya
:s/nix/NIX
```

komutlarını verebilirsiniz. DİKKAT! Bu komut sadece İLK RASTLADIĐI **Unix** karakter dizisini **UNIX** dizisiyle (veya **nix** dizisini **NIX** dizisiyle) değiştirecektir.

Bir satırdaki tüm **Unix** karakter dizilerini **UNIX** olarak değiştirmek için

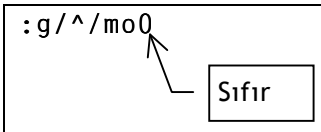
```
:s/Unix/UNIX/g
```

komutunu kullanmalısınız.

Şimdi, **s** komutuna bir kaç gelişmiş örnek vererek bu komutun başka marifetlerini de göstermek istiyorum:

vi komutu	Anlamı
`:1,100s/Unix/UNIX/g`	Dosyanın 1. ve 100. satırları arasında rastlanan tüm **Unix** dizilerini **UNIX** olarak değiştir.
`:1,$s/Ugur/Ugur Ayfer/g`	Dosyanın 1. ve sonuncu satırları arasında rastlanan tüm **Ugur**'ları **Ugur Ayfer** olarak değiştir.
`:%s/Ugur/Ugur Ayfer/g`	**%** işareti **tüm dosya** anlamına gelir.
`:%s/teh/the/gc`	Dosyadaki tüm **teh**' leri **the** olarak değiştirir; ancak her bir değişiklik için kullanıcı onayı ister. *(c : confirmation).*

`:g/Ayfer/s/Ugur/U./g`	Tüm dosyada '**Ayfer'** dizisini arar (g/Ayfer/); bulduğu her satırdaki tüm '**Ugur**' dizilerini '**U.**' ile değiştirir.
`:%s/Fortran/\U&/g`	Dosyadaki tüm '**Fortran'** dizilerini '**FORTRAN'** ile değiştirir (**\U** : büyük harfe dönüştürür; **&** ise, aranan diziyi baştan yazmamak için bir kısaltma)
`:g /^$/d`	Tüm boş satırları bulur ve siler. (**^** işareti satır başı anlamına; **$** ise satır sonu anlamına gelir. Bu örnekte içinde hiç bir karakter olmayan satırlar silinecektir; boşluk karakterleri içeren satırlar bu kalıba uymayacağından silinmeyecektir. Eğer içinde boşluk karakterleri olan satırları da silmek istiyorsanız **:g/^ *$/d** komutunu kullanmalısınız. (**^** işaretinden sonraki boşluğa dikkat!)



Çılgın bir örnek !
Bir dosyadaki satırların sırasını ters çevirir... (Son satırı birinci satır, sondan ikinci satırı ikinci satır, ...)

Nasıl mı? Başlangıcı olan her satırı (^) (zaten her satırın bir başlangıcı vardır) sıfırıncı satırın altına taşır. (**mo** : *move* anlamındadır). Bu kısacık komutun bu işi yapacağına inanmıyorsanız bir deneyin. Ben bu örneğe ilk rastladığımda çalışacağına inanmayıp hemen denemek ihtiyacını duymuştum.

Metnin İçinde Satır Numaralarını Kullanarak Dolaşma

vi editörü, normalde, ekranda satır numaralarını göstermez. Eğer metnin içinde satır numaralarını kullanarak dolaşmanız gerekiyorsa (40. satıra git, sonra 75. satıra git, vs), ekranda satır numaralarını görmek çok yararlı olacaktır.

Satır numaralarını ekranda görebilmek için :

`:nu` *(Bulunduğun satırın numarasını göster) veya*

`:#`

Bütün satırların numaralarını ekranda görmek için

`:set number`

Artık istediğiniz ve sıra numarasını bildiğiniz bir satıra gitmek için kullanacağınız komut :

`nnG` *(nn numaralı satıra git, örneğin 55G, 134G)*

Tuş Kısaltmaları *(Abbreviations)*

Metninizi yazarken bazı kelime ya da kalıpları çok sık tekrarlamanız gerekebilir. Örneğin, metninizin bir çok yerinde 'Aircraft Owners and Pilots Association' adlı organizasyonun adını yazmanız gerekecekse

```
:ab aopa Aircraft Owners and Pilots Association
```

komutuyla bir kısaltma ***(abbreviation)*** tanımı yapabilirsiniz. Artık, klavyeden her **aopa** yazdığınızda, sanki açık açık **'Aircraft Owners and Pilots Association'** yazmışsınız gibi kabul edilecektir. Bu kısaltmanın iptal edilmesini istediğiniz zaman

```
:unab aopa          (unabbreviate)
```

komutu yetecektir. (**vi**'dan çıktığınızda da kısaltma yok olacaktır.)

ab komutu ile yapılan tuş tanımlamaları, sadece **insert** modunda; yani araya metin girme konumunda anlamlıdır (tanımlamış olduğunuz bir kısaltmayı kullanmadan önce **i**, **a**, **A**, **o** veya **O** komutlarından birini vermiş olmanız gerekir).

Eğer sık kullandığınız **vi** komutlarına ilişkin bir kısaltma tanımlamak istiyorsanız, bu tanımlamanızı **map** komutuyla yapmanız gerekir. Örneğin,

```
map ^Y dd
```

tanımlaması; Ctrl-Y tuşunun, bulunduğunuz satırın silinmesini sağlayan **dd** komutu ile eş anlamlı olarak kullanılmasını sağlar.

Eğer **map** komutu ile F2 **fonksiyon tuşuna** 'tüm boş satırları silme' komutunu tanımlamak isterseniz

```
map #2 :g/^/d
```

komutunu kullanabilirsiniz. **map** ile yapılmış tanımlamaları iptal etmek için **unmap** komutunu kullanmalısınız. (**unmap #2** gibi).

Bu tip kısaltmalarınızın kalıcı olmasını istiyorsanız **.exrc** dosyası ile ilgili bölümü okuyunuz.

vi Başlatırken Verebileceğiniz Komutlar *(Startup Commands)*

Çok önemli olmamakla birlikte, editörü başlatırken komut satırından verebileceğiniz bir kaç **vi** komutu vardır.

`% vi +230 mektup1.mail`	**mektup1.mail** dosyasını aç ve imleci 230. satıra götür.
`% vi + telefonlar`	**telefonlar** dosyasını aç ve imleci son satıra götür.
`% vi + /Hasan telefonlar`	**telefonlar** dosyasını aç ve imleci içinde **Hasan** geçen ilk satıra götür.

Dosya İşlemleriyle İlgili Komutlar

vi editörü ile yarattığınız veya üzerinde çalışarak değişiklikler yaptığınız dosyaları diske geri kaydetmek ve buna benzer işlemler için kullanılan komutlar aşağıda sıralanmıştır.

Bu komutları kullanmadan önce komut düzeyine geçmiş olmanız gerekmektedir: komut düzeyine geçmek için en az bir kez **esc** tuşuna basınız.

`ZZ`	Dosyayı son haliyle diske kaydet ve **vi**'dan çık (**:wq** komut dizisine eşdeğerdir).
`:q`	Dosyada değişiklik yapılmayacak, **vi**'dan çık ***(quit)***
`:q!`	Yapılan değişikliklerden vaz geçildi, dosyayı değiştirmeden **vi**'dan çık ***(quit)***
`:w`	Dosyayı diske kaydet (**vi**'da kal) ***(write)***
`:wq`	Dosyayı diske kaydet ve vi'dan çık ***(write - quit)***
`:x`	**vi**'dan çık, değişmişse dosyayı diske kaydet ***(exit)***
`:wdosya2`	Üzerinde çalışılmakta olan dosyayı, **dosya2** adıyla diske kaydet. ***(write)***
`:1,100wbolum1`	Üzerinde çalışılmakta olan dosyanın ilk 100 satırını **bolum1** isimli bir dosyaya kaydet.

`:rdosya3`	**dosya3** adlı dosyayı oku ve imlecin bulunduğu noktadan başlayarak ve araya ekle. ***(read)***

vi İçinden UNIX Komutu Verme

Bazen, vi programıyla, bir dosya üzerinde çalışırken, geçici olarak **kabuğunuza** dönüp, başka bir UNIX komutu çalıştırmanız gerekebilir. Diyelim ki, üzerinde çalışmakta olduğunuz dosyanın içine, bir başka dosyayı kopyalamanız gerekti, ama bu dosyanın tam adını hatırlayamadınız. **ls** komutunu bir kullanabilseniz, bu dosyanın adını hemen hatırlayacaksınız. Böyle durumlar için, **vi**, size kabuk programınıza bir çıkış olanağı vermektedir. Bu olanaktan yaralanabilmek için **Esc** tuşu ile komut düzeyine geçip

```
!ls
```

komutunu veriniz.

Bir başka örnek : Size gelen bir mesaja cevap yazıyorsunuz, ama bir an için o mesaja tekrar bir göz atmak istediniz; hemen

```
!mail
```

komutu ile geçici olarak **mail** programına girebilirsiniz.

UNIX'in zerafetine bir örnek :

Diyelim ki, bir program için kullanım kılavuzu yazıyorsunuz ve kılavuzunuzun bir bölümüne, söz konusu programın bir çıktısını eklemek istiyorsunuz.

Kullanabileceğiniz komut

```
:r !prog
```

Bu komutu verdiğinizde, **prog** isimli program çalıştırılacak ve çıktısı **vi** ile edit etmekte olduğunuz dosyada, imlecin bulunduğu noktaya yerleştirilecektir. (Sanki diskten bir dosya okumuşsunuz gibi...)

Birden Fazla Dosyayı Peşpeşe İşleme

vi programını başlatırken, dosya adı olarak birden fazla parametre verebilirsiniz. Örneğin

```
% vi  dosya1  dosya2        veya
% vi  dosya*
```

Bu durumda, **vi** önce **dosya1** isimli dosyayı edit edilmek üzere ekrana getirecektir. Bu dosyayla işiniz bitip de

`w` komutuyla birinci dosyayı (**dosya1**) kaydettikten sonra
`n` komutuyla ikinci dosyaya, (**dosya2**) geçebilirsiniz.

.exrc DOSYASI

.exrc dosyası, **vi** programıyla ilgili özel tercihlerinizi belirttiğiniz dosyadır. Eğer **home** dizininizde **.exrc** isimli bir dosya (dosya adının başındaki noktaya dikkat ediniz) varsa, **vi** programını her başlattığınızda, bu dosyanın içindeki **vi** komutları otomatik olarak çalıştırılacak ve böylece tercihleriniz ayarlanmış olacaktır. **.exrc** dosyası basit bir text dosyası olup, **vi** dahil her türlü editörle yaratılabilir. Bu dosyada yer alabilecek bazı **vi** komutlarına örnekler vermek istiyorum.

```
map ^Y dd                    Ctrl-Y "satır sil" anlamında
map ^2 :g/^$/d               F2 tuşu, "boş satırları sil"
                             anlamında
ab cua Can Ugur Ayfer        "cua" bir kısaltma olarak kullanılmış
ab idg International Data Group
```

Daha Detaylı Bilgi İçin

vi programı hakkında daha yazılabilecek çok şey var. Eğer ilginizi çekiyorsa, O'Reilly & Associates yayınevinin **Learning the vi Editör** (yazarı Linda Lamb; ISBN 0-937175-67-6) isimli kitabını hararetle tavsiye ederim.

vi Komutları Özeti

A	Satır sonuna eklemeye başla	x	İmlecin bulunduğu karakteri sil
a	İmlecin sağına eklemeye başla	dd	İmlecin olduğu satırı sil
I	Satır başına eklemeye başla	d3	İmlecin olduğu yerden 3 karakter sil
i	İmlecin sağına eklemeye başla	d$	İmleçten sonrasını sil
O	Bu satırın üstüne satır ekle	u	**undo** (son değişikliği iptal et)
o	Bu satırın altına satır ekle	U	Satır için Undo

b	Bir önceki kelimenin başına git	r	Tek karakter değiştir
w	Bir sonraki kelimenin başına git	cw	Kelime değiştir
e	Bir sonraki kelimenin sonuna git	cc	Satırın tamamını değiştir
$	Satır sonuna git		
0	Satır başına git	H	Ekranın başına git
^	Satır başına git	L	Ekranın sonuna git
h	Sola bir karakter git	^B	Sayfa geri (Ctrl-B)
j	Aşağı bir satır git	^F	Sayfa ileri (Ctrl-F)
k	Yukarı bir satır git	[[	Dosya başına git
l	Sağa bir karakter git	]]	Dosya sonuna git

tx	Satırdaki bir sonraki x 'e git	<esc>	Insert konumundan çık
fx	Satırdaki bir sonraki x'i bul	:	vi komut satırına git
S	Satırın tamamını değiştir	:w	Dosyayı yaz (w yeni-isim de olabilir)
/dizi	Dosyada '**dizi**' yi bul (ileriye doğru)	:q	Quit
?dizi	Dosyada '**dizi**' yi bul (geriye doğru)	zz	Sakla ve çık
:nu	Satırları numarala	:q!	Değişikliklerden vazgeç ve çık

Bu sayfanın fotokopisini çekinmeden çekebilirsiniz. Dava açmam...

Blok taşımak için

- Blok başına gidiniz
- **8dd** ile (örneğin 8) satır siliniz (Windows'daki *kes* gibi)
- Taşıyacağınız yere gidip, **p** komutunu veriniz. (Windows'daki *yapıştır* gibi)

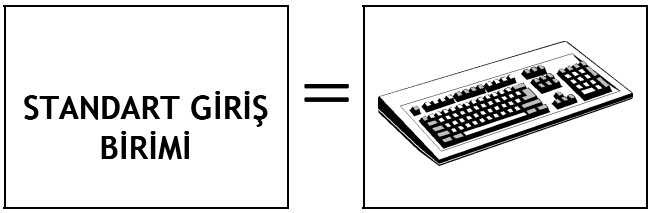
Blok kopyalamak için

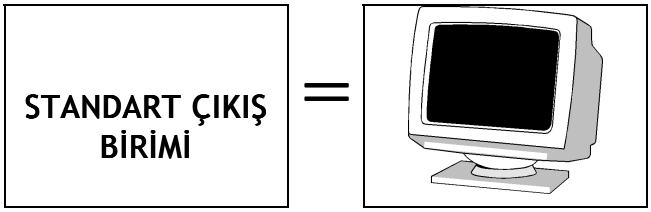
- Blok başına gidiniz
- **8yy** ile (Örneğin 8) satır alınız (Windows'daki *kopyala* gibi)
- Taşıyacağınız yere gidip, **p** komutunu veriniz (Windows *paste* gibi)

STANDART GİRİŞ ve STANDART ÇIKIŞ

(Standard Input & Standard Output)

Standart Giriş ve Standart Çıkış, UNIX işletim sisteminin çok önemli iki kavramıdır. UNIX komutlarının yüzde doksanı, işlevlerini **standart giriş** biriminden okuyacakları veriler üzerinde yerine getirip, varsa sonuçlarını **standart çıkış** birimine gönderir. Bir başka deyişle, UNIX komutlarının yüzde doksanı, işlevlerini **klavyeden** okuyacakları veriler üzerinde yerine getirip, varsa sonuçlarını **ekrana** gönderir.





Verilerini standart girişten okuyup, çıktısını standart çıkış birimine gönderen ve çok sık kullanılan bir UNIX programı, daha önce de sözünü etmiş olduğum **cat** programıdır.

Parametresiz kullanıldığında, bu program, **standart girişi standart çıkışa** kopyalar. Bir başka deyişle, **cat** programını parametresiz başlattığınızda, klavyeden yazdığınız her şey aynen ekrana kopyalanır. **cat** komutunun tanımına göre, bu kopyalama işi giriş dosyasının sonuna kadar devam edecektir. Şimdi size ilginç ve önemli bir soru...

Standart giriş birimindeki dosyanın (klavyenin) sonu ne demektir ?
(End Of File on Standard Input)

Bu sorunun yanıtı şöyle : Giriş dosyasının (klavyenin) kontrolü sizde olduğuna göre, giriş dosyasının (klavyeden yazacaklarınızın) sonunu belirlemek te size düşmektedir. Standart giriş dosyasının sonunu belirlemek için **Ctrl-D** tuşuna basmanız yeterlidir. (Bazı programlar sadece satır başında Ctrl-D tuşuna basıldığında bunu dosya sonu olarak yorumlarlar.)

Standart girişte dosya sonunu belirtmek için gereğinden fazla Ctrl-D basmamaya dikkat ediniz. Fazladan basacağınız bir Ctrl-D, kullandığınız kabuk programı tarafından standart girişin sonu olarak algılanabilir (kabuk programınız girişini doğal olarak klavyeden beklemektedir). Bu durumda da, artık daha fazla komut vermek istemediğiniz kabul edilip **logout** etmiş sayılabilirsiniz.

Eğer kabuk programınız c-shell (**csh**) ise,

```
% set ignoreeof                    (ignore end of file)
```

komutunu vererek, (daha da iyisi, bu komutu **home** dizininizdeki **.cshrc** dosyanıza yerleştirerek) Ctrl-D tuşunun **logout** komutu olarak yorumlanmasını önleyebilirsiniz.

.cshrc dosyasında nasıl mı değişiklik yapacaksınız? **vi** editörüyle ilgili bölümü okumadınız mı? Eğer biraz yardım isterseniz...

cd *(parametresiz kullandığınız için çalışma dizininizi home dizininiz olarak değiştirir.)*
vi .cshrc
gereği kadar aşağı ok tuşu
ekleme yapmak istediğiniz noktaya gelince **i** *tuşu*
set ignoreeof *ve RETURN*
Esc *tuşu*
:wq *komutu*

Yaptığınız değişikliğin etkisini görmek için, önce bir **logout** ve tekrar **login** etmeniz gerekir.

cat komutunun bu formda (parametresiz olarak) kullanılması doğal olarak pek bir işe yaramaz. Ancak , **cat**, UNIX **GİRİŞ / ÇIKIŞ YÖNLENDİRME** kavramları ile bir arada kullanıldığında oldukça kullanışlı bir programa dönüşmektedir.

GİRİŞ ve ÇIKIŞ YÖNLENDİRME *(Input & Output Redirection)*

Giriş ve Çıkış yönlendirme, MS-DOS deneyimi olan kullanıcılar için pek yabancı sayılmaz. Her ne kadar MS-DOS işletim sistemi altında bu kavramlar pek sık kullanılmasa da, bir ara mutlaka karşılaşmış olmalısınız. Gene de, hatırlatma amacıyla, bu konuda bir kaç söz etmek istiyorum :

Diyelim ki, içinde bir kaç kelime olan bir dosyaya gereksiniminiz var. Örneğin, bilgisayar ağı bağlantınızla ilgili olarak, kullandığınız bilgisayarın adını tanımladığınız **/etc/hostname.le0** (bu dosyada sadece bilgisayarınızın adından oluşan tek bir sözcük olmalıdır) dosyasını yaratmanız gerekiyor. Bu kadarcık bilgi girmek için **vi** veya benzeri bir editör başlatmak yerine, **cat** programını kullanabilirsiniz.

Girişini klavyeden alacak olan bu basit kopyalama programının **çıkışını bir dosyaya yönlendirirseniz** amacınıza uygun olarak bir dosya yaratmış olursunuz. Şöyle ki :

```
# cat > /etc/hostname.le0
abc
Ctrl-D
#
```

(/etc dizinindeki dosyaları değiştirebilmek için root yetkilerine sahip olmanız gerekir.)

Aynı şekilde, programların girişini de yönlendirebilirsiniz...

Örneğin,

```
% sort < /tmp/sirasiz
```

sort komutu, tanımı gereği, girdisini (sıraya dizilecek satırları) standart giriş biriminden (klavyeden) alacak; sıralanmış satırlarıysa standart çıkış birimine (ekrana) listeleyecektir.

Örneğimizde yer alan **<** işareti, standart giriş biriminin **/tmp** dizinindeki **sirasiz** adlı dosyaya yönlendirildiğini; yani sıralanacak satırların bu dosyadan alınacağını belirtmektedir. Standart çıkışsa, yönlendirilmediğinden, sıralanmış satırlar (**sort** programının çıktısı) ekrana listelenecektir. Eğer sıralanmış satırları saklamak isterseniz, **sort** komutunu

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

şeklinde kullanmanız gerekir.

Biraz Nefes Alalım...

UNIX işletim sisteminde yüzlerce komut var! Bunların bir kısmı hiç kullanılmaz; bir kısmı pek az, bir kısmı da çok sık kullanılır. Genellikle hangi komutlardan yararlanacağınız tamamen bilgisayarı ne amaçla kullandığınıza bağlıdır. Tüm UNIX komutlarını bir kitaba sığdırmak olanaksız olduğu için ve okuyucunun UNIX kullanmaktaki amaçlarını bilemeyeceğim için, bu kitapta sadece kendi kullandığım komutlara yer verdim ve bu UNIX komutlarını alfabetik sıraya göre değil; kullanım amaçlarına göre gruplandırdım. Bir önceki bölümde biraz fazla teknik detaya kaçtığımın farkındayım; bu yüzden bu bölümde biraz nefes almak amacıyla, çok önemli olmayan, fakat kullanımı da hoş olan bir kaç komuttan söz etmek istiyorum. Bu komutlardan söz ederken kullanacağım genel form :

```
komut  [ -secenekler ] [ parametre] [ parametre] ...
```

Bu formda **[ ]** karakterleri arasında yer alan seçenek ve/veya parametrelerin isteğe bağlı olduğunu *(optional)* göstermektedir. Bir kaç örnek vermek gerekirse...

`cp [-ri] dosya1 dosya2`	**-r** veya **-i** veya **-ri** isteğe bağlı; **dosya1** ve **dosya2** ise zorunlu parametreler.
`rm [-ri] isim1 [isim2 ...]`	**-r** veya **-i** isteğe bağlı; en az bir **isim1** zorunlu parametre; **isim2** veya fazlası isteğe bağlı parametreler.

Hoş UNIX Komutları

```
% cal   [ay] [yil]                    (calendar)
```

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

```
% cal
   February 1995
 S  M Tu  W Th  F  S
          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
```

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

```
% cal 1995
```

```
                                1995

        Jan                     Feb                     Mar
 S  M Tu  W Th  F  S     S  M Tu  W Th  F  S     S  M Tu  W Th  F  S
 1  2  3  4  5  6  7                  1  2  3  4           1  2  3  4
 8  9 10 11 12 13 14     5  6  7  8  9 10 11     5  6  7  8  9 10 11
15 16 17 18 19 20 21    12 13 14 15 16 17 18    12 13 14 15 16 17 18
22 23 24 25 26 27 28    19 20 21 22 23 24 25    19 20 21 22 23 24 25
29 30 31                26 27 28                26 27 28 29 30 31

        Apr                     May                     Jun
 S  M Tu  W Th  F  S     S  M Tu  W Th  F  S     S  M Tu  W Th  F  S
                   1        1  2  3  4  5  6              1  2  3
 2  3  4  5  6  7  8     7  8  9 10 11 12 13     4  5  6  7  8  9 10
 9 10 11 12 13 14 15    14 15 16 17 18 19 20    11 12 13 14 15 16 17
16 17 18 19 20 21 22    21 22 23 24 25 26 27    18 19 20 21 22 23 24
23 24 25 26 27 28 29    28 29 30 31             25 26 27 28 29 30
30
```

Parametre olarak ay ve yıl girerseniz, o ay ve yıl için bir takvim yaprağı listelenir.

```
% cal 7 1995
    July 1995
 S  M Tu  W Th  F  S
                   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 29
30 31
```

% banner [text]

*(Denemek istediğinizde, komutu **/usr/5bin/banner** şeklinde vermeniz gerekebilir)*

Parametresi olarak verilen karakter dizisini blok harflerle geri isteler.

```
% banner  abc

   ##     ######     ####
  #  #    #     #   #    #
 #    #   ######    #
 ######   #     #   #
 #    #   #     #   #    #
 #    #   ######     ####
%
```

% sleep n

UNIX, **n** parametresi olarak verilen saniye kadar bekler. Sanırım 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 yarayabilir. Nitekim, bir SUN iş istasyonuna **login** ettiğinizde karşınızda

```
Please wait for Openwindows or hit Ctrl-C to return to C-shell
```

gibilerinden bir mesaj görebilirsiniz. İşte bu mesajı ekranda gördüğünüz sürece **.cshrc** dosyanızdaki **sleep 5** komutu çalışıyor olacaktır. Bu süre içinde Ctrl-C tuşu ile beklemeyi keserseniz, **openwin** programı çalıştırılmayacak ve kontrol kabuk programına dönecektir.

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

Parametresi olan dosyadaki satır, kelime 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, sadece satırları;
-w seçeneği verilirse, sadece kelimeleri;
-c seçeneği verilirse, sadece karakterleri sayar.

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

Örnekler :

```
% wc /etc/printcap
25      89    1271 /etc/printcap
% wc -l /home/ayfer/kitaplar
25 /etc/printcap
```

```
% at [-scm] saat [tarih] komut
```

Herhangi bir programın (**komut** parametresi olarak belirtilen); **saat** parametresi ile belirtilen zamanda başlatılmasını sağlar. İsteğe bağlı olarak tarih de belirtebilirsiniz.

Saat belirtirken **1330** veya

8a (sabah saatlerini belirten *am* anlamında) veya
3p (öğleden sonrayı belirten *pm* anlamında)

formlarını kullanabilirsiniz.
Tarih vermek isterseniz kullanmanız gereken form

aaa gg olmalıdır. Burada **aaa**, ay isimlerinin İngilizcelerinin 3 harfli kısaltmalarından biri olmalıdır *(Jan, Feb, ...*gibi).
gg ise doğal olarak gün...

-s seçeneği, programın çalıştırılması sırasında **sh**,
-c seçeneği, programın çalıştırılması sırasında **csh** kabuklarının kullanılmasını;
-m seçeneği ise, programın başarıyla çalıştırılması durumunda komutu veren kullanıcıya bir mesaj *(mail)* gönderilmesini sağlar.

Örnekler :

```
% at -c 2359 /usr/local/bin/backup
% at 1200 'mail reyyan < mesaj'
```

Bu son örnekte, saat 12:00 da, **mesaj** adlı dosyanın içeriğini, **reyyan** adlı kullanıcıya **mail** programı yardımıyla elektronik posta olarak yolluyoruz. **mail** programı, mesaj metnini standart giriş biriminden bekler; bu yüzden **<** işareti kullarak bu girişi mesaj adlı dosyaya yönlendirmemiz gerekiyor.
Ancak, komut yorumlayıcısının, **<** işaretinin **at** komutuna değil de **mail** komutuna ait olduğunu anlayabilmesi için **mail** komutunun tamamını tırmak içine almamız gerekmektedir. Alışıncaya kadar; yeni UNIX kullanıcılarının komut satırında yapacakları hataların çoğu bu **tırnak içine alma meselesiyle** ilgili olacaktır.

ÖNEMLİ UNIX KAVRAMLARI

Bir önceki bölümde söz ettiğim **Standart Giriş** ve **Standart Çıkış** kavramlarının yanısıra, UNIX işletim sisteminde, iyi kavranması gereken bir kaç önemli kavram daha vardır. Okuyucunun, UNIX işletim sistemine biraz ısındığını varsayarak bu kavramlardan üç tanesini daha açıklamak istiyorum. Bu kavramlar, ilk okuduğunuzda çok karışık ya da anlaşılmaz gelebilir; ama lütfen dikkatlice okuyunuz, gerek duyarsanız başka kaynaklara başvurunuz ama bu kavramları anlamadan geçmeyiniz.

Bu kavramların ilki **Dosya Sistemleri**... İngilizce; daha doğrusu UNIX'cesiyle ***FILE SYSTEMS***. İkincisiyse **Süreç kavramı**; UNIX'cesiyle ***PROCESS***. Bir üçüncü kavramsa, **bağlantılar** ***(LINKS)***.

Dosya Sistemleri *(File Systems)*

Şimdi biraz MS-DOS'a dönelim. MS-DOS işletim sisteminde dosya ve dizinlerin yerini belirtirken A:, C: gibi sürücü isimleri kullanılır. Eğer kişisel bilgisayar bir bilgisayar ağına bağlıysa, başka bilgisayarlar üzerinde yer alan disk sürücülerine ise F:, G: gibi isimlerle ulaşılabilir. UNIX'de durum biraz; hatta oldukça farklı.

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

Disklerinizden ilki büyük olasılıkla en az 3 parçaya (**partition**) bölünmüştür. (Bunu belki siz istemediniz ama bilgisayarınıza UNIX işletim sistemini yükleyen kişi böyle yapmak zorundaydı.) İkinci diskinizse büyük olasılıkla tek parçadır. Disket sürücünüzse, bildiğimiz, 1.44 MB kapasitede 3.5" disketleri kullanan bir sürücü olsun. CD-ROM sürücünüz için zaten pek fazla seçenek yok.

MS-DOS işletim sisteminden de hatırlayacağınız gibi, parçalara ayrılmış diskler (*partition*'lara ayrılmış diskler) sistemde, sanki her bir parça farklı bir fiziksel diskmiş gibi davranır. UNIX'de de böyledir. Yukarıdaki varsayımlarımıza göre bilgisayarımızın 4 diski varmış gibi düşünebiliriz (3 parçaya ayrılmış birinci disk ve tek parça olan ikinci disk). MS-DOS kullanıyor olsak, bu disklere C: D: E: ve F: isimleriye ulaşırdık.