

Kim Korkar UNIX'ten?

Can Uğur Ayfer

Aralık 1995

Tüm hakları PUSULA Yayıncılık'a aittir.
PUSULA Yayıncılık'ın izni olmadan
çoğlatılamaz ve alıntı yapılamaz.

İçindekiler

Önsöz	3
İşletim Sistemlerinin Kraliçesi : UNIX	5
UNIX'le Tanışma	11
Isınma Hareketleri	17
UNIX Dosya Yapısı	23
Dosyalar	33
UNIX'de Erişim Yetkileri	44
csh ve sh Kabukları	53
vi	63
Standart Giriş ve Standart Çıkış	86
Önemli UNIX Kavramları	93
Önemli UNIX Komutları	104
UNIX Pipe Kavramı	114
Yazıcı Kullanımı	116
Kabuklar : C Shell ve Shell	121
Kabuk Programlama	141
Çevreyi Tanıyalım	148
Teyp Kullanımı	164
Kullanışlı UNIX Komutları	180
UNIX Bilgisayar Ağları	190
Sistem Yöneticisine	206
Yedekleme	226
TCP/IP	234
Güvenlik	238
Sonsöz	240

ÖNSÖZ

Bugüne kadar 1000'e yakın sayıda farklı bilgisayar modeli üretildiği sanılıyor. Bu sayının içinde, binlerce değişik firma tarafından üretilen PC'ler tek bir model olarak yer almaktadır.

Bu kadar geniş donanım yelpazesi içinde yüzlerce değişik **işletim sistemi** geldi geçti. Adını bilgisayar tarihine altın harflerle yazdırmayı başarabilenlerden biri de UNIX oldu..

Oldu ama pek fazla da sempati toplayamadı. UNIX'le bir şekilde ilgilenen ya da ilgilenmek zorunda kalan pek çok kişiden duyduklarım genellikle UNIX'in sevimsiz, kullanması zor, kaprisli bir işletim sistemi olduğu doğrultusunda oldu. İtiraf edeyim ki, UNIX'le ilk tanıştığım 1983 yılında benim de görüşüm bu yöneydi.

Uzun yıllar ticari uygulamalarda, yalnızca "çok kullanıcılık" uğruna insanlar UNIX'e katlandılar. Derken, UNIX altında grafik ekran kullanımını sağlayan X-Windows ortaya çıktı; hemen ardından bilgisayar ağlarının ve doğal olarak Internet'in yıldızı parladı. İşte o zaman kullanıcılar ve programcılar UNIX'i bir daha değerlendirme gereksinimi duyular.

İçinde bulunduğumuz yıllarda UNIX çok önemli bir işletim sistemi! UNIX'le konuşamayan, TCP/IP desteği olmayan bilgisayar ağı yazılımları satamıyor; bir çok kişisel bilgisayar yazılımının UNIX uyarlamaları var. Kısacası UNIX'in gelişmesi ve yaygınlaşması hızlanmış durumda. Önümüzdeki bir kaç yıl içinde, mesleği bilgisayar kullanımı gerektiren herkesin, ucundan da olsa UNIX'e bulaşmadan çalışmasının olanaksız olacağı görüşü oldukça yaygın.

Bu durumda, bilgisayar dünyasına kişisel bilgisayarlarla adım atmış kullanıcı kitlesine UNIX'i tanıtmak ve hazırlıklı olmalarına yardımcı olabilmek amacıyla bu kitabı yazmaya başladım. Önce, DTK şirketince üretilmekte olan SPARC serisi iş istasyonları için notlar halinde ortaya çıkan bir döküman zamanla elinizdeki bu kitaba dönüştü.

Bu kitap UNIX hakkındaki her şeyi anlatmıyor; zaten sonlu sayıda sayfa kullanılarak UNIX hakkındaki her şeyi anlatmak da pek olası değil! Tek amacım, UNIX hakkında ön yargısı veya kötü deneyimleri olanlara UNIX'in kötü bir işletim sistemi olmadığını; aslında bir sanat eseri olduğunu; iyi kullanmayı bilen birisinin elinde neler yapabileceğini anlatmak. Ne demişler; at binenin ...

Kitapta anlatılanları izleyebilmek için, en azından MS-DOS işletim sistemi konusunda deneyimli olmanız gerekiyor. Bu kitap, bilgisayarlar hakkında genel bilgi arayışı içinde olan okuyucular için hiç de uygun değil.

Bütün bilgisayar kitaplarında olduğu gibi, bu kitabı da okurken, anlatılan komutları ve örnekleri kendi bilgisayarınızda denemelisiniz. Ancak, UNIX dünyasında bu henüz pek kolay değil. Nedeniyse, henüz evlerdeki bilgisayarlara UNIX'in girmemiş olması. Kitabın ekindeki disketin içinde, bu kitapta adı geçen UNIX komutlarının bir kısmının PC'lerde, MS-DOS altında çalışabilen modellerini bulacaksınız. "Modellerini" diyorum; çünkü bu MS-DOS programları, tam tamına UNIX karşılıklarının eşdeğeri değil. Ama, gene de, okuyucunun evindeki ya da bürosundaki kişisel bilgisayarda denemeler yapması için yeterli olacağı inancındayım.

Kitabın düzenlenişini biraz garip bulabilirsiniz. Bilgisayarın açılışını ve kapanışını kitabın ortasından sonra bir yerlerde anlattım. İlk bakışta, bu konuların en başta anlatılması gerekiyor gibi düşünebilirsiniz ama tipik UNIX kullanıcıları bilgisayarı hiç açıp kapatmazlar ki... Gene de kitabın düzeninin kusursuz olduğunu savunmuyorum. Konuları bana doğal geldiği şekilde sundum; ancak kitap bitince bir de baktım; bir UNIX referans kitabından çok, UNIX hakkında bir macera kitabına benzemiş. O nedenle korkarım başından sonuna kadar okumanız gerekecek. Aslında, UNIX ve komutları hakkında referans kitaplarını her yerde bulabilirsiniz; hatta ekranınızda bile...

Bazı konuların bir kaç yerde tekrarlandığını göreceksiniz. Bu tekrarları özellikle yaptım. UNIX'de bir komut ya da kavramın öneminin ilk karşılaşıldığında iyi anlaşılamayacağını biliyorum; kendim yaşadım. O nedenle, okuyucunun da başına aynı şeyin geleceğini düşünerek, bazı konuları, öneminin vurgulanabileceği bir yere gelince tekrarlamaktan kaçınmadım. Bu kitabı yazarken verdiği destek ve katkıları için eşim Reyhan Ayfer'e; müsveddeleri büyük bir dikkatle okuyan ve çok değerli katkılarda bulunan arkadaşım Lale Morgül'e ve bir çok yanlışımlı bularak düzelten oğlum Ömer Ayfer'e çok çok teşekkür ederim.

Can Uğur Ayfer
Kasım 1995, Ankara



İŞLETİM SİSTEMLERİNİN KRALİÇESİ UNIX

UNIX dünyasına hoşgeldiniz.

Nerelerde kaldınız? Hiç gelmeyeceksiniz sanmıştık...

Dünyada hiç bir işletim sistemi, UNIX kadar uzun ve sürekli gelişerek gündemde kalmayı başaramamıştır. IBM PC'ler için geliştirilmiş olan MS-DOS kadar yaygın olmamakla birlikte, dolaylı yoldan da olsa, UNIX işletim sisteminin hizmet vermekte olduğu kullanıcı sayısının, MS-DOS kullancılarının sayısına yakın olduğu sanılmaktadır.

UNIX işletim sistemi genellikle güçlü bilgisayarlarda kullanılmaktadır. UNIX felsefesinin temelinde, bir bilgisayarın birden fazla kullanıcı arasında paylaşılması; ya da bir kullanıcının aynı anda birden fazla iş yapmasına olanak sağlamak yatmaktadır. Bu nedenle, UNIX altında kullanılacak bilgisayarın, kaynaklarının birden fazla iş arasında paylaşılması durumunda performansını kabul edilebilir düzeyde tutabilecek güçte olması gerekmektedir. Bilgisayar teknolojisindeki gelişmeler, donanımları hızla güçlendirmekte ve ucuzlatmaktadır; bunun doğal sonucu olarak da, UNIX işletim sistemi denetiminde kullanılan bilgisayarların sayısı hızla artmaktadır.

Çok İş; Çok Kazanç...

UNIX İşletim Sistemi, bilgisayar bilimcilerinin '**çok kullanıcı**' (*multi-user*) ve '**çok iş**' (*multi-tasking*) adını verdikleri çalışma koşullarını sağlar. Bir başka deyişle; UNIX altında çalışan bir bilgisayarı, birden fazla kullanıcı birbirlerinden bağımsız olarak ve aynı anda kullanabilirler. Bu birlikte kullanım sırasında, bilgisayarın kaynaklarını (merkezi işlem birimini, ana belleğini (RAM), disk-teyp gibi yan bellek birimlerini, yazıcılarını) paylaşırlar. UNIX, kaynakların kullanımını, paylaşımından kaynaklanan performans düşmelerini en aza indirgeyecek şekilde düzenlemeye çalışır. Bu tür paylaşımlar, donanıma

yapılan yatırımı bir miktar azaltacağı için bir kazanç unsurudur. Yan bellek paylaşımıysa kayıtlı veri ve programları da paylaşmak demektir ki; bu da değeri oldukça yüksek başka bir kazançtır.

Bir kullanıcının aynı anda birden fazla iş yapabilmesi de bir başka kolaylıktır. Bilgisayarınızda uzun bir iş başlattığınızı varsayalım; ancak bu iş, her bir kaç dakikada bir sizin klavyeden müdahale etmenizi gerektirsin. Bu durumda, bu uzun işi başlatıp yemeğe gidemezsiniz. Tek iş düzeninde kullanım için tasarlanmış bir işletim sistemi kullanıyorsanız (MS-DOS gibi), söz konusu programın yaptığı iş tamamlanıncaya kadar bilgisayarın karşısında oturmak zorundasınız. Eğer bu işi UNIX altında çalışan bir bilgisayarda yapıyorsanız, uzun programınız bir yandan işinizi yaparken, siz öte yandan (gene aynı ekran ve klavyeyi kullanarak) bir başka iş yapabilirsiniz. Eğer başka işiniz yoksa, ikinci iş olarak bir oyun başlatıp, bekleme sürenizin biraz daha zevkli geçmesini sağlayabilirsiniz. Örneğin, bu tür beklemeelerde, **Internet** üzerinde bir gezintiye çıkabilirsiniz (internet : neredeyse tüm dünyaya yayılmış olan bilgisayar ağı, *Information Super Highway*).

UNIX Her Yerde Aynı UNIX...

İlk kez 1970 yılında ortaya çıkan UNIX işletim sistemi, ticari bir amaçla tasarlanmamıştı; bu yüzden, bu yeni işletim sistemine ilgi duyan tüm bilgisayar üreticilerine ve bilgisayarını UNIX desteği ile kullanmak isteyen herkese çok küçük bir ücret karşılığında dağıtıldı. Bu sayede, UNIX kısa sürede gelişti ve yayıldı. Bu gelişmelere katkıda bulunan bilgisayarlılar, UNIX'in ilk günlerinde ortaya atılan standartları gelenekleştirerek korudular. Böylece bir **UNIX Kültürü** ve sağlam bir **UNIX Geleneği** oluştu. Kullanıcılar açısından bunun anlamı oldukça basit : UNIX İşletim sistemini bir kez öğrendiniz mi, UNIX'le çalışan herhangi bir bilgisayarı kolaylıkla kullanabileceğiniz gibi; alışık olduğunuz komut ve kavramların yüzde 99'u farklı bilgisayarlarda bile aynen geçerli kalacaktır.

Çok İyi Tasarımlanmış Bir İşletim Sistemi

UNIX İşletim Sistemi'nin 25 yıllık bir geçmişi var. Bu süre bilgisayar endüstrisi için çok ama çok uzun. Son 25 yıl içinde bilgisayarlar çok değişti, gelişti, hızlandı, küçüldü; 25 yıl önceki donanım tasarımları çoktan unutuldu ama UNIX İşletim sistemi, ilk yıllarında sahip olduğu özellikler ve yeteneklerle dimdik ayakta duruyor. 25 yaşındaki yaşlı UNIX, (belkide sadece 'olgun' demek daha doğru) günümüz bilgisayar-larına çok kolay uyum sağladı ve bundan sonraki gelişmelere de rahatça ayak uydurabilecek gibi görünüyor.

Kraliçe, Çünkü Herkes Saygı Duyuyor

Bilgisayar dünyası, yaklaşık 50 yıllık tarihinin son 30-35 yılında, **İşletim Sistemleri**'ne bir çok örnek gördü geçirdi. Bunlardan bazıları çok başarılı oldu, bazıları özel uygulamalara hizmet etti ve ömrünü tamamladı, bazıları piyasaya çıkamadan yok oldu, unutuldu gitti. İşletim sistemleri genellikle donanım üreticileri tarafından, ürettikleri bilgisayar modelleri için özel olarak geliştirildiler. Söz konusu donanım modelleri ortadan kalktıkça, bu bilgisayarların işletim sistemleri de sahneden ayrıldılar.

UNIX için böyle olmadı; çünkü belirli bir marka veya model donanım için tasarlanmamıştı. Donanım modelleri geliştikçe, UNIX bu yeni platformlara uyarlandı ve eski deneyim, yazılım birikimleri zarar görmeden yeni bilgisayar nesillerine taşındı. Bu uyumluluğun yararını gören bilgisayar üreticilerinin neredeyse tamamı, işletim sistemi repertuarlarına UNIX'i eklemek zorunluluğunu hissettiler. Hatta bir çok bilgisayar üreticisi dev firma, kendi UNIX türevlerini geliştirdiler. AIX (IBM), ULTRIX (DEC), HPUX (HP), SINIX (SIEMENS) gibi...

Biraz da Tarih...

1960'lı yıllarda kullanılan bilgisayarlar, ancak 'Sıralı İş Düzeni'nde çalışabilmekteydi. (*Batch Processing*). Bir diğer deyişle, kullanıcılar ve programcılar, bilgisayarda yapmak istedikleri işle ilgili komut ve/veya programları bilgisayarın operatörüne teslim ederler ve sıranın kendi işlerinin yapılmasına gelmesini beklerlerdi. Bu sıra artık onbeş dakikada mı yoksa üç günde mi gelir, bilinmezdi.

Bu yıllarda, üç önemli kuruluş (AT&T, MIT Üniversitesi ve General Electric) bir arada yürüttükleri bir projeyle ilk '**Zaman Paylaşımlı İşletim Sistemi**' üzerinde çalışmaya başladılar. Proje, bir bilgisayarın bir anda birden fazla kullanıcıya hizmet etmesini sağlayan; kullanıcıların bilgisayar programında olup bitenleri izleyebileceği ve programlarla etkileşimli (*interactive*) olarak çalışabilecekleri bir ortam yaratmaya yönelikti. Çalışmalar sonunda **MULTICS** işletim sistemi ortaya çıktı (**MULT**iplexed **I**nformation and **C**omputing **S**ystem). Her şey akademik olarak çok iyiydi; fakat, MULTICS yazılımı, o zamanki bilgisayarlar için biraz büyük ve hantal kalıyordu.

MULTICS ekibiyle birlikte çalışan ve uzay araştırmalarında kullanılan benzetim (*simulasyon*) yazılımları üretmekte olan **Ken Thompson** hayatından pek memnun değildi. Proje arkadaşları, onun üzerinde çalıştığı programların sistem kaynaklarını çok zorladığından sürekli şikayet ediyorlardı. Bu yüzden, Thompson, sadece başkalarının bilgisayarı kullanmadığı zamanlarda çalışabiliyordu. Bu böyle devam edemezdi. Thompson, çalışmalarını kendisine ait olan eski ve küçük bir DEC PDP-7 bilgisayarında tamamlamaya karar verdi. Ama bu bilgisayarın işletim sistemi de gereksinimlerini karşılamıyordu; bu yüzden kendi istekleri ve gereksinimleri doğrultusunda bir işletim sistemi

geliştirmeye koyuldu. MULTICS'in yararlı bulduğu ve beğendiği özelliklerinin tümünü kullandı. Hatta, o kadar ki, UNIX isminin MULTICS den esinlenildiği; önce UNICS olarak konulduğu, sonradan UNIX'e dönüştürüldüğü **Brian Kernighan** (C Programlama dilini ve UNIX'i yaratan ekibin önemli isimlerinden) tarafından anlatılmaktadır.

1970 yılında UNIX işletim sisteminin ilk sürümü DEC PDP-7 modeli bir bilgisayarda tamamlanmıştı. İşletim sistemi, **programcılar için** yararlı olacak şekilde tasarlanmış ve özellikle metin işleme yetenekleri (*text processing*) oldukça gelişmişti. 1971 yılında Bell Labs şirketi UNIX işletim sistemini, yeni metin işleme sistemlerinde kullanılacak standart olarak kabul etti. 1972 Haziran ayında gelindiğinde, artık dünyada 10 kadar bilgisayar UNIX işletim sistemi ile çalışmaktaydı. Bu arada, Dennis Ritchie ve Brian Kernighan, **C programlama dili** üzerindeki çalışmalarını büyük ölçüde tamamlamışlardı. 1973 yılında, UNIX işletim sistemi, C programlama diliyle baştan yazıldı. Böylece bilgisayar tarihinin '**yüksek seviyeli bir dil ile yazılmış olan ve donanımdan bağımsız**' ilk işletim sistemi ortaya çıkmış oldu.

1974 yılından başlayarak, AT&T şirketi, bu yeni işletim sisteminin kaynak programlarını, başta Columbia Üniversitesi olmak üzere bir çok üniversite ve yüksek okula ÜCRETSİZ olarak dağıttı. UNIX işletim sisteminin önlenemez yükselişi başlamıştı (aslında önlemek isteyen olduğunu da sanmıyorum).

1975 yılına gelindiğinde, AT&T, UNIX Sürüm 6'yı kullanmaktaydı ve artık UNIX kullanmak isteyenler, küçük de olsa bir lisans ücreti ödemek zorundaydılar. UNIX, standart bir C kütüphanesi ile birlikte dağıtılmaya başlandı. Böylece; C dili, UNIX işletim sistemi için yazılım geliştirmek isteyenlerin öğrenmesi gereken bir dil olarak yaygınlaştı.

1977 yılında, Berkeley Üniversitesi, UNIX üzerindeki birikimlerini ilgililenenlere **1BSD : 1st Berkeley Software Distribution** adlı bir ürün olarak dağıtmaya başladı.

1978 yılında Bill Joy ve Özalp Babaoğlu (University of California-Berkeley'de yüksek lisans öğrencisi) UNIX işletim sistemine sanal bellek özelliğini *eklediler* (*virtual memory*). Artık UNIX tam bir işletim sistemi olmuştu. (Ref: *Unix Administration Guide for System V, Rebecca Thomas, ISBN 0-13-942889-5*).

1979 yılında, AT&T yedinci sürümü piyasaya çıkardı. UNIX'in yaratıcılarından Ken Thompson'un Berkeley Üniversitesi'nde ders vermeye başlamasıyla AT&T ve Berkeley ekipleri UNIX'i hızla geliştirmeye başladılar. Sonunda, ABD Savunma Bakanlığı'na bağlı DARPA (*Defence Advanced Research Projects Agency : İleri Savunma Araştırma Projeleri*) bölümü, UNIX için bir bütçe ayırmaya karar verdi.

1979'da UNIX artık iyice yaygınlaşmıştı. Üniversite yıllarında UNIX öğrenen, kullanan ve beğenen öğrenciler UNIX'i sanayiye taşımaya ve donanım üreticileri, tasarım aşamalarında UNIX işletim sistemini de göz önünde bulundurma zorunluluğunu hissetmeye başladılar.

1980 yılı sonunda, büyük bilgisayar üreticilerinin hepsi, hiç değilse bazı modellerinde, UNIX kullanmaya başladılar.

Günümüzde (1995) Hewlett-Packard, DEC (Digital Equipment Corporation), IBM, Unisys, Cray Research, SONY, Motorola, NCR, SUN Microsystems gibi devler, UNIX İşletim Sistemi'ni standart olarak desteklemektedir. Kişisel bilgisayarın devi Microsoft'un ve Santa Cruz Operations'un (SCO) UNIX'i PC dünyasına taşımasıyla da yayılım tamamlanmış oldu. Bugün, UNIX kullanılan bilgisayar sayısı tam olarak bilinmemekle birlikte, bu sayının milyonlarla ifade edileceği kesindir.

UNIX Geleneği

Çok geniş bir araştırmacı kitlesi tarafından geliştirilmesine rağmen, UNIX, ilk tasarımı olduğu günlerdeki özelliklerinden pek uzaklaşmamıştır. Bunun en önemli nedeni, bu araştırmacıların yazılı olmayan geleneklere bağlı kalmış olmalarıdır. Belki de UNIX, başarısını bu gelenekselleşmeye borçludur. (Japon'ların ekonomik mucizesinin de geleneklere bağlılık olduğu söylenmez mi?) Örneğin, dizinlerdeki dosyaların detaylı listesini veren **ls** komutunun 100 Megabyte'dan büyük dosyalarda ortaya çıkan hatası hala düzeltilmemektedir.

BSD

SVR4

Her ne kadar çok tutucu bir tablo çizmiş olsamda, 1990'lı yıllarda iki ayrı UNIX ekolü olduğundan söz etmek gerekmektedir: Berkeley Üniversitesinin yürüttüğü **BSD** ekolü ve AT&T şirketinin yürüttüğü AT&T UNIX (**SVR4** : System 5 Release 4) ekolü. Bu iki tip UNIX, kullanıcıları açısından pek önemli farklılıklar göstermese de, sistem yöneticileri açısından çok farklıdır. 1992 yılından başlayarak AT&T UNIX'i geliştiren ekipler, BSD UNIX'in üstün özelliklerini AT&T UNIX ile birleştirerek SVR4 UNIX'i ortaya çıkardılar ve BSD ekolüne göre önemli bir üstünlük kazandılar.

UNIX'i UNIX Yapan Özellikler

Belki bazı noktalar tekrar edilmiş olacak ama, UNIX'i UNIX yapan özellikleri bir kez daha sıralamak istiyorum. Kitabın okunması sırasında ve daha önemlisi UNIX İşletim Sistemi'ni kullanırken yararlı olacağı inancındayım.

- ✓ **UNIX 'çok kullanıcı' bir işletim sistemidir.** Kullanıldığı bilgisayarın bir anda birden fazla kişi tarafından kullanılmasını; daha doğrusu paylaşılmasını sağlayabilmektedir.
- ✓ **UNIX 'çok iş düzeni'ni sağlayan bir işletim sistemidir.** Kullanıcıların, herbirinin, aynı anda birden fazla iş yapmalarına olanak sağlar.
- ✓ **UNIX, donanımdan bağımsızdır.** Hangi bilgisayar üzerinde kullanılırsa kullanılsın, kullanıcılarına görüldüğü şekli aynıdır. Öğrendikleriniz kalıcıdır.

- ✓ **UNIX iyi tasarımılanmıştır.** Teknolojideki gelişmelere kolaylıkla uyum sağladığı ve sağlayacağı kanıtlanmıştır.
- ✓ **UNIX, bir işletim sistemi standardı olarak kabul edilmiştir.** Bu sayede farklı marka ve model bilgisayarlar birbirleriyle uyumlu kılınabilmektedir. Son günlerde sıkça sözü edilen 'Bilgi Süper Otoyolu' (*Information Super Highway : Internet*) bu sayede oluşabilmiştir.

UNIX'le Tanışma

UNIX işletim sistemi ile çalışan bir bilgisayarı kullanabilmek için sahip olmanız gereken üç şey vardır :

- a) UNIX altında çalışan bir bilgisayara bağlı bir **TERMINAL'e** (ekran+klavye) erişim yetkisi,
- b) UNIX altında çalışan bu bilgisayara erişim hakkınızın anahtarı olan '**kullanıcı hesabınız**' (*user account*),
- c) Eğer yeni başlıyorsanız; bol miktarda sabır.

Bu üç özelliğe sahip olduğunuzu varsayarak devam edelim.

Terminalinizi açınız (eğer terminal olarak kullandığınız ekran ve klavye, bilgisayarın ana ekran ve zaten açık olması gerekir.)

Bir kaç saniye içinde ekranda

`login :`

mesajını görmeniz gerekir. (Bazı terminallerde bu mesajı görebilmek için bir kaç kez ENTER (ya da RETURN) tuşuna basmanız gerekebilir).

Bu mesaj, bilgisayarın, daha doğrusu UNIX'in, kendinizi tanıtmanızı istediğini belirtmektedir. Her UNIX kullanıcısının bir adı olmalıdır. Bu ad, kullanıcılara **sistem yöneticisi** görevini üstlenmiş olan bilgisayar uzmanları tarafından verilir. Bu mesaja yanıt olarak klavyeden kullanıcı adınızı girmeniz ve ENTER tuşuna basmanız gerekir. Kendi adınızı veya rastgele bir ad girmenizin bir yararı olmayacaktır. UNIX, sadece daha önceden kendisine tanıtılmış olan kullanıcı isimlerini kabul edecektir. Eğer bir kullanıcı adınız yoksa daha fazla vakit kaybetmeden sistem yöneticisini bulup, size bir kullanıcı adı vermesini isteyiniz.

Neyse, geçerli bir kullanıcı adınız olduğunu varsayarak devam edelim...

```
login :ayfer
```

ENTER tuşuna basmanızla birlikte

```
Password :
```

mesajıyla şifrenizi girmeniz istenecektir. Kullanıcı olarak bilgisayara erişiminiz bir şifre ile korunmamışsa, yani sizin için henüz bir şifre girilmemişse, bu mesajı görmezsiniz. Şifreniz yoksa ve bunun özel bir nedeni yoksa, ilk fırsatta kendinize bir şifre seçip, bunu UNIX'e bildirmenizi öneririm. Bu işlem için kullanmanız gereken komut '**passwd**' komutudur. (**passwd** komutunu bir kaç sayfa sonra anlatacağım).

Eğer şifreniz varsa, siz klavyeden bu şifreyi girerken bastığınız tuşlar ekranda görünmeyecektir. (Siz farkında olmadan arkanızdan sizi gözleyenler varsa şifrenizi görmesinler diye...)



UNIX işletim sisteminde **büyük harf - küçük harf** farkı ÇOK önemlidir. **Ayfer**, **AYFER** ve **ayfer** farklı kullanıcı adlarıdır. Aynı fark, şifrelerde de söz konusudur. UNIX geleneği hep küçük harf kullanmanızı (şifreniz hariç) gerektirir.

Doğru şifreyi girdiğinizde (eğer şifre varsa tabii) ekranınızdaki görüntü

```
login : ayfer
```

```
Password :
```

```
ABC Bilgisayar sistemine hos geldiniz.  
Sistem, 12/1/1995 gunu saat 17:00 da bakim icin  
kapatilacaktır.
```

```
$ _
```

veya

```
login : ayfer
```

```
Password :
```

```
ABC Bilgisayar sistemine hos geldiniz.  
Sistem, 12/1/1995 gunu saat 17:00 da bakim icin  
kapatilacaktır.
```

```
% _
```

veya

```
login : ayfer
```

```
Password :
```

```
ABC Bilgisayar sistemine hos geldiniz.  
Sistem, 12/1/1995 gunu saat 17:00 da bakim icin  
kapatilacaktır.
```

```
abc:/home/ayfer $ _
```

gibi olacaktır.

Bu ekranlardaki

```
ABC Bilgisayar sistemine hos geldiniz.  
Sistem, 12/1/1995 gunu saat 17:00 da bakim icin  
kapatilacaktır.
```

satırları, sistem yöneticisinin kullanıcılara bir mesajıdır (**günün mesajı** : *message of the day*). Sistemdeki yenilikler, kullanıcılara haberler ve duyurular genellikle bu satırlarda yer alır; o nedenle bu mesajları okuma alışkanlığını edinmenizi öneririm.

En son satırlarda yer alabilecek olan

```
$  
%  
abc:/home/ayfer %
```

satırlarıysa, UNIX'in sizden komut almaya hazır olduğunu belirten '**hazır işareti**'dir (*prompt*).

Bu hazır işaretlerinde, UNIX'in sizden komut almaya hazır olduğundan başka çok önemli bir bilgi daha vardır. Bu bilgi, % veya \$ karakterleridir. Şimdi sıkı durun, hazır işaretinizde % görüyorsanız kullanacağınız **kabuk** (shell) **Bourne Shell** 'dir, \$ görüyorsanız **C Shell** 'dir. (Sabırlı olmanız gerektiği konusunda uyarılmıştım....)



Kabuk (Shell) kavramı, UNIX kullanıcılarının iyi anlaması gereken bir kavramdır. Bu noktada MS-DOS işletim sistemi ile bir benzerlik kurmak istiyorum.

MS-DOS'daki C:\> benzeri bir hazır işaretinin karşısına yazacağınız komutu irdeleyen, yapılmasını istediğiniz işe ait programı belleğe yükleyen, gerekli parametreleri bu programa aktaran, işletim sisteminin bir parçası COMMAND.COM isimli programdır.

UNIX işletim sisteminde de, aynı şekilde, kullanıcının klavyeden yazacağı komutu irdeleyen, kullanıcının ne yapılmasını istediğini çözümleyen ve bu işin yapılabilmesi için gerekli programları belleğe yükleyen, komut parametrelerinin bu programlara aktarılmasını sağlayan bir program vardır. Bu programların genel adı **kabuk (shell)** sözcüğüdür. MS-DOS işletim sisteminden farklı olarak, UNIX'de, kullanıcının tercihiyle bağlı olarak kullanabileceği birden fazla **komut yorumlayıcısı** (kabuk = shell) vardır. Bu kabuklara örnek olarak

sh	Bourne Shell	S.R. Bourne	AT&T
csh	C Shell	Bill Joy	Berkeley
ksh	Korn Shell	David Korn	AT&T
bash	Bourne Again Shell		
tcsh	Geliştirilmiş csh		

gösterilebilir.

'Yeni kullanıcılar için şimdilik bu kadar bilgi yeter' deyip devam edelim.

Eğer kullanmakta olduğunuz kabuğun (sistem yöneticisinin sizin için uygun gördüğü kabuk) hangisi olduğunu kesin olarak öğrenmek istiyorsanız

```
% cat /etc/passwd | grep ayfer
```

ayfer sözcüğü yerine kendi kullanıcı adınızı yazmayı unutmayınız!

komutunu yazınız. Göreceğiniz

```
ayfer:As@cX*as:1234:200:Ugur Ayfer:/home/ayfer:/bin/csh
```

benzeri bir satırın en sonuna bakınız. Burada göreceğiniz kabuk programının adı, aradığınız yanıttır.

Eğer kullandığınız UNIX bilgisayarı bir SUN iş istasyonuysa ve verdiğiniz bu komuta yukarıdaki örneğe uygun bir yanıt alamazsanız, bir de

```
% ypcat passwd | grep ayfer
```

komutunu deneyiniz. Gerek duyarsanız sistem yöneticisinden yardım isteyebilirsiniz.

Kabuk Programının Adı	Kabuk Tipi
/bin/csh	C Shell
/bin/sh	Bourne Shell
/bin/ksh	Korn Shell
/bin/bash	Bourne Again Shell
/bin/tcsh	T C Shell

Kullandığınız kabuk programı hangisi olursa olsun, temel UNIX kuralları değişmeksizin geçerli olacaktır. Yeni başlayanların, eğer mümkünse, **csh** kabuk programını kullanmalarını öneririm. Bu kitapta göreceğiniz örneklerin büyük çoğunluğu **csh** için verilecektir.

Hangisi olursa olsun; UNIX kabuk programları, MS-DOS işletim sisteminin komut yorumlayıcısı olan COMMAND.COM'la karşılaştırılamayacak kadar gelişmiş ve yeteneklidirler. (Tabii bir o kadar da karmaşık!).

UNIX işletim sistemi ile yapmakta olduğunuz işi tamamladığınızda ve terminalin başından ayrılacağınız zaman

```
% logout
```

komutunu vermeyi unutmamalısınız.

Bu komut, UNIX ile bağlantınızı kesecektir; ve terminal bir sonraki kullanıcıyı bekleme konumuna geçecektir. (login :)



UNIX işletim sisteminde **BİR** bilgisayarı paylaşan kullanıcı**LAR** söz konusudur. Bu durumda kullanıcıların kayıtlı bilgilerini birbirlerine karşı korumak gereklidir. Bir sabah işe geldiğinizde tüm kayıtlı bilgilerinizin kaybolduğunu düşünebiliyor musunuz?

Kullanıcıların kayıtlı bilgilerinin yanısıra, işletim sistemi, kendisini de hatalı komutlara ve kötü niyetli kullanıcılara karşı korumak zorundadır. Bu koruma mekanizmasının temelinde **kullanıcı adı** ve **şifresi** yer almaktadır. Her UNIX kullanıcısı şifresini iyi korumak zorundadır. Şifrenizi belki iyi koruyor olabilirsiniz; ancak **logout** komutunu vermeden terminalinizin başından kalkarsanız, arkanızdan terminalin önüne oturan birisi sizin kişiliğinizle UNIX'e vereceği komutlarla bilerek ya da bilmeyerek kayıtlı dosyalarınıza zarar verebilir.

Bir UNIX bilgisayarıyla işiniz bittiğinde **logout** komutunu kullanarak bilgisayarla bağlantınızı kesmelisiniz. Ancak, **logout** etmeniz, bilgisayarı da kapatabileceğiniz anlamına gelmez.



Lütfen; ama lütfen, UNIX işletim sistemi ile çalışan bir bilgisayar işiniz bittiğinde **küüt diye kapatmayınız**. Bir UNIX bilgisayarının sağlıklı bir şekilde kapatılabilmesi için bir dizi törensel işlem yapılması gerekir. Eğer bu işlemleri yapmadan kapatırsanız, bilgisayarı bir daha açamayabilirsiniz; hatta kayıtlı tüm veri ve programları kaybedebilirsiniz.



Bir UNIX bilgisayarın kapatılması için gereken törensel işlemler, bu kitabın 'Sistem Yöneticisine' başlıklı bölümünde anlatılacaktır.

Isınma Hareketleri

Kullanıcı ile UNIX İşletim Sistemi arasındaki tüm haberleşme **kabuk** (shell) programı aracılığı ile yürütülmektedir. Klavyeden yazacağınız her komut, kullanmakta olduğunuz kabuk programı tarafından yorumlanmaya çalışılacaktır. Eğer kullanmakta olduğunuz kabuk için anlamı olmayan komutlar yazacak olursanız, beklemediğiniz hata mesajları ile karşılaşabilirsiniz. Bu bölümdeki örnekler **cs** kabuğu için hazırlanmıştır. Eğer kullandığınız kabuk Bourne Shell (**sh**) ise (hazır işaretinizin sonunda **\$** karakteri varsa), klavyeden

```
$ /bin/csh
```

komutunu vererek **C Shell kabuğuna** geçmeyi deneyiniz. Eğer bir hata mesajı almazsanız ve hazır işaretinizin sonunda **%** karakteri olan bir diziye dönüşürse başardınız demektir.

```
login : ayfer
Password :
```

Günün mesajları

```
$ /bin/csh
```

```
abc:/home/ayfer %
```

c-shell'e geçiş başarılı...

Eski kabuğunuza dönmek istediğinizdeyse, **Ctrl-D** ye basmalı veya **exit** komutunu vermelisiniz.

Ben Kimim?

İlk bakışta çok anlamlı değilmiş gibi görünen bu soru UNIX dünyasında zaman zaman sorulması gereken bir sorudur. Eğer kullandığınız UNIX bilgisayar büyük bir bilgisayar ağının bir parçasıysa ve siz bu ağ üzerinden bir çok bilgisayara ulaşabiliyorsanız ve bu değişik bilgisayarlardaki kullanıcı isimleriniz (*user-id*) farklıysa; uzun çalışma seansları sırasında, o anda geçerli olan kullanıcı kimliğinizi şaşırtabilirsiniz.

Hemen

```
% whoami
```

```
% who am i
```

BSD UNIX'lerde

SV5R4 UNIX'lerde

komutunu verip, UNIX'in sizi o anda hangi kimlikle tanıdığını öğrenebilirsiniz. Özellikle sistem yöneticileri, zaman zaman başka kullanıcıların kimliğine bürünme gereksinimi duyarlar (bu işi **su** - *switch user* komutuyla yaparlar). Bir o - bir bu kullanıcı kimliğine büründüklerinde de bazen şaşırmalar olur.

Böyle bir durumda hemen **whoami** komutunu vererek o andaki kimliklerini öğrenebilirler.

Başka Kimler Var?

UNIX işletim sistemi altında çalışan bilgisayarların, bir anda birden fazla kullanıcı tarafından kullanılabilceğini belirtmiştim. İsterseniz, şu anda bilgisayarınızdan başka kullanan kimse var mı, onu öğrenelim. Bunun için vermeniz gereken komut:

```
% who
```

```
abc:/home/ayfer % who
ayfer          tty01          Jan 12    15:12
hakan          tty03          Jan 12    10:09
root           console        Jan 11    23:40
abc:/home/ayfer %
```

Yukarıdaki örneğe göre, şu anda bilgisayarınız paylaştan 3 kişi olduğunuz anlaşılıyor. Diğer ortaklarınızın isimleri **hakan** ve **root**. Hakan 3 numaralı terminalin, **root** ise ana terminalin (*konsol*) başında oturuyor. **hakan** 12 Ocak günü saat 10:09 da **login** etmiş; **root** ise bir gün önce gece yarısına doğru çalışmaya başlamış. Eğer, **root** gerçekten dün gecedan beri çalışıyorsa mesele yok; ama eğer gece eve gitmiş ve giderken **logout** komutunu vermemişse önemli bir güvenlik hatası yapmış demektir.



UNIX kullanıcılarının isimleri genellikle kullanıcıların gerçek kimliklerini yansıtacak şekilde seçilir. Sistem yöneticisi; bir kullanıcı tanıtımı yaparken, kullanıcı hesap ismi yanısıra, bu kullanıcının bilgisayardaki kaynaklara erişim yetkilerini de tanımlar.

Ancak, UNIX işletim sisteminde adı hiç bir zaman değişmeyen **ÖZEL** bir kullanıcı vardır. Bu kullanıcının adı, **root** sözcüğüdür. Adı **root** olan kullanıcı HER ŞEYİ YAPMAYA YETKİLİDİR. İsteddiği dosyayı siler, yaratır, yerini ve içeriğini değiştirir vs. vs. Bu kullanıcıya "süper kullanıcı" (*super user*) adı da verilir.

Eğer bir UNIX bilgisayarına **root** kullanıcı olarak erişme hakkınız varsa (yani **root** şifresini biliyorsanız), gerekmedikçe bu isimle **login** etmeyiniz. Yapacağınız hatalar sisteminizi çalışmaz hale getirebilir. UNIX işletim sistemi, **root** isimli kullanıcının yaptığı işi çok iyi bildiğini varsayıp, hiç bir uyarıda bulunmaksızın verilen komutları yerine getirir. (**her şeyi sil** komutu dahil!)

Arayan Soran Var mı?

UNIX işletim sisteminde, kullanıcılar arasında elektronik posta haberleşmesinin yapılmasını sağlayan **e-mail** (*electronic mail*) yazılımı standarttır. Kullanıcılar birbirlerine göndermek istedikleri mesajları (**elektronik posta** veya kısaca **mektup**)

```
% mail
```

komutunun yardımıyla yazarlar, gönderirler ve kendilerine gelen mektupları gene bu komutla okurlar.

mail komutunu parametresiz olarak kullandığınızda :

```
abc:/home/ayfer % mail
You have no mail.          (Mektubunuz yok.)
veya
No messages
```

yanıtları yanısıra, size gönderilmiş mektup(lar) varsa:

```
Mail ver 4 Thu Jan 31 12:54 EST 1995 Type ? for help
"/usr/mail/ayfer":3 messages 2 new
U 1 cil@bilkent    Fri Jan 12 14:32 23/567 Yeni uygu.
N 2 tayfun@abc     Fri Jan 12 15:34 34/762 Onemli
N 3 kerem@abc      Fri Jan 23 09:12 45/947 SUNOS4.1
&
```

gibi size gelen mektupların bir listesini görebilirsiniz. Bu mektup listesinde, size mektubu gönderen kullanıcının adı, mektubun konusuna ilişkin kısa bir not ve mektup sıra numarası yer alır. Tamamını okumak istediğiniz mektubun numarasını girdiğinizde elektronik mektubunuzun tamamını okuyabilirsiniz. Okumak istediğiniz mektuplar bitince, **x** tuşuna basarak **mail** programından çıkabilirsiniz. Bu komutun kullanımı ile ilgili detayları daha sonraki bölümlerde anlatacağım.

Siz sisteme bağlı değilken, adresinize (kullanıcı adınıza) bir mektup gelirse, ilk **login** ettiğinizde, UNIX sizi

```
You have new mail
```

diye uyaracaktır.

Bu uyarıyı gördüğünüzde **mail** komutunu kullanarak gelen mektuplara bakabilirsiniz; bu mektupların sizi ilgilendirmediklerini ya da başka birisini de ilgilendirdiklerini düşünüyorsanız, mektubu başka bir adrese yönlendirebilirsiniz, mektubu saklayabilirsiniz ya da çöpe atabilirsiniz. (Eğer Internet bağlantınız varsa, her gün bir sürü çöpe atılacak mektup alacağınızdan emin olabilirsiniz).

Şifrenizi Değiştirmek İstediyinizde...

UNIX altında çalışan bir bilgisayara sizin adınızı (yani **kullanıcı adınızı** demek istiyorum) kullanarak ulaşabilen herkes, size gelen elektronik mektupları da okuyabilir. Başkalarının size ait dosyaları ve elektronik mektupları okumasını istemiyorsanız, UNIX'in şifre mekanizmasından yararlanmanız gerekecektir. Bilgisayara erişim şifrenizi (password) değiştirmek istediğinizde

```
% passwd
```

komutunu kullanmalısınız. Eğer şifreli bir kullanıcı adı ile çalışıyorsanız, yeni şifre verebilmek için o anda geçerli olan şifreyi bilmeniz gerekecektir.

```
abc:/home/ayfer % passwd
```

```
Changing old password for ayfer
```

```
Old password :
```

eski şifreyi veriniz

```
New password :
```

yeni şifreyi giriniz

```
Retype new password :
```

yeni şifreyi bir kez daha giriniz.

Şifreyi iki kez girmenizin istenmesi oldukça mantıklı değil mi? Klavyeden yazarken ekranda göremeyeceğiniz bir şifreyi hatalı yazarsanız, bir daha bu sisteme **login** etmeniz olanaksız hale gelecektir.

Şifrenizi seçerken bazı noktalara dikkat etmelisiniz!

Seçtiğiniz şifre, sizin tarafınızdan kolayca hatırlanacak; ancak başkaları tarafından kolayca tahmin edilemeyecek bir karakter dizisi olmalıdır. Eşinizin veya çocuğunuzun adı, soyadınız, arabanızın plakası, doğum tarihiniz şifre olarak kullanılması sakıncalı olan dizilerdir. Şifre olarak çok karmaşık diziler seçip, sonra da bu şifreyi unutmamak için bir kenara yazmak da oldukça sık yapılan güvenlik hatalarındandır.

Şifrenizi seçerken, mümkün olduğunca harf ve sayıları karıştırınız. Daha iyisi hem büyük, hem küçük harfleri bir arada kullanınız.

Şifreniz ne çok uzun, ne de çok kısa olsun. 6 - 8 karakterlik diziler hem kolay hatırlanır, hem de klavyeden yazılırken pek hata yapılmaz.

ayfer

Çok kötü bir şifre, hemen tahmin edilir.

AyfeR-1995

Hiç fena değil.

123456

Çok ciddiyetsiz, üstelik klavyeden yazarken kolayca izlenir.

aBcDeF

Fena değil ama çok kişi buna benzer şifre kullandığı için kötü niyetli kişilerce ilk denenene kalıplardandır.

x1e34TQ?w/&1+

Harika bir şifre, ama siz hatırlayabilecek misiniz?

Sisteme **login** ettiğinizde, UNIX genellikle bir önceki **login** seansınızın hangi tarihte gerçekleştiğini size hatırlatır. Bu hatırlatmaya her **login** edişinizde dikkatlice bakmanızı öneririm. Bu mesaj sayesinde, sizin adınızı kullanarak sisteme ulaşan birileri varsa, durumu farkedebilirsiniz. Böyle bir durumdan şüphelendiğiniz anda şifrenizi değiştiriniz. Daha da iyisi, şifrenizi en az ayda bir kez değiştiriniz. Nitekim, bazı sistem yöneticileri, kullanıcıları, şifrelerini belirli sıklıklarda değiştirmeye otomatik olarak zorlarlar. (*Password aging* kavramı).



Şifreniz, sistem yöneticisinin, diğer adıyla **root** kullanıcısının, dosyalarınıza bakmasına engel olamaz. **root** herşeyi olduğu gibi, mektuplarınızı ve diğer dosyalarınızı da okumaya yetkilidir.

İmdaaaaat !..

UNIX işletim sisteminde kullanılabilecek yüzlerce komut vardır. Seyrek kullanılan komutların genel yapılarını ve parametrelerinin hepsini hatırlamak pek kolay olmadığı için; UNIX işletim sistemi, tüm komutlarının kullanım kılavuzlarını standart olarak size sunmaktadır. Bir komutun nasıl kullanılacağını öğrenmek ya da hatırlamak istediğinizde

```
% man komut-adi
```

(manual)

komutunu vermeniz, *komut-adi* adlı komutun kullanım kılavuzu sayfalarının ekranınızda görüntülenmesini sağlayacaktır. Örneğin, **passwd** komutunun nasıl kullanılacağını merak ederseniz

```
% man passwd
```

mail komutunun nasıl kullanılacağını hatırlamak içinse

```
% man mail
```

komutlarını kullanabilirsiniz.

İşiniz bittiğinde...

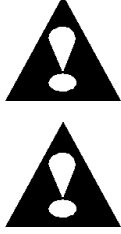
Bilgisayarla işiniz bittiğinde, terminalinizin başından ayrılmadan önce

```
% logout
```

komutunu veriniz.

Sistem yöneticiniz (ya da siz) aksini belirtmediyseniz, (**.cshrc Dosyasıyla** ilgili bölümde **ignoreeof** parametresine bakınız) **Ctrl-D** tuşuna basarak da sistemle bağlantınızı kesebilirsiniz. Ctrl-D aslında bağlantı kesme komutu değil, o anda aktif olan komut yorumlayıcınızı (kabuk) öldürme komutudur. Eğer öldürdüğünüz kabuk, yegane kabuğunuzsa sistemle bağlantınız kesilir; yok

ikinci ya da üçüncü kabuğunuzsa, bir önceki kabuğunuza dönersiniz. Bu kavram biraz karışık geldiyse aldırmayın, zamanla açıklığa kavuşacaktır.



Bir UNIX bilgisayarın başında işiniz bittiğinde **logout** komutunu kullanarak bilgisayarla bağlantınızı kesmelisiniz. Ancak, **logout** etmeniz, bilgisayarı da kapatabileceğiniz anlamına gelmez.

Lütfen; ama lütfen, UNIX işletim sistemi ile çalışan bir bilgisayarı işiniz bittiğinde **küüt diye kapatmayınız**. Bir UNIX bilgisayarının sağlıklı bir şekilde kapatılabilmesi için bir dizi törensel işlem yapılması gerekir. Eğer bu işlemleri yapmadan kapatırsanız, bilgisayarı bir daha açamayabilirsiniz; hatta kayıtlı tüm veri ve programları kaybedebilirsiniz.

Bir UNIX bilgisayarın kapatılması için gereken törensel işlemler, bu kitabın 'Sistem Yöneticisine' başlıklı bölümünde anlatılacaktır.

UNIX Dosya Yapısı (UNIX File System)

Tüm bilgisayar işletim sistemlerinin olduğu gibi, UNIX'in de en temel amacı kullanıcıların verilerini ve programlarını bilgisayar ortamında düzenli bir şekilde saklamalarına yardımcı olmaktır. UNIX işletim sisteminde tüm veriler, programlar ve herbiri aslında bir program olan komutlar, **dosya**'larda (*file*); dosyalarsa dizinlerde (**dizin** : *directory*) gruplanmış olarak saklanır.

UNIX dosya yapısını anlatırken okuyucunun MS-DOS işletim sistemine aşina olduğunu varsayacağım ve bu nedenle sık sık MS-DOS'la karşılaştırmalar yapacağım. Bu arada da sık sık UNIX'in MS-DOS'a karşı ezici üstünlüğünü vurgulamış olacağım. Bu nedenle MS-DOS hayranlarından şimdiden özür dilerim.

UNIX işletim sisteminde dosya isimlerine ilişkin kurallar oldukça esnektir.

En başta, MS-DOS'daki gibi 8 karakterden oluşan isim ve 3 karakterden oluşan uzantı (*extension*) kavramı yoktur. Dosya isimleri, UNIX uyarlamasına bağlı olarak değişmekle birlikte 255 karaktere kadar uzunlukta olabilir (bu uzunlukta dosya isimlerini kim hatırlayıp klavyeden yazacaksa...).

Nokta (.) karakterinin özel bir anlamı yoktur. Dosya adı içinde istediğiniz kadar nokta kullanabilirsiniz. Ancak, nokta ile başlayan dosya isimleri bir anlamda özeldir; adı nokta ile başlayan dosyalar yarı gizli dosyalardır. Özellikle belirtmedikçe, dosya isimleri listelerinde bu tür dosyaları göremezsiniz.

Dosya isimlerinde büyük harf-küçük harf ayırımı VARDIR. **ayfer.mektup**, **Ayfer.Mektup** ve **AYFER.MEKTUP** tamamen farklı dosya isimleridir.

Bir kaç örnek vermek gerekirse :

ayfer.mektuplar	<i>Geçerli bir dosya adı,</i>
a1	<i>Geçerli bir dosya adı,</i>
1a	<i>Geçerli bir dosya adı,</i>
1-a	<i>Geçerli bir dosya adı,</i>
muhasebe_1995_mizan	<i>Geçerli bir dosya adı,</i>
Sinanin.Muhasebe.Programi	<i>Geçerli bir dosya adı,</i>
.login.eski	<i>Geçerli bir dosya adı,</i>
lotus.exe	<i>Geçerli bir dosya adı,</i>
prog1.com	<i>Geçerli bir dosya adı,</i>

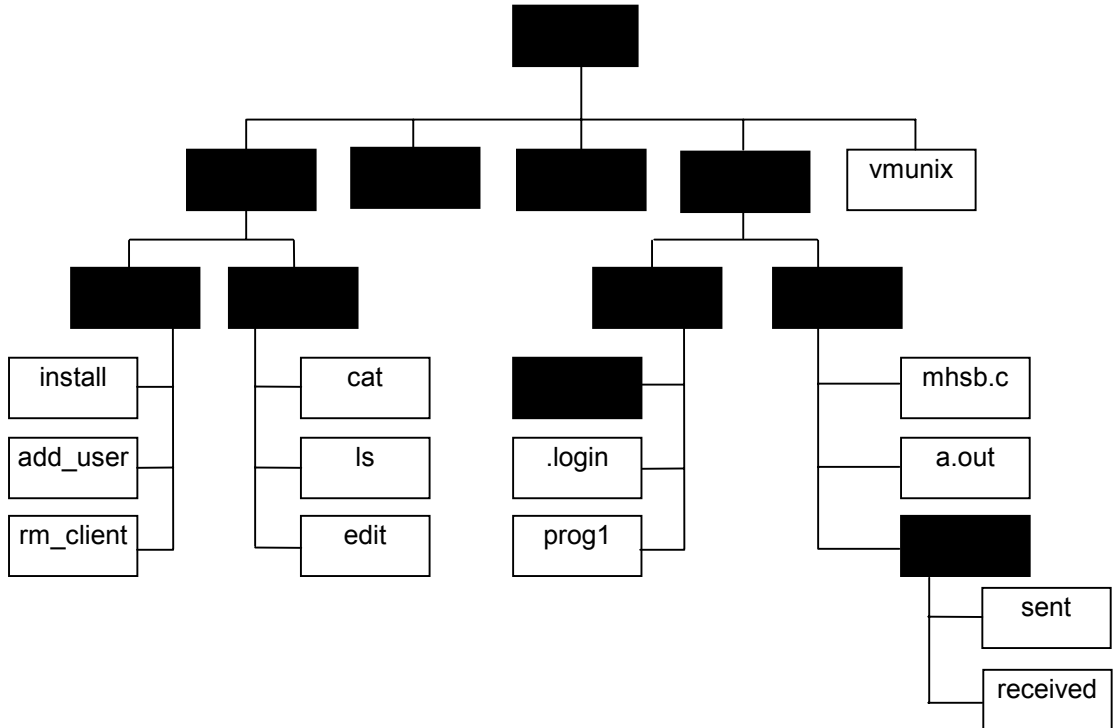


Dosya isimleriyle, dosyaların program olup olmaması arasında bir ilişki yoktur.

Örneğin, **lotus.exe** isimli bir dosyanın, bir program dosyası olması gerekmez. Bir dosyanın program dosyası olup olmadığını isminden anlayamazsınız. Program dosyalarının diğer dosyalardan nasıl ayırdedileceğini bir kaç sayfa sonra anlatacağım.

UNIX, MS-DOS'dan tanıdığınız hiyerarşik dosya-dizin yapısını kullanmaktadır. En üst düzeyde bir **root** dizini ve bunun altında istendiği gibi yerleştirilmiş olan dosya ve alt-dizinler ile gene bu alt-dizinler altında yerleştirilmiş dosyalar ve gene alt-dizinler...

Şematik olarak göstermek gerekirse :



Dikkat ederseniz, MS-DOS dosya yapısından farklı olarak '**root**' dizininin adı \ (back-slash) değil, normal / (slash) karakteridir. Aynı şekilde, bir dosyanın dizinler arasındaki yerini tanımlarken, MS-DOS'daki \ karakteri yerine / karakteri kullanılır. Bunu örneklerle göstermek gerekirse; yukarıdaki dosya-dizin yapısında yer alan bazı dosyaların tam isimleri şöyle yazılır :

UNIX	MS-DOS
/usr/bin/cat	C:\USR\BIN\CAT
/home	\home
/home/sina/Mail/sent	\HOME\SINA\MAIL\SENT
/vmunix	C:\VMUNIX

Her horoz kendi çöplüğünde...

UNIX işletim sisteminde, her kullanıcının kendisine ait bir '**kullanıcı dizini**' (UNIX terminolojisinde : **home directory**) vardır. Bu dizin, kullanıcının sisteme tanıtımı sırasında, sistem yöneticisi tarafından yaratılır. Her kullanıcının kendi '**kullanıcı dizini**'nde sınırsız yetkileri vardır. Bu dizin altında istediği gibi dosya ve alt dizinler yaratır, bunları siler, isimlerini ve içeriklerini değiştirir vs. vs.

Her kullanıcının kendi dizinindeki bu yetkileri, başka kullanıcıların dizinleri üzerinde yoktur. Bir başka deyişle, **ayfer** isimli kullanıcı, **sina** isimli kullanıcının dizinindeki dosyaları silemez, değiştiremez, **sina** izin vermedikçe okuyamaz; hatta varlığından bile haberdar olamaz.

Sisteme **login** eden her kullanıcı, çalışma dizini, kendisine ait kullanıcı dizini olacak şekilde çalışmaya başlar. Sistem yöneticileri, kullanıcı dizinlerini, genellikle **/home** dizini altına açtıkları dizinler olarak düzenlediklerinden (tipik bir UNIX geleneği) **ayfer** isimli kullanıcının **login** ettiğinde kendini **/home/ayfer** adlı dizinde bulması doğaldır.

```
login : ayfer
Password :
... Günün mesajları ...
abc:/home/ayfer %
```

Bu örnekteki '**abc**', kullandığınız UNIX bilgisayarının adıdır. Eğer bilgisayarınız bir bilgisayar ağına bağlıysa, bu ad çok önemli olacaktır.

root isimli kullanıcı, kime ait olursa olsun, tüm dosya ve dizinler üzerinde, bu dosya ve dizinler sanki kendisininmiş gibi tam yetkilidir. İsterse siler, isterse değiştirir.



Lütfen **root** isimli kullanıcıyla, izin yapısının en üst düzeyindeki **root** dizinini ('/') karıştırmayınız. Her iki kavram için de aynı sözcüğün kullanılmasının nedeni, her şeye yetkili olan **root** isimli super kullanıcının, kendisine ait olan 'kullanıcı dizini' nin tüm izin yapısını temsil eden / dizini olmasıdır.

Her ne kadar, UNIX sizi kendi kullanıcı dizininize yerleştirdiyse de, bu yerleşim mutlak değildir. İsterseniz **cd** komutu ile çalışma dizininizi (*default directory*) değiştirebilirsiniz.

```
abc:/home/ayfer % cd /usr/etc
abc:/usr/etc %
```

cd komutunu kullanarak çalışma dizininizi değiştirdiğinizde, **hazır işareti**nde (**prompt**) yeni çalışma dizinin adının yer alması bir tercihtir. Eğer hazır işaretinizde çalışma dizininizi göremiyorsanız sistem yöneticisine başvurunuz. Bu işi kendiniz halletmek istiyorsanız, kitabın ileri bölümlerinde bu işin nasıl yapılacağını anlatacağım.

Neredeyim?

Eğer kullandığınız sistem, hazır işareti içinde size bulunduğunuz çalışma dizinini göstermiyorsa

```
% pwd (print working directory)
```

komutunu kullanarak çalışma dizininizi öğrenebilirsiniz.

Örneğin,

```
login : ayfer
Password :
...
Günün mesajları
...
% pwd
/home/ayfer
%
```

Ne var ne yok?

Çok doğal olarak, bulunduğunuz dizinde yer alan dosyaların listesini görmek isteyeceksiniz.

Kullanacağınız komut en basit haliyle :

% ls	(list)
------	--------

Hemen bir örnek vereyim :

```
/home/ayfer % cd /
/ % ls
bin          export      lost+found   tmp
boot         home        mnt          usr
dev          kadb       sbin         var
etc          lib         sys          vmunix
/ %
```

Pek açıklayıcı olmadı galiba; değil mi? Hangisinin dosya, hangisinin dizin olduğu belli değil; oysa MS-DOS bu farkı **<DIR>** sembolü ile belirtirdi (!).

Daha açıklayıcı bir liste isterseniz **ls** komutunun yanına **-F** seçeneğini koyabilirsiniz (Dikkat! F büyük harf olmalı) :

```
/home/ayfer % cd /
/ % ls -F
bin/         export/      lost+found/ tmp/
boot         home/        mnt/         usr/
dev/         kadb*       sbin/         var/
etc/         lib/         sys/         vmunix*
/ %
```

Bu listede dizinler, isimlerinin sonuna yerleştirilen **"/** karakterleriyle; program veya komut dosyalarıysa **"*"** ile belirtilmiş olarak karşınıza çıkacaktır. Herhangi bir eki olmayan isimlerse, program dosyası veya dizin olmayan, diğer tip dosyalara aittir.

Bazı isimlerin sonunda **"@"** işareti göreceksiniz. Bu işaretin anlamını açıklamak için henüz biraz erken; ama şu kadarını söyleyebilirim : **"@"** işaretli dosya veya dizinler, aslında orada olmayan dosya ve dizinleri belirler. Nasıl ama? Esrareniz değil mi? Var ama aslında yok...

Bu liste her zaman alfabetik sırada ve dosya isimlerinin izin verdiği ölçüde birden fazla sütun halinde dökülecektir. Bu listeye önce ilk sütunu, sonra diğer sütunları göreceğiz şekilde bakmaya alışmalısınız.

Dosyalar ve izinler hakkında daha detaylı bilgi istiyorsanız aşağıdaki komutu denemelisiniz. **ls** komutunun bu formunu MUTLAKA deneyiniz ve bu form ile alacağınız listenin nasıl yorumlandığını lütfen ÇOK ÇOK İYİ ANLAYINIZ. UNIX mantığını iyi kavrayabilmeniz açısından buradan başlayarak anlatacaklarım oldukça önemlidir.

```
% ls -l
```

(long list)

```
/home/ayfer % cd /
/ % ls -l
total 3166

lrwxrwxrwx 1 root          7 Jan 12 12:09 bin -> /usr/bin
-r--r--r-- 1 root    110912 Jan 12 12:11 boot
drwxr-sr-x 2 bin        7680 Jan 12 12:23 dev
drwxr-sr-x 7 bin        1536 Jan 15 08:45 etc
drwxr-sr-x 4 root        512 Feb  1 11:56 export
drwxr-xr-x 5 root        512 Mar 23 09:03 home
-rwxr-xr-x 1 root   239783 Feb 09 13:34 kadb
lrwxrwxrwx 1 root          7 Mar 01 18:23 lib -> /usr/lib
drwxr-xr-x 2 root      8192 Jun 15 23:09 lost+found
drwxr-sr-x 2 bin        512 Mar 01 20:09 mnt
drwxr-sr-x 2 bin        512 Mar 09 08:59 sbin
lrwxrwxrwx 1 root        13 Jan 24 07:45 sys -> /usr/kvm/sys
drwxrwsrwt 2 bin        512 Feb 24 09:56 tmp
drwxr-xr-x 20 root       512 Nov 23 16:08 usr
drwxr-xr-x 11 root       512 Nov 23 16:11 var
-rwxr-xr-x 1 root   1101191 Jan 11 09:35 vmunix
/ %
```

Bu ayrıntılı liste, inanamayacağınız kadar çok bilgi içermektedir. Bu aşamada bütün detaylara girmeyeceğim; sadece satırlardan birini örnek olarak ele alıp, bir fikir verecek şekilde kısaca açıklayacağım.

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

-rwxr-xr-x : Bu satırın bir dosyayla ilgili olduğunu (en baştaki - işaretinden anlıyoruz); bu dosyanın sahibinin bu dosyada okuma (**r** : read), yazma (**w** : write) ve çalıştırma (**x** : execute) yetkilerinin olduğunu; diğer kullanıcıların, sadece okuma ve çalıştırma yetkilerinin bulunduğunu; dolayısıyla bu dosyanın bir **program dosyası** olduğunu;

root : Bu dosyanın sahibinin **root** isimli kullanıcı olduğunu;

239783 : Dosyanın uzunluğunun 239,783 byte olduğunu;

Feb 09 13:34 : Dosyanın en son 9 Şubat saat 13:34 de değişikliğe uğradığını;

kadb : Dosyanın adının **kadb** olduğunu göstermektedir.

Dizinler içinse, bu **ls** satırı biraz farklıdır :

```
drwxr-xr-x 20 root          512 Nov  23 16:08 usr
```

En baştaki **d** harfi, listenin bu satırının bir dizine ait olduğunu göstermektedir. Dosya uzunluğu yerinde yazılı olan sayıysa, dizinlerde pek anlamlı değildir; daha doğrusu anlamı konumuzun tamamen dışındadır.

Her satırdaki **rwxr-xr-x** benzeri kalıplarda gördüğümüz kodlar, kullanıcıların dosya (ya da dizin) üzerindeki erişim yetkilerini tanımlamaktadır. Programın ilerleyen saatlerinde bu erişim yetkileri konusu detaylı olarak ele alınacaktır. Bizden ayrılmayın.

% man ls

man ls komutunu verdiğinizde, aşağıda göreceğiniz uzun açıklamalar ekranınıza listelenecektir. Bu açıklamalar, kullandığınız UNIX işletim sistemine ait kullanım kılavuzunun **ls** komutu ile ilgili bölümlerinin aynısıdır. **man** komutunu verdiğinizde, listelenecek satırlar bir ekran sayfasından fazlaysa, birinci sayfanın listelenmesi tamamlanınca, ekranın sol alt tarafında

--- more ---

işareti göreceksiniz. Bu mesaj; listelenen açıklamaların devamı olduğunu; bu sayfayı okumayı tamamlayınca klavyeden bir komut vererek listenin devamını görmeyizin mümkün olduğunu belirtmektedir.

--- more --- un karşısına :

boşluk	tuşuna (Space Bar) basarsanız, bir sonraki sayfa,
RETURN	tuşuna basarsanız, bir sonraki satır,
b	(küçük b) tuşuna basarsanız bir ÖNCEKİ sayfa listelenir. Bu geri gitme özelliği her UNIX uyarmasında (örneğin SCO UNIX) çalışmaz.

Şimdi, **ls** komutunun detaylarını öğrenmek için **man ls** komutunu bir deneyiniz.

```
abc:/home/ayfer % man ls
Reformatting page. Wait... done
```

LS(1V)	USER COMMANDS	LS(1V)
NAME		
ls - list the contents of a directory		
SYNOPSIS		
ls [-aAcCdFfGiLLqrRstu] filename ...		
/usr/5bin/ls [-abcCdFfGiLLmnopqrRstux] filename ...		
AVAILABILITY		
The System V version of this command is available with the System V software installation option. Refer to Installing SunOS 4.1 for information on how to install optional software.		
DESCRIPTION		
For each filename which is a directory, ls lists the contents of the directory; for each filename which is a file, ls repeats its name and any other information requested. By default, the output is sorted alphabetically. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments are processed before directories and their contents.		
In order to determine output formats for the -C, -x, and -m options, /usr/5bin/ls uses an environment variable, COLUMNS, to determine the number of character positions available on one output line. If this variable is not set, the terminfo database is used to determine the number of columns, based on the environment variable TERM. If this information cannot be obtained, 80 columns are assumed.		
Permissions Field		
The mode printed under the -l option contains 10 characters interpreted as follows. If the first character is:		
d entry is a directory; b entry is a block-type special file; c entry is a character-type special file; l entry is a symbolic link; p entry is a FIFO (also known as "named pipe") special file; s entry is an AF_UNIX address family socket, or - entry is a plain file.		
The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next refers to permissions to others in the same user-group; and the last refers to all others. Within each set the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, "execute" permission is interpreted to mean permission to search the directory. The permissions are indicated as follows:		
r the file is readable; w the file is writable; x the file is executable; - the indicated permission is not granted.		
The group-execute permission character is given as s if the file has the set-group-id bit set; likewise the owner-execute permission character is given as S if the file has the set-user-id bit set.		
The last character of the mode (normally x or '-') is t if the 1000 bit of the mode is on. See chmod(1V) for the meaning of this mode. The indications of set-ID and 1000 bits of the mode are capitalized (S and T respectively) if the corresponding execute permission is not set.		
When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks is printed.		

OPTIONS

- a List all entries; in the absence of this option, entries whose names begin with a `.' are not listed (except for the super-user, for whom `ls`, but not `/usr/5bin/ls`, normally prints even files that begin with a `.').
- A (ls only) Same as -a, except that `.' and `..' are not listed.
- c Use time of last edit (or last mode change) for sorting or printing.
- C Force multi-column output, with entries sorted down the columns; for `ls`, this is the default when output is to a terminal.
- d If argument is a directory, list only its name (not its contents); often used with `-l` to get the status of a directory.
- f Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off `-l`, `-t`, `-s`, and `-r`, and turns on `-a`; the order is the order in which entries appear in the directory.
- F Mark directories with a trailing slash (`/'), executable files with a trailing asterisk (`*'), symbolic links with a trailing at-sign (`@'), and AF_UNIX address family sockets with a trailing equals sign (`=').
- g For `ls`, show the group ownership of the file in a long output. For `/usr/5bin/ls`, print a long listing, the same as `-l`, except that the owner is not printed.
- i For each file, print the i-number in the first column of the report.
- l List in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file. If the file is a special file the size field will instead contain the major and minor device numbers. If the time of last modification is greater than six months ago, it is shown in the format `month date year'; files modified within six months show `month date time'. If the file is a symbolic link the pathname of the linked-to file is printed preceded by `->`. `/usr/5bin/ls` will print the group in addition to the owner.
- L If argument is a symbolic link, list the file or directory the link references rather than the link itself.
- q Display non-graphic characters in filenames as the character `?'; for `ls`, this is the default when output is to a terminal.
- r Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.
- R Recursively list subdirectories encountered.
- s Give size of each file, including any indirect blocks used to map the file, in kilobytes (`ls`) or 512-byte blocks (`/usr/5bin/ls`).
- t Sort by time modified (latest first) instead of by name.
- u Use time of last access instead of last modification for sorting (with the `-t` option) and/or printing (with the `-l` option).
- 1 (ls only) Force one entry per line output format; this is the default when output is not to a terminal.

SYSTEM V OPTIONS

- b Force printing of non-graphic characters to be in the octal \ddd notation.
- m Stream output format; the file names are printed as a list separated by commas, with as many entries as possible printed on a line.
- n The same as `-l`, except that the owner's UID and group's GID numbers are printed, rather than the associated

character strings.

- o The same as -l, except that the group is not printed.
- p Put a slash ('/') after each filename if that file is a directory.
- x Multi-column output with entries sorted across rather than down the page.

ENVIRONMENT

The environment variables LC_CTYPE, LANG, and LC_default control the character classification throughout ls. On entry to ls, these environment variables are checked in the following order: LC_CTYPE, LANG, and LC_default. When a valid value is found, remaining environment variables for character classification are ignored. For example, a new setting for LANG does not override the current valid character classification rules of LC_CTYPE. When none of the values is valid, the shell character classification defaults to the POSIX.1 "C" locale.

FILES

/etc/passwd	to get user ID's for 'ls -l' and 'ls -o'.
/etc/group	to get group ID for 'ls -g' and 'usr/5bin/ls -l'.
/usr/share/lib/terminfo/*	to get terminal information for 'usr/5bin/ls'.

SEE ALSO

chmod(1V)

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as 'ls -s' is much different than 'ls -s | lpr'. On the other hand, not doing this setting would make old shell scripts which used ls almost certain losers.

None of the above apply to /usr/5bin/ls.

Unprintable characters in file names may confuse the columnar output options.

Dosyalar (Files)

Kullanıcıların, bir bilgisayarla yaptıkları çalışmaların meyveleri dosyalardır. Çizim programları kullanarak hazırladığınız çizimler, bir sonraki çalışma adımı için dosyalarda (disk, disket veya teypde) saklanır. Yazdığınız programlar dosyalar olarak saklanır ve kullanılır. Muhasebe kayıtları yıl boyunca dosyalarda biriktirilir. Bütün bu dosyaları yaratıp yaşatabilmeniz için gerekli desteği, size, kullandığınız bilgisayarın işletim sistemi verir. Dosyalarınızı düzenlemek, kopyalarını çıkarmak, yedeklerini almak, günlük işlerinizin önemli bir parçası olacaktır; bu nedenle UNIX'in dosya kavramını uzun uzun anlatmak istiyorum.

UNIX'de dosyalar, aynı MS-DOS'da olduğu gibi, ilgili oldukları uygulamaya göre düzenlenmiş dizinlerde (*directory*) saklanır. UNIX'in çok kullanıcı bir işletim sistemi olmasından dolayı, diskin veya disklerin kullanıcılar arasında paylaşılması, dosyaların konuları yanı sıra, kullanıcıların kimliklerine göre de gruplanmasını gerektirir. (Hatırlarsanız, her kullanıcının bir 'kullanıcı dizini' olduğundan daha önce söz etmiştim.)

Dosya ve dizinlerin kullanıcılar arasında paylaşılmasından dolayı, bir kullanıcıya ait dosyaların ve dizinlerin bir 'erişim yetkisi' mekanizması ile diğer kullanıcılara karşı korunması gerekmektedir. UNIX'de bu koruma mekanizması, kullanıcıların sisteme tanıtılması sırasında verilen '**kullanıcı ismi**', '**kullanıcı numarası**' ve kullanıcının ait olduğu '**çalışma grubunun numarası**' na dayandırılır. Normal şartlar altında, kullanıcıların kendilerine ait 'kullanıcı numarası'nı bilmelerine gerek yoktur, kullanıcı ismi ve kullanıcı numarası arasındaki bağlantı, işletim sistemi tarafından otomatik olarak sağlanır.

Dosya Yaratma

Dosya yaratmanın bir çok yöntemi vardır :

- Bir editör kullanarak metin dosyaları;
- Bir muhasebe programı kullanarak, yeni bir yıl için veri dosyaları;
- Bir bilgisayar destekli tasarım program paketi kullanarak çizim dosyaları;
- Yazdığınız bir C programını derleyerek amaç program (*object code*) ve makina kodu (*machine code, executable code*) dosyaları;
- Eski bir dosyanın kopyasını çıkararak yeni bir dosya;
- Teypten ya da CDROM'dan diske kopyalamak yoluyla diskte bir dosya;
- Program çıktılarını yönlendirerek (I/O Redirection) döküm dosyaları
- vs. vs. vs.

yaratabilirsiniz.

Biz en basitinden başlayalım :

cat komutu

UNIX'de çok sık kullanılan, çok işlevli bir komuttur. Bu işlevlerden bir tanesi,, MS-DOS daki TYPE komutu ile aynıdır.

En basit kullanım formu :

% cat dosya_adi	(concatenate)
-----------------	---------------

şeklindedir. Bu formda kullanıldığı zaman, **dosya_adi** adlı dosyayı ekrana (daha doğrusu, UNIX diliyle **STANDART ÇIKTI BİRİMİ**'ne : *Standard Output*) gönderir. Standart çıktı birimi genellikle ekran olduğu için, bir dosyayı ekrana listelemek için kullanılır.

Denemek için

```
% cat /etc/motd
```

komutunu verebilirsiniz.

Dosya yaratmak için kullanacağımız form ise biraz daha farklı....

% cat > yenisosya

Bu formda kullanıldığında, **cat** komutu, **STANDART GİRDİ BİRİMİ**'nden (klavyeden, *Standard Input*) aldığı bilgileri, **yenisosya** isimli bir dosyaya yönlendirecektir (bir başka deyişle kopyalayacaktır).

Şimdi isterseniz **dosya1** isimli ilk küçük dosyamızı yaratalım:

```
% cat > dosya1
```

komutunu veriniz,

daha sonra imleç (*cursor*) yeni satırın başına geldiğinde, dosyanın içinde yer almasını istediğiniz satırları giriniz; örneğin

```
abc:/home/ayfer % cat > dosya1
Bu bizim ilk deneme dosyamız.
İçinde sadece iki satır var.
^D
abc:/home/ayfer %
```

Girmek istediğiniz satırlar tamamlandıncaya, imleç satır başındayken Ctrl ve D tuşlarına birlikte basarak (*EOF : End of file karakteri*) standart giriş biriminizde dosya sonuna geldiğini belirtin.

Dosya adı verirken dizin adı belirtmediğiniz için, **dosya1** adlı dosya çalışma dizininizde yaratılır. Herhangi bir hata mesajı almadıysanız; dosya problemsiz yaratıldı demektir. Eğer **dosya1** in yaratılıp yaratılmadığını kontrol etmek isterseniz iki yöntem önerebilirim :

```
% cat dosya1
ve
% ls
```

Dikkat ederseniz ">" karakteri yok...

Birinci komut (**cat**), **dosya1** dosyasının içine yazdıklarınızı ekrana görüntüleyecek, böylece yaratma işleminin tamamlanıp tamamlanmadığını çok sağlam bir şekilde kontrol etmiş olacaksınız. İkinci komutlarsa (**ls**) sadece dosyanızın adını, uzunluğunu, ne zaman yaratıldığını ve sahibinin kim olduğunu göreceksiniz.

Bence iki yöntemi de deneyiniz.



Eğer **cat** komutunu parametresiz olarak verirsiniz, komut pek de anlamlı olmayan bir iş yapmaya; standart giriş biriminden okuyup, standart çıkış birimine kopyalamaya başlayacaktır. Yani klavyeden (standart giriş birimi) bastığınız her tuş, standart çıkış birimine (ekran) kopyalanacaktır.

Yanlışlıkla düşebileceğiniz bu durumdan kurtulmak için, imleç satır başındayken **Ctrl-D** tuşuna basınız. Bu hareketiniz kopyalama işini sona erdirecektir. Bu işlem, diskteki dosyalarınızı hiç bir şekilde etkilemez.

Küçük bir varyasyonla **cat** komutunu ilginç bir iş yapmak için de kullanabiliriz.

```
abc:/home/ayfer % cat >> dosya1
Bu satırlar, dosya1 dosyasının arkasına eklenecek.
ve dosya1 toplam 4 satır olacak.
^D
abc:/home/ayfer %_

abc:/home/ayfer % cat dosya1
Bu bizim ilk deneme dosyamız.
İçinde sadece iki satır var.
Bu satırlar, dosya1 dosyasının arkasına eklenecek.
ve dosya1 toplam 4 satır olacak.
abc:/home/ayfer %_
```



cat komutunu bir de şu şekilde deneyiniz :

```
cat > /dosya1
Bu bizim ilk deneme dosyamız.
İçinde sadece iki satır var.
^D
Access denied.
```

Evet, tahmin ettiğim gibi bir hata mesajı aldınız (*Access denied* : Bu işi yapmaya yetkiniz yok!).

Sebebi açık... **dosya1** isimli dosyayı **root** (/) dizinin hemen altında yaratmaya çalıştınız ve sizin bu dizine kayıt yapmaya yetkiniz olmaması da çok doğal.

Bu örneği **root** kullanıcı olarak yapmış olsaydınız (lütfen denemeyiniz!), böyle bir mesajla karşılaşmayacaktınız.

cp komutu (copy kelimesinden türemiştir)

Bu komutun ne işe yaradığını söylemeye gerek olduğunu sanmıyorum; ama nasıl kullanıldığı önemli...

En basit formuyla

```
% cp dosya_adi_1 dosya_adi_2 (copy)
```

dosya_adi_1 isimli dosyayı **dosya_adi_2** isimli dosyaya kopyalayacaktır.



Eğer **dosya_adi_2** isimli dosya yoksa, yaratılacaktır (tabii bu dosyanın yer alacağı dizinde dosya yaratmaya yetkiniz varsa...) Eğer bu isimde bir dosya eskiden varsa, üzerine kopyalama yapılacaktır ve eski içeriği bozulacaktır. Böyle bir durumda, **eski bir dosyanın üzerine kayıt yapmak üzere olduğunuz konusunda uyarılmayacaksınız!** Dikkatli olmanız gerekir.

Eğer dikkatinize güvenmiyorsanız, **cp** komutunu



```
% cp -i dosya_adi_1 dosya_adi_2
```

formunda kullanın. **-i** parametresi (*interactive*), eski bir dosyanın üzerine kayıt yapılması durumunda kullanıcının **Overwrite?** mesajı ile uyarılmasını ve ancak **y** yanıtı verilirse devam edilmesini sağlar.



-i parametresini kullanmayı unutmaktan korkuyorsanız, kitabın **alias** komutu ile ilgili bölümünü okuyunuz; bu bölümde çeşitli UNIX komutlarını kalıcı olarak değiştirmenin, hatta kendinize özgü UNIX komutları yaratmanın yollarını bulacaksınız.

Bir başka form :

```
% cp dosya_adi dizin_adi
```

dosya_adi isimli dosyayı, **dizin_adi** isimli dizinin altına kopyalar. İsterseniz -i opsiyonunu gene kullanabilirsiniz.



Bu formla ilk form arasında görünüş olarak hiç bir fark yoktur. İkinci parametreyle verilen isim bir dizine aitse, verdiğiniz komut ikinci form olarak kabul edilir ve birinci dosya bu dizinin altına kopyalanır. İkinci parametreyle belirtilen isimde bir dosya varsa, ya da bu isimde hiç bir şey (dosya veya dizin) yoksa; ilk form kabul edilerek ilk parametredeki dosyanın kopyası çıkarılır.

Hemen bir örnek :

Çalışma dizininizde

```
-rw-rw-rw- 1 root 918 Jan 12 12:09 ocak1995
-rw-rw-rw- 1 root 918 Jan 12 12:09 subat1995
```

dosyaları varken,

```
% cp ocak1995 veriler
```

komutu verirseniz, çalışma dizininizdeki yeni dosyal listesi

```
-rw-rw-rw- 1 root 918 Jan 12 12:09 ocak1995
-rw-rw-rw- 1 root 918 Jan 12 12:09 subat1995
-rw-rw-rw- 1 root 918 Jan 14 17:43 veriler
```

şekline dönüşür.

Eğer komutu vermeden önce; çalışma dizininizin görünüşü

```
-rw-rw-rw- 1 root 918 Jan 12 12:09 ocak1995
-rw-rw-rw- 1 root 918 Jan 12 12:09 subat1995
drwxrwxrwx 1 root 918 Jan 14 17:43 veriler
```

gibi idiyse; (**veriler** isimli bir dizin bulunduğuna dikkatinizi çekerim) **cp** komutundan sonra, **ocak1995** dosyası, **veriler** dizinin altına kopyalanmış olacaktır.

Bir başka form :

```
% cp dosya1 dosya2 dosya3 .... dizin_adi
```

dosya1, dosya2, dosya3, vs. vs. isimli dosyaları, **dizin_adi** isimli dizinin altına kopyalar. İsterseniz **-i** parametresini kullanabilirsiniz.



```
% cp dosya1 dosya2 ... dosyaN dizin_adı
```

formunu kullandığınızda, **dizin_adi** adlı dizin bulunamazsa, garip şeyler olacaktır. Önce **dosya1** isimli dosya **dizin_adi** isimli bir dosyaya kopyalanacaktır. Sonra **dosya2** isimli dosya gene **dizin_adi** isimli dosyanın üzerine kopyalanacaktır. Daha sonra da bu işlem **dosya3** ve varsa diğer dosyalar için tekrarlanacaktır. Sonunda, **dosyaN** in bir kopyası **dizin_adi** isimli bir dosyaya çıkarılmış olacaktır.



MS-DOS kullanıcıları... Dikkat!

UNIX **cp** komutunda, kopyalamanın **nereden nereye** yapılacağını mutlaka açıkça belirtmelisiniz. Yani,



```
% cp /dizin1/dosya1
```

şeklinde bir komut **kullanamazsınız**. Bu şekilde yazacağınız bir komut, **/dizin1**'in altındaki **dosya1** isimli dosyayı, çalışma dizinine kopyala anlamına **gelmez**; hatalı bir komuttur.

Dizin Kopyalama

UNIX'de, **dizin** kopyalamak için de **cp** komutu kullanılır; ancak özel bir parametreyle birlikte..


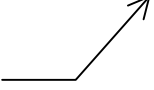

Dizin kopyalamak için kullanılan form :

```
% cp -r dizin1 dizin2          dizin kopyalama
```

şeklinde. Bu formda verilen kopyalama komutu; varsa, dizinlerin alt-dizinlerinin de kopyalanmasını sağlar. (MS-DOS'da **/S** parametresiyle kullanılan **XCOPY** komutu gibi...)

MS-DOS kullanıcıları, el alışkanlığıyla sık sık, **cp** yerine **copy** yazacaklardır. Sizde böyle bir sıkıntınız olur ve kurtulmak isterseniz **alias** komutuyla

ilgili bölümünü okuyunuz; bu komutla **copy** ismiyle kendi kopyalama komutunuzu tanımlayabilirsiniz. **cp** komutu üzerine bir kaç gelişmiş örnek vererek bu konuyu geçmek istiyorum :

<pre>% cp /etc/motd /tmp/motd2</pre>	<p>/etc isimli dizinin altındaki motd isimli dosyayı, /tmp isimli dizinin altına, adını motd2 olarak değiştirerek kopyala.</p>
<pre>% cp /etc/motd /tmp</pre>	<p>/etc isimli dizinin altındaki motd isimli dosyayı, /tmp isimli dizinin altına, adını değiştirmeden kopyala.</p>
<pre>% cp /etc/motd .</pre> <div data-bbox="651 748 762 808" style="border: 1px solid black; padding: 2px; display: inline-block;">Nokta</div> 	<p>/etc isimli dizinin altındaki motd isimli dosyayı çalışma dizinine (bir başka deyişle, buraya) kopyala. (Tek nokta, 'burası' anlamındadır; aynı MS-DOS'daki gibi).</p>
<pre>% cp /home/ayfer/prg1 ..</pre> <div data-bbox="399 981 529 1041" style="border: 1px solid black; padding: 2px; display: inline-block;">İki nokta</div> 	<p>/home isimli dizinin altındaki ayfer dizinin altındaki prg1 isimli dosyayı çalışma dizininin bir üstündeki dizine kopyala. (İki nokta yanyana, 'bir üst dizin' anlamındadır; aynı MS-DOS'daki gibi).</p>
<p>Çarpıcı bir örnek:</p> <pre>% rcp abc:/vmunix temel:/sakla</pre> <div data-bbox="539 1263 699 1323" style="text-align: center;">  </div> <p>rcp : Remote Copy</p>	<p>abc bilgisayarının root dizinindeki vmunix isimli dosyayı temel isimli bir başka bilgisayardaki, varsa sakla isimli dizinin altına; böyle bir dizin yoksa, root dizinin altına sakla ismiyle kopyala.</p> <p>(Bu komutu verebilmeniz için, abc ve temel isimli bilgisayarların bir bilgisayar ağı ile birbirlerine bağlı olmaları ve sizin iksine dei erişim hakkınız olması gerekmektedir.) Bu ve buna benzer komutlara daha sonra ayrıntılı olarak değineceğim.</p>

Dizin Yaratma

UNIX'de, dizin yaratmak için **mkdir** komutu kullanılır.

Formu basittir :

```
% mkdir dizin (make directory)
% mkdir eski_dizin/yeni_dizin
```

gibi...

Doğal olarak, yalnızca yetkiniz olan yerlerde dizin yaratabilirsiniz...



Unutmayınız !

Komutlar hakkında daha detaylı bilgiye gereksinim duyarsanız; örneğin parametrelerini hatırlayamazsanız; **man** komutu her zaman kullanımınıza hazırdır.

Sırf alışkanlık kazanmak amacıyla hemen şimdi

```
% man mkdir
```

komutunu vermeye ne dersiniz?

Dosya Silme

Artık diskte yer almasını istemediğiniz dosyaları silmek için kullanacağınız komut

```
% rm dosya (remove)
% rm dosya1 dosya2 dosya3 ... dosyaN
```

formlarındadır.

Bir seferde (tek komutta), farklı dizinlerde yer alan dosyaları da silebilirsiniz.

```
% rm /dizin1/dosya1 /baska_dizin/dosya2 ...
```

Eğer dosyalar silinmeden önce onaylamak istiyorsanız **-i** parametresini kullanabilirsiniz:

```
% rm -i /dizin1/dosya1 /baska_dizin/dosya2 ...
```


komut formunu kullandığınızda, silinecek her dosya için teker teker

remove ?

sorusu sorulacak ve sadece **y** yanıtını verdiğiniz dosyalar silinecektir.

Dizin Silme

Artık diskte yer almasını istemediğiniz dizinleri, altlarındaki dosya ve alt dizinleriyle birlikte silmek için kullanacağınız komut

```
% rm -r dizin (remove)
% rm -r dizin1 dizin2 dizin3 ... dizinN
```

formlarındadır.

Bir seferde, farklı dizinlerde yer alan dizinleri de silebilirsiniz.

```
% rm /dizin1/alt_dizin1 /baska_dizin/dizin2 ...
```



Eğer dizinlerin silinmeden önce tarafınızdan onaylanmasını istiyorsanız **-i** parametresini kullanabilirsiniz:

```
% rm -i /dizin1/alt_dizin1 /baska_dizin/dizin2 ...
```

komut formunu kullandığınızda, silinecek her dizin için teker teker

remove ?

sorusu sorulacak ve sadece **y** yanıtını verdiğiniz dizinler silinecektir.



Çok Önemli !

UNIX işletim sisteminde UNDELETE görevini yerine getirecek bir program ya da komut yoktur. Sildiğiniz dosya ve dizinler, bir daha geri getirilemeyecek şekilde silinir. Bu nedenle rm komutunu kullanmadan önce iyi düşünmelisiniz.

Dosya/Dizin Adı Değiştirme

Bu iş için kullanacağınız komut **mv** (move) komutudur. Dosya ve dizin ismi değiştirmek için kullanılan formu basittir :

```
% mv eski_dosya_ismi yeni_dosya_ismi      (move)
% mv eski_dizin_ismi yeni_dizin_ismi
```

Bir isim değişikliği yapmak istediğinizde, doğal olarak, söz konusu dosya veya dizinin yer aldığı dizinde, yeni isimde bir dosya ya da dizin bulunmamalıdır.

Dosya/Dizin Yeri Değiştirme

Bu iş için kullanacağınız komut yine **mv** (move) komutudur. Dizin yeri değiştirmek için kullanılan özel **-R** parametresine dikkatinizi çekerim.

```
% mv eskiyeri\dosya yeniyeri\dosya      move
% mv -R eskiyeri\dizin yeniyeri\dizin
```

Bir yer değişikliği yapmak istediğinizde, doğal olarak, söz konusu dosya veya dizinin yer alacağı yeni dizinde, aynı isimde bir dosya ya da dizin bulunmamalıdır.

Çalışma Dizinini Değiştirme


Aynı MS-DOS işletim sisteminde olduğu gibi, bir dosyanın adını verirken, dosyanın yer aldığı dizini tam olarak belirtmezseniz, dosyanın o andaki çalışma dizininizde bulunduğu varsayılır. Çalışma dizinini değiştirmek için kullanılan komut

```
% cd yeni_calisma_dizini      (change directory)
```

aynı MS-DOS'daki CD komutu gibidir.

Örnekler vermek gerekirse :

% cd /home/ayfer/proje1	Pek açıklama gerektirmiyor sanırım...
% cd ../proje2	Bir üstteki dizinin altındaki proje2 isimli dizine geç.

% cd ../../mektuplar	İki üst düzeydeki dizinin altındaki mektuplar isimli dizine geç.
Kullanışlı bir olanak : % cd ~omer	Kullanıcı adı omer olan kullanıcının kullanıcı dizinine geç. (omer 'in home dizini)
Hoş bir olanak daha... % cd 	Her nerede olursan ol, şu anda geçerli olan kullanıcı adına ait home dizinine geç. (Yuvaya dönüş!)



Dizinler arasında gidip gelirken, zaman zaman kaybolmanız doğaldır. Özellikle **prompt**'unuz (hazır işaretiniz) çalışma dizininiz hakkında bilgi vermiyorsa...

Kaybolduğunuzda, **pwd** komutu ile (*print working directory*) o andaki çalışma dizininizin hangi dizin olduğunu öğrene-bilirsiniz.

Buraya kadar temel bir kaç UNIX komutundan; sık sık da yetkilerden söz ettik. Sanırım şu **yetki** meselesini biraz daha açmanın zamanı geldi...

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 tanıttacaktır. Bu işlemler tamamlandığında UNIX bilgisayarının herhangi bir terminalinden **login** edebilirsiniz. Sisteme ilk bağlantınızda 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.

r	w	x	r	w	x	r	w	x
Dosya sahibinin yetklileri			Grubun Yetklileri			Diğerlerinin Yetklileri		

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



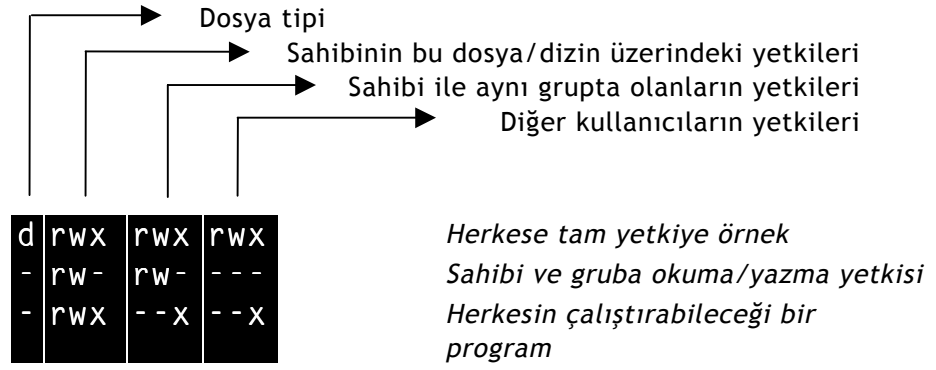
Dizinler için de `rw-rw-rw-` yetki kodları söz konusudur. Dosya yetki kodlarına çok benzemekle beraber, detaylarda bazı önemli farklılıklar vardır. Bu farkları daha sonra açıklayacağım.



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.

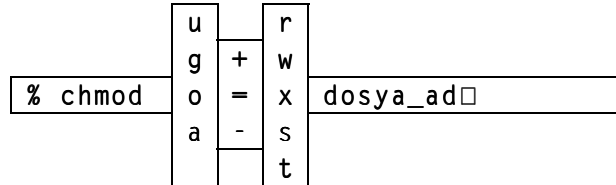
Dosya ve dizinlerin erişim yetkilerini değiştirmek için

% 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	4	2	1	4	2	1
r	w	x	r	w	x	r	w	x
Owner			Group			Others		

Diyelimki **adresler** dosyasının erişim yetkilerinin **rw-r-xr-x** olmasını istiyorsunuz. Bu yetki kalıbını üçer üçer ayrılmış olarak düşünüp (**rw** **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
```

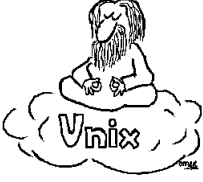


Bir dosya veya dizinin erişim yetkilerini **SADECE** dosyanın sahibi ve **root** kullanıcı değiştirebilir.

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 **rwrx-rx-rx** 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)

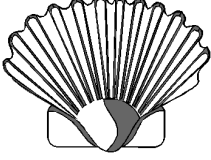
Bu komutu sadece root kullanıcı kullanabilir!

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/dizin o kullanıcı tarafından tam yetkiyle kullanılmasını isterse, bunu sağlamanın en kolay yolu, bu yeni yaratılan dosya/dizin 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 irdelleyecek, **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 yazılmışçasına 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

<code>% cat kitap[1-3] > hepsi</code>	kitap1 , kitap2 ve kitap3 dosyalarını peşpeşe ekleyerek hepsi isimli dosyaya kopyalar.
<code>% chmod a=x *. [o,sh]</code> veya <code>% chmod a=x * [o,.sh]</code>	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 .!* .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	<i>csh için</i>
-----------------------------------	-----------------

Bourne shell (sh) için

\$ OPENWINHOME=/usr/openwin	<i>sh için</i>
\$ 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 .)	<i>csh için</i>
--	-----------------

veya bunun eşdeğeri olan

\$ PATH=/bin:/usr/bin:~/bin:.	<i>sh için</i>
-------------------------------	----------------

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                csk için
```

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

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

```
% setenv TERM vt100                    csk için
```

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

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

```
% setenv MANPATH /usr/man:/usr/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 "Güle Güle..."
```

.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.

```
#####
# Örnek bir .cshrc dosyası#
#####
#
set lpath=( /home/ayfer/bin )
set path= ( /usr/bin /usr/local/bin /bin $lpath )
set cdpath = (~ /src ~/bin ~ )
set noclobber
set history=30
set ignoreeof
#
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 ignoreeof
```

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, izin isimlerinin sonuna bir / işareti; çalıştırılabilir 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 izinler için erişim yetkileri söz konusuydu. Her dosya ve izin 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 **rxwxrwx** 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 **rx rx rx** yetki kodları ile alt alta yazın.

Dosyalar için	Dizinler için
rw- rw- rw-	rx rx rx
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, **rxr-xr-x** olarak belirlenecektir.

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



Kararsız kullanıcılara önerim, **umask** olarak 022 kullanmalarıdır.

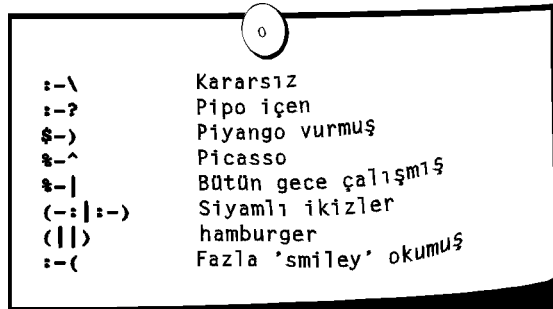


UNIX işletim sistemiyle birlikte kullanılabilecek pek çok editör vardır; **emacs**, **edit**, **ed**, **ex** gibi.... Ancak bunlar arasında, tüm UNIX uyarlamalarında standart olarak bulunan editör **vi**'dir. Bu nedenle, her UNIX kullanıcısının, az da olsa **vi** editörünü kullanmayı bilmesi gerekmektedir. 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!

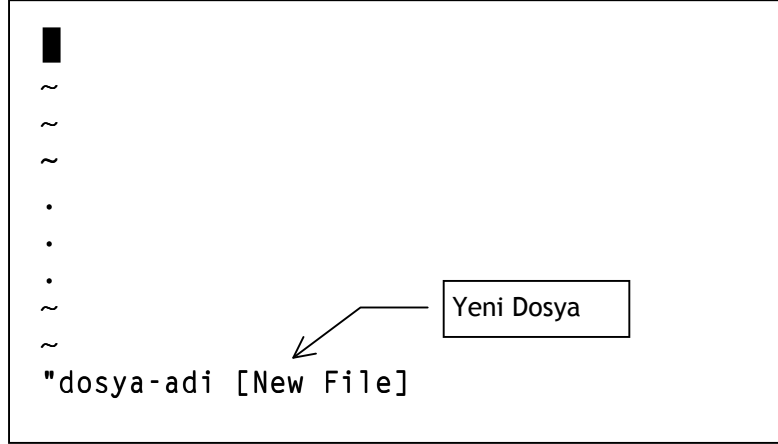
vi bir kelime işlemci değildir! Sadece bir editördür. UNIX altında kelime işlem uygulaması yapmak isterseniz **latex**, **nroff** ve **troff** gibi yazılımlar kullanmanız gerekecektir; ama, bu yazılımların kullanımının temelinde gene **vi** bulunacaktır.




Dosya Açma / Yaratma

% vi dosya-adi komutuyla **dosya-adi** adlı dosyayı edit etmeye başlayabilirsiniz

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



Bu ekranda  işareti imleci (*cursor*) temsil etmektedir. ~ işaretleri ise 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 dosyamızdır.
Enter (veya RETURN) tusuna basarak alt satıra 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 karşılaşılabileceğiniz 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 izin 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.

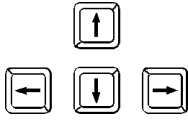
Disk Kotası :

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, imleci ekranda dolaştırma komutlarıdır.

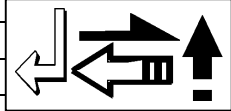
İmleç Dolaştırma Komutları



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; eksiğe 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 işlemi geri döndürebilirsiniz (**Aslında sadece son silme işlemi 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, **'Aynı Windows ...'** kelimeleri ile başlayan satırın arkasına taşımak istediğimizi varsayalım : Önce imleci bu satırın üzerine getirmeliyiz (**dd** komutuna hazırlık)

vi editoru ile metin parçalarının yerini değiştirmek
istediginizde, önce eski yerinden silmeli, daha
sonra yeni yerine yapıştırmalıyız.
Aynı Windows cut & paste gibi...
Oldukça kolay, değil mi ?

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

vi editoru ile metin parçalarının yerini değiştirmek
sonra yeni yerine yapıştırmalıyız.
Aynı Windows cut & paste gibi...
Oldukça kolay, değil mi ?

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

vi editoru ile metin parçalarının yerini değiştirmek
sonra yeni yerine yapıştırmalıyız.
Aynı Windows cut & paste gibi...
Oldukça kolay, değil mi ?

p (küçük p) komutu ile yeni yerine yapıştırabiliriz.

vi editoru ile metin parçalarının yerini değiştirmek
sonra yeni yerine yapıştırmalıyız.

Aynı Windows cut & paste gibi...

istediginizde, önce eski yerinden silmeli, daha

Oldukça kolay, değil 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, önce...**' kelimeleri ile başlayan ikinci satırı, '**Aynı Windows ...**' kelimeleri ile başlayan satırın arkasına kopyalamak istediğimizi varsayalım :

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

vi ile metin parçaları başka yerlere kopyalanmak

istendiginde, önce eski yerinde belleğe alınmalı, daha

sonra yeni yerine yapıştırmalıyız.

Aynı Windows cut & paste gibi...

Oldukça kolay, değil mi ?

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

vi ile metin parçaları başka yerlere kopyalanmak

istendiginde, önce eski yerinde belleğe alınmalı, daha sonra yeni yerine yapıştırılmalıdır.

Aynı Windows cut & paste gibi...

Oldukça kolay, değil 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 parçaları başka yerlere kopyalanmak

istendiginde, önce eski yerinde belleğe alınmalı, daha sonra yeni yerine yapıştırılmalıdır.

Aynı Windows cut & paste gibi...

Oldukça kolay, değil mi ?

p (küçük p) komutu ile yeni yerine yapıştırabiliriz.

vi ile metin parçaları başka yerlere kopyalanmak

istendiginde, önce eski yerinde belleğe alınmalı, daha sonra yeni yerine yapıştırılmalıdır.

Aynı Windows cut & paste gibi...

istediginizde, önce eski yerinde belleğe alınmalı, daha

Oldukça kolay, değil mi ?



Kopyalama ve yer deęiřtirmede kullanılan **p** (küçük p) komutu yerine **P** (büyük p) komutu kullanılırsa, yapıştırma, imlecin bulunduęu satırın altına deęil, ÜSTÜNE yapılır.

Son Komutu Tekrarlama

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

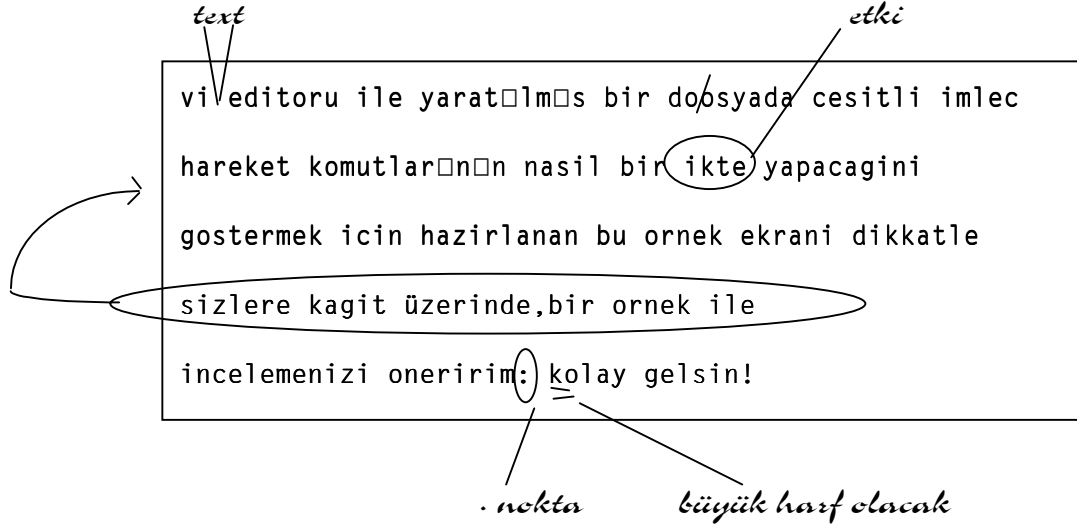
Metin Eklemenin / Deęiřtirmenin Bir Kaç Deęiřik Yolu

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

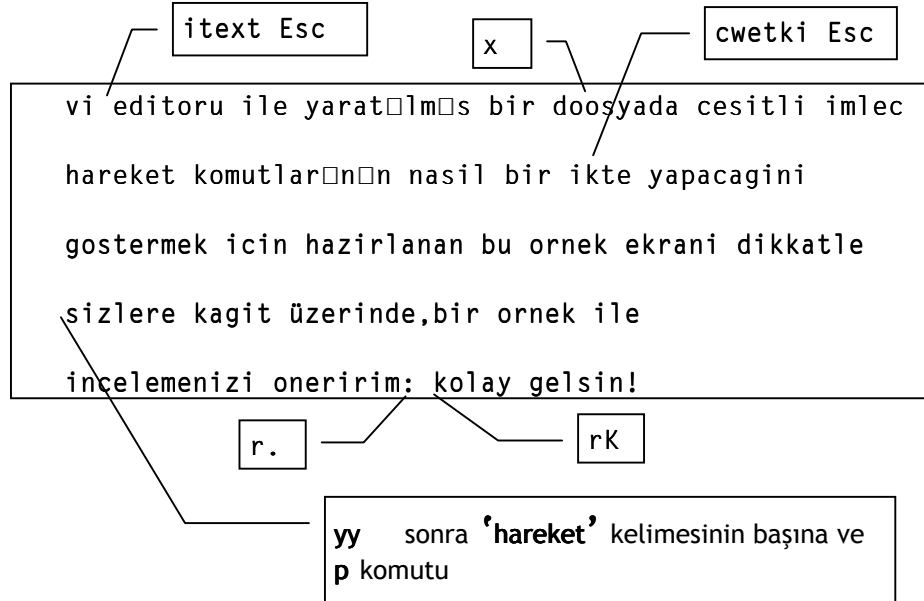
Bir sonraki sayfada, daha önceden hatalı olarak dökümü yapılmıř bir metin üzerinde kalemle yapılmıř düzeltmelerin dosyaya işlenmesini canlandıran bir örnek bulacaksınız. Dikkatle incelemenizi öneririm.

Bir Örnek

İstenen Düzeltmeler



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



Düzeltilmiş Metin

vi text editoru ile yaratılmıs bir dosyada cesitli imlec hareket komutların nasıl bir etki yapacağını sizlere kagit üzerinde,bir ornek ile gostermek icin hazirlanan bu ornek ekrani dikkatle incelemenizi öneririm. 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 (<i>Forward</i>)	Bir ekran İLERİYE
CTRL B (<i>Backward</i>)	Bir ekran GERİYE
CTRL D (<i>Down</i>)	Yarım Ekran İLERİYE
CTRL U (<i>Up</i>)	Yarım Ekran GERİYE
CTRL R (<i>Redraw</i>)	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 (<i>Home</i>)	EKRANIN EN ÜST satırına git.
M (<i>Mid Screen</i>)	EKRANIN ORTA satırına git
L (<i>Lower Screen</i>)	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 RASTLADIDI **Unix** karakter dizisini **UNIX** dizisiyle (veya **nix** dizisini **NIX** dizisiyle) değiştirecektir.

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

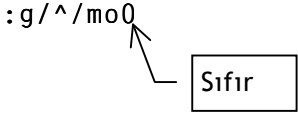
```
:s/Unix/UNIX/g
```

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!)

	<p>Çılgın bir örnek ! Bir dosyadaki satırların sırasını ters çevirir... (Son satırı birinci satır, sondan ikinci satırı ikinci satır, ...)</p> <p>Nasıl mı? Başlangıcı olan her satırı (^) (zaten her satırın bir başlangıcı vardır) sıfırıncı satırın altına taşır. (mo : <i>move</i> anlamındadır). Bu kısacık komutun bu işi yapacağına inanmıyorsanız bir deneyin. Ben bu örneğe ilk rastladığımda çalışacağına inanmayıp hemen denemek ihtiyacını duymuştum.</p>
---	--

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 yararlanabilmek 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		
O	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)

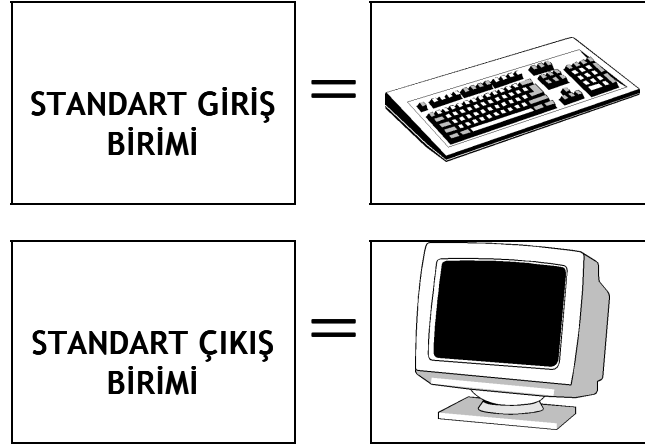
Blok kopyalamak için

- 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,

<code>% set ignoreeof</code>	<i>(ignore end of file)</i>
------------------------------	-----------------------------

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 kullanılsa 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 yönlendirerseniz** 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, tanımlı 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 bilgisayar 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 gruptandırımdım. 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 [ -seccenekler ] [ 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] [yıl] (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 kullanarak 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. Alışınca 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.

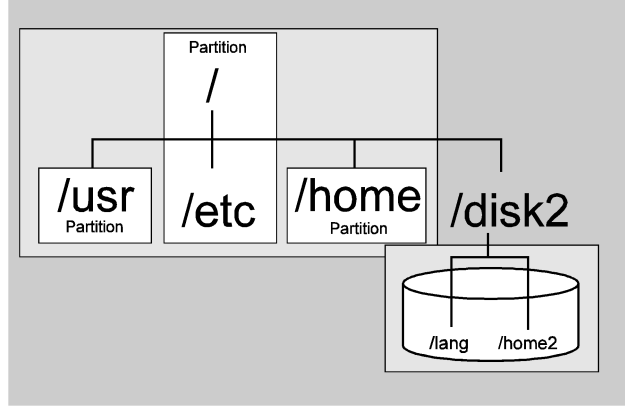
Şimdilik, UNIX bilgisayarınızın tek başına çalışan (bir bilgisayar ağına bağlı olmaksızın) bir bilgisayar olduğunu varsayalım. Bilgisayarınızın iki disk, bir disket, bir de CD-ROM sürücüsü olsun.

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 diskler C: D: E: ve F: isimleriyle ulaşırdık.

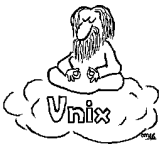
Şimdi sıkı durun: UNIX kullanıcılarının, disklerin ne şekilde ayrılmış olduğundan; hatta bilgisayarda kaç disk sürücü bulunduğu haberi olması bile gerekmemektedir. UNIX'de tüm diskler ve disk parçaları (*partition*'lar), root (/) dizinin altında birer alt dizin olarak yer alacaktır.

Şematik olarak göstermek gerekirse :



Her disk ve disk parçası üzerinde diğerlerinden bağımsız bir **dosya sistemi (file system)** bulunmalıdır. Bu sistemler, diskler (ya da disk parçaları) üzerinde, formatlama işleminden sonra **mkfs** komutu ile yaratılmaktadır. (Merak etmeyin; bu işin yapılmasından siz değil; sistem yöneticiniz sorumludur.)

Genellikle, **boot** diskinizin (bilgisayar açıldığında UNIX işletim sisteminin yüklendiği disk) ilk parçası size root (/) dizini olarak görünür. Diğer disk ve disk parçalarıysa bu dizinin altındaki alt dizinler olarak görünür. UNIX geleneğine göre **boot** diskleri an az 3 parçaya bölünür. İlk parça / , ikinci parça **/usr**, üçüncü parça ise **/home** dizini olarak isimlendirilir. Aslında pek yeri değil ama sanırım biraz daha ayrıntılı açıklama yararlı olacak.

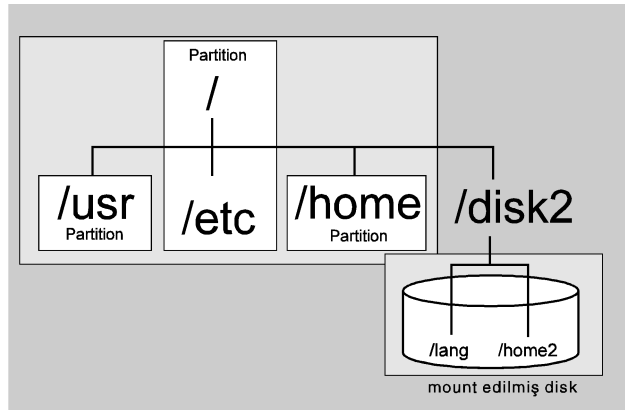
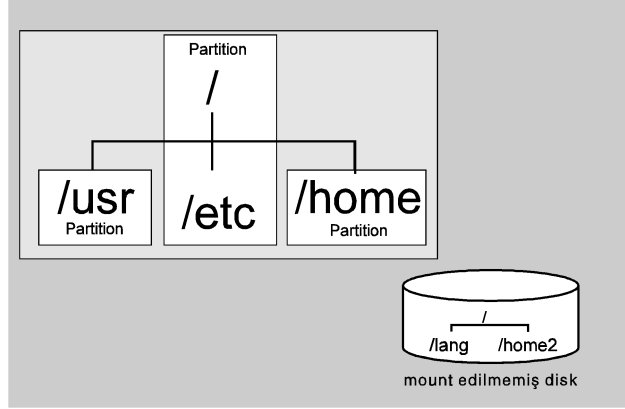


/ dizini, bilgisayarın açılabilmesi için gerekli olan dosyaların ve alt dizinlerin yer aldığı dizin; **/usr** dizini, tüm kullanıcıların ortak olarak kullanacağı çeşitli derleyici ve servis programlarının yer aldığı dizin; **/home** diziniyse, adından da anlaşılacağı gibi kullanıcıların kendilerine özgü dosyalarını yerleştirecekleri **home** dizinlerinin yer aldığı dizindir. Bu yerleştirme tarzı UNIX geleneğinin bir parçasıdır. Aynen uygulanması gerekmez de, genellikle tüm UNIX sistemlerinde diskler bu veya buna çok benzeyen bir şekilde düzenlenir. Bu düzenlemenin yararlarını daha ilerideki bölümlerde (özellikle sistem yöneticilerini ilgilendiren konulara gelince) anlatacağım.

Üzerinde bir dosya sistemi olan bir disk birimine veya parçasına, okuma veya yazma amacıyla ulaşabilmeniz için, o dosya yapısının, / dosya yapınızda bir alt dizine **mount** edilmiş olması gerekmektedir. (/ dizini, bilgisayarın açılması sırasında otomatik olarak **mount** edilmektedir. Eğer bu / dizini, bilgisayarın açılması aşamasında **mount** edilemezse, o bilgisayar zaten açılmaz; bu durumda mutlaka teknik desteğe gereksiniminiz vardır. Diğer disk veya disk

parçalarının otomatik olarak **mount** edilmesi için gerekli işlemlerse, sistem yöneticiniz tarafından yapılmalıdır.

UNIX'deki dosya-dizin yapılarını ters duran bir ağaca benzetirsek, **mount** etme işlemini, bir ağacı, bir başka ağacın dallarından birine iliştmek (monte etmek) gibi düşünebilirsiniz.



Şimdi isterseniz, kullandığınız bilgisayarda kaç disk ve/veya disk parçası olduğuna ve bunların hangi dizinlere **mount** edildiğine bir bakalım. Bu iş için lütfen terminalinizden şu komutu veriniz :

```
% mount
```

Tipik olarak şöyle bir liste almalısınız

```
% mount
/dev/sd0a on / rw 4.2
/dev/sd0g on /usr rw 4.2
/dev/sd0h on /home rw 4.2
%
```

(Bu örnek, SUNOS 4.1.3 UNIX işletim sistemiyle çalışan bir iş istasyonundan alınmıştır. Sizin kullandığınız bilgisayarda alacağınız liste bununla aynı olmayabilir).

Bu listeden, bilgisayarımızda sadece bir disk olduğunu (sadece **/dev/sd0** serisi bulunduğundan anlaşılıyor) ve bu diskin en az üç parçaya ayrıldığını (parça isimleri a, g ve h) veya sadece üç parçasının o anda **mount** edilmiş durumda olduğunu (**a** parçası **/** olarak, **g** parçası **/usr** dizini olarak ve nihayet **h** parçası da **/home** dizini olarak) anlıyoruz. **mount** komutunun verdiği listedeki satırlarda yer alan **rw** harfleri, söz konusu disk parçalarının oku-yaz (**read-write**) olarak kullanıma sunulduğunu (tabii ki, kullanıcıların yetkilerinin izin verdiği ölçüde) belirtmektedir. 4.2 sayıysa, SUNOS 4.1.3 işletim sistemine özgü bir dosya sistemi sürüm kod numarasıdır (*File System Version*).



mount komutu hakkında daha detaylı bilgiyi sistem yönetimi ile ilgili bölümlerde bulabilirsiniz. UNIX işletim sisteminin bir çok türevinde **mount** komutunu parametrelerle birlikte kullanabilmeniz için süper kullanıcı yetkilerine sahip olmanız gerekecektir; yani eğer süper kullanıcı (**root**) değilseniz, zaten, **mount** komutunu yalnızca parametresiz olarak kullanmanıza izin verilecektir.

Bir UNIX bilgisayarı açıldığında, otomatik olarak **mount** edilmesi istenen diskler ve **mount** edilecekleri dizinler **/etc/fstab** (BSD UNIX) veya **/etc/vfstab** (SVR4 UNIX) dosyalarında tanımlanır. Bu dosyalara sadece **root** kullanıcının yazma yetkisi vardır; bu nedenle bu dosyalara korkmadan bakabilirsiniz. (**more /etc/fstab** gibi bir komut işe yarayabilir; ne dersiniz?)

Disket sürücüler ve CD-ROM sürücüler de küçük birer disk sürücü olarak düşünülebilirler; bu nedenle kullanılabilmeleri için önce **mount** edilmeleri gerekir. Ancak; hem disketler, hem CD'ler, takılıp çıkarılabilir birimler olduklarından, bilgisayar açılırken otomatik olarak **mount** edilmezler. Normal olarak, bir disket veya CD'yi **mount** etmek için yeterli yetkiniz olmayacağından, bu tip birimlerin **mount** edilmesi konusunda sistem yöneticisinden yardım istemelisiniz. **mount** edilecek birimin, **mount** işleminden sonra hangi dizin altında görünmesi gerektiğine siz karar verebilir ve bu dizini siz yaratabilirsiniz. Bazı sistem yöneticileri, normal kullanıcılar tarafından çalıştırılabilen ve disket/CD **mount** işlemini yapan komutlar yaratırlar. Sizin çalıştığınız sistemde de böyle bir olanak olup olmadığını araştırınız. (Hayat Bilgisi kitabı gibi oldu, değil mi?)

İşi biten disket ve CD'ler **unmount** edilmelidir; yani, bu birimlere takılı medyalar üzerindeki dosya sistemlerinin, **root** dosya sistemiyle bağlantısı kesilmelidir. Aynı **mount** komutu gibi **unmount** komutu için de yöneticinizin yardımına gereksiniminiz olabilir..



Haaaaa. Bu arada.... Sistem yöneticinizle iyi geçinin. Zaman zaman kendisinin ne kadar inanılmaz yeteneklerle donanmış olduğunu; bu kadar çok şeyi bilebildiğine göre dehşetli zeki olduğunu kendisine hatırlatmayı ihmal etmeyin. Aslında bu özelliklerini kendisi çok iyi biliyordur; ama gene de hatırlatılmasından hoşlanacaktır. Arada bir ona hediyeler alın; özellikle sizin için geç saatlere kadar çalışacaksa yemek zamanında onun için bir pizza ve büyük kola getirmeyi sakın ihmal etmeyin. Yalnız dikkatli olun; UNIX guruları, içinde yeşil malzeme olan pizza yemezler (UNIX geleneği)!

Süreçler

(Processes)

UNIX işletim sisteminin **çok kullanıcı** ve **çok işli** bir işletim sistemi olduğunu şimdiye kadar bir kaç kez vurgulamıştım. Burada bir daha açıklamak gerekirse; UNIX işletim sisteminin denetimindeki bir bilgisayar hem aynı anda birden fazla kullanıcıya hizmet edebilir, hem de her kullanıcının aynı anda birden fazla işi yapmasına olanak sağlar. UNIX, kendi işlerini de bir sürü programı aynı anda çalıştırarak yapar. Örneğin, kullanılmayan terminallerin açılıp açılmadığını kontrol eden **getty** (bazı UNIX'lerde **init**) programları, kullanıcıların birbirlerine gönderdikleri mesajları gözleyen ve gelen-giden mesajları uygun posta kutularına yönlendiren **mail server** programı, bilgisayar ağı üzerinden gelen istekleri değerlendiren **inetd** programı, belirli aralıklarla disklere yapılan kayıt işlemlerinin fiziksel olarak disklere kaydedilmesini (*flushing disk buffers*) sağlayan **update** programı gibi... (Tipik bir UNIX bilgisayarında, kullanıcı programları dışında 30-40 sistem programı sürekli çalışıyor durumdadır.)

Bir UNIX bilgisayarında, belirli bir anda, merkezi işlem birimini (ya da birimlerini) ve belleği paylaşarak, birlikte çalışan programlara genel anlamda **PROCESS (süreç)** adı verilir. Süreçlerin Merkezi İşlem Birimi (MİB) zamanını paylaşımları UNIX tarafından koordine edilir (İşletim Sistemlerine ilişkin İngilizce terminolojide : *Process Scheduling* işlevi). MİB paylaşımına ilişkin önemli bir terim de '**zaman dilimi**' (*time slice*) kavramıdır. Her süreç, MİB'ni belirli ve kısa bir süre için (tipik olarak 10 - 100 milisaniye) sürekli olarak kullanabilir. Zaman dilimini dolduran süreçler beklemeye alınıp, MİB sırada bekleyen bir başka sürece tahsis edilir. Bu şekilde tüm süreçler aynı anda çalışıyormuş gibi bir etki elde edilir. Bu süreçlerin birden fazla kullanıcıya ait olmaları durumunda da, MİB kullanıcılar arasında paylaştırılmış olur. UNIX'in çok kullanıcı olma özelliğinin altında yatan temel mekanizma budur

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

% ps -axl	<i>Berkeley UNIX için process status</i>
% ps -efl	<i>SVR4 UNIX için process status</i>

olacaktır. Aslında, bu komutun gerek BSD (Berkeley), gerekse SVR4 UNIX için daha bir çok parametresi olabilmektedir. Bu parametreleri merak eden kullanıcılar, **man** komutu yardımıyla kullandıkları UNIX'in **ps** komutunun detaylarını öğrenebilirler.

Komutu parametresiz kullanırsanız, yalnızca kendinize ait süreçlerin bir listesini alırsınız. (Alacağınız bu liste BSD veya SVR4 UNIX'ler için biraz farklı olacaktır, fakat içerdikleri bilgi açısından eşdeğer sayılabilirler; bu nedenle sadece BSD UNIX'den örnekler vereceğim).

% ps

```

PID TT STAT  TIME COMMAND
1210 co IW    0:01 sunview
1234 p0 S      0:43 shelltool
1226 p0 R      2:46 shelltool
1456 co R      0:04 ps
1605 co S      0:01 -bin/csh (csh)
. . .
. . .
%
```

Bu listedeki önemli bilgiler şunlardır:

PID	(<i>Process ID</i>) UNIX'de, her sürecin birer tanıtım numarası vardır. Aynı numaraya sahip iki süreç olamaz.
TT	(<i>Teletype</i> : Çok eskilerden kalan bir alışkanlık) Sürecin hangi terminalden başlatıldığı (genellikle co : <i>console</i> , ttya : a isimli seri arabirim, p0 bilgisayar ağı üzerinden bağlanmış bir ekran). TT bilgisi kullandığınız donanımın özelliklerine göre değişebilir.
STAT	(<i>Status</i>) Sürecin bulunduğu duruma ilişkin bir kod. R: <i>Runnable</i> : Çalışabilir durumda, sırasını bekliyor S: <i>Sleeping</i> : Uyuyor Z: <i>Zombie</i> : Bu süreç ile ilgili tüm diğer süreçler bitmiş veya ölmüş; bununda bitmiş olması gerekirdi ama bir nedenle ölememiş . ps listesinde hala görünüyor olması zararsızdır.
TIME	Sürecin ne kadar zamandır çalıştığını gösterir
COMMAND	Süreci başlatan komut satırı

Süreçler hakkında daha detaylı bilgi isterseniz :

% ps -l	(process status -long list)
---------	-----------------------------

Sistemdeki tüm süreçler hakkında bilgi isterseniz :

% ps -ax	(process status -all, extended)
% ps -ef	

Sistemdeki tüm süreçler hakkında detaylı bilgi isterseniz :

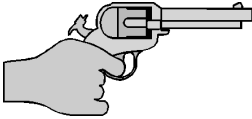
% ps -axl	(process status -all, extended, long)
% ps -efl	

ps komutu, sistem yöneticileri için (**sysad**, **sysadmin** : system administrator) için çok önemlidir. Sistemde neler olup bittiğini, kullanıcıların ne gibi programlar kullanmakta olduklarını, sisteme kullanıcıların nerelerden eriştiğini, hep bu komut yardımı ile gözlerler. Ayrıca, sistemin çalışmasında bir gariplik olduğu zaman hemen bu komutla bilgisayarda çalışmakta olan süreçlerin bir listesini alırlar.

ps komutu, zaman zaman normal kullanıcılar için de çok önemli olur. İşte size hemen bir senaryo...

Süreç Öldürme

(Killing Processes)



Diyelim ki başlattığınız bir iş kontrolden çıktı ve istediğiniz ya da beklediğiniz gibi davranmıyor. Doğal olarak bu işi hemen kesmek istiyorsunuz. İlk denemeniz gereken Ctrl-C tuşu. Olmazsa Ctrl-D tuşu... (Fazladan basacağınız Ctrl-D **logout** edilmenize neden olabilir.) Gene olmadı diyelim. **BİLGİSAYARI ELEKTRİK ANAHTARINDAN KAPATMAYI VEYA RESET DÜĐMESİNE BASMAYI AKLINIZDAN DAĐİ GEÇİRMEMELİSİNİZ!**

Böyle bir durumda, eğer yapabiliyorsanız, ekranınızdaki başka bir pencereden veya bir başka kullanıcı terminalinden :

- Uygun bir **ps** komutuyla (**ps** veya **ps -axl** veya **ps -efl**) çalışmakta olan süreçlerin bir listesini alın.
- Bu listeye bakarak, sorun çıkaran sürecin numarasını öğrenin ve

```
% kill nnn
```

(process kill)

(Burada **nnn**, öldürülmek istenen sürecin numarasıdır).

- Eğer sorun yaratan süreci bu komutla öldüremezseniz

```
% kill -9 nnn
```

komutunu deneyiniz. (-9 seçeneği '*koşulsuz öldürme*' isteğinizi belirtir.)

```
% ps
  PID TT  STAT   TIME COMMAND
 1234 p0  S      0:43  shelltool
 1266 p0  R      2:46  problemlı prog
 1456 co  R      0:04  ps
 1605 co  S      0:01  -bin/csh (csh)
% kill 1266
% kill -9 1266
```

almazsa

Süreci hala öldüremiyorsanız, kabuk programınızı öldürmeyi deneyiniz. Hala direniyorsa sistemin **USULÜNE UYGUN OLARAK** kapatılmasını sağlayınız. Eğer **root** yetkilerine sahip olabiliyorsanız, bu işi kendiniz de yapabilirsiniz; ancak sistemde başka kullanıcılar olabileceğini unutmayıp, bu kullanıcılara bir mesaj gönderip (**write** ve **wall** komutları), onlara makul bir süre tanıyıp; ancak ondan sonra **shutdown** komutunu kullanarak sistemi kapatınız. Sistemleri kapatma yöntemlerini daha sonraki bölümlerde anlatacağım.

Link Kavramı ve ln Komutu



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

Sanırım en iyisi bir örnekle anlatmak :



Farzedin ki bir UNIX sisteminin yöneticisisiniz. Sizden, bilgisayara **matlab** isimli yeni bir uygulama programı yüklemenizi istediler. Ancak, uygulama programının bir gereği olarak, program paketine ilişkin dosyaların **/usr/local** dizininin altında açılacak bir dizinde yer alması gerekiyor. Eh! Olabilir. Ancak, bir sorun var! **/usr** diskinde, yeni programa ilişkin dosyalar için yeterli boş yer yok; ve silebileceğiniz gereksiz dosyalar da yok!

Mistik **ln** kavramını kullanarak bu işi UNIX'in şanına yaraşır bir yöntemle çözebilirsiniz. Disklerin birinde; örneğin **/home** dizininin bulunduğu disk bölümünde (*partition*), yeni yükleyeceğiniz program için bir dizin yaratınız :

```
# mkdir /home/matlab
```

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

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

komutunu veriniz.

Böylece, gerçekte **/home** altında yer alan **matlab** dizini, aynı zamanda **/usr/local** altında da varmış gibi olacaktır. Bu dizini kullanırken isterseniz **/home/matlab**; isterseniz **/usr/local/matlab** dizin adreslerini kullanabilirsiniz. Bir başka deyişle, dosyalarının **/usr/local** altında bulunmasını isteyen matlab yazılımını kandırmış olursunuz.



link kavramının çok işe yarayabileceği, bir öncekine benzeyen bir senaryo daha anlatabilirim. Diyelim ki elinizde **mhsb1995** isimli bir dosya var ve muhasebe departmanının kullandığı muhasebe programı bu dosyayı mutlaka bu isimde görmek istiyor. Öte yandan yeni satın aldığınız bir mali analiz programı, aynı muhasebe verilerini **acct95** adıyla görmek istiyor.

Söz konusu dosyanın adı **mhsb1995** olduğu zaman muhasebe departmanının sorunu yok ama siz mali analiz programını çalıştıramıyorsunuz. Analiz çalışmaları için dosyanın adını değiştirseniz, siz çalışabiliyorsunuz ama bu sefer muhasebe departmanındaki program kullanılamıyor. Dosyanın adını **mhsb1995** olarak tutup, kendi analiz çalışmalarınız için **acct95** adlı bir kopyasını çıkardığınızda ve siz bu kopya üzerinde çalıştığınızda problem kısmen çözülüyor ama çok kullanıcı ortamında siz analizler üzerinde çalışırken öte taraftan muhasebe personeli yeni kayıtlar girip sizin analizlerinizin eskimiş kayıtlar üzerinde yapılmasına neden oluyorlar. İşte böyle bir durumda **link** kullanımı sizi kurtaracaktır.

```
# ln ./mhsb1995 ./acct95
```

Bu komutla **mhsb1995** dosyasını **acct95** isimli bir dosyaya bağladığınızda (aslında sadece tek bir asıl kopya var; o da **mhsb1995**. **acct95** isimli bir dosya aslında yok sadece diğer dosyanın bir başka adı.) Bu sayede **mhsb1995** dosyasında yapılan her değişiklik **acct95** diye tanınan dosyada da aynen gözlenebilecektir. İşin bir başka yaralı tarafı da; **acct95** isimli dosyanın diskte hiç yer kaplamayacak olmasıdır.



Bu örnekler arasında, dikkatinizi çekmiş olduğumu umduğum bir fark var. İlk örnekte (matlab), **ln** komutunda **-s** diye bir parametre kullandım; oysa ikinci muhasebe örneğinde kullanmadım!



- Eğer, **ln** komutuyla birbirlerine bağlanacak olan dosya sistemi elemanları birer dizinse; **-s** parametresini kullanmak zorundasınız.
- Eğer, **ln** komutuyla birbirlerine bağlanacak olan dosya sistemi elemanları birer dosyaysa ve farklı disk parçalarında (bir başka deyişle; farklı dosya sistemleri altında) yer alıyorsa, gene **-s** parametresini kullanmak zorundasınız.
- **ln** komutuyla, bir dizini ve bir dosyayı birbirlerine bağlayamazsınız. Bağlanacak olan elemanların ikisi de dizin; ya da ikisi de dosya olmalıdır.

Aynı dosya sisteminde yer alan ve birbirine bağlı olan dosyalardan birini silmeniz diğerini etkilemez. Asıl dosyayı silseniz bile, UNIX, bağlantıyı farkedip dosyayı diskten gerçekten silmeyecektir. UNIX, her dosya için bağlantıları sayar ve her silme işleminde bağlantı sayısını bir azaltır. Gerçek silme işi bu bağlantı sayısı sıfırlanınca yapılır.



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

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

```

/home/ayfer % cd /
/ % ls -l
total 3166

lrwxrwxrwx 1 root      7 Jan 12 12:09 bin -> /usr/bin
-r--r--r-- 1 root 110912 Jan 12 12:11 boot
drwxr-sr-x 2 bin      7680 Jan 12 12:23 dev
drwxr-sr-x 7 bin      1536 Jan 15 08:45 etc
drwxr-sr-x 4 root      512 Feb  1 11:56 export
drwxr-xr-x 5 root      512 Mar 23 09:03 home
-rwxr-xr-x 1 root 239783 Feb 09 13:34 kadb
lrwxrwxrwx 1 root      7 Mar 01 18:23 lib -> /usr/lib
drwxr-xr-x 2 root     8192 Jun 15 23:09 lost+found
drwxr-sr-x 2 bin      512 Mar 01 20:09 mnt
drwxr-sr-x 2 bin      512 Mar 09 08:59 sbin
lrwxrwxrwx 1 root     13 Jan 24 07:45 sys -> /usr/kvm/sys
drwxrwsrwt 2 bin      512 Feb 24 09:56 tmp
drwxr-xr-x 20 root     512 Nov 23 16:08 usr
drwxr-xr-x 11 root     512 Nov 23 16:11 var
-rwxr-xr-x 1 root 1101191 Jan 11 09:35 vmunix
/ %

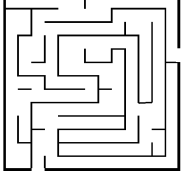
```

Bu örnek listeye göre, aslında **/bin** diye bir dizin bulunmamakta, bu isimde bir bağlantının **/usr/bin** dizinine yapılmış olduğu anlaşılmaktadır.

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

İpin ucunu kaçırmayacağınıza eminseniz, bağlantılara bağlantı yapabilirsiniz.

Önemli UNIX Komutları



Günümüzün tipik UNIX bilgisayarlarında, GigaByte düzeyinde diskler bulunmaktadır. Bu kadar büyük disklerde de doğal olarak çok sayıda dizin ve binlerce dosya yer almaktadır. Zaman zaman adının bir kısmını hatırlayabildiğiniz; bulunduğu diziniyse bir türlü hatırlayamadığınız dosyalar olacaktır. Tek tek bütün dizinlere girip **ls** komutuyla bu dosya ya da dosyaları aramak pek akıllıca bir yöntem değildir. Böyle bir durumda kullanacağınız komut **find** dir.

```
find baslama-dizini kriter[ler] [-exec komut ";"]
```

find komutuyla yapabileceğiniz aramalarda tek kriter dosya adı değildir. Bu komutla

- a) erişim yetkileri belirli bir kalıpta olan,
- b) belirli özelliklere sahip,
- c) belirli bir kullanıcıya ait,
- d) belirli bir boydan büyük ya da küçük,
- e) belirli bir tarihten veya saatten bu yana değişmemiş, erişilmemiş

dosyaları veya dizinleri bulabilirsiniz.

Üstelik verdiğiniz arama kriterlerine uyan dosyalar üzerinde uygulanacak UNIX komutlarını da **find** komutuna parametre olarak verebilirsiniz.

başlama-dizini

Arama işlemi, **find** komutunun bu ilk parametresinde belirtilen dizinden başlar ve varsa bu dizinin alt dizinleri de arama ağacına dahil edilir.



Eğer arama işleminin, bilgisayarınıza bağlı ve **mount** edilmiş tüm disklerinde yapılmasını istiyorsanız, bu ilk parametre olarak **/** sembolünü kullanınız.



Bilgisayarınızın CD-ROM sürücüsü varsa ve bu sürücüye bir CD takılıysa ve bu CD **mount** edilmiş durumdaysa ve arama, / dizinin hiyerarşisi boyunca yapılsa, CD-ROM sürücüsünü de kapsayacaktır. CD'lerin kapasitelerinin büyüklüğü ve erişim hızlarının düşüklüğünden dolayı bu arama uzun sürecektir. Aynı mantıkla, bilgisayar ağı üzerinden başka bilgisayarların diskleri de sizin dosya sisteminize **mount** edilmiş durumdaysa, o diskler de arama ağacına girecektir. Zaman kaybına yol açmamak için, gerekmedikçe aramayı / dizininden başlatmamanızı öneririm.

kriter[ler]

Aranan dosya ve dosyaların ortak özelliklerini tanımlayan kriterlerdir

Bir kaç örnek vermek gerekirse :

- name isim adı "isim" olan dosyalar
(farklı dizinlerde aynı isme sahip dosyalar olabilir)
- name "abc*" adı "abc" ile başlayan dosyalar
- name "a*data" adı "a" ile başlayan ve adının sonunda "data" olan dosyalar
- name "[a-k]95" adı a95, b95, ..., j95 veya k95 olan dosyalar



Dikkatinizi çektiyse, -name kriteri kullanıldığında, dosya adını verirken, dosya adını tam olarak yazıyorsak tırnak (") kullanmıyoruz; oysa * karakterini içeren bir kalıp kullanıyorsak (**wildcard**) bu kalıbı tırnak (") içinde yazıyoruz.

Bunun nedeni şu :

Bir komut verdiğinizde, bu komut önce kabuk programınız tarafından irdelenir. Bu irdeleme sırasında rastlanan * karakterleri dosya adı kalıpları olarak kabul edilip, bu kalıba uyan dosya isimleriyle değiştirilmeye çalışılır. Oysa, kalıplara uyan dosya isimlerinin kabuk programı tarafından değil, **find** programı tarafından bulunması gerekmektedir. Kabuk programlarının irdeleme sırasında karşılaştıkları * karakterlerine dokunmamaları için, kalıp tanımları tırnak içine alınır.

- user ayfer sahibinin adı **ayfer** olan dosyalar
- group yönetim sahibi **yönetim** grubuna dahil olan dosyalar
- perm 755 erişim yetki düzeyi **755** olan dosyalar

-newer dosya1	dosya1 isimli dosyadan daha sonraki bir saat ya da tarihte değişikliğe uğramış olan dosyalar
-size 10	diskte kapladığı alan 10 blok olan dosyalar (1 blok = 512 Byte)
-size +100	diskte kapladığı alan 100 bloktan büyük olan dosyalar (51 KByte'dan büyük dosyalar)
-size -45	diskte kapladığı alan 45 bloktan küçük olan dosyalar
-ctime 3	Tam 3 gün önce değişikliğe uğramış olan dosyalar
-ctime +8	8 günden daha uzun bir süre önce değişikliğe uğramış olan dosyalar
-ctime -8	8 günden daha kısa bir süre önce değişikliğe uğramış olan dosyalar
-mtime 3	Tam 3 gün önce değişikliğe uğramış olan dosyalar
-mtime +8	8 günden daha uzun bir süre önce değişikliğe uğramış olan dosyalar
-mtime -8	8 günden daha kısa bir süre önce değişikliğe uğramış olan dosyalar
-atime -3	3 günden daha kısa bir süre içinde bir şekilde erişilmiş olan dosyalar

-ctime ve **-mtime** parametrelerinin her ikisi de dosyanın değişikliğe uğramasıyla ilgili süreleri kontrol eder; ancak aralarında küçük bir fark vardır. **-mtime**, dosyanın içeriğinde bir değişiklik yapıp yapılmadığına; **-ctime** ise dosyanın içeriği yanısıra özelliklerinin de değişip değişmediğini kontrol eder. Örneğin, sahibi değişen bir dosya **-mtime** tarafından farkedilmezken **-ctime** tarafından dikkate alınır.

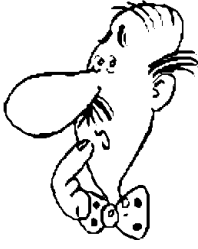
-atime 3	Tam 3 gün önce bir şekilde erişilmiş dosyalar
-atime +8	8 günden daha uzun bir süre önce erişilmiş olan dosyalar
-atime -8	8 günden daha kısa bir süre önce erişilmiş olan dosyalar
-type f	Basit birer 'dosya' olan dosyalar

-type d Dizinler

Bu arama kriterlerini bir arada kullanabilirsiniz; örneğin, sahibi **hakman** adlı kullanıcı olan ve son 40 gündür kullanılmamış dosyaları bulmak isterseniz, kullanmanız gereken **find** komutu

```
% find /home -user hakman -atime +40 -print
```

olmalıdır.



Komutun sonundaki **-print** parametresini kullanmayı unutursanız, **find** programının verdiği kriterlere uygun dosya bulup bulmadığını öğrenemezsiniz. Bulunan dosyaların isimlerinin listelenmesi için bu parametreyi kullanmak şarttır. İlk bakışta anlamsızmış gibi geldiğini biliyorum. Eğer bulunan dosyaların adını görmek istemiyorsanız, **find** komutunu neden kullanasınız ki? Bu sorunun yanıtı şöyle : **find** komutunu bir kabuk programı içinde çalıştırıyorsanız ve sizin için verdiği kriterlere uygun dosya bulunup bulunmadığını bilmek yetiyorsa (hangi dosyalar olduğunu görmeniz gerekmiyorsa) aramanın başarılı olup olmadığını belirten bir sistem değişkeninin değerine bakmanız yeterli olacaktır. (*condition code veya completion code*).

Şimdi, sık kullanılan **find** formları için bir kaç örnek vereyim :

```
% find /home/ayfer -name onemli.dosya -print
```

/home/ayfer dizininden başlayarak bu dizinde ve alt dizinlerinde **onemli.dosya** isimli dosyaları arar ve bulduklarının adını standart çıktıya (ekrana) listeler.

```
% find / -name core -exec /bin/rm {} ";"
```

/ dizininden başlayarak tüm disklerde **core** isimli dosyaları arar ve bulduklarını siler.

find komutunu **-exec** parametresiyle birlikte kullanırken sondaki ";" parametresini UNUTMAMALISINIZ. Bu ";" karakter dizisinin gerekliliği tamamen **find** programının yazılışından kaynaklanmaktadır.

Bu komut, sistem yönetiminden sorumlu olanların oldukça sık kullanacakları bir komuttur. UNIX, çeşitli programların kullanımı sırasında bir sistem problemi olduğunda "**core dumped**" mesajıyla birlikte, belleği **core** isimli bir dosyaya kopyalar. Bu **core** dosyaları, problemin nedenini bulmasına yardımcı olmak amacıyla yaratılır. Bu dosyaları irdeleyerek problemin nedenini bulmak pek kolay

olmadığından, genellikle bu dosyalar içeriklerine bakılmaksızın silinebilirler. Zaman içinde biriken **core** dosyaları diskte oldukça önemli yer harcadıklarından, rastladıkça bu dosyaları silmenizi öneririm.

```
% find /home -user hasan -exec /bin/rm {} ";"
```

/home dizininden başlayarak **hasan** isimli kullanıcıya ait dosyaları arar ve bulduklarını siler.

Sisteme erişim hakları iptal edilen kullanıcılara ait dosyaları tek harekette silmek için kullanılabilir.

```
% find /home -name "*.tmp" -exec /bin/rm {} ";"
```

/home dizininden başlayarak adı ***.tmp** kalıbına uyan dosyaları arar ve bulduklarını siler.

```
% find /home -type d -name [tmp, temp] -print
```

/home dizininden başlayarak adı **tmp** veya **temp** olan dizinleri bulur ve listeler.

find komutuyla birlikte kullanılan kriterleri çeşitli mantık operatörleriyle birleştirebilirsiniz. Bunlar

```
-a : "ve"  
-o : "veya"  
\! : "değil"
```

operatörleridir.

Örnekler :

```
% find /home -name "*.tmp" -a -size +100 -print
```

adı ***.tmp** kalıbına uyan **ve** büyüklüğü 100 bloktan fazla olan dosyaları bulur. (1 blok = 512 byte)

```
% find /home/ayfer \! -user ayfer -print
```

ayfer isimli kullanıcının home dizininde yer alan ama **ayfer**'e ait **olmayan** dosyaları bulur.

Bu örnekteki "**değil**" anlamında kullanılan **\!** operatöründeki **** işareti ardından gelen **!** işaretinin özel bir anlamı olduğunu ve kabuk programı (sh veya csh) tarafından yorumlanmaya çalışılmaması gerektiğini belirtmek için kullanılmaktadır.

Hatırlarsanız, daha önceki bölümlerden birinde, UNIX işletim sisteminde kendi komutlarınızı yaratabileceğinizden bahsetmiştim. Sanırım bu uygulamaya bir örnek vermek için uygun bir noktadayız.

find komutu oldukça yetenekli ve seçenekli bir komut; ama bunun karşılığında da yazması oldukça uzun. Dosyaları sadece adlarıyla arayan daha kısa bir UNIX komutu yaratmaya ne dersiniz?

Önce **vi** editörünü kullanarak home dizininizde **ff** isimli ve içinde aşağıdaki satırlar bulunan bir dosya yaratınız. (**% vi ~/ff**)

```
#!/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
```

ff program dosyasını oluşturan bu satırların anlamları üzerinde şimdilik durmayınız.

Daha sonra,

```
% chmod a+x ~/ff
```

komutuyla, bu dosyanın erişim yetki kalıbını, tüm kullanıcılar tarafından çalıştırılabilen bir komut dosyası olacak şekilde değiştiriniz.

Bu komut, aksi belirtilmedikçe, aramalara bulunduğunuz dizinden başlar. Eğer tek parametreyle çalıştırılırsa, bu parametreyi bir dosya adı olarak kabul edip, bulunduğunuz dizinde ve alt dizinlerinde bu dosyayı arayacaktır. Eğer iki parametreyle başlatılırsa, birinci parametre aramanın başlayacağı dizin, ikinci parametreyle aranacak dosyanın adı kabul edilecektir. Eğer dosya adı içinde * kullanmak istiyorsanız *'li ifadeyi çift tırnak içine almayı unutmayınız.

Örnekler :

```
% ff aranan.veri.dosyasi
% ff /home/ugur prog.c
% ff ~ file001.dat
% ff "*dat"
% ff /cdrom "openwin*"
```



Yeni yarattığınız **ff** komutunu verdiğinizde, komut programının bulunamadığına ilişkin bir mesaj alıyorsanız, **path** değişkeninizde **home** dizininiz olmayabilir. Çalıştırmak istediğiniz programı oluşturan dosyanın çalışma dizininizde bulunması yetmez. Bir programın çalıştırılabilmesi için

- i) ya yeri tam olarak komutta belirtilmelidir (**~/ff** gibi)
- ii) ya da program dosyasının bulunduğu dizin, **path** değişkeninde tanımlanmış olmalıdır.

path ile ilgili bir sorun olmamasına rağmen 'komut bulunamadı' (*ff : Command not found.*) mesajını alıyorsanız; **chmod** komutuyla, **ff** programının "**çalıştırılabilir**" (*executable*) bir dosya olduğunu belirtmeyi unutmuş olabilirsiniz.

Bir olasılık ta; **ff** komut dosyasını girerken bir hata yapmış olabileceğinizdir.

Arama - Tarama

find komutuyla; dosyaları, adları ve sahipleri gibi özelliklerine göre taramayı öğrendiniz. Peki.... Dosyaların içinde kayıtlı verilere göre aramaları nasıl yapacaksınız? Örneğin, içerdiği kayıtlar arasında **ayfer** sözcüğü geçen dosyaları bulmak istediğinizde hangi komutu kullanmalısınız?

```
% grep [-ilnc] patern dosya(lar)      general purpose regular
                                         expression search program
```

Hemen bir kaç örnek...

İçinde yaklaşık 20,000 satır bulunan **/etc/termcap** dosyasında (terminal karakteristikleri tanımlama dosyası) "wyse50 marka terminallerle ilgili bir tanım var mı?" diye merak ettiğinizde

```
% grep wyse50 /etc/termcap
```

komutunu kullanabilirsiniz. Eğer bu dosyanın içinde **wyse50** sözcüğü geçiyorsa, bu satırlar standart çıktı birimine (ekrana) listelenecektir.

wyse50 sözcüğünün büyük harflerle yazılmış olma olasılığı varsa

```
% grep -i wyse50 /etc/termcap
```

formunu denemelisiniz. (**-i** : *ignore case*; büyük-küçük harf ayrımı yapılmasını)

Bulunan satırların satır numaralarını da görmek isterseniz

```
% grep -ni wyse50 /etc/termcap
```

formunu kullanabilirsiniz. (**-n** : *numbered*)

Bulunduğunuz dizinde, adı **mektup** ile başlayan dosyalar arasında bir veya birkaç tanesinin içinde **ayfer** sözcüğünün bulunduğunu biliyorsunuz ama hangileri olduğunu hatırlayamıyorsunuz! İşte çözüm :

```
% grep ayfer mektup1 mektup2 mektup3 ...
veya
% grep ayfer mektup*
```

grep komutu, arama işini birden fazla dosya üzerinde yaptığı zaman, kullanıcıya kolaylık olması için; bulunan satırları ekrana listelerken, her satırın başına, satırın bulunduğu dosyanın adını ekler.

Eğer, bulunan satırlar için yalnızca dosya adlarını görmek istiyorsanız,

```
% grep -l ayfer mektup*
```

formunu kullanmalısınız.

Adı **mektup**'la başlayan dosyalarda **ayfer** sözcüğünün kaç defa geçtiğini öğrenmek isterseniz

```
% grep -c ayfer mektup*
```

komutunu kullanılabilirsiniz.

grep komutu (ve onun biraz geliştirilmişleri olan **egrep** ve **fgrep**), UNIX işletim sisteminin en çok kullanılan komutlarından. Bu komutun daha yararlı kullanımlarına ilişkin örneklere devam etmeden önce çok önemli bir UNIX kavramından daha söz etmek istiyorum: **PIPE**.

pipe kavramını anlatırken kullanacağım örnekler arasında **grep** komutuyla ilgili olanları dikkatle incellerseniz yukarıda verilen örneklerden daha yararlı kullanımlarını öğrenmiş olacaksınız.



UNIX kullanıcılarının günlük hayatta karşılaşacağı tipik bir sorun ve bu sorunun çözümünden söz etmek istiyorum : Sorun (ya da soru) şu :

home dizinin altında yer alan bir takım dizinlerde bir takım dosyaların içinde "piper" sözcüğü geçiyor. Bu dosyaların hangileri olduğunu

```
grep piper *
```

komutuyla bulabileceğimi biliyorum; ama **home** dizinin altında o kadar çok alt-dizin var ki! Her birine teker teker geçip aynı **grep** komutunu tekrarlamak istemiyorum. Bu arama işini hem **home** dizinimde, hem de onun alt dizinlerinde tek komutla yapabilir miyim?

Elbette yapabilirsiniz! UNIX'de, sadece standart UNIX komutları kullanarak, hiç program yazmadan, veri tabanı sistemi bile geliştirebilirsiniz!

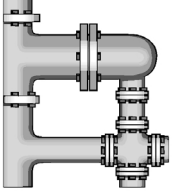
Bu küçük problemin çözümü şu iki komut :

```
cd /home/ayfer
grep piper `find . -print`
```

tırnak işaretlerine dikkat!
ASCII kodu desimal 96 olan
tırnak işaretidir. Burada '
veya " kullanamazsınız.

Komutun çalışma sistemi aslında basit. Kabuk programınız, komut satırında ` işaretleri arasında bir başka komut görünce önce onu çalıştıracaktır (**find . -print** komutu). Bu programın standart çıktıya gönderdiği listeyi de **grep** komutunun sonuna ekleyecek; **grep** komutunu ondan sonra çalıştıracaktır. Bir başka deyişle, **grep** komutuna, içinde **piper** sözcüğü aranacak dosyaların listesini klavyeden yazmak yerine, bu işi **find** komutuna yaptırmış olacaksınız. Zarif, değil mi?

UNIX PIPE Kavramı



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

Piping işlemiyse, gene bir çeşit yönlendirmedir; ancak şu farkla ki, bir programın standart çıktısı, bir başka programa standart girdi olarak yönlendirilir.

Pipe kurmak için, aynı komut satırında en az iki program birden başlatmalı ve bu iki programa ilişkin komutların arasına

|

karakterini yerleştirmeniz gerekir.

Şimdi **grep** komutu ve **pipe** kavramının birlikte kullanımına bir kaç örnek vereyim :

```
% grep ayfer mektup* | more
```

Bu komut, daha doğrusu komut ikilisinin anlamı şu :

grep ve **more** programlarını aynı anda başlat. Adı '**mektup**'la başlayan dosyalar içinde **ayfer** sözcüğünü ara, bulduğun satırları **more** programına gönder, **more** programı bu satırları alsın ve kendi görev tanımını doğrultusunda işlesin. (Yani sayfa sayfa listelesin).

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

ps ve **grep** programlarını aynı anda başlat.. **ps** programının oldukça uzun olabilecek çıktısını **grep** programına girdi olarak gönder.

grep kendisine gönderilen satırlar arasında, içinde **in.named** sözcüğü geçenleri bulsun ve sadece ilgilendiğimiz bu satırları listelesin. Böylece **grep** programı bir filtre gibi kullanılmış olacaktır.

Şimdide **sıkı** bir *pipe* örneği...

□ tuşunun solundaki
tırnak işareti...

```
% echo Sistemde `who | wc -l` kullanıcılarda var
```

Bu komut satırında bir kaç kademeli bir işlem istenmektedir. İlk olarak **who** programı çalıştırılacaktır. Aynı anda **wc** programı da çalıştırılacak ve **who** programının çıktısı standart girişindeki satır, kelime ve karakterleri sayan **wc** (-l seçeneği yalnızca satırların sayılmasını sağlıyor) programına gönderilecektir. **wc** programının çıktısıysa (**who** komutunun listelediği satırların sayısı) tırnaklar arasına yerleştirilerek elde edilen; örneğin üç kullanıcı varsa, '**Sistemde 3 kullanıcı var**' dizisi de **echo** programına girdi olarak transfer edilir. **echo** programıysa parametrelerini aynen ekrana gönderir. Bu örnekteki komutu, **home** dizinizdeki **.login** veya **.cshrc** dosyasına eklerseniz, sisteme her **login** ettiğinizde, sistemde siz dahil, kaç kişinin çalıştığını öğrenmiş olursunuz.

Yazıcı Kullanımı

MS-DOS işletim sistemi ile çalışan kişisel bilgisayarlarda yazıcı kullanımı oldukça kolaydır. Bilgisayarı yalnızca siz kullandığınız için, diğer tüm kaynaklar gibi yazıcı da sadece sizin kullanımınıza tahsis edilmiş durumdadır. Canınız istediği zaman programınıza 'yaz' komutunu verir ve yazıcının başına geçip, çıktıların kağıda aktarılmasını beklersiniz.

UNIX dünyasında durum farklı... Kullanıcılar, bilgisayarın tüm kaynakları gibi yazıcısını da başkalarıyla paylaşmak zorundalar. Yazıcıya gönderilen her döküm, MS-DOS'da olduğu gibi anında basılmaya başlamayabilir; çünkü o anda yazıcı bir başka kullanıcının bir çıktısını döküyor olabilir. Kağıda bir kaç satır bir kullanıcıdan, bir kaç satır da başka kullanıcıdan döküm yapmak pek sağlıklı olmayacağı için, tüm çok kullanıcıli işletim sistemlerinde olduğu gibi, UNIX'te de; yazıcı dökümlerinin sıraya konmasını sağlayan **SPOOLING (Shared Peripheral Operation Online)** işlemi uygulanmaktadır.

UNIX'de, kullanıcı programlarından gelen '**yazıcıya yolla**' emirleri sanki yerine getirilmişcesine olumlu karşılır; ancak yazıcıya gönderilmesi istenen bilgiler diskte önceden belirlenmiş bir alana kaydedilir (**spool area**). Zaman içinde kullanıcılardan gelen döküm istekleri UNIX **spooler** programı tarafından sıraya konur ve yazıcı boş kaldığında, diskte saklanan dökümler kağıda aktarılmak üzere sırayla yazıcıya veya yazıcılara gönderilir.

Bir başka deyişle, uygulamanız, kağıda bir döküm almanızı gerektiriyorsa ve siz bu doğrultuda yazdırma komutu verdiyseniz; isteğiniz hemen yerine getirilemeyebilir (yazıcı meşgul olabilir veya hazır olmayabilir). Ancak, bu durum size yansıtılmaz ve kağıda dökülmesini istediğiniz her şey, **spooler** tarafında diske kaydedilerek sıraya sokulur ve ilk fırsatta yazıcıya gönderilir.

UNIX altında çalışan bilgisayarlar genellikle büyükçe sistemler olduğundan, birden fazla yazıcıya sahip olabilirler. Özellikle bilgisayar ağlarında bu durumla daha da sık karşılaşılır. Bir döküm almak istediğinizde, yazıcı seçme şansınızın da olabileceğini unutmayınız.

BSD ve SVR4 UNIX türevlerinde yazıcı kullanma komutları oldukça farklı olduğundan, bu iki tip UNIX için ayrı ayrı bölümler hazırladım. Genel kültür açısından her iki bölümü de okumanızı öneririm.

BSD UNIX'de Yazıcı Kullanımı



Herhangi bir dosyayı yazıcıya göndermek istediğinizde kullanabileceğiniz en basit komut formu şudur :

```
% lpr dosya_adi (line printer)
```

Bu komutu verdiğinizde, **dosya_adi** isimli dosya, adı **lp** olan yazıcının (veya **PRINTER** isimli kabuk değişkeninde belirtilmiş olan isme sahip yazıcının) sırasına sokulur. Sisteminizde adı **lp** olan bir yazıcı bulunmasını sağlamak, sistem yöneticisinin görevidir.

Eğer, dosyanızı, özel bir yazıcıya göndermeniz söz konusuysa kullanacağınız komut

```
% lpr -Pyazici_adi dosya_adi
```

Bu komutta **P** harfinin büyük **P** olduğuna ve yazıcı adının bu **P** harfine bitişik olarak yazıldığına dikkatinizi çekerim. (Bazı UNIX'ler **P** harfiyle yazıcı adı arasında boşluk kullanılmasına izin verir.)

```
% lpq [-Pyazici_adi] (line printer queue)
```

Yazıcı için sıra bekleyen işler hakkında bilgi verir. Sıra bekleyen her dökümün bir tanıtma numarası vardır.

```
% lprm [ nnn [mmm ...] ] (line printer remove)
```

Sıra bekleyen dökümler arasında tanıtma numarası **nnn** (ve **mmm** vs) olan işleri iptal eder. **nnn** verilmezse, komutu veren kullanıcıya ait olan ve o sırada dökülmekte olan ya da sıradaki ilk işi iptal edilir.

```
% lprm kullanıcı_adi (line printer remove)
```

Sıra bekleyen işler arasında sahibi **kullanici_adi** olan dökümleri iptal eder. **nnn** (ve **mmm** vs)

```
% lpstat [-Pyazici_adi] (line printer status)
```

Yazıcının durumunu gösterir. (Hazır olup olmadığını vs.)

```
% lpr -#n -Plazer dosya_adi
```

dosya_adi isimli dosyanın, **lazer** isimli yazıcıdan **n** kopyasının basılmasını sağlar.

```
% lpr -m onemli
```

onemli isimli dosyanın, basılmak üzere **lp** isimli yazıcıya gönderilmesini ve basım tamamlandığında, komutu veren kullanıcıya bir mesaj (*mail*) gönderilmesini sağlar.

```
% lpr -r onemsiz
```

onemsiz isimli dosyanın, basılmak üzere **lp** isimli yazıcıya gönderilmesini ve dosyanın, ilgili yazıcının sırasına alınmasından hemen sonra diskten silinmesini sağlar.

```
% lpr dosya1 dosya2 ...
```

Birden fazla dosyanın tek komutla yazıcı sırasına gönderilmesini sağlar.

```
% sort < sirasiz | lpr
```

lpr komutu ve **pipe** kavramının birlikte kullanılmasına bir örnek... Bu örnekte, **sirasiz** isimli dosya **sort** programıyla sıraya dizilmekte ve sıralanmış hali doğrudan yazıcıya gönderilmektedir.

Örneklerini verdiğim çeşitli **lpr** seçeneklerini birleştirebileceğinizi ayrıca belirtmem gerek yok. Örneğin;

```
% lpr -rmPepson onemsiz
```

Kullanıcısı olduğunuz bilgisayar sistemine bağlı olan yazıcıların özelliklerini ve isimlerini sistem yöneticisinden öğrenebilirsiniz.

SVR4 UNIX'de Yazıcı Kullanımı



Herhangi bir dosyayı yazıcıya göndermek istediğinizde kullanabileceğiniz en basit komut formu şudur :

```
% lp dosya_adi (line printer)
```

Bu komutu verdiğinizde, **dosya_adi** isimli dosya, adı **lp** olan yazıcının (veya **PRINTER** isimli kabuk değişkeninde belirtilmiş olan isme sahip yazıcının) sırasına sokulur. Sisteminizde adı **lp** olan bir yazıcı bulunmasını sağlamak, sistem yöneticisinin görevidir.

Eğer, dosyanızı, özel bir yazıcıya göndermeniz söz konusuysa kullanacağınız komut

```
% lp -dyazici_adi dosya_adi
```

Bu komutta **d** harfinin küçük d olduğuna ve yazıcı adının bu **d** harfine bitişik olarak yazıldığına dikkatinizi çekerim.

```
% lpstat [-a] line printer status
```

Yazıcının durumunu gösterir. (Hazır olup olmadığını vs.) **-a** seçeneği tüm yazıcıların durumunu gösterir. Durum raporlarında, yazıcılar için sıra bekleyen işler ve tanıtım numaraları da listelenir.

```
% cancel nnn [mmm ...]
```

Sıra bekleyen dökümler arasında tanıtma numarası **nnn** (ve **mmm** vs) olan işleri iptal eder.

```
% cancel -u ugur
```

Sıra bekleyen dökümler arasında **ugur** isimli kullanıcıya ait olan döküm işlerini iptal eder.

```
% lp -nk -dlazer dosya_adi
```

dosya_adi isimli dosyanın, **lazer** isimli yazıcıdan **k** kopyasının basılmasını sağlar.

```
% lp -m onemli
```

onemli isimli dosyanın, basılmak üzere **lp** isimli yazıcıya gönderilmesini ve basım tamamlandığında, komutu veren kullanıcıya bir mesaj (*mail*) gönderilmesini sağlar.

```
% lp dosya1 dosya2 ...
```

Birden fazla dosyanın tek komutla yazıcı sırasına gönderilmesini sağlar.

```
% sort < sirasiz | lp
```

lp komutu ve **pipe** kavramının birlikte kullanımına bir örnek... Bu örnekte, **sirasiz** isimli dosya **sort** programıyla sıraya dizilmekte ve sıralanmış hali doğrudan yazıcıya gönderilmektedir.

Örneklerini verdiğim çeşitli **lp** seçeneklerini birleştirebileceğinizi ayrıca belirtmem gerek yok. Örneğin;

```
% lp -mdepson onemsiz
```

Kullanıcısı olduğunuz bilgisayar sistemine bağlı olan yazıcıların özelliklerini ve isimlerini sistem yöneticisinden öğrenebilirsiniz.



SVR4 UNIX'lerde yazıcı yönetimi ile ilgili olan bir kaç komut daha vardır. Bu komutların görev ve yetenekleri kitabın sınırlarını çok aştığı için; sadece meraklı kullanıcılar için bu komutların isimlerini verip geçeceğim. Bu komutlar hakkında daha fazla bilgi almak için **man** komutunu kullanabilir veya UNIX dökümantas-yonuna başvurabilirsiniz. Yazıcı yönetimine ilişkin diğer SVR4 UNIX komutları :

```
accept, lpadmin, disable, enable,  
lpmove, pr, reject, lpsched
```


Kabuklar - C Shell ve Shell

Komut Satırının Yorumlanması ve Parametreler

UNIX işletim sistemi, kullanıcıların verdikleri komutları çözümlemek ve bu komutları yerine getirecek programları başlatmak için kabuk (*shell*) programlarını kullanır. Bir başka deyişle, kabuk programları, kullanıcılarla bilgisayar arasındaki yazılım arabirimidir. Aslında, bu tip komut yorumlayıcıları (*command interpreter*), tüm işletim sistemlerinde kullanılmaktadır; örneğin MS-DOS işletim sisteminde bu görevi COMMAND.COM üstlenmiş durumdadır.

UNIX işletim sisteminde, kullanıcıların birden fazla kabuk programı arasından seçim yapma ve beğendikleri komut yorumlayıcısını kullanma hakları vardır. Hatta, aynı anda birden fazla kabuk programı bile kullanılabilirler. Daha fazla detaya girmeden, genel olarak bir kabuk programının ne işler yaptığını bir örnekle açıklamaya çalışacağım. Bu örneğimizle ilgili bir kaç tane de varsayımımız olacak; şöyle ki :

- Kullanıcı C-Shell kabuk programını kullanıyor olsun,
- Kullanıcının adı **ayfer** ve komutu verdiği anda kendi çalışma dizini **/home/ayfer** olsun,

Başarılı bir **login**'den sonra, UNIX, komut beklediğini,

```
abc:/home/ayfer % _ veya sadece % _
```

hazır işaretiyle (*prompt*) belli edecektir. (Eğer C-shell yerine **sh** kabuk programı kullanılıyor olsaydı, % işareti yerine \$ işareti görünüyordu).

Kullanıcı klavyesinden; örneğin;

```
cp eski-dosya yeni-dosya
```

komutunu verdiğinde, kabuk programı, **cp** harflerini kullanıcının çalıştırmak istediği programın adı olarak; **eski-dosya** ve **yeni-dosya** kelimeleriniyse bu **cp** programının iki parametresi olarak kabul edecektir.

cp	eski-dosya	yeni-dosya
(Komut)	(1. parametre)	(2. parametre)

Bir sonraki iş, kullanıcının çalıştırmak istediği bu **cp** programının saklandığı disk dosyasını bulmak olacaktır. Bu arama işinin temelinde, kullanıcımız için

tanımlanmış olan **PATH** ve/veya **path** kabuk değişkeninin o andaki değeri yatmaktadır. Bu değişkenlerin değerleri

```
PATH = /bin:/usr/bin:/usr/local/bin:~ayfer/bin:.  
veya  
path= ( /bin /usr/bin /usr/local/bin ~ayfer/bin .)
```

benzeri bir karakter dizisi olacaktır ve bu değerler, kullanıcının **home** dizininde yer alan **.cshrc** ve/veya **.login** dosyalarında tanımlanmış olmalıdır. Şimdilik, bu dosyaların, sizin için, sistem yönetici tarafından hazırlanmış olduğunu kabul edebilirsiniz.

csh programı, ":" işaretleriyle (ya da boşluk karakterleriyle) birbirlerinden ayrılmış olan dizinlerde; **cp** isimli bir dosya arayacaktır. Arama, dizin isimlerinin verilmiş sırasına göre yapılacaktır. Örneğimize göre, **cs**h programı, **cp** isimli dosyayı önce **/bin** dizininde; orada bulamazsa **/usr/bin** dizininde; orada da bulamazsa **/usr/local/bin**; olmazsa **ayfer** adlı kullanıcının **home** dizininin altındaki **bin** dizininde (**~ayfer/bin**); o da olmazsa o andaki çalışma dizininde (**.**) arayacaktır. Söz konusu dosyayı bu dizinlerden hiç birinde bulamazsa

cp : Command not found.

diye, komutu tanıyamadığına ilişkin bir hata mesajı vererek yeniden komut bekleme durumuna dönecektir.

Eğer, **cp** program dosyası, bu dizinlerden birinde bulunursa, bu dosyanın erişim yetkileri kontrol edilir; **ayfer**'in bu programı çalıştırmaya yetkisi varsa (**execute** yetkisi) **cp** programı kabuk tarafından belleğe yüklenir ve çalıştırılır. Komut satırında verilen parametrelerse, gerekirse çözümlenip, **cp** programına aktarılır.

Artık kontrol, **cp** programına geçmiştir. Bu programın mantığına göre son parametre, kopyalananın yapılacağı dosya ya da dizin adını, önceki parametrelerse buraya kopyalanacak dosyaların isimleri olmalıdır. Bir başka deyişle, **cp** komutunun en az iki parametresi bulunmalıdır. kabuk programı bu detayları bilemeyeceği için, bu tip mantık kontrolleri komut programı tarafından yapılmalıdır. Parametrelerin doğru sırada ve sayıda verilip verilmediğini her program kendisi kontrol eder ve gerekirse uygun hata veya uyarı mesajları üreterek, kullanıcıyı uyarır.

Şimdi ortalığı biraz karıştıralım....



Kullanıcımız

```
cp * /disk2/home2/ayfer
```

komutunu vermiş olsun. Bu komutla kullanıcının yapmak istediği iş, çalışma dizinindeki tüm dosyaları (*****), **/disk2/home2/ayfer** dizinine kopyalamak... Bu komutu gören **cs**h, komut adı olan **cp** sözcüğünü bulduktan sonra, bu komutun parametrelerini bulup çıkarmaya çalışacaktır. Komut satırını tararken

(*parsing*) * karakterine rastlayınca, **cs**h, "tüm dosyalar" anlamına gelen * yerine, çalışma dizininde yer alan dosyaların isimlerini yanyana gelecek şekilde yerleştirecektir.

Yani, komut satırı

```
cp abc dosya1 dosya2 xyz x123 muhasebe.dat /disk2/home/ayfer
```

şekline dönüştürülecektir (çalışma dizininde sadece **abc**, **dosya1**, **dosya2**, **xyz**, **x123** ve **muhasebe.dat** dosyalarının yer aldığı varsayımıyla). Bu dönüşümü ekranda gözleyemezsiniz; ancak bu tip dönüşümlerin olduğunu bilmeniz ve komutları verirken bu dönüşümleri dikkate almanız çok önemlidir.

Bazı komutların doğru çalışması için, komut satırlarının, kabuk programları tarafından dönüştürülmeden komut programlarına aktarılması gerekmektedir. Bu gerekliliği açıklayan en iyi örnek **find** komutudur.



Hatırlarsanız, **find** komutuna ilişkin verdiğim örneklerden biri, adı ***.tmp** kalıbına uyan ve büyüklüğü 100 bloktan fazla (51200 byte'dan fazla) olan dosyaları bulup listelemeye yönelikti.

```
find /home -name "*.tmp" -a -size +100 -print
```



Bu örnekte ***.tmp** yazarken kullanılan " işaretleri **çok çok önemlidir**. Kabuk programı, tırnak içinde yer alan komut bölümlerini çözümlemeye çalışmayacaktır. Komut satırında tırnak içinde yer alan bölümler, hiç bir değişikliğe uğramadan, ilgili programa parametre olarak iletilecektir. Şimdi, yukarıdaki **find** örneğindeki komutu tırnak işaretlerini kullanmadan yazdığımızı farzedelim....

```
find /home -name *.tmp -a -size +100 -print
```

hatalı

Bu komutu gören kabuk programı, **find** sözcüğünü program adı olarak değerlendirip, bu programın parametrelerini saptamak amacıyla satırı taramaya devam edecektir. **/home** birinci; **-name** ise ikinci parametre olarak çözümlenecektir. Buraya kadar sorun yok.... Ancak ***.tmp** kalıbına rastlandığında, çalışma dizininde yer alan ve adı bu kalıba uyan dosyaların isimleri komut satırına üçüncü, dördüncü, beşinci vs parametre olarak yerleştirilecektir (tabii çalışma dizininde adı bu kalıba uyan dosyalar varsa). Diyelimki, bu komutu verdiğimizde, çalışma dizinimizde şu dosyalar bulunmaktaydı :

```
a          dosya1      dosya2
a.tmp      dosya1.tmp  dosya3
```

kabuk programı tarafından çözümlenen ve dönüştürülen komut satırı

```
find /home -name a.tmp dosya1.tmp -a -size +100 -print
```

olacaktır.

BU ŞEKİLDE ÇÖZÜMLENMİŞ KOMUT BİRKAÇ NEDENLE HATALIDIR.

Birincisi; **find** komutunun mantığına göre arama sadece adı **a.tmp** olan dosyalar için yapılacaktır; oysa biz adı ***.tmp** kalıbına uyan tüm dosyaları aramak istiyoruz.

İkincisi; dosya1.tmp parametresi tanımlayıcı işaretli (-name, -size gibi) kalmıştır. (Nitekim, komutu verdiğinizde **find : missing conjunction** mesajı alırsınız).

Bu hataların olmaması için, kabuk programının, komut satırımızla oynamamasını ve ***.tmp** parametresini, **find** programına AYKEN göndermesini sağlamamız gerekmektedir. İşte, " tırnak karakterleri burada işe yaramaktadır.

KABUK PROGRAMLARI, " TIRNAK KARAKTERLERİ ARASINDA YER ALAN KOMUT PARÇALARINI ÇÖZÜMLEMEYE ÇALIŞMAZ VE PROGRAMA OLDUĞU GİBİ İLETİR:

Komutumuzu

```
find /home -name "*.tmp" -a -size +100 -print
```

doğru

olarak verince, **find** komutunun birinci parametresi **/home**, ikinci parametresi **-name**, üçüncüsü ***.tmp**, dördüncüsü **-a**, beşincisi **-size**, vs. olarak kabul edilecek ve **find** programı bu parametre yapısıyla çalıştırılacaktır. Yani, ***.tmp** kalıbı kabuk tarafından değil, **find** programı tarafından yorumlanacak ve komut istediğimiz şekilde çalışacaktır.



Eğer, kabuk programının irdelemeden komuta aktarmasını istediğiniz özel karakter tek bir karakterden oluşuyorsa, o karakteri tırnak içine almak yerine, önüne bir **** (*back slash*) yerleştirebilirsiniz. Bir başka deyişle

***.tmp** ile ***.tmp** ve
\! ile **\"!** eşdeğerdir.



Bu arada, kabuk tarafından çalıştırılan programların **sıfırıncı parametrelerinin** de bulunduğunu söylemeden geçemeyeceğim. Bir program çalıştırıldığında, sıfırıncı parametresi, programın kendi adıdır. Böylece, her program, hangi isimle kullanıldığını bilebilmektedir. Bu özelliğe tipik örnek **compress** ve **uncompress** komutlarıdır. Bu iki komut aslında tek bir program dosyasıdır. **compress** isimli dosya gerçekten bu isimle diskte yer alırken, **uncompress** sadece bu dosyaya bir bağlantıdır (*link*).

```
% which compress
/usr/ucb/compress

% ls -lF /usr/ucb/compress /usr/ucb/uncompress
-rwxr-xr-x 1 root 23783 ... compress*
```

```
lrwxr-xr-x 1 root 23783 ... uncompress --> ./compress
```

which komutu da nereden çıktı diyorsunuz... Çok önemli bir komut değil... Parametresi olarak belirtilen komut verilmiş olsaydı, hangi dosyanın çalıştırılacağını bildirir. Bir diğer deyişle, parametresi olan komutu **PATH** ve/veya **path** değişkenlerine göre disk(ler)de arar ve ilk bulduğunun (bulursa) yerini bildirir.

compress program dosyasını **compress** adıyla kullanırsanız, dosya sıkıştırma işlemi; **uncompress** adıyla çalıştırırsanız, daha önce sıkıştırılmış olan bir dosyayı açma işlemi yapılmasını sağlarsınız.

Kabuk Değişkenleri

Kullandığınız kabuk programı içinde çeşitli değişkenler tanımlamanız mümkündür; hatta bazı standart değişkenler zaten tanımlıdır. Kabuk değişkenleri arasında en önemlileri

Bourne Shell (sh)	C Shell (csh)	Görevi
PATH	path	Bir komut verildiğinde, komut programını oluşturan dosyanın aranacağı dizinler listesini belirleyen değişken.
HOME	HOME	Kullanıcının home dizinin adını içeren değişken. login ettiğinizde kabuk programı tarafından otomatik olarak yaratılır.
MAIL	ma i l	Size elektronik posta (e-mail) gelip gelmediğini anlamak için kontrol edilecek dosyaların listesi.
PS1 Tanımlanmazsa \$ kabul edilir	prompt Tanımlanmazsa % kabul edilir	Sistem hazır işaretini tanımlayan değişken.

TERM	TERM	<p>Kullandığınız terminalin tipini belirleyen değişkendir. Eğer vi editörünü kullanmak istediğinizde ekranınıza garip işaretler yanı sıra satırlar da garip bir düzensizlik içinde çıkıyorsa, büyük olasılıkla TERM değişkeniniz hatalı bir değere sahiptir ya da hiç tanımlı değildir.</p> <p>Bu değişkenin kullanımı, /etc/termcap dosyasındaki binlerce değişik terminal tipinin tanımlarıyla yakından ilgilidir. TERM değişkeninize vermeniz gereken değer için sistem yöneticinize danışınız.</p>
	history	<p>Sadece cs kabuğunda anlamlıdır. Bu değişkenin değeri, klavyeden gireceğiniz komutlardan son kaç tanesinin saklanacağını ve ! ile birlikte tekrar kullanılabileceğini belirtir. Tipik değeri 30-50 arasındadır.</p>

Örnekler	
Bourne Shell (sh)	C Shell (csh)
PATH=/bin:/usr/local/bin:.	set path=(/bin /usr/local/bin .) veya setenv PATH /bin:/usr/local/bin:.
MAIL=/usr/mail/ayfer	setenv mail /usr/mail/ayfer
PS1="ayfer@abc \$ "	set prompt="ayfer@abc % "
TERM=vt100	setenv TERM vt100
	setenv history 100



Bazı UNIX'lerdeki **sh** uyarlamalarında, bir kabuk değişkenine değer verdikten sonra değişkeni **export** komutu ile geçerli kılmanız gerekebilir. Şöyle ki :

```
PATH=/bin:/usr/local/bin:.    komutundan hemen sonra
export PATH                  komutunu vermeniz gerekebilir.
```

Herhangi bir anda, tanımlı olan kabuk değişkenlerini ve/veya değerlerinin ne olduğunu merak ederseniz

% set	<i>sh ve csh için</i>
% env	<i>sh için</i>
% setenv	<i>csh için</i>

komutlarını kullanabilirsiniz

Kabuk değişkenleri, standart isimli bir takım değişkenlerle sınırlı değildir. Kullandığınız uygulama programları, çalışma ortamını tanımlamak için bir takım değişkenlerin tanımlanmasını ve özel değerler verilmesini gerektirebilir. Örneğin X Windows uygulamaları, kullanılacak ekranın bağlı bulunduğu bilgisayarın adının **DISPLAY** isimli bir değişkende tanıtılmasını gerektirecektir.

C Shell'e Özgü Özellikler

UNIX dünyasında kabuk programı olarak **csh** kullanımı gittikçe yaygınlaşmaktadır. Bu nedenle kitabın bundan sonraki kısımlarında kullanıcıların kabuk programı olarak C Shell kullandıklarını varsayacağım. Eğer bu kitapta bundan sonra anlatılacak konulardaki örnekleri denemek istiyorsanız, **csh** kabuk programını kullanıyor olmalısınız. Hangi kabuk programını kullandığınızı bilmiyorsanız, sisteme login ettiğinizde sizin için başlatılacak olan kabuk programınızı değiştirmeniz gerekiyorsa sistem yöneticinizden yardım isteyiniz. Kullandığınız kabuk **tcsh** ise bir değişiklik yapmanız gerekmeyecektir.

csh'in bazı önemli özelliklerini örneklerle açıklamak istiyorum :

Hatalı Komutları Düzeltme

Diyelimki uzun bir UNIX komutunu yanlış yazdınız....

```
cp /home/hakman/.Xdesksetdefault /home/ayfer
```

(.Xdesksetdefault olmalıydı...)

ve doğal olarak **.Xsetdefault** diye bir dosya bulunamadığına dair bir hata mesajı aldınız. Eğer **csh** kullanıyorsanız, bu karışık satırı baştan bir kez daha yazmak yerine

```
% ^fual^faul^
```

yazıp ENTER tuşuna basmanız yeterli olacaktır. ("Bir önceki komuttaki **fual** karakter dizisini **faul** karakter dizisi ile değiştir ve komutu tekrarla" anlamında...)

Son Komutu Tekrarlama

Diyelimki MS-DOS işletim sisteminden alışkanlıkla

```
% cp /home/hakman/.Xdesksetdefaults
```

yazdınız. UNIX kurallarına göre kopyalamanın nereye yapılacağını da belirtmiş olmanız gerekirdi. **cs**h kullanıyorsanız, böyle bir durumda, tüm komutu tekrarlamak yerine

```
% !! /home/ayfer
```

yazmanız yeterli olacaktır. Böylece; bir önceki komutunuz aynen tekrarlanacaktır; ancak sonuna **/home/ayfer** eklenmiş olarak...

Eski Bir Komutu Tekrarlama

Eğer **history** isimli kabuk değişkeniniz tanımlıysa, bu değişkenin değeri kadar sayıda UNIX komutu kabuk tarafından bir ara bellekte saklanacaktır. Her hangi bir anda, eski komutlarınızı

```
% history
```

komutuyla listeyebilirsiniz.

```
% history
.....
7 23:12 ls -l
8 23:13 cat /etc/printcap
9 23:13 cp /usr/bin/.... ..
```

Bu eski komutlardan birini tekrar çalıştırmak isterseniz, bir ünlem işaretinin ardından o komutun listedeki sıra numarasını girmeniz yeterli olacaktır.

```
% !8
```

8 numaralı komutu tekrarlama

İlk Birkaç Harfini Hatırladığınız Eski Bir Komutu Tekrarlama

Her hangi bir anda, ilk harfini (ya da birkaç harfini) hatırladığınız eski bir komutu tekrarlamak isterseniz

% !c	<i>c harfiyle başlayan son komutu tekrarla</i>
% !ca	<i>ca harfleriyle başlayan son komutu tekrarla</i>

Kendi Gereksinimlerinize Göre Özel Komut Yaratma

UNIX kullanıcılarının çok sık tekrarladıkları bazı uzun komutları, daha kısa ve kolay yazılan komutlarla değiştirmeleri mümkündür. Bu iş **csh**'in **alias** komutu ile yapılır.

Örneğin, **history** komutunu her seferinde uzun uzun yazmaktansa,

% alias h history	<i>artık history yerine h kullanabilirsiniz</i>
-------------------	---

ls listelerinde isimlerin çalıştırılabilir program olup olmadığını * işaretiyle, dizinlerinse / işaretiyle belirlenmesi için kullanılan **-F** seçeneğinin standart hale getirilmesi için :

% alias ls "ls -F"	<i>yeni tanımda birden fazla sözcük olduğu için tırnak kullanmak gerekir</i>
--------------------	--

Eğer **more** komutunu sık sık hatalı yazıyorsanız :

% alias mroe more

MS-DOS alışkanlıklarınızdan vaz geçemiyorsanız :

% alias dir "ls -F"
% alias copy "cp -i"
% alias del "rm -i"
% alias ren mv
% alias edit vi

Herhangi bir anda geçerli olan **alias**'ları listelemek için

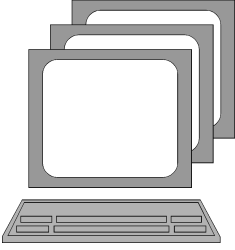
```
% alias
```

Hazır işaretinizin her zaman çalışma dizininizi göstermesi için

```
% alias cd 'cd \!*;set prompt="\`hostname\`:$cwd> "'
```

biraz çetrefilli ama çalışır... Bana güvenin..

Programları Arka Planda Çalıştırma



Diyelim ki, çok büyük bir disk dosyasındaki (söz gelimi 42 Mbyte) müşteri kayıtlarını alfabetik sıraya dizmek istiyorsunuz. Bu iş için kullandığınız bilgisayar sisteminde yarım saat süreceğini varsayalım. Eğer tek iş düzeninde çalışan bir işletim sistemi kullanıyor olsaydınız (MS-DOS gibi), sıralama komutunu verdikten sonra (**sort**) yemeğe çıkabilir veya köpeğinizi dolaştırmaya götürebilirdiniz; çünkü sıralama bitinceye kadar bilgisayarınızdan bir başka amaçla yararlanmanız söz konusu olamazdı. Oysa, UNIX işletim sisteminde, sıralamayı **arka planda** bir iş olarak başlattıktan sonra, ön planda başka işler yapmanız mümkündür. Bunu yapabilmek için tek yapmanız gereken, arka planda yapılmasını istediğiniz işi başlatan komutun sonuna bir **&** işareti eklemekten ibarettir.

```
% sort muster-i-dosyasi &
```

&, işi arka planda yürütmek istediğinizi belirtiyor

Elbetteki her iş bu şekilde arka planda çalıştırılmaya uygun değildir. Örneğin, bir muhasebe fiş giriş programı gibi; kullanıcının sürekli olarak klavyeden bilgi girmesini gerektiren programlar arka planda çalıştırılsa bile, sürekli ilgi istedikleri için bu tip bir çalışma rahat olmaz. Oysa, yukarıdaki sıralama örneğimizde, sıralama süresince kullanıcıdan herhangi bir bilgi istenmeyecektir. Sıralama programı arka planda sessizce çalışıp işini bitirecektir.

Bazı programlar, kullanıcıdan bir bilgi istememekle birlikte, sürekli olarak ekrana, yaptıkları işin gelişmesini açıklayan bilgiler dökerler. Bu tip bir programı arka planda çalışmak üzere başlattığınızda; sürekli olarak ekrana gelen bilgiler yüzünden ön planda başka bir iş yapmanıza pek olanak kalmaz. Örneğin, genellikle teybe yedekleme yapmak için kullanılan **tar** komutunu

```
% tar -cvf /dev/rst1 /home/ayfer &
```

şeklinde vererseniz (bu komutla ilgili detaylı bilgiyi daha ileride vereceğim; şimdilik komutun ne yaptığı ve parametrelerinin ne olduğu üzerinde durmayınız), program arka planda teybe yedekleme yapacaktır, ama bir

yandan da kopyalamayı tamamladığı bütün dosyaların isimlerini ekrana listeleyecektir. Böyle her saniye yeni bir satır gelen ekranda başka bir iş yapmak pek kolay olmayacaktır.

Ancak aynı komutu

```
% tar -cvf /dev/rst1 /home/ayfer > tarmesajlari &
```

şeklinde verirsiniz ekrana gelmesi gereken tüm mesajlar, çalışma dizininizde **tarmesajlari** isimli bir dosyaya yönlendirilmiş olur. İş bittikten sonra **tarmesajlari** dosyasına bakarak teybe kopyalama işinin başarıyla bitip bitmediğini ve kopyalanan dosyaların listesini görebilirsiniz.

Ön Planda Çalışan Programları Arka Plana Atma

Sadece C Shell'de geçerlidir

Bazı durumlarda, başlattığınız bir programın ne kadar süreyle çalışacağını önceden kestiremezsiniz. İşin uzun süreceğini ve sessiz çalışan bir iş olduğunu sonradan farkedersiniz; ya da işin bu özelliklerini bilseniz bile, boş bulunup komut satırının sonuna **&** koymadan Enter tuşuna basıverirsiniz. Örneğin, bir dizindeki tüm dosyaları bir başka diske ya da dizine çekme komutunu

```
cp -r /home/ayfer /disk2/home2
```

şeklinde verdiğiniz ve programı ön planda çalıştırdığınızı varsayalım. Başlattıktan bir kaç saniye (ya da birkaç dakika) sonra işin uzun süreceğini farkettiler ve **''Tüh! Keşke arka planda başlatsaydım!''** dediniz. Eğer kabuk programı olarak **csh** kullanıyorsanız sorun değil...

Klavyenizden

```
^Z
```

tuşuna basarsanız (Control tuşu basılıyken Z tuşuna da basarsanız) ekranda

Suspended.

mesajını görürsünüz. Bu mesaj, o sırada ön planda çalışan işinizin geçici olarak askıya alındığını göstermektedir; ancak, buradaki **durdurma** kelimesi, işinizin tamamlanmadan kesildiği anlamında kullanılmamaktadır. Buradaki **durdurma**, müzik kaseti çalan teyplerdeki PAUSE düğmesinin görevine benzeyen bir durdurmadır. İşiniz çalışmaya ara vermiş ve devam edebilmek için sizden bir komut bekler durumdadır.

Bu noktada

% bg	Background
------	------------

komutu verirsiniz işiniz arka planda çalışmaya devam edecektir. Ancak, o anda ekranda bir de

```
[1] cp ... &
```

mesajı görünecektir. Bu mesajın kısaca anlamı şudur :

*Programınız (komutunuz) arka planda çalışır duruma alındı...
Bu şekilde arka plana atılan işler arasında sıra numarası 1 oldu.*

Bu işi tekrar ön plana almak isterseniz

% fg %1	Foreground
---------	------------

komutunu verebilirsiniz. (Eğer birden fazla arka plana atılmış işiniz varsa, % işaretinden sonra o işin numarasını yazmayı unutmamalısınız.)

Arka planda çalışmak üzere başlatılacak; ya da sonradan arka plana atılacak işlerin sayısı ile ilgili herhangi bir sınırlama yoktur. Ancak arka plan ya da ön plan olsun, çalışan her işin bilgisayarın performansından bir pay alacağını unutmamalısınız.

Bazan, arka planda başlattığınız ya da sonradan arka plana attığınız işlerin hesabını şaşırabilirsiniz. Böyle bir durumda

% jobs

komutunu verirsiniz, arka plana atılmış işlerin bir listesini alırsınız.

Benzeri bir listeyi

% ps	process status
------	----------------

komutuyla da alırsınız. Ancak **ps** komutunun görevleri biraz daha farklı olabilmektedir. **ps** komutu, parametresiz olarak verildiğinde, kullanıcı olarak sizinle ilgili olarak başlatılmış olan işlerin listesini verir. UNIX işletim sisteminde, siz tek bir iş yaparken (hatta hiç program çalıştırmazken bile) sizinle ilgili birkaç iş, UNIX **kabuk** programı tarafından çalıştırılmaktadır (bu arada kabuk programının kendisi de çalışmaya devam etmektedir.) Hele X Windows, Motif, OpenWindows gibi grafik kullanıcı arabirimleri (*GUI : Graphical User Interface*) kullanıyorsanız; sizinle ilgili olarak onlarca iş başlatılmış olduğunu göreceksiniz. **jobs** komutu sizin tarafınızdan arka plana atılmış işleri; **ps** komutuysa, daha geniş kapsamlı olarak sistemdeki işleri ve bu işlerle ilgili çalışma istatistiklerini listeler.

UNIX'de SÜREÇ (process) Kavramı - Tekrar Bir Göz Atış



Süreçler, UNIX işletim sisteminde çok iyi anlaşılması gereken bir kavramdır. Bu nedenle, bu konuyu bir kez daha tekrarlamak istiyorum. Okuyucunun eski bir bölümü tekrar okumasını istemektense, burada tekrarlamamın daha sempatik ve yararlı olduğunu düşünüyorum. Üstelik, okuyucunun UNIX'e daha yatkın bir duruma geldiğini dikkate alarak biraz daha ayrıntılı olarak anlatma ve örnekleme olanağı bulmuş olacağım. Tekrarlamaya gerek görmeyen okuyucular bu bölümü atlayabilirler.

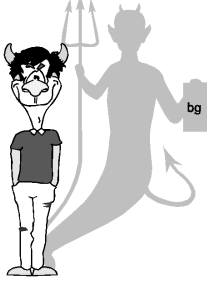
UNIX, tasarımından kaynaklanan nedenlerle, çeşitli problemlerin çözümü için yazılan programların, problemi parçalayıp, her bir parçanın ayrı bir (ya da birkaç) programdan oluşacak şekilde yazılmasına olanak sağlamaktadır. Pek açık olmadı, farkındayım. İsterseniz örneklerle biraz daha açmaya çalışayım.

UNIX'i yazan insanların bakış açısıyla düşünelim... İşletim sistemi aynı anda şu temel işlerle ilgilenmek zorunda : (bu temel işler herhangi bir sıraya göre verilmemiştir)

- 1) Kullanıcı programlarının veya bu programları oluşturan süreçlerin bir seferinde ve kesintisiz olarak 200 milisaniyeden fazla MİB kullanmalarını önlemek için işletim sisteminin bir modülü sürekli olarak zamanı izlemeli.
- 2) Hangi kullanıcının hangi terminalden ne zaman sisteme gireceği belli olmadığından, işletim sisteminin bir modülü, devamlı olarak terminallerin bağlandığı arabirimleri gözlemeli, sisteme **login** etmek isteyen kimse varsa, ona hizmet edecek programları başlatmalı.
- 3) Hangi kullanıcının ne zaman hangi yazıcıya bir şeyler göndereceği bilinemediğinden ve ayrıca yazıcıların herhangi bir anda, ne durumda olacakları kestirilemeyeceğinden dolayı bir (ya da birkaç) program sürekli olarak yazıcılarla ilgilenmeli.
- 4) Elektronik posta (*mail*) keza...
- 5) Hele hele bilgisayar bir bilgisayar ağına bağlıysa durum daha da yürekler acısı... İzlenmesi gereken çevre birimleri yetmezmiş gibi bir de sağdan soldan gelen erişim isteklerine yanıt vermek gereklidir.
- 6) vs. vs.

UNIX işletim sistemini tasarlayanlar, bu problemlerin çözümüne doğru önemli bir kavram geliştirmişler : **SÜREÇ (PROCESS)**.

Bu tasarıma göre, yapılması gerek tüm işler için ayrı ve bağımsız çalışabilen programlar yazılır, bu bağımsız programlar gerektiğinde birbirleriyle mesaj alışverişinde bulunabilirler ve sıradan kullanıcı programları gibi bilgisayarların MİB, bellek gibi kaynaklarını paylaşırlar.



Bazı süreçlere **daemon** (sözlük anlamıyla : kötü ruh, iblis, iblisin uşağı) adı verilir. Neden bu adın seçildiğini bilmiyorum ama bu tip süreçlerin ortak özellikleri şunlar:

- ◆ Arka planda tamamen sessizce çalışırlar,
- ◆ Hata mesajları olsa bile bunu genellikle ekrana değil, kendilerine ait bir disk dosyasına yazarlar (*Log File*),
- ◆ Klavyeden müdahale gerektirmezler,
- ◆ İsimleri genellikle **d** harfiyle biter.

ngi bir anda sistemde çalışmakta olan süreçlerin listesini almak için **ps** komutu kullanılır. BSD veya System V UNIX için küçük farklar gösteren bu komutun en çok kullanılan kalıpları ve bu kalıplar için örnekleri izleyen sayfalarda bulacaksınız.

BSD

BERKELEY UNIX (BSD)

<code>ps</code>	<i>Kullanıcıyla ilgili süreçleri listeler</i>
<code>ps -a</code>	<i>Tüm kullanıcılarla ilgili süreçleri listeler</i>
<code>ps -ax</code>	<i>Tüm kullanıcılar ve sistemle ilgili süreçleri listeler</i>
<code>ps -axl</code>	<i>l (le harfi) seçeneği ayrıntılı liste verilmesini sağlar</i>

SVR4

AT&T UNIX (System V)

<code>ps</code>	<i>Kullanıcıyla ilgili süreçleri listeler</i>
<code>ps -a</code>	<i>Tüm kullanıcılarla ilgili süreçleri listeler</i>
<code>ps -ae</code>	<i>Tüm kullanıcılar ve sistemle ilgili süreçleri listeler</i>
<code>ps -ael</code>	<i>l (le harfi) seçeneği ayrıntılı liste verilmesini sağlar</i>

BSD UNIX'de ps komutu kullanımına bir kaç örnek....

```
abc:/home/ayfer> ps
  PID TT STAT  TIME COMMAND
29011 a  R    0:00 ps
abc:/home/ayfer>
```

```
abc:/home/ayfer> ps -x
  PID TT STAT  TIME COMMAND
28731 a  S    0:00 -tcsh (tcsh)
29010 a  R    0:00 ps -x
```

```
abc:/home/ayfer> ps -axl
  F UID  PID  PPID CP PRI NI  SZ  RSS WCHAN  STAT TT  TIME COMMAND
 80003  0    0    0  0 -25  0  0    0 runout  D   ?   1:20 swapper
20088000  0    1    0  0  5  0 52    0 child  IW   ?   0:08 /sbin/init -
 80003  0    2    0  0 -24  0  0    0 child  D   ?   0:04 pagedaemon
 88000  0   58    1  0  1  0 68    0 select  IW   ?   0:58 portmap
 88000  0   63    1  0  1  0 224    0 select  IW   ?  14:39 ypserv
 88000  3   65    1  0  1  0 36    0 select  IW   ?    0:01 ypbind
 88000  0   67    1  1  1  0 40    0 select  IW   ?   0:00 rpc.yppupdate
 88000  0   69    1  0  1  0 40    0 select  IW   ?   0:00 keyserver
 88001  0  114    1  0  1  0 40   64 select  I   ?   1:09 in.routed
 88000  0  117    1  0  1  0 324    0 select  IW   ?   2:02 in.named
 88001  0  120    1  0  1  0 16    0 nfs_dnlc I   ?   0:00 (biody)
 88000  0  134    1  0  1  0 60    0 select  IW   ?   0:16 syslogd
 88000  0  148    1  0  1  0 108    0 select  IW   ?   1:09 rpc.mountd -
 88000  0  153    1  1  1  0 52    0 select  IW   ?   0:00 rpc.statd
 88001  0  154  149  0  1  0 28    0 socket  I   ?  23:57 (nfsd)
 88000  0  161    1  0  1  0 84    0 select  IW   ?   0:00 rpc.lockd
 88000  0  167    1  0  1  0 80    0 select  IW   ?   0:06 /usr/etc/rpc
 80201  0  182    1  0 15  0 12    4 kernelma S   ?  109:22 update
488000  0  185    1  0  1  0 56    0 Heapbase IW   ?   0:00 cron
 88000  0  191    1  0  1  0 48    0 select  IW   ?   0:47 inetd
 88000  0  194    1  0  1  0 52    0 select  IW   ?   0:00 /usr/lib/lpd
20488020 560 4969    1  0  1  0 132    0 socket  IW   ?   0:04 ncftp ramiga
 88401  0 24725  167  0 25  0  0    0      Z   ?   0:00 <defunct>
20088000  0 27454    1  0  3  0 40    0 Heapbase IW  co   0:00 - cons8 cons
204882018700 28731    1  0 15  0 216 700 kernelma S   a   0:00 -tcsh (tcsh)
200000018700 29007 28731 24 31  0 216 468      R   a   0:00 ps -axl
abc:/home/ayfer>
```

```

abc:/home/ayfer> ps -ax
PID TT STAT  TIME COMMAND
  0 ?  D    1:20 swapper
  1 ?  IW    0:08 /sbin/init -
  2 ?  D    0:04 pagedaemon
 58 ?  IW    0:58 portmap
 63 ?  IW   14:39 ypserv
 67 ?  IW    0:00 rpc.yupdated
 69 ?  IW    0:00 keyserv
114 ?  I    1:09 in.routed
117 ?  IW    2:02 in.named
120 ?  I    0:00 (biob)
134 ?  IW    0:16 syslogd
142 ?  IW    0:05 /usr/lib/sendmail -bd -qlh
148 ?  IW    1:09 rpc.mountd -n
149 ?  I   24:20 (nfsd)
...

```



Meraklı olan okuyucular için yukarıdaki örneklerde adı geçen bazı süreçlerin açıklamalarını yapmak istiyorum. Ancak bunların sadece birer örnek olduğunu, gerçek süreç listelerinin daha uzun ve/veya daha farklı olacağını unutmamalısınız.

Süreç	Görevi
lpd	<i>Line printer daemon</i> . Yazıcılarla ilgilenir. Yazıcı(lar)dan dökülmek üzere gönderilen bilgilerin sıraya konmasından ve yazıcıların yönetiminden sorumludur. kill komutunu kullanarak bu süreci öldürürseniz artık kimse yazıcılardan döküm alamaz.
inetd	<i>Internetworking daemon</i> . Bilgisayar ağı üzerinden gelip giden servis istekleriyle ilgilenir.
cron	Günün, haftanın, ayın belirli zamanlarında çalıştırılması gereken programları izler; zamanı gelen programı başlatır.
update	Disk tampon bellek alanlarının belirli aralıklarla disklere kaydedilmesini sağlar(<i>flush</i>); böylece bir arıza ya da enerji kesintisi durumunda ortaya çıkabilecek bilgi kaybını en aza indirir.

<code>syslogd</code>	Sistemde meydana gelen önemli olayları uygun log dosyalarına kaydeder. (Sisteme giren/çıkan kullanıcıları, root kullanıcı kimliğini alan kullanıcıları, donanımla ilgili sorunları vs. kaydeder.)
<code>swapper</code>	Ana belleğin yetmediği durumlarda diskte ayrılmış olan swap alanının sanki ana belleğin bir uzantısıymış gibi kullanılmasını sağlar.
<code>init</code>	Kullanıcı terminallerini dinler. Sisteme girmek üzere terminalini açan bir kullanıcıya rastlarsa, onun sisteme login edebilmesi için gerekli hazırlıkları yapar.
<code>in.routed</code>	Bilgisayar ağı üzerinde, çeşitli mesajların doğru bilgisayarlar arasında ulaşmasından sorumludur.
<code>in.named</code>	Bilgisayar ağı üzerinde, adresi bilinmeyen bilgisayarların yer ve adreslerinin bulunabilmesi ile ilgili protokolleri yürütmekle görevlidir.
<code>-tcsh</code>	İlgili olduğu kullanıcı terminali için çalışan bir kabuk programıdır.

Önceki sayfalardaki **ps** komutlarının çıktılarının oldukça karmaşık bir görünümde olduklarını kabul etmek lazım. Ancak, gözünüzü, sürecin adının gösterildiği son kolonla **PID** kolonlarına alıştırırsanız, tipik bir UNIX kullanıcısının gerek duyabileceği tüm bilgileri almış olursunuz. Aradığınız sürecin adı bu listede varsa, programınız çalışıyor demektir (aslında çalışmaktan çalışmaya fark var... UNIX açısından çalışıyor, fakat sizin istediğiniz işleri yapmıyor olabilir; o başka mesele). **PID** kolonundaki numaraysa UNIX'in sizin programınızı izlemek ve denetlemek için kullandığı **süreç tanıtım numarasıdır (PROCESS ID)**. UNIX altında çalışan programların herbirinin özgün (*unique*) bir **PID** numarası vardır. Bu numaranın büyüklüğü veya küçüklüğü bir anlam taşımaz. Nitekim, UNIX, programlara verdiği PID numarası 65535'e ulaştıkça, tekrar birden başlatmakta bir sakınca görmez. (PID numaraları her UNIX'de 65535 ile sınırlı değildir; bazı uyarlamalarda bu sayı daha da büyüyebilir; ancak bunun kullanıcılar için pek önemi yoktur.)

Eğer bir süreç (ya da program) çakılıp kaldıysa; ya da sizin istediğiniz gibi davranmıyorsa o süreci **ÖLDÜREBİLİRSİNİZ**. Deyim çok tatmin edici; değil mi?

% kill nnn	(kill process)
------------	-----------------

Bu komut, tanıtım numarası (**PID**) olan süreci öldürmek için kullanılır. Tek bir komutla birden fazla süreci beraber öldürebilirsiniz.

kill 154 185 117 gibi...

Eğer öldürmek istediğiniz süreç ölmek için direniyorsa,

% kill -9 nnn	şartsız öldürme
---------------	-----------------

formunu deneyiniz. Eğer süreç gene ölmezse daha fazla uğraşmayınız. Bazı süreçler ölemezler. Bu tip süreçlere **zombie** adı verilir. (Hoş... Değil mi ?) **Zombie** süreçler genellikle pek sistem zamanı ya da bellek harcamazlar. Bu tip süreçlerin sistemde çalışır durumda olması genellikle zararsızdır.

Öldürülen bir süreç, daha önce başka süreçler yarattıysa; yani eğer bir *ebeveyn* süreçse (**parent process**), büyük olasılıkla o yavru süreçler de (**child process**) ölecektir. Bu nedenle mecbur olmadıkça süreçleri sona erdirmek için bu öldürme yöntemini kullanmayınız.

kill komutu ile sadece kendinize ait süreçleri öldürebilirsiniz. Sisteme, ya da başka kullanıcılara ait süreçleri öldürme yetkisi sadece **root** kullanıcıya aittir.



Zaman zaman klavyenizin kilitlendiği ve sistemin hiç bir komuta tepki göstermediği durumlarla karşılaşacaksınız. **Böyle bir durumda, bilgisayarı kapatıp açmayı aklınızdan bile geçirmemelisiniz.**

- Eğer UNIX bilgisayarını bir terminalden kullanıyorsanız, terminalinizi açıp kapatmayı bir denemenizde sistem açısından herhangi bir tehlike yoktur.
- Eğer bilgisayarı sistem konsolundan kullanıyorsanız (iş istasyonlarında olduğu gibi) ekranı kapatıp açmak bir yarar sağlamaz. Bilgisayarı kapatmayı düşünmemelisiniz dahi.. Peki ne yapılmalı?

- İlk denemeniz gereken Ctrl-Q tuşu. Daha önce yanlışlıkla Ctrl-S tuşuna basmış olabilirsiniz. Ctrl-S tuşu, ekrana gelen dökümleri durdurmak için kullanılan bir komuttur. (Ctrl-Q ise '**devam et**' anlamındadır). (**XON/XOFF** seri haberleşme protokolu).
- Olmazsa Ctrl-C tuşu ile çalışan işi kesmeyi deneyin. Gene olmuyorsa Ctrl-Z tuşuyla çalışan işi askıya almayı denemelisiniz. Her iki durumda da denemeniz başarılıysa ekranınızda **sistem hazır işaretini (prompt) göreceksiniz**. Hemen **ps** komutuyla çalışan işlerin bir listesini alın. Sorun yaratan sürecin numarasını (PID) öğrenip onu öldürmeyi deneyin. Ölmüyorsa çok ısrar etmeyin. Gerek duyarsanız, sistem yöneticisinden yardım isteyin.
- Eğer klavyeniz kilitlenmişse; sisteme bir başka terminal veya varsa, bilgisayar ağındaki bir başka bilgisayar üzerinden **login** etmeyi deneyin (**telnet** veya **rlogin** komutları). Bunu başarabiliyorsanız hemen **ps** komutuyla çalışan işlerin bir listesini alın. Sorun yaratan sürecin numarasını (PID) öğrenip onu öldürmeyi deneyin. Ölmüyorsa çok ısrar etmeyin. Gerek duyarsanız sistem yöneticisinden yardım isteyin.
- Hiç bir çareniz kalmadıysa en az 5 dakika bilgisayara dokunmadan bekleyin (eğer çalışıyorsa **update** daemon'u tampon belleği boşaltsın diye) sonra bilgisayarı kapatın. Tüm bilgisayarlarda olduğu gibi en az 30 saniye kadar bekleyip tekrar açın. **Bu tip zoraki işlemlerden sonra disk kayıtlarınızda büyük çaplı kayıplara hazırlıklı olmalısınız.**

% nohup	no hangup
---------	-----------

Bir UNIX bilgisayarına ulaşmak için kullandığınız terminali ya da terminal gibi davranan bir PC'yi (Terminal Emulation yazılımı çalışan bir PC) kapatırsanız veya **logout** komutu ile sistem bağlantınızı keserseniz, o terminal bağlantısıyla ilgili tüm süreçler (hem ön, hem arka plandaki süreçler) UNIX tarafından öldürülür.

Ender de olsa, bazı durumlarda sistemden çıkmanıza rağmen, başlatmış olduğunuz bir işin kesilmeden devam ettirilmesini isteyebilirsiniz. Örneğin, **Internet** üzerinden çok uzun bir dosya çekiyor ve bu kopyalama işinin siz eve gittiğinizde de devam etmesini istiyor olabilirsiniz. Böyle bir durumda ilk akla gelen "**logout** etmeden" terminali açık bırakarak eve gitme çözümü pek iyi bir çözüm değildir. Sizin yokluğunuzda, açık bırakmış olduğunuz terminalin başına oturan birisi, sizin kimliğinizle hoşlanmayacağınız işler yapabilir.

İşte böyle durumlarda **nohup** komutu kullanılır. **logout** ettiğinizde kesilmesini istemediğiniz bir programı başlatırken kullanmanız gereken komut satırı

```
% nohup komut [varsa parametreleri] &
```

olmalıdır.

```
% tcsh
```

```
t c-shell
```

tcsh, **csch**'e göre oldukça üstün özellikleri olan bir kabuk programıdır; ancak, şimdilik hiç bir UNIX uyarlamasında standart olarak bulunmamaktadır. Sistem yöneticinize bir danışınız; eğer sisteminizde varsa, sizin için **login** kabuğu olarak **tcsh** çalıştırılmasını sağlamasını isteyiniz. (Sistem yöneticileri; kola, kahve, piza gibi rüşvetleri kabul ederler. Para falan teklif etmeyiniz. Paranın ne olduğunu bilseler, UNIX sistem yöneticisi olmazlardı...)

tcsh'in belki de en iyi iki özelliği, aynı MS-DOS'daki DOSKEY yardımıyla olduğu gibi, yukarı aşağı tuşlarla eski komutlar arasında dolaşmanızı sağlaması ve dosya adlarının tamamını yazmadan komut yazmanıza olanak sağlamasıdır. **tcsh** hakkında daha fazla reklama gerek yok. Sisteminizde varsa nasıl olsa öğrenirsiniz; yoksa, zaten özelliklerini öğrenip gıpta etmenin bir anlamı yok.

“**tcsh**'i nereden bulurum ?” diyenlere ise cevabım “**Internet'den**” olacaktır. Birçok üniversitenin bilgisayarında, herkese açık alanlarda **tcsh** ve daha bir sürü ilginç program bulabilirsiniz. “**Internet'de neyin nesi ?**” diyorsanız, bu kitabı okumakla daha fazla vakit kaybetmemenizi öneririm.

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

Gail & Paul Anderson
Prentice-Hall, 1986
ISBN 0-13-937468-X 025

UNIX POWER TOOLS

J. Peek, Tim O'Reilly & M. Loukides
O'Reilly & Associates, 1993
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. (<i>Disk Free</i>). 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 dosyaları silmek 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 <i>'object'</i> 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 (<i>disk usage</i>) durumunu görmek için.

Aslında bütün bu silme işlerini tek bir **rm** komutuyla, yapabildik; 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 **cs**h 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 Mayıs 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 programının çalıştırılması sırasında **cs**h kabuğunun kullanılması gerektiğini belirtir. Eğer bir başka kabuk çalışıyorsa; yeni bir **cs**h kabuğu başlatılır ve program bitince bu yeni **cs**h 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 büyükse, 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 kullanıcının 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 :

- ff** adlı bir kabuk programı yaratacağız.
- Bu program tek parametre ile kullanılırsa, çalışma dizinimizde ve alt dizinlerinde, parametrede verilen dosyayı arayan **find** komutunu çalıştıracacağız.
- Eğer program iki parametre ile kullanılırsa, ikinci parametredeki dosyayı birinci parametredeki dizinden başlayarak arayacak bir **find** komutu çalıştıracacağız.
- 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 **cs** 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. **\$#** sembolü 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...

<code>ff kitaplar</code>	Çalışma dizini ve altındaki dizinlerde kitaplar adlı dosya veya dizini ara.
--------------------------	--

<code>ff /usr kitaplar</code>	/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 kullandığım 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. Sanıldığı gibi 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 listeyi 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*
```

* 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 tanımlandığı dosya (passwd), bilgisayar ağıyla ilgili tanımların 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 tanımlar 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 kontrolü, o donanım unsurunu tüm özellikleriyle tanıyıp denetleyebilen bir programa geçirecektir (**device driver**).

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 firmasının geliştirdiği, BSD uyumlu SunOS 4.1.x UNIX'de kullanıldığı şekliyle söz edeceğim.

BSD

/dev Dizinde 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 (iş bitince kaseti başa saracak şekilde kullanım için) (<i>r : rewind</i>)
/dev/rst1	Sistemdeki ikinci teyp birimi (iş bitince kaseti başa saracak şekilde kullanım için)
/dev/nrst0	Gene sistemdeki ilk teyp birimi; ancak bu kez; iş bitince kaseti başa sarmayacak şekilde kullanılması söz konusu. (<i>n : no rewind</i>). 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) (<i>r : raw device</i>)
/dev/ttya	Sistemdeki birinci seri arabirim (RS232)
/dev/ttyb	Sistemdeki ikinci seri arabirim (RS232)
/dev/bpp0	Sistemdeki ilk paralel yazıcı arabirimi (Centronics)

/dev/tty0	Ethernet üzerinden bağlanan terminal emülatörleri için sanal seri arabirimlerden ilki (<i>pseudo tty port</i>).
/dev/tty1	Ethernet üzerinden bağlanan terminal emülatörleri için sanal seri arabirimlerden ikincisi (<i>pseudo tty port</i>).
/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 iliştmek 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 tercihinine 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

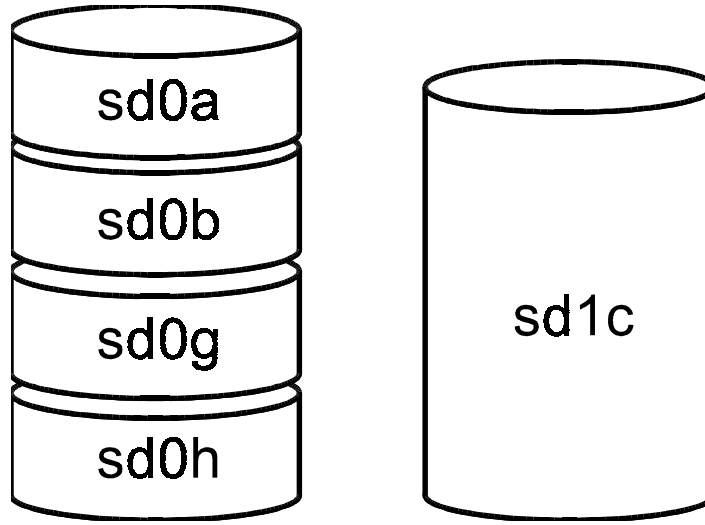
/dev/sd0a	SCSI adresi, sistemdeki sıfırıncı (ilk) diske karşılık gelen diskin a isimli bölümü
/dev/sd1c	SCSI adresi, sistemdeki bir numaralı (ikinci) diske karşılık gelen diskin c isimli bölümü
/dev/sd3h	SCSI adresi, sistemdeki 3 numaralı diske karşılık gelen diskin h isimli bölümü

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 :

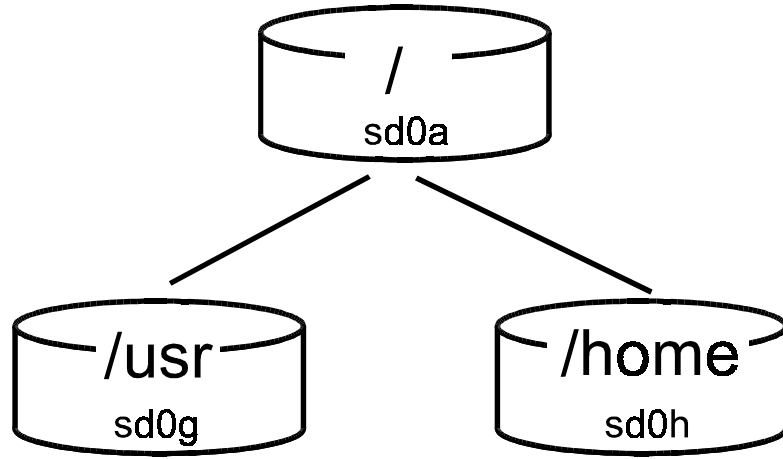
Sıfırıncı disk : **/dev/sd0a** Birinci disk :

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

Bu disk bölümlerinin UNIX altında kullanılabilmesi 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...)

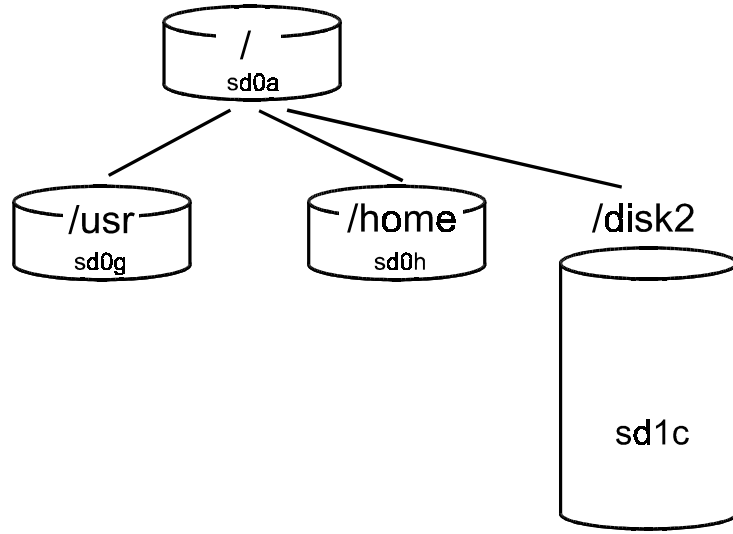
Sonra, / dizini (birinci diskin a bölümü : /dev/sd0a) altında **disk2** isimli **boş** bir dizin bulunduğuna emin olmalıdır. Eğer bu isimde bir dizin yoksa, **mkdir /disk2** komutu iş görecektir. (Hatırlatayım; bu işleri ancak **root** kullanıcı yapabilir).

Son olarak

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

mount noktası
(*mount point*)

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 :

- Sistemde sadece bir disk var (ikinci disk varsa bile, **mount** edilmemiş). Çünkü disk isimlerinin hepsi **sd0** ile başlıyor.
- İlk diskin **a** bölümü **/** olarak, **g** bölümü **/usr** olarak, **h** bölümüyse **/home** olarak **mount** edilmiş.
- Tüm disk bölümlerindeki dosya yapılarının tipi 4.2 imiş. (SunOS tarafından dosya yapısı sürüm numarasıyla (*version*) ilgili olarak kullanılan özel bir kod).
- Disk bölümlerinin hepsi okuma ve yazmaya açıkmiş (**rw**). (Bu dosya sistemindeki dosya ve dizinlerin kullanıcı yetki kalıplarında belirtilen yetkiler saklı kalmak kaydıyla).
- /** 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.



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** kullanıcısı 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	umount
veya	
# umount /mmm	

komutu kullanılmalıdır. Burada **xxx** sürücünün /dev dizindeki 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** komutları, 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 uyarılırsınız; 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; bulunsan 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.
                           Kopyalama işleri...
                           CD'yi umount etmek
                           için...
Device Busy                Bindığınız dalı
                           kesemezsiniz..
# cd /home/ayfer           Çalışma dizinini değiştirdik
# umount /cdrom
# eject cdrom              CD'yi çıkarmak için
```

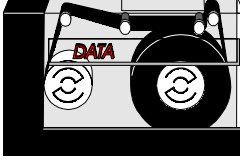
SunOS 5.x.x (SOLARIS 2.x) Kullanıcılarına...



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. İş 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 **yedeklemedir**. 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 ortamıdır**. Örneğin bir UNIX program paketi satın aldığınızda genellikle size bir teyp kaseti gönderilir. (Son yıllarda yazılım dağıtımını 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 yanı sıra; 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. Hepsinin için kullanılan komutlar aynıdır. (**tar**, **cpio**, **mt**, **dump**, **ufsdump** gibi komutlar).

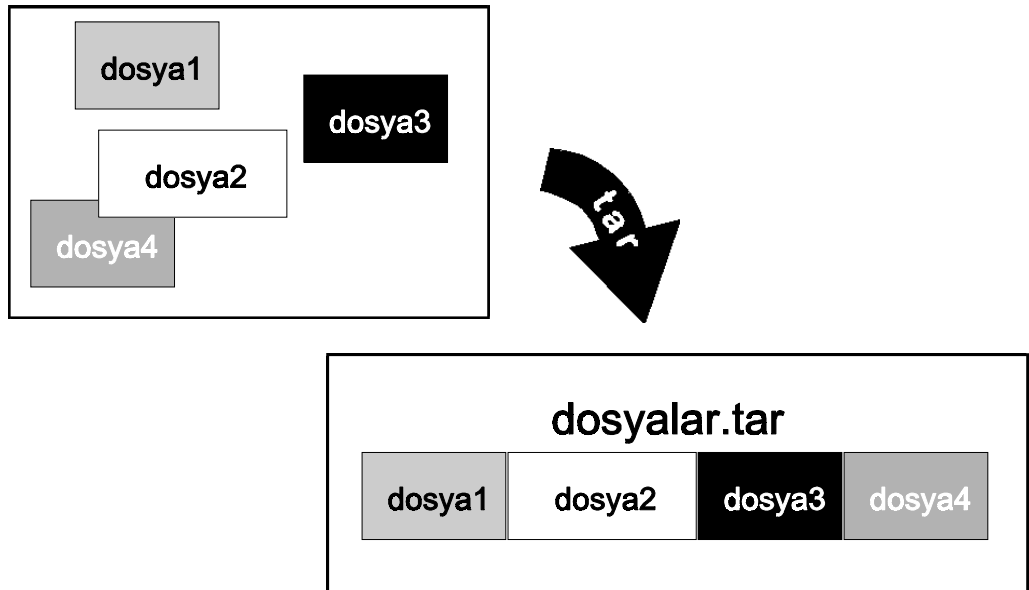
MS-DOS teyp kullanıcılarını çatlatacak bir haberim var. UNIX'de kullanılacak **teyplerin formatlanması gerekmektedir**.

UNIX tar Dosyaları

(tape archive files)

UNIX işletim sisteminde yedeklenecek veya teybe çekilip bir yere 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	<i>table of contents</i>
------------------------	--------------------------

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	<i>extract</i>
------------------------	----------------

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 vererseniz, **tar** komutu teyp üzerinde çalışacaktır. Örneğin

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



komutunu vererseniz, **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 teypten, 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

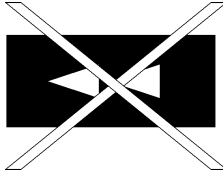


İşi bitince başa saran teyp sürücüler

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 sürücüler

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 indirmanız mümkün olacaktır. Ancak, burada küçük bir sorun var. Teypin 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
```

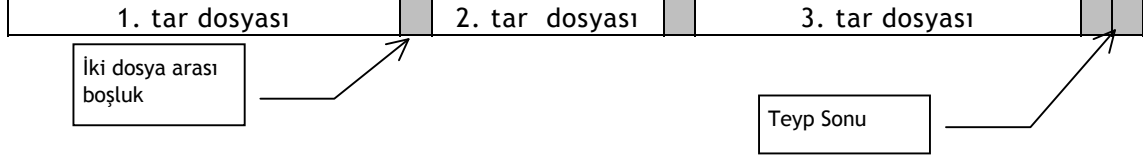
Bu hata mesajını aldığınızda okuyucu kafa bu boşluğu atlamış olacağından son komutu tekrarlıyorsanız, ikinci **tar** dosyasını işlemeye başlamış olursunuz; tabii ikinci **tar** dosyası gerçekten varsa... Eğer tekrar aynı hata komutunu alırsanız, kasetin gerisi boş demektir. Daha fazla uğraşmanız anlamsızdır.

Şimdi bir kaç senaryo oynayalım:



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

Teyp şeridi



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 başlayan 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 kullanacağınız 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 içgüdüünüz 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 yalnızca 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 izin 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 .* *"
```

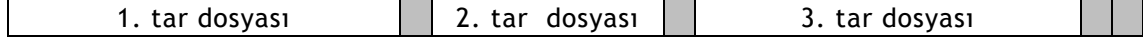


şeklindeki bir komut tehlikeli olabilir. Bulunduğunuz dizinde, normal dosyalar ve izinler yanısıra yer alan **.** ve **..** isimli iki özel izin vardır. Bunlardan **..** bir üst düzey dizini gösterdiği için; verdiğiniz komut, bir üst düzey dizini de teybe çekmek istediğiniz şeklinde yorumlanabilir.

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.

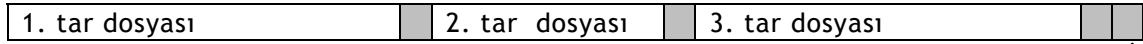
4. Sık tekrarlanan bir başka hatayı ise bir örnekle açıklamak istiyorum. Diyelim ki elimizde aşağıdaki şekilde kaydedilmiş bir teyp kaseti var :



Kasetin sonuna ekleme yapmayı planlarken yanlışlıkla, **tar cvf** komutunu kaset teybin başındayken verdiniz. Anında hatanızı farkettiler 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

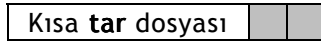


şeklindeyken, kısa bir **tar** dosyası oluşturmak için

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

Kayıt Sonu
İşareti

komutlarını vererseniz 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; “bulduğum 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 iliştip 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 (<i>file</i>)" 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 (<i>pipe</i> 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.)

Parametrelili 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 büyük_dosya > /dev/null
büyük_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 büyük_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 büyük_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. (Yalnızca 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 saharlar 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, sirasız hali			sort sirasız > 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 sirasız > sirali komutundan sonra			sort +1 -2 sirasız > 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 verir durur.
Örneğin

dosya1	dosya2
Elma, insanlar tarafından çok sevilen bir meyvadır. Uzun yıllardır ben de elma yemeyi bir alışkanlık haline getirdik.	Elma, insanlar tarafından çok sevilen bir meyvadır. Uzun yıllardır biz de elma yemeyi bir alışkanlık 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 unutmamamız 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
```

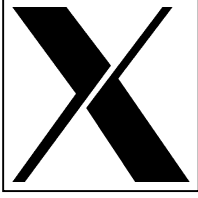
Bazen, içinde hiç bir şey olmayan boş bir dosya yaratmanız gerekebilir. Böyle bir durumda **touch** komutundan yararlanabilirsiniz.

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.

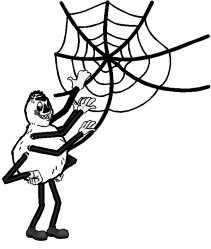
X-Windows; X11R5 (*X11 Release 5*) sürümüyle birlikte grafik ekran sürebilen tüm UNIX bilgisayarlarında kullanılan bir Grafik Kullanıcı Arabirim standardına dönüştü. Bu standart etrafında çeşitli Pencere Yönetim (*Window Manager*) yazılımları üretildiyse de hepsinin temelinde X11 yazılımı yatmaktadır. 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.

UNIX Bilgisayar Ağları

Networks



Bilgisayar kullanımı yaygınlaştıkça insanların da bilgisayarlardan beklentileri artıyor. Söz gelimi, 1970 li yılların başında orta boy bilgisayar denince 16 veya bilemediniz 64 KiloByte belleği olan, iki disket sürücülü veya taş çatlarsa 10 MegaByte'lık bir disk sürücüsü olan ve bu kadarcık kapasiteyle bile bir kaç kişiye birden hizmet etmeye çalışan bilgisayarlar akla gelirdi.

Elektronik mühendisleri deliler gibi çalışıp bilgisayarların hız, bellek kapasitesi gibi özelliklerinde büyük sıçramalar gerçekleştirdikçe, kullanıcılar ve programcılar daha fazlasını istediler.

İlk UNIX uyarlamaları için 16, 48 KiloByte gibi bellek kapasitesine sahip olan bilgisayarlar kullanılmaktaydı; bugün UNIX kullanmak için en az 16 MegaByte bellek gerektiğini söylüyoruz. Nasıl bu hale gelindi dersiniz?

UNIX gene UNIX; ama kullanıcıların devamlı artan isteklerine yanıt verebilmek için devamlı olarak bir şeyler eklendi. UNIX'e eklenen bu yeni özelliklerden belki de en önemlileri "bilgisayar ağı" kurmaya yönelik özellikleridir. Bu özellikler sayesinde, önce bir bilgi işlem merkezindeki bilgisayarlar birbirlerine bağlanarak güçleri birleştirildi; daha sonra bir iş yerindeki bilgisayarlar daha geniş çapta iş birliğine zorlandılar. İşler büyüyünce farklı bina, hatta farklı şehirlerdeki bilgisayarlar birbirlerine bağlandı.

Rakip bankalar bile müşterilerine daha iyi ve daha hızlı hizmet vermenin an akılcı yolu olarak donanımlarını bir bilgisayar ağı üzerinden bağlayarak iş birliğine gittiler.

Derken insanlar "**Internet**" diye bir kelime kullanmaya başladılar. Önceleri kimse bunun ne denli önemli bir kavram olduğunu anlamadan duydu ve unuttu. Kimileri bunu dünyanın bir yerlerinde kurulu bir süper bilgisayar olarak hayalinde canlandırdı.

1994 yılının sonlarından başlayarak bir çok kişinin kartvizitinde posta adresi, telefon ve faks numaraları yanısıra bir de garip görünümlü **e-mail** adresi yer almaya başladı. (**cayfer@bilkent.edu.tr**).

Internet üzerinde bir **e-mail** adresine sahip olmak neredeyse bir ayrıcalık olmaktan çıkıp, bir zorunluluk olmaya başladı.

Nedir bu Internet?

Basit... Bilgisayar ağıdır... Daha doğrusu; bilgisayar ağlarının ağıdır.

Nasıl evinizdeki telefon, uluslararası telefon ağının bir parçasıysa, önünüzdeki bilgisayar da uluslararası bir bilgisayar ağının bir parçası olduğunda siz de **Internet**'e katıldınız demektir. Artık, **e-mail** adresini bildiğiniz kişi ya da kuruluşlara mesaj gönderebilirsiniz. Üstelik, sizin kullanmanıza izin verilen bilgisayarları, dünyanın neresinde olursa olsun, kullanabilirsiniz. Hemen aklınıza uluslararası haberleşme ücretlerinin yüksekliği geldi, değil mi? Neyseki **Internet** dünyasında durum normal telefon dünyasındakinden farklı. Bir haberleşme kanalının bir çok bilgisayar tarafından paylaşılabilmesi sayesinde, uluslararası bile olsa, "Internet görüşmeleri" düşündüğünüz kadar pahalı olmamaktadır.

Neyse, biz konumuza dönelim (Internet dünyası bir başka "**Kim Korkar.....**" kitabının konusu)...

Bilgisayarları birbirlerine bağlama konusunda ilk çabalar UNIX etrafında gelişti. Bunun iki nedeni vardı:

- 1) Bağlanacak bilgisayarlar en azından yazılım açısından birbirleriyle uyumlu olmalıydı.
- 2) Bu bilgisayarların çok iş ve çok kullanıcı düzenlerini desteklemesi gerekiyordu.

Bu şartları sağlayan, UNIX'ten daha iyi bir işletim sistemi aklınıza geliyor mu? Kimsenin de gelmedi ve UNIX bilgisayar ağlarının temeli yavaş yavaş atılmaya başlandı.

XEROX şirketi Ethernet arabirimini tanımladı ve gerçekleştirdi. Artık farklı üreticilerin bilgisayarlarının birbirleriyle konuşması için gereken her şey tamamı. Ortak işletim sistemi UNIX, ortak haberleşme alt yapısı Ethernet; ortak haberleşme diliyse TCP/IP olarak belirlendi. (**TCP/IP** : *Transmission Control Protocol / Internet Protocol*)

Günümüzde bilgisayar ağı kurmak için gerekli olan donanım ve yazılım unsurları, UNIX bilgisayarlarının standart aksesuarları oldu. Artık bir iş istasyonu satın aldığınızda; TCP/IP yazılımı ve Ethernet arabirimi istemeseniz de verilmekte.

Bu şartlar altında birden fazla UNIX bilgisayar sistemine sahip olan kuruluşlar da hemen bilgisayarları arasında bir kablo çektiler bilgisayar ağlarını kurmaya başladılar.

Peki... Bilgisayar ağları ne gibi avantajlar sağlar?

En başta, bilgi işlem donanım yatırımlarını zamana yayarlar.

Eskiden, bir bilgisayar kiralanacağı zaman (satın alma 1980 lerde başladı) işletmenin en az 4 yıl boyunca işini görecektir bir donanımın planlaması yapıldı. Satın alma devrinin başında da bu gelenek devam etti; ancak bilgisayar ağlarının yararları görüldükçe kuruluşlar daha kısa dönem planlar yapmaya başladılar. Artık, gerek duydukça yeni bilgisayarlar alıp, bunları bilgisayar ağına ekleyip, ellerindeki bilgisayarların güçlerini birleştirebiliyorlar..

Kaynakların daha verimli paylaşılmasını sağlarlar.

Bir bilgisayardaki yazıcıyı, atıl disk kapasitesini bir başka bilgisayardan kullanarak değerlendirmek çok kolaydır.

Bilgisayar arızalarına karşı daha güvenilir bir bilgi işlem düzeni kurmayı sağlarlar.

Kuruluşların haberleşme olanaklarını zenginleştirirler. (e-mail gibi).

Bu kadar reklam yeter. Şimdi, bir bilgisayar ağındaki UNIX bilgisayarında neler yapabileceğinizi görelim.

IP Adresi, Ethernet Adresi

Bir TCP/IP bilgisayar ağındaki her bilgisayarın yalnızca kendisine ait olan iki numarası olmalıdır. **IP adresi** ve **Ethernet adresi**.

Ethernet adresi, Ethernet arabiriminin donanımı üzerine, üretim sırasında vurulan bir numaradır. Dünyadaki hiç bir iki Ethernet arabirim donanımı aynı numaraya sahip olmamalıdır ve değildir. (Ethernet arabirim üreticileri, ürünlerinde kullanacakları **61.0.05.20.3c.1f** görünümündeki numaraları XEROX şirketinden alırlar. Kullandığınız bilgisayarın Ethernet adresi size hiç bir zaman gerekmeyecektir. Ancak, bazı özel durumlarda sistem yöneticilerinin bu numarayı kullanmaları gerekebilir.

Kullanıcı için önemli olan **IP** adresleridir (*Internet Protocol* adresi). Bu adres **194.27.129.1** gibilerinden, noktalarla ayrılmış, herbiri en fazla 255 değerini alabilen dört gruptan oluşur. (IP adresleme sisteminde bazı adresler özel amaçlar için ayrılmış durumdadır; örneğin IP adresinin ilk sayısı 223 den fazla olamaz. Ancak bu sınırlamaların nedenleri ve mantığı konumuzun çok dışında). Bir bilgisayar ağındaki iki bilgisayarın IP adresi aynı olamaz. Bir bilgisayar ağına yeni bir bilgisayar ekleneceği zaman, bu bilgisayarın IP adresi ağ yönetimi tarafından verilir. **IP** adreslerinin dünya üzerinde dağıtılmasından ABD de bulunan NIC isimli bir kuruluş sorumludur. Bu kuruluş, tüm ülkelerde

akademik birer temsilci seçer (Türkiye'de O.D.T.Ü) ve IP adresi dağıtımını işi bu temsilciler tarafında yürütülür.

IP adresleri kolay hatırlanacak diziler olmadığı için, her bilgisayarın, **IP** adresine karşılık gelen bir de adı olur. İnternet'deki bilgisayarların IP adresleriyle isimleri arasındaki ilişkiyi DNS (*Domain Name Service*) adı verilen yazılımlar kurar. Böylece kullanıcılar numara ezberlemek yerine bilgisayar ismi ezberlerler. (139.179.40.1 yerine **temel.ctp.bilkent.edu.tr** gibi)

Çok sık rastlayacağınız için burada açıklamakta fayda görüyorum : UNIX terminolojisinde bu bilgisayar isimlerinin genel adı *host id* sözcükleridir.

Bilgisayarlara verilen isimleri genellikle sistem yöneticileri seçerler. (Bilgisayarlara verdiği isimlerden, sistem yöneticisinin kişiliğini sezebilirsiniz. arslan, kaplan, kartal gibi bilgisayar isimleri veren bir sistem yöneticisinden korkulur doğrusu.)

Bir kullanıcı olarak, elinizin altındaki bilgisayarın ve bu bilgisayara doğrudan bağlı diğer bilgisayarların isimlerini öğrenmenizde yarar var.

Diyelim ki, doğrudan eriştiğiniz bilgisayarın adı **temel**; kuruluşunuzda bir de **safnaz** isimli bilgisayar var (UNIX bilgisayarı).

temel isimli bu bilgisayara bilinen yöntemlerle **login** edin.

```
temel login : ayfer
Password :
```

ve

```
temel:/home/ayfer > % ping safinaz
safinaz is alive
temel:/home/ayfer > %
```

komutunu verin.

Aynı işi, safinaz ismi yerine, bu bilgisayarın IP adresini vererek de yapabildiniz. (ping 139.179.40.1).

% ping bilgisayar-adi

Bu komut, bilgisayar ağınızdaki **bilgisayar-adi** adlı bilgisayarın normal çalışıp çalışmadığını ve/veya ağ üzerinden erişilebilir olup olmadığını anlamak için kullanılır.

Eğer

bilgisayar-adi **is alive**

(alive : canlı)

diye bir yanıt alırsanız, diğer bilgisayar ulaşılabilir durumda demektir. Yani o bilgisayar çalışıyor, bilgisayar ağı hizmeti veren programları aktif ve arabirim kabloları veya haberleşme kanalları da sağlam demektir.

Şimdi isterseniz bu bilgisayara **login** etmeyi bir deneyin. (Kendi bilgisayarınızdan **logout** etmeden).

```
temel:/home/ayfer > % rlogin safinaz
Password :
```

% rlogin bilgisayar-adi	<i>(remote)login</i>
--------------------------------	-----------------------

Bu komut, kendi bilgisayarınızın (ya da terminalinizin) başından kalkmadan bilgisayar ağındaki **bilgisayar-adi** adlı bilgisayara **login** etmenizi sağlar.

Bu komutu kullanarak başka bir bilgisayara **login** edebilmeniz için, o diğer bilgisayarın sizi kullanıcı olarak tanıması gerekmektedir. Diğer bilgisayarda da geçerli bir kullanıcı hesabınız yoksa veya şifresini bilmiyorsanız **rlogin** komutuyla bir iş yapamazsınız.

Eğer uzaktaki diğer bilgisayar sizi, bulunduğunuz bilgisayardakinden farklı bir isimle tanıyorsa, **rlogin** komutunda bu isminizi belirtmelisiniz :

```
% rlogin -l oradaki-isminiz -L bilgisayar-adi
Password :
```

Kullanıcı adı ve şifre engellerini aşıp diğer bilgisayara bağlandığınızda artık önünüzdeki ekran ve klavye diğer bilgisayara bağlıymış gibi çalışabilirsiniz. Diğer bilgisayardaki işleriniz bitip de kendi bilgisayarınıza dönmek için **logout** komutunu vermeniz yeter.

```
temel:/home/ayfer > % rlogin -l reyyan -L safinaz
Password : *****
saferaz:/home/reyyan > %
....
...
..
saferaz:/home/reyyan > % logout
temel:/home/ayfer > %
```

saferaz'daki işlerinizi yaptınız....

saferaz'a
bağlanmak için

Geriye, temel'e
dönmek için

Oturduğunuz yerden bir başka bilgisayarın müşterisi olmanın bir yolu daha var.

% telnet bilgisayar-adi

Bu komut, kendi bilgisayarınızın (ya da terminalinizin) başından kalkmadan bilgisayar ağınızdaki **bilgisayar-adi** adlı bilgisayara **login** etmenizi sağlar.

Bu komutu kullanarak başka bir bilgisayara **login** edebilmeniz için, o diğer bilgisayarın sizi kullanıcı olarak tanıması gerekmektedir. Diğer bilgisayarda da geçerli bir kullanıcı hesabınız yoksa veya şifresini bilmiyorsanız **telnet** komutuyla bir iş yapamazsınız.

```
temel:/home/ayfer > % telnet safinaz
Login : reyyan
Password :*****
safinaz:/home/reyyan > %
....
...
..
safinaz:/home/reyyan > % logout
temel:/home/ayfer > %
```

safinaz'a
bağlanmak için

safinaz'daki işlerinizi yaptınız

Geriye, temel'e
dönmek için

Sizin de hemen farketmiş olmanız gerektiği gibi **rlogin** ve **telnet** komutları arasında pek fark yok. **telnet** komutuyla, kim olursanız olun, kullanıcı adınız ve varsa şifre sorulacaktır. Oysa **rlogin** komutunu verdiğinizde, karşıdaki bilgisayar sizi tanıyorsa, bazı durumlarda şifre bile sormadan sizi kullanıcı olarak kabul edecektir.

% rcp [-r] bs1:dosya bs2:dosya

(remote copy)

bs : bilgisayar sözcüğünün kısaltması olarak kullanılmıştır.

Yavaş yavaş bilgisayar ağının nimetlerinden yararlanmaya başlıyoruz....

rcp, bir bilgisayardan başka bir bilgisayara dosya kopyalamak için kullanılır. (Her iki bilgisayarda da dosyaların ve dizinlerin yetki kalıpları önemlidir.)

-r parametresi, tahmin edebileceğiniz gibi dizinleri kopyalamak için kullanılır.

Bir bilgisayardan diğerine, canınızın istediği dosyayı çekememeniz son derece doğaldır. Her sistem yöneticisi, bu şekilde uzaktan gelebilecek kopyalama isteklerinin kendi sistemlerinin güvenliğine zarar vermemesi için gerekli önlemleri almış olacaktır.

% rsh host komut	<i>(remote shell)</i>
------------------	-----------------------

host : UNIX terminolojisiinde standart olarak kullanılan bir terimdir.
 "Ağ üzerinden ulaşılabilen bilgisayarın adı" anlamında kullanılmaktadır.

komut komutunu **host** isimli bilgisayarda çalıştırmak için kullanılır.

Daha önce, **tar** komutunu anlatırken, hava atmak için kullandığım bir örnekte **rsh** komutundan söz etmişim.

% ftp host	<i>(file transfer protocol)</i>
------------	---------------------------------

Biraz önce açıkladığım **rsh** komutunu kullanabilmeniz için, karşıdaki bilgisayarın da UNIX işletim sistemi ile çalışıyor olması gerekmektedir. Oysa bilgisayar ağlarında UNIX'den farklı işletim sistemleriyle çalışan bilgisayarlar da yer almaktadır. Farklı işletim sistemleri altında çalışan bilgisayarlar arasında dosya transferi yapabilmek için, TCP/IP protokolünün bir parçası olan **ftp** isimli bir protokol geliştirilmiştir. UNIX **ftp** komutu, bu protokolu kullanmanızı sağlayan komuttur.

ftp komutunun kullanımı biraz garip olmakla birlikte oldukça kolaydır.

```
temel:/home/ayfer > % ftp safinaz
Connected to safinaz.bilkent.edu.tr
220 safinaz.bilkent.edu.tr FTP server (SunOS 4.1) ready.
Name (safinaz:ayfer):anonymous
331 Guest login ok, send e-mail address as password.
Password: ayfer@ctp.bilkent.edu.tr
230-
230- Welcome to UUNET archive
ftp > binary
ftp > get istenen-dosya-adi
ftp > put gonderilen-dosya-adi
ftp > quit
temel:/home/ayfer > %
```

safinaz'la **ftp**
bağlantısı kurmak için

Bu karman çorman görünen mesaj trafiğinin anlamı şu :

Connected to safinaz.bilkent.edu.tr : **ftp** bağlantınız başarılı

Name (safinaz:ayfer) : Kendinizi tanıttın, kullanıcı ismini boş geçerseniz **ayfer** kabul edilecek. Eğer **safinaz** bilgisayarı sizi (ayfer'i) tanımıyorsa **anonymous** kullanıcı adını deneyin. Bir çok bilgisayar sınırlı erişim hakkı verse de, **anonymous** isimli misafir kullanıcıların **ftp** protokolüyle, sınırlı da olsa, bazı işler yapmasına izin verir. (*anonymous* : **tanınmayan** demektir)

331 Guest login ok, send e-mail address as password.

Password: Misafir kullanıcı olarak kabul edildiniz, şifre olarak **e-mail** adresinizi giriniz. (Şifre olarak **e-mail** adresinizi girmek zorunda değilsiniz ama Internet geleneklerine göre, misafir olduğunuz sisteminin yöneticisinin size ulaşmak istemesi olasılığına karşı, şifre olarak elektronik posta adresinizi girmelisiniz.)

binary komutu, transfer edilecek dosyanın 8 bitlik byte'lardan oluştuğunu belirtmek için verilmiştir. Genel olarak ASCII text dosyası olmayan dosyaların transferinden önce bu komut verilmelidir. Eğer transfer edeceğiniz dosyanın ASCII olup olmadığını bilmiyorsanız, transferi başlatmadan önce bir **binary** komutu verin gitsin. Zararı olmaz.

get istenen-dosya-adi komutuysa, karşınızdaki bilgisayardan (**safinaz**'dan) bu isimdeki dosyayı buraya göndermesini istemek içindir.

put gönderilen-dosya-adi komutuysa, bizim bilgisayarımızdaki dosyayı **safinaz**'a göndermek için verilmiştir.

quit komutuysa **ftp** programını sona erdirir.

ftp komutunun daha bir çok marifeti var tabii, ama bunları zamanla, kullana kullana öğreneceksiniz.

Buraya kadar olan ağ komutları, karşıdaki bir bilgisayarla konuşmanızı sağlayan komutlardı. Bir de, diğer bilgisayarların kullanıcılarıyla konuşmanızı (daha doğrusu görüşmenizi) sağlayan komutlar var.

% finger kullanıcı

Sizin kullandığınız bilgisayarın bir başka kullanıcısına mesaj göndermek istiyorsunuz diyelim. Fakat o kullanıcının sistemdeki kullanıcı adını bilmiyorsunuz. Ne yapmalısınız?

finger komutu yardımcı olabilir. Farzedinki aradığınız kullanıcının gerçek hayattaki adı "Mustafa Arslantunalı".

```
temel:/home/ayfer> finger arslantunalı
Login name: marslan      In real life: Mustafa Arslantunalı
Directory: /home/marslan Shell: /usr/local/bin/csh
Last login Sat May 13 20:35 on ttya
Mail last read Thu May 11 13:30:04 1995
temel:/home/ayfer>
```

Adının ya da soyadının tamamını **finger** komutuna parametre olarak vererek bir kullanıcı hakkında bilgi almanız mümkündür.

Artık, Mustafa'nın sistemdeki adının **marслан** olduğunu öğrendik. Üstelik kendisinin **home** dizinin neresi olduğunu, sisteme en son 13 Kasım da **login** ettiğini, 11 Kasım saat 13:30'dan bu yana mesajlarına bakmadığını öğrendik.

Eğer, Mustafa sistemde çalışıyor olsaydı, kaç saattir çalıştığını da öğrenecektik.

% talk kullanıcı

Eğer Mustafa sistemde çalışıyorsa ve ona söylemek istediğimiz bir şeyler varsa, hazır ikimizde aynı sistemdeyken **talk** komutu yardımıyla bir haberleşme kanalı kurmayı deneyebiliriz.

```
temel:/home/ayfer> talk marслан
```

Bu komutu verir vermez Mustafa'nın ekranında **ayfer** isimli kullanıcının kendisiyle görüşmek istediğini belirten bir mesaj çıkacaktır. Eğer Mustafa görüşme isteğimizi kabul eder ve kendi klavyesinden

```
temel:/home/marслан> talk ayfer
```

komutunu yazarsa karşılıklı olarak ekranlarımız ikiye bölünür ve bizim ekranımızda ekranın üst tarafı Mustafa'ya, alt tarafı da bize (Mustafa'nın ekranında da tam tersi) ayrılarak karşılıklı görüşmemiz (daha doğrusu yazışmamız) başlar.

Bizim Ekranımız

```
[Connected]
```

```
Mustafa'nın yazdığılar
```

```
-----
ayfer'in yazdığılar
```

İkimizden biri **Ctrl-C** tuşuna basıncaya kadar bu görüşme devam eder.

Birisi size **talk** isteği gönderdiğinde ekranınızda, o sırada yapmak olduğunuz işle ilgili görüntü bozulacaktır. Eğer başkalarının size **talk** isteği göndermelerini istemiyorsanız **telefonun fişini çekmeye** benzer bir iş yapabilirsiniz.

% mesg n

Ekranınızı tekrar **talk** isteklerine açmak istediğinizdeyse

```
% mesg y
```

komutunu kullanabilirsiniz.

```
% write kullanıcı
```

talk komutu karşılıklı görüşme sağlar. Eğer haberleşme gereksiniminiz tek yönlüyse, **talk** yerine **write** komutunu kullanabilirsiniz.

Kendi terminalinizden **write kullanıcı** komutunu verdikten sonra istediğiniz mesajı yazıp, mesaj sonuna geldiğinizde satır başına bir **Ctrl-D** tuşu basarsanız, yazdığınız mesaj karşıdaki kullanıcının ekranına gönderilir. Eğer mesaj gönderdiğiniz kullanıcı kendi terminalinden daha önce

```
% mesg n
```

komutunu vermemişse, gönderdiğiniz mesajı aynen görecektir.

```
temel:/home/ayfer> write marslan
Selam,
Ben yemege cikiyorum. Pizaciya gidecegim,
istersen sen de gel.
Ctrl-D
temel:/home/ayfer>
```



talk veya **write** komutları yardımıyla haberleşmek istediğiniz kullanıcı ağ üzerinde **sizinkinden farklı bir bilgisayar üzerindeyse**, komutunuzun parametresi olan **kullanıcı-adi** kısmında, mesajı alacak olan şahsın çalıştığı bilgisayarın adını da içeren bir adres tamamlayıcı kısım olmalıdır.

```
temel:/home/ayfer> write marslan@safinaz
.....
Ctrl-D
temel:/home/ayfer>
```

veya

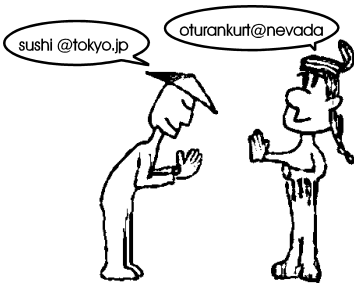
```
temel:/home/ayfer> talk ulker@eagle.cs.umist.edu.uk
(İngiltere'deki bir üniversitede çalışan ulker isimli
kullanıcıya gönderilen talk isteği.)
```

% mail kullanıcı

Haber ulaştırmak istediğiniz kullanıcı o anda bilgisayarının başında değilse veya **"mesg n"** komutuyla **talk** ve **write** kanallarını kapattıysa ne olacak? Haberleşemeyecek misiniz? Elbetteki haberleşebilirsiniz, fakat kullanmanız gereken komut farklı.

mail komutuyla bir elektronik mektup yollayabilirsiniz. Ancak bunu yapabilmek için mektubu göndereceğiniz şahsın **e-mail** adresini bilmek zorundasınız. Aynı normal posta gibi. (Bana PTT aracılığıyla bir mektup gönderebilirmisiniz? Adresimi bilmediğinize göre, elbette hayır. Aynı şekilde elektronik posta göndereceğiniz kişinin de **e-mail** adresini bilmelisiniz.)

Mektup göndermek istediğiniz kişiyle aynı bilgisayarı paylaşıyorsanız işiniz kolay. **e-mail** adresi olarak sadece kullanıcının bilgisayardaki kullanıcı adını belirtebilirsiniz. (Mustafa veya Arslantunalı değil, **marslan** kullanmalısınız).



```
temel:/home/ayfer> mail marslan
Subj : Özel bir soru
Cc :
```

Konu:
Bilgi için:

Sevgili Mustafa
Ne zamandır senden kitabım hakkındaki görüşlerini bekliyorum.
Lütfen bir an önce okuyup önerilerini gönder.
Sevgiler


```
Ugur
Ctrl-D
temel:/home/ayfer>
```

Eğer mektup göndermek istediğiniz şahsın **e-mail** adresi, sizinkinden farklı bir bilgisayardaysa, **mail** komutundaki kullanıcı adını, adresiyle birlikte tam yazmalısınız.

```
temel:/home/ayfer> mail ugur@piper.best.com.tr
temel:/home/ayfer> mail marslan@safinaz
temel:/home/ayfer> mail ulker@eagle.cs.umist.edu.uk
gibi
```

% mail

Parametresiz kullanıldığında dikkatinizi çekerim

mail komutu, parametresiz olarak kullanıldığında size gelen mektuplar hakkında bilgi verir.

Sisteme **login** ettiğinizde

You have new mail

gibi bir mesajla karşılaştıysanız, sisteme son girişinizden bu yana size en az bir yeni mesaj gelmiş demektir.

Eğer

You have mail

gibi bir mesajla karşılaştıysanız, posta kutunuzda en az bir tane eski mesaj var demektir..

Bu mesajları okumak istediğinizde

% mail

komutunu parametresiz olarak vermelisiniz.

```
Mail ver 4 Thu Jan 31 12:54 EST 1995 Type ? for help
"/usr/mail/ayfer":3 messages 2 new
U 1 cil@bilkent      Fri May 12 14:32 23/567 Yeni uygu.
N 2 tayfun@safinaz  Fri May 12 15:34 34/762 Onemli
N 3 kerem@abc       Wed May 23 09:12 45/947 SUNOS4.1
&
```

Yukarıdaki listeye göre posta kutusunda 3 mesaj var. Bunlardan birincisi eski bir mektup ama henüz hiç okunmamış. (Başındaki **U** (*unread*) harfinden anlaşılıyor). 2 ve 3 numaralı mesajlarsa yeni gelmiş (**N** : *new*), dolayısıyla onlar da okunmamış olmalı.

& işareti, **mail** programının hazır işaretidir. Bu işaretin karşısına

- **mail** programının bu düzeyde kabul edeceği komutlar hakkında yardım almak için "?" girebilirsiniz.
- okumak istediğiniz mesajın numarasını girebilirsiniz,
- Programdan çıkmak istediğinizde **q** girebilirsiniz,



Büyük olasılıkla bilgisayarınızda **mail**'den daha güçlü ve kolay kullanılan bir elektronik posta programı vardır. Sistem yönetinizden bu konuda bilgi alabilirsiniz. İçinde bulunduğumuz yıllarda en yaygın **mail** programları

elm (iyi **vi** veya **emacs** kullananlar için) ve
pine (bence en kullanışlı **elektronik posta** programı;
üstelik **vi** veya **emacs** deneyimi de gerektirmiyor).

Her iki program da ücretsiz dağıtılan ürünlerdendir (*freeware*); Internet üzerindeki bir çok bilgisayardan **ftp** ile çekilebilirsiniz.

Elektronik posta ve ftp hizmetlerinden iyi yararlanabilmek için, Internet üzerindeki bilgisayarların organizasyonu hakkında biraz daha detaylı bilgiye gereksiniminiz olacak. Bu nedenle, bundan sonraki bölümü biraz dikkatli okumanızı öneririm.

Internet Organizasyonu

Internet Adresi

Her Internet bilgisayarı bir **domain**'de yer almalıdır. (Bu sözcüğün Türkçe karşılığını bulmaya çalışmadım bile, çünkü artık bu sözcük *Internetçe* oldu. Tüm dillerde aynen kullanılıyor (*domeyn* okunur)). **Domain**, bir bilgisayarın içinde bulunduğu idari bölüme verilen isimdir. Çok sayıda Internet bilgisayarı olan kuruluşlar, bu bilgisayarları bulundukları departmanlara göre **domain**'lere ayırıp hem adreslenmelerini kolaylaştırırlar, hem de sistem yöneticilerinin iş bölümü yapmalarını sağlarlar.

Bir sonraki hiyerarşik düzeyde **domain**'ler gruplanıp daha büyük **domain**'ler oluşturulur. Bu daha büyük **domain**'ler tekrar gruplanıp kuruluşun tipine göre isimler alır; en son olarak da ülkenin **domain**'i tanımlanır. Bir üst düzey zaten Internet'dir.

Birkaç örnek :

Bilkent Üniversitesi'nden örnek vermek istiyorum. Bu Üniversitemiz'de 100'e yakın UNIX iş istasyonu ve 2000'den fazla da PC, MacIntosh gibi küçük bilgisayar var. Bunların hepsi Üniversite'ye yayılmış olan bilgisayar ağına bağlı. Bu ağ da, ODTÜ üzerinden **Internet** dünyasına açılıyor.

Bilkent Üniversitesi'ndeki bu bilgisayarlar bulundukları bölümlere göre **domain**'lere ayrılmış durumda. Örneğin, **temel** isimli bilgisayar Bilgisayar

Programcılığı Meslek Yüksek Okulu'nda; bu bölümdeki diğer bilgisayarlar gibi, **ctp** isimli bir **domain**'de yer alıyor.

Bilgisayar Mühendisliği bölümündeki **hitit** isimli bilgisayarsa **cs domain**'inde.

Bilgisayar Merkezi'ne bağlı bilgisayarların yer aldığı **domain**'e verilen isimse **bcc**.

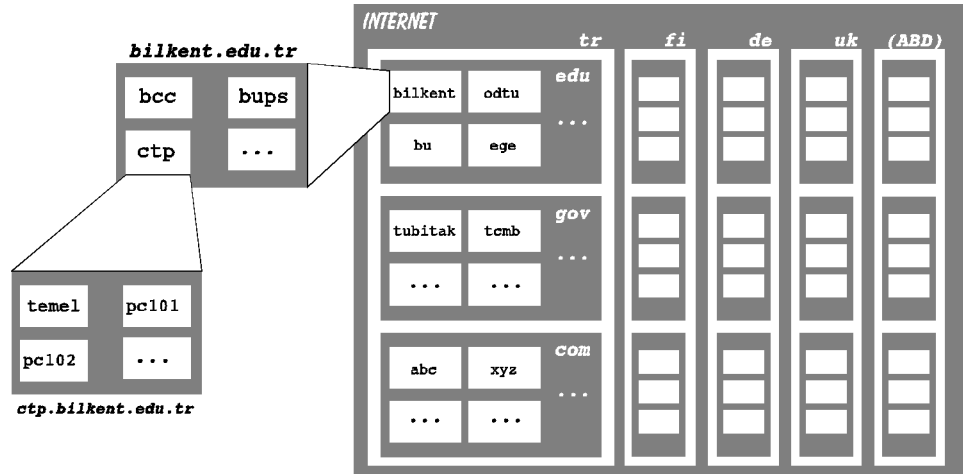
Bilkent Üniversitesi'nde ki tüm **domain**'ler bir araya gelince bir üst düzey olan **bilkent domain**'ini oluşturuyorlar.

ODTÜ, BOĞAZİÇİ gibi üniversiteler de buna benzer bir şekilde yapılanmış durumdalar.

Türkiye'de Internet'de yer alan tüm eğitim kuruluşları **edu** isimli bir **domain** altında toplanıyorlar. (**edu** : Internet terminolojisinde *educational* : eğitim kurumu anlamına geliyor.)

Devlet kuruluşları **gov** (*government*) adlı bir **domain** altında; askeri kuruluşlar **mil** (*military*), ticari kuruluşlar **com** (*commercial*), genel bilgisayar ağı hizmeti verenler **net** (*network*) **domain**'lerinde toplanmış durumdalar.

Tüm bu **domain**'lerse **tr** adlı Türkiye **domain**'inin şemsiyesi altındadır. Aynı yapılanma (edu, gov, mil, com) İrlanda için **ei**, İngiltere için **uk**, Finlandiya için **fi** vs. vs. **domain**'leri altında gerçekleştirilmiş; böylece bütün dünya birbirine bağlanmıştır. (ABD'de bulunan bilgisayarların ülke **domain** kodu yoktur. Başka deyişle, bir **domain** tanımında ülke kodu göremiyorsanız, o bilgisayar grubu ABD'de bir yerlerde demektir.)



Bu yapılanma, bilgisayarların **Internet adresine** ve kullanıcıların **e-mail adreslerine** aynen yansır. Örneğin, Bilkent Üniversitesi'ndeki **temel** adlı bilgisayarın açık Internet adresi

temel.ctp.bilkent.edu.tr dir.

Benim bu **domain**'deki e-mail adresimse

cayfer@ctp.bilkent.edu.tr dır.

Çekinmeden bana **e-mail** gönderebilirsiniz. Okuyuculardan mesaj almak kadar güzel bir şey olabilir mi?

Birbirleriyle aynı **domain**'de tanımlı kullanıcıların, e-mail adresi yazarken, tam adresi yazmalarına gerek yoktur. Örneğin, ben, **ctp domain**'inde birisine mesaj göndereceğim zaman, adres olarak sadece kullanıcı adını kullanırım. Oysa, Bilkent Üniversitesi'nin Bilgisayar Mühendisliği Bölümü'nden birisine mesaj göndereceğim zaman **isim@cs.bilkent** kullanmam gerekir. Boğaziçi Üniversitesi'nden birisine mesaj göndereceğim zaman tam adres yazarım. (**tarhan@ee.boun.edu.tr** gibi)

Internet Sözlüğü

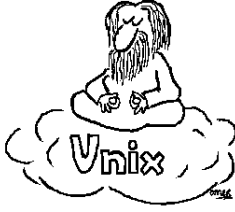
Güncel bir **Internet** sözlüğü hazırlamak olanaksız. Hergün yeni uygulamalar; yeni programlar ve yeni kavramlar geliştiriliyor ve **Internet** kullanıcılarının hizmetine sunuluyor. Internet dünyasına katılan bir kullanıcının sık sık karşılaşacağı bazı önemli terimleri, hiç değilse birazcık, tanıtmak istiyorum.

archie veronica WAIS	İlginizi çeken konularla ilgili olarak, Internet'de, anahtar sözcüklerle arama yapmanızı sağlayan programlara üç örnek. (<i>WAIS : Wide Area Information Search</i>)
whois	e-mail adresleri için uluslararası bir rehber servisi. e-mail adresini bilmediğiniz kişi ve kurumlara ulaşabilmek için yararlanabilirsiniz.
ftp	(<i>File Transfer Protocol</i>) Internet bilgisayarları arasında dosya alışverişini sağlayan protokol ve programı.
gopher	Internet bilgisayarlarındaki herkese açık bilgilerin konulara göre sınıflandırılarak sunulmasını sağlayan ve başka Internet bilgisayarlarına geçişleri sağlayan bir servistir. (WWW çıktığından beri pek kullanılmıyor). Bu servisten yararlanmak için gopher hizmeti veren bir bilgisayara telnet (veya rlogin) ile bağlanıp gopher programını çalıştırmanız gerekir.
news	Yüzlerce çeşit özel ilgi alanı ve hobi için dünyadaki Internet abonelerinin haberleşmesini sağlayan bir servis.

WWW	<i>(World Wide Web)</i> İnternet bilgisayarlarındaki herkese açık bilgilerin konulara göre sınıflandırılarak; grafik menüler aracılığıyla sunulmasını sağlayan ve WWW sunan başka bilgisayarlara geçişleri sağlayan bir servistir. Bu servisten yararlanabilmek için bilgisayarınızın bir IP adresi olmalıdır. Bunu sağlamak için ya bilgisayarınızı doğrudan bilgisayar ağı kablosuna bağlayabilmeli; ya da uzaktan MODEM'le bağlanıyorsanız SLIP veya PPP bağlantısı sağlamanız gerekir.
SLIP ve PPP	<i>(Serial Line Internet Protocol ve Point To Point Protocol)</i> Seri arabirimler kullanarak (genellikle MODEM bağlantılarında anlamlıdır) çeşitli bilgisayarları IP adresi vererek bilgisayar ağına dahil etmek için kullanılan protokoller ve ilgili yazılımlar.
WINSOCK ve TWINSOCK	<i>(Windows Socket)</i> MS Windows altında SLIP ve PPP bağlantısı sağlayabilmek için gereken Windows uygulamaları.
html	<i>(Hyper Text Markup Language)</i> Grafik ve fotoğraf içeren; değişik yazı stillerinde WWW sayfaları hazırlamakta kullanılan özel bir dil.
http	<i>(Hyper Text Transfer Protocol)</i> html ile tanımlanmış sayfaların hızlı bir şekilde bilgisayarlar arasında aktarılmalarını sağlayan veri iletişim protokolu.
mosaic netscape cello	SLIP bağlantısı ile İnternet'e giren bilgisayarların MS Windows veya X-Windows (UNIX için grafik kullanıcı arabirimi) altında WWW servisi veren bilgisayarlardaki sayfaları taramak için kullanılan yazılımlara bir kaç örnek. <i>(WWW Browser).</i>

Bu küçük sözlükteki hizmet ve yazılımlar hakkında daha ayrıntılı bilgi ve özellikle nasıl kullanılacaklarını başka kaynaklara başvurarak öğrenebilirsiniz. Bu yazılım ve kavramlarının herbiri başlı başına bir kitap konusu olabilecek kadar geniş olduğu için, burada yalnızca kısa birer tanımlarını verebiliyorum.

Sistem Yöneticisine...



Sistem yöneticileri, (kısaca **sysadmin**'ler), UNIX işletim sistemi altında çalışan bilgisayar sistemlerinin ayrılmaz parçalarıdır. İdeal olarak, sistem yöneticileri bilgisayar ya da yazılım mühendisliği konusunda formal eğitim görmüş kişiler olmalıdır; ancak genellikle bu görev, kuruluşun en meraklı, sabırlı ve bekar kullanıcısı tarafından üstlenilir. Bekar olmasının önemi çalışma saatlerinin düzensizliğinden kaynaklanmaktadır. Bilgisayar sisteminin tipi, kullanıcıların bu sistemden beklentileri ve belki de en önemli kullanıcıların deneyim düzeyi sistem yöneticisinin iş yükünün en önemli parametreleridir. Deneyimi az ve beklentisi yüksek bir kullanıcı kitlesi karşısında, sistem yöneticileri pek eve gitmeye vakit bulamazlar.

sysadmin'lerin belli başlı görevleri şöyle sıralanabilir .

- Sistemin mümkün olduğunca ayakta kalmasını sağlamak,
- Uygulama ve sistem programlarını bilgisayara yüklemek ve sağlıklı bir şekilde çalışır ve kullanılabilir durumda tutmak,
- Verilerin yedeklemesini yapmak veya koordine etmek,
- Sistemdeki kaynakların verimli bir şekilde kullanılmasını sağlamak,
- Kullanıcıların sisteme ve sistem üzerindeki kaynaklara erişimini düzenlemek (kullanıcı tanıtımlarını yapmak, dosya ve dizinlere erişim yetki kalıplarını düzenlemek),
- Kullanıcıların ve varsa yardımcılarının eğitimi koordine etmek.
- Sistemin kötü niyetli veya acemi kullanıcılara karşı güvenliğini sağlamak (*System Security*)

Bu görevlerin yerine getirilmesiyle ilgili olarak sistem yöneticisi adaylarına bazı önerilerim olacak :

- **Kesintisiz güç kaynağı kullanın.** UNIX bilgisayarların kullanımı sırasında güç kesintisi olması çok ciddi bilgi kayıplarına yol açabilir; tabii, sonunda kabak sistem yöneticisinin başında patlar. Ne tip ve ne çapta bir kesintisiz güç kaynağı kullanılacağı konusunda bilgisayarı satan veya servisini veren kuruluştan yardım isteyebilirsiniz.

- **Yedeklemeyi (backup) kesinlikle ihmal etmeyin.** Yönetiminden sorumlu olduğunuz sistemin toplam disk kapasitesine uygun bir teyp sürücüsünün elinizin altında bulunmasını sağlayın. Hergün veya en az haftada bir disklerinizi teyp kasetlerine yedekleyin.
- **UNIX geleneklerine, bilgisayarınızın üreticisinin tavsiye ve standartlarına uyun.** Disklerinizi, dizinlerinizi gelenek ve önerileri dikkate alarak düzenleyin. Disk ve dizinlerinizin bir haritası sürekli elinizin altında olsun.
- **Sistemde yapacağınız önemli değişiklikleri, yeni öğrendiğiniz konuları, ilk kez yaptığınız işleri vs. yazacağınız bir defteriniz olsun.**
- **Herhangi bir sistem dosyasında değişiklik yapacağınız zaman, ilk önce o dosyanın bir kopyasını çıkarın.**
- **root kullanıcı şifresini iyi koruyun.** Diğer kullanıcıların bu şifreyi öğrenmemesi için elinizden geleni yapın. Şifreyi korumanın en iyi yollarından biri sık sık değiştirmektir. Ama, şifreyi değiştirirken de dikkatli olun; **root** şifresini unutursanız başınız derde girer.
- **Bilgisayarınızın işletim sisteminin teyp kasetinde veya CD üzerinde sağlıklı bir kopyasını kesinlikle bilgisayara yakın bir yerlerde bulundurun.** Ne zaman sıfırdan yükleme yapmanızın gerekeceği kesinlikle belli olmaz. Sıfırdan işletim sistemi yüklemeyi iyi öğrenin. En iyisi, en az bir kez kendi başınıza yapmayı deneyin. Ancak, bu denemeye sırasında başınız derde girdiği takdirde size yardımcı olabilecek birisinin yakınlarda olmasına dikkat edin.
- **Kullanıcılarınızı iyi eğitin.** Sürekli olarak hata düzeltmektense; hataları azaltmak daha kolaydır.
- **Bilgisayarı kapatma törenlerini titizlikle yapın.** Uygun şekilde kapatılmayan UNIX bilgisayarlarının bir daha açılmadığı ya da diski tamamen boşalmış olarak açıldığı oldukça sık rastlanan durumlardır.
- **Sabırlı olun.** UNIX sizi değil; siz UNIX'i yönetmelisiniz.

Murphy kuralları UNIX dünyasında geçerlidir; bunu hiç aklınızdan çıkarmayın.

Bilgisayarın Açılması

UNIX bilgisayarların açılması sırasında dikkat edilmesi gereken pek önemli bir nokta yoktur. Bilgisayarınızın (ya da bilgisayarlarınızın) güç anahtarlarını kapalıdan açık konumuna getirirsiniz; o kadar...

Tüm bilgisayarlar gibi, UNIX bilgisayarları da, güç verilir verilmez ana bellekleri test etmeye başlarlar. Bellek kapasitesine ve test yöntemine göre değişmekle beraber en geç bir kaç dakika sonra sistem diskinin / dizindeki çekirdek UNIX dosyası (*kernel*) belleğe yüklenmeye başlar.

Hemen ardından sistemdeki diskler ve disk bölümleri (*partition*) test edilir. (BSD UNIX için */etc/fstab*; SVR5 için */etc/vfstab* dosyasında belirtilen diskler ve bölümleri).

Testler başarılıysa, bu diskler ve bölümleri otomatik olarak **mount** edilir sistemin açılışıyla ilgili kabuk programları çalıştırılır. (BSD için */etc/rc**; SVR5 için */etc/rc*/**).

Bu kabuk programları, bilgisayar ağıyla, yazıcı yönetimiyle, **mail** trafiğiyle ve kullanıcı terminalleriyle ilgili **daemon**'ları başlatırlar. Herşey normalse, sistem "çok kullanıcılu duruma" (*multi user mode*) geçer ve kullanıcıların **login** etmelerini beklemeye başlar.

Sisteminizin açılışı sırasında, ilk bakışta "kargacık burgacık" diye nitelendireceğiniz mesajlar ekrandan akıp gidecektir. Zamanla bu mesajlar size çok şey ifade etmeye başlayacaktır; merak etmeyin.

Sistemin açılışı sırasında ekrana listelenen bu mesajların birer kopyası */var/adm/messages* dosyasının sonuna da eklenecektir. Ekrandaki mesajları kaçıırırsanız veya sonradan bakmak isterseniz bu dosyaya bir göz atabilirsiniz. Sistemin her açılışında yeni kayıtlar eklenen bu dosya zamanla büyüyecek ve diskte gereksiz olarak yer harcayacaktır. Zaman zaman bu dosyayı küçültmeniz gerekebilir. Bu dosyayı tamamen silmeyiniz. Eğer içindeki kayıtların hiç biri gerekli değilse

```
# rm /var/adm/messages
# touch /var/adm/messages
```

komutlarıyla önce silip; sonra boş olarak tekrar yaratınız.

Eğer bu dosyayı; örneğin son 50 satırını saklayarak kısaltmak istiyorsanız

```
# tail -50 /var/adm/messages > /var/adm/gecici
# rm /var/adm/messages
# mv /var/adm/gecici /var/adm/messages
```

komutlarını kullanabilirsiniz.

Disklerin Test Edilmesi

fsck programı

UNIX dosya sistemi oldukça karmaşık veri yapıları içeren bir sistemdir. Bu karmaşıklığın önemli bir kısmı, hızlı erişim ve esneklik sağlamaya yöneliktir. Bu esneklik ve hızın bedeli de, kolayca karman çorman olabilen bir yapıdır. Özellikle elektrik kesintileri ve bilgisayarın uygun olmayan bir şekilde kapatılması sonucunda, bu dosya sistemi tamamen kullanılmaz hale gelebilmektedir. Bu nedenle, sistem her açıldığında, **mount** edilmeye aday tüm diskler, **fsck** adlı bir program tarafından test edilir. Bu test, sadece disk veya disk bölümü üzerindeki dosya sisteminin bütünlüğünü kontrol etmeye yöneliktir. Disk yüzeyinin taraması yapılmaz.

Eğer **fsck** (*file system check*) programı dosya yapılarında hata bulursa, kendisi onarmaya çalışır; genellikle de başarır. Bazı durumlarda, anlaşılabilir sorularla sizin de fikrinizi sorar. Bu sorulara *yes* anlamında **y** yanıtı vermek dışında pek bir seçeneğiniz yoktur. Bu duruma düştükten sonra tüm sorulara **y** yanıtını verin; bırakın **fsck** bildiği gibi yapsın.

Sistem yöneticisi olarak sizde aklınıza geldikçe **fsck** programını çalıştırarak disklerin durumunu kontrol edebilirsiniz. Bu programın kullanımıyla ilgili olarak önemli bir önerim var : Sistemde başka kullanıcılar çalışırken **fsck** yapmasanız daha iyi olur. Ayrıca, **/** dizini olarak kullanılan disk bölümüyle ilgili olarak her **fsck** çalıştırışınızda bir hata mesajı alırsınız; bu önemli değildir. *yes* deyip geçebilirsiniz.

```
piper# fsck
** /dev/rsd0a
** Currently Mounted on /
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
UNREF FILE I=33797 OWNER=root MODE=100755
SIZE=0 MTIME=Aug 16 17:54 1995
CLEAR? y
```

```
# fsck [ disk ]
```

file system check

fsck komutunu parametresiz kullanırsanız, **/etc/fstab** veya **/etc/vfstab** dosyasındaki tüm disk bölümleri peşpeşe test edilir. Eğer tek bir disk bölümünü test etmek istiyorsanız onun **/dev** dizinindeki adını parametre olarak vermelisiniz.

BSD

```
# fsck /dev/sd0g
# fsck /dev/rsd1h
```

SVR4

```
# fsck /dev/dsk/c0t3d0s4
# fsck /dev/rdsk/c0t3d0s4
```

gibi.

Bu örnekteki `/dev/rsd1h` ve `/dev/rdsk/c0t3d0s4` `/dev` dizin adlarındaki **"r"** (*raw*) harfi, üzerinde henüz dosya yapısı oluşturulmamış ya da öyle kabul edilecek disk bölümlerini belirtmektedir. **mount** edilmemiş olmak kaydıyla, üzerinde dosya yapısı olan disk bölümlerine, *raw device* olarak **fsck** testini uygulamanızın bir sakıncası yoktur.

Sistemin açılışı sırasında otomatik olarak çalıştırılan **fsck**, disklerin birinde bir hata bulursa, hatayı düzeltmeye çalışır, başarılı olursa açılış devam eder. Eğer düzeltme denemesi başarılı olmazsa, sistem, çok kullanıcılu duruma geçmeden tek kullanıcılu durumda kalır (*single user mode*) ve sistem yöneticisinin duruma el koymasını ister. Bu duruma düştüğünüzde, genellikle arkası iyi gelmez. **fsck** programını bir kez de sizin çalıştırmanızı ve mesajları dikkatle izleyerek hatanın nedenini anlamaya çalışmanızı ve **fsck** programını duruma uygun seçeneklerle tekrar çalıştırmanızı öneririm. **fsck** programının seçeneklerinden burada söz etmek istemiyorum. Kullandığınız UNIX'e ait **man** sayfaları ve sistem referans kitapları bu durumda en iyi yardımcılarınız olacaktır.

Hatayı düzeltebilerseniz veya ne pahasına olursa olsun devam etmek isterseniz Ctrl-D tuşuna basarak çok kullanıcılu duruma geçerek açılışa devam edebilirsiniz.

Açılışını normal olarak tamamlayabilen bir UNIX bilgisayarının konsolunda ve açık olan tüm terminallerinde

Log in :

belirecektir.

Bilgisayarın Kapatılması

Daha önce bir kaç kez UNIX bilgisayarlarının kapatılmasına ilişkin **törensel** işlemlerden söz etmiştim. Şimdi biraz ayrıntıya girelim.

BSD ve SVR5 UNIX'lerde bu töreni başlatma komutları farklıdır; ama her iki tip UNIX'de de ortak olan ve dikkat edilmesi gereken noktalar vardır.

UNIX çok kullanıcılu bir çalışma düzeni sağladığı için, sizin işinizi bitirmiş olmanız, bilgisayarı rahatça kapatabileceğiniz anlamına gelmez. Bir başka terminaldeki veya bilgisayar ağı üzerindeki bir kullanıcı önemli işler yapıyor olabilir. Eğer sistemi bir nedenle acil olarak kapatmanız gerekmiyorsa

```
# wall "Sistem 1 saat sonra bakım için kapatılacaktır..."
```

gibilerinden bir *write-to-all* (tüm kullanıcılara duyuru) komutuyla, önce durumu herkese duyurmalısınız.

Eğer BSD UNIX kullanıyorsanız bu duyuruyu gönderdikten sonra

BSD

```
# shutdown -h 13:00
```

komutuyla sistemin saat 13:00 da kapanmaya başlamasını sağlayabilirsiniz.

Sistemin hemen kapanması gerekiyorsa

```
# shutdown -h now
```

komutunu kullanabilirsiniz.

Sistemin kapanma saati geldiğinde, tüm *daemon'lar* sırayla ölmeye başlayacak ve en sonunda artık sistemin elektriğini kesebileceğinize ilişkin bir mesaj gelecektir. Bu noktaya kadar sabırla beklemelisiniz. Eğer **shutdown** komutunu verir vermez sistemi kapatırsanız, komutunuz hiç bir işe yaramaz. Kapatma süresi ve kapanış sırasında ekrana gelecek mesajlar bilgisayarınızın marka ve modeline göre değişebilir; sabırlı ve dikkatli olunuz.

Kapatma işleminin başlamasıyla birlikte kullanıcılara ait tüm süreçlerin UNIX tarafından öldürüleceğini ve bunun kullanıcı veri dosyalarına zarar verebileceğini unutmamalısınız. (Nasılsa zararları onarmak da size düşecek). O nedenle kapatma komutunu vermeden önce

```
# who
```

komutuyla, sistemde sizden başka çalışan olup olmadığını kontrol etmenizi; varsa

```
# write inatci  Lutfen logout edin!  
      veya  
# mail inatci  
      Lutfen logout edin!  
Ctrl-D  
#
```

gibi komutlarla o kullanıcıların kendiliklerinden işlerini bitirip **logout** etmelerini sağlayınız.



Eğer SVR5 UNIX kullanıyorsanız, diğer kullanıcılarla ilgili kontrol (**who**) ve duyuru (**wall**) komutlarını aynen kullandıktan sonra, **init daemon'unu** öldürmeniz gerekir. Bu işi yapmak için verebileceğiniz iki komut var

```
# shutdown
# /etc/telinit 5
```

Hangisini isterseniz kullanabilirsiniz.

init sürecini öldürecek komutu vermenizden yaklaşık 10 saniye sonra tüm *daemon'lar* sırayla ölmeye başlayacak ve en sonunda artık sistemin elektriğini kesebileceğinize ilişkin bir mesaj gelecektir. Bu noktaya kadar sabırla beklemelisiniz. Eğer komutu verir vermez sistemi kapatırsanız, komutunuz hiç bir işe yaramaz. Kapatma süresi ve kapanış sırasında ekrana gelecek mesajlar bilgisayarınızın marka ve modeline göre değişebilir; dikkatli olunuz.

Yaklaşık 20 yıllık bilgisayar deneyimim, hiç kapatılmayan bilgisayarların, sık sık kapatılanlara göre çok daha az donanım arızası verdikleri doğrultusunda. O nedenle, gerekmedikçe bilgisayarınızı kapatmamanızı öneririm (hafta sonları bile açık bırakmayı düşünebilirsiniz). Artık bilgisayarlar eskisi gibi çok enerji harcamıyorlar. Yanlızca, tüplerindeki fosforun yanmaması için ekranları kapatmanızı veya belirli bir süre klavyeye dokunulmadığında ekranı karartan yazılımları (*screensaver*) kullanmanızı öneririm.

Kullanıcı Hesapları

User Accounts

Herhangi bir kimsenin, bir UNIX bilgisayarını kullanabilmesi için sisteme **login** edebilmesi; bunun için de geçerli bir kullanıcı hesabına sahip olması (*user account*) gerekir. (Banka hesabı gibi düşünebilirsiniz.)

Kullanıcı hesaplarının açılması, kapatılması ve zaman zaman değiştirilmesi sistem yöneticisinin en önemli görevlerindendir. Yeni bir kullanıcıyı sisteme tanıtmak için

1. Kullanıcının gerçek kimliğini çağrıştıran 5 ila 8 harfli bir isim seçmelisiniz. Örneğin, kullanıcı adının ilk harfi ve bitişik olarak soyadını kullanabilirsiniz.

maslan	Mustafa Aslantunalı
rayfer	Reyyan Ayfer
cayfer	Can Ugur Ayfer gibi.

Kullanıcı adı içinde boşluk kullanamazsınız, eğer bir ayırıcı karakter gerekiyorsa "-" işaretini kullanabilirsiniz (**c-ayfer** gibi).

2. Kullanıcınız için bir numara seçmelisiniz. Bu numaranın daha önce başka bir kullanıcıya verilmemiş olması önemlidir. Eğer iki kullanıcıya aynı numarayı vererseniz, UNIX açısından bu iki

kullanıcı da aynı kimliğe sahip olur. Kullanacağınız numaraların sıralı olması gerekmez; o nedenle kullanıcı tiplerini belirlemek üzere bu numaraları gruplayabilirsiniz. (200'le başlayan kullanıcı numaraları Muhasebe bölümüne; 300'le başlayan Personel Müdürlüğü çalışanlarına gibi).

3. Sisteme tanıtacağınız yeni kullanıcının hangi kullanıcı grubuna ait olacağına karar veriniz. Bu gruplandırma dosya ve dizin erişim yetkilerini düzenlemek için kullanılacaktır. Kullanıcılarınızı gruplara ayırırsanız, dosya ve dizin erişim yetkilerini düzenlemek kolaylaşır.

Örneğin, "personel müdürlüğü" grubuna ait kullanıcıların, bu müdürlüğe ait dosyalara erişmesini sağlamak; fakat bu grupta olmayan kullanıcıların personel kayıtlarını görmesini önlemek çok kolay olacaktır.

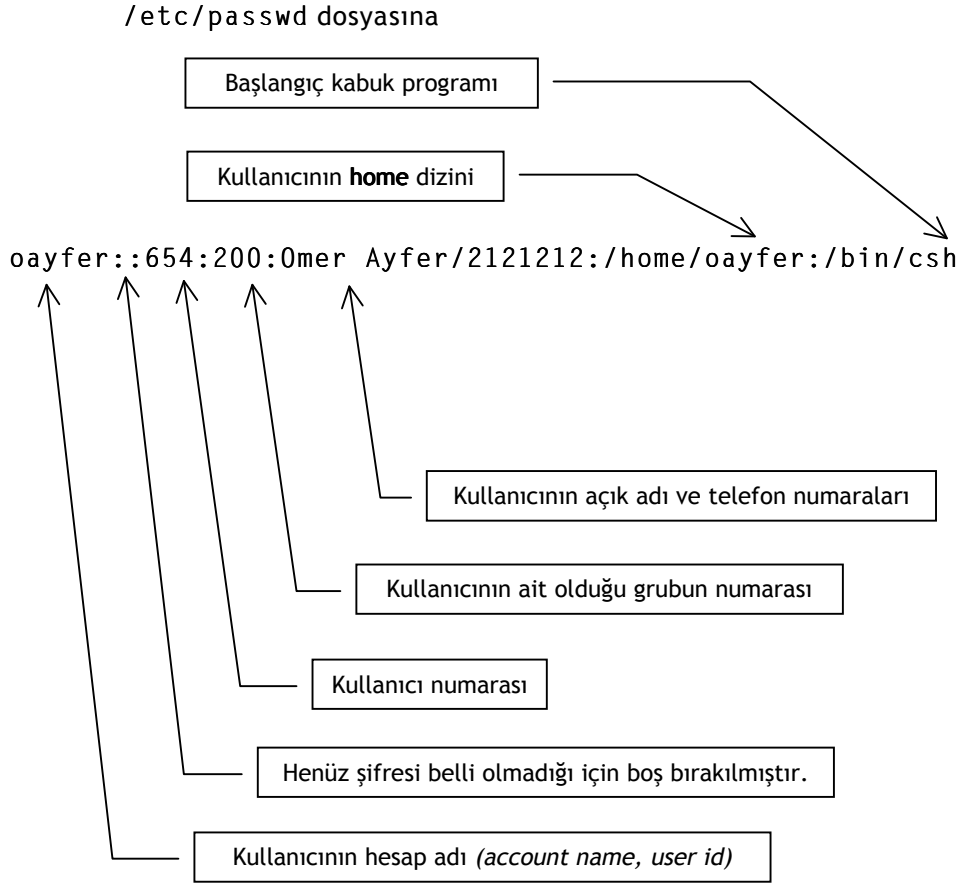
Bir üniversitede, öğrenci kayıtları dosyasına okuma-yazma yetkisi olan kullanıcıların hepsi 230 numaralı grupta; Personel dairesinde çalışan kullanıcılar 240 numaralı grupta yer alabilir. Bu yöntemle, öğrenci kayıt dosyasının sahibi olan kullanıcının grup numarası 230; ve dosyanın erişim yetki kalıbı **rw-rw----** ise, grup numarası 240 olan kullanıcılar bu dosyaya hiç erişemezken, grup numarası 230 olanlar dosya üzerinde okuma-yazma yetkisine sahip olacaktır.

4. Kullanıcınızın **home** dizininin hangi diskin, hangi dizininde ve hangi isimle yer alacağına karar vermelisiniz. **/home/kullanici-adi** geleneklere uygun bir seçimdir.

5. Kullanıcınızın sisteme **login** ettiğinde hangi kabuk programıyla işe başlayacağını belirlemelisiniz. (Kendisi isterse sonra başka bir kabuk programı başlatabilir). (**/bin/sh**, **/bin/csh** gibi)
6. Yukarıda kararlaştırılan parametreleri içeren bir satırı **/etc/passwd** dosyasına eklemelisiniz.
7. Kullanıcının **home** dizinini yaratıp, bu yeni dizinin içine standart **.login**, **.logout** ve **.cshrc** (kullanıcınız **csh** kabuğu kullanacaksa) dosyalarını kopyalamalısınız. (Kullanıcınız X-Windows kullanacaksa **.twmrc** **.xinitrc** gibi standart dosyaları da unutmamalısınız).
8. Bu yeni yarattığınız **home/oayfer** dizininin ve altına kopyaladığınız dosyaların sahibini yeni kullanıcınız olarak değiştirmelisiniz. (Eğer bu "sahip değiştirme" işini yapmayı unutursanız; dizini ve dosyaları **root** kullanıcı yarattığı için bunların sahibi **root** kalır. Bu yüzden, kullanıcı **home** dizini altında dosya yaratamaz ve **.login**, **.cshrc** gibi dosyalar üzerinde kendi kişisel tercihleri doğrultusunda değişiklik yapamaz.)

Şimdi, örnek olarak **oayfer** (Ömer Ayfer) isimli bir kullanıcının sisteme tanıtılma işini birlikte yapalım. Ömer Bey için

kullanıcı adı olarak **oayfer**,
kullanıcı numarası olarak **654**,
ait olduğu grup numarası olarak **200**,
home dizini olarak **/home/oayfer**,
kabuk programı olaraksa **csh** seçmiş olalım.



görünümünde bir satır eklemeliyiz.

Daha sonra, kullanıcının **home** dizinini yaratmak için :

```
# mkdir /home/oayfer
```

Standart **.login**, **.logout** ve **.cshrc** dosyalarını bu yeni **home** dizinine kopyalamak için :

```
# cp /etc/login.std /home/oayfer/.login
# cp /etc/logout.std /home/oayfer/.logout
# cp /etc/cshrc.std /home/oayfer/.cshrc
```

Son olarak da bu dizin ve dosyaların sahibini değiştirmek için :

```
# chown oayfer /home/oayfer /home/oayfer/.1* /home/oayfer/.c*
```

Kullanıcımız sisteme giripte, kendi şifresini tanıttınca **/etc/passwd** satırı

```
oayfer:3e*&TH23<:654:200:0mer Ayfer/2121212:/home/oayfer:/bin/csh
```



gibi bir şekle dönüşecektir. Bu satırdaki **3e*&TH23<** dizisi kullanıcının şifresinin **kriptolanmış** şeklidir. Bu kriptoyu çözmek mümkün değildir; bu nedenle şifresini unutan kullanıcıların unuttukları şifrenin ne olduğunu hiç bir şekilde bulamazsınız. Böyle durumlarda yapabileceğiniz iki şey var :

Birincisi, **/etc/passwd** dosyasının kullanıcıyla ilgili satırındaki şifre bilgi sahasını boşaltarak kullanıcın şifresini tamamen iptal etmek;

Diğeriye, **root** yetkilerinizden yararlanarak

```
# passwd oayfer
```

komutuyla bu kullanıcıya yeni bir şifre verip bu şifrenin ne olduğunu kendisine söylemeniz. (**root** kullanıcı olarak **passwd** komutuyla başka bir kullanıcının şifresini değiştirmek istediğinizde, sizden eski şifreyi girmeniz istenmez.)

Kullandığınız bilgisayarda, kullanıcı tanıtımı, iptali gibi işler için bir takım hazır programlar olma olasılığı çok yüksektir. Sisteminizin Sistem Yönetimi ile ilgili kitaplarında bu tip programların olup olmadığını; varsa bu programların nasıl kullanılacağını bulabilirsiniz. Örneğin SUN OS 4.x.x işletim sisteminde **/usr/etc/install/add_user**; SUN OS 5.x.x işletim sisteminde **/bin/admintool** gibi yazılımlar kullanıcı tanıma işlemlerini biraz kolaylaştırmaktadır.

Bazı UNIX'lerde kullanıcı şifreleri **/etc/passwd** dosyasında değil, **/etc/shadow** isimli bir dosyada saklanır. Bazı bilgisayar ağlarında **NIS** (*Network Information Services*) adlı bir servisten yararlanarak, ağdaki tüm bilgisayarların kullanıcı tanımları ve şifreleri tek bir bilgisayarda tutulur. Bu gibi özel durumlarda kullanıcı tanıma yöntemlerinde sizin sisteminize özgü farklılıklar olabileceğine dikkatinizi çekerim.

Kullanıcılarla Haberleşme

Sistem yöneticisi olarak, zaman zaman, kullanıcılarınıza duyurular yapmanız gerekecektir. Eğer duyurunuz bir ya da birkaç kullanıcıyı ilgilendiriyorsa, o kullanıcılara birer **e-mail** göndermek doğal olarak en kolay ve en elverişlisi. Ancak duyurunuz tüm kullanıcıları ilgilendiriyorsa daha kullanışlı bir yöntem bulmalısınız. Belki de ilk aklınıza gelen **wall** (*write to all*) komutu çok yanlış bir seçim olacaktır. Sebebi basit! Bu **wall** komutu, sadece kullanıldığı anda sistemde çalışmakta olan kullanıcıların ekranına mesaj gönderir. Bir başka yöntem olarak aklınıza gelebilecek olan "tüm kullanıcılara birer **e-mail** göndermek" pek iyi bir fikir değil. Mail listesi dosyanıza eklemeyi unuttuğunuz kullanıcılar olabilir. Ayrıca giden mesajın birer kopyası kullanıcıların posta

kutusu dosyalarında yer işgal edecektir). Tüm kullanıcılara duyuru yapmanın en kolay ve kullanışlı yöntemi **/etc/motd** dosyasını kullanmaktır.

/etc/motd (Günün mesajı)

(Message Of The Day)

Bu dosyaya yazacağınız her türlü duyuru, mesaj; sisteme **login** eden tüm kullanıcıların ekranına görüntülenecektir. Falanca programın falanca isimle hizmete girdiğini; Sistemin falanca gün bakım için kapatılacağını kullanıcılara duyurmanın en kolay yolu, bu mesajlarınızı **/etc/motd** dosyasına yazmak olacaktır. Prensip olarak, bu dosyaya 20 satırdan uzun mesaj yazmayınız (uzun mesajları kimse sonuna kadar okumayacaktır); güncelliğini kaybeden duyuruları bu dosyadan çıkarınız.

% wall

(Write to All)

Zaman zaman sistemi kapatmanız gerekecektir. Özellikle bilgisayarın donanımıyla ilgili bir çalışma yapılacaksa sistemi kapatmalısınız. Bazı durumlarda da kontrolden çıkan programları bir türlü öldüremediğiniz için sistemi kapatıp hemen tekrar açmak ihtiyacı duyabilirsiniz. Kullanıcılara haber vermeden **shutdown** komutunu verip sistemi durdurmak pek kibar bir davranış olmayacaktır. Sistemde çalışan kullanıcıların tümüne birden **wall** komutuyla mesaj gönderebilirsiniz.

```
# wall
```

```
Sistem 10 içinde kapatılacaktır.  
Lutfen islerinizi bitirip cikiniz.
```

```
^D  
#
```

Kullanıcı Terminallerinin Denetimi

UNIX işletim sistemiyle çalışan bilgisayarlarda, kullanıcı terminalleri genellikle ana sisteme seri arabirimler üzerinden (RS232, RS422 gibi arabirimler) bağlanır. Son yıllarda bu yöntem yerini hızla Ethernet ağı üzerinden bağlanan terminallere terketmektedir ama gene de seri arabirim yönetimi, sistem yöneticilerinin önemli görevleri arasındadır.

Seri arabirim denetimleri **/etc/ttytab** (BSD UNIX) veya **/etc/gettytab** (SVR4 UNIX) dosyaları kullanılarak yapılır. (Sisteminiz SVR4'se, seri arabirim denetimi çok farklı olabilir; bilgisayarınızın referans kitaplarına başvurunuz).

Bu dosyalar üzerinde yapacağınız değişikliklerle

- Seri arabirimlere bağlı terminallerin hizmete açılıp açılmamasını,
- Terminallerin (veya bu arabirimlere bağlı yazıcıların) veri transfer özelliklerini

denetleyebilirsiniz. Bu hatlara MODEM bağlı olup olmadığını da bu dosyalarda vereceğiniz parametrelerle belirtirsiniz. Bu dosyaların kayıt desenleri, kayıtlarda yer alacak parametrelerin anlamlarını öğrenmek için lütfen UNIX'inizin kitaplarına başvurunuz.

Benim kullandığım BSD UNIX'deki **/etc/ttytab** dosyasının görünümü şöyle :

```
# cat /etc/ttytab
#
# @(#)ttytab 1.7 92/06/23 SMI
#
# name      getty                type      status    comments
#
console    "/usr/etc/getty cons8"    sun        on        local secure
ttya       "/usr/etc/getty D38400"   vt100      on        remote
ttyb       "/usr/etc/getty D38400"   vt100      on        remote
tty00      "/usr/etc/getty std.9600"        unknown   off       local secure
tty01      "/usr/etc/getty std.9600"        unknown   off       local secure
tty02      "/usr/etc/getty std.9600"        unknown   off       local secure
```

Bu dosyadaki bazı parametrelerin anlamları şöyle :

- Konsolda (ana ekran) **login** sürecini denetlemek için **/usr/etc/getty** programı çalıştırılmakta.
- Konsol "**sun**" tipi bir ekran (çünkü bilgisayar SUN).
- **status=on**; yani terminal login'lere açık.
- **local** : bağlantı yerel bir bağlantı, MODEM kullanımı söz konusu değil.
- **secure** : Terminalin bulunduğu yer emniyetli, bu nedenle bu terminalden **root** olarak **login** edilebilir. (Bu sözcüğün bulunmadığı terminallerden **root** kullanıcı adını kullanarak **login** edemezsiniz. Böyle bir durumda önce sıradan bir kullanıcı olarak **login** etmeniz; sonra "**su**" komutuyla **root** olmanız gerekecektir.)

ttya arabirimi (sistemin ilk seri arabirimi) de login'lere açık; ancak bu sefer bağlantıda MODEM protokolleri kullanılıyor ve veri iletişim hızı 38400 baud. Bu uçlara bağlı terminaller **vt100** gibi davranabiliyor; ve bu hattan **root** login yapılmasına izin verilmiyor.

```
% stty
```

```
(set tty)
```

Sistem yöneticisi olarak, yeni bir kullanıcı tanıttığınızda, bu şahıstan duyacağınız ilk şikayet şudur :

“Klavyem bozuk! Backspace tuşu ve ok tuşları çalışmıyor!”

Kullanıcı haklı olabilir tabii; ama bu sorunun nedeni yüzde doksan dokuz terminal arabiriminin bir parametresinde yatmaktadır.

Backspace tuşunun çalışması için, kullanıcı **login** ettikten sonra

```
% stty erase ^H
```

komutunu giriniz. Bu backspace tuşunun yazılmış karakterleri silmesini sağlarsa, bu komutu kullanıcının **.login** dosyasına ekleyiniz.

Benzeri bir iş yapan

```
% stty werase ^W
```

komutu da **Ctrl-W** tuşlarına basıldığında son yazılmış sözcüğün silinmesini sağlayacaktır.

Ok tuşlarının görev yapması içinse;

csH için

```
% stty dec
```

```
% setenv TERM vt100
```

TERM değişkeni, kullanılan terminalin marka ve modeline göre seçilmelidir. En yaygın terminal tipi DEC marka VT100 modeli veya bununla eşdeğer terminaller olduğu için TERM=vt100 genellikle uygun olmaktadır.

sh için

```
$ stty dec
```

```
$ set TERM=vt100
```

```
$ export TERM
```

komutlarını deneyiniz. Sorun hallolursa, bu komutları **.login** veya **.cshrc** dosyasına ekleyiniz.

Yazıcıların Tanıtımı

Sistem yöneticisinin önemli görevlerinden birisi de yazıcıların kullanıcılar arasında kolaylıkla paylaşılmasını sağlamaktır. Büyük sistemlerde genellikle birden fazla ve farklı özelliklere sahip yazıcı bulunur. Bu yazıcıları kurmak ve tanıtımlarını yapmak sizin göreviniz olacaktır.

BSD ve SVR4 UNIX'lerde yazıcı tanıtımları çok farklıdır. **SVR4** UNIX'lerde bu işlemler genellikle **admintool** veya benzeri bir destek programıyla yapılır. **BSD** UNIX'lerdeyse, **/etc/printcap** dosyası üzerinde yapılacak değişikliklerle yazıcılarınızı denetleyebilirsiniz.

Ben, bu kitapta sadece BSD UNIX'lerde kullanılan **/etc/printcap** dosyalarına, SUN tarafından geliştirilmiş olan SUN OS 4.1.3 UNIX'den bir örnek vermekle yetineceğim.

Seri arabirimle bağlanan bir yazıcıyı sisteme tanıtmak için **/etc/printcap** dosyasına yerleştirilmesi gereken satırlara örnek :

```
lp|prt|Seri Yazıcı:\
:br#9600:if=/usr/bin/lpf:lf=/usr/adm/lpd-errs:\
:lp=/dev/ttya:sd=/usr/spool/printer:\
:ms=cs8,-parenb:sh:sf:xc#8:xs#1:
```

Paralel arabirim ile bağlanacak bir yazıcı için :

```
lp|prt|Paralel Yazıcı:\
:lp=/dev/bpp0:sd=/usr/spool/printer:sh:\
:lf=/dev/usr/adm/lpd-errs:
```

Bilgisayar ağı üzerinden, bir başka bilgisayara bağlı olan yazıcıyı kullanabilmek için (**kaya** isimli bilgisayardaki **matrix** isimli yazıcıyı) :

```
lp|prt|Uzak Yazıcı:\
:lp=:rm=kaya:rp=matrix:sd=/usr/spool/printer:sh:\
:lf=/dev/usr/adm/lpd-errs:
```

Bu satırlarda gizlenmiş olan veriler şunlardır :

<code>lp prt:</code>	Yazıcının, lp ya da prt adıyla kullanılabileceğini;
<code>lp=/dev/ttya:</code>	yazıcının birinci seri arabirime (ttya) bağlı olduğunu;
<code>br#9600</code>	Seri arabirim için seçilen haberleşme hızının 9600 baud olduğunu;
<code>sd=/usr/..</code>	Bu yazıcıya gönderilen dökümlerin hangi dizinde sıraya konulacağını;
<code>lf=/usr/adm/..</code>	Bu yazıcıyla ilgili hata mesajlarının hangi dosyada biriktirileceğini;
<code>if=/usr/bin/..</code>	Bu gönderilecek dosyaların hangi filtre programından geçirileceğini;
<code>sf</code>	Kullanıcılara ait dökümlerin arasına bir başlık sayfası konmayacağını (<i>suppress header page</i>) belirtmektedir.
<code>rm=</code>	Diğer bilgisayar (Remote Machine)
<code>rp=</code>	Diğer bilgisayardaki yazıcının adı. (<i>Remote Printer</i>)

/etc/printcap dosyasında yer alabilecek daha onlarca parametre var. Bunların neler olduğunu merak ediyorsanız, lütfen **man printcap** komutunu kullanınız (Yalnızca BSD UNIX için söz konusudur.)

Sistemin Açılışının Kontrolü

(rc* dosyaları)

Bildiğiniz gibi, MS-DOS işletim sistemiyle çalışan kişisel bilgisayarlarda, çeşitli donanım ve yazılım gerekesinimleri doğrultusunda, açılış sırasında belleğe yüklenmesi gereken yazılımlar ve bunların parametreleri CONFIG.SYS ve AUTOEXEC.BAT dosyalarıyla düzenlenmektedir.

UNIX'de durum biraz farklı olmakla beraber mantık aynıdır. Sistemin açılış sırasında, çeşitli parametreleri düzenlemek amacıyla; ya da sistemdeki bazı özel donanımları sürece programların belleğe yüklenmesi için gerekli komutlar **/etc** dizininde **rc** harfleriyle başlayan bir takım dosyalara yerleştirilir. UNIX'in BSD ya da SVR4 oluşuna göre farklılık göstermesine rağmen, bu dizindeki **rc kabuk programı dosyaları** belirli bir sırayla çalıştırılarak, yapılması gereken kontroller ve yüklemeler gerçekleştirilir.

BSD UNIX'den bir kaç örnek vermek gerekirse;

/etc/rc.boot UNIX'in çekirdek programının yüklenmesinden sonra çalıştırılan kabuk programını içerir. Bu dosya üzerinde, genellikle bir değişiklik yapmanız gerekmez.

/etc/rc **rc.boot** dosyasındaki komutların yerine getirilmesi tamamlanınca çalıştırılır. Bu dosyada da genellikle değişiklik gerekmez.

/etc/rc.local Sistem, tüm testlerini tamamladıktan ve tek kullanıcı düzeyden çok kullanıcı düzeyine geçerken çalıştırılır. Sistem yöneticilerinin yapacakları değişiklikler genellikle bu dosya üzerinde olacaktır.

rc dosyalarında değişiklik yapabilmek için kabuk programlama (sh ve csh) konusunda deneyimli olmalısınız. Sisteminizin açılışıyla ilgili problemler varsa, bu dosyalarla oynayarak düzeltebileceğinizi sanıyorsanız yanılıyorsunuz. Ne yaptığınızı iyi bilmeden bu dosyalarda değişiklik yapmayınız.

Disklerin, Açılıştan Otomatik mount Edilmesi (/etc/fstab dosyası)

Bilgisayarın açılışı sırasında bazı disk bölümleri (*partition*); özellikle **/** ve **/etc** dizinlerinin bulunduğu disk bölümü veya bölümleri UNIX tarafından otomatik olarak **mount** edilir. Eğer bu **mount** işlemi bir nedenle başarısız olursa, bilgisayarınız açılmayacaktır.

Bilgisayarınızdaki diğer diskler ve bölümlerin (sistemin açılabilmesi için gerekli olmayanlar) sistem yöneticisinin kontrolüne bırakılmıştır. Sistem yöneticisi bu kontrolleri

/etc/fstab (BSD UNIX) veya
/etc/vfstab (SVR4 UNIX)

dosyalarını düzenleyerek yaprlar.

Bu dosyalara birer örnek vermek gerekirse :

/dev/sd0a	/	4.2	rw	1 1
/dev/sd0g	/usr	4.2	rw	1 2
/dev/sd0h	/home	4.2	rw	1 3
/dev/sr0	/cdrom	hsfs	ro,noauto	0 0
piper:/disk2	/pdisk2	nfs	rw	0 0

BSD

Yukardaki örneğe göre, ilk diskin (sd0) **a** bölümü sistemin **/** dizini olarak **mount** edilecek. Söz konusu disk bölümü üzerindeki dosya yapısının sürüm numarası (*version*) 4.2 olarak tanımlanmış. Bölüm, oku-yaz (*rw*) olarak **mount** edilecek (*rw*). **fsck** komutu parametresiz olarak kullanıldığında bir kez ve ilk sırada kontrol edilecek. (1 1).

İlk diskin **g** bölümüyse, **/usr** dizini olarak **mount** edilecek (gene 4.2 ve *rw*) ve **fsck** tarafından ikinci sırada ve gene bir kez kontrol edilecek (1, 2).

Bilgisayarın **/dev/sr0** sürücüsü (SUN marka iş istasyonlarında CDROM sürücüsüdür) **/cdrom** dizinine **mount** edilebilecek ancak bu **mount** işlemi sistemin açılışıyla hemen otomatik olarak yapılmayacak (*no auto*).

Peki, CDROM sürücüsünün mount işlemi otomatik olarak yapılmayacaksa, bu satırın bu dosyada ne işi var?

Şu işi var :



Bu satırın **/etc/fstab** dosyasında olması sayesinde **root** kullanıcı (BSD UNIX'de **mount** işlemini sadece root yapabilir) CDROM **mount** etmeye gerek duyduğu zaman, klavyeden

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

komutunu uzun uzun yazmak yerine, yalnızca

```
# mount /cdrom            yazabilecek.
```



```
# cat > /usr/local/bin/cdumount          CD umount etmek için
#!/bin/sh
umount /cdrom
^D
#
# chmod 4755 /usr/local/bin/cdumount
```

Yedekleme



En önemli göreviniz...

Sisteminizin operatörleri olabilir; yedekleme işini onlar yapıyor olabilir; ama bu yedekleme işlerini düzenlemek yalnızca sizin sorumluluğunuzdadır. Kullanıcılar ve işi bilgisayarın düzgün çalışmasına bağlı olan kimseler, kaybolan bilgilerin hesabını sizden sorarlar; üstelik güç kesintisi, disk arızası gibi mazeretleri de kabul etmezler.

Yedekleme, hangi tip bilgisayar kullanıyor olursanız olun çok önemlidir. Bu konuda sanırım aynı fikirdeyiz. Yedekleme UNIX'de daha da önemlidir. Çünkü, UNIX dosya yapısı güç kesilmelerine ve yanlışlıkla dosya silmelere karşı çok hassastır. Örneğin, bilgisayarınızın **/etc** dizinini bir silerseniz (sisteme **root** olarak **login** etmiş birisi için bu, yapılması son derece kolay olan bir hatadır) o sistemi bir daha kolay kolay açamazsınız; çünkü UNIX'de UNDELETE yoktur. Bu hatanın yapıldığı bir UNIX bilgisayarını tekrar ayağa kaldırmanın genellikle en kolay yolu, işletim sistemini tekrar yüklemektir. Bu da yarı yarıya sistem disklerini formatlamak demektir. Eğer sistem dosyalarınızın (özellikle **/etc** dizinindeki) yedekleri yoksa sistemi eski haline getirmek pek kolay olmayacaktır. Tüm kullanıcıları yeniden tanıtmanız gerekecek, herkesin şifreleri kaybolacaktır. Yazıcı tanımlarını, bilgisayar ağı bağlantılarıyla ilgili tanımları, **rc** kabuk programlarında yapmış olduğunuz değişiklikleri; hepsini baştan yapmak zorunda kalacaksınız. Bütün bu düzenlemeleri yeniden yapmak için uzun bir zamana ve sağlam sinirlere sahip olmanız gerekecektir.

Ama çaresi var! Yedekleme yapın; hem de sık sık.

Disk kapasitelerinin büyüklüğü ve yedekleme programlarının özelliklerinden dolayı, UNIX altında çalışan bilgisayarlarda yedekleme genellikle teyple yapılır. Eğer, yönetiminden sorumlu olduğunuz bilgisayarın bir teyp yedekleme birimi yoksa hemen bugün; evet bugün; bir tane satın almak için gerekli işlemleri başlatın. Emin olun, sistemin göçmesi durumunda maddi kaybınız bir teyp fiyatının çok üstünde olabilir. Neyse, teyp biriminiz olduğunu var sayarak devam edelim...

UNIX altında, teybe yedekleme bir kaç değişik şekilde yapılabilir. Eğer varsa, teyp biriminizin kendi yedekleme yazılımını kullanmak en doğrusudur. Eğer böyle bir yazılım yoksa, standart UNIX teyp yedekleme yazılımlarından istediğiniz bir tanesini kullanabilirsiniz. Seçenekleriniz :

dump	(SVR4'de ufsdump)
cpio	ve
tar	programlarıdır.

Bu programlar arasında kullanımı en kolay olanı **tar** programıdır. Bu programın kullanımını daha önceki bölümlerde detaylı olarak anlatmıştım.

tar Programının Yedekleme Açısından Zayıf Tarafları

Bu bölümde, **tar** programının, yedekleme açısından zayıf olan bir kaç özelliğinden söz etmek istiyorum :

1. **tar** komutuyla teybe yedeklenecek dosya ve dizinlerin tamamı tek bir teyp kasetine sığmak zorundadır. (Her ne kadar birden fazla kasete peşpeşe kayıt yapabilen **tar** varyasyonları varsa da; örneğin "**bar**"; bu tür standart olmayan programlara pek güvenmemelisiniz).
2. Bağlantılı dosya ve dizinlerle (**ln** komutuyla yapılmış dosya ve dizin bağlantıları) başa çıkmak biraz zordur. **du** komutuyla toplam uzunluğunu kontrol ederek bir kasete sığacağına emin olduğunuz dosyalar, yedekleme sırasında teyp kasetine sığmayınca şaşkınlıktasınız. Ayrıca, **tar**'la alınmış bir yedeği geri yüklemeniz gerektiğinde, geri yükleme tamamlandığında başa dönüp tüm bağlantıları tekrar yapmanız gerekebilir.
3. **tar** komutunu kullanırken, siz özellikle belirtmedikçe, adı . (nokta) ile başlayan dosyalar teybe kopyalanmayacaktır. **tar cvf /dev/rst0 ./*** komutu ilk bakışta " tüm dosyaları çek " anlamında gibi görünüyorsa da, bu komut bulunduğu dizindeki, adı noktayla başlayan dosyaları kopyalayacaktır.

dump Yedekleme Programı

(SVR4'de *ufsdump*)

dump, bence en sağlıklı teyp yedekleme programıdır. Her şeyden önce standarttır. Bir kez alıştıktan sonra da kullanımı oldukça kolaydır. **tar** programına göre en önemli üstünlükleri :

1. Bir kasete sığmayan disklerin yedeğini almak sorun değildir; yeter ki elinizde yeteri kadar boş kaset bulunsun.
2. Bir diskin (daha doğrusu disk bölümünün) tamamının yedeğini tek komutla alabilirsiniz.
3. Bağlantılar (**ln** komutuyla yapılmış dosya ve dizin bağlantıları) hiç bir şekilde sorun değildir. **dump**, bağlantıların kaydını kendisi tutar ve geri yükleme durumunda gerekli bağlantıları kendisi tekrar kurar.
4. Adı neyle başlarsa başlasın, tüm dosyaları kopyalar.
5. Değişik **yedekleme düzeyleri** kavramı yardımıyla, son yedeklemeden bu yana değişmiş dosyaları ayıklayıp, sadece onların kopyasını çıkarabilir.

dump Yedekleme Düzeyleri

(dump levels)

"Yedekleme düzeyleri", **dump** (ve doğal olarak **ufsdump**) programlarının verimli olarak kullanılabilmesi için iyi anlaşılması gereken bir kavramdır.

dump programında 0'dan 9'a kadar numaralanmış 10 değişik yedekleme düzeyi bulunmaktadır. Bu düzeylerden yalnızca 0. düzey özel anlamlıdır. Bir yedekleme işlemini sıfırdan başlatırsanız belirttiğiniz disk bölümünün tamamı yedeklenecektir. Sıfırdan farklı bir düzey belirtirseniz; bu düzeyden daha düşük veya eşit düzeyde yapılmış son yedeklemeden bu yana değişmiş olan dosyalar yedeklenir.

Bunun pratik faydalarını görmek için, tipik bir UNIX bilgisayarlarında dosya ve dizinlerin geleneksel düzenlenişine bir kez daha göz atalım isterseniz.

Hatırlayacağınız gibi kullanıcıların kendilerine veya uygulama programlarına ait veri dosyaları **/home** dizininde yer alır. Uygulama programları, programlama dili derleyicileri **/usr** dizinin altına yerleştirilir. **/etc**'de sistemin yönetimiyle ilgili dosyalar yer alır. **/bin** ve **/usr/bin** dizinleri ise UNIX'inizin standart komut programlarının yeridir. Bu yerleşim planında, **/home** dizini her saat ve dakika değişebilmekte; öte yandan **/usr** dizini ancak siz, sistem yöneticisi olarak, yeni bir program yüklediğinizde veya yüklü programlar üzerinde bir güncelleştirme yaptığınızda değişmektedir. Bu durumda **/home** dizininin sık sık yedeğini almak gerekmesine rağmen, **/usr** dizini oldukça değişmez bir yapıda olduğundan, sadece değişiklik olduğunda yedeklenmelidir.

Bu değişikliklerin elle hesabını tutmak neredeyse olanaksız olduğu gibi son derece tehlikelidir. **dump** programı, değişik yedekleme düzeyleri kavramıyla bu konuda size yardımcı olacaktır.

Şimdi; diyelim ki, 1 Ocak günü sıfırdan düzeyde bir yedekleme yaptınız. Yani, bir önceki yedeklemenin ne zaman ve nasıl yapıldığına bakmaksızın, tüm disk bölümlerinin yedeklerini aldınız. Aradan bir hafta geçti; 3. düzey bir yedekleme başlattınız. Bu yedekleme sırasında, 1 Ocak'tan bu yana yaratılmış veya değişikliğe uğramış olan dosyalar yedeklenecektir. Tabii ki yedekleme de bir öncekine göre çok daha kısa sürede tamamlanacaktır.



Yalnız; çok dikkat etmeniz gereken bir konu var: 7 Ocak'ta yapacağınız 3. düzey yedekleme için, 1 Ocak'takinden **farklı bir kaset takımı** kullanmak zorundasınız. Ayrıca, kasetlerin üzerindeki etiketlere de hangi tarihte ve hangi düzeyde yedekleme için kullanıldığını açıkça kaydetmelisiniz; çünkü; yedeklediğiniz dosyaları geri yüklemeniz gerekirse, elinizdeki en son sıfırdan düzey yedekten başlayarak, sırayla daha yüksek düzeydeki yedek kasetlerini yüklemelisiniz.

Yedekleme düzeylerinin bu mantığını dikkate alarak günlük, haftalık, aylık ve 6 aylık yedeklemeler planlayabilirsiniz.

Disk kapasiteniz ile teybinizin kapasitesi uyumluysa; bir diğer deyişle, diskinizin tamamı tek bir kasete sığıyorsa, her seferinde sıfırdan düzey

yedekleme yapmanızı öneririm. Her ne kadar uzun sürerse de, akşam bürodan çıkmadan yedeklemeyi başlatabilirsiniz. Sabah geldiğinizde bitmiş olur.

Yedekleme yaparken dikkat etmeniz gereken bir kaç nokta var :

Yedekleme yapılırken, bilgisayarın günlük işler için kullanılmamasına dikkat ediniz. Yedekleme için en uygun zaman akşam saatleridir. Kullanıcılar terminallerini kapatıp gitmiş, sistem üzerindeki dosyalarda değişiklik yapan kimse kalmamıştır.

Yedekleme için kullandığınız kasetlerin sağlam, temiz olmasına dikkat ediniz. Fazla kirlenmiş ve eskimiş kasetleri kullanmayınız ve hemen yenileriyle değiştiriniz.



HİÇ BİR ZAMAN ELİNİZDEKİ TEK YEDEK KOPYA KASET ÜZERİNE YENİ YEDEKLEME YAPMAYINIZ. YEDEKLEME BİR NEDENLE YARIM KALIRSA ELİNİZDE HİÇ SAĞLAM YEDEK KALMAZ. YEDEKLEME İÇİN KULLANDIĞINIZ KASETLER EN AZ İKİ TAKIM HALİNDE OLSUN VE BU TAKIMLARI DÖNÜŞÜMLÜ OLARAK KULLANIN.

dump Komutu

(SVR4'de ufsdump)

dump komutunu kullanabilmek için girilmesi gereken komut satırı oldukça sevimsiz ve karmaşıktır. Hep aynı komutu kullanacaksanız, **csh**'in **alias** özelliğinden yararlanabilirsiniz veya sisteminize uygun **dump** komutunu bir kabuk programı haline getirebilirsiniz.

dump komutunun genel formunu göstermektense, somut bir örnek üzerinde açıklamalar daha kolay olacak sanırım.


Benim, iş yerinde yedekleme için kullandığım **dump** komutu şu :

```
# dump 0cdstfu 1000 700 18 /dev/rst0 /dev/sd0h
```

Tercümesiye :

0	sıfırıncı düzey yedekleme yapılacak,
c	yedekleme kartuşlar üzerine yapılacak (sanki çanağa yapılabilmemiş gibi) (<i>cartridge</i>)
d ve 1000	kullandığım kasetlerin yoğunluğu 1000 bpi (<i>density, bytes per inch</i>)
s ve 700	kullandığım kartuşların 700 ft uzunluğunda teyp şeritleri var, (<i>size</i>)
t ve 18	kullandığım teyp, kasetlerde 18 iz üzerine kayıt yapıyor, (<i>tracks</i>)
f ve /dev/rst0	Hangi teyp birimine kayıt yapılacağını ben belirteceğim, (<i>file name</i>)
u	Yedekleme bitince, düzey ve yedeklenen dosyaların arşivinin tutulduğu dosyayı güncelleştir. (<i>Update dump records</i>).
/dev/sd0h	Diskin h bölümü yedeklenecek (<i>h partition</i>).

```
dump 0cdstfu 1000 700 18 /dev/rst0 /dev/sd0h
```



Eğer 8 mm lik Exabyte standardında teyp ve kaset kullanıyor olsaydım, üçüncü düzey bir **dump** komutu verirken

```
# dump 3cdsbfu 54000 6000 126 /dev/rst0 /dev/sd0h
```

şeklinde bir komut kullanırdım.



dump komutunda vereceğiniz parametrelerin sırası ve doğru olmaları çok önemlidir. Komutun parametrelerini seçerken ve klavyeden yazarken çok dikkatli olmalısınız. Parametre seçiminde yararlanabileceğiniz en önemli kaynak; öncelikle teyp biriminizin kitapları; sonra da UNIX **man** sayfalarınızdır.

dump komutu başladığında, belirttiğiniz düzeyi ve daha önce yapılmış olan yedeklemelerin tarihçesinin saklandığı (**/etc/dumpdates** dosyası) dosyadaki kayıtlara bakarak, hangi dosyaların yedekleneceğine karar verir. Daha sonra, belirttiğiniz kaset parametrelerini dikkate alarak, yedeklemenin tamamlanabilmesi için kaç kaset gerekeceğini hesaplar ve yedeklemeye başlar. Kasetler doldukça da sizden yeni bir kaset takmanızı ister ve yedeklemeyi tamamlar.

restore Komutu

(SVR4'de **ufsrestore**)

dump komutuyla alınmış yedekleri geri diske yüklemek için kullanılır. Önemli bir özelliği, **dump**'la yaratılmış bir yedek kasetinin içinden tek bir dosya veya dizin, birkaç dosya veya dizin seçmenize ve sadece bunları geri indirmenize izin verir.

dump'la yaratılmış bir kasetten geri yükleme yapmak için

```
# restore -if /dev/rst0
```

komutunu vermelisiniz. Program kaseti biraz okuyup

```
restore>
```

hazır işaretini verecektir.

Bu konumda, teypte neler bulunduğunu bir hatırlamak isterseniz

```
restore>ls
```

komutunu verip, kasetteki (ya da kasetlerdeki) dosya ve dizinlerin bir listesini görebilirsiniz.

Listenin sonunda tekrar **restore>** işaretiyle karşılaştığınızda, geri yüklenmesini istediğiniz dosya veya dizinlerin isimlerinden oluşan bir liste oluşturabilirsiniz. Bu listeyi oluşturmak için **add** komutunu kullanmalısınız.

```
restore>add /etc/passwd      Sadece /etc/passwd dosyası
restore>add /bin/a*          /bin'de "a" ile başlayanlar
restore>add /var              /var dizini
```

gibi. Peşpeşe **add** komutlarıyla istediğiniz kadar dosya ve dizin adı sıralayabilirsiniz.

Listeniz tamamlandığında, geri yükleme işini başlatmak için

```
restore>extract
```

komutu kullanılır.

Bu konumda garip bir soruyla karşılaşacaksınız:

```
restore>Specify next volume #
```

Eğer yedeğiniz tek bir kasetten oluşuyorsa sorun yok; "1" yanıtını verip devam ediniz.

Eğer yedeğiniz birden fazla kasetten oluşuyorsa, en kestirme yol :

teyp birimindeki kaseti, son kasetle değiştirip bu kasetin sıra numarasını girmenizdir. **dump** programı (nedense) **add** komutuyla yarattığınız listedeki dosyaları son kasetten başlayarak geri yüklemeye başlayacaktır.



"Son kaset" sözcükleriyle "son aldığınız yedeğe ait kaset" demek istemiyorum. "Son yedekleme sırasında kullandığınız sonuncu kaset" demek istiyorum.

Geri yükleme tamamlandığında yine bir garip soruyla karşılaşacaksınız :

```
restore>Set owner/mode for . ? [y/n]
```



Bu soruya, normal koşullarda "no" anlamında "**n**" yanıtını vermeniz gerekir. "**n**" yanıtını verdiğinizde, geri yüklenmiş olan dosya ve dizinlerin sahipleri ve erişim yetki kalıpları, yedeklemeden önceki durumlarında bırakılır. Eğer "**y**" yanıtı verirsiniz, dosyaların yeni sahibi geri yüklemeyi yapan kullanıcı (normal olarak **root**) ve yetki kalıpları da bu kullanıcının **umask** değişkeninde belirtildiği şekle dönüştürülür.

cpio Yedekleme Programı

(copy input-output)

Aslında yalnızca teybe yedekleme için hazırlanmış bir komut değildir; ama genellikle bu amaçla kullanılır.

Programın en önemli özelliği, kopyalanacak dosyaların isimlerini standart giriş biriminden okuyabilmesidir. Bu sayede, kopyalanacak dosyaların isimleri başka bir program tarafından üretilebilir.

Örneğin, bir dizinde, sahibi "falanca" olan dosyaları **find** komutuna buldurup, bu dosyaların isimlerini **cpio** programına **pipe** ederek, yalnızca bu dosyaları teybe kopyalayabilirsiniz.

```
# find /home -user falanca -print | cpio -oc > /dev/rst0
```

Bu komutun ilk bölümü olan **find**, /home dizini altında sahibi "falanca" adlı kullanıcı olan dosyaları bulacak, ekrana görüntülenmesi gereken liste **cpio** programına gönderilecek ve **cpio**'da bu listedeki dosyaları /dev/rst0 teyp birimine yönlendirecektir.

cpio komutunun **-o** parametresi "*copy to output*" anlamındadır. **-c** parametresiye, çıktı dosyasının başına, diğer UNIX sistemlerle uyum sağlayabilmek amacıyla bir "**başlık**" bloğu (*header block*) kaydedilmesini istediğini belirtir. Hep aynı sistemde kullanılacak teypler için **-c** parametresi gerekmemekle beraber, bu parametreyi kullanmak iyi bir alışkanlıktır. (Kim bilir; bakarsınız teybinizi bir başka bilgisayarda okumak zorunda kalabilirsiniz).

cpio ile üzerine yedeklenme yapılmış bir teyp kasetinde neler kayıtlı olduğunu merak ederseniz

```
# cpio -civt < /dev/rst0
```

komutunu kullanabilirsiniz. (*i : input, t : table of contents, v : verbose*).

cpio ile teybe kopyalanmış dosyaları diske geri indirmek içinse; önce dosyaları indirmek istediğiniz dizine geçiniz; sonra aşağıdaki gibi bir **cpio** komutu giriniz :

```
# cd /home/ayfer
# cpio -civt < /dev/rst0
```



Önemli : **cpio** programıyla yedekleme yaparken kullandığınız dosya isimleri listesinde, dosyaların isimlerinin başında dosyaların bulunduğu dizinin tam adı belirtilmiş idiye (**/home/ugur/dosya1** gibi); geri yükleme komutunu hangi dizinde vermiş olursanız olun; geri yüklenen dosyalar diskte bu eski mutlak adreslere kopyalanırlar.

Eğer, yedekleme sırasında kullanılan dosya isimleri listesinde **./dosya1** gibi göreceli dizin tanımları kullanılmışsa (**./dosya1** :

bulduğum dizindeki **dosya1**), bu dosyaları istediğiniz dizinin altına indirebilirsiniz.

cpio ile teybe kopyalanmış dosyaların arasından **seçim** yaparak diske indirmek için; önce dosyaları indirmek istediğiniz dizine geçiniz; sonra aşağıdaki gibi bir **cpio** komutu giriniz :

```
# cd /home/ayfer
# cpio -icv "dosya1 dosya2.dat listeler vs" < /dev/rst0
```

Gerek duyarsanız, dosya isim kalıpları (*wild card*) kullanabilirsiniz :

```
# cd /home/ayfer
# cpio -icv "dos*" < /dev/rst0
```

Yedekleme Sırasında Dikkat Etmeniz Gereken Noktalar

- Üzerine yedekleme yaptığınız teyplerin etiketlerine sıra numaralarını, hangi yedekleme komutu kullanılarak yedeklendiğini, tarih ve saati ve içinde hangi disk bölümü ya da dizinin yedekleri bulunduğunu açıkça yazınız.
- Yedekleme yaparken mümkün olduğunca sistemi sizden başka kimsenin kullanmamasını sağlayınız.
- Yedek kasetlerinizi kilit altında saklayınız. Boş kasete gereksinim duyan birileri yanlışlıkla sizin yedekleme kasetlerinizi kullanmasın.
- Elinizdeki tek yedekleme kaseti üzerine yeniden yedekleme yapmayınız.

TCP/IP

Transmission Control Protocol /Internet Protocol



UNIX işletim sisteminin sağladığı en önemli olanaklardan biri TCP ve IP protokolleridir. Bu protokollerin isimleri hep bir arada anılır; o nedenle de bu kavramlara TCP/IP adı verilmiştir. TCP/IP kullanarak birden fazla UNIX bilgisayarını ve bu protokolleri destekleyen başka bilgisayarları markaları ve en önemlisi, işletim sistemlerinin cinsinden bağımsız olarak birbirlerine bağlayabilirsiniz.

TCP/IP kullanarak bilgisayar ağı kurmak için bir kaç unsura gereksiniminiz var:

- TCP/IP desteği bulunan bir işletim sistemi,
- TCP/IP desteği olan diğer bilgisayarlara bir bağlantı. (Ethernet, SLIP veya PPP).

Bu bölümde yalnızca UNIX ve TCP/IP konularına değineceğim. TCP/IP desteği bulunan bir UNIX işletim sisteminiz ve uygun donanımınız varsa, bilgisayarınızı bilgisayar ağına sokmak için yapmanız gereken işlemleri kısaca anlatmaya çalışacağım. Bir UNIX bilgisayar ağı oluşturmak için gerekli tüm adımlar yer değil; yalnızca hangi dosyalarda ne tip değişiklikleri neden ve nasıl bir mantıkla yapmanız gerektiğinden söz edeceğim.

IP Adresi ve /etc/hosts

TCP/IP protokolünün kullanılacağı her bilgisayarın bir **IP adresi** bulunmalıdır. Bu konudan daha önceki bölümlerde bahsetmiştim. Bilgisayarınıza uygun bir IP adresi saptadıktan sonra, **/etc/hosts** dosyasına makinalarınıza ilişkin bu IP adresiyle, ağınızdaki diğer bilgisayarların IP adreslerini girmeniz gerekecektir.

Tipik bir **/etc/hosts** dosyası

```
#
# Host Database
#
127.0.0.1      localhost
#
194.27.129.1   piper best mailhost
194.27.129.2   cessna
194.27.129.3   seneca
194.27.129.4   mooney
```

görünümündedir. Bu dosyadaki “127.0.0.1 localhost” satırını kesinlikle değiştirmemeniz gerekir.

/etc/hostname.le0

İkinci olarak bilgisayarınızın ethernet arabirimiyle ilgili tanıtım dosyasına (büyük olasılıkla **/etc/hostname.le0** dosyası) bilgisayarınızın adını girmelisiniz.

```
# cat > /etc/hostname.le0
piper
^D
#
```

/etc/defaultdomain

Benzeri bir şekilde, bilgisayarınızın içinde bulunduğu **domain** adını da **/etc/defaultdomain** dosyasına girmelisiniz. Bu dosyanın sistemde bulunmaması durumunda da bilgisayar ağınız çalışacaktır. Ancak, bazı ileri TCP/IP olanakları (**e-mail** gibi) ya da TCP/IP'ye dayalı üçüncü parti yazılımların (**NIS : Network Information Services**, **DNS : Domain Name Services** gibi) sağladığı olanakları kullanamayabilirsiniz.

```
# cat > /etc/defaultdomain
best.com.tr
^D
#
```

/etc/exports

Daha sonra, bilgisayar ağındaki diğer bilgisayarların sizin bilgisayarınız üzerindeki disk kaynaklarından hangilerini kullanmalarına izin verecekseniz; o disk kaynaklarını belirtmeniz gerekecektir. Bu iş için **/etc/exports** dosyasını yaratmanız gerekecektir. Örneğin, **/home** ve **/usr** dizinlerinizi başka bilgisayarların kullanımına açmak; bir başka deyişle, başka bilgisayarların bu dizinleri kendi üzerlerine **mount** edebilmeleri için :

```
# cat > /etc/exports
/home
/usr
^D
#
```

Son adım olarak da, **/etc/rc*** dosyalarınızda değişiklik yapmanız gerekip gerekmediğini kontrol etmelisiniz. Bir çok UNIX uyarlamasında **rc** dosyalarında değişiklik gerekmemektedir; ancak genede, sistem referans kitaplarınıza bakarak bir kontrol etmekte yarar var.

Bilgisayarınızı kapatıp tekrar açtığınızda, TCP/IP hazır durumda olmalıdır. Denemek için :

ping

```
# ping mooney
```

(**mooney** sözcüğü yerine ağınızdaki bir başka bilgisayarın adını yazınız)

komutuyla, ağınızdaki bir başka bilgisayarı görüp göremediğinizi kontrol ediniz. Eğer bu komutu verdiğinizde

```
mooney is alive
```

mesajını görürseniz, bilgisayar ağınız kuruldu demektir.

telnet

Hemen

```
# telnet mooney
```

komutuyla **mooney** bilgisayarına **login** etmeyi deneyin. (**mooney**'de geçerli bir kullanıcı hesabınız olmalıdır).

Diğer TCP/IP programları

Bir kez TCP/IP ağı kurduktan sonra, bilgisayar ağlarının yararlı özellikleri kullanmaya başlayabilirsiniz. Bu özelliklerin neler olduğu bu kitabın kapsamı dışında kalmaktadır; ancak UNIX ağları konusunda daha fazla bilgi edinmek için şu komutlar hakkında bilgi ve deneyim edinmeye çalışmanızı öneririm :

rlogin (*remote login*) Kendi bilgisayarınızın ekranından bir başka bilgisayara **login** etmek için (genellikle aynı kullanıcı tanııtım kodunu kullanarak) kullanılır. **telnet** komutundan pek farklı değildir.

rcp (*remote copy*) Aynı ağa bağlı iki bilgisayarın diskleri arasında dosya ve dizin kopyalamak için kullanılır.

rsh	(<i>remote shell</i>) Bir başka bilgisayar üzerinde bir program çalıştırmak için kullanılır. Diğer bilgisayarda tek bir program çalıştıracaksanız, boş yere rlogin veya telnet kullanmamanız için geliştirilmiştir.
rdump	(<i>remote dump</i>) Bir başka bilgisayara takılı olan bir teyp birimine yedekleme yapmak için kullanılır.
ftp	(<i>file transfer protocol</i>) Bir başka bilgisayara dosya göndermek ya da o bilgisayardan dosya çekmek için kullanılır. (Karşıdaki bilgisayarın UNIX işletim sistemiyle çalışıyor olması gerekmez)



Yönetiminiz altındaki UNIX bilgisayarlarında TCP/IP olanakları varsa, bir bilgisayar ağı kurduysanız ya da kurmayı düşünüyorsanız; kütüphanenizde **kesinlikle** bulunması gereken bir kitap var :

TCP/IP Network Administration
Craig Hunt

O'Reilly & Associates, Inc.
ISBN : 0-937175-82-X

Ne yapıp edip bir tane edinin.

GÜVENLİK

Security



Güvenlik, UNIX işletim sisteminin en kuvvetli aynı zamanda da en zayıf olduğu konulardan birisidir.

En kuvvetli; çünkü işletim sistemi kendisini ve kullanıcılarının sahip oldukları dosyaları çok iyi bir şekilde koruyabilmektedir.

En zayıf; çünkü bir kez kötü niyetli birisi **root** şifresini eline geçirirse sisteminizi çok kolayca mahvedebilir.

UNIX altında her türlü erişim; kullanıcı tanımlama kodları ve şifreleri temeline dayanmaktadır. Eğer kullanıcılarınız şifreleri konusunda yeteri kadar hassas davranmıyorlarsa, işyerinde salonun bir ucunda öbürüne "Yahu, senin şifren neydi?" sorusuna aynı şekilde bağırarak cevap veriyorlarsa, sisteminizde güvenlik yok demektir.

Hele hele **root** şifresi son derece önemlidir. Kullanıcı psikolojisi olsa gerek, insanlar **root** yetkileriyle çalışmaktan hoşlanıyorlar. Bu nedenle de eğer şifreyi biliyorlarsa, gerekmeseyse bile, **root** kullanıcı olarak **login** etmeyi tercih ediyorlar.

Bu son derece sakıncalı bir uygulama. Sistem yöneticisi olarak siz bile gerekmedikçe **root** kimliğine bürünmemelisiniz. Yapacağınız küçük bir klavye hatası herşeyi mahvetmenize neden olabilir. Örneğin

```
# /bin/rm a*
```

yerine yanlışlıkla

```
# /bin/rm a *
```

yazarsanız ve bu sırada da çalışma dizininiz **/etc** ise; geçmiş olsun; sistemi mahvettiniz.

Bütün bu nedenlerle **root** şifresini iyi koruyunuz. Yetkili olmayan kimselere vermeyiniz. Şifreyi sık sık değiştiriniz.

Kullanıcılarınızı şifre kullanmaya zorlayınız. Şifrelerini birbirlerine vermemeleri konusunda onları uyarınız. Bu sistemin yürümesi için her kullanıcıya farklı bir hesap açmaya üşenmeyiniz.

Kullanıcılarınızı, kolay tahmin edilebilecek şifreler seçmemeleri konusunda uyarınız. Hatta onları tehdit ediniz. Bir örnek olarak, bir başkasının şifresini tahmin eden ve onun hesabıyla sisteme girip genel müdüre hakaret dolu **e-mail** mesajları gönderen insanlardan söz edin.

İşten ayrılan ya da görev yeri değişen kullanıcıların hesaplarını hemen erişilmez hale getiriniz. Bunun en kolay yolu, ilgili kullanıcının **/etc/passwd**

dosyasındaki kaydında şifre bölümünün ilk karakteri olarak bir * eklemektir. Bir sayede, kullanıcı hesabını tekrar açmanız gerekirse, bu * işaretini kaldırırsınız, olur biter.

Sisteminizin **/var/adm/messages** veya benzeri dosyalarına sık sık bakınız. Sistemde meydana gelen başarısız **login** denemeleri, **root** olarak yapılan **login**'ler ve **su** komutuyla **root** olan kullanıcılarla ilgili kayıtlar bu dosyalarda arşivlenmektedir. Bu dosyadaki kayıtlar, sisteminize girmeye çalışan kimseler olup olmadığı konusunda bir fikir verecektir.

Terminal bağlı seri arabirimlerden ve bilgisayar ağı üzerinden gelen **telnet**, **rlogin** gibi bağlantılarda **root** olarak **login** edilmesini önleyiniz. Bu önleme işini BSD UNIX'lerde **/etc/ttytab** dosyasında; SVR4 UNIX'lerde de **/etc/defaults/login** dosyalarında gerekli değişiklikleri yaparak halledebilirsiniz.

SUID programlar en tehlikeli güvenlik gedikleridir. Hatırlarsanız, **SUID** özelliğine sahip programlar hangi kullanıcı tarafından kullanılıyor olurlarsa olsunlar, çalıştıkları sürece **root** yetkilerine sahiplerdir. Eğer bir **SUID** program, bir şekilde bir kabuk programına çıkış veriyorsa (örneğin **vi** programı verir), bunu keşfeden bir kullanıcı şifre vermeden **root** oldu demektir. Ne isterse yapar.

Sisteminiz ilk kurulduğunda **SUID** programların bir listesini alın ve bu listeyi iyi saklayın. Zaman zaman sistemdeki **SUID** programların bir listesini alıp, elinizdeki ilk listeyle karşılaştırın. **SUID** programların bir listesini almak için şu komutu kullanabilirsiniz :

```
# find / -user root -perm -4000 -exec ls -l {} \;
```

/bin, **/etc**, **/usr** ve **/** dizinlerindeki dosyaların sahipleri dışında kullanıcılar tarafından yazılabilir durumda olmamalarına dikkat edin. Özellikle **/etc/passwd** dosyası kesinlikle **rw-r--r--** yetki kalıbına sahip olmalıdır.

Bilgisayarınız Internet üzerinden dünyaya açıksa bu bilgisayar üzerinde gizli ve hayati önemi olan işler yapmayınız. Siz istediğiniz kadar önlem alın, bir yerlerde, hayattaki tek zevki başkalarının bilgisayarlarına yetkisizce girip ortalığı dağıtmak olan birileri olacaktır. İngilizce deyimiyse *Hacker* adı verilen bu insanlar, virüs programı yazanlarla aynı mantıkta çalışıp, ortalığı birbirine katmaktan zevk almaktadırlar.

Eğer bilgisayarlarınızı hiç bir kuşku duymayacak şekilde korumak istiyorsanız onları Internet'e bağlamayın ve kimseye dışardan erişim olanağı vermemek için hiç modem bağlantısı yapmayın. Bu yöntem, çok güvenli olmasına karşın, bir o kadar da kullanışsız olacaktır. Güvenliğin önemli olduğu uygulamalarda, bir grup bilgisayarı Internet'e bağlarken; araya "*firewall*" adı verilen ve yazılımları özel bir şekilde düzenlenmiş olan bilgisayarlar kullanılabilmektedir.

SON SÖZ

Bu kitapta UNIX hakkında söylenebileceklerin çok ama çok az bir kısmından söz edebildim. Zaten amacım da UNIX konusunda deneyimi olmayan ve/veya UNIX konusunda ön yargılı olan bilgisayar kullanıcılarına UNIX dünyasını sadece birazcık tanıtmaktı.

UNIX, profesyonel bir bilgisayarının tüm bir meslek hayatını dolduracak kadar geniş bir işletim sistemi. Öğrenmekle, okumakla ve kullanmakla bitmiyor. Her gün birileri bir şeyler ekliyor ve UNIX hep sizin önünüzde gidiyor.

UNIX dünyasındaki kullanıcılar arasında gözlediğim ortak bir özellik var: Eğer bir kullanıcı MS-DOS veya Macintosh işletim sistemi konusunda biraz deneyimliyse, UNIX'le ilk tanıştığında "Bu ne yahu!" diye tepki gösteriyor. Eğer işi gereği ya da sabrı sayesinde, UNIX'de biraz ilerlerse UNIX'e karşı büyük bir saygı ve sevgi duymaya ve diğer kişisel bilgisayar işletim sistemlerini de küçümsemeye başlıyor.

UNIX piyasasında "şiir gibi UNIX kullanan" insanlardan söz edildiğini duydum. Hatta bu ifadeyi, tanıdığım bir sistem yöneticisi için kendim de kullandım. Gerçekten iyi UNIX kullanan insanları seyretmek zevkli oluyor; aynı usta birç oyuncuları seyretmek gibi. İyi bir UNIX sistem yöneticisi olmak için en az bir yıllık bir çıraklık sürecine gereksiniminiz var. Eğer ustanız "şiir gibi" UNIX kullanıyorsa, sizin ondan öğreneceğiniz çok şey olacaktır.

Eğer bu kitapla size UNIX'i tanıtabildiysem ("öğretebildiysem" dememeye dikkat ediyorum) ve var idiyse, ön yargınızı kırabildiysem ne mutlu bana. Ama okuyucu olarak yapabileceğiniz en büyük hata "UNIX'i öğrendim!" demeniz. **Hayır, bu kitabı okumakla UNIX'i öğrenmediniz!** Örneğin X-Windows (X11R5 veya X11R6) hakkında hiçbir şey yazmadım. **motif, awk, emacs, perl, troff, rpc, NIS, DNS, sendmail, make, tex, sccs**'den hiç söz etmedim. Bu komut ve kavramların her biri hakkında ciltler dolusu kitaplar var; "öğrendim" diyebilmek için onları da okumalısınız.

UNIX konusunda kendini geliştirmek isteyen okuyuculara bir önerim var:

Bir yerden **LINUX** işletim sistemini bulun; kendiniz kurun ve kullanın. LINUX, ticari olarak değeri olmayan, sadece CD fiyatına (40 - 50 ABD \$) satın alınabilecek ve INTEL 386-486-Pentium tabanlı bilgisayarlar için, binlerce kişiden oluşan bir meraklı kitlesi tarafından geliştirilmiş bir işletim sistemi. Adını, projeyi ilk olarak başlatan Linus Torvalds (Finlandiyalı) isimli bir öğrenciden alan bu işletim sistemi çok iyi bir eğitim aracı. Kafa göz yara yara insana UNIX öğretiyor. GNU (GNU is Not UNIX) adı verilen bir klübün yayınladığı bedava (*public domain*) programlarla da desteklenen LINUX, eksiksiz bir uyarlama. X-Windows'dan NFS'e kadar her kavramı içeriyor. Bir çok kuruluş, LINUX'u profesyonelce kullanıyor; siz de çekinmeden kullanabilirsiniz.

EKTEKİ DİSKET

Bu kitapta anlatılan UNIX komut ve kavramlarını denemeden öğrenmek ve gerektiğinde hatırlamak pek olası değil. Elinizin altında UNIX işletim sistemiyle donatılmış bir bilgisayar yoksa işiniz zor demektir.

Bu nedenle, kitaba bir disket ekledim. Bu diskette, kitaptaki tüm komutlar için olmasa da, bir çoğu için MS-DOS altında çalışan modellerini bulacaksınız. Komutların isimleri (tabii ki, dosya isimlerinin sonundaki **.EXE**'ler hariç) ve kullanma kuralları UNIX'deki karşılıklarının hemen hemen aynısı. Deneme çalışmaları için işinize yarayacağına inanıyorum.

Programları, MS-DOS bilgisayarının diskine yüklemek için şu adımları izlemelisiniz :

1. Yaklaşık 1 MegaByte boş yer olan bir diskinize \UNIX isimli bir dizin yaratın.
2. Disketteki UNIX.EXE isimli dosyayı (zaten başka dosya yok) bu dizine kopyalayın.
3. Çalışma dizininizi \UNIX olarak değiştirip, UNIX programını çalıştırın.
4. Program aşağıda listesi bulunan dosyaları kendisi yaratacaktır.
5. Bu programları herhangi bir dizinden kullanabilmek için isterseniz C:\AUTOEXEC.BAT dosyasındaki PATH komutunuza bu dizini de ekleyebilirsiniz.

cal	du	sleep
cat	find	sort
chmod	grep	tail
compress	head	tar
cp	ls	touch
cpio	mv	wc
df	rm	vi

İyi eğlenceler....

