

- **find**
- **grep**
- **tar Komutu ve tar Dosyaları**
- **tar Komutu ile Yedekleme**
- **rsh**
- **xargs**
- **at**
- **date, hwclock**
- **lynx**
- **cut**
- **tee**
- **script**
- **split**

find

Günümüzün tipik kişisel bilgisayarlarında disk kapasiteleri artık onlarca GigaByte ile ölçülmektedir. Bu kadar büyük disklerde de doğal olarak çok sayıda dizin ve onbinlerce dosya yer alabilmektedir. Zaman zaman adının yalnızca bir kısmını bildiğiniz ama bulunduğu dizini bir türlü hatırlayamadığınız dosyalar ve dizinler olacaktır. Tek tek bütün dizinlere girip **ls** komutuyla bu dosya ya da dosyaları aramaktansa **find** komutunu kullanmak hayatı kolaylaştıracaktır. Örneğin;

```
find /home/cayfer -name nerede.dat
```

komutu, **/home/cayfer** dizininden başlayarak, bu dizinde ve daha derinlerdeki dizinlerde adı “**nerede.dat**” olan dosyayı arar; bulursa yerini görüntüler.



```
cayfer@pusula.bilkent.edu.tr: /home/cayfer - Konsolle - Konsolle
[cayfer@pusula cayfer]# find /home/cayfer -name nerede.dat
/home/cayfer/tmp/nerede.dat
/home/cayfer/fp/nerede.dat
```

Yukarıdaki örnekte, “**nerede.dat**” isimli iki dosya, **/home/cayfer** dizini altındaki **tmp** ve **fp** dizinleri altında bulunmuştur.

find komutuyla yalnızca adı ya da adının bir parçası bilinen dosyaları aramak için kullanılmaz. Dosyalar adları dışında da özelliklerine göre aranabilir.

Genel formu:

```
find başlama-dizini kriter[ler] [-exec komut “;”]
```

olan komutla:

- belirli bir tarihte değişikliğe uğramış dosyaları,
- belirli bir tarihten bu yana değişmiş dosyaları,
- belirli bir boydan daha büyük ya da küçük dosyaları,
- belirli erişim yetkilerine sahip dosyaları ve dizinleri,
- belirli bir kullanıcıya ait dosya ve dizinleri de arayıp bulabilirsiniz.

Üstelik, verdiğiniz arama kriterlerine uyan dosyalar ve dizinler üzerinde uygulamak isteyebileceğiniz LINUX 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 dosya yapılarında (disk, CD ve başka bilgisayarların dosya sistemleri de olabilir) yapılmasını istiyorsanız, ilk parametre olarak “/” sembolünü kullanınız; yani aramayı tüm dosya sistemlerinizin en tepesinden başlatınız.

Bilgisayarınızın CD-ROM sürücüsü varsa, bu sürücüye bir CD takılıysa, bu CD mount edilmiş durumdaysa ve aramayı "/" dizininininden başlatırsanız, arama ağacı 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 "/" dizininininden başlatmamanızı öneririz. **-mount** parametresiy-le aramanın başka dosya sistemlerine de atlamasını önleyerek bu dertten de kurtulmak mümkündür elbette.



kriter[ler]: Aranılan dosya veya dizinlerin ortak özelliklerini tanımlayan kriterlerdir. Birkaç 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-k]9"** Adı **a9**, **b9**, ..., **j9** veya **k9** olan dosyalar.

Dikkatinizi çektiyse, **-name** kriterinde dosya adı tam olarak yazıldığında tırnak (") kullanılmıyor; oysa * karakterini içeren bir kalıp kullanıldığında (wildcard) bu kalıbı tırnak (") içinde yazmak gerekiyor. Bunun nedeni şudur: Bir komut verdiğinizde, bu komut önce kabuk programınız tarafından irdelenir. Bu irdeleme sırasında rastlanan * karakterleri dosya adı kalıplarının bir parçası olarak kabul edilip, * içeren parametre 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 karşılaşacakları * karakterlerine dokunmadan, parametreleri oldukları gibi çalıştıracak programa aktarmaları için, kalıp tanımları tırnak içine alınır.



- user ayfer** Sahibinin adı "**ayfer**" olan dosyalar ve dizinler.
- group yönetim** Sahibi "**yonetim**" grubuna dahil olan dosyalar ve dizinler.

-perm 755	Erişim yetki düzeyi 755 olan dosyalar ve dizinler.
-newer dosya1	dosya1 isimli dosyadan daha sonraki bir saat ya da tarihte değişikliğe uğramış olan dosyalar ve dizinler.
-size 10	Diskte kapladığı alan 10 blok olan dosyalar. (1 blok = 512 Byte)
-size +100k	Diskte kapladığı alan 100 Kbyte'dan büyük olan dosyalar.
-size -100c	Diskte kapladığı alan 100 byte'dan az olan dosyalar.
-ctime 3	Tam 3 gün önce değişikliğe uğramış olan dosyalar ve dizinler.
-ctime +8	8 günden daha uzun bir süre önce değişikliğe uğramış olan dosyalar ve dizinler.
-ctime -8	8 günden daha kısa bir süre önce değişikliğe uğramış olan dosyalar ve dizinler.
-mtime 3	Tam 3 gün önce değişikliğe uğramış olan dosyalar ve dizinler.
-mtime +8	8 günden daha uzun bir süre önce değişikliğe uğramış olan dosyalar ve dizinler.
-mtime -8	8 günden daha kısa bir süre önce değişikliğe uğramış olan dosyalar ve dizinler.
-atime -3	3 günden daha kısa bir süre içinde bir şekilde erişilmiş olan dosyalar ve dizinler.
-amin -3	3 dakikadan daha kısa bir süre içinde bir şekilde erişilmiş olan dosyalar ve dizinler.
-mmin -3	3 dakikadan daha kısa bir süre içinde değişikliğe uğramış veya yaratılmış olan dosyalar ve dizinler.
-type f	Dosyalar.
-type d	Dizinler.

-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 yapılmadığını; **-ctime** ise dosyanın içeriği yanısıra özelliklerinin de değişip değişmediğini kontrol eder.

Örneğin, sahibi değişen bir dosya **-mtime** tarafından farkedilmezken **-ctime** tarafından dikkate alınır.

Bu arama kriterlerini bir arada kullanabilirsiniz. Örneğin, sahibi “**hakman**” olan ve son 40 gündür kullanılmamış dosyaları bulmak isterseniz, kullanmanız gereken **find** komutu

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

olmalıdır.

Şimdi, sık kullanılan **find** formları için birkaç örnek verelim:

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

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

```
find / -name core -exec /bin/rm {} “;”
```

“/” dizininden başlayarak tüm dizin yapısında “**core**” isimli dosyaları arar ve bulduklarını siler.

find komutu **-exec** parametresiyle birlikte kullanıldığında, bulunan dosya ve izin isimleri, “{ }” arasına parametre olarak yerleştirilecek ve **-exec**’den hemen sonra belirtilmiş olan program bu parametre ile çalıştırılacaktır. Yukardaki örnekte bulunan her bir “**core**” dosyası için **/bin/rm core** komutu çalıştırılmış olacaktır. En sondaki “;” parametresi çok önemlidir ve unutulmamalıdır. Sondaki bu “;” karakter dizisinin gerekliliği tamamen **find** programının yazılışından kaynaklanmaktadır.



Kim Korkar LINUX'tan?

find, sistem yönetiminden sorumlu olanların oldukça sık kullanacakları bir komuttur. LINUX, çeşitli programların kullanımı sırasında sistemin bütünlüğünü tehdit eden bir problemle karşılaştığında (ki bu genellikle hatalı yazılmış programlar yüzünden olur) “**core dumped**” mesajıyla birlikte, belleği **core** isimli bir dosyaya kopyalar. Bu **core** dosyaları, programcıların problemin nedenini bulmasına yardımcı olmak amacıyla yaratılır. Bu dosyaları irdeleyerek problemin nedenini bulmak pek kolay olmadığından, bu dosyaları içeriklerine bakmaksızın silebilirsiniz.

Bir başka önemli örnek:

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

/home dizininden başlayarak **hasan** isimli kullanıcıya ait dosyaları arar ve bulduklarını siler. Sisteme erişim hakları iptal edilen bir kullanıcıya ait dosyaları tek harekette silmek için kullanılabilir.

```
find /home -name "*.mp3" -exec /bin/rm {} ";"
```

/home dizininden başlayarak adı ***.mp3** kalıbına uyan dosyaları arar ve bulduklarını siler.

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

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

find komutuyla birlikte kullanılabilen kriterleri çeşitli mantık operatörleriyle birleştirebilirsiniz.

Bunlar:

- a:** “ve”
- o:** “veya”
- \!:** “değil”

operatörleridir.

Örneğin:

```
find . -name "*.tmp" -a -size +1000k
```

çalışma dizininde (“.”) ve varsa altındaki dizinlerde adı ***.tmp** kalıbına uyan

ve büyüklüğü 1000 KByte'den fazla olan dosyaları bulur.

```
find /home/cayfer \! -user cayfer
```

cayfer isimli kullanıcının kişisel dizininde yer alan ama **cayfer**'e ait olmayan dosyaları bulur.

Bu örnekteki “**değil**” anlamında kullanılan “\!” operatöründeki “\” işareti, ardından gelen “!” işaretinin özel bir anlamı olduğunu ve kabuk programı tarafından yorumlanmaya çalışılmaması gerektiğini belirtmek için kullanılmaktadır.

Hatırlarsanız, daha önce UNIX işletim sisteminde kendi komutlarınızı yaratabileceğinizden bahsetmiştik. İşte bu uygulamaya bir örnek vermek için uygun bir noktadayız.

find komutu oldukça yetenekli ve çok seçenekli bir komut olmakla birlikte klavyeden yazması da oldukça uzun bir komuttur. Dosyaları yalnızca adlarıyla arayan, **find** komutundan daha kısa bir LINUX komutu yaratmaya ne dersiniz?

Aşağıdaki **bash** kabuk programını herhangi bir editörle (tabii ki **vi** ile) kişisel dizininizde “**ff**” isimli bir dosyaya giriniz:

```
#!/bin/bash
case $# in
  1) find . -name "$1" ;;
  2) find "$1" -name "$2" ;;
  *) echo "Hata! Komutun kullanımı : ff [dizin] isim"
    echo "                ff [dizin] \"xyz*\""
    echo "                ff [dizin] \"*xyz\""
esac
```

Daha sonra,

```
chmod a+x ff
```

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

Bu kabuk programı, önce kendini çalıştıran komut satırında verilmiş olan parametrelerin sayısını kontrol ediyor. (“\$#”) Eğer tek parametreyle başlatılmışsa “**find . -name parametre**” komutunu çalıştırıyor. Eğer iki parametreyle başlatılmışsa, birinci parametreyi aramanın başlatılacağı dizin kabul edip “**find param1 -name param2**” komutunu çalıştırıyor. Eğer parametre sayısı bir veya iki değilse üç satırdan oluşan bir hata mesajı veriyor.

Örnekler:

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

Yeni yarattığınız **ff** komutunu kullanmak istediğinizde komut programının bulunamadığına ilişkin bir mesaj alıyorsanız, çalışma dizininiz **PATH** değişkeninizde yer almadığı için olabilir. O zaman programı **./ff** komutuyla çalıştırabilirsiniz.

Yeni **ff** komutunuzu iyice denedikten sonra genel kullanıma sokmak için **ff** dosyasını **/usr/local/bin** altına kopyalayabilirsiniz. Ancak bu kopyalamayı yapabilmek için **root** kullanıcı yetkilerine gereksiniminiz olacaktır.



locate

Birçok modern Linux dağıtımında, arka planda çalışan "slocate" programı genellikle günde bir kez olacak şekilde disk(ler)inizdeki dosyaların isimlerini indeksler. Bu sayede "locate xyz" komutunu kullanarak normalde birkaç dakika sürebilecek find /-name "*xyz*" benzeri komutun sonucunu saniyeler içinde görebilirsiniz. Tabi bu sonuç sisteminizin o andaki hali için değil, en son indekslemenin yapıldığı zaman için geçerli olacağından, sisteminizde ne kadar değişiklik olduğuna bağlı olarak kimi zaman doğru olmayabilir.

grep Komutu

Dosyaları adları ve sahipleri gibi özelliklerine göre arayıp bulma işini “**find**” komutu ile halledebilirsiniz. Ama bazen de dosyaları isimlerine göre değil, içeriklerine göre aramanız gerekecektir. Belli bir karakter dizisini içeren dosyaları ve/veya bir dosya grubu içinde belli bir karakter dizisi geçen satırları bulmak için;

grep [-irvnc] dizi dosya(lar)

komutunu kullanmalısınız.

Hemen birkaç örnek...

İçinde sisteminizin tüm tanımlı kullanıcıları için birer satır yer alan **/etc/passwd** dosyasında “**murat**” diye bir sözcük olup olmadığını kontrol etmek istediğinizde

grep murat /etc/passwd

komutunu kullanabilirsiniz. Bu dosyanın içinde “**murat**” sözcüğü geçen tüm satırlar standart çıktı birimine listelenecektir. Ancak komutu bu örnekteki gibi kullanırsanız içinde “**Murat**” geçen satırları yakalayamazsınız. Büyük-küçük harf ayırımı yapılmaksızın arama yapılmasını istiyorsanız komutu;

grep -i murat /etc/passwd

şeklinde vermeniz gerekir.

İçinde “**Murat**” ya da “**murat**” geçen satırların satır numaralarını da görmek isterseniz:

grep -ni murat /etc/passwd

formunu kullanabilirsiniz.

Diyelim ki bulunduğunuz dizinde, adı “**prog**” ile başlayan dosyalar arasında bir veya birkaç tanesinin içinde “**cayfer**” sözcüğünün bulunduğunu biliyorsunuz ama hangileri olduğunu hatırlayamıyorsunuz!

İşte çözüm:

grep cayfer prog*

Adı “**prog**”la başlayan dosyalarda “**cayfer**” sözcüğünün kaç defa geçtiğini öğrenmek isterseniz **grep** komutunu:

grep -c cayfer prog*

şeklinde **-c** parametresiyle kullanılabilir.

Kim Korkar LINUX'tan?

Eğer içinde belirli bir karakter dizisi geçen dosyaları bir dizin ağacında arayacaksanız **-r** parametresi çok işinize yarayacaktır.

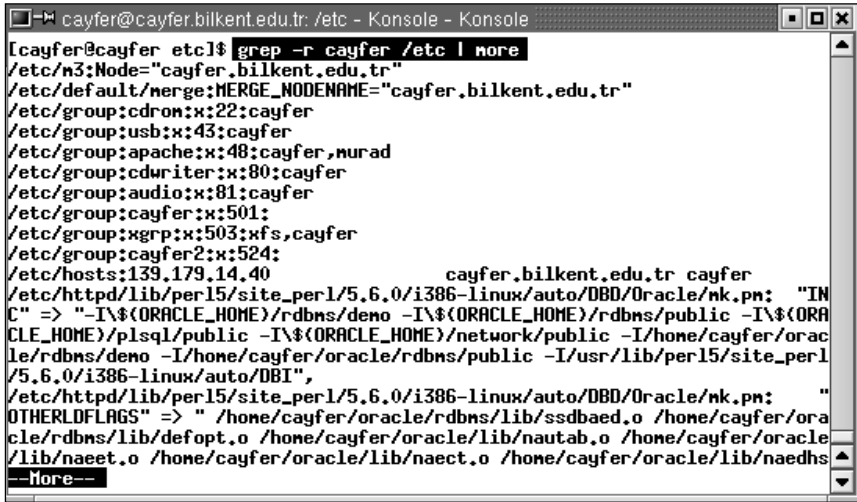
```
grep -r cayfer *
```

komutu “**cayfer**” karakter dizisini çalışma dizinindeki ve bu dizin altındaki dizinlerdeki dosyalarda arar ve içinde “**cayfer**” geçen dosyaların isimlerini listeler.

grep komutunun ürettiği listenin çok uzun olması durumunda, komutu

```
grep -r cayfer * | more
```

şeklinde kullanırsanız, **grep** komutunun ürettiği listeyi **more** komutuna yönlendirerek listenin ekrana sayfa sayfa görüntülenmesini sağlayabilirsiniz.



```
cafer@cayfer etc]# grep -r cayfer /etc | more
/etc/n3:Node="cayfer.bilkent.edu.tr"
/etc/default/merge:MERGE_NODENAME="cayfer.bilkent.edu.tr"
/etc/group:cdrom:x:22:cayfer
/etc/group:usb:x:43:cayfer
/etc/group:apache:x:48:cayfer,nurad
/etc/group:cdwriter:x:80:cayfer
/etc/group:audio:x:81:cayfer
/etc/group:cayfer:x:501:
/etc/group:xgrp:x:503:xfs,cayfer
/etc/group:cayfer2:x:524:
/etc/hosts:139.179.14.40          cayfer.bilkent.edu.tr cayfer
/etc/httpd/lib/perl5/site_perl/5.6.0/i386-linux/auto/DBD/Oracle/nk.pn: "IN
C" => "-I$(ORACLE_HOME)/rdbms/deno -I$(ORACLE_HOME)/rdbms/public -I$(ORA
ACLE_HOME)/plsq1/public -I$(ORACLE_HOME)/network/public -I/home/cayfer/orac
le/rdbms/deno -I/home/cayfer/oracle/rdbms/public -I/usr/lib/perl5/site_perl
/5.6.0/i386-linux/auto/DBI",
/etc/httpd/lib/perl5/site_perl/5.6.0/i386-linux/auto/DBD/Oracle/nk.pn: "
OTHERLDFLAGS" => " /home/cayfer/oracle/rdbms/lib/ssdбаed.o /home/cayfer/ora
cle/rdbms/lib/defopt.o /home/cayfer/oracle/lib/nautab.o /home/cayfer/oracle
/lib/naect.o /home/cayfer/oracle/lib/naect.o /home/cayfer/oracle/lib/naedhs
--more--
```

Bazı durumlarda size bir dosyada içinde “**cayfer**” geçen satırlar değil de, “**cayfer**” geçmeyen satırlar gerekir.

```
grep -v bash /etc/passwd
```

komutu **-v** parametresinden dolayı **/etc/passwd** dosyasındaki satırlar arasında, içinde “**bash**” geçmeyenleri listeleyecektir.

grep komutu ille de dosyalar içinde arama için kullanılmaz. Başka programların ürettiği çıktılar arasında da arama yapabilirsiniz. Örneğin, sisteminizin

internet servislerini ve bağlantılarını yöneten **xinetd** isimli sürecin konfigürasyonunda değişiklik yaptığınızda ve bu programı yeniden başlatmanız gerektiğinde **xinetd**'nin süreç numarasını bilmeniz gerekecektir. LINUX altında çalışan bilgisayarlarda, çok meşhur bazı işletim sistemlerinde olduğu gibi ayar değişikliklerinden sonra sistemi kapatıp açmak gerekmez. **xinetd** yazılımına kendisini yeniden başlatması için; daha doğrusu ayar dosyalarını yeniden yüklemesi için **-HUP** sinyalini göndermek yeterli olacaktır.

xinetd sürecinin numarasını, oldukça uzun olan “**ps e | grep xinetd**” listesinde gözle aramak yerine bu listeyi **grep** filtresinden geçirebilirsiniz.

```

root@notebook.lojman.bilkent.edu.tr: /root - Shell - Konsolle
[root@notebook root]# ps e | grep xinetd
1238 ?      00:00:00 xinetd
[root@notebook root]#

[root@notebook root]# kill -HUP 1238
[root@notebook root]# █

```

grep komutunun **gzip** veya **compress** ile sıkıştırılmış dosyalar içinde de arama yapabilen bir varyasyonu vardır: **zgrep**.

Sisteminizin **/var/log** dizinindeki dosyalara bakarsanız bazılarının isimlerinin “**1.gz**”, “**2.gz**” gibi karakterlerle bittiğini görürsünüz. Bunlar sistemin **logrotate** işleviyle belirli aralıklarla sıkıştırıp arşivlediği log dosyalarıdır. (**logrotate** ile ilgili ayrıntılı bilgiyi kitabın “Sistem Yönetimi” bölümünde bulabilirsiniz.) Bir dosya sıkıştırıldığında artık içinde okunabilir karakterler yer almayacaktır. Yani, sıkıştırılmış bir dosya içinde “**cayfer**” sözcüğünü **grep** ile aramak pek işe yaramaz.

Bu sıkıştırılmış dosyalarda bir şeyler arayacağınız zaman **grep** komutunu kullanabilmek için önce bu dosyaları **gunzip** gibi bir komutla açmanız ve ondan sonra **grep** komutuyla içlerinde arama yapmanız gerekir. Ama durun bir dakika... LINUX kullandığınızı unuttunuz herhalde... **grep** yerine

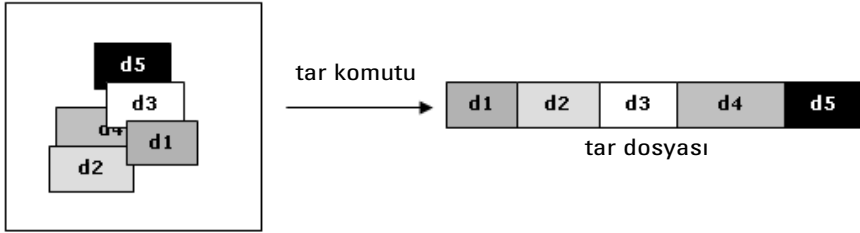
```
zgrep cayfer /var/log/mail/*
```

komutuyla **cayfer** sözcüğünü **/var/log/mail** altındaki tüm dosyalarda arayabilirsiniz. Üstelik **zgrep**, sıkıştırılmış dosyaları ve normal dosyaları ayırdeyip arama işlemini ona göre yapacak, aradığını bulursa da rapor edecektir.

tar Komutu ve tar Dosyaları

LINUX dünyasında çalışan birinin; hele sistem yöneticisiyse, çok sık karşılaştacağı bir kavramdır “**tar**”... “**Tape Archive**” sözcüklerinden türetilmiştir. UNIX işletim sisteminin yazılmaya başlandığı yıllarda teyp makaralarına ya da kasetlerine dosya kaydetmek ve bu dosyaları geri indirmek için geliştirilmiştir. Artık teyp kullanımı, yaygınlığını kaybetmiş olsa da (aslında büyük sistemlerde hala en yaygın yedekleme ve arşivleme ortamı teyptir, ancak küçük ofis ortamlarında artık teyp sürücülere pek rastlanmamaktadır) “**tar dosya**” mantığı aynen ve daha da yaygınlaşarak kullanılmaktadır.

tar dosyası hazırlamanın çok basit bir mantığı vardır: “**tar’lanmak**” istenen dosyaları peşpeşe ekleyip tek bir dosya elde etmek.



Yukarıdaki şemadaki gibi **d1**, ..., **d5** isimli dosyaları tek bir **d.tar** dosyasında birleştirmek için:

```
tar -cvf d.tar d1 d2 d3 d4 d5      veya
tar -cvf d.tar d?
```

komutlarından biri kullanılabilir.

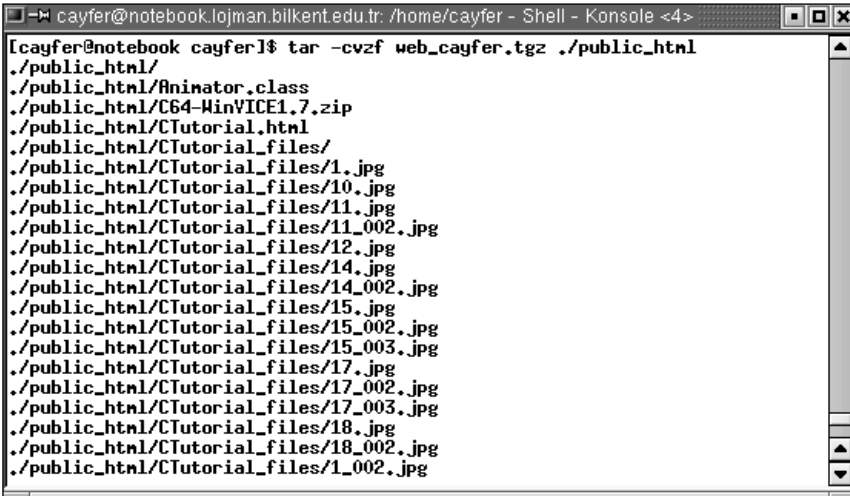
tar dosyaları, ya da LINUX jargonuna uygun olarak söylemek gerekirse, “**tar yumakları**” (*tar balls*) son yıllarda program paketlerini taşımak için en çok kullanılan yöntemdir. Windows dünyasında da aynı bu şekilde çok kullanılan bir dosya paketleme yöntemi vardır: ZIP.

Diyeceksiniz ki “ZIP dosyaları peşpeşe ekliyor eklemesine ama hem dosyaları sıkıştırarak toplam paketi küçültüyor hem de alt dizinleri de paketin içine yerleştirebiliyor.”

Bir kere şu noktada anlaşalım: Windows serisi işletim sistemleri kişisel kullanım için tasarlanmıştır ve bu kavram içinde de oldukça başarılıdır. Oysa UNIX, 30 yıldan fazla bir süredir geliştirilmektedir ve profesyonel kullanım için tasarlanmıştır. Bu nedenle bilişim dünyasında Windows'un yapıp da LINUX'un yapamadıkları değil, bunun tam tersi konuşulur. Sözün kısası

```
tar -cvzf web_cayfer.tgz ./public_html
```

komutuyla `/home/cayfer/public_html` dizini altındaki herşeyi `web_cayfer.tgz` dosyası içine sıkıştırarak paketleyebilirsiniz.



```

cayfer@notebook cayferl$ tar -cvzf web_cayfer.tgz ./public_html
./public_html/
./public_html/Animator.class
./public_html/C64-MinVICE1.7.zip
./public_html/CTutorial.html
./public_html/CTutorial_files/
./public_html/CTutorial_files/1.jpg
./public_html/CTutorial_files/10.jpg
./public_html/CTutorial_files/11.jpg
./public_html/CTutorial_files/11_002.jpg
./public_html/CTutorial_files/12.jpg
./public_html/CTutorial_files/14.jpg
./public_html/CTutorial_files/14_002.jpg
./public_html/CTutorial_files/15.jpg
./public_html/CTutorial_files/15_002.jpg
./public_html/CTutorial_files/15_003.jpg
./public_html/CTutorial_files/17.jpg
./public_html/CTutorial_files/17_002.jpg
./public_html/CTutorial_files/17_003.jpg
./public_html/CTutorial_files/18.jpg
./public_html/CTutorial_files/18_002.jpg
./public_html/CTutorial_files/1_002.jpg

```

Şimdi `tar` komutunda kullanılacak bazı önemli parametrelere ve `tar`'ın çok kullanıldığı işlere bir göz atalım:

tar Parametreleri	
c	Create: tar dosyası yaratılacağını belirtir.
x	Extract: Bir tar dosyasının açılacağını (çözüleceğini) belirtir.
t	Table of contents: Bir tar dosyasının içeriğinin listeleneceğini belirtir.
v	Verbose: Bir tar dosyası yaratılırken ya da açılırken elden geçen dosyaların isimlerini ekrana listelemek için kullanılır.

z	tar dosyasının sıkıştırılmış bir dosya olarak kullanılacağını belirtir. Yani, dosya yaratılıyorsa sıkıştırılarak yaratılacaktır; dosya çözülyorsa, önce gunzip ile açılması gerektiğini belirtir.
f	File: Yaratılacak, açılacak ya da içindekiler tablosu listelenecek tar dosyasının adının komut satırında verileceğini belirtir. tar dosyası yaratırken, yaratılacak dosya adının verileceğinin belirtilmesi biraz garip geldi, değil mi? Evet, haklısınız ama diskte gerçek bir dosya yaratmaksızın tar dosyası oluşturmak oldukça anlamlıdır. Bu şekilde kullanımın bir örneğini tar Komutu ile Yedekleme başlığı altında bulabilirsiniz.

Şimdi sıra örneklerde:

```
tar -cvf dat_dosyalar.tar *dat
```

Çalışma dizininde yer alan ve isimleri “**dat**” ile biten tüm dosyaları **dat_dosyalar.tar** adıyla birleştirir.

```
tar -cvzf dat_dosyalar.tgz *dat
```

Aynı işi dosyaları sıkıştırarak yapar. Sıkıştırma işi **gzip** programı kullanılarak yapılır.

```
tar -tf dat_dosyalar.tar
```

dat_dosyalar.tar isimli dosyanın içindeki dosya ve dizinlerin listesini döker.

```
tar -xvf dat_dosyalar.tar
```

dat_dosyalar.tar dosyasının içindeki tüm dosyaları çalışma dizinine açar.

```
tar -xvf dat_dosyalar.tar birinci.dat
```

dat_dosyalar.tar dosyasının içinden yalnızca **birinci.dat** isimli dosyayı çalışma dizinine açar. Eğer açmak istediğiniz dosya, **tar** dosyası oluşturulurken bir dizin altında yer alıyor idiyse, bu dosyayı açarken o dizini de belirtmelisiniz. (**tar -xvf d.tar /home/cayfer/mail** gibi.)

```
tar -xvf dat_dosyalar.tar -C /tmp/yeni_dizin
```

dat_dosyalar.tar dosyasının içindeki tüm dosyaları **/tmp/yeni_dizin** dizinine açar.

```
tar -cvf /dev/rst0 /home
```

/home dizinindeki herşeyi birinci SCSI teyp birimindeki kasete kaydeder (**st0**) ve işi bitirince kaseti başa sarar. (**r:** *rewind*)

```
tar -cvf /dev/nrst0 /home
```

/home dizinindeki herşeyi birinci SCSI teyp birimindeki kasete kaydeder (**st0**) ve işi bitirince kaseti kaldığı yerde bırakır. (**nr:** *no rewind*)

```
tar -cvzf yedek.tgz /home -exclude /home/cayfer
```

/home/cayfer dizini hariç, **/home** altındaki tüm dosya ve dizinleri **yedek.tar** dosyasında sıkıştırarak birleştirir.

tar programının yarattığı dosyaların uzantıları “**.tar**” ya da “**.tar.gz**” olmak zorunda değildir. Ancak tar dosyalarını bu şekilde isimlendirmek önemli bir alışkanlığınız olmalıdır. Aylar sonra karşınıza çıkan bir dosyanın tar dosyası olduğunu hatırlamayabilirsiniz; ancak isim verirken “**.tar**” veya “**.tar.gz**” uzantısı verdiğiniz dosyaları ne yapmanız gerektiğini her zaman için hatırlarsınız.



tar programının **c**, **v**, **f** gibi parametrelerinin başına “-” işareti koymak zorunda değilsiniz, yani “**tar cvf home.tar /home**” geçerli bir komuttur.

tar Komutu ile Yedekleme

Tüm akli başında bilgisayar kullanıcıları gibi dosya ve dizinlerinizi yedeklemelisiniz. Bilgisayarınızın diskinin arızalanmasına, bilgisayarınızın olduğu gibi çalınmasına, yanmasına ya da disklerin yanlışlıkla formatlanmasına hazırlıklı olmalısınız.

Kullanıcı olarak kendi dosyalarınızı, periyodik olarak (örneğin her akşam) bir başka disk üzerine kopyalayabilirsiniz. Bir başka bilgisayara kopyalamak

Kim Korkar LINUX'tan?

elbette daha güvenli olacaktır. Artık yedekleyeceğimiz dosyaların değerine göre kendiniz bir strateji geliştirirsiniz nasılsa...

Yedekleme amacıyla bu kopyalama işini sistem yöneticisi sıfatıyla, yani root kimliğiyle yapmanız gerektiğinde sizi küçük bir sorun bekliyor olacaktır:



Tipik bir UNIX bilgisayarda **/home** dizini altında kullanıcıların kişisel dizinleri yer alır. Normal koşullarda her kişisel dizinin ve altındaki dosyaların sahibi farklı kullanıcılar olacaktır. root kullanıcı olarak

```
cp -r /home /disk2
```

komutuyla **/home** dizinindeki herşeyi ikinci disk üzerine kopyalayabilirsiniz, ama **disk2** altına kopyalanan tüm dosyaların ve dizinlerin sahibi root olur!

Bu durumda yedekleri geri indirmezsek gerektiğinde indirilen tüm dosyaların sahiplerini yeniden düzenlemeniz gerekecektir. Zor iş! Oysa, tüm dosya ve dizinlerin baştan gerçek sahiplerinin kimliğiyle yedeklenmesi çok daha anlamlı olurdu. Bu nedenle **cp** komutunu yedekleme için kullanmanızı önermeyiz.

Aslında LINUX işletim sisteminin basit kopyalama komutu olan **cp** programı “**-p**” parametresiyle bu sorunu ortadan kaldıracaktır; yani, değişik kullanıcılara ait dosyaları

```
cp -rp /home /disk2
```

gibi bir komutla (“**-p**” parametresini kullanarak) kopyalarsanız, kopyalanan dosyalar yeni yerlerinde asıl sahipleri ve erişim özellikleriyle birlikte kopyalanır. Ancak bu özellik standart bir UNIX özelliği olmadığı için LINUX dışındaki UNIX türevlerinin hepsinde işe yaramaz. Sistem yöneticisi olarak yedek almanız gerektiğinde **tar** kullanmaya alışmanız daha yararlı olacaktır.

tar komutu yedekleme komutları arasında en kullanışlı olanıdır. Bir **tar** yumağını çözmek üzere gerekli **tar** komutunu “root” kullanıcı kimliğiyle verdiğinizde bile çözülen tüm dosya ve dizinler orijinal sahiplerinin kimliğine uygun olarak çözülür.

tar komutunun “**-f**” parametresini hatırlıyor musunuz? Hani yaratılacak tar dosyasının adını vereceğinizi belirten parametre... Bu parametreden söz

ederken “**tar** komutunu bir **tar** dosyası yaratmadan da kullanmak olasıdır” demiştik. Evet; şimdi bu garip olayı açıklamanın zamanı geldi:

Diyelim ki bilgisayarınıza ikinci bir disk taktınız ve **/home** dizinini bu yeni diske taşımak istiyorsunuz. **/home** dizininizdeki dolu alan 24 GB olsun, yeni diskiniz de 40 GB olsun.

“**cp -r /home /disk2**” komutunun iş görmeyeceğini anlatmıştık. Peki, önce:

```
tar -cvf /disk2/eski_home.tar /home
```

komutuyla **/home** dizininin **/disk2** üzerinde bir tar yumağını oluşturursak, sonra da

```
cd /disk2  
tar -xvf eski_home.tar
```

komutlarıyla bu yumağı **/disk2**'de açsak olur mu acaba?

Aslında olmaz! Çünkü ikinci diskinizde yeteri kadar boş yer yok! 40 GB'lık disk üzerine hem 24 GB'lık bir tar yumağını hem de bu yumağın açılmış halini sığdıramazsınız! **tar** komutunu **-z** parametresiyle birlikte kullanıp tar yumağını küçültmeyi düşünebilirsiniz ama bu taklalara hiç gerek yok! Çözüm UNIX'in zerafetinde yatıyor. Aynı anda iki tane **tar** programı başlatıp, birinin yarattığı tar yumağını arada hiç disk kaydı yapmadan, yani tar dosyası oluşturmadan, ikinci **tar** programına pompalayabilirsiniz:

```
tar -cv /home/cayfer | (cd /disk2; tar -x)
```

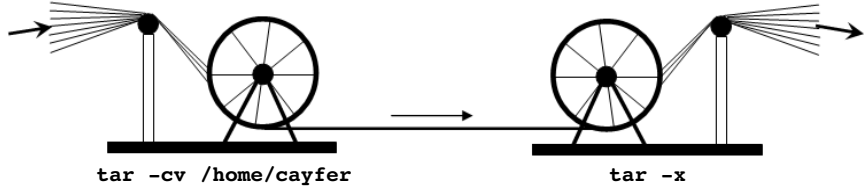
Dikkat ederseniz iki LINUX komutu birlikte başlatılıyor:

1. **tar -cv /home/cayfer**
2. **(cd /disk2; tar -x)**

Bunlardan birincisi (“**tar cv /home/cayfer**”) **-f** parametresi verilmeden kullanıldığı için, yani yaratılacak tar dosyası belirtilmediği için, oluşturacağı tar yumağını standart çıktıya gönderecektir.

İkinci komut bileşik bir komuttur. Noktalı virgülle ayrılmış iki komuttan oluşan ve parantezler arasına yazılmış olan bu bileşik komut da **tar** komutuyla birlikte başlatılacaktır.

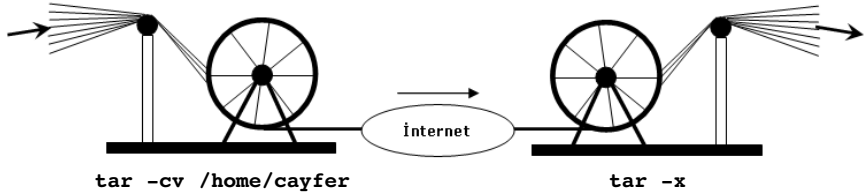
Bu bileşik komut, girdisini “**pipe**” işlemiyle bir önceki komutun çıktısından alacaktır. Bileşik komutun ilk parçası “**cd /disk2**” olduğu için önce çalışma dizini **/disk2** olarak değiştirilecek, sonra da **tar** komutu **x** parametresiyle çalıştırılacaktır. Bu ikinci **tar** komutunda da **-f** parametresi kullanılmadığı için çözülecek tar yumağı standart girişte aranacaktır. Birinci **tar** programı bir yandan yumağı oluştururken ikinci **tar** programı da bu yumağı yeni dizine çecektir. “(**cd /disk2; tar -x**)” bileşik komutu yerine “**tar -x -C /disk2**” komutu da kullanılabilirdi elbette.



İşi biraz daha karıştıralım isterseniz:

Birinci **tar** programını bir bilgisayarda; ikinci **tar** programını da bir başka bilgisayarda (ikisinin de UNIX bilgisayarı olması kaydıyla elbette) çalıştırarak yedeklemeyi bir başka bilgisayar üzerine yapmaya ne dersiniz?

tar -cv /home/cayfer | rsh 192.168.1.2 (cd /yede; tar -x)



Evet, birinci **tar** komutu sizin makinenizde çalışırken ikinci **tar** komutu 192.168.1.2 IP numaralı bir başka makinedeki **/yede** dizini çalışma dizini iken çalışacak. Bu komutun çalışabilmesi için 192.168.1.2 IP numaralı bilgisayarda, sizin bilgisayarınız tarafından komut çalıştırılmasına izin verilmiş olması gerekir ki bu da başka bir bölümün konusudur.

tar Komutunu Kullanırken Dikkat Edilmesi Gereken Noktalar

tar komutunu kullanırken çok tekrarlanan bazı hatalara dikkatinizi çekmek istiyoruz.

- **tar** komutu, **tar** dosyası yaratırken dosya ve dizin ayırımı yapmaz. Parametre olarak verilen dosya kalıbına uyan her şey **tar** dosyasının içine paketlenir.

Dizinler ve alt dizinleri buna dahildir.

- “**tar -cvf yedek.tar ***” komutu (çalışma dizinindeki her şeyi **yedek.tar** dosyası olarak birleştir gibi okunan komut) aslında tam olarak istediğiniz işi yapmayacaktır. Komutun bu şekilde kullanılması durumunda adı “.” (nokta) ile başlayan dosyalar **tar** dosyasına dahil edilmeyecektir. Adları noktayla başlayan dosyaları da paketlemek istiyorsanız tüm dizini tar’lamak zorundasınız. Bir dizinin tümünü paketlerken istemediğiniz alt dizinleri **-exclude** parametresiyle paket dışında bırakabileceğinizi unutmayın.
- **tar** programı, dosya çözerken diskte aynı isimde bir dosya/dizin olsa bile uyardan üzerine yenisini indirecektir. Diskteki eski dosyaların üzerine kayıt yapılmaması için **-k** (*keep*) parametresini kullanabilirsiniz.
- **tar** programı, tar dosyası yaratırken bağlantılı dosyaları (linkleri) kopyalamaz ve bu bağlantıları izlemez. Örneğin **/var/spool/mail** dizini **/disk2/mail** dizinine linkli ise, **/var** dizini paketlenirken **/var/spool/mail** dizininde görünen ama aslında **/disk2/mail** dizininde yer alan dosyalar pakete alınmaz. Alınmasını istiyorsanız **-h** parametresini kullanmalısınız.

rsh Komutu

Bir önceki sayfada, bir dizinin, olduğu gibi bir bilgisayardan bir başkasına transfer edilmesine ilişkin **tar** örneğinde

```
rsh 192.168.1.2 (cd /yedek; tar -x)
```

şeklinde bir komut kullanmıştık.

Adı “Remote Shell” sözcüklerinin kısaltmasından gelen **rsh** komutu, bir bilgisayarın terminalinde çalışırken, başka bir bilgisayarda komut çalıştırıp, o komutun varsa ürettiği STDOUT’a gidecek görüntüyü çalıştığınız terminalle alabilmek için kullanılır.

Örneğin, 168.4.4.2 IP adresli bilgisayarda çalışırken 139.179.211.10 IP adresli bilgisayardaki kişisel dizininizdeki dosyaların listesini görmeniz gerekirse; ikinci bir terminal ekranı açıp, orada 139.179.211.10 bilgisayarına **telnet** veya **ssh** ile bağlanıp **ls** komutunu vermeniz gerekmez. Bu uzun yöntem yerine 168.4.4.2 IP adresli bilgisayarın terminal penceresinde

Kim Korkar LINUX'tan?

```
rsh 139.179.211.10 ls /home/cayfer
```

komutunu verirseniz “**ls /home/cayfer**” komutu, 139.179.211.10 IP adresli bilgisayarda çalıştırılacak, dosya ve dizin listesi, komutu verdiğiniz terminal penceresine listelenecektir.



ssh, telnet yerine kullanılabilir, daha doğrusu kullanılmasını hararetle önerdiğimiz güvenli bir terminal emülasyon programıdır.

Buraya kadar çok iyi; ama önlem alınmazsa, **rsh** komutu insanı dehşete düşürecek bir güvenlik riskini de yanında getirecektir. Düşünsenize; yerini bile bilmediğiniz bir bilgisayardan birileri sizin bilgisayarınıza yönelik olarak

```
rsh 168.4.4.2 /bin/rm -r /etc
```

komutunu verirse neler olur?

İşte bu yüzden **rsh** komutunun çalıştırılabilmesi için özel izin gerekir.

Bu özel izin iki şekilde verilebilir:

1. Her kullanıcı kendi izinlerini kendisi düzenler.
2. Sistem yöneticisi tüm sistem için geçerli izinleri düzenler.

Kullanıcılar kendi kimlikleriyle ilgili izinleri düzenlemek için kendi kişisel dizinlerinde adı **.rhosts** olan bir dosya (dosyanın adının başındaki noktaya dikkat!) hazırlarlar. Bu dosyada, **rsh** komutunun hangi bilgisayarlardan, hangi kimliklerle verilmesi durumunda komutun çalıştırılabileceği belirtilir. Örneğin **cayfer** isimli kullanıcı kendi kişisel dizininde (**/home/cayfer**), içinde

```
139.179.210.4  
www.abc.com.tr  murat  
www.xyz.edu.tr  omer
```

satırları olan bir **.rhosts** dosyası yaratırsa;

- 139.179.210.4 bilgisayarından kullanıcı adı “**cayfer**” olan kullanıcıya bu bilgisayara yönelik olarak “**cayfer**” kimliğiyle **rsh** komutu verme yetkisi verilmiş olur. Ayrıca,

- **www.abc.com.tr** bilgisayarından “**murat**” ve **www.xyz.edu.tr** bilgisayarı-
rından “**omer**” kimliği ile verilen **rsh** komutlarının “**cayfer**” kimliğiyle
çalıştırılmasına izin verilmiş olur.

Eğer bir başka bilgisayardaki tanımlı tüm kullanıcıların sizin bilgisayarınızda-
ki aynı isimle tanımlanmış kullanıcı kimlikleriyle **rsh** komutunu çalıştırmala-
rına izin vermek istiyorsanız her kullanıcı dizinine teker teker **.rhosts** dos-
yası yerleştirmektense bir **/etc/hosts.equiv** dosyası yaratıp içine diğer bil-
gisayarın ya da bilgisayarların IP adreslerini veya açık adlarını yazabilirsiniz.
Örneğin, bilgisayarınızda **/etc/hosts.equiv** dosyası varsa ve içinde

139.179.2.123

abc.xyz.edu.tr

satırları yer alıyorsa, 139.179.2.123 ve abc.xyz.edu.tr bilgisayarında kayıtlı
kullanıcılar, sizin bilgisayarınıza yönelik **rsh** komutu verebilirler. Sizin bilgi-
sayarınızda da aynı adla tanımlı kullanıcılar olması kaydıyla, **rsh** ile gönderi-
len bu komut(lar) sizin bilgisayarda çalıştırılacak, varsa komutun STDOUT’a
göndereceği mesajlar, rsh komutunu veren bilgisayara gönderilecektir.

Hatırlarsanız, daha önce LINUX’ta kullanıcı tanımlarının isimle değil, kul-
lanıcı numarasıyla tutulduğunu söylemiştik. Uzaktan çalıştırılan komutlarda
kullanıcı tanımlaması bunun bir istisnasıdır. Aynı insana ait hesap adının iki
değişik bilgisayarda aynı kullanıcı numarasıyla kaydedilmesini sağlamak ne-
redeyse olanaksız olduğu için, **rsh** izni düzenlemelerinde numara değil, isim
esas alınır.

rsh aslında UNIX’te “**Remote Commands**” (Uzaktan komutlar) olarak anı-
lan bir komut ailesinin bir bireyidir. **rsh** ile aynı mantıkta çalıştırılan ve izin-
leri **.rhosts** ile **/etc/hosts.equiv** dosyalarıyla denetlenen

rcp (*remote copy*)

rlogin (*remote login*)

komutları da birer “**remote command**” olarak kullanılabilir.

İki ayrı bilgisayarda yer alan dosya sistemleri arasında dosya/dizin kopyala-
mak için **r**cp komutunu kullanabilirsiniz. (Tabii ki **.rhosts** veya
/etc/hosts.equiv dosyalarıyla uygun izinlerin verilmiş olması kaydıyla.)

Kim Korkar LINUX'tan?

Örneğin:

```
rcp sunucu:/var/www/html/* /home/cayfer
rcp -r /home/cayfer/public_html/* 139.179.1.1:/var/www/html
```

gibi.

rlogin komutunu ise şifre vermeden bir başka makinedeki aynı isimli hesabınıza bağlanmak için kullanabilirsiniz. Ancak biz **rlogin** yerine **ssh** komutunu kullanmanızı ve şifre girmeye üşenmemenizi öneririz.

xargs Komutu

Kabuk programlarının dosya ismi kalıplarını işlerken karşılaşılabilecekleri bir sorun vardır. Ama bu sorunu bir örnekle açıklamak daha kolay olacak galiba...

Diyelim ki bir dizin içinde 5000 dosya var ve bunların yarısının isimleri ***.log**, gerisi de ***.bak** kalıbında. Siz adı ***.bak** kalıbına uyan dosyaları silmek istiyorsunuz ve **"rm *.bak"** komutunu veriyorsunuz, ardından da **"Argument list too long"** hata mesajını alıyorsunuz.

Aslında şöyle bir düşününce bu hata mesajını almanız son derece normaldir. Siz **"rm *.bak"** komutunu verdiğinizde, kabuk programınız çalışma dizininizde yer alan ve adı ***.bak** kalıbına uyan tüm dosya isimlerini bulup bunları aralarında birer boşlukla komutunuzun **rm** kısmının ardına dizmeye çalışacaktır. Bu kalıba uyan 2500 dosya olsa ve her birinin adı 10 karakter uzunluğunda olsa siz yaklaşık 25 Kbyte uzunluğunda bir komut satırı yazmış gibi olacaksınız. Eh, herşeyin bir sınırı olmalı değil mi? İşte sizin bu komut, **bash** programının komut satırı için ayırdığı tampon alanının dışına taşıdığı için **"too long"** mesajını alıyorsunuz.

UNIX'te her problemin bir çözümü olduğunu farketmişsinizdir artık herhalde. İşte bu sorunun çözümü de **xargs** komutudur.

```
ls *.bak | xargs /bin/rm
```

Yukardaki bileşik komutun ilk parçası çalışma dizininde yer alan adı ***.bak** kalıbına uygun dosyaların isim listesini üretecek (binlerce dosya isminden oluşan bir liste olabilir); ikinci bölümü de bu listedeki her bir dosya için **/bin/rm** komutunu çalıştıracaktır.

Benzeri bir örnek daha: Diyelim ki bir dizin ve altındaki tüm alt dizinlerdeki dosyaların arasından, içinde “**www.bilkent.edu.tr**” karakter dizisi yer alan dosyaları bulmak istiyorsunuz.

İlk akla gelen çözüm:

```
grep "www.bilkent.edu.tr" `find .`
```

komutunu yazmaktır.

Haydaaaa.. Bu da nerden çıktı diyebilirsiniz. Komutun analizi şöyle: Önce **find** komutu yalnızca “.” dan oluşan bir parametreyle başlatılıyor. **find** komutu bu haliyle çalışma dizini ve altındaki tüm dosyaların ve dizinlerin isimlerini listeliyor.

find komutu `` tırnakları arasında yer aldığı için önce bu komut çalıştırılacak ve komutun standart çıktıya gönderdikleri bu tırnaklar arasına yerleştirilecektir. Böylece **grep** komutunun sonuna uzunca bir dosya adı listesi eklenmiş olacaktır.

Buraya kadar her şey iyi; ancak dosya listesi çok uzunsa komut satırı da **bash** kabuğunun sınırlarını aşacaktır ve **grep** komutu çalıştırılmayacaktır.

Çözüm için gene **xargs** önereceğiz:

```
find . | xargs grep "www.bilkent.edu.tr"
```

Aslında yukardaki komutla yapılan iş

```
grep -r www.bilkent.edu.tr *
```

komutuyla da yapılabilirdi; ancak amacımız **xargs** komutuna bir örnek vermektir.

find komutunu anlatırken verdiğimiz “belirli bir kullanıcıya ait tüm dosyaları silme” işini hatırlıyor musunuz? İşte **xargs** ile benzeri bir iş yapan komut:

```
find / -user hasan | grep -v \.dat xargs /bin/rm
```

Yukardaki komut sahibi **hasan** olan dosyalar arasında adında “.dat” geçmeyen dosyaları silecektir.

find komutu sahibi **hasan** olan dosyaları listeleyecektir. Bu liste **grep** programına “-v \.dat” parametresiyle gönderilecek; -v’den dolayı içinde “.dat” geçen satırlar değil; geçmeyenler listelenecektir. Elde edilen bu dosya listesi de **xargs** aracılığıyla **/bin/rm** komutuna gönderilerek dosyalar silinecektir. Bu örnekte “.dat” yerine “\.dat” yazıldığı, yani noktanın işaretlenmiş olduğu dikkatinizi çekmiş olmalı. Bunun nedeni şudur: Eğer “.dat” yazsaydık, bu ifade **grep** tarafından “herhangi bir karakter ve ardından gelen dat” olarak değerlendirilirdi. Böylece adında “**sedat**” geçen dosyalar liste dışında kalırdı. Oysa basit nokta yerine “\.” yazmakla **grep**’e noktayı, “her karaktere uyan” bir joker karakter değil de, bildiğimiz “.” olarak değerlendirmesini istediğimizi belirtmiş olduk.

at Komutu

at [-m] saat

Vereceğiniz komutların belirli bir gün ve saatte başlatılmasını sağlar. Tarih belirtmezseniz “içinde bulunduğumuz gün” kabul edilir. Web sitenizin ana sayfasının 31 Aralık günü tam geceyarısı değişmesini istiyorsanız, önceden hazırlayacağınız **index_yeni.html** dosyasının saat tam 23:59’da **index.html** üzerine kopyalanması için:

```
at -m Dec 31 23:59
```

komutunu kullanarak

```
cd /var/http/html/  
/bin/cp index_yeni.html index.html
```

komutlarını geceyarısına bir dakika kala çalışacak şekilde programlayabilirsiniz.

```
Saat belirtirken    13:30  
                    now + 10 minutes  
                    4pm tomorrow  
                    midnight  
                    Jun 30 4am
```

gibi formlar kullanabilirsiniz.

Tarih vermek istediğinizde kullanmanız gereken form, “**aaa gg**” olmalıdır. Burada **aaa**, ay isimlerinin İngilizcelerinin üç harfli kısaltmalarından biri olmalıdır. (Jan, Feb, ... gibi) **gg** ise doğal olarak gündür... **-m** seçeneği ise, programın başarıyla çalıştırılması durumunda komutu veren kullanıcıya bir e-posta mesajı gönderilmesini sağlar.

at komutunu yukardaki formlardan birinde verdiğinizde belirttiğiniz saatte çalıştırılmasını istediğiniz komutları girmenizi isteyecektir. Bu komutları girmeniz tamamlandığında Ctrl-D tuşuna basarak standart girişten girilecek verilerin bittiğini belirtmelisiniz.

Aşağıdaki örnek **at** komutu, **/usr/local/bin/backup** komutuyla **/dev/hda1** diskinin yedeğini alma işini gece yarısına bir dakika kala başlatılacak şekilde kuyruğa atıyor. Ayrıca bu komutun çalışması bitince **/home/cayfer** dizinindeki “**mesaj**” dosyasını da “**mail ayfer@ieee.org**” komutuyla Ömer’e e-posta olarak gönderiyor.

```

cayfer@notebook.1ojman.bilkent.edu.tr: /home/cayfer - Shell - Konsol <4>
[cayfer@notebook cayfer]# at -m 23:59
warning: commands will be executed using (in order) a) $SHELL b) login shell c) /bin/sh
at> /usr/local/bin/backup /dev/hda1
at> mail ayfer@ieee.org < /home/cayfer/mesaj
at> <EOT>
job 3 at 2003-02-11 23:59
[cayfer@notebook cayfer]#

```

at komutuyla zamanının gelmesini beklemek üzere kuyruğa atılmış işleri **atq** komutuyla görebilirsiniz.

```

cayfer@notebook.1ojman.bilkent.edu.tr: /home/cayfer - Shell - Konsol <4>
[cayfer@notebook cayfer]# atq
1
 2003-02-11 23:59 a cayfer

[cayfer@notebook cayfer]#

```

Bu listede kuyrukta bekleyen işlerin herbirinin bir numarası (7, 8, 9) ve beklediği kuyruğun adı (a) görünür.

Kuyrukta bekleyen işlerin ne olduğunu hatırlamak istediğinizde

```
at -c 1
```

Kim Korkar LINUX'tan?

gibi bir komutla, örneğin 1 numaralı işin ayrıntılı komut dizisini görebilirsiniz.

```
[cayfer@notebook cayfer]# at -c 1
#!/bin/sh
# atrun uid=501 gid=501
# mail cayfer 1
unmask 22
HOSTNAME=notebook.lojman.bilkent.edu.tr; export HOSTNAME
HISTSIZE=1000; export HISTSIZE
OLDPWD=/home/cayfer/post; export OLDPWD
USER=cayfer; export USER
MAIL=/var/spool/mail/cayfer; export MAIL
PATH=/usr/local/bin:/bin:/usr/bin:/home/cayfer/bin; export PATH
PWD=/home/cayfer; export PWD
LANG=en_US; export LANG
HOME=/home/cayfer; export HOME
LOGNAME=cayfer; export LOGNAME
nc=$(\\ \ { \ \ nkdir \ -p \ \ $HOME/.nc/tnp \ 2 \ } /dev/null \ ;"
"\ chmod \ 700 \ \ $HOME/.nc/tnp \ ;"
"\ MC=\ $HOME/.nc/tnp/nc-\ $ \ ;"
"\ /usr/bin/nc \ -P \ \ "$@ \ \ \ \ \ "$MC \ ;"
"\ cd \ \ \ cat \ \ $MC \ \ \ ;"
"\ rm \ -i \ -f \ \ \ $MC \ \ ;"
"\ unset \ MC"
"}; export nc
cd /home/cayfer || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
/usr/local/bin/backup /dev/hda1
mail ayfer@ieee.org < /home/cayfer/mesaj

[cayfer@notebook cayfer]# █
```

Bu ayrıntılı döküm ilk bakışta sizi dehşete düşürebilir. Çünkü sizin yazdığınız komut satırlarının önüne bir çok kabuk komutu yerleştirilmiş olduğunu göreceksiniz. Sistem tarafından eklenen bu komutların espirisi şudur: Kuyruğa attığınız işin çalışma zamanı geldiğinde büyük olasılıkla siz sistemde olmayacaksınız. Dolayısıyla sizin için çalışmakta olan bir kabuk da olmayacaktır. Eh... peki sistem verdiğiniz komutu nasıl bir ortamda çalıştırır? İşte bu baştaki komutlar, işinizin çalıştırılma zamanı geldiğinde sizin yerinize bir kabuk başlatıp sizin kişisel kabuk ortamınızı oluşturmak içindir. Bu laflar hoşunuza gitmediyse boşverebilirsiniz. Alt tarafı gece yarısı çalıştırmanız gereken bir iş olursa diğer işletim sistemi yöneticileri gibi kalkar iş yerine gider, işi adam gibi elle çalıştırırsınız.

at komutuyla ileri bir saatte çalıştırılmak üzere programladığınız bir işi iptal etmek istediğinizde

atrm n

komutunu kullanabilirsiniz. Burada **n**, çalışmaya başlamak için zamanın gelmesini bekleyen işin **atq** komutu tarafından bildirilen sıra numarasıdır. Yani kuyruktaki bir işi iptal etmeden önce **atq** komutuyla o işin kuyruk sıra numarasını öğrenmeniz gerekir.

at komutunun çalışabilmesi için arka planda sürekli çalışan ve geçen zamanı kollayan bir yazılım olmalıdır. LINUX'da bu işi **atd** programı yapar. **at** komutunuzun çalışması için arkada **atd** çalışır durumda olmalıdır. (**at daemon**) Eğer **at** komutunu verdiğinizde **atd**'nin çalışmadığına ilişkin bir hata mesajı alırsanız **atd**'yi siz elinizle başlatabilirsiniz. **atd**'nin bilgisayarınızın her açılışında otomatik olarak başlatılması için birtakım işler yapmalısınız. Bu aşamada yalnızca **atd**'nin elle başlatılmasını göstermekle yetineceğiz:



```
/etc/rc.d/init.d/atd start
```

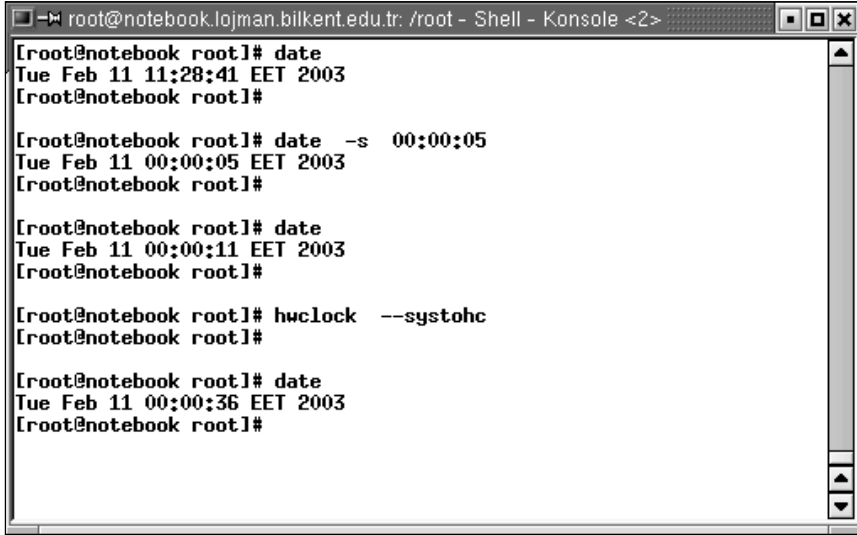
Açılış sırasında otomatik başlatma işini sistem yönetimiyle ilgili bölümlerde öğreneceksiniz.

at komutuyla belirli bir tarih ve saat için programlanan işler, zamanı geldiğinde yalnızca bir kez çalıştırılır. Periyodik olarak, örneğin, her gece yarısı tekrarlanmasını istediğiniz işler için **at** komutundan yararlanamazsınız.

Eğer belirli bir işin “her saat başı”, “her gece yarısı”, “her Pazartesi sabah 09:03’de” otomatik olarak başlatılmasını istiyorsanız, **cron** yazılımından yararlanabilirsiniz. **cron** yazılımı doğrudan bir komutla kullanılmaz. Periyodik olarak yapılmasını istediğiniz işleri **cron** yazılımının ayar dosyası üzerinde yapacağınız düzenlemelerle belirtirsiniz. Bu düzenlemeler için de “**crontab -e**” komutu kullanılır. **cron** hakkında daha ayrıntılı bilgiyi kitabın “**Sistem Yönetimi**” bölümünde bulabilirsiniz.

date, hwclock

Bazen sisteminizin saatinin yanlış olduğunu; biraz ileri gittiğini ya da geri kaldığını farkedersiniz. LINUX’un saatini **date** komutuyla değiştirebilirsiniz ama bu komut bilgisayarınızın saat devresinin ayarını değiştirmeyecektir. İşletim sisteminin tuttuğu saat ve tarihi bilgisayarın saat devresine kaydetmek için **hwclock** komutunu kullanmalısınız.



```
root@notebook.lojman.bilkent.edu.tr: /root - Shell - Konsolle <2>
[root@notebook root]# date
Tue Feb 11 11:28:41 EET 2003
[root@notebook root]#

[root@notebook root]# date -s 00:00:05
Tue Feb 11 00:00:05 EET 2003
[root@notebook root]#

[root@notebook root]# date
Tue Feb 11 00:00:11 EET 2003
[root@notebook root]#

[root@notebook root]# hwclock --systohc
[root@notebook root]#

[root@notebook root]# date
Tue Feb 11 00:00:36 EET 2003
[root@notebook root]#
```

hwclock --systohc

komutu LINUX sistem saatini bilgisayarın saat devresine kaydeder.

hwclock --hctosys

komutu ise bilgisayarın saat / takvim devresindeki zamanı LINUX sistem saati olarak alır.

lynx

Bir senaryo: Yaptığımız bir çalışma için web adresini ezbere bildiğiniz bir yerden bir dosya indirmeniz gerekti. O sırada da Netscape, Konqueror veya Mozilla gibilerinden bir grafik tarayıcınız açık değil. Dosya indireceğiniz sitenin web sayfalarının grafik unsurları da sizi ilgilendirmiyor.

lynx böyle bir durumda çok işinize yarayacaktır. Karakter tabanlı terminal pencereleri için yazılmış küçük, hızlı ama yetenekli bir web tarayıcısıdır **lynx**. Belleğe çabuk yüklenmesi sayesinde indirmek istediğiniz dosyaya hızla erişebilirsiniz.

```

cayfer@cayfer.bilkent.edu.tr: /home/cayfer - Konsole - Konsole
[USEMAP:zenin1.jpg]
[USEMAP:zenin1.jpg]
MAP: http://www.pusula.com/#FPMap0
1. http://www.pusula.com/hakkinda/index.html
2. http://www.pusula.com/kinkorkar/index.html
3. http://www.pusula.com/bilgisayarokulu/default.htm
4. http://www.pusula.com/multimedia/index.html
5. http://www.pusula.com/virgul/default.htm
6. http://www.pusula.com/other/index.html
7. mailto:pusula@pusula.com
8. http://www.pusula.com/english/index.html

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-
Arrow keys: Up and Down to move. Right to follow a link; Left t
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=h

```

cut

cut, **more** ve **less** komutları basit metin dosyalarını ekrana görüntülemek için oldukça kullanışlı komutlar olmakla birlikte, bazen dosyaları olduğu gibi listelemek yerine bazı satırların bazı alanlarını listelemek de isteyebilirsiniz.

Örneğin **/etc/passwd** dosyasının yalnızca hesap ismi ve kullanıcı ismi alanlarından oluşan bir listeye gereksinim duyduğunuzda

```
cut -d: -f 1,5 /etc/passwd
```

komutuyla “:” ile ayrılmış olan alanlardan yalnızca birinci ve beşincileri listelebilirsiniz.

```
cayfer@cayfer.bilkent.edu.tr: /home/cayfer - Shell - Konsole
[cayfer@cayfer cayfer]$ cut -d: -f 1,5 /etc/passwd
root:root
nobody:Nobody
cayfer:Can Ugur Ayfer
oner:Oner Ayfer
sener:Melih Sener
paul:Paul Nicholls
tulin:Tulin Bakiler
dundar:Dundar Kara
yturk:Keren Yurekli
bcc:Generic User
dost:Doga Sporlari Klubu
listmaster>List Master
dunkley:Ed Dunkley
cagan:Cagan Turhan
ayfer:Regyan Ayfer
[cayfer@cayfer cayfer]$
```

Kolay incelenebilmesi için gerçek örnek sadeleştirilmiştir.

cut komutuna **-d** parametresiyle satırlardaki alanları ayıran araç tanıtılır. Araç belirtmezseniz araç karakteri olarak <Tab> kabul edilir. İlk 5 alanı listelemek istediğinizde “**-f1, 2, 3, 4, 5**” parametresi yerine “**-f 1-5**” parametresini kullanabilirsiniz.

Elde ettiğiniz liste çok uzunsa, tüm LINUX komutlarında olduğu gibi **cut** programının standart çıktıya gönderdiklerini **more** veya **less** programına yönlendirebilirsiniz.

tee

Bazı durumlarda bir programın çıktısını iki değişik yere yönlendirmeye gereksinim duyabilirsiniz. Örneğin uzun bir dosyayı sıralayıp, sıralı dosyayı hem **less** ile sayfa sayfa görüntülemek hem de bu sıralı kayıtları bir dosyaya yönlendirmek isteyebilirsiniz.

```
sort adresler | tee adresler.sirali | less
```

komutu “**adresler**” dosyasını sıralayarak sıralı halini **tee** programına yönlendirilecek; **tee** programı standart girdisinden gelen verileri hem **adresler.sirali** dosyasına hem de kendi standart çıktısına yönlendirecek; son olarak da “**less**” komutu bu satırları sayfa sayfa listeleyecektir.

script

Diyelim ki çok sayıda komuttan oluşan bir dizi iş yapmak ve bu komutları ve aldığınız yanıtları sonradan incelemek üzere saklamak istiyorsunuz. **script** programı, sizi bu iş boyunca, hangi komutu verdiğiniz ve ne yanıt aldığınızı kağıda not etmekten kurtaracaktır.

```
script /tmp/log1
```

komutu, verildiği andan başlayarak Ctrl-D tuşuna basıncaya kadar verdiğiniz tüm komutları ve bu komutlara ilişkin STDOUT'a gönderilen herşeyi **/tmp/log1** dosyasında saklar. **script**'i Ctrl-D ile durdurduktan sonra bu **/tmp/log1** dosyasını bir anlamda "kaptanın seyir defteri" olarak kullanabilirsiniz.

script komutunu

```
script -a /tmp/log1
```

şeklinde kullanırsanız, **seyir** kayıtları **/tmp/log1** dosyasının olası eski içeriğine eklenir.

split

Bazen büyük dosyaları küçük parçalara ayırmak zorunda kalırsınız. Örneğin 50 Mbyte uzunluğunda bir dosyayı bir başkasına göndermek için e-posta'dan başka olanağınız yoksa, tek çıkar yol dosyayı e-posta sunucularının kabul edeceği büyüklükte (tipik olarak 10 Mbyte) parçalara bölüp öyle göndermektir.

split komutu, dosyaları isterseniz belli büyüklükte parçalara, isterseniz belli sayıda satır içeren parçalara bölebilir.

```
split -b 1m uzun_dosya parca_
```

uzun_dosya isimli dosyayı birer megabyte'lık parçalara ("**-b 1m**" parametresi) bölecektir. Parçaları oluşturan dosyaların isimleri de sırasıyla **parca_aa**, **parca_ab**, **parca_ac**, ..., **parca_az**, **parca_ba** gibi isimler olacaktır.

```
cayfer@notebook.lojman.bilkent.edu.tr: /home/cayfer - Shell - Konsol
[cayfer@notebook cayfer]$ split -b 2n uzun_dosya parca_
[cayfer@notebook cayfer]$ ls -al parca_*
-rw-r--r-- 1 cayfer cayfer 2097152 Şub 11 02:33 parca_aa
-rw-r--r-- 1 cayfer cayfer 2097152 Şub 11 02:33 parca_ab
-rw-r--r-- 1 cayfer cayfer 2097152 Şub 11 02:33 parca_ac
-rw-r--r-- 1 cayfer cayfer 2097152 Şub 11 02:33 parca_ad
-rw-r--r-- 1 cayfer cayfer 2097152 Şub 11 02:33 parca_ae
-rw-r--r-- 1 cayfer cayfer 1421240 Şub 11 02:33 parca_af
[cayfer@notebook cayfer]$ █
```

Öte yandan,

```
split -l 1000 uzun_dosya parca_
```

komutu, **uzun_dosya** isimli dosyayı 1000'er satırlık parçalara (“-l 1000”) bölecek; parçaları **parca_aa**, **parca_ab**, **parca_ac**, ..., **parca_az**, **parca_ba** gibi isimlendirecektir. Satır uzunlukları eşit değilse, doğal olarak parça dosyaların büyüklükleri de byte ölçüsüyle aynı olmayabilecektir.

Daha sonra, bu parçaları tekrar birleştirmeniz gerektiğinde

```
cat parca_* > uzun_dosya
```

komutunu kullanabilirsiniz. Kabuk programınız “**parca_***” dosya isim kalıbını, bu kalıba uyan dosyaların isim listesine dönüştürürken alfabetik sıralama kullanacağı için parçalanmış dosyalar doğru sırada birleştirilecektir.

BUNLARI BİLİYOR MUYDUNUZ?

İşletim sistemi olmayan UNIX'ler

“UNIX” markasının kullanıldığı ilginç bazı ürünleri araştırırsanız:

- mikrodalga fırın uyumlu gıda saklama kapları
- yangın söndürücüsü
- ahşap kitap raf sistemi
- tükenmez kalem
- çamaşır askısı
- kimyasal tiner
- buğday gevreği
- televizyon anteni
- masaj aleti
- gözlük çerçevesi
- saç kurutma makinesi
- otomobil parçaları
- çocuk bezi
- kiralık oto hizmetleri
- bar, berber salonu
- mobilya

gibi ürün ve hizmetler bulabilirsiniz. (Ayrıntılar için bkz.
<http://cm.bell-labs.com/cm/cs/who/dmr/otherunix.html>)

Kim Korkar LINUX'tan?