LİNUX KOMUT SATIRI

Kemal Demirez
kemaldemirez@gmail.com
facebook.com/kemaldemirez0
twitter.com/kemaldemirez

EYLÜL 2013

İÇİNDEKİLER

Giriş

Linux ile ilgili bir çok kaynak bulabilirsiniz. Hem basılı kaynaklar hem de net üzerinde pdf, doküman vs. şeklinde bir çok kaynaktan faydalanabilirsiniz. Ben de Linux'a gönül vermiş ve öğrenmek isteyen kişilere küçük bir katkı yapmak amacıyla bu kaynağı hazırlamak istedim. Tabi bilenler bilir, ben daha çok güvenlik konularında çalışmalar yapmaya çalışıyorum. Fakat Linux kullanmayı ve özellikle de güvenlik profesyonelleri ve güvenlikle ilgili amatör kullanıcılar için bir dağıtım olan **BackTrack** ve onun devamı olan **Kali Linux**'u kullanmayı daha çok seviyorum. Bu çalışmada verilen örnekler de Kali Linux dağıtımı üzerinden olacaktır. Amacım Linux işletim sisteminin (daha doğrusu **GNU/Linux**) tamamını anlatmak değildir. Çalışma başlığında yazdığı gibi komut satırında Linux kullanımına örnekler verilecektir. Bu örnekler hemen tüm dağıtımlarla büyük ölçüde aynı olacaktır. Dağıtımlar hangi tabanda olursa olsun genel olarak komut kullanımı küçük farklar haricinde aynıdır. Farklı olan dağıtımın kullandığı paket yapısı, konfigürasyon dosyalarının yerleri, kullanılan masaüstü ortamı gibi şeylerdir.

Kali Linux, **Debian** (**Debian Wheezy**) tabanlı bir dağıtım olduğundan verilen örnekler tüm Debian tabanlı dağıtımlarda geçerli olacaktır. **RedHat**, **Fedora** ve **CentOS** gibi dağıtımlarla da büyük oranda benzerlik bulunacaktır. Bu sebeple burada verilen bilgiler iyi öğrenildiğinde Linux dağıtımlarının (distro) %90 ı rahatlıkla kullanılabilecektir.

Anlatım diline gelince, farkettiğiniz gibi rahat ve sade bir anlatım dili tercih edilecek ve çalışmanın kendine özgü bir düzeni olacaktır. Standart bir düzenden ziyade, belli konular anlatılırken eğer o konuyla bağlantısı olan bir başka konu varsa, ondan da kısaca bahsedilecektir. Amaç, meramın en iyi ve anlaşılır şekilde ifade edilmesidir. Bazı kullanıcılar burada anlatılan konuların bir çoğunu bilebilirler. Fakat bilmeyen kişiler için faydalı bir çalışma olduğunu düşünüyorum. Bilenler de bilgilerini tazeleyip, ellerinin altında bulundurmak isteyebilirler.

Kapakta da görüldüğü gibi sosyal medyada hesaplarım ve bir de mail adresimden başka dikili bir ağacım bulunmamaktadır.

Çok fazla teknik terim kullandığımı zanneden okuyucular hayal kırıklığı yaşayabilirler. Yukarıda da ifade ettiğim gibi sade basit ve rahat bir anlatımı tercih ettim. Umarım bu çalışma tüm Linux kullanıcılarına ve kullanıcı adaylarına faydalı olur. Evet bu kısa açıklamadan sonra asıl konumuza geçebiliriz. İyi okumalar.

Linux Terminali (Konsol)

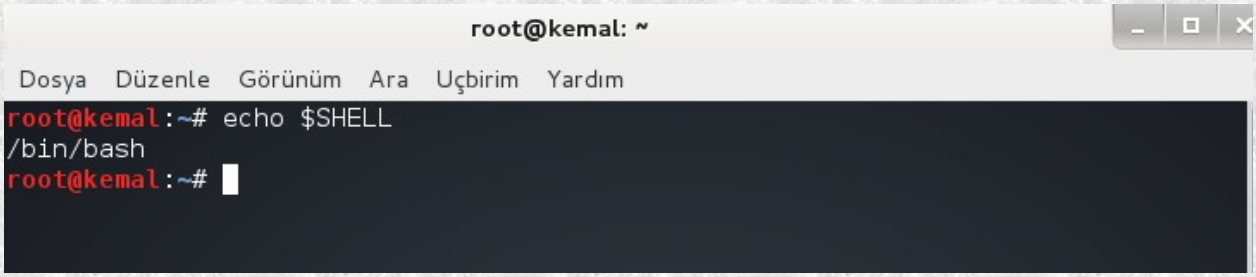
Bilindiği üzere **kernel** yani **çekirdek** işletim sisteminin kalbidir. En önemli vazifesi donanımla yazılımın haberleşmesini sağlamaktır. Ömemli görevlerinden bir diğeri de sistem belleğini

ve sistemde çalışan **süreçleri** denetlemektir. Evet şimdi klasik olarak donanım-yazılım-kullanıcı ilişkilerinde (biraz karmaşık bir ilişki) biz bir kullanıcı olarak isteklerimizi nasıl donanıma ve bir yönüyle de yazılıma ileteceğiz? Tabi konumuz Linux komut satırıyla ilgili olduğundan cevabımız da ona göre olmalı. Yani Windows sistemin grafiksel kullanıcı ara yüzü ile zaten her şeyi grafiksel olarak halledebiliyoruz. Aslında Linux gui den de yani grafiksel kullanıcı ara biriminden de Windows'da yaptığımız aynı şeyleri yapıyoruz. Fakat arka planda çalışan komutlar aslında isteklerimizi yerine getiriyor. Yani ister **MS-DOS** komut satırı ister **Linux terminalden** komutları vererek ister de grafiksel ara yüz kullanarak işletim sistemi-program-donanıma istediğimiz şeyleri yaptırırken aslında olan şey arka planda komutların çalışmasından ibarettir. **Komut satırı** ise bu işlemleri biraz daha görünür hale getirmekte ve bize çok fazla esneklik sağlamaktadır. Ayrıca bütün programların da görsel ara yüzü olacak diye bir kural bulunmamaktadır. **Linux konsol** (**terminal** ve **uçbirim** de deniyor) basit bir komut satırı olmanın çok ötesinde işler yapmaktadır. **Shell** yani **kabuk** dediğimiz bir programla terminalden verdiğimiz komutlar yorumlanarak istediğimiz işler yaptırılmaktadır.

Shell (Kabuk)

Shell, bir nevi kullanıcı ile **çekirdek** (**kernel**) arasında yer alarak, kullanıcıdan gelen komutları yorumlayarak çekirdeğe iletmektedir. Kernel ile shell arasında da adına **sistem çağrıları** denen yakın bir ilişki bulunmaktadır.

Günümüzde en yaygın kullanılan kabuk (shell) programı **bash** kabuk programıdır. Sistemimizde kullanılan kabuk programını görmek için konsoldan (terminalden) **echo $SHELL** komutu verilmelidir.

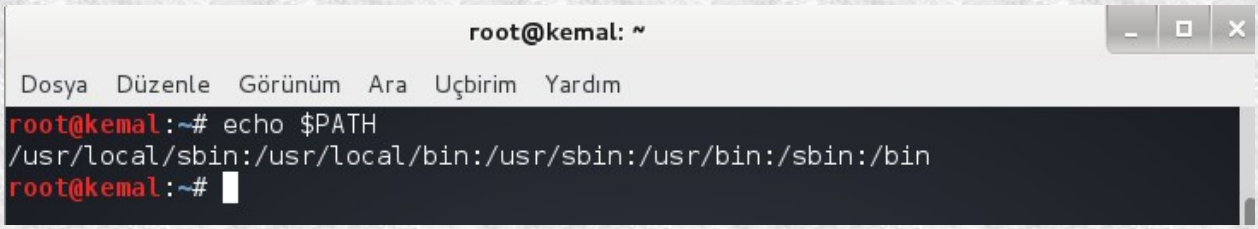


Evet sayın seyirciler, pardon yani okuyucular..çaktırmadan konsoldan bir komut kullanmışız bile. Olsun elimiz alışsın.

Biraz önce ne demiştik? Konsoldan bir komut verildiğinde, bash kabuk programı bu komutu yorumlar ve çalıştırır. Peki verilen komutu çalıştırırken bunu nerelere bakarak yapar? Bunu öğrenmek için de yine bir komut kullanmalıyız. Az aşağıda :)

Ortam Değişkenleri

Kullanmamız gereken komut **echo $PATH** komutudur.



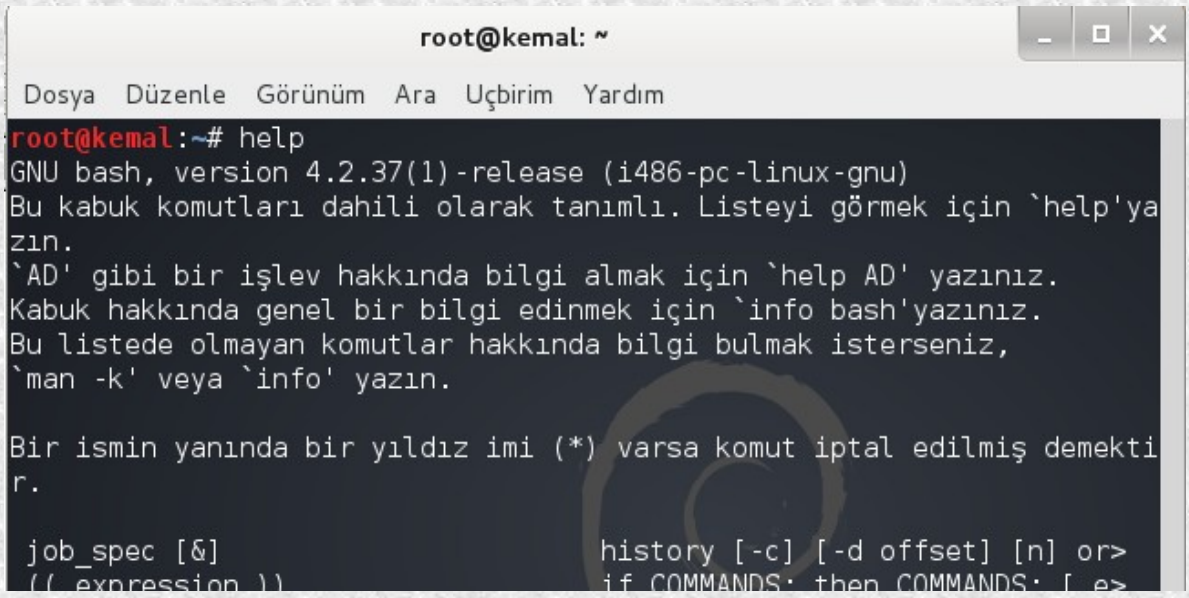
Komut çıktısı bize ne anlatmaktadır? Demek istediği şudur: Sen bir komut verdiğin zaman ben sırasıyla bu dizinlere bakarım. Eğer verdiğin komutla ilgili çalıştırılabilir dosya bu dizinlerin altındaysa çalıştırırım. Yoksa çalıştıramam arkadaş.

Görüldüğü gibi çalıştırılabilir dosyalar **/bin** ve **/sbin** altında.Yukarıdaki ekran çıktısında görülen ve birbiriyle **:** işaretiyle ayrılmış dizinlerin oluşturduğu yapıya ortam değişkenleri denir. Yani bir komut verildiğinde, **ortam değişkenlerinde** görülen sıraya göre dizinler kontrol edilecek ve komutla ilgili program dosyası bulunduğunda program kabuk tarafından belleğe yüklenecek ve çalıştırılacaktır.

Bu kısa bilgilerden sonra şimdi biraz daha işin içine girelim. Öncelikle konsolda komut kullanırken nelere dikkat etmeli ondan bahsedelim. En dikkat edilecek şey komutu doğru girmektir. Bunun için de bilinmesi gereken şey, Linux konsolun büyük-küçük harf duyarlı olduğudur. Yani **Help** ile **help** komutu aynı şey değildir. Komutu doğru girmek ve doğru **parametrelerle** kullanmak için bizzat sistemimiz bize yardımcı olmak için hazır beklemektedir.

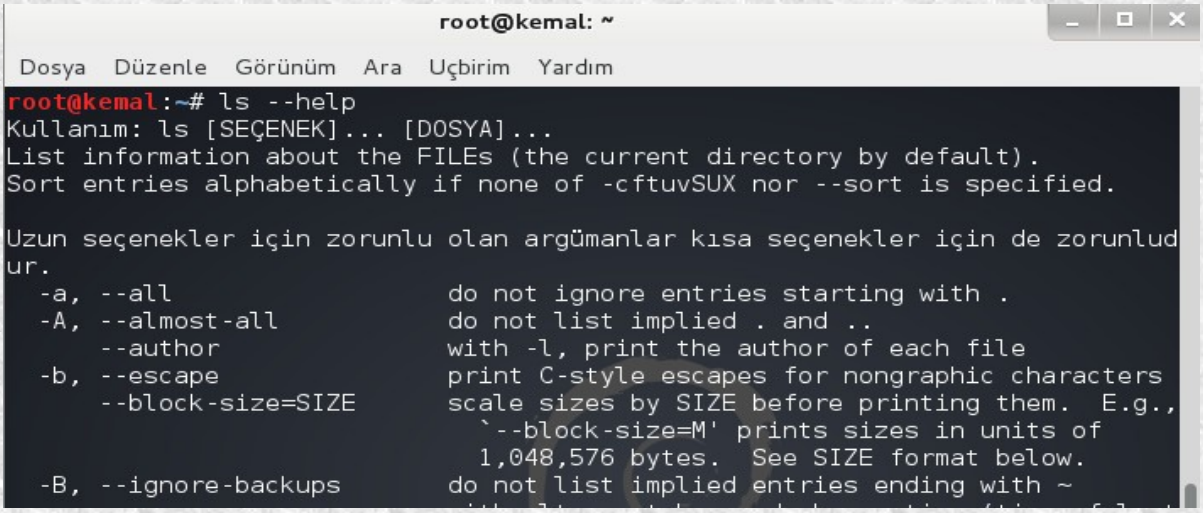
Yardım Alma

Yardım almak için kullanabileceğimiz bazı komutlar vardır. Bu komutlardan en basiti **help** komutudur.



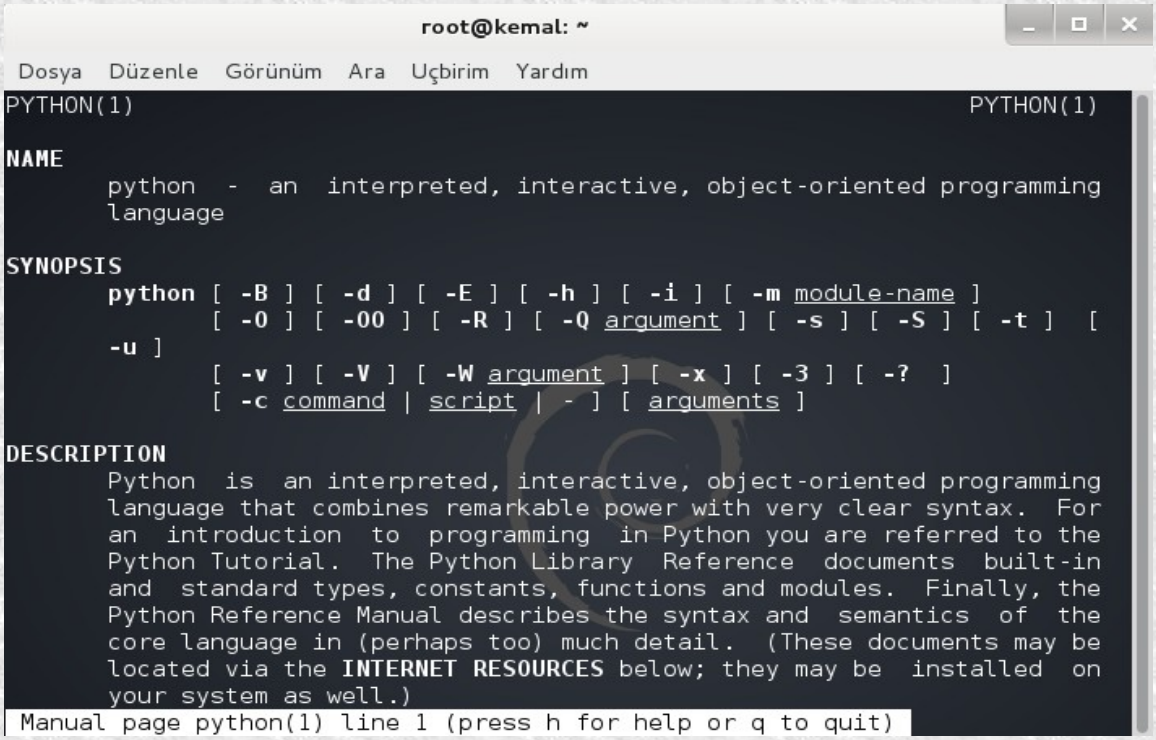
Komutumuzu parametresiz kullandığımız zaman yukarıdaki gibi

bir çıktı alırız. Zaten orada da anlatmış bize nasıl kullanacağımızı. Şimdi basit bir örnek yapalım. **help** komutuyla yardım alırken **[yardım_alınacak_komut] --help** (bazen de **-h**) şeklinde kullanılır.



Şimdi Linux'un çok güzel bir özelliği olan **kılavuz sayfaları** özelliğinden bahsedelim. Linux'da manual kelimesinin kısaltılmışı olan **man** komutunu kullanarak, her hangi bir komutla-programla ilgili geniş bilgi alabilirsiniz.

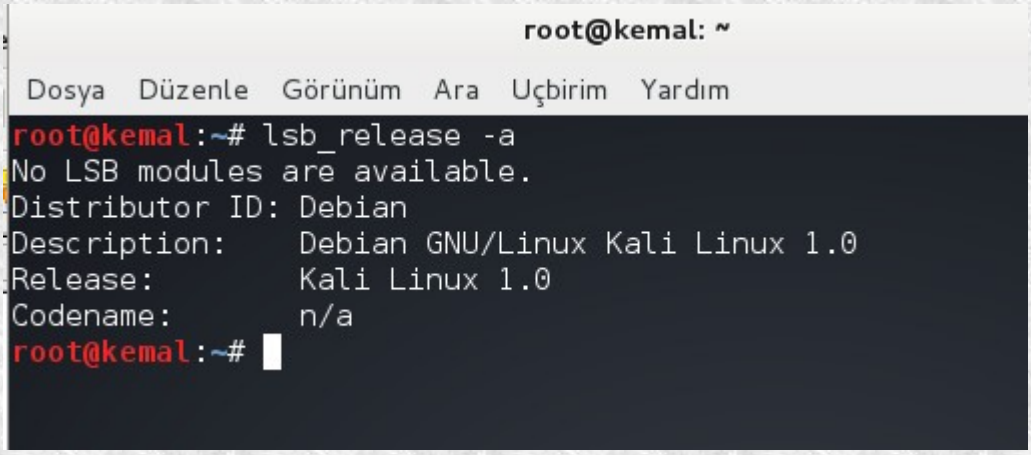
Örneğin **#man python** komutunun çıktısı aşağıdaki gibi olacaktır.



Klavuz yardım sayfasından çıkmak için **q** tuşuna basmanız yeterli olacaktır.

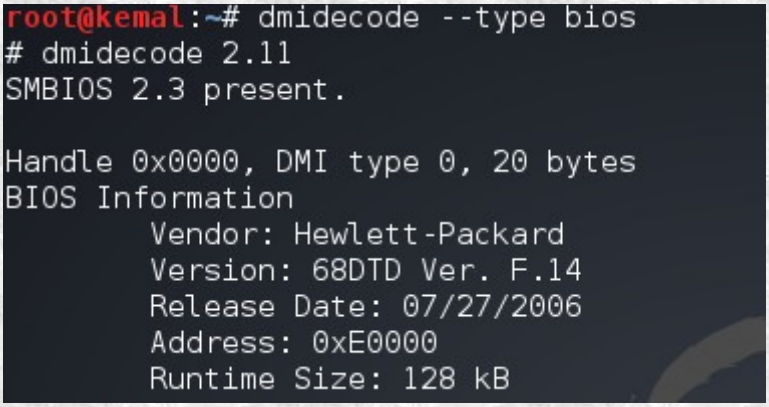
Dağıtım Adı Öğrenme

Dağıtımla ilgili bilgiyi görmek için **lsb_release -a** komutunu kulanabiliriz.



Sistemle İlgili Bilgi Alma

Sistemimizle (**bios**, **memory**, **cache** vs.) ilgili bilgi almak için de **dmidecode** komutunu kullanabiliriz. Komutu parametresiz kullandığımızda uzun bir çıktı verebilir. Fakat örneğin bios ile ilgili bilgi almak istiyorsak type parametresiyle birlikte **dmidecode --type bios** komutunu kullanmalıyız.



Aynen bu şekilde **--type** kullanarak

system
baseboard
chassis
processor
memory
cache
connector
slot

Bilgisi de alabiliriz. Sistemi kurcalamaya devam edelim. Donanımla ilgili geniş bilgi almak için **lshw** (hardware lister) komutunu kullanabiliriz. Bazı dağıtımlarda kurulu gelmeyebilir. Sisteminize kurarak kullanabilirsiniz. İlk önce komutu deneyelim. Eğer sistemde kuruluysa zaten çalışacaktır.

```
root@kemal:~# lshw
bash: lshw: komut yok
root@kemal:~#
```

Evet gördüğünüz gibi benim sistemimde kurulu değilmiş. Bakalım **depolarda** var mı?

```
root@kemal:~# apt-cache search lshw
lshw - information about hardware configuration
lshw-gtk - graphical information about hardware configuration
root@kemal:~#
```

Evet depolarda var. İstersem **#apt-get install lshw** komutuyla kurup kullanabilirim. (**Debian** tabanlı dağıtımlarda böyle, eğer **RedHat/Centos/Fedora** tarzı bir dağıtım kullanıyorsanız kullanmanız gereken komut **#yum install lshw** olacaktır.)

Buna benzer konulara ilerde yeri geldikçe değineceğiz. Ben şimdilik kısaca bahsettim.

Kernel (Çekirdek) Versiyonunu Öğrenme

Kernelden (çekirdekten) bahsettik hatırlarsınız. Şimdi bakalım sistemimiz hangi kerneli kullanıyor?

```
root@kemal:~# uname -a
Linux kemal 3.7-trunk-686-pae #1 SMP Debian 3.7.2-0+kali8 i686 GNU/Linux
root@kemal:~#
```

Disk Bölümlerini Görme

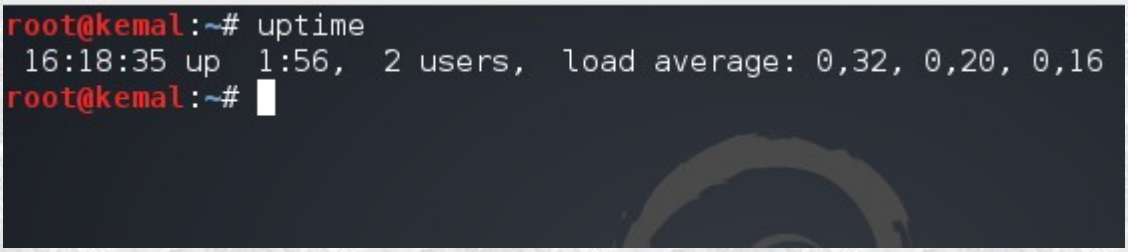
Disk bölümlerimizle ilgili bilgileri görmek için de **fdisk -l** komutunu kullanabiliriz.

```
root@kemal:~# fdisk -l

Disk /dev/sda: 40.0 GB, 40007761920 bytes
255 heads, 63 sectors/track, 4864 cylinders, total 78140160 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002549d

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    76081151    38039552   83  Linux
/dev/sda2        76083198    78139391     1028097    5  Extended
/dev/sda5        76083200    78139391     1028096   82  Linux swap / Solaris
root@kemal:~#
```

Gördüğünüz gibi çok mütevazi (gariban da denebilir) bir sistemim var :)

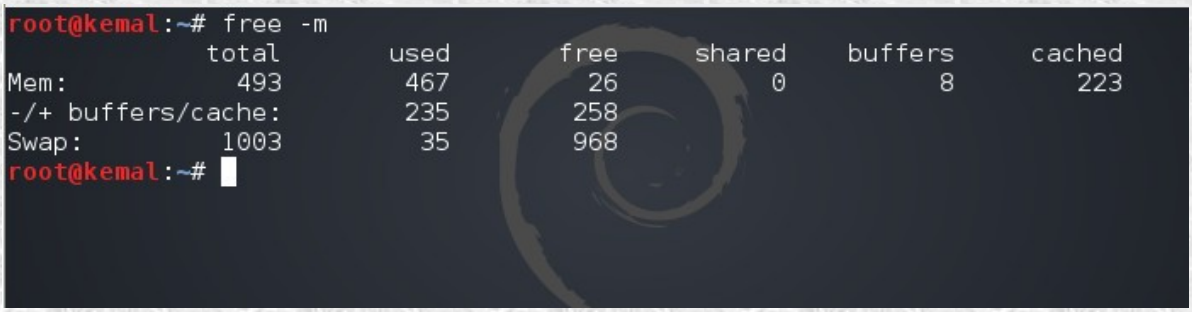
Uptime

Merak ettim acaba sistemim ne kadar zamandır açık?



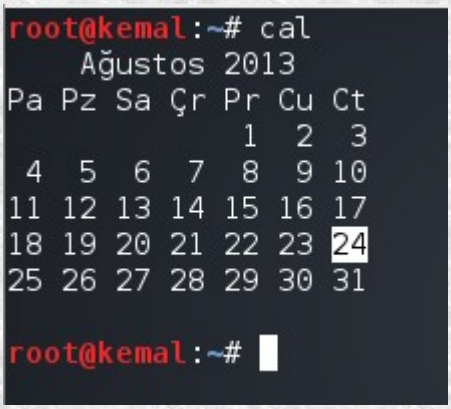
Kullanılan Bellek Miktarı

Sistemimizde kullandığımız bellek miktarını görmek için de **free** komutunu kullanabiliriz. Komutu **free -m** şeklinde kullanırsak çıktıyı **MB** olarak alırız.



Takvim

Madem onu bunu merak ediyoruz o zaman takvimi de merak edelim.

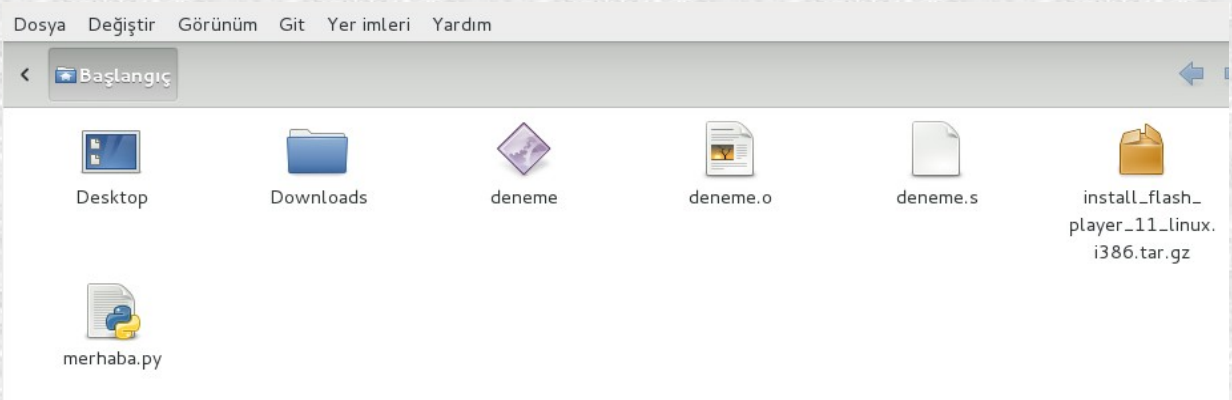


Her hangi bir yıla ait tüm takvimi de görebiliriz. Bunun için aşağıdakine benzer bir komut kullanabilirsiniz. Doğum tarihinin hangi güne denk geldiğini öğrenmek isteyenler için faydalı bir hizmet :)

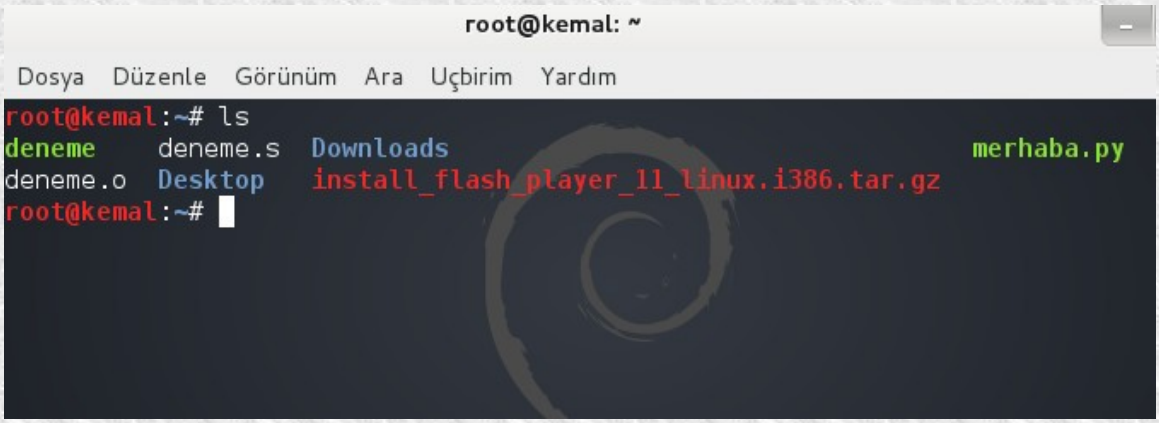
```
root@kemal:~# cal 2014
                                2014

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

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

Dizinler Arasında Gezme ve İçeriğini Listeleme (cd, ls)

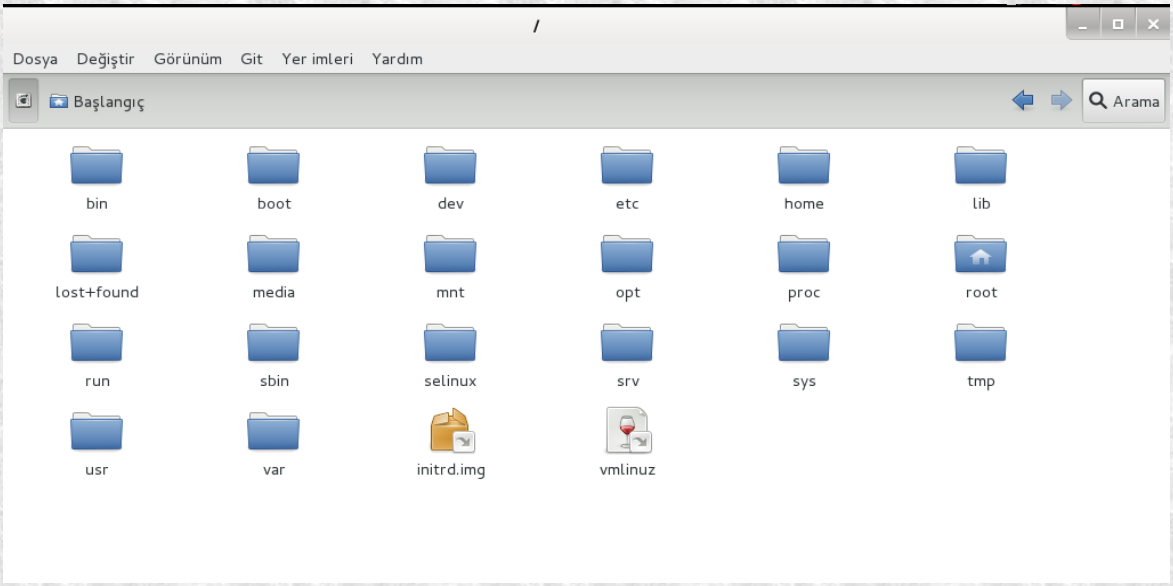
Evet buraya kadar küçük çaplı ısınma turları yaptık. Hazır ısınmışken şimdi de dizinler arasında gezinme ve dizin ve alt dizinlerdeki dosyaları listeleme komutlarına bakalım. Öncelikle **dizinden** bahsedelim. Linux dünyasında her şey (programlar, dosyalar, aygıtlar vs.) bir dizin altında alt dizin/dosya görünümünde yer alır. Bütün dizinler/alt dizinler **hiyerarşik** şekilde **kök dizine** (**root dizin**) bağlıdır. Root dizin / ile gösterilir. Burada önemli bir nokta; root dizinle root kullanıcı karıştırılmamalıdır. Biri dizin biri kullanıcı değil mi? :)
Bunu neden söyledim? Dizin meselesinden bahsederken yine Linux'un bir özelliği olan daha doğrusu geleneği olan **home** (ev) dizininden (klasöründen de denebilir) bahsetmeliyiz. Linux sistemlerde root kullanıcı da dahil her kullanıcının kendi home klasörü (**ev dizini**) bulunmaktadır. Konsol varsayılan olarak kullanıcının **kendi ev dizininde** çalışmaya başlar. Şimdi root dizinle root kullanıcıya ait olan ev dizinine (klasörüne) göz atalım. Öncelikle grafiksel olarak görelim.



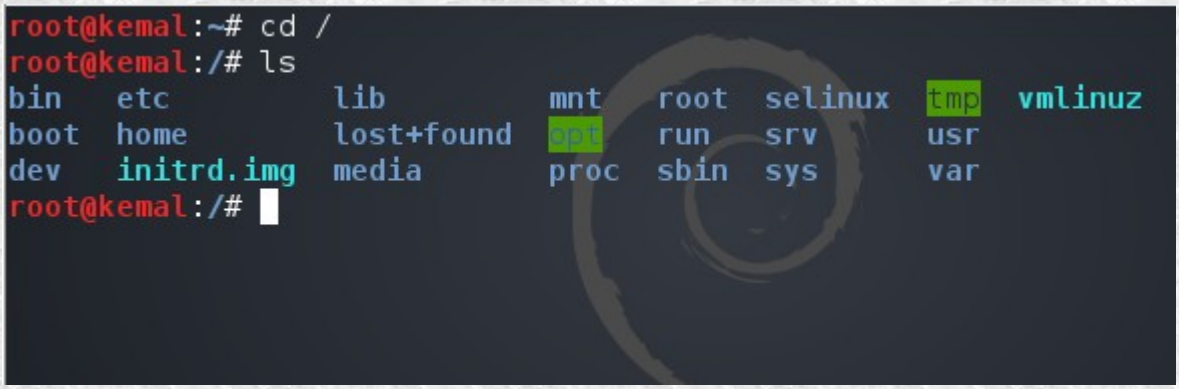
Evet üstte gördüğünüz benim (root kullanıcıyım ya) ev klasörüm. Bunu bir de konsoldan görelim.



Şimdi de root (kök) dizine görsel olarak bakalım. Görsel olarak lafından kastım **dosya yöneticisini** kullanarak kök dizini (yani / dizinini) açmaktır. Evet root dizin aşağıdakine benzer şekilde görünecektir.



Dikkatlice incelerseniz yukarıda / işaretini göreceksiniz. Ayrıca dizinlere(**klasörlere**) baktığınızda üzerinde ev simgesi olan (olmayabilir de) root dizinini(klasörünü) göreceksiniz. İşte bu dizin **root** kullanıcıya ait **ev dizini**. Bundan başka bir de **home** klasörü görüyoruz. Bu klasör(dizin) de sisteme kullanıcı eklediğimizde **yeni kullanıcıların kişisel klasörlerinin ekleneceği** dizindir. Örneğin sisteme **zeynep** kullanıcısını ekledik diyelim. Bu durumda zeynep kullanıcısının ev klasörü **/home/zeynep** şeklinde oluşturulacaktır. Şimdi root dizine bir de konsoldan bakalım.



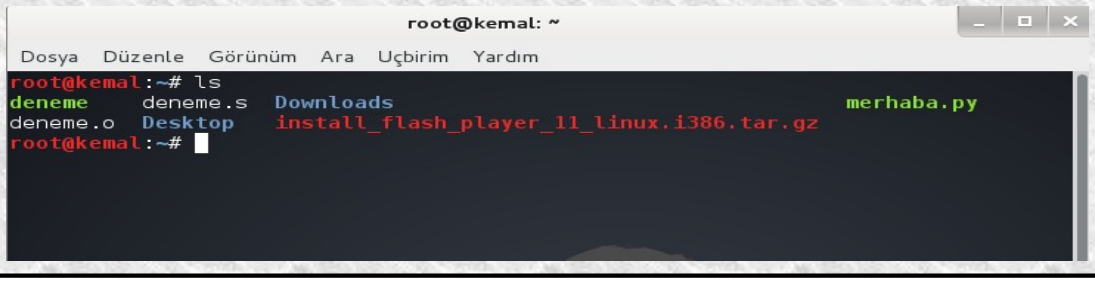
Yukarıdaki ekran görüntüsünü incelediğinizde bir kaç şey dikkatinizi çekmiş olmalı. Öncelikle **cd /** komutuyla / dizinine geçiyoruz. Sonrasında da **ls** komutuyla altında bulunan dosya ve dizinleri **listeliyoruz**. Başka bir şey de kullanıcı kendi ev klasöründe çalışırken **promptta** ~ işareti, başka dizine geçip o dizinde çalışmaya başladığında da o dizinin (örneğin /) ismi yer alır.

ls ve **cd** komutlarıyla ilgili örneklere devam etmeden önce, o an hangi dizinde çalıştığımızı öğrenmemize yarayan **pwd** komutundan bahsedelim. Söylediğim gibi pwd komutunu kullanarak o an çalışılan dizin görülebilir.

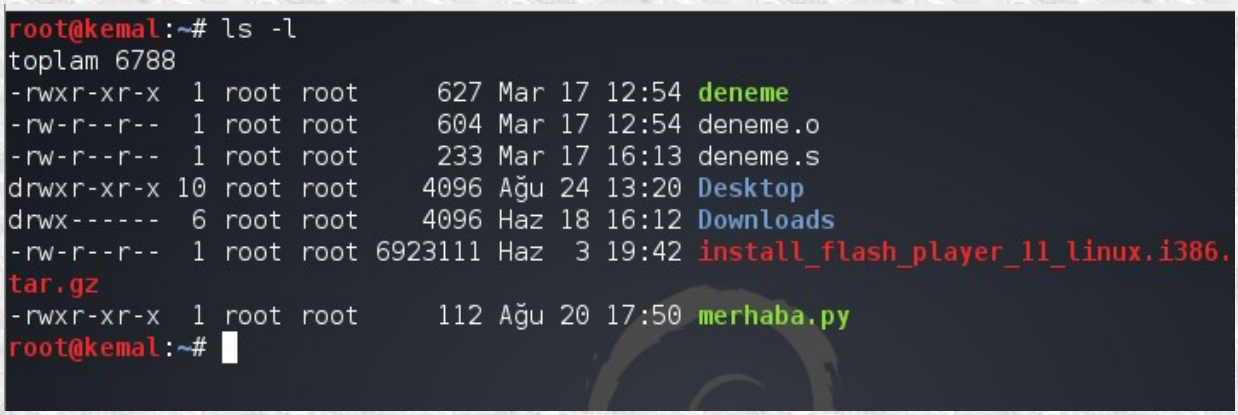
Evet yukarıdaki ekran görüntüsüyle ne demek istediğimizi özetlemiş olduk sanırım.

Şimdi kaldığımız yerden devam edelim ve hem cd hem de ls komutuyla ilgili örnekler verelim. Tekrar hatırlatalım, cd komutuyla dizinler arasında gezebiliriz, ls komutuyla da bu dizin-klasörler altındaki alt dizin/klasör ve dosyaları görüntüleyebiliriz.

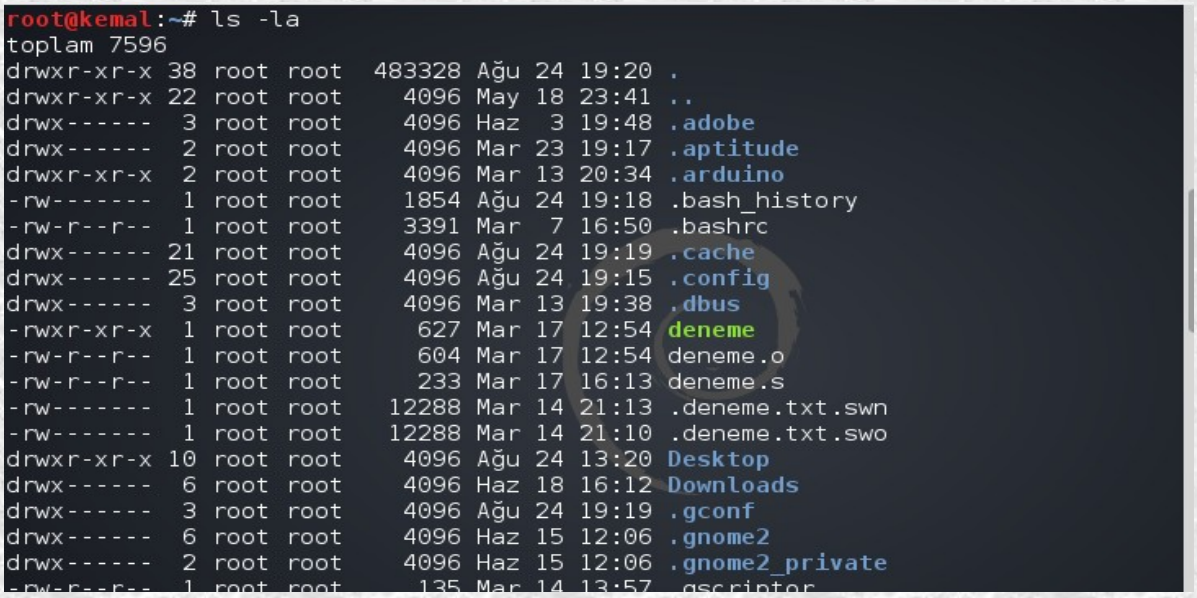
ls komutunu **parametresiz** şekilde kullanırsak aşağıdakine benzer bir çıktı alırız.



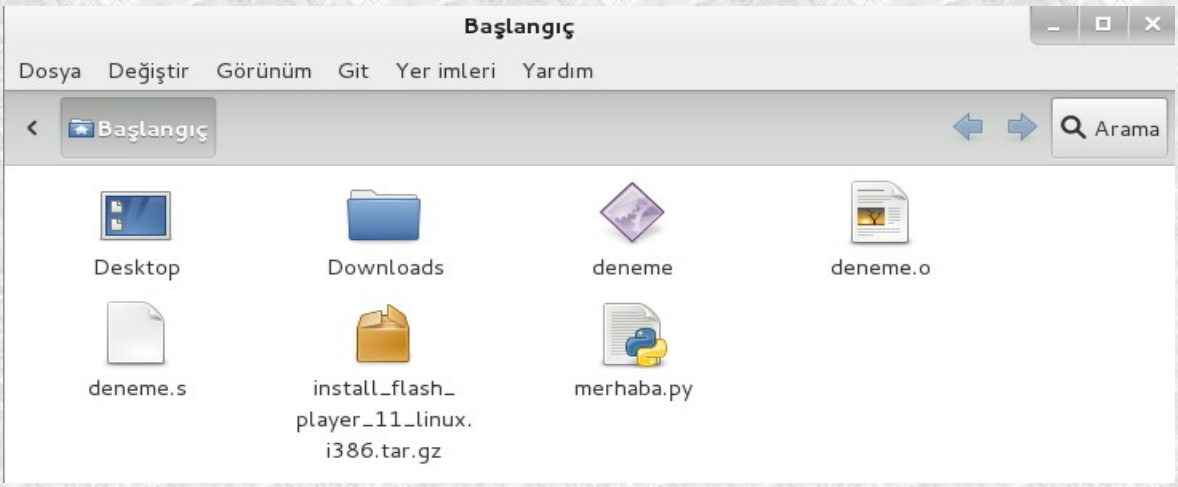
Eğer **ls -l** şeklinde kullanırsak çıktı aşağıdaki gibi olacaktır.



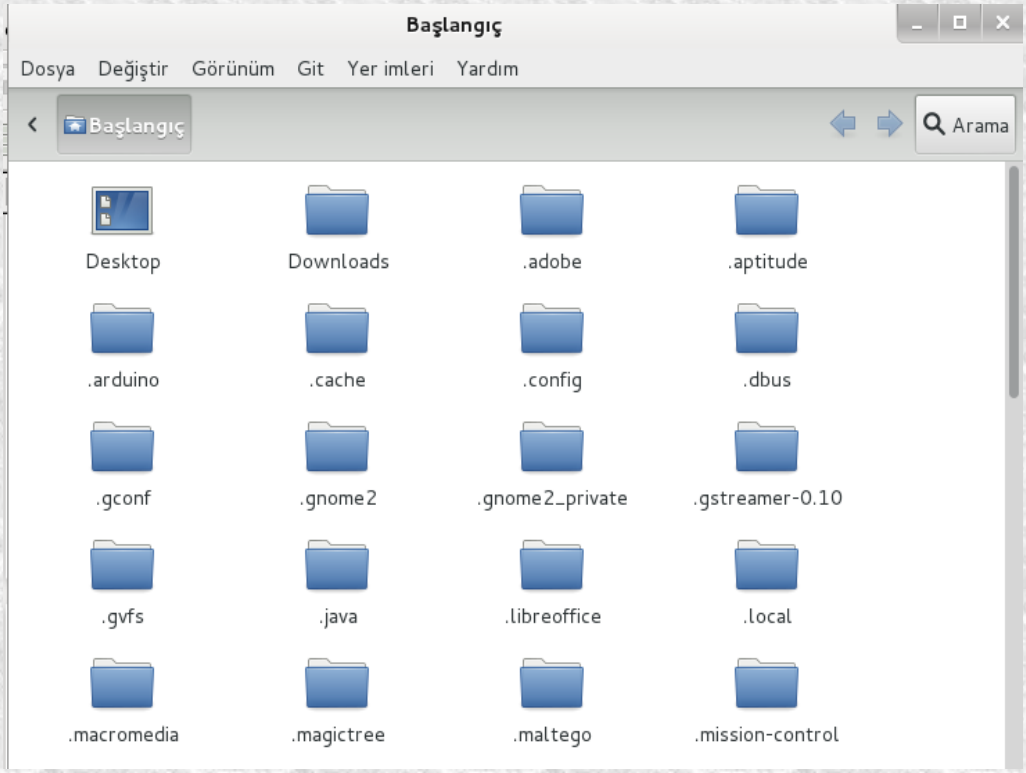
Gördüğünüz gibi biraz daha düzgün ve ayrıntılı olarak listeledi. Son olarak da **ls -la** komutunu kullanarak çıktımızın nasıl olacağını görelim.



Ne var ne yok listeledi. **Gizli dosyalar** da dahil buna. Gizli dosyaların önünde nokta işareti bulunmakta.
Burada ek bir bilgi verelim. Eğer komut satırından değil de dosya yöneticisini kullanarak gizli doayaları görmek isterseniz bazı dağıtımlarda (örneğin **BackTrack**) **<alt+.>(alt ve nokta)** kombinasyonunu, bazı dağıtımlarda da (örneğin **Kali Linux**) **<ctrl+h>** kombinasyonunu kullanmalısınız. Hemen örnekleyelim. Aşağıda dosya yöneticisiyle açtığımız dizine bakalım.



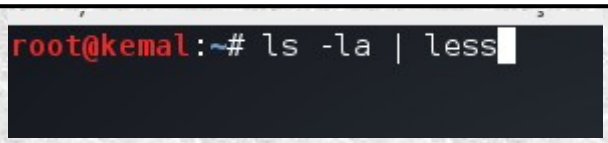
Şimdi **<ctrl+h>** tuşlarına basalım. Gizli dosyalar da ortaya çıkacak ve ekran görüntüsü aşağıdaki gibi olacaktır.



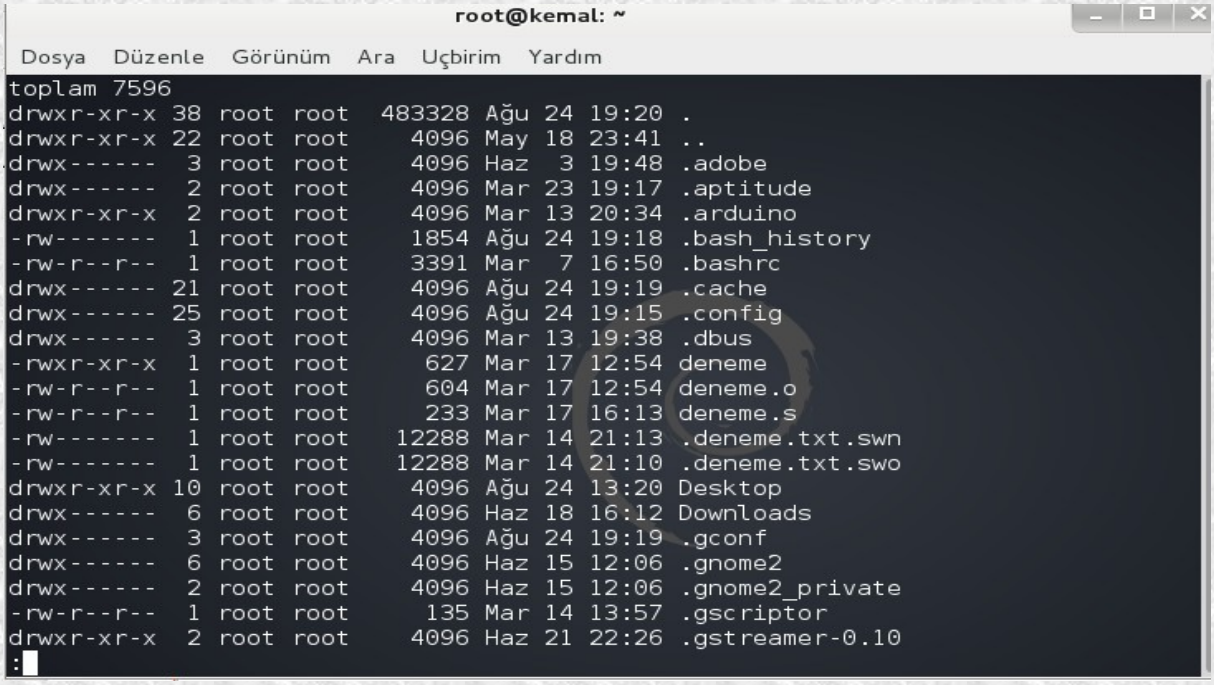
Dizinin eski haline dönüp gizli dosyaların yeniden kaybolması için yine **<ctrl+h>** kombinasyonunu kullanabilirsiniz.

Buradan bir şey daha öğreniyoruz. Bir dosyayı/klasörü gizli hale getirmek için dosya/klasör isminin önüne nokta koymamız yetiyor. Örneğin **deneme** isminde bir klasörümüz/dosyamız var diyelim. Bu klasörü gizli hale getirip görünmez yapmak için isminin önüne nokta koyup **.deneme** olarak kaydediyoruz. Bu kadar basit.

ls veya başka bir komutla aldığımız çıktı çok uzun olabilir. Bu durumda daha düzgün ve parçalı çıktı almak için **less** ve **more** komutlarından faydalanabiliriz.

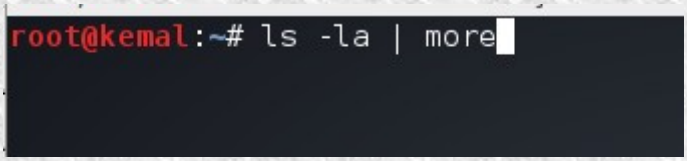


Komutu bu şekilde kullandıktan sonra aşağıdaki ekran görüntüsüne benzer bir konsol çıktısı alacaksınız.

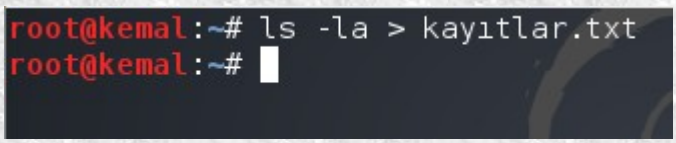


Enter a bastığınızda satır satır, **space** ye bastığınızda sayfa sayfa listeleme olacaktır. Çıkmak için de **q** tuşuna basmanız yeterlidir.

more kullandığımızda da **less** deki gibi bir çıktı verecektir.

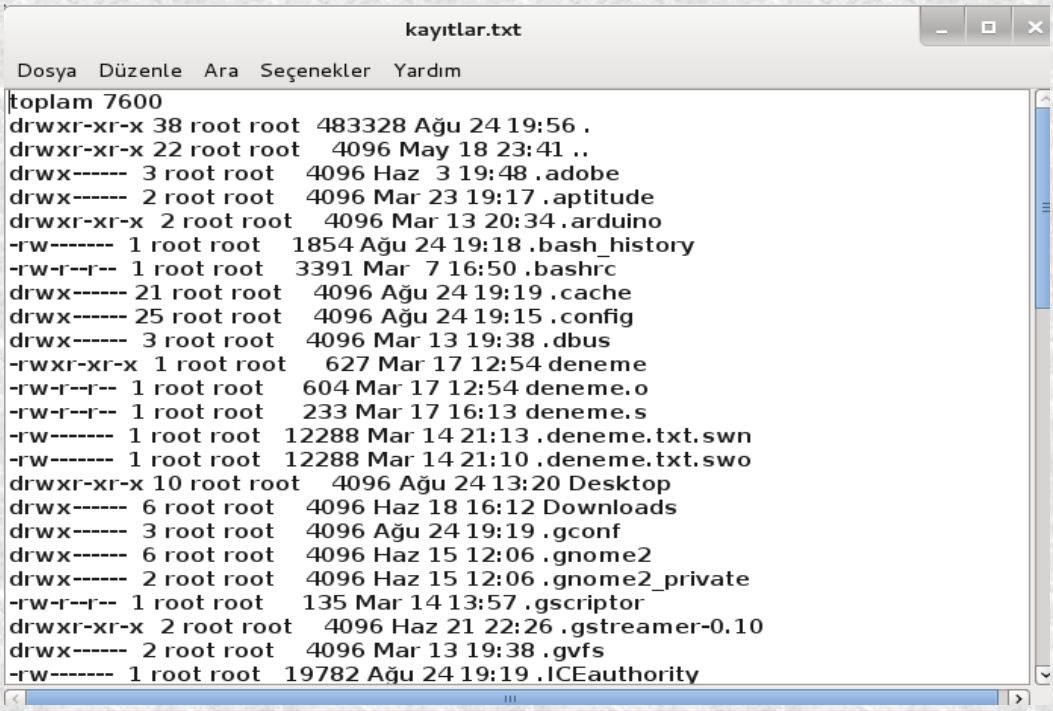


Gerçi zamanı gelince bahsedeğim ama burada küçük bir örnek verelim. Örneğin ls komutunun çıktısını konsolda görüntülemek yerine bir **txt dosyasına kaydetmek** istiyorsunuz. Bunun için kullanmanız gerken komut **ls -la > kayıtlar.txt** gibi bir komut olmalıdır. Yani **>** operatörünü kullanmamız gerekecek.



Gördüğünüz gibi konsolda komutun çıktısı görülmüyor. Komut çıktısı **kayıtlar.txt** dosyasının içine yazıldı. Daha önce başlangıç dizinimizde olmayan kayıtlar.txt dosyası komutumuzla birlikte oluşturularak komut çıktısı bu oluşturulan txt dosyasının içine yazıldı. Kontrol edelim hemen.

Dosya içeriğine bakalım.



Evet bizim istediğimiz gibi dosya oluşturulmuş ve içine yazılmış.

Son olarak, ls komutunu ille de bulunduğumuz dizin içinde kullanmak zorunda değiliz. Örneğin **/etc** dizininin içeriğini listelemek için **ls -l /etc** komutunu kullanabiliriz. Öğrendiklerimizi de kullanarak komutu **ls -l /etc | less**, **ls -l /etc | more** gibi farklı şekillerde kullanabiliriz. Ya da dosyaya yazdırmak için **ls -l /etc > deneme.txt** benzeri bir komut kullanabileceğimizi de biliyoruz. Konsol çıktısını içine yazdıracağımız dosyanın masaüstünde oluşturulması için de **ls -l /etc > /root/Desktop/deneme.txt** gibi bir komut da kullanabiliriz. Burada **/root/Desktop** yerine direk **Desktop/deneme.txt** de kullanılabilir. Seçenekler çoğaltılabilir. Biz şimdilik sadece **ls -l /etc** komutunu örnek verelim.

```
root@kemal:~# ls -l /etc
toplam 1932
-rw-r--r--   1 root root     2981 Mar 11 22:01 adduser.conf
-rw-r--r--   1 root root       44 Ağu 24 19:18 adjtime
-rw-r--r--   1 root root      185 Mar 11 22:16 aliases
drwxr-xr-x   2 root root    20480 Mar 30 11:52 alternatives
drwxr-xr-x   2 root root     4096 Mar 13 19:14 amap
drwxr-xr-x   7 root root     4096 Mar 30 12:08 apache2
-rw-r--r--   1 root root      112 Haz 20  2007 apg.conf
drwxr-xr-x   3 root root     4096 Mar 13 19:14 apm
drwxr-xr-x   4 root root     4096 Haz 15 11:48 apparmor.d
drwxr-xr-x   6 root root     4096 May 13 08:29 apt
-rw-r-----   1 root daemon    144 Haz  9  2012 at.deny
drwxr-xr-x   2 root root     4096 Mar 13 19:14 at-spi2
drwxr-xr-x   3 root root     4096 Mar 30 12:06 avahi
-rw-r--r--   1 root root   550242 Eki 19  2011 avrdude.conf
-rw-r--r--   1 root root     1895 Oca  1  2013 bash.bashrc
-rw-r--r--   1 root root       45 Haz 17  2012 bash_completion
drwxr-xr-x   2 root root     4096 Haz 15 11:46 bash_completion.d
-rw-r--r--   1 root root      356 Ara 30  2012 bindresvport.blackl
-rw-r--r--   1 root root      246 Mar 13 20:56 blkid.tab
drwxr-xr-x   3 root root     4096 Mar 13 19:14 bluemaho
```

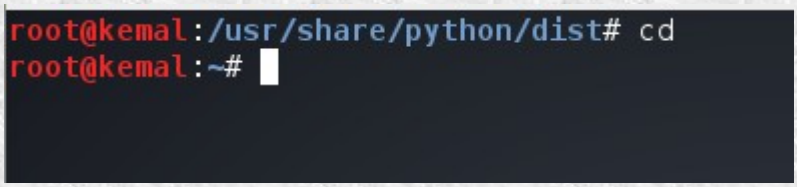
Şimdi **cd** komutundan bahsedelim. Dizinler/klasörler arasında **geçiş yapmak** için cd komutunu kullanırız. Örneğin **/usr** dizinine gitmek için **cd /usr** komutunu kulanmamız gerekir. /usr dizinindeyken ls ile alt dizinleri kontrol edebiliriz. Örneğin alt dizin olan **/share** dizinine geçmek için de **cd share** komutunu kullanabiliriz.

```
root@kemal:~# cd /usr
root@kemal:/usr# ls
bin  games  i486-linux-gnu  include  lib  local  sbin  share  src  var
root@kemal:/usr# cd share
root@kemal:/usr/share#
```

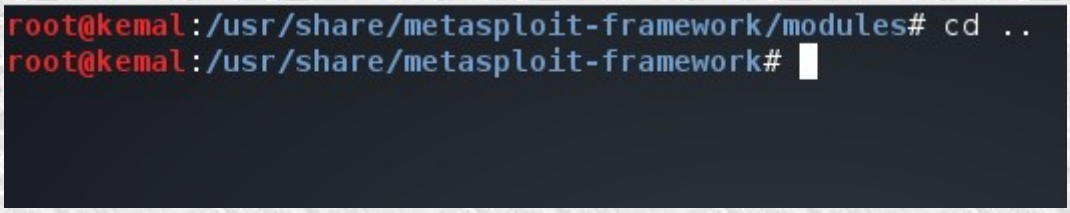
Direk /share dizinine geçmek için **cd /usr/share** komutunu kullanırız.

```
root@kemal:~# cd /usr/share
root@kemal:/usr/share#
```

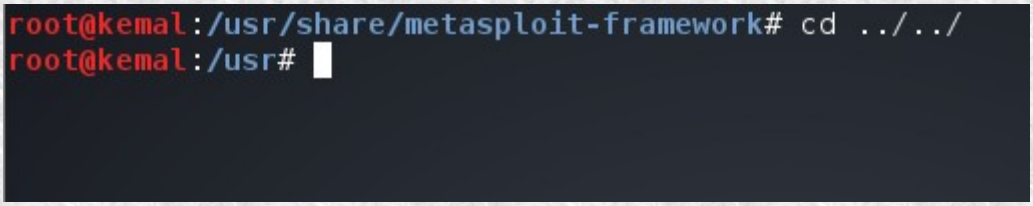
Bu şekilde alt dizinlere geçiyoruz. Peki ev dizinimize (**yani çalıştığımız ana dizine**) dönmek için ya da üst dizine geçmek için nasıl bir komut kullanmalıyız? Tabii ki yine cd komutunu kullanacağız fakat ufak bazı farklılıklarla. İlk önce ana dizine geçmek için kullandığımız cd komutundan bahsedelim. Örneğin **/usr/share/python/dist** dizinindeyiz diyelim. Ana dizinimize dönmek için sadece **cd** komutunu kullanıyoruz.



Evet ana dizinimize nasıl döneceğimizi öğrendik. Eğer bir **üst dizine** ya da iki vs. üst dizine geçmek istersek ne yapacağız? Bunun için **cd ..** ya da **cd ../** komutlarını kullanacağız. Hemen örnekle gösterelim. Örneğin **/usr/share/metasploit-framework/modules** dizinindeyiz. Bir üst dizine geçmek için **cd ..** ya da **cd ../** komutunu kullanabiliriz.



İki üst dizine geçmek için de **cd ../../** (eğer üç dizinse **../../../** vs.) komutunu kullanıyoruz.



Şimdi biraz daha farklı bir şey yapalım. Mesela iki dizin üstteki dizinin içindeki başka bir klasöre geçelim. Nasıl olduğunu aşağıdaki ekran görüntüsünden inceleyebilirsiniz.

Yukarıdaki ekranda gördüğünüz komutla şunu demek istedik: İki dizin üste git ve gittiğin dizin altındaki /joomscan alt dizinine geç. Evet bu kadar kolay. Son olarak son çalışılan iki dizin arasında nasıl geçiş yapılacağına bakalım. Bunun için de kulanmamız gereken komut **cd –** komutu. Örneğin /root dizininden /home dizinine geçmiş olalım. Bu iki dizin arasında kolayca geçmek için **cd –** komutunu kullanıyoruz.

```
root@kemal:~# cd /home
root@kemal:/home# cd -
/root
root@kemal:~# cd -
/home
root@kemal:/home#
```

Konsolda Bazı Pratik Metotlar

Konsolda komutları kullanırken bir takım pratik metotlardan faydalanabiliriz. Örneğin konsolu temizlemek için kullanılan **clear** komutu yerine **ctrl+L** kombinasyonunu kullanabiliriz. Ya da komutu yanlış yazdıysak promptun başına dönmek için **ctrl+U** tuşlarını kullanabiliriz.

```
root@kemal:~# mesela yanlış yazdık diyelim
```

Bu durumda **ctrl+U** ile başa döneriz.

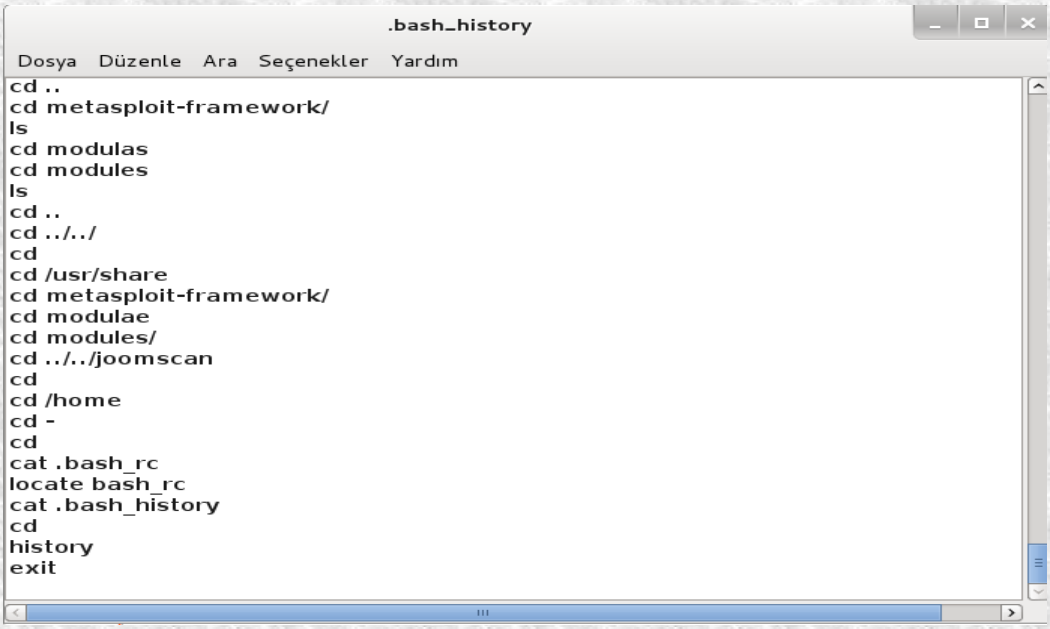
```
root@kemal:~#
```

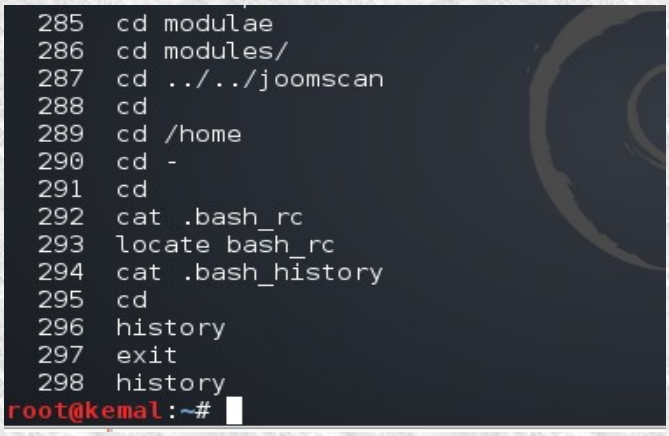
Konsolda verdiğimiz komut çalışırken komutu kesmek için **ctrl+C** kombinasyonunu kullanabiliriz. Konsoldan (terminal/uçbirim) çıkmak için kullandığımız **exit** komutu yerine **ctrl+D** tuşlarını kullanabiliriz. Böylece işlerimizi biraz daha kolay halletmiş oluruz.

History

İyi güzel konsolda komutları kullanıyoruz. Daha önceden kullandığımız komutları listeleyebilir miyiz? Evet **history** komutuyla bunu yapabiliriz. Konsolda kullandığımız komutlar **.bash_history** dosyası içine kaydedilir. Bizim kullandığımız **history** komutu bu dosyanın içeriğini referans alır. Bahsettiğm bu .bash_history dosyasının içeriği silinirse dolayısıyla history komutuyla görüntülediğimiz daha önceden kullanılan komutlar da silinecektir. Fakat biz konsoldan komut kullanmaya devam ettikçe yine bu dosyanın içine kullandığımız komutlarla ilgili bilgiler

yazılacak ve biz silene kadar tutulmaya devam edilecektir. Farkına vardığınız gibi (noktayla başlamış) .bash_history dosyası gizli bir dosya. İlk önce bu dosyanın içeriğine bakalım daha sonra da konsoldan **history** komutunu kullanarak kontrol edelim.





Yukarıdaki görüntü konsoldan **history** komutunu kullandıktan sonraki konsol çıktısının görüntüsüdür. Gördüğünüz gibi son kullandığımız history komutu da listede yer almış. Bu listelenen komutlardan istediğimiz bir tanesini yeniden kullanmak için kullanmak istediğimiz komutun numarasının soluna ünlem işaretini koymamız yeterlidir. Mesela **289** nolu **cd /home** komutunu yeniden kullanmak için **!289** komutunu vermem yeterli olacaktır.

Burada ek bir bilgi olarak, eğer son kullandığımız komutu yeniden kullanmak istersek bunun için iki ünlem (!!) işaretini komut olarak veriyoruz.

```
root@kemal:~# ls
deneme     deneme.s  Downloads                                  merhaba.py
deneme.o   Desktop   install_flash_player_11_linux.i386.tar.gz
root@kemal:~# !!
ls
deneme     deneme.s  Downloads                                  merhaba.py
deneme.o   Desktop   install_flash_player_11_linux.i386.tar.gz
root@kemal:~#
```

```
root@kemal:~# !! /etc
ls  /etc
adduser.conf            hosts.deny          polipo
adjtime                 ImageMagick         polkit-1
aliases                 icedtea-web         postgresql
alternatives            iceweasel           postgresql-common
amap                    idmapd.conf         ppp
apache2                 ifplugd             privoxy
apg.conf                init                profile
apm                     init.d              profile.d
apparmor.d              initramfs-tools     protocols
apt                     inittab             proxychains.conf
at.deny                 inputrc             pulse
```

Yukarıdaki ekran görüntülerine baktığınızda son kullandığım komut **ls** komutu olduğundan **!!** ile yeniden ls komutunu kullanmış oluyorum. Dolayısıyla **!! /etc** komutu ile **ls /etc** aynı işi yapıyor.

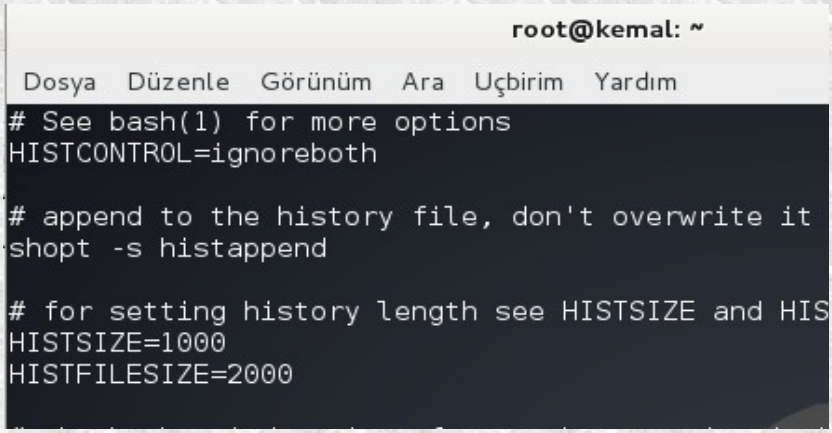
history komutuyla ilgili bir şey daha söyleyelim. Eğer kullanılan komutların tamamını değil de örneğin son 15 tanesini listelemek istiyorsak bu durumda **history 15** komutunu kullanmamız gerekecektir.

```
root@kemal:~# history 15
  317  cd!
  318  cd!cd!
  319  ls
  320  ls  /etc
  321  ls  /etc ls  /etc
  322  cd /home && ls
  323  cd
  324  cd /home,ls
  325  cd /home;ls
  326  cd
  327  cd /home; cd /etc
  328  cd
  329  cd /home;cd /etc
  330  cd
  331  history 15
root@kemal:~#
```

Eğer daha önceden kullandığımız komutlardan örneğin **k** ile başlayan son komutu tekrarlamak için **!k** komutunu kullanabiliriz. Eğer bir kısmını hatırlıyorsak bu durumda örneğin **!ke** gibi bir komut kullanmalıyız.

Peki kullandığımız komutların kaç tanesi saklanıyor. Biz history komutuyla daha önceden kullanılan komutları görebiliyoruz ama acaba kaç tanesi saklanıyor? Bunu öğrenmek için **echo $HISTSIZE** ile **$HISTSIZE** ortam değişkeninin değerine bakmamız lazım.

Anlaşılabileceği gibi 1000 tane komut saklanıyormuş. Bunu değiştiremez miyiz? Kişisel dizinimizdeki (ev dizinimizde) **.bashrc** dosyasının içinde yer alan HISTSIZE değerini değiştirerek istediğimiz değeri verebiliriz.



Yukarıda **.bash_history** içeriğini silerek daha önce kullandığımız komutların kaydını da sileriz demiştik. Bunu yapmak için pratik bir yol daha var. O da **history -c** komutu ile history i temizlemek, yani önceki kullanılan komutları silmek.

Konsolda Birden Fazla Komut Kullanma

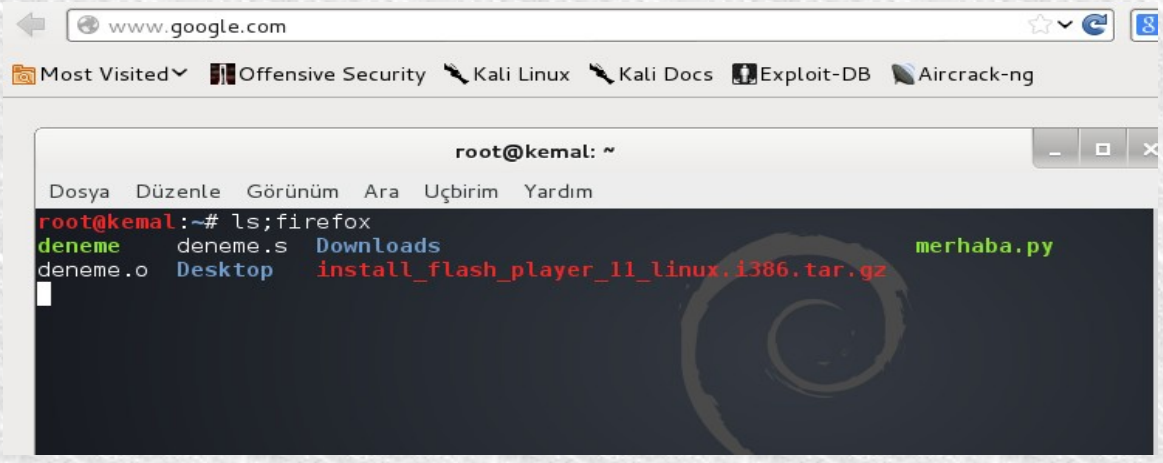
Yeni bir konuya geçebiliriz artık. Konsolda birden fazla komutu da yan yana yazarak kullanma şansımız vardır. Bunun için ya iki adet **&** işaretini kullanacağız ya da iki komut arasına **noktalı virgül** koyacağız.

```
root@kemal:~# ls;ls -l
deneme     deneme.s  Downloads
deneme.o   Desktop   install_flash_player_11_linux.i386.tar.gz
toplam 6788
-rwxr-xr-x  1 root root     627 Mar 17 12:54 deneme
-rw-r--r--  1 root root     604 Mar 17 12:54 deneme.o
-rw-r--r--  1 root root     233 Mar 17 16:13 deneme.s
drwxr-xr-x 10 root root    4096 Ağu 25 22:27 Desktop
drwx------  6 root root    4096 Haz 18 16:12 Downloads
-rw-r--r--  1 root root 6923111 Haz  3 19:42 install_flash_play
tar.gz
-rwxr-xr-x  1 root root     112 Ağu 20 17:50 merhaba.py
root@kemal:~#
```

Yukarıdaki ekran görüntüsünü incelediğimiz zaman; **ls;ls -l** komutunu verdik ve komutun çıktısı önce ilk komut sonra diğer komutun işletilmesi şeklinde oldu. Eğer **cd /var/log;ls** komutunu kullansaydık bu durumda konsol çıktımız altta görüldüğü gibi olacaktı.

```
root@kemal:~# cd /var/log;ls
alternatives.log  dmesg.3.gz      lynis-report.dat  polipo
apache2           dmesg.4.gz      mail.err          postgresql
apt               dpkg.log        mail.info         privoxy
auth.log          exim4           mail.log          pycentral.log
bootstrap.log     faillog         mail.warn         samba
btmp              fontconfig.log  messages          stunnel4
chkrootkit        fsck            mysql             syslog
ConsoleKit        gdm3            mysql.err         tor
daemon.log        guymager.log    mysql.log         user.log
debug             installer       news              wtmp
dmesg             kern.log        ntpstats          Xorg.0.log
dmesg.0           lastlog         openvas           Xorg.0.log.old
dmesg.1.gz        lpr.log         pm-powersave.log  Xorg.1.log
dmesg.2.gz        lynis.log       pm-suspend.log    Xorg.1.log.old
root@kemal:/var/log#
```

Son örnek olarak da aşağıdaki çıktıyı verelim.

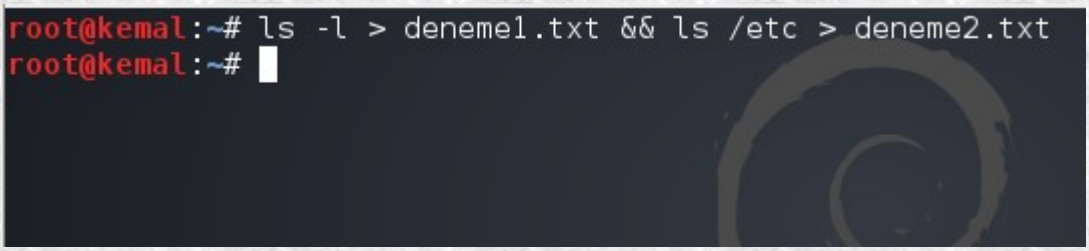


Yukarıdaki görüntüyü incelediğimizde ilk önce **ls** komutunun işletildiğini sonra da **firefox** komutunun işletildiğini görüyoruz. Arka planda konsol komutuyla açtığımız tarayıcı çalışmakta. Tarayıcıyı kapattığımızda konsol ls çıktısını gösterdikten sonra komut satırı önceki haline dönecektir.

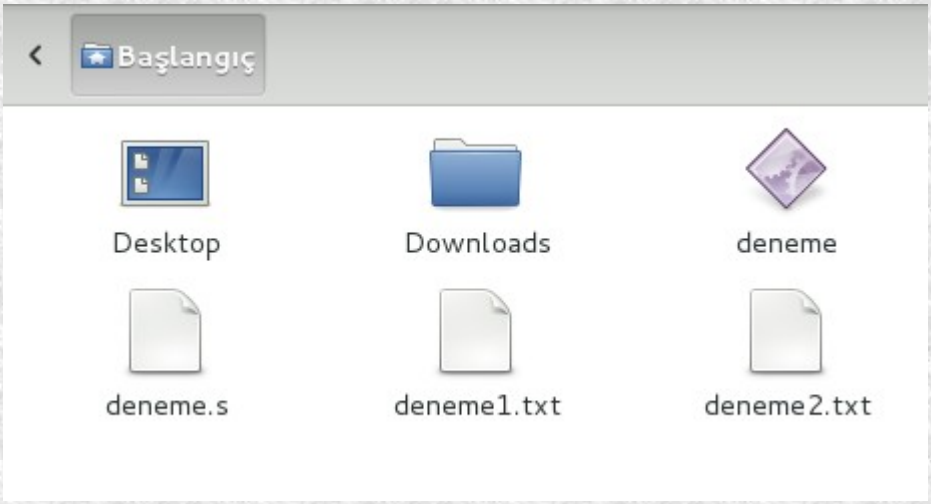
Şimdi de diğer operatörümüz olan **&&** kullanımını inceleyelim. Bununla ilgili olarak yine bir kaç örnek yapalım. Örneklere geçmeden bu operatörün en çok nerede kullanıldığını hatırlatmak istiyorum. Sistemimizi güncellemek istediğimizde daha sonra da bahsedeceğimiz iki komutumuz bu işi halletmektedir. Bu komutlar **apt-get update** ve **apt-get upgrade** komutlarıdır. Bu komutları konsolda ayrı ayrı kulanabileceğimiz gibi tek satırda yazıp sırasıyla işletilmesini de sağlayabiliriz. Bunun için **apt-get update && apt-get upgrade** şeklinde bir komut işimizi görecektir.

Şimdi örnek olarak **ls && cd /etc** komutunu deneyelim.

Görüldüğü gibi ilk önce **ls** komutu çalıştırılmış sonrasında ise **/etc** dizinine geçilmiştir. Son bir örnek vererek bu konuyu da bitirelim.



Yukarıdaki komutla şunu demiş olduk: Çalıştığım dizinin içeriğinin listesini, oluşturulan **deneme1.txt** listesine yaz, /etc dizininin içerik listesini de **deneme2.txt** dosyası oluşturarak bu dosyanın içine yaz. Bakalım istediklerimizi yapmış mı?



Görüldüğü gibi istediğimiz dosyalar oluşturulmuş ve konsol çıktısı bu dosyaların içine yazılmış.

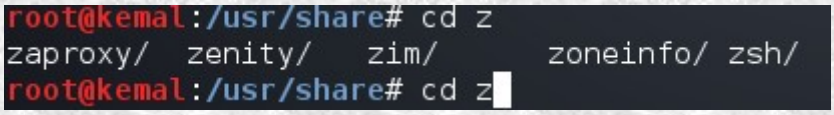
Bir de **||** kullanımından bahsetmek istiyorum. Konsolda iki komut kullanırken komutlar arasına bu operatörü koyarsak, birinci komut **başarısız** olursa **diğer komutu** çalıştırmasını söylemiş oluruz. Örneğin **ls-** diye bir komut yok. Dolayısıyla bu komut çalıştırılmaz. Biz **ls- || ls -l** diye bir komut kullandığımızda birinci komut başarısız olacağından diğer komut çalıştırılacaktır.

```
root@kemal:~# ls- ||ls -l
bash: ls-: komut yok
toplam 428
-rwxr-xr-x  1 root root    627 Mar 17 12:54 deneme
-rw-r--r--  1 root root    604 Mar 17 12:54 deneme.o
-rw-r--r--  1 root root    233 Mar 17 16:13 deneme.s
drwxr-xr-x 12 root root   4096 Eyl  5 14:26 Desktop
drwx------  6 root root   4096 Eyl  5 10:16 Downloads
-rw-r--r--  1 root root 180803 Eyl  5 10:00 explore-2013-09-04
drwxr-xr-x  2 root root   4096 Ağu 29 19:16 firefox-flash
-rwxr-xr-x  1 root root    112 Ağu 20 17:50 merhaba.py
-rw-r--r--  1 root root 219371 Eyl  5 10:21 nmap-1.51.tar.gz
-rw-r--r--  1 root root    171 Eyl  4 20:05 zeytin
root@kemal:~#
```

Linux sistemlerde komutlar çok farklı kombinasyonlarla ve parametrelerle kullanılabilir. Bu yüzden bize esneklik sağlar. Komut satırı bu güçlü özelliğinden dolayı tecrübeli kullanıcıların kullanmaktan zevk aldıkları ve bir yerde aslında kullanmak zorunda oldukları önemli bir yardımcıdır. Tecrübe kazandıkça, karmaşık gibi görünen işleri ne kadar kolaylıkla hallettiğinize şaşıracaksınız.

Komut Tamamlama

Konsolda bir komutun ilk bir kaç harfini yazıp **tab** tuşuna bastığınızda komutunuz otomatik olarak tamamlanacaktır. Eğer tab tuşuna **iki kez** üst üste basarsanız bu sefer olası ifadeler listelenecektir.



Yukarıda da gördüğünüz gibi **/usr/share** altındayken **cd z** yazıp **iki kere tab** tuşuna bastığımda olası dizinler listelendi. Eğer **cd zap** yazdıktan sonra bir kere tab tuşuna basarsak bu sefer otomatik olarak **zaproxy** ye tamamlanacaktır.

Yine aklıma gelmişken geride bıraktığımız history komutunun kullanımına ek olarak küçük bir hatırlatma yapalım. Konsolda pratik yoldan az önce kullandığınız komutları yeniden kullanmak isterseniz **yukarı ok** tuşuyla kullandığınız komutları görüp tekrar kullanabilirsiniz. Bu hatırlatmadan sonra yeni konumuza geçebiliriz artık. Yeni konumuzda sistem açıp-kapatma, reboot etme vs. konularından bahsedeceğim.

Açma-Kapatma-Reboot

Linux sistemimizi konsoldan komut vererek açıp-kapatabiliriz. Reboot edebiliriz vs. Bunların nasıl olduğunu görmeden önce bahsetmemiz gereken bir konu var. Bahsedeceğimiz konu **sistem çalışma seviyeleri** yani **runlevels** konusu. Linux işletim sistemi başladığında bildiğiniz gibi kernel yani çekirdek belleğe yüklenerek çalıştırılır ve donanımları tarayarak sürücü yazılımlarını yüklemeye başlar. Sonra root dosya sistemi (yani /) **mount** edilir (bağlanır) ve **init** programı başlatılarak sistem çalışır hale getirilir. Bu işlem yapılırken **sistemin çalışma düzeyi** yani **runlevel** ayarlanır. Gerçi günümüzde çoğu Linux sistem varsayılan olarak grafiksel arayüzle başlatılsa da diğer çalışma seviyelerinde açıldığı zaman o çalışma seviyesinde hangi hizmetleri verecekse ilgili servisler-programlar başlatılır. Kabaca sistem bu şekilde işler.

Linux sistemlerde **7 farklı çalışma seviyesi** vardır:

0:Hiç bir servisin çalışmadığı ve kapatma işlemlerinin başladığı seviye (**halt**)

1:Tek kullanıcılı (**single user**) kullanıcı seviyesi. Ağ servisleri çalışmaz. Sistem bakımı için kullanılabilir.

2:Ağ desteği olmadan çok kullanıcılı çalışma seviyesi.

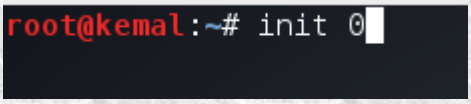
3:Ağ destekli çok amaçlı çalışma seviyesi.

4:Kullanılmaz. Fakat kullanıcı tarafından özel olarak tanımlanabilir.

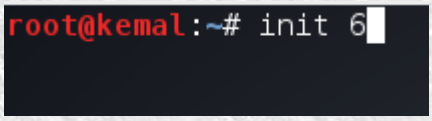
5:Grafiksel kullanıcı ara yüzünün çalıştığı seviye. Günümüzün kişisel kullanım için olan Linux dağıtımlarının hemen hepsi varsayılan olarak bu çalışma seviyesinde başlatılır.

6:Sistemi yeniden başlatma (**reboot**) seviyesi.

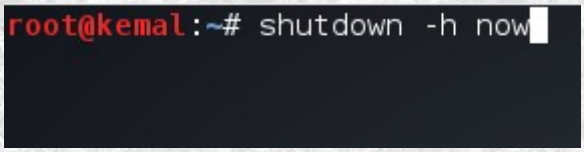
Sistemimizi kapatmak, yeniden başlatmak için vs. bu çalışma seviyeleriyle ilgili komut kullanabiliriz. Örneğin **init 0** komutuyla sistemi 0. level konumuna getirebiliriz. Bu levelde hiç bir servis-program çalışmadığından sistem kapanışa geçecektir. Yani konsolda init 0 komutunu kullandığımızda **sistemimiz kapanacaktır**.



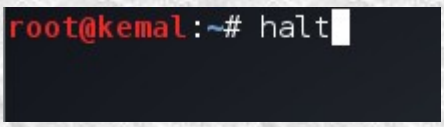
Eğer sistemimizi 6. çalışma seviyesine geçirmek istiyorsak bu durumda **init 6** komutunu kullanmalıyız. Bu çalışma seviyesinde sistem **reboot** olacaktır.



Sistemi kapatmak için, yeniden başlatmak için başka komutlar da kullanabiliriz. Örneğin **shutdown -h now** komutuyla sistemimiz beklemeden kapanma işlemine geçecektir.



Eğer hemen değil de belli bir süre sonra sistemin kapanmasını istiyorsak, örneğin 10 dk. Sonra kapanması için **shutdown -h now+10** komutunu kullanabiliriz. Sistemi kapatmak için kullanabileceğimiz bir komutumuz daha var. O komut da **halt** komutu. Konsoldan halt komutunu verdiğimizde sistem kapanmaya başlayacaktır.

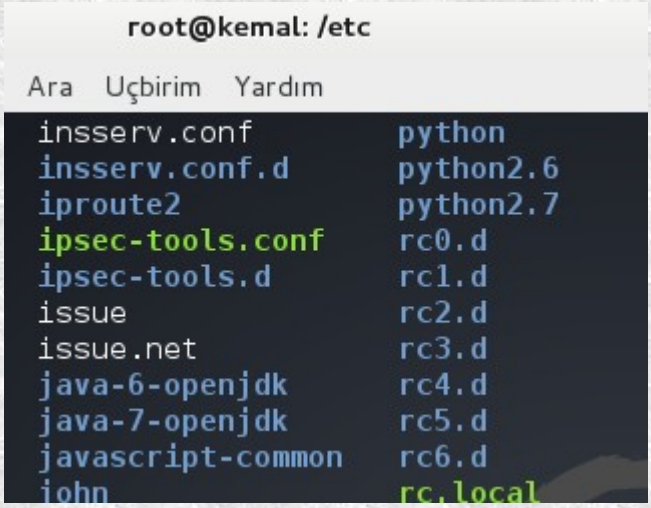


Sistemi reboot etmek yani yeniden başlatmak için kullanabileceğimiz bir komut da **reboot** komutudur. Aynen init 6 komutunda olduğu gibi reboot komutunda da sistem yeniden başlatılacaktır.

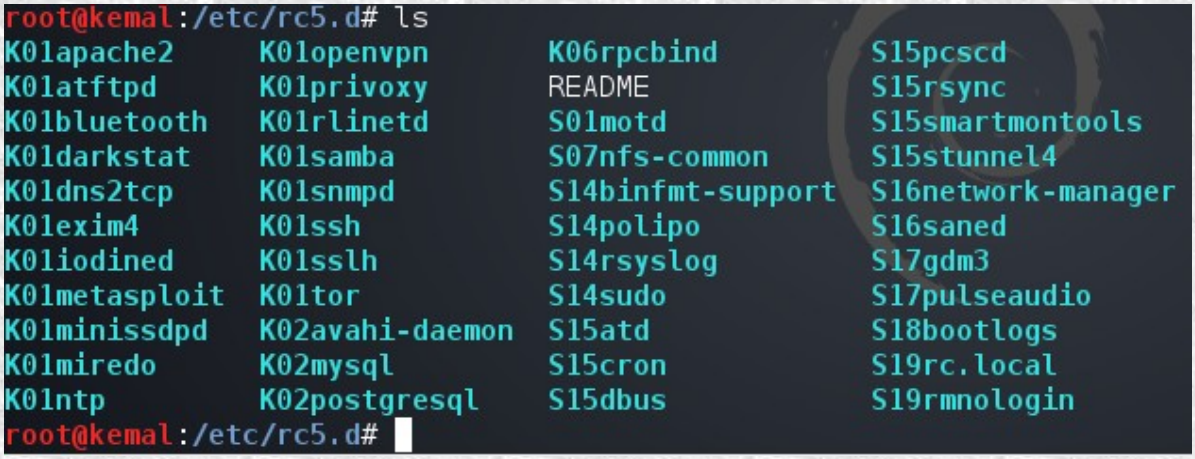
Burada bir hatırlatma yapmak istiyorum. Eğer **shutdown** komutunu parametresiz olarak kullanırsak sanki **init 1** komutunu kullanmışız gibi sistem 1. çalışma sevitesine (**single user mod**) geçecektir.

Sistemi kapatma-yeniden başlatma komutlarını öğrendik artık. Evet çalışma seviyeleri dedik fakat hangi çalışma seviyesinde hangi **programların-servislerin-scriptlerin** çalıştırılacağıyla ilgili bilgilerin hangi dizinlerde olduğuna bakmadık.

Hangi çalışma seviyesinde hangi servislerin çalıştırılacağı ile ilgili scriptler(betikler) **/etc** dizini altındaki **rcx.d** içinde bulunur. Burada **x** yerine çalışma seviyesi ile ilgili rakamlar gelir. Örneğin **rc5.d** den kasıt **5. çalışma seviyesidir**.



/etc dizinini incelediğimizde yukarıdaki görüntüde olduğu gibi **rcx.d** (**rc0.d....rc6.d**) dosyaları görürüz. Şimdi örneğin grafiksel arayüz ile ilgili olan 5. çalışma seviyesini ifade eden **rc5.d** dosyasını inceleyelim.



Görüntüyü incelediğinizde bazı dosyaların önünde **K** harfinin, bazı dosyaların da önünde **S** harfinin olduğunu görürsünüz. K ile başlayan dosyalar bu çalışma seviyesinde **durdurulacak servisleri**, S ile başlayan dosyalar ise **çalıştırılacak servisleri** belirtmektedir. K ve S harfinden sonra gelen sayıya göre scriptlerin çalıştırılma sırası belirlenir. Daha küçük değere sahip olan script daha önce çalıştırılır. Bir çalışma seviyesine geçildiğinde ilk önce durdurma scriptleri daha sonra da başlatma

scriptleri çalıştırılır.

Sanal Konsollar

Linux sistemlerde aynı anda birden çok sanal konsolda çalışabilirsiniz. **<ctrl+alt+f1>.....<ctrl+alt+f6>** ile **6** tane sanal konsol açıp bu konsollarda çalışabilirsiniz. Sanal konsoldan kasıt grafik arayüz olmadan sadece konsoldan çalışmaktır. Yoksa grafik çalışma modundayken de zaten istediğiniz kadar konsol açıp çalışabilirsiniz. Tabi kullanıcı adı ve parola girerek çalışmaya başlayabilirsiniz. Eğer tekrar grafik çalışma alanına dönmek isterseniz **<ctrl+alt+f7>** ile bunu yapabilirsiniz. Gerçi her hangi bir sanal konsolda çalışırken sadece **<alt+f1.....f6>** ile diğer sanal konsollara geçebilir ve **<alt+f7>** ile de grafiksel ara birime dönebilirsiniz. Ama grafiksel moddan her hangi bir sanal konsola ilk defa geçerken **<ctrl+alt>** ile beraber **f1,...,f6** kullanmanız gerekir.

Servislerin Başlatılması-Durdurulması

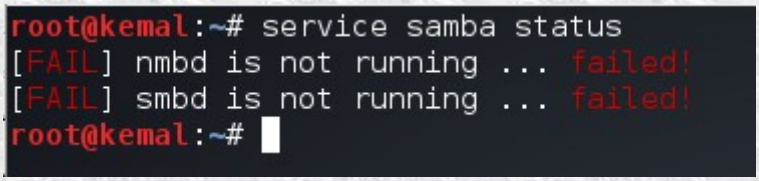
Linux'la beraber standart olarak gelen ya da bizim sonradan kurduğumuz bazı servisler olabilir. Örneğin, **Samba** dosya paylaşım servisi, **MySQL** veritabanı, **Apache** web sunucu servisi vs.
Sistemimizdeki servislerle ilgili betikler **/etc/init.d** altındadır.

```
root@kemal:/etc/init.d# ls
apache2                    minissdpd
atd                        miredo
atftpd                     motd
avahi-daemon               mountall-bootclean.s
binfmt-support             mountall.sh
bluetooth                  mountdevsubfs.sh
bootlogs                   mountkernfs.sh
bootmisc.sh                mountnfs-bootclean.s
checkfs.sh                 mountnfs.sh
checkroot-bootclean.sh     mtab.sh
checkroot.sh               mysql
```

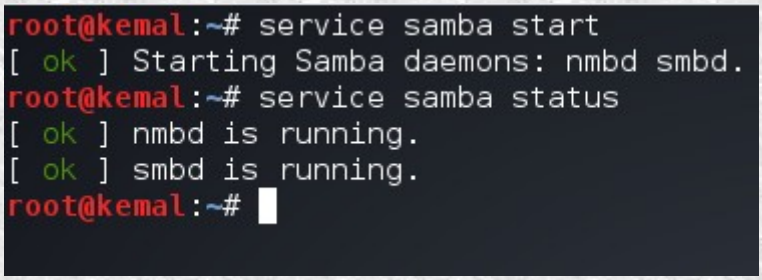
Servisleri başlatmak-durdurmak-durumuna bakmak için iki farklı yol kullanabiliriz. Birincisi **/etc/init.d/<servis> start|stop|status** kalıbıdır. Diğeri ise **service <servis_adı> start|stop|status** kalıbıdır. Tabi eğer bir servisi yeniden başlatmak isterseniz de **start|stop** kullanımında olduğu gibi **restart** da kullanabilirsiniz. Şimdi ne demek istediğimizi örnekle aşıklayalım. Örneğin **Samba** servisine bakalım.

```
root@kemal:~# /etc/init.d/samba status
[FAIL] nmbd is not running ... failed!
[FAIL] smbd is not running ... failed!
root@kemal:~#
```

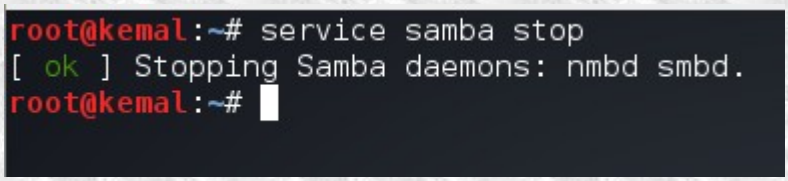
Gördüğünüz gibi samba servisi çalışmıyor. Bir de diğer komutu kullanarak bakalım.



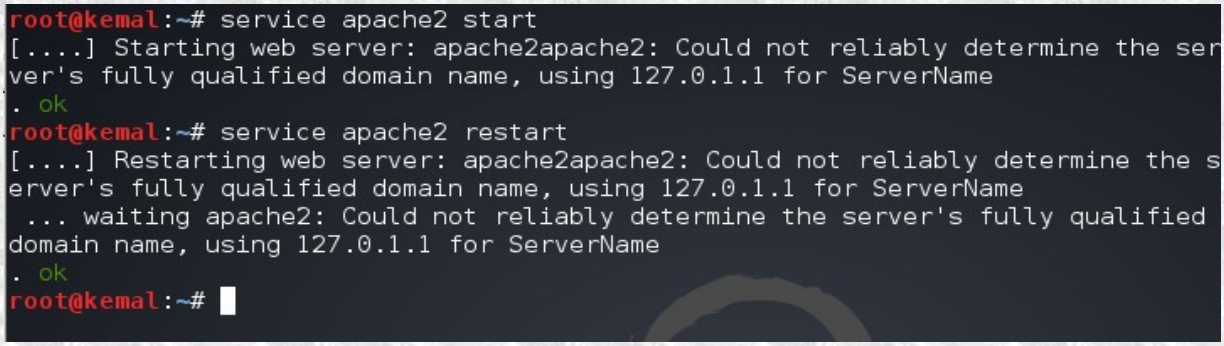
Evet gördüğünüz gibi her iki komutu da kullanabiliyoruz. Ben ikinci örnekteki kullanımı daha kolay bulduğumdan genelde o şekilde kullanıyorum. Siz de hangisi kolayınıza gelirsa onu kullanabilirsiniz. Şimdi de samba servisini çalıştıralım.



Servisimizi çalıştırdık ve durumunu sorguladık. Artık servisimiz çalışıyor. Durdurmak için de **stop** diyorduk.



Şimdi de yeniden başlatmayla ilgili bir örnek yapalım. Apache web sunucu servisimizi başlatalım ve **restart** edelim.



Küçük bir hatırlatma yaparak başka konuya geçelim. **Debian** tabanlı dağıtımlarda **server** programlarını kurmak için kullanabileceğimiz bir araç vardır. Bu aracı **tasksel** aracıdır. Konsoldan parametresiz olarak **tasksel** komutunu kullandığımızda konsol grafiksel bir arayüze geçerek seçim yapmamızı isteyecektir. Buradan istenilen servisler seçilebilir.

Konsoldan da kullanım mümkündür.

```
root@kemal:~# tasksel --help
Unknown option: help
Kullanım:
tasksel install <görev>...
tasksel remove  <görev>...
tasksel [seçenekler]
        -t, --test          sınama kipi; herhangi bir işlem yapma
            --new-install   bazı görevleri otomatik olarak kur
            --list-tasks    gösterilecek görevleri listele ve çık
            --task-packages görevdeki mevcut paketleri listele
            --task-desc     görevle ilgili açıklamayı göster
root@kemal:~# tasksel install --list-tasks
i penetration-testing   Kali's penetration testing tools
u desktop       Debian desktop environment
u web-server    Web server
u print-server  Print server
u database-server       SQL database
u dns-server    DNS Server
u file-server   File server
u mail-server   Mail server
u ssh-server    SSH server
u laptop        Laptop
root@kemal:~#
```

Süreçler

Süreç (**process**), çalışır dosyaların çalışır durumdaki haline verilen isimdir. Yani bir programı çalıştırdığımızda belleğe yüklenen o programa süreç adı verilir. Tabi bizim başlattığımız süreçler haricinde Linux sistemlerde daha bir çok servise ait

süreçler çalışmaktadır. Yani kullanıcının programları yanında servisler, sistemin kendine ait programlar vs. de çalışır. Bunların hepsi birer süreçtir. Aynı zamanda Linux sistemde çalışan bir program birden fazla süreçten oluşabilir.

Sistemde çalışan süreçleri görmek için bir kaç alternatitimiz vardır. Örneğin her hangi bir anda çalışan süreçleri görmek için **ps** komutunu kullanabiliriz. Komutun kullanım parametrelerini görmek için **ps --help a** (**a=all**) komutu işimize yarayacaktır.

```
root@kemal:~# ps --help a

Usage:
 ps [options]

Basic options:
 -A, -e               all processes
 -a                   all with tty, except session leaders
  a                   all with tty, including other users
 -d                   all except session leaders
 -N, --deselect       negate selection
  r                   only running processes
  T                   all processes on this terminal
  x                   processes without controlling ttys

Selection by list:
 -C <command>         command name
 -G, --Group <gid>    real group id or name
 -g, --group <group>  session or effective group name
 -p, --pid <pid>      process id
     --ppid <pid>     select by parent process id
 -s, --sid <session>  session id
```

Komutun çıktısından da anlaşılabileceği gibi bir çok farklı parametre kullanabiliriz. Fakat biz bunlardan en çok kullanılan komut olan **ps aux** komutuna bakalım.

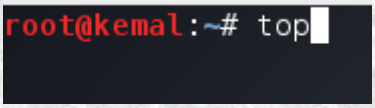
```
root@kemal:~# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.1   2280   652 ?        Ss   19:39   0:00 init [2]
root         2  0.0  0.0      0     0 ?        S    19:39   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    19:39   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   19:39   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        S    19:39   0:00 [kworker/u:0]
root         7  0.0  0.0      0     0 ?        S<   19:39   0:00 [kworker/u:0H]
root         8  0.0  0.0      0     0 ?        S    19:39   0:00 [migration/0]
root         9  0.0  0.0      0     0 ?        S    19:39   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S    19:39   0:00 [rcu_sched]
```

Siz de bir **ps alx** komutunu kullanarak çıktısını inceleyebilirsiniz. Önceki komutla arasındaki farklara bakabilirsiniz.

Her sürecin **PID** (**process id**) denilen bir numarası vardır. Bu numaralar süreçlerin haberleşmesi diyebileceğimiz bir işe yarar. Yani bir süreçle ilgili işlem için o sürecin numarasına mesaj gönderilir. Örneğin süreçleri sonlandırmak için sonlandırılmak istenen sürecin numarasına özel bir mesaj göndermek gerekir. Bu

konuya döneceğiz.

Süreçleri görüntülemek için kullanabileceğimiz komutlardan bir tanesi de **top** komutudur. Bu komutla süreçleri canlı olarak izleyebiliriz.




```
top - 20:31:41 up 51 min,  2 users,  load average: 0,25, 0,15, 0,14
Tasks: 114 total,   1 running, 113 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5,7 us,  3,7 sy,  0,0 ni, 90,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem:    505844 total,   440124 used,    65720 free,    16700 buffers
KiB Swap:  1028092 total,    24472 used,  1003620 free,   188096 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2478 root      20   0 61220 8648 3392 S   4,6  1,7   1:03.58 Xorg
 3698 root      20   0  122m  14m   9m S   2,7  2,9   0:00.75 gnome-screensho
 3173 root      20   0  141m  15m  11m S   2,0  3,2   0:03.31 gnome-terminal
 2913 root      20   0  226m  19m  11m S   0,3  3,9   0:04.61 gnome-settings-
 2943 root      20   0  196m  12m 8508 S   0,3  2,4   0:03.91 metacity
 3697 root      20   0  4324 1392 1032 R   0,3  0,3   0:00.03 top
    1 root      20   0  2280  636  604 S   0,0  0,1   0:00.98 init
    2 root      20   0     0    0    0 S   0,0  0,0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S   0,0  0,0   0:00.31 ksoftirqd/0
    5 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 kworker/0:0H
```

Konsola dönmek için **q** tuşu kullanılabilir.

Süreçleri ayrıntılı olarak bir ağaç yapısı şeklinde görmek için **pstree** komutu kullanılır.

Evet süreçleri nasıl görebileceğimizi öğrendik. Şimdi sıra her hangi bir süreci nasıl sonlandırabileceğimize geldi. Önce sonlandırmak istediğiniz sürecin numarasını (**pid**) öğrenin. Bunu süreçleri listeleyerek görebilirsiniz. Daha sonra **kill <pid>** komutuyla süreci sonlandırın. Sonra yine süreç listesinden kontrol edin. Eğer süreç hala çalışıyorsa bu sefer **kill -9 <pid>** komutunu kullanın. Bazı servisler bir kaç süreç başlatmış dolayısıyla bir kaç süreç numarasına sahip olabilirler. Bu durumda **killall <süreç_ismi>** ya da daha öldürücü darbe olan **killall -9 <süreç_ismi>** komutunu kullanabilirsiniz. Dikkat

ederseniz killall komutunu kullanırken pid yerine sürecin adını kullanıyoruz. Şimdi basit bir örnek yapalım.

Önce **apache** servisini başlatalım.

```
root@kemal:~# service apache2 start
[....] Starting web server: apache2apache2: Could not reliably determine the ser
ver's fully qualified domain name, using 127.0.1.1 for ServerName
. ok
root@kemal:~#
```

Sonra **ps aux | grep apache2** komutuyla pid numarasına bakalım.

```
root@kemal:~# ps aux | grep apache2
root      4473  0.0  1.5  36792  7608 ?        Ss   20:59   0:00 /usr/sbin/apach
e2 -k start
www-data  4482  0.0  0.8  36816  4192 ?        S    20:59   0:00 /usr/sbin/apach
e2 -k start
www-data  4483  0.0  0.8  36816  4192 ?        S    20:59   0:00 /usr/sbin/apach
e2 -k start
www-data  4484  0.0  0.8  36816  4192 ?        S    20:59   0:00 /usr/sbin/apach
e2 -k start
www-data  4485  0.0  0.8  36816  4192 ?        S    20:59   0:00 /usr/sbin/apach
e2 -k start
www-data  4486  0.0  0.8  36816  4192 ?        S    20:59   0:00 /usr/sbin/apach
e2 -k start
root      4558  0.0  0.1   3368   768 pts/0    S+   21:02   0:00 grep apache2
root@kemal:~#
```

Tek süreç numarası yok. Gördüğünüz gibi bir kaç süreç numarası var. Ben asıl süreç numarasını tespit edip onu sonlandırırsam apache servisini de sonlandırmış olurum. Ya da daha kolay bir yol olan killall komutunu kullanabilirim.

```
root@kemal:~# service apache2 status
Apache2 is running (pid 4473).
root@kemal:~#
```

Bir önceki ekran görüntüsünde en üstte yer alan 4473 süreç numarasını kullanıyormuş. Şimdi bu süreci öldürelim (terminoloji öyle ne yapalım) ve tekrar **service apache2 status** komutuyla servisimizin çalışıp çalışmadığını görelim.

```
root@kemal:~# kill 4473
root@kemal:~# service apache2 status
Apache2 is NOT running.
root@kemal:~#
```

Evet süreci öldürdükten sonra apache servisinin sonlandığını gördük. Peki aynı şeyi **killall** komutuyla yapsaydık.

```
root@kemal:~# service apache2 start
[....] Starting web server: apache2apache2: Could not reliably determine the ser
ver's fully qualified domain name, using 127.0.1.1 for ServerName
. ok
root@kemal:~# killall apache2
root@kemal:~# service apache2 status
Apache2 is NOT running.
root@kemal:~#
```

Süreçlerle ilgili olarak kullanabileceğimiz **pgrep** komutundan da bahsedelim. Örneğin **pgrep -lu root** komutuyla root kullanıcıya ait süreçleri görebiliriz.

```
root@kemal:~# pgrep -lu root
1 init
2 kthreadd
3 ksoftirqd/0
5 kworker/0:0H
6 kworker/u:0
7 kworker/u:0H
8 migration/0
9 rcu_bh
10 rcu_sched
11 watchdog/0
12 cpuset
```

Sistemde çalışan süreçlerin süreç numarasını (pid) öğrenmek için de bu komuttan faydalanabiliriz. Örneğin **pgrep apache2** komutuyla apache servisinin süreç numarasına bakabiliriz.

```
root@kemal:~# service apache2 start
[....] Starting web server: apache2apache2: Could not reliably determine the ser
ver's fully qualified domain name, using 127.0.1.1 for ServerName
. ok
root@kemal:~# pgrep apache2
3661
3681
3682
3683
3684
3685
root@kemal:~#
```

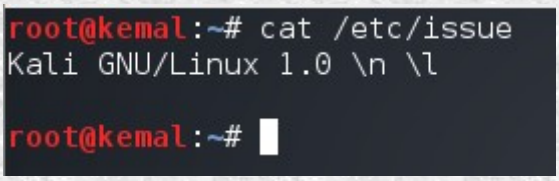
Kullanıcı İşlemleri

Linux işletim sisteminde bir çok kullanıcı oluşturulabilir. Bu kullanıcılar yetkileri çerçevesinde işlemler yapabilirler. Bilindiği gibi Linux sistemde en yetkili kullanıcı **Root** kullanıcıdır. Root kullanıcı sistemde her türlü değişikliği yapmaya yetkilidir.

Kullanıcı hesabı nasıl oluşturulur konusuna geçmeden önce öenmli iki dosyadan bahsetmek istiyorum. Bu dosyalar **/etc/passwd** ve **/etc/shadow** dosyalarıdır. Linux işletim sisteminde kullanıcı bilgileri ve kullanıcı parolalarıyla ilgili bilgiler bu

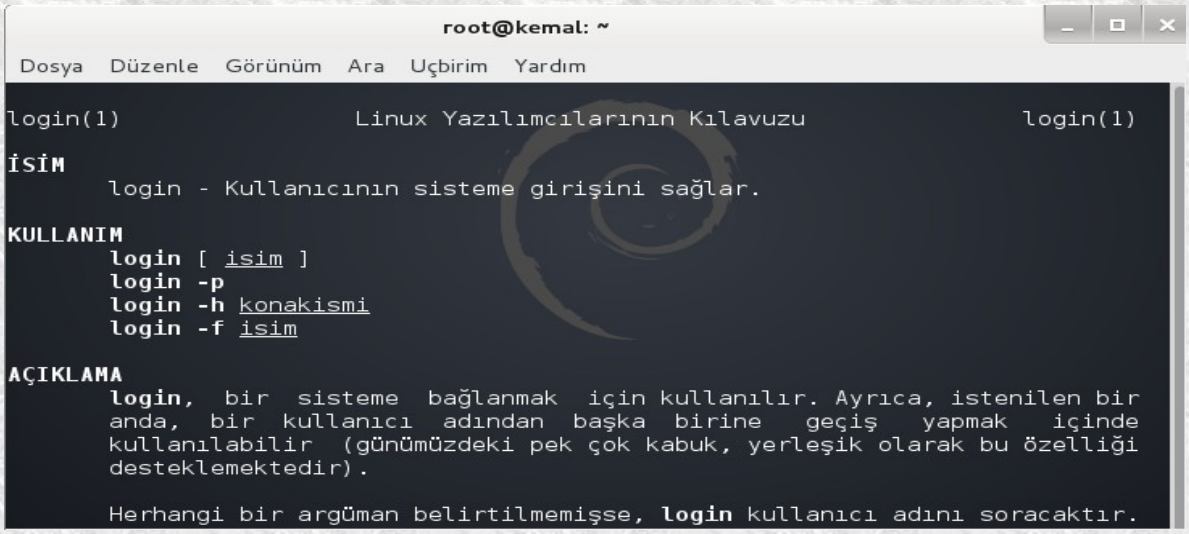
dosyalarda saklanır. **/etc/passwd** dosyasında kullanıcılarla ilgili bilgiler, **/etc/shadow** dosyasında ise kullanıcı parolalarının şifrelenmiş hali yer almaktadır. Sisteme yeni bir kullanıcı eklendiğinde eklenen kullanıcıyla ilgili bilgiler bu iki dosyaya da yazılır. Yeri geldikçe bu konudan bahsedeğim.

Bir kullanıcı sisteme giriş yapmak istediğinde **/etc/issue** içinde bulunan mesaj ekrana yansıtılır ve bir kulanıcı adı girilmesi beklenir.



Login isteminden önce gösterilecek mesajlar burada tutulur. Bu mesajı istediğiniz şekilde değiştirebilirsiniz. Evet biz devam edelim :) Kullanıcı adı girildikten sonra **login** programı çalışır ve parola girilmesi beklenir. Bilgiler uyuşursa login programı bu kullanıcı için önceden tanımlanmış olan kabuğu çalıştırır ve kullanıcı sistemde çalışmaya başlar.

Bahsettiğimiz login programını inceleyelim. **login --help** ile yardım alabileceğimizi biliyoruz fakat biz daha ayrıntılı bilgi almak için **man login** komutunu kullanalım.



Kılavuzu Türkçe'ye de çevirmişler. En altta **Yalçın Kolukısa**'nın çevirdiği bilgisi var. Türkçe de olduğuna göre açıklamaları daha rahat anlayabiliriz. Örnek komut olarak **login -p -h kemal -f root** kullanalım. Host yerine sistemimizin adını yani kemal yazdık, kullanıcı olarak da root dedik.

```
root@kemal:~# login -p -h kemal -f root
Son giriş:  Sal Ağu 27 18:03:22 EEST 2013 kemal'dan pts/0 üzerinde
Linux kemal 3.7-trunk-686-pae #1 SMP Debian 3.7.2-0+kali8 i686

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@kemal:~#
```

Burada küçük bir bilgi daha verelim. Sisteme her başarılı girişten sonra görüntülenen mesaj **/etc/motd** altında tutulur. Yukarıdaki ekran görüntüsünde yer alan ve **The programs included...** diye devam eden mesaja iyi bakın. Şimdi de **cat /etc/motd** ile görüntülediğimiz mesaja bakalım.

```
root@kemal:~# cat /etc/motd

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@kemal:~#
```

Görüldüğü üzere aynı mesaj. Bu mesajı kullanıcıları uyarmaya yarayacak şekilde değiştirelim.

```
root@kemal: ~
Dosya  Düzenle  Görünüm  Ara  Uçbirim  Yardım
 GNU nano 2.2.6          File: /etc/motd                Modified

Hoş gelmişsen gurban...çalış boş durma :)
```

Nano konsol editörünü sonra anlatacağım. Biz şimdi işimize bakalım :) Evet mesajı değiştirdik bakalım olmuş mu?

```
root@kemal:~# login -p -h kemal -f root
Son giriş:  Sal Ağu 27 18:34:27 EEST 2013 kemal'dan pts/0 üzerinde
Linux kemal 3.7-trunk-686-pae #1 SMP Debian 3.7.2-0+kali8 i686
 Hoş gelmişsen gurban...çalış boş durma :)
root@kemal:~#
```

Gerçekten de çok uyarıcı(!) bir mesaj olmuş. Hazır bu konulara girmişken bir konudan daha bahsedelim. Sisteme giriş yapmış olan ve halen sistemde olan herkes **/var/run/utmp** içinde listelenir. Bu dosya sistem kapatıldığında veya yeniden başlatıldığında içeriği silinen bir doayadır. Sistem açık olduğu müddetçe kayıtları tutar. Ayrıca bu dosya basit txt dosyası

değildir. İkilik formatta bir data dosyasıdır.

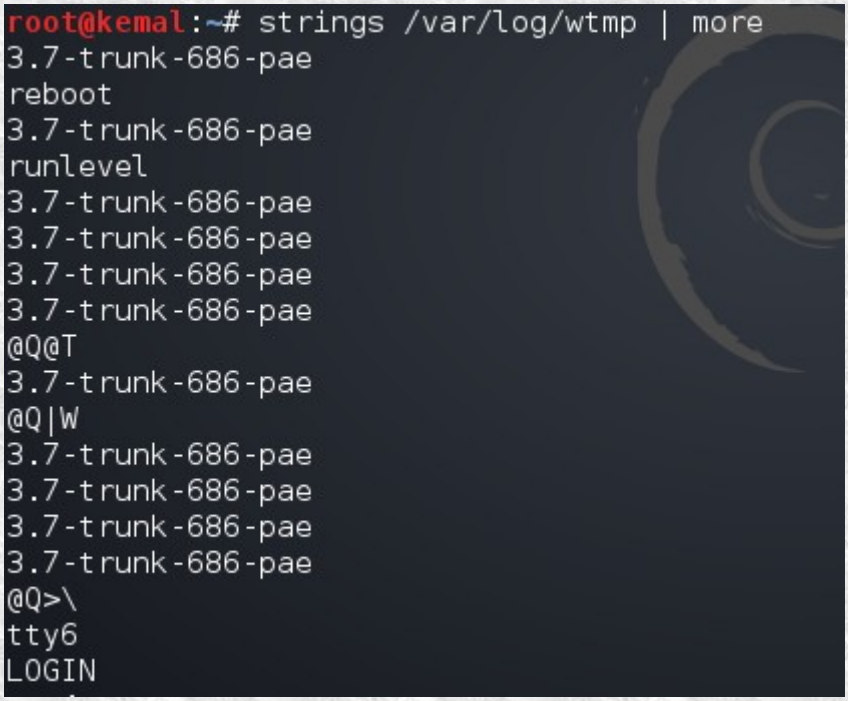
```
root@kemal:~# file /var/run/utmp
/var/run/utmp: data
root@kemal:~#
```

Dosyaların ne tür dosyalar olduğunu kontrol etmemize yarayan **file** komutundan yeri geldikçe bahsedeceğim. Bu ikilik formattaki dosyayı normal text editörlerle okuyamayız. Fakat **strings** komutuyla inceleyebiliriz. Bu yararlı komuttan da yeri geldikçe bahsedeceğim.

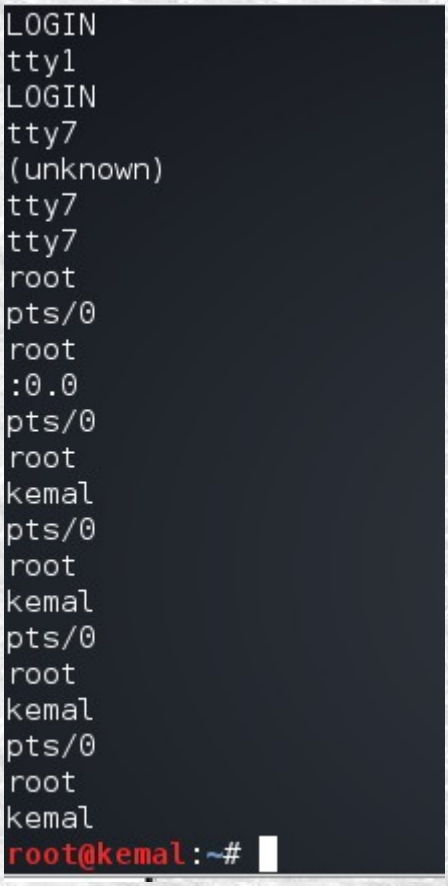
```
root@kemal:~# strings /var/run/utmp
reboot
runlevel
tty6
LOGIN
tty5
LOGIN
tty4
LOGIN
tty1
LOGIN
tty3
LOGIN
tty2
LOGIN
tty7
root
pts/0
root
kemal
root@kemal:~#
```

Sisteme kullanıcı ekledikten sonra da yine bu komutu kullanıp tuttuğu kayıtlara bakarız.

Şimdi bir de **/var/log/wtmp** dosyasına bakalım. Sisteme yapılan bütün başarılı bağlantılar /var/log/wtmp dosyası içindedir. Bu dosya da utmp dosyası gibi ikilik formattadır. Bu nedenle bu dosyayı da incelerken yine **strings** komutundan faydalanacağız. Komutun konsol çıktısı uzun olacağından **strings /var/log/wtmp | more** şeklinde bir kullanım uygun olacaktır.



Şimdi size iki ekran görüntüsü vereceğim ve bu ekran görüntülerini karşılaştırmanızı isteyeceğim. Birinci ekran görüntüsü:



Bu ekran görüntüsü biraz önce kullandığımız **strings /var/log/wtmp | more** komutunun çıktısının en son sayfası.

İkinci ekran görüntüsü:

```
root@kemal:~# last | more
root     pts/1        :0.0             Tue Aug 27 19:16   still logged in
root     pts/0        kemal            Tue Aug 27 18:49   still logged in
root     pts/0        kemal            Tue Aug 27 18:46 - 18:49  (00:02)
root     pts/0        kemal            Tue Aug 27 18:34 - 18:46  (00:12)
root     pts/0        kemal            Tue Aug 27 18:03 - 18:34  (00:31)
root     pts/0        :0.0             Tue Aug 27 17:16 - 18:03  (00:46)
root     tty7         :0               Tue Aug 27 17:08   still logged in
(unknown tty7         :0               Tue Aug 27 17:04 - 17:08  (00:04)
reboot   system boot  3.7-trunk-686-pa Tue Aug 27 17:03 - 19:16  (02:13)
root     pts/0        :0.0             Mon Aug 26 19:47 - down   (01:31)
root     tty7         :0               Mon Aug 26 19:40 - down   (01:38)
(unknown tty7         :0               Mon Aug 26 19:40 - 19:40  (00:00)
reboot   system boot  3.7-trunk-686-pa Mon Aug 26 19:40 - 21:18  (01:38)
root     pts/0        :0.0             Mon Aug 26 17:26 - down   (00:44)
root     pts/0        :0.0             Mon Aug 26 16:45 - 17:26  (00:41)
root     tty7         :0               Mon Aug 26 15:28 - down   (02:41)
(unknown tty7         :0               Mon Aug 26 15:28 - 15:28  (00:00)
reboot   system boot  3.7-trunk-686-pa Mon Aug 26 15:28 - 18:10  (02:42)
```

Bu görüntü de **last | more** komutunun çıktısı. **last** komutuyla sisteme yapılan en son giriş-çıkış kayıtları yani bağlantılar görüntülenir ve bu komut **/var/log/wtmp** içeriğini baz alır. Bu iki ekran görüntüsünü karşılaştırdığınız zaman kayıtların aynı olduğunu göreceksiniz.

Bu kadar şeyden bahsettik fakat hala yeni bir kullanıcıyı nasıl oluşturacağımızı görmedik. Şimdi kullanıcı oluşturmayı görelim. Sisteme yeni bir kullanıcı ekleyeceğimiz zaman kullanacağımız temel komutlar **adduser** ve **useradd** komutlarıdır. Siz her iki komutun da nasıl kullanılacağıyla ilgili bilgileri görüntülemek için **--help** komutunu kullanabilirsiniz. Ben daha kolayıma gelen **adduser** komutunu kullanacağım. Şimdi sistemde **zurna** adında yeni bir kullanıcı oluşturalım.

```
root@kemal:~# adduser zurna
Adding user `zurna' ...
Adding new group `zurna' (1000) ...
Adding new user `zurna' (1000) with group `zurna' ...
Creating home directory `/home/zurna' ...
Copying files from `/etc/skel' ...
Yeni parolayı girin:
Yeni parolayı tekrar girin:
passwd: şifre başarıyla güncellendi
zurna için kullanıcı bilgileri değiştiriliyor
Yeni değeri girin, veya varsayılan değer için ENTER'a basın
        Tam İsim []:
        Oda Numarası []:
        İş Telefonu []:
        Ev Telefonu []:
        Diğer []:
Is the information correct? [Y/n] y
root@kemal:~#
```

Nur topu gibi bir kullanıcımız oldu :) Zaten açıklamalar yapılmış ekranda fakat biz nelerin değiştiğine bakalım. Sistem zurna isminde bir **grup** oluşturmuş ve zurna kullanıcısını bu grubun üyesi yapmış. Oluşturulan bu yeni kullanıcı için **/home** altında **/zurna** adında bir dizin oluşturmuş. Daha önceden de hatırlayacağınız gibi kullanıcıların ev dizininden bahsetmiştik. İşte **/home/zurna** dizini de zurna kullanıcısına ait kişisel dizin (**ev dizini**). Bu dizine kendi kişisel dosyalarını atabilir. Ayrıca sisteme giriş yaptığında varsayılan olarak bu dizinde çalışmaya başlar. Şimdi **cd /home/zurna** ile kontrol edelim.

```
root@kemal:~# cd /home/zurna
root@kemal:/home/zurna#
```

Şu ana kadar işler yolunda. Yine hatırlarsanız sisteme yeni kullanıcı eklendiği zaman **/etc/passwd** ve **/etc/shadow** dosyalarına bu kullanıcıyla ilgili bilgiler yazılır demiştik. Bu iki dosyayı da inceleyelim. **cat /etc/passwd** komutuyla konsoldan passwd dosyasının içeriğine bakalım.

```
statd:x:117:65534::/var/lib/nfs:/bin/false
sslh:x:118:129::/nonexistent:/bin/false
saned:x:119:130::/home/saned:/bin/false
Debian-gdm:x:120:131:Gnome Display Manager:/var/lib/gdm3:/bin/false
privoxy:x:121:65534::/etc/privoxy:/bin/false
debian-tor:x:122:132::/var/lib/tor:/bin/false
zurna:x:1000:1000:,,,:/home/zurna:/bin/bash
root@kemal:~#
```

Bizim zurna en altta yerini almış. Bir de **cat /etc/shadow** komutuyla shadow dosyasını görelim.

```
saned:*:15775:0:99999:7:::
Debian-gdm:*:15775:0:99999:7:::
privoxy:*:15871:0:99999:7:::
debian-tor:*:15871:0:99999:7:::
zurna:$6$ttZw70Ii$LjmBdapfyNA6kKrXnAcF1xkgikQCGcLN3hsNrwZmEZkVNLPmo
0FAlecJine.Qv5v28u02p8z.:15944:0:99999:7:::
root@kemal:~#
```

Yine en altta bizim zurna ile ilgili kaydı görüyoruz. Şimdi de passwd ve shadow dosyalarına yeni kullanıcıyla ilgili yapılan kayıtların ne anlama geldiğine bakalım.

Önce passwd dosyasındaki kayıta bakalım. Kayıt şu şekilde yapılmış:

zurna:x:1000:1000:,,,:/home/zurna:/bin/bash

Yukarda gördüğünüz satır birbirinden **:** işaretiyle ayrılmış bir kaç bölümden oluşmuş. İlk bölüm kullanıcı ismi. x ile gösterilen bölüm eskiden parolaların şifrelenmiş halde bulunduğu yerdi. Şimdiyse parolalar şifrelenmiş olarak shadow dosyasında yer

almaktadır. Üçüncü bölümdeki 1000, kullanıcının numarasını, dördüncü bölümdeki 1000 ifadesi de kullanıcının bağlı olduğu grubu belirtir. Virgülle ifade edilen yerler de bizim kullanıcı oluştururken verdiğimiz bilgilerin (kullanıcının tam adı vs.) açıklamasıyla ilgilidir. Biz her hangi bir ek bilgi vermediğimizden o şekilde bırakılmış. Diğer bölümde kullanıcının home yani ev dizininin /home/zurna dizini olduğunu gösterir. Son bölüm ise kullanıcı sisteme girdiği zaman çalışacağı kabuğu belirtir.

Farkına vardığınız gibi **passwd** ve **shadow** dosyalarında kullanıcılar haricinde bir çok hesapla ilgili kayıtlar vardır. Bu kayıtlar sistemin kendine ait sistem hesaplarıyla ilgili bilgilerdir.

Kullanıcı hesaplarıyla ilgili shadow dosyasında tutulan kayıtın ne olduğuna bakalım. Yeni eklediğimiz zurna kullanıcısıyla ilgili tutulan kayıt şudur:

zurna:6ttZw7OIi$LjmBdapfyNA6kKrXnAcF1xkgikQCGcLN3hs NrwZmEZkVNLPmoRqD9KYF5TngV60FAlecJine.Qv5v28uO2p8z.:15944:0:99999 :7:::

Bu kayıt parola ile ilgilidir. En baştaki bölüm parolanın hangi kullanıcıya ait olduğunu gösterir. İkinci bölüm ise parolanın şifrelenmiş halidir. Bizim için en önemli bölüm bu ikinci bölümdür. Şimdi parolanın şifrelenmiş haline yakından bakalım:

6ttZw7OIi$LjmBdapfyNA6kKrXnAcF1xkgikQCGcLN3hsNrwZmEZkVNLPmoRqD9 KYF5TngV60FAlecJine.Qv5v28uO2p8z.

Linux sistemlerde parola **hash+salt** şeklinde saklanır. Hash, kriptografik özet demektir. Yani bir ifadenin (örneğin parolanın) çeşitli algoritmalara göre kriptolanmasıyla oluşturulan yeni ifadeye hash denir. Salt ise **tuzlama** denektir. Yani siz bir sisteme kullanıcı adı ve şifreyle kayıt olurken sizin belirlediğiniz parola şifrelenerek veri tabanında saklanır fakat güvenliği artırmak için sizin belirlediğiniz parolaya şifrelenmeden önce rastgele değerler eklenir ve bu şekilde şifrelenir. İşte rastgele eklenen bu değerlere salt denir. Evet bu kısa açıklamadan sonra yukarıdaki şifrelenmiş parolaya dönecek olursak:

İlk **$** işareti ve ikinci **$** işareti arasındaki sayı hangi şifreleme/hash algoritmasının kullanıldığını gösterir. Bu değer;

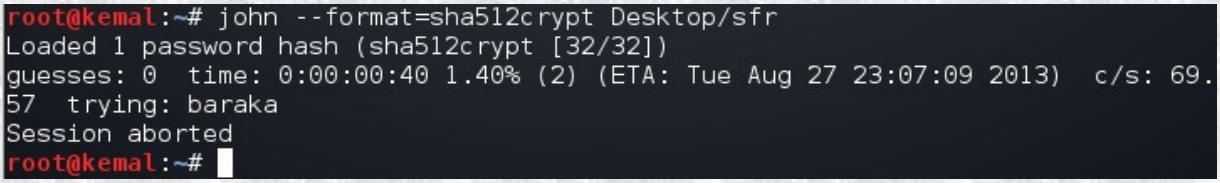
1 ise **MD5**
2 ise **Blowfish**
5 ise **SHA256**
6 ise **SHA512** algoritması ile şifrelendiğini gösterir.

İkinci **$** ile üçüncü **$** işareti arasındaki ifade **salt** değeridir. (tabi o da hash değeridir). Sonraki ifade ise parolanın şifrelenmiş halidir.

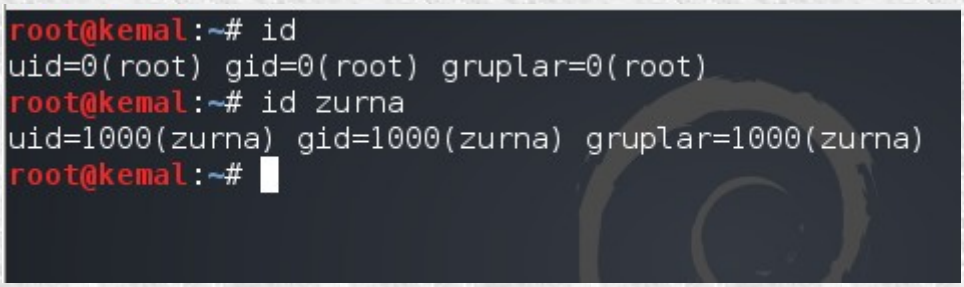
Tabi her şifreleme algoritmasının özet çıktısı (hash) farklı uzunlukta olacaktır. En çok tercih edilen şifre kırma yazılımlarından biri de **John The Ripper** (**JTR**) dir. Kırılmak istenilen şifreler ya da şifre bir dosyaya kayıt edilerek en temel kullanım şekliyle **john Desktop/sfr** gibi bir komutla denemeler yapılabilir. Bu komut benim masaüstümde kayıtlı olan

sfr isimli dosyamın içindeki şifreleri kırmaya çalışması için kullandığım komuttur. **JTR** bir çok farklı parametreyle ve wordlistlerle vs. kullanılabilir. Benim örnek olması için verdiğim komut en temel komuttur.

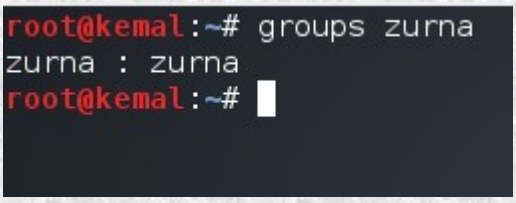
Tabi isterseniz şöyle bir komut da kullanabilirsiniz.



Her Linux sistemde JTR kurulu olacak diye bir kaide yok. Ben **Kali Linux** kullandığımdan her şey elimin altında :) Neyse biz kaldığımız yerden devam edelim. Demiştik ki bu zurna kullanıcısı zurna grubuna üye yapıldı. Bakalım öyle miymiş?

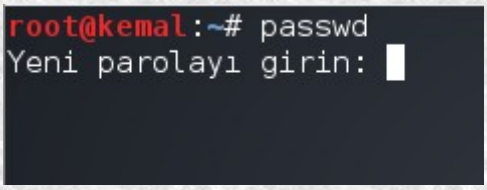


İlk önce kendime baktım (root) sonra da zurna kullanıcısına baktım. Bir kullanıcı farklı gruplara da üye yapılabilir. Bir kullanıcının hangi gruplara üye olduğunu görmek için de **groups <kullanıcı_adı>** komutunu kullanırız.

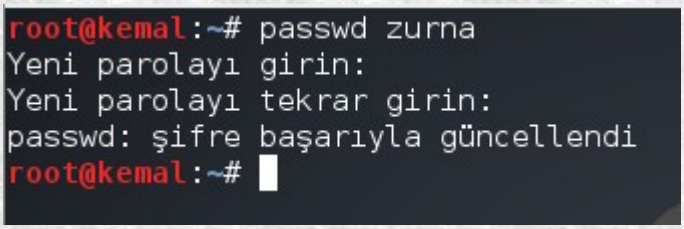


Yeni grup oluşturma vs. gibi konulardan da behsedilebilir fakat şu an için bunlar yeterli diye düşünüyorum.

Evet kullanıcının nasıl oluşturulduğunu gördük. Peki kullanıcının şifresini değiştirme işlemini nasıl yapıyoruz? Çok basit bir şekilde **passwd** komutunu kullanarak. Komutu bu haliyle kullandığınız zaman kendi root şifrenizi değiştirmiş olursunuz.



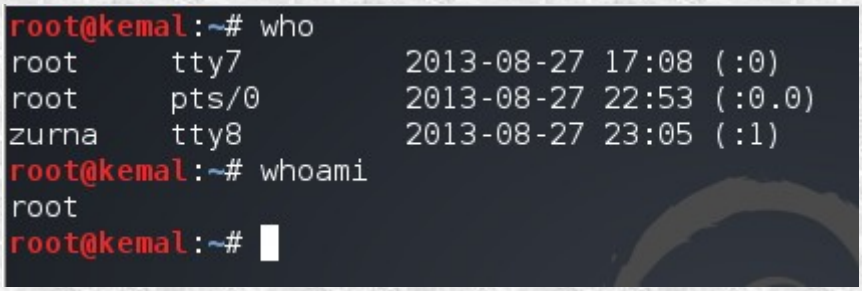
Eğer herhangi bir kullanıcının şifresini değiştirmek isterseniz **passwd <kullanıcı_adı>** komutunu kullanmanız gerekir.



Kullanıcı hesabını kilitlemek için **usermod -L <kullanıcı_adı>** komutu kullanılır. Tekrar aktif hale getirmek için de **usermod -U <kullanıcı_adı>** komutu kullanılmalıdır.

Tabi **usermod** komutu sadece bunun için değildir. Kullanıcı hesaplarıyla ilgili bir çok değişikliği bu komutla yapabilirsiniz. Kullanım parametreleriyle ilgili yardım ekranını görüntüleyerek bilgi edinebilirsiniz.

Şimdi de **who** ve **whoami** komutlarından bahsetmek istiyorum.Bu komutlardan who komutu ile sisteme bağlı kullanıcılar listelenir, whoami (**ben kimim**) komutuyla da sisteme giriş ismi görülür.



Bir **chage** komutunun kullanımına bakalım. Bu komutla sistemdeki kullanıcıların geçerli olan şifre parametreleri görülebilir.

```
root@kemal:~# chage -l zurna
Son Parola Değişimi                                     : Ağu 27, 2013
Parola Kullanım Süresi Dolumu                                   : Hiçbir zaman
Parola Pasif                                          : Hiçbir zaman
Hesap Bitimi                                            : Hiçbir zaman
Şifre değişiklikleri arasındaki en az gün sayısı                : 0
Maksimum giriş denemesi sayısı aşıldı          : 99999
Şifre süresinin dolumundan önceki uyarı gün sayısı      : 7
root@kemal:~#
```

Aynı zamanda chage komutuyla parola süresi değiştirilebilir. Örneğin zurna kullanıcısının parolasının süresinin 27 Ağustos 2014 de dolması isteniyorsa **chage -E 2014/08/27 zurna** komutu kullanılmalıdır.

```
root@kemal:~# chage -E 2014/08/27 zurna
root@kemal:~# chage -l zurna
Son Parola Değişimi                                     : Ağu 27, 2013
Parola Kullanım Süresi Dolumu                                   : Hiçbir zaman
Parola Pasif                                          : Hiçbir zaman
Hesap Bitimi                                            : Ağu 27, 2014
Şifre değişiklikleri arasındaki en az gün sayısı                : 0
Maksimum giriş denemesi sayısı aşıldı          : 99999
Şifre süresinin dolumundan önceki uyarı gün sayısı      : 7
root@kemal:~#
```

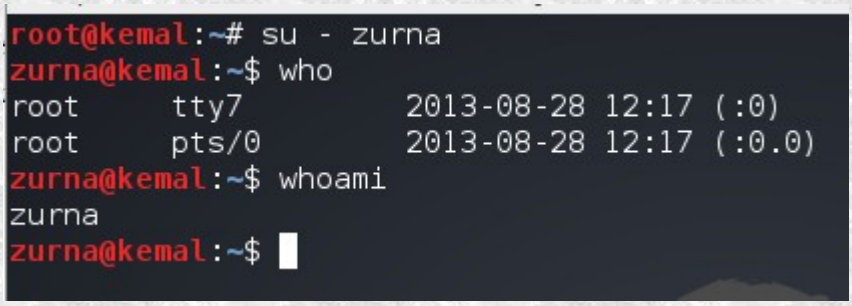
Yukarıdaki görüntüde bulunan hesap bitimi bölümüne dikkat edin.

Root kullanıcı sistemdeki süper yetkili kulanıcı olduğundan her şey serbestti ona. O zaman başka kullanıcıların kimliğine de rahatça bürünebilir. Evet root kullanıcı herhangi bir kullanıcının kimliğine bürünmek için yani onun kimliğiyle sistemde çalışmak için **su <kullanıcı_adı>** ya da daha güzel bir komut olan **su - <kullanıcı_adı>** komutunu kullanabilir. İlk komutla diğer kullanıcının kimliğini kullanmaya, diğer komutla da hem o kullanıcı olmaya hem de direk onun kabuğunda çalışmaya başlarsınız. Örnekle gösterelim.

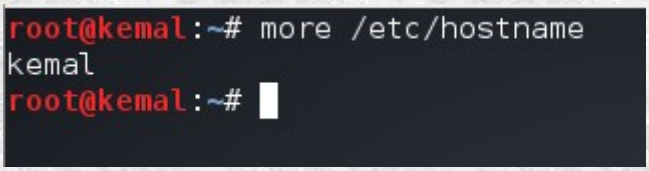
```
root@kemal:~# su zurna
zurna@kemal:/root$ ls
deneme     deneme.s  Downloads
deneme.o   Desktop   install_flash_player_11_linux.i386.tar.gz
zurna@kemal:/root$
```

```
root@kemal:~# su - zurna
zurna@kemal:~$ ls
Desktop
zurna@kemal:~$
```

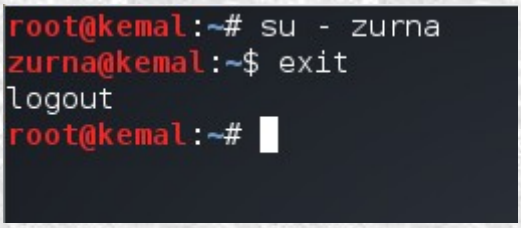
Mesela anlaşılmıştır zannederim. Şimdi daha önce bahsettiğimiz **who** ve **whoami** komutlarının arasındaki farkı daha iyi anlayabilmek için zurna kullanıcısının kimliğindeyken bu komutları kullanalım.



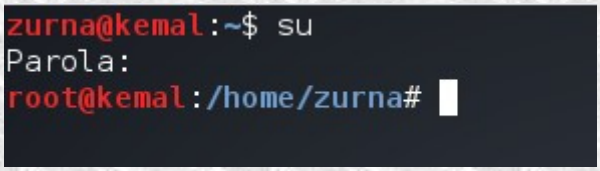
Kullanıcı ismini zurna olarak seçmek hiç de iyi olmadı :) Orada gördüğünüz **kemal** ifadesi host adıdır. Yani bilgisayarımızın adı. Host adı **/etc/hostname** dosyasında bulunur. Bu dosyadaki ismi istediğiniz gibi değiştirebilirsiniz. Şimdi **cat /etc/hostname** ya da **more /etc/hostname** komutunu kullanarak dosyaya bakalım.

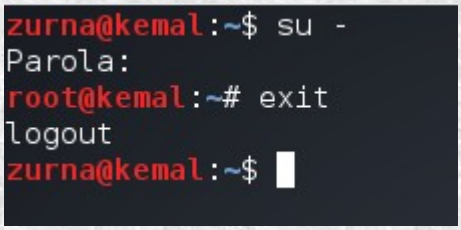


Bu arada bir kullanıcı kimliğinde çalışırken kendi kimliğimize nasıl döneceğiz derseniz, **exit** komutuyla bu işi yapabiliriz.



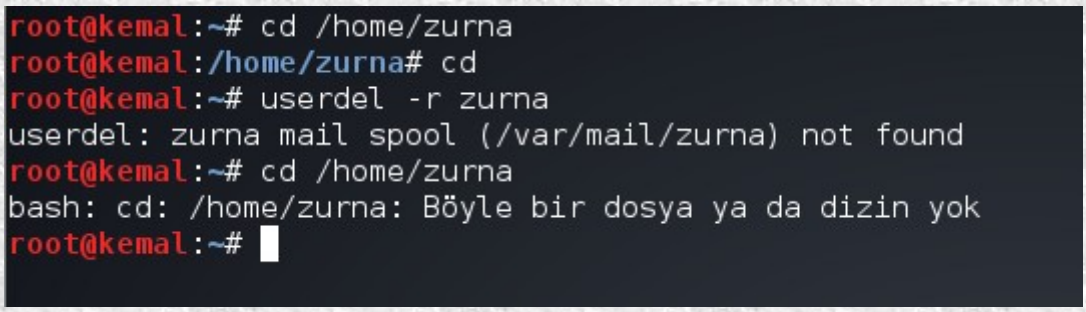
Yine başka bir kullanıcı kimliğindeyken, **su** ve **su –** komutlarının kullanımına bakalım. Hatırlarsanız su komutuyla diğer kullanıcı oluruz fakat kendi dizinimizde çalışırız, su – komutuyla da hem diğer kullanıcı oluruz hem de onun dizininde çalışırız demiştim. Herhangi bir kullanıcı **root** kullanıcı olmak isterse **su** ya da **su –** komutlarını olduğu gibi kullanabilir. Yani **su root** ya da **su – root** demesine gerek yoktur. Root kullanıcı olmak isterseniz root kullanıcının **parolasını** bilmek zorundasınız.





İlerleyen konularda bahsedeceğimiz gibi her kullanıcı kendi yetkileri çerçevesinde sistemde işlemler yapabilir. Root kullanıcı için her hangi bir kısıtlama yoktur. Fakat diğer kullanıcılar yetkileri çerçevesinde hareket etmek zorundadır.

Root kullanıcı sistemdeki kullanıcılarla ilgili her türlü değişikliği yapmaya yetkilidir. Kullanıcının parolasını değiştirebilir, engelleyebilir, belli bir süre sonra parolanın geçerliliğinin bitmesini sağlayabilir ve hesabı tümden silebilir. Şimdi bir kullanıcı hesabının nasıl silineceğini inceleyelim. Sistemdeki bir kullanıcının hesabını silmek için **userdel** komutu kullanılır. Bu komutu **userdel <kullanıcı_adı>** şeklinde kullanırsak passwd ve shadow dosyalarındaki o kullanıcıyla ilgili kayıtlar silinir. Eğer **userdel -r <kullanıcı_adı>** şeklinde kullanırsak bunlara ek olarak kişisel ev dizini de silinir.



Ben **userdel** kullandım fakat siz isterseniz **deluser** komutunu da kullanabilirsiniz. Her iki komutla ilgili **userdel --help** ve **deluser --help** komutlarıyla kullanım parametreleri hakkında bilgi alabilirsiniz.

Dosya Ve Dizinlere Erişim Yetkileri

Linux işletim sisteminde, dosya ve dizinlerle ilgili güçlü bir koruma sistemi vardır. Önceki konuda da bahsettiğimiz gibi root kullanıcı sistemdeki en yetkili kullanıcıdır ve onunla ilgili her hangi bir yetki kısıtlaması yoktur. Fakat sistemdeki diğer kullanıcılar her istediklerini yapma yetkisine sahip değildirler. Sonuçta her kullanıcı dizin ve dosyalarla çalıştığı

için sistem güvenliğinin sağlanması bu dosya ve dizinlerin koruma altında olmasıyla sağlanır. Her kullanıcı kendi yetkisi dahilinde işler yapabilir. Örneğin root kullanıcının yazma yetkisinin bulunduğu önemli bir dosyaya sıradan bir kullanıcının da yazma yetkisi olursa bu durum önemli bir zaafiyet oluşturur. Bu güvenlik mekanizması sadece kötü niyetli kullanıcılar için değil aynı zamanda acemi kullanıcılar için de bir tedbirdir.

Kullanıcılar bir dosya ya da dizinle ilgili üç farklı eylemde bulunabilirler. Bunlar **okuma**, **yazma** ve **çalıştırma** eylemleridir. Sistemdeki kullanıcıların ya da grupların neleri yapıp neleri yapamayacağı düzenlenebilir.

Önceki konulardan da hatırlayacağınız gibi bir dosya ya da dizin içeriğini listelemek için **ls -l** komutunu kullanıyorduk. Bu komutun çıktısında dosya ve dizinlerle ilgili bazı ifadeler yer alıyordu. Şimdi tekrar bakalım ve bu ifadelerin ne anlama geldiğini görelim.

```
root@kemal:~# ls -l
toplam 6788
-rwxr-xr-x  1 root root     627 Mar 17 12:54 deneme
-rw-r--r--  1 root root     604 Mar 17 12:54 deneme.o
-rw-r--r--  1 root root     233 Mar 17 16:13 deneme.s
drwxr-xr-x 10 root root    4096 Ağu 28 14:06 Desktop
drwx------  6 root root    4096 Haz 18 16:12 Downloads
-rw-r--r--  1 root root 6923111 Haz  3 19:42 install_flash_pl
tar.gz
-rwxr-xr-x  1 root root     112 Ağu 20 17:50 merhaba.py
root@kemal:~#
```

Listelediğimiz dizin içeriğinde bulunan ve dosya/dizinlerle ilgili bilgi satırlarının başındaki kodlamalara dikkat edin. **rwxrwxrwx** formatındaki bu kodlamaların başında bulunan **–** işareti bunun bir dosya olduğunu, **d** harfi ise bunun bir dizin (directory) olduğunu gösterir.9 karakterden oluşan bu kodlamaları üçerli gruplar halinde düşünmeliyiz. Yani **rwx rwx rwx** gibi. Birinci grup dosya/dizin **sahibinin** yetkilerini, ikinci grup **dosyanın sahibiyle aynı grupta** bulunan kullanıcıların yetkilerini, üçüncü grup ise **diğer** (genel) kullanıcıların yetkilerini ifade etmektedir. Kodlamadaki karakterler şu anlama gelir:

r : okuma yetkisi (read)
w : yazma yetkisi (write)
x : çalıştırma yetkisi (execute)

Şimdi yukarıdaki ekran görüntüsünde bulunan örneğin deneme dosyasıyla ilgili satırda bulunan ve yetkileri ifade eden kodlamaları inceleyelim. Kodlamamız şu:

-rwxr-xr-x

En başta bulunan **–** işareti bunun bir dosya olduğunu gösterir demiştik. Şimdi diğer ifadeyi üçerli gruplara ayıralım. Yani şu şekilde düşünelim: **rwx r-x r-x** Evet şimdi bunlar ne anlama geliyor tek tek inceleyelim:

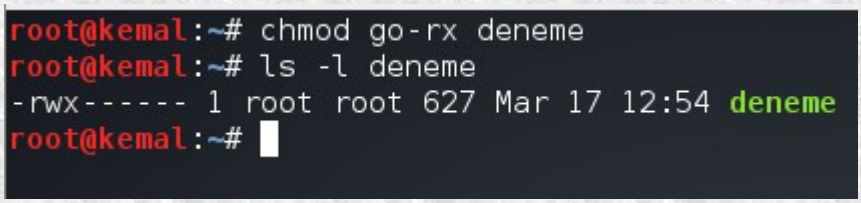
rwx : Dosyanın sahibinin okuma, yazma ve çalıştırma yetkileri var.
r-x : Dosyanın sahibiyle aynı grupta bulunan kullanıcıların okuma ve çalıştırma yetkisi var.
r-x : Diğer kullanıcıların okuma ve çalıştırma yetkisi var.

Dosyamızla ilgili yetkiler bunlarmış. Şimdi bir kaç alıştırma yapalım. Aşağıdaki kodlamaların ne anlama geldiğine bakalım.
rwxrwxrwx : Bu dosyayı tüm kullanıcılar okuyabilir, yazabilir, eğer çalıştırılan bir dosya ise çalıştırabilir. Dosyayı silebilir. Yani tehlikeli bir durum.
rwxr-xr-- : Bu dosyayı herkes okuyabilir, sahibi ve sahibiyle aynı gruptaki kullanıcılar çalıştırabilir(eğer çalıştırılabilir dosya ise), sadece sahibi yazabilir (isterse silebilir anlamına da gelir) ve diğer kullanıcılar da sadece okuyabilir.
rwx------ : Dosyanın sahibi dosyayla ilgili her türlü işlemi yapabilir, diğer kullanıcılara dosya tamamen kapalı.

Dosya ve dizinlerle ilgili erişim yetkilerini istediğimiz gibi değiştirebiliriz. Tabi root kullanıcı olarak. Bunun için **chmod** komutundan faydalanabiliriz. Komutun en kolay kullanım şekli **chmod <ugoa> <+=-><rwxst><dosya/dizin>** kullanımıdır.
u : dosya ya da dizinin sahibi
g : dosya ya da dizin sahibiyle aynı gruptaki kullanıcılar
o : diğer kullanıcılar
a : herkes
+ : yetki ekleme
– : yetki çıkarma
= : yetki eşitleme
r : okuma yetkisi
w : yazma yetkisi
x : çalıştırma yetkisi
s : suid biti
t : sticky bit

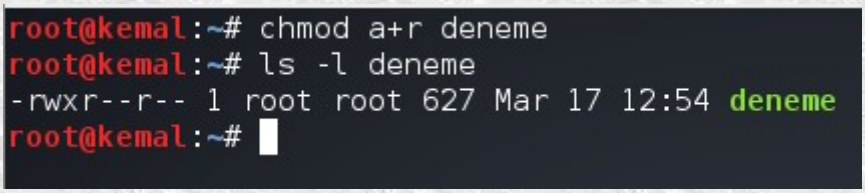
Örnek bir dosya üzerinde işlemler yapalım.

Yukarıda da görüldüğü gibi deneme dosyasının erişim yetkileri **rwxr-xr-x** olarak görülüyor. Bu dosyanın erişim yetkilerini değiştirelim. Örneğin sahibi haricinde dosya diğer tüm kullanıcılara kapalı olsun. Bunun için **chmod go-rx** komutunu kullanalım.



Evet istediğimiz gibi düzenlenmiş. Bu komutla şunu demiş olduk: Grup (g) ve diğer (o) kullanıcıların okuma (r) ve çalıştırma (x) yetkilerini kaldır (-). Zaten bu kullanıcıların yazma (w) yetkisi yoktu.

Şimdi de tüm kullanıcılar için okuma yetkisi verelim. Bunun için de **chmod a+r deneme** komutunu kullanmamız yeterli olacaktır.



Kullandığımız komutla, bütün (all) kullanıcılara okuma (r) yetkisi ver (+) demiş olduk.

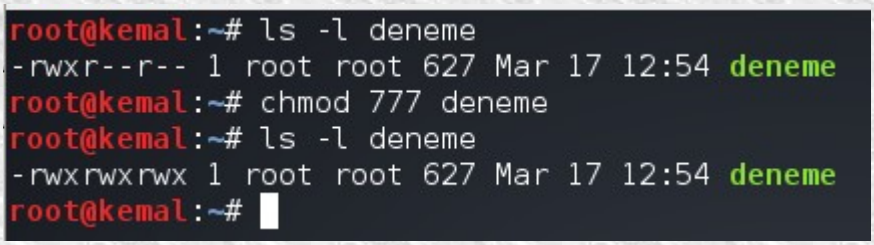
Yetkileri düzenlemek için kullandığımız **chmod** komutunu daha pratik bir formatta kullanabiliriz. Yetkileri sayısal olarak belirterek pratik bir şekilde değiştirebiliriz. Yetki durumunu gösteren kalıbın sayısal karşılıkları şu şekildedir:

dosyanın sahibi	dosyanın sahibiyle aynı gruptakiler	diğer kullanıcılar
r = 4	**r = 4**	**r = 4**
w = 2	**w = 2**	**w = 2**
x = 1	**x = 1**	**x = 1**

Yukarıdaki eşleştirmelere göre bundan sonra örneğin **rwx** yerine 4+2+1 yani **7** kullanabiliriz. Birkaç örneğe bakalım:

rwxrwxrwx = **777**
rwx------ = **700**
rwxr-xr-x = **755** (r-x 4+0+1 olduğundan 5 ile gösterilir)

Örnekleri incelerseniz bu yöntemin çok daha kolay ve pratik olduğunu göreceksiniz. Şimdi bu formatta kullnıma da bir örnek yapalım. Örneğimiz yine aynı dosyayla ilgili olsun.



Görüntüyü incelersek **chmod 777 deneme** komutuyla **deneme** dosyasının erişim yetkilerini **rwxrwxrwx** yapıyoruz. Yani tüm kullanıcılar istediği her değişikliği yapabilir dosya üzerinde. Biz böyle olmasını istemeyiz. O zaman yetkileri değiştirelim. Örneğin sahibi tam yetkili olsun (rwx=7), sahibiyle aynı gruptakiler ve diğer kullanıcılar da okuma ve çalıştırma (r-x=4+0+1) yetkisine sahip olsun. Yani yetkiler rwxr-xr-x şeklinde olsun. Bu durumda kullanmamız gereken komut **chmod 755 deneme** olacaktır.

```
root@kemal:~# ls -l deneme
-rwxrwxrwx 1 root root 627 Mar 17 12:54 deneme
root@kemal:~# chmod 755 deneme
root@kemal:~# ls -l deneme
-rwxr-xr-x 1 root root 627 Mar 17 12:54 deneme
root@kemal:~#
```

Bu formatta kullanım bazen çok daha pratiktir.

Küçük bir örnek daha yapalım. Örneğin root kullanıcı kendi ev dizininde önemli adında bir dosya oluşturmuş olsun. Bu dosyanın izinlerine bakalım. Sonra erişim izinlerini sahibinin dışındaki kullanıcılara tamamen kapatalım.

```
root@kemal:~# ls -l önemli
-rw-r--r-- 1 root root 0 Ağu 28 17:59 önemli
root@kemal:~# chmod 700 önemli
root@kemal:~# ls -l önemli
-rwx------ 1 root root 0 Ağu 28 17:59 önemli
root@kemal:~# su zurna
zurna@kemal:/root$ ls
deneme    deneme.s  Downloads                                merhaba.py
deneme.o  Desktop   install_flash_player_11_linux.i386.tar.gz  önemli
zurna@kemal:/root$ cat önemli
cat: önemli: Erişim engellendi
zurna@kemal:/root$
```

Gördüğünüz gibi erişim engellendi. Diğer kullanıcılar bu dosya üzerinde okumak da dahil her hangi bir değişiklik yapamazlar.

Eğer bir dizinin erişim yetkilerini alt dizin ve onların da altındaki dosyalarla birlikte değiştirmek isterseniz **chmod -R <755> <dizin>** gibi bir komut kullanmanız gerekir.

Yeri gelmişken bir konudan daha bahsedelim. Bazen yanlışlıkla bir dosyayı silebilirsiniz. Ya da sistemle ilgili bir konfigürasyon dosyasını düzenlersiniz fakat sistem yeniden başladığında sizin düzenlediğiniz ayarların kaybolduğunu görürsünüz. Tüm bunları engellemek için **chattr** komutundan faydalanabilirsiniz. Bu komutla bir dosya üzerinde root kullanıcı olarak bile değişiklik yapılmasını engellemiş olursunuz. Tabi

tekrar aynı komutla dosyanın özelliğini değiştirene kadar. Şimdi bir örnek verelim. Ev dizinimizde belge1 adında bir dosya olsun. Bu dosyayı değiştirilemez yapmak için **chattr +i belge1** komutunu kullanabiliriz. Sistemde bu şekilde dosya olup olmadığını da **lsattr** komutuyla görebiliriz. Eğer bir dizinin altındaki tüm dosyaları kontrol etmek isterseniz komutu **lsattr -R <dizin>** şeklinde kullanabilirsiniz.

```
root@kemal:~# chattr +i belge1
root@kemal:~# lsattr
-------------e-- ./deneme.s
-------------e-- ./deneme
----i--------e-- ./belge1
-------------e-- ./install_flash_player_11_linux.i386.tar.gz
-------------e-- ./Desktop
-------------e-- ./Downloads
-------------e-- ./merhaba.py
-------------e-- ./deneme.o
root@kemal:~#
```

Artık belge1 dosyamız değiştirilemez olduğuna göre deneyelim bakalım gerçekten root kullanıcı olarak bile dosya üzerinde herhangi bir değişiklik yapabiliyor muyuz?

```
root@kemal:~# rm belge1
rm: `belge1' silinemedi: İşleme izin verilmedi
root@kemal:~#
```

Dosyamızı silmeye çalıştık fakat silemedik. Şimdi bir de dosyaya bir şeyler yazmaya çalışalım.

```
root@kemal:~# echo "bakalım yazabilecek miyiz?" > belge1
bash: belge1: Erişim engellendi
root@kemal:~#
```

Gördüğünüz gibi root kullanıcı olduğumuz halde hiç bir şey yapamıyoruz. Amaç da zaten yanlışlıkla ya da başka sebeplerden bir şey yapılmasını engellemek. Dosyayı geri eski haline getirmek için de **chattr -i belge1** komutunu kullanıyoruz.

```
root@kemal:~# chattr -i belge1
root@kemal:~# lsattr
-------------e-- ./deneme.s
-------------e-- ./kk
-------------e-- ./deneme
-------------e-- ./belge1
-------------e-- ./install_flash_player_11_linux.i386.tar.gz
-------------e-- ./Desktop
-------------e-- ./Downloads
-------------e-- ./merhaba.py
-------------e-- ./deneme.o
root@kemal:~#
```

Evet artık dosyamız eski haline geldi ve değiştirilemez özelliğinden kurtuldu. Bunu **lsattr** komutunu kullanarak da anlayabiliyoruz. Şimdi dosyaya yazmaya çalışalım bakalım izin verecek mi?

```
root@kemal:~# echo "kkjkdjk" > belge1
root@kemal:~# cat belge1
kkjkdjk
root@kemal:~#
```

Evet işler yolunda. Bu konuyu da kapatalım başka konuya geçelim.

Son olarak şu gizemli **suid biti** meselesinden de bahsederek bu bölümü bitirelim. Bir programı çalıştıran kullanıcıların program çalıştığı sürece program dosyasının sahibinin yetkilerine sahip olmalarını sağlayan şey suid biti dediğimiz şeydir. Yani sistemde bir programı kullanması gereken fakat o programı kullanmasına normal şartlarda yetkisi olmayan bir kullanıcının geçici olarak programı kullanabilmesi de diyebiliriz. Örneğin sistemde normal bir kullanıcı var. Bu kullanıcı hesabıyla ilgili bazı değişiklikler yapmak istiyor. Fakat root kullanıcı olmadığından bunu yapamayacakken eğer kullanacağı program daha önceden suid biti ayarlanmış bir programsa bu programı kullanarak gerekli olan değişiklikleri yapabiliyor. Aslında o programın çalışırken erişmesi gereken dosyaya eğer kullanıcının izni yoksa, geçici olarak bu izni olmuş oluyor ilgili programı kullandığı sürece. Mesele kısaca böyle. Sistemde suid bite sahip programları bulmak için, örneğin /usr/bin altındaki suid bite sahip dosyaları bulmak için **find /usr/bin -perm -4000** komutu kullanılır.

```
root@kemal:~# find /usr/bin -perm -4000
/usr/bin/X
/usr/bin/chfn
/usr/bin/at
/usr/bin/procmail
/usr/bin/slock
/usr/bin/fping6
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/fping
/usr/bin/lppasswd
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/sudoedit
/usr/bin/chsh
root@kemal:~#
```

Şimdi çalıştırılabilir bu program dosyalarından bir tanesinin erişim yetkilerine bakalım. Örneğin **sudo** nun erişim

yetkilerine bakalım.

```
root@kemal:~# ls -l /usr/bin/sudo
-rwsr-xr-x 2 root root 119172 Mar  1 07:44 /usr/bin/sudo
root@kemal:~#
```

Erişim yetkilerine baktığımızda **rws** ifadesini görüyoruz. Buradaki **s** , suid bitinin aktif olduğunu gösteriyor.

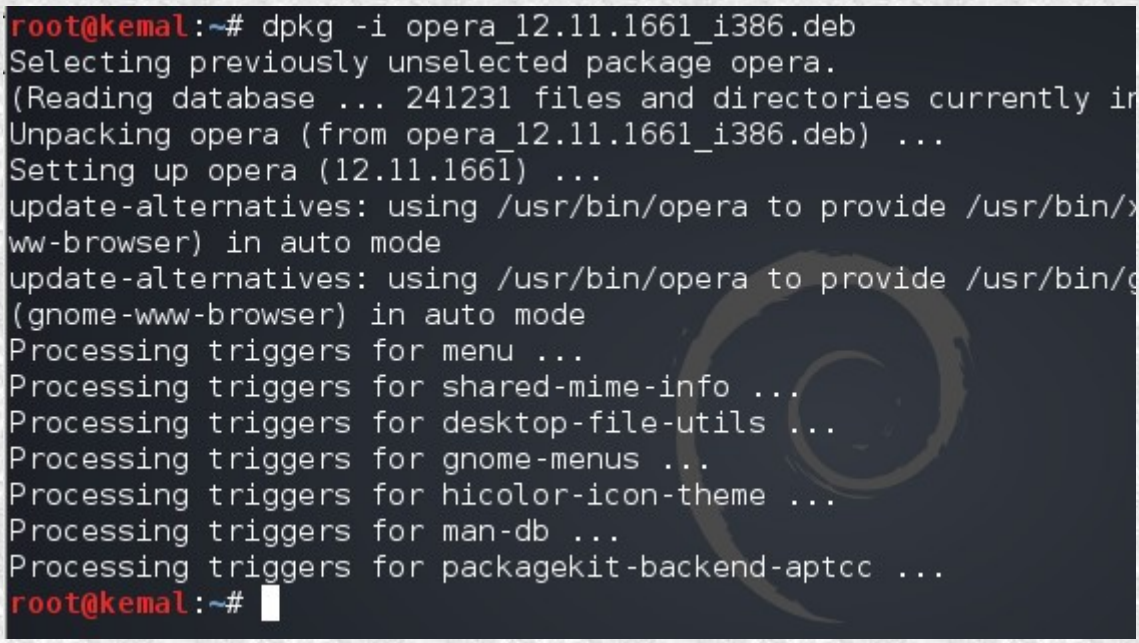
Program Kurma-Kaldırma-Sistem Güncelleme

Linux'ta program kurmak Windows'tan oldukça farklıdır ve aslında çok basittir. Linux'ta program kurmak için bir kaç seçeneğimiz vardır. Bunlardan bir tanesi ihtiyacımız olan programı **kaynak koddan** derleyerek kurmaktır. İkincisi, kullandığımız dağıtıma uygun olan program kurulum paketlerinden **paket yönetim sistemi** yardımıyla kurulum yapmaktır. Bir diğer seçenek de **repository** denen dağıtımın kullandığı depolardan otomatik kurulum yapmaktır.

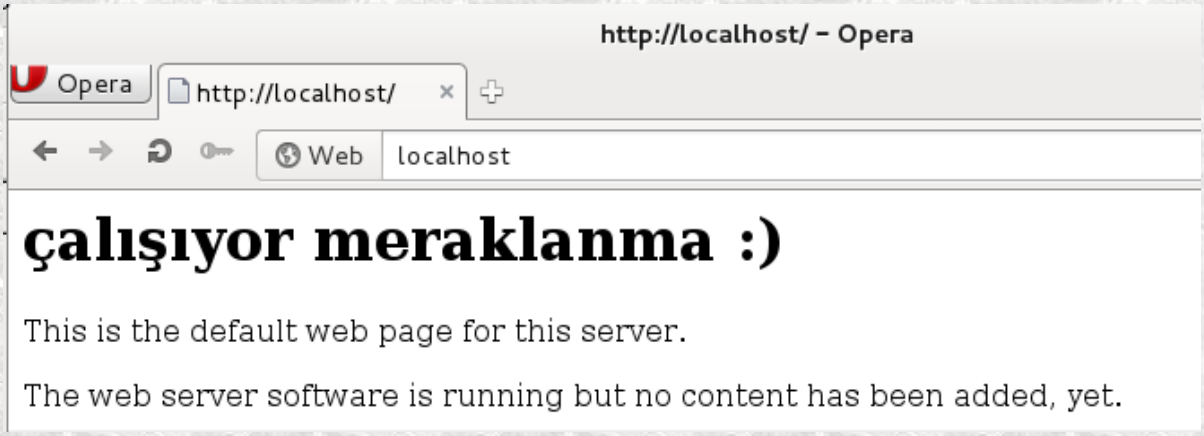
Derlenmiş ve paket yönetim sistemiyle kuruluma hazır dosyalara **paket** denilmektedir. Farklı Linux dağıtımları farklı paket dosyaları kullanmaktadır. Günümüzdeki Linux dağıtımların büyük çoğunluğu iki tip paket türü kullanmaktadır. **RedHat/CentOS/Fedora** ve benzer dağıtımlar **rpm** paketini kullanırken **Debian/Ubuntu** ve benzer Linux dağıtımları da **deb** paketini kullanmaktadır. Tabi bu paket dosyalarının kurulumunu yapmak için de farklı paket yönetim sistemleri kullanılmaktadır. Yine **RedHat/CentOS/Fedora** dağıtımları **rpm** paket yöneticisini kullanırken **Debian/Ubuntu** dağıtımları **dpkg** paket yöneticisini kullanmaktadır. Bu bahsettiklerim manual paket yönetim sistemleridir. Bir de program repolarından otomatik olarak paketlerin indirilip kurulmasını sağlayan yöneticiler vardır. Bunlar da rpm paketlerini kullanan dağıtımlar için **yum** , deb paketlerini kullanan dağıtımlar için de **apt** paket yöneticileridir.

Şimdi ilk önce **Debian** tabanlı Linux dağıtımlarda program kurma, program kaldırma, sistem güncelleme işlemlerine bakalım daha sonra da **RedHat** tabanlı sistemlerde bu işler nasıl oluyor onu inceleyelim. Örneğin sisteminize Opera tarayıcıyı kurmak istiyorsunuz. Opera tarayıcıya ait deb paketini indirdiniz. Manuel olarak kurulumunu yapmak istiyorsunuz. Bunun için **dpkg** paket yönetim sistemini kullanacağız.

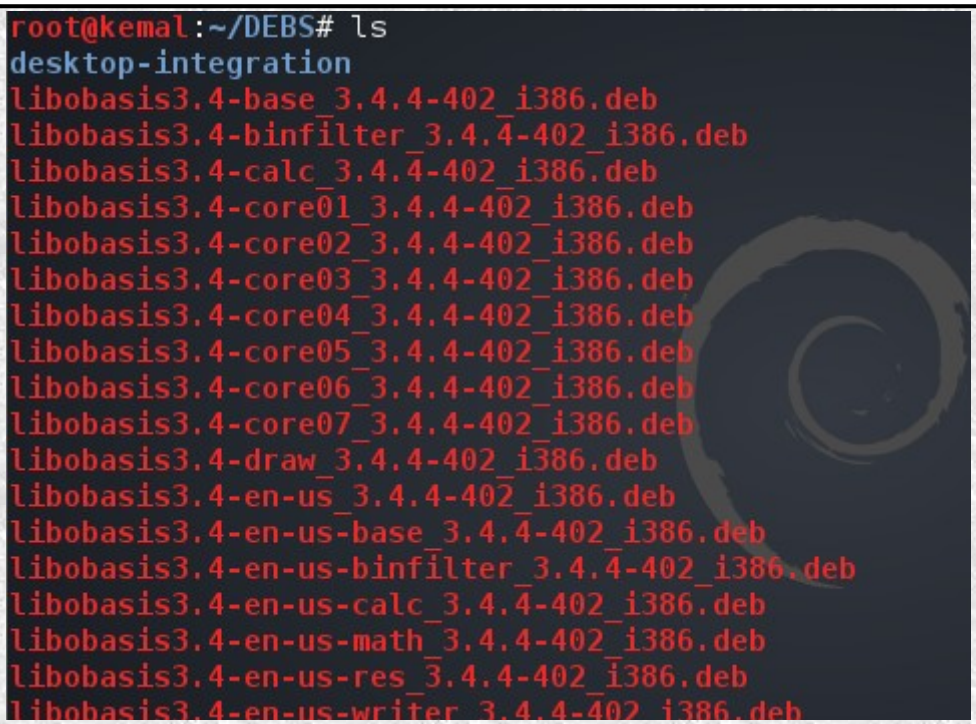
Gördüğünüz gibi paketimiz deb uzantılı. Paketimiz hangi dizindeyse o dizinin yolunu belirterek komutumuzu veriyoruz. Örneğin ben ev (home) dizinime kaydettim o yüzden komutu şu şekilde kullanıyorum: **dpkg -i <paket.deb>** Komutu kullanırkan **dpkg -i** kısmından sonra deb paketinin isminin ilk bir kaç harfini yazarak **tab** tuşuna basarsanız komut otomatik tamamlanacaktır. Örneğin **dpkg -i opera** yazıp **tab** tuşuna bastığınızda komut tamamlanacaktır.



Opera tarayıcıyı kurduk. Ben kendi localhostumda denemek istiyorum.



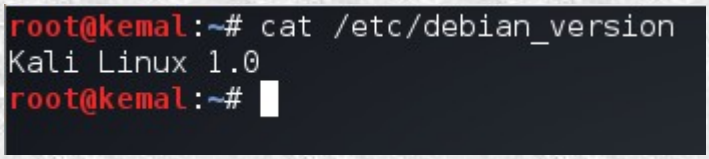
Yukarda yazan "çalışıyor meraklanma" ifadesi Opera ile ilgili değil. Apache nin çalışmasıyla ilgili :) Evet deb uzantılı program dosyalarının kurulumu temel olarak bu şekilde. Diyelimki ofis programına ihtiyacınız var ve **LibreOffice** ofis programını kurmak istiyorsunuz. Bununla ilgili paketleri indirdiniz fakat baktığınızda bir çok deb paketinden oluştuğunu gördünüz. Bu paketleri tek tek kurmak yerine hepsini birden kurabiliriz.



Gördüğünüz gibi LibreOffice ile ilgili bir çok paket var ve hepsinin de kurulması gerekiyor. Bunu **dpkg -i *.deb** komutunu kullanarak kolayca halledebiliriz. Tabi bu deb paketleriyle ilgili dizinde olmanız gerekir.

Bu komutla, uzantısı deb olan bütün paketlere işlemi uygula demiş oluyoruz.

Burada küçük bir bilgi verelim. Debian versiyonunu öğrenmek için **cat /etc/debian_version** komutunu kullanabiliriz.



Manuel kurulumda sisteminize uygun paketleri bulup yüklemeniz gerekir. Yani o program paketi (deb paketi) düzgün kurulup çalışabilmesi için sisteminizde diğer bazı paketlere ihtiyaç duyabilir. Bu duruma bağımlılık (depends) denir. Program çalışmazsa o programla ilgili **bağımlı paketlerin** de yüklenmesi gerekir. Eğer daha önceden sisteminizde bu paketler varsa sorun çıkmadan program kurulup çalışacaktır. Bir paketle ilgili bilgi almak için **dpkg --status <paket_adı>** komutu kullanılabilir. Örneğin kurduğumuz opera tarayıcısıyla ilgili bilgileri almak için **dpkg --status opera** komutunu kullanalım. İsterseniz komutu **dpkg -s opera** şeklinde de kullanabilirsiniz.

```
root@kemal:~# dpkg --status opera
Package: opera
Status: install ok installed
Priority: optional
Section: non-free/web
Installed-Size: 43881
Maintainer: Opera Packaging Team <packager@opera.com>
Bugs: https://bugs.opera.com/wizard/
Architecture: i386
Version: 12.11.1661
Provides: www-browser, mail-reader, imap-client, news-reader
Depends: libc6 (>= 2.3.6-6~), libc6 (>= 2.8), libfontconfig1,
.2.1), libgcc1 (>= 1:4.1.1), libglib2.0-0 (>= 2.16.0), libgstr
0.10-0 (>= 0.10.16), libgstreamer0.10-0 (>= 0.10.15), libice6,
6 (>= 4.1.1), libx11-6, libxext6, libxml2 (>= 2.6.27), libxrer
0-plugins-good, debconf (>= 0.5) | debconf-2.0, fonts-liberati
n | ttf-mscorefonts-installer
```

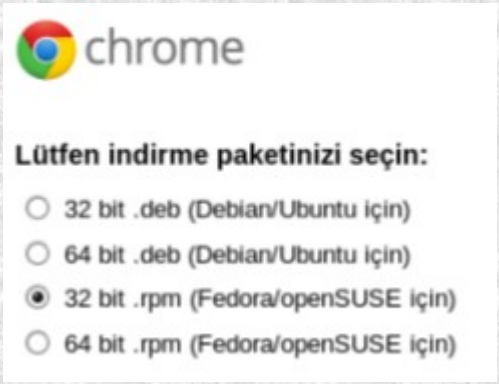
Bilgiler arasında **depends** diye bir bölüm göreceksiniz. Yani operanın çalışması için bunlara ihtiyacı varmış. Çalıştığına göre demek ki sistemimizde bu paketler yüklü. Henüz kurulum yapmadığımız bir deb paketiyle ilgili bilgi almak için de **dpkg --info <program.deb>** komutu kullanılır.

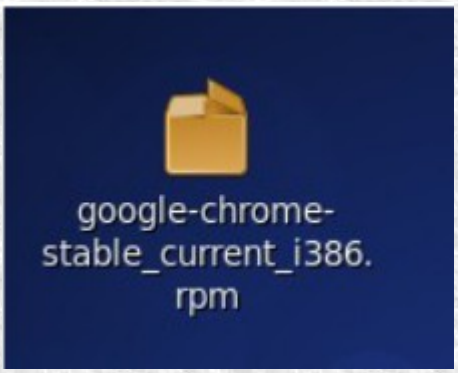
```
root@kemal:~# dpkg --info ghex_2.24.0-1_i386.deb
 new debian package, version 2.0.
 size 942020 bytes: control archive=3856 bytes.
    1024 bytes,    12 lines      control
    7915 bytes,    99 lines      md5sums
     496 bytes,    17 lines   *  postinst             #!/bin/sh
     627 bytes,    22 lines   *  postrm               #!/bin/sh
     180 bytes,     7 lines   *  prerm                #!/bin/sh
 Package: ghex
 Version: 2.24.0-1
 Architecture: i386
 Maintainer: Ubuntu MOTU Developers <ubuntu-motu@lists.ubuntu.com>
 Original-Maintainer: Sebastien Bacher <seb128@debian.org>
 Installed-Size: 2856
 Depends: libart-2.0-2 (>= 2.3.18), libatk1.0-0 (>= 1.20.0), libbo
15.0), libbonoboui2-0 (>= 2.15.1), libc6 (>= 2.4), libcairo2 (>= 1
config1 (>= 2.4.0), libfreetype6 (>= 2.3.5), libgconf2-4 (>= 2.13.
-0 (>= 2.12.0), libgnome2-0 (>= 2.17.3), libgnomecanvas2-0 (>= 2.1
```

Evet programları paketlerden manuel olarak nasıl kurabileceğimizi gördük. Şimdi de sistemde kurulu bir programı kaldırmak için ne yapacağımıza bakalım. Sistemde kurulu bir programı kaldırmanın en temel yolu **dpkg --purge remove <program_adı>** komutunu kullanmaktır. Ya da **dpkg --purge <program_adı>** komutu da kullanılabilir. Bu şekilde program tüm dosyalarıyla birlikte kaldırılacaktır.

```
root@kemal:~# dpkg --purge opera
(Reading database ... 241680 files and directories current
Removing opera ...
update-alternatives: using /usr/bin/iceweasel to provide
(x-www-browser) in auto mode
update-alternatives: using /usr/bin/iceweasel to provide
ser (gnome-www-browser) in auto mode
Purging configuration files for opera ...
Processing triggers for packagekit-backend-aptcc ...
Processing triggers for man-db ...
Processing triggers for hicolor-icon-theme ...
Processing triggers for desktop-file-utils ...
Processing triggers for gnome-menus ...
Processing triggers for shared-mime-info ...
Processing triggers for menu ...
root@kemal:~#
```

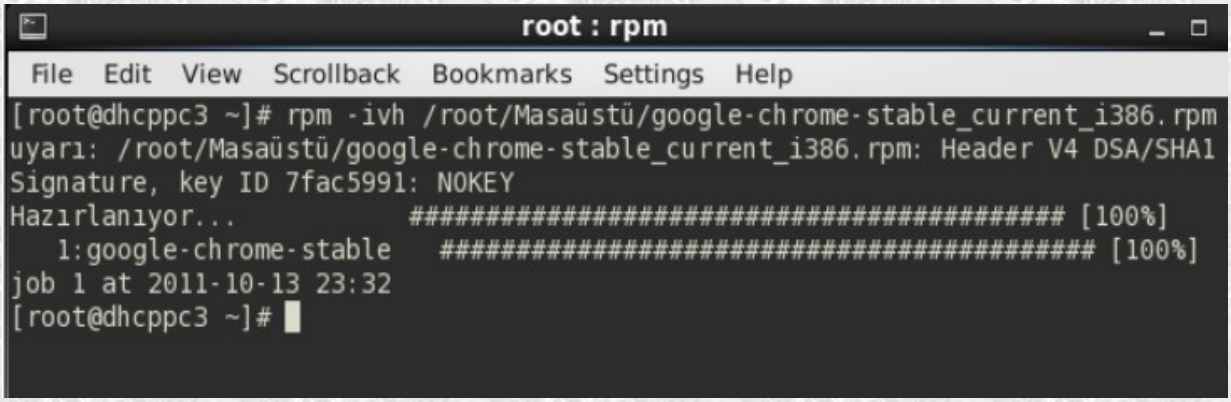
Şimdi de **rpm** paketini kullanan Linux dağıtımlarında bu işlemler nasıl yapılıyor ona bakalım. **Debian** sistemlerdeki manuel paket yönetici sistem olan **dpkg** yerine **RedHat** sistemlerde kullanılan sistem **rpm** dir. Yani kullandığı paketle aynı adı taşır. Bir örnekle program kurulumunun nasıl yapıldığını gösterelim. Örneğin sistemimize uygun Chrome browser paketini indirerek kurulumunu yapalım.





Gördüğünüz gibi indirdiğimiz paket **rpm** uzantılı. Bu şekilde rpm uzantılı paketleri kurmak için kullandığımız en temel komut

rpm -ivh <program.rpm> komutudur.



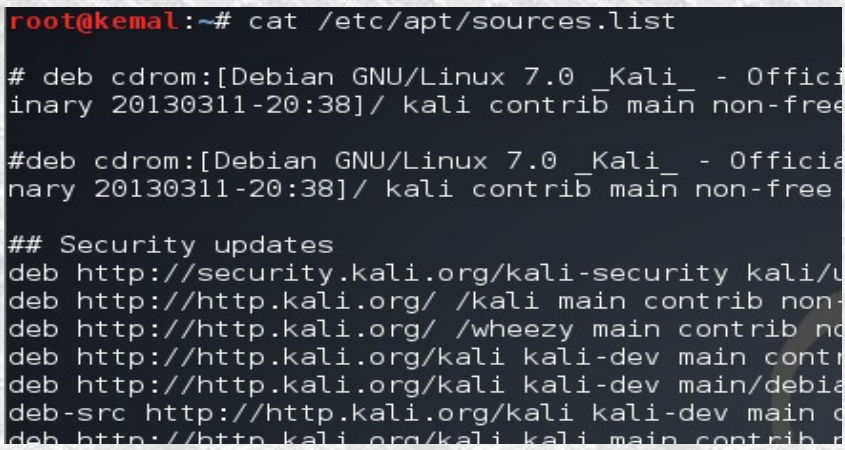
Programımız sisteme kuruldu. Sisteme manuel olarak kurmak istediğiniz rpm program paketlerini yine bu temel komutu kullanarak kurabilirsiniz. Buradaki **-ivh** şu anlama gelmektedir:

i : paketi kur
v : işlemleri ekranda göster
h : kurulum düzeyini göster

Kurulan paketi yani programı sistemden kaldırmak için de **rpm -ev <paket_adı>** komutu kullanılır.

Manuel olarak program kurulumu ve nasıl kaldırılacağını gördük. Şimdi bu işlemleri otomatik olarak nasıl yapabileceğimize bakalım. Kallanılan manuel paket yönetim sistemlerinin haricinde çok daha kolay program kurmamızı sağlayacak ve kurmak istediğimiz programları otomatik olarak tüm bağımlılıklarıyla beraber program deposundan indirerek sistemimize kurulumunu yapacak sistemler de vardır. Biz bu sistemlerden **Debian** tabanlı dağıtımların kullandığı **apt** ve **RedHat** tabanlı dağıtımların kullandığı **yum** paket yönetim sistemlerinden bahsedeğiz.

Bu paket yönetim sistemleri, **repository** ya da kısaca repo dediğimiz dağıtımın internet üzerindeki program depolarından istediğimiz programları sistemimize tüm bağımlılıklarıyla beraber otomatik olarak indirerek kurulumunu yapar. Hangi depolardan paketlerin indirileceği Debian sistemlerde **etc/apt/sourcest.list** altında bulunur.



Depolar **rpm** tabanlı dağıtımlarda da **/etc/yum.repos.d** altındaki dosyalarda bulunur.

Yukarıdaki ekran görüntüsüne bakacak olursak **deb http://** şeklinde başlayan adresler görüyoruz. İşte biz sisteme bir program kurmak istediğimizde **apt** paket yönetim sistemi bu adresteki depolardan programı indirerek sistemimize kuruyor. Depo adreslerinin yanındaki ifadeler de şu anlama geliyor:

main : debian ana paketleri
contrib : katkıcıların yardımıyla geliştirilen paketler
non-free : özgür olmayan paketler

Başta yazan **deb-src** ise depodaki paketlerin kaynak kodlarını ifade eder.

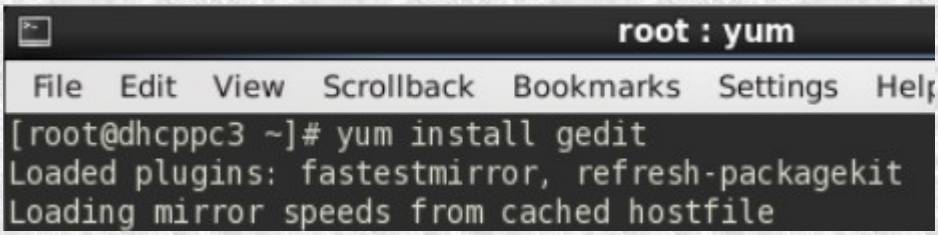
Bir program yüklemek için kullanacağımız temel komut **apt-get install <program_adı>** komutudur. **RedHat ve türevi** dağıtımlar için de temel olarak **yum install <program_adı>** komutu kullanılır. Bu komutları kullandıktan sonra eğer program depolarda varsa indirilip kurulumu yapılacaktır. Kurmak istediğimiz program ve depolarda olup olmadığıyla ilgili bilgi almak için de **apt-cache search <program_adı>** komutu kullanılabilir. Tabi güncellemelerle birlikte yeni programlar depoya eklenmiş olabilir.

```
root@kemal:~# apt-cache search bleachbit
bleachbit - delete unnecessary files from the system
root@kemal:~#
```

Programı kurmak için örneğin yukarıda arattığımız bleachbit için **apt-get install bleachbit** komutunu kullanırız.

```
root@kemal:~# apt-get install bleachbit
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  python-notify
The following NEW packages will be installed:
```

Eper **rpm** paketi kullanan dağıtımlarda program kurmak isterseniz, örneğin gedit text editör programını kurmak isteyelim. Komutumuz **yum install gedit** olmalıdır.



Her iki durumda da mantık aynı fakat dağıtımların kullandığı paket sistemi ve dolayısıyla paket yönetim sistemleri farklı.

Kurmak istediğimiz programla ilgili paket hakkında bilgi almak için **apt-cache show <program_adı>** komutunu kullanabiliriz.

```
root@kemal:~# apt-cache show bleachbit
Package: bleachbit
Version: 0.9.2-2
Installed-Size: 1744
Maintainer: Luca Falavigna <dktrkranz@debian.org>
Architecture: all
Depends: python (>= 2.6.6-7~), python-gtk2 (>= 2.14), menu
Recommends: python-notify
Description: delete unnecessary files from the system
Homepage: http://bleachbit.sourceforge.net
Description-md5: a958efd51e414316ebd3cb47958129ea
Tag: implemented-in::python, interface::x11, role::program,
 scope::application, uitoolkit::gtk, x11::application
Section: admin
Priority: optional
Filename: pool/main/b/bleachbit/bleachbit_0.9.2-2_all.deb
Size: 326192
MD5sum: a6d5689128b4d5cead0a49068d951eec
SHA1: b7355d6997352b7f08d42ef06f721c3e5134593d
SHA256: 2df0cc7520a9fc4df311ff01f9a13fa92c570bbee479cf47bb186cdc7b164e53
```

Sisteme kurulan bir programı kaldırmak için de **apt-get remove** komutu kullanılır. Komutu **apt-get --purge remove <program_adı>** şeklinde kullanırsanız programı konfigürasyon dosyalarıyla birlikte kaldırırsınız. Bu komut **apt-get purge <program_adı>** tarzında da kullanılabilir. Örneğin, daha önceden kurulumunu yaptığımız **gedit** metin editörü programını kaldırmak isterseniz aşağıdaki gibi bir komut kullanmalısınız.

```
root@kemal:~# apt-get --purge remove gedit
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer r
equired:
  gedit-common
Use 'apt-get autoremove' to remove it.
The following packages will be REMOVED:
  gedit*
0 upgraded, 0 newly installed, 1 to remove and 42 not upgraded.
After this operation, 2.805 kB disk space will be freed.
Do you want to continue [Y/n]?
```

Sistemi güncellemek için de **apt-get update** ve **apt-get upgrade** komutları kullanılabilir.

```
root@kemal:~# apt-get update
Get:1 http://security.kali.org kali/updates Release.gpg [836 B]
Get:2 http://http.kali.org /kali Release.gpg [836 B]
Get:3 http://http.kali.org kali-dev Release.gpg [836 B]
Get:4 http://security.kali.org kali/updates Release [11,0 kB]
Get:5 http://http.kali.org kali Release.gpg [836 B]
Get:6 http://http.kali.org /kali Release [21,1 kB]
Get:7 http://http.kali.org kali-dev Release [21,1 kB]
Get:8 http://security.kali.org kali/updates/main Sources [52,6 kB
Hit http://security.kali.org kali/updates/contrib Sources
Get:9 http://http.kali.org /kali/main i386 Packages [8.436 kB]
74% [9 Packages 6.242 kB/8.436 kB 74%] [Waiting for headers] [Wai
```

```
root@kemal:~# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages have been kept back:
  binwalk ewf-tools icedtea-7-jre-jamvm iceweasel libs
  libwbclient0 metasploit openjdk-7-jdk openjdk-7-jre
  openjdk-7-jre-headless openjdk-7-jre-lib samba samba
  samba-common-bin smbclient volatility wpscan
  xserver-xorg-input-all
The following packages will be upgraded:
  android-sdk apt apt-utils armitage bind9-host cewl c
```

Komutun ikisini birden konsolda kullanmak için daha önce anlattığımız gibi **apt-get update && apt-get upgrade** diyebilirsiniz. Dağıtımınızı yani versiyonunu yükseltmek istiyorsanız bu sefer **apt-get dist-upgrade** komutundan faydalanabilirsiniz.

```
root@kemal:~# apt-get dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Sistemde daha önceki programlardan kalan eski ve gereksiz artıkları temizlemek için kullanabileceğimiz iki komut vardır. Bunlar **apt-get autoremove** ve **apt-get autoclean** komutlarıdır. Böylece çöplerden kurtulmuş olursunuz.

```
root@kemal:~# apt-get autoremove
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  gedit-common
0 upgraded, 0 newly installed, 1 to remove and 42 not upgraded.
After this operation, 12,5 MB disk space will be freed.
Do you want to continue [Y/n]?
```

Sistemi güncellemek istediğinde bazen güncellenmediğini görürsünüz. Bunun nedeni sisteminize kurulan 3.parti programlar olabilir. Öncelikle bunlardan kurtulmak gerekir. Bunun için **apt-get -f install** komutunu kullanmanız lazım. Artık büyük ihtimalle sistemi güncelleyebileceksiniz.

RedHat tabanlı sistemlerde kurulan paketi sistemden kaldırmak için **yum remove <paket_adı>** komutu kullanılır. Sistem güncellemek için de **yum update** ve **yum upgrade** komutları kullanılır. Kaldırılan programla ilgili kalıntıları temizlemek için de **yum clean all** komutu işe yarayacaktır. Paket aramak ve bilgi almak için de **yum search <paket_adı>** komutu kullanılır diyelim ve diğer bir konu olan kaynak koddan program kurma konusuna geçelim.

Linux sistemlerde program kurmak için bir kaç yol var demiştik. O yollardan bir tanesi de ihtiyacımız olan programın **kaynak kodlarını** indirerek derlemek ve kurulumu bu şekilde yapmak. Bunun için kulanılan komut kalıbı şu şekildedir:

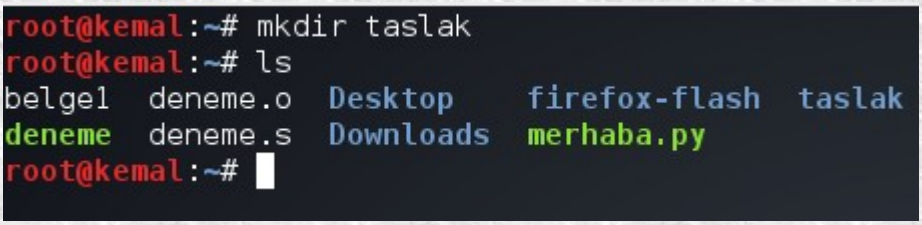
./configure
make
make install

Kaynak kodu indirilen paketin içinde bulunan readme, install vs. gibi yerler incelenmelidir. Genel kurulum kalıbı yukarıdaki gibi olmakla beraber bazen intall scriptleri konsoldan çalıştırılarak kurulum yapılabilmektedir.

Kurulacak programın kaynak kodlarının olduğu dosya içindeyken, örneğin dosyamız ev klasörümüzdeyse **cd <kaynak_kod_dosyası>** diyerek o dosya içindeyken sırasıyla **./configure**, **make** ve **make install** komutlarını kullandığımızda programımız kurulmuş olacaktır.

Dizin Oluşturma-Dizin Silme

Yeni dizin/klasör oluşturmak için **mkdir** komutunu kullanamız gerekir. Örneğin **mkdir taslak** komutuyla **taslak** isminde yeni bir dizin oluşturalım.



Eğer birden fazla dizin oluşturmak istiyorsak **mkdir <dizin1> <dizin2>..** komutu işimizi görür.

```
root@kemal:~# mkdir birinci ikinci
root@kemal:~# ls
belge1   deneme    deneme.s  Downloads      ikinci      taslak
birinci  deneme.o  Desktop   firefox-flash  merhaba.py
root@kemal:~#
```

İstediğimiz gibi **birinci** ve **ikinci** adında dizinlerimiz oluşturulmuş. İsterseniz bir dizin ve onun da alt dizinlerini tek komutla oluşturabiliriz. Bunun için örneğin **mkdir -p taslak2/kitap/kaynak** gibi bir komut kullanalım ve **taslak2** adında bir dizin ve onun alt dizini olarak **kitap** ve onun da alt dizini olan **kaynak** dizinlerini oluşturalım. Dikkat ederseniz bunu yapabilmek için **-p** kullanıyoruz. Eğer **p** parametresini kullanmayı unutursanız hata mesajı alırsınız.

```
root@kemal:~# mkdir -p taslak2/kitap/kaynak
root@kemal:~# ls
belge1  deneme.o  Desktop    firefox-flash  taslak
deneme  deneme.s  Downloads  merhaba.py     taslak2
root@kemal:~# cd taslak2
root@kemal:~/taslak2# ls
kitap
root@kemal:~/taslak2# cd kitap
root@kemal:~/taslak2/kitap# ls
kaynak
root@kemal:~/taslak2/kitap#
```

Dizinleri silmek için de **rm** komutu kullanılır. Eğer dizini alt dizinlerindeki dosyalarla birlikte silmek istiyorsanız **rm -r <dizin>** komutunu kullanmalısınız. Eğer silerken bana sorsun derseniz **rm -ir <dizin>**, sorgusuz sualsiz ne var ne yok silsin derseniz **rm -rf <dizin>** komutunu kullanmalısınız.

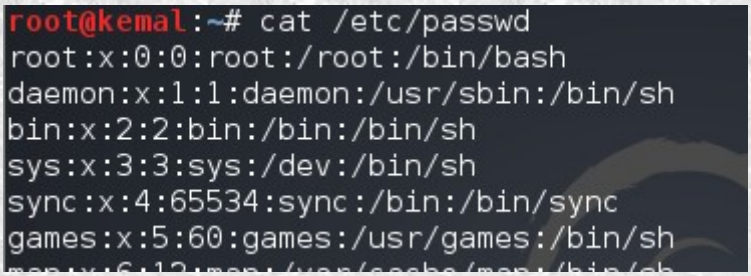
```
root@kemal:~# rm -ir taslak2
rm: `taslak2' dizininin içine inilsin mi?e
rm: normal boş dosya `taslak2/kjkıjh' silinsin mi?e
rm: `taslak2/kitap' dizininin içine inilsin mi?e
rm: normal boş dosya `taslak2/kitap/ghgfg' silinsin mi?e
rm: normal boş dosya `taslak2/kitap/ugvhvh' silinsin mi?e
rm: `taslak2/kitap/kaynak' dizininin içine inilsin mi?e
rm: normal boş dosya `taslak2/kitap/kaynak/jıjıjı' silinsin mi?e
rm: normal boş dosya `taslak2/kitap/kaynak/878787' silinsin mi?e
rm: dizin `taslak2/kitap/kaynak' silinsin mi?e
rm: dizin `taslak2/kitap' silinsin mi?e
rm: normal boş dosya `taslak2/bdh' silinsin mi?e
rm: dizin `taslak2' silinsin mi?e
root@kemal:~#
```

Birkaç farklı dizin silmek isterseniz dizinleri yan yana belirtebilirsiniz. Örneğin **rm -ir <dizin1> <dizin2>** gibi.

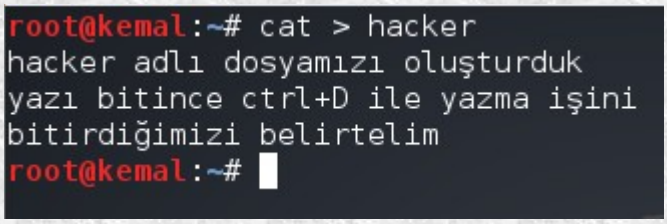
Dosya İşlemleri

Dosya oluşturma, kopyalama, taşıma, içeriğini okuma, komut çıktılarını dosyaya yönlendirme vs. işlemleri nasıl yapılır ya da kaç farklı şekilde yapılabilir onlara bakalım.

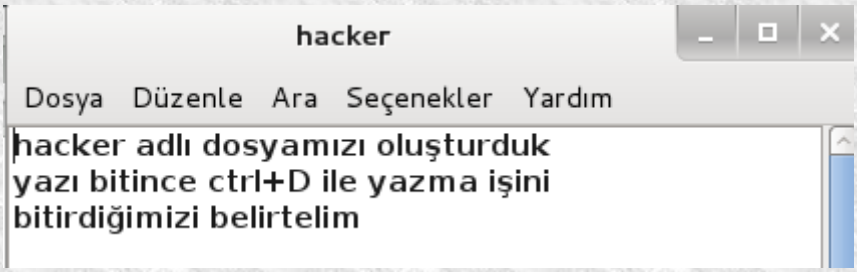
Bahsedilen bu işlemler için bir çok komut kullanılmaky tadır. Biz **cat** komutuyla başlayalım. Bu komutun işlevlerinden biri text dosyalarının içeriğini okumaktır. Aslında en fazla bu iş için kullanılır. Bir text dosyasının içeriğini konsoldan okumak istediğimizde genellikle bu komutu kullanırız. Şimdi örneğin **/etc** dizini altındaki **passwd** dosyasının içeriğini okuyalım. Bunun için kullanmamız gereken komut **cat /etc/passwd** komutudur.



Yeni bir text dosyası oluşturmak için de **cat** komutunu kullanabiliriz. Örneğin cat > hacker komutuyla hacker adında bir dosya oluşturalım. Aynı zamanda bu dosyanın içine istediğimiz ifadeyi de yazalım.



Yazacağımız şeyler bittikten sonra imleç satır başındayken **ctrl+D** tuşlarına basarak yazma işini tamamlayabiliriz ve komut satırı eski haline döner. Dosyamız istediğimiz gibi oluştu.



Metin dosyası oluşturmak için kullanabileceğimiz komutlardan bir tanesi de **touch** komutudur. Mesela touch test1 komutuyla test1 dosyasını oluşturalım.

```
root@kemal:~# touch test
root@kemal:~# ls
belge1  deneme.o  Desktop    firefox-flash  taslak
deneme  deneme.s  Downloads  merhaba.py     test
root@kemal:~#
```

Aslında bazı komutların çok farklı kullanım alanları olabilir. Bu komutları biz yerine göre kullanabiliriz. Bazen aynı işi yapan komutlardan bize daha kolay gelenini kullanmak isteriz. Örneğin bir text dosyasının içeriğini okumak için **more** komutunu da kulanabiliriz. Aslında bu komut genelde konsol komut çıktısı uzun olduğu zaman bu çıktıyı düzenli görüp rahat okummızı sağlamak için kullanılır. Şimdi **more /etc/passwd** komutuyla passwd dosyasını okuyalım.

```
root@kemal:~# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

Evet bir başka komutumuz da **echo** komutu. Bu komutu kullanarak da yeni bir dosya oluşturabiliriz ya da bir dosyanın içine yazabiliriz. Örneğin komutu **echo "of ulen of" > zeynep** tarzında kullanırsak ev dizinimizde **zeynep** dosyası oluşturulacak ve içine **of ulen of** yazılacaktır.

```
root@kemal:~# echo "of ulen of" > zeynep
root@kemal:~# cat zeynep
of ulen of
root@kemal:~#
```

Eğer zaten **zeynep** isminde bir dosya varsa ve içi **boşsa** yine bu şekilde **>** operatörünü kullanarak dosyanın içine istediğimizi yazdırabiliriz. Eğer dosya boş değilse bu sefer **>>** operatörünü kullanarak dosya içindeki ifadenin altına istediğimiz ifadeyi yazdırabiliriz.

```
root@kemal:~# echo "artık uyusana gözlerin kamyon tekeri gibi oldu" >> zeynep
root@kemal:~# cat zeynep
of ulen of
artık uyusana gözlerin kamyon tekeri gibi oldu
root@kemal:~#
```

Evet anlaşılabileceği gibi **>** ve **>>** operatörleriyle bir dosyaya yönlendirme yapabiliyoruz. Bunu komutların konsol çıktısını bir dosya içine yazdırmak için de kullanabiliriz.

```
root@kemal:~# find /sbin -perm 777 > rapor
root@kemal:~# cat rapor | more
/sbin/mkfs.ext4dev
/sbin/umount.hal
/sbin/mkfs.ext3
/sbin/mount.ntfs-3g
/sbin/mkfs.ntfs
/sbin/poweroff
```

Bir dosyanın son on satırını konsolda görüntülemek için de kullanabileceğimiz bir komut vardır. Bu komut **tail** komutudur. Örnek olarak **/etc/shadow** dosyasının son on satırını ekranda görelim. Bunun için **tail /etc/shadow** komutunu kullanmalıyız. Eğer son beş satırı görmek isterseniz **tail -n 5 /etc/shadow** diyebilirsiniz.

```
root@kemal:~# tail /etc/shadow
sshd:*:15775:0:99999:7:::
rtkit:*:15775:0:99999:7:::
snmp:*:15775:0:99999:7:::
stunnel4:!:15775:0:99999:7:::
statd:*:15775:0:99999:7:::
sslh:!:15775:0:99999:7:::
saned:*:15775:0:99999:7:::
Debian-gdm:*:15775:0:99999:7:::
privoxy:*:15871:0:99999:7:::
debian-tor:*:15871:0:99999:7:::
root@kemal:~#
```

Dosya içeriğini görüntülemek için kulanabileceğimiz komutlardan biri de **less** komutudur. Örneğin **less /etc/hostname** diyelim. Konsola geri dönmek için **q** tuşuna basmalıyız.

```
kemal
/etc/hostname (END)
```

Bir başka komutumuz da **sort** komutu. Bu komut belge çıktısını alfabetik tarzda konsolda gösterir. Eğer **sort -r <dosya>** şeklinde kullanırsak ters alfabetik olarak yansıtır.

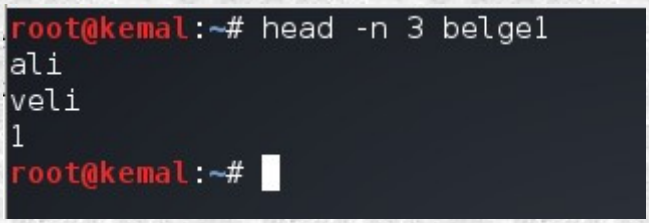
```
root@kemal:~# cat belge1
ali
veli
1
64646
zerrin
pis hacker
ahlaklı hacker
root@kemal:~# sort belge1
1
64646
ahlaklı hacker
ali
pis hacker
veli
zerrin
root@kemal:~# sort -r belge1
zerrin
veli
pis hacker
ali
ahlaklı hacker
64646
1
root@kemal:~#
```

Evet yukarıdaki görüntüden zaten her şey anlaşılıyor. Burada ek bilgi olarak şunu da söyleyelim. Bu komut sadece dosya işlemlerinde değil komut çıktılarının konsolda düzgün görünmeleri işinde de kullanılır.

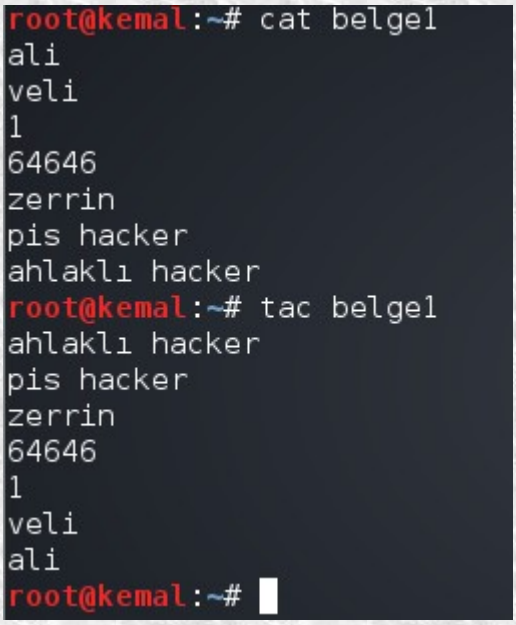
```
root@kemal:~# ls -l | sort
drwx------  6 root root 4096 Haz 18 16:12 Downloads
drwxr-xr-x 10 root root 4096 Ağu 29 22:16 Desktop
drwxr-xr-x  2 root root 4096 Ağu 29 19:16 firefox-flash
drwxr-xr-x  2 root root 4096 Ağu 29 19:16 taslak
-rw-r--r--  1 root root  233 Mar 17 16:13 deneme.s
-rw-r--r--  1 root root   52 Ağu 29 22:11 belge1
-rw-r--r--  1 root root  604 Mar 17 12:54 deneme.o
-rw-r--r--  1 root root  622 Ağu 29 21:42 rapor
-rwxr-xr-x  1 root root  112 Ağu 20 17:50 merhaba.py
-rwxr-xr-x  1 root root  627 Mar 17 12:54 deneme
toplam 40
root@kemal:~# ls -l | sort -r
toplam 40
-rwxr-xr-x  1 root root  627 Mar 17 12:54 deneme
-rwxr-xr-x  1 root root  112 Ağu 20 17:50 merhaba.py
-rw-r--r--  1 root root  622 Ağu 29 21:42 rapor
-rw-r--r--  1 root root  604 Mar 17 12:54 deneme.o
-rw-r--r--  1 root root   52 Ağu 29 22:11 belge1
-rw-r--r--  1 root root  233 Mar 17 16:13 deneme.s
drwxr-xr-x  2 root root 4096 Ağu 29 19:16 taslak
drwxr-xr-x  2 root root 4096 Ağu 29 19:16 firefox-flash
drwxr-xr-x 10 root root 4096 Ağu 29 22:16 Desktop
drwx------  6 root root 4096 Haz 18 16:12 Downloads
root@kemal:~#
```

Bir dosya içindeki satır, karakter ya da kelime sayılarını merak ediyorsanız merakınızı gidermek için **wc** (wordcount) komutunu kullanabilirsiniz. **-l** ile satır, **-c** ile karakter ve **-w** parametresi ile de kelime sayısı öğrenilebilir. Eğer sadece **wc** şeklinde parametresiz kullanırsanız hepsini gösterir.

Yukarıda **tail** komutunun ne işe yaradığını gördük. Şimdi buna benzer olan **head** komutuna bakalım. **tail** komutu ile text dosyanın ya da komut çıktısının som on satırını görüntülerken **head** tam tersini yapıyor yani ilk on satırını görüntülüyor. Tabi isterseniz **-n** ile kaç satır görmek istediğinizi belirtebilirsiniz. Mesela ilk beş satırın görüntülenmesini istiyorsanız **head -n 5 <dosya/komut>** tarzında bir komut kullanabilirsiniz.



Yine yukarıda anlattığımız **cat** komutuna benzer bir de **tac** komutumuz var. Bakalım bu komutumuz ne yapıyor?



İşimize yarayacak faydalı komutlardan bir tanesi de **nl** komutudur. Bu komut özellikle programlamayla uğraşanların daha çok işine yarayacaktır. Bir programın kaynak kodlarını

inceliyorsunuz fakat satırlar numaralandırılmadığı için biraz zorlanıyorsunuz. İşte **nl** komutu her satırın başına artırarak sayı ekler.

```
root@kemal:~# cat belge1
ali
veli
1
64646
zerrin
pis hacker
ahlaklı hacker
root@kemal:~# nl belge1
     1  ali
     2  veli
     3  1
     4  64646
     5  zerrin
     6  pis hacker
     7  ahlaklı hacker
root@kemal:~#
```

Komut çıktısını ya da belge içeriğini sayfalara bölmek için **pr** komutunu kullanabiliriz.

```
root@kemal:~# pr rapor


2013-08-29 21:42                    rapor                    Sayfa 1


/sbin/mkfs.ext4dev
/sbin/umount.hal
/sbin/mkfs.ext3
/sbin/mount.ntfs-3g
/sbin/mkfs.ntfs
/sbin/poweroff
```

Siz de örneğin **ls -la | pr** komutunu kullanıp çıktıyı inceleyebilirsiniz.

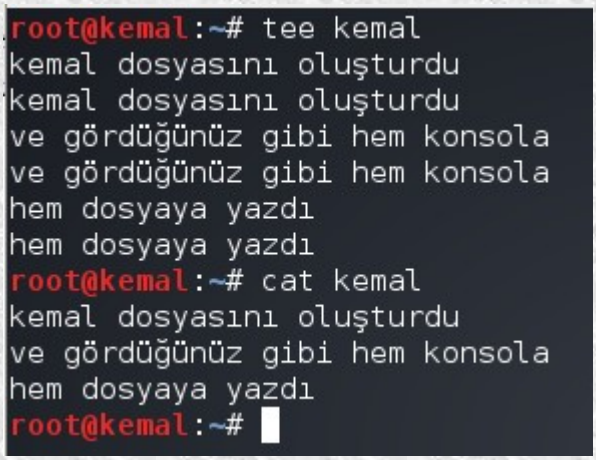
Eğer dosya içindeki ifadeleri **16 lık** sayı sisteminde görüntülemek isterseniz bunun için **od -x <dosya>** komutunu kullanabilirsiniz. Eğer **-x** kullanmazsanız **8 lik** sistemde çıktı alırsınız.

```
root@kemal:~# od -x /etc/hostname
0000000 656b 616d 0a6c
0000006
root@kemal:~# echo "kemal" | xxd
0000000: 6b65 6d61 6c0a                           kemal.
```

Bildiğiniz gibi **/etc/hostname** dosyasında hostumuzun yani pc mizin adı bulunuyor. Benim hostun adı kemal olduğu için çıktıda bu ifadeyi 16 lık sistemde gösterdi. Ben de sağlamasını göstermek

için echo komutundan faydalandım. Bu komut yazılan ifadeyi konsolda görüntüler. Ben **echo "<string>"** yani herhangi bir ifade yazdığımda bunu aynen ekranda gösterir. Ben **echo "kemal" | xxd** komutuyla kemal ismini ekrana 16 lık sistemde yansıt demiş oldum.

Bir de **tee** komutuna bakalım. Hatırlarsanız daha önce **cat** komutuyla dosya oluşturup içine istediğimiz ifadeleri yazmayı anlatmıştık. Hatta **cat > hacker** diye bir örnek vermiştik. Şimdi kullanacağımız **tee** komutu biraz daha farklı olarak yazılan ifadeyi hem **standart çıktıya** (**yani konsola**) yansıtır hem de oluşturulan dosyaya yazar. Yazma işlemini bitirdiğimize **ctrl+D** ile çıkabiliriz.



Bir başka komutumuz olan **paste** komutunu incelemeye geldi sıra. Bu komut iki ya da daha fazla dosyayı alarak, satırları ardışık olarak birbirine ekler ve buna göre olan çıktıyı gösterir. Aşağıdaki örnekle kolayca anlayacaksınız ne demek istediğimi.

Daha başka komutları da inceleyeceğiz fakat burada küçük bir hatırlatma yapalım ondan sonra devam edelim. Bazı komutları kullanırken **| operatörünü** de kullandığımızı gördünüz. Bu işareti

pipe (borulama) denen işlemi yaparken kullanırız. Borulama bir komut çıktısını alıp diğer komuta girdi yapmak demektir. Örneğin en basit haliyle **ls -la | wc -l** gibi bir komutu deneyebilirsiniz. Tabi bu kullanışlı komut çok farklı şekillerde kullanılabilir.

```
root@kemal:~# ls -la | wc -l
72
root@kemal:~#
```

Bir hatırlatma daha yapalım. Örneğin **echo** komutunda tek tırnak ya da çift tırnak içindeki string konsola yansıtılıyordu. Fakat **ters tırnak** (`) arasındaki ifade string olarak algılanmaz.

```
root@kemal:~# echo kemal
kemal
root@kemal:~# echo 'kemal'
kemal
root@kemal:~# echo "kemal"
kemal
root@kemal:~# echo `kemal`
bash: kemal: komut yok

root@kemal:~#
```

Örnekten de görüldüğü gibi demek ki ters tırnak arasındaki ifade komut olarak algılanıyor. Hem bununla ilgili hem de pipeleme ile ilgili Kim Korkar Linux'tan? isimli kitapta hoş bir örnek vardı. Aynen şöyle:

```
root@kemal:~# echo sistemde `who | wc -l` kullanıcı var
sistemde 2 kullanıcı var
root@kemal:~#
```

Güzel bir örnek. Evet bu hatırlatmaları yaptıktan sonra başka bir komutu incelemeye geçebiliriz. Sıradaki komutumuz **grep** komutu. Bu komut da işleri bir hayli kolaylaştıran faydalı komutlardan biridir. Dosya ya da komut çıktısında bir karakter dizisi (string) aradığımız zaman **grep** komutunu kulanırız.

```
root@kemal:~# cat belge2
program:
-linux özet
-genel kavramlar
-neden Linux?
root@kemal:~# grep linux belge2
-linux özet
root@kemal:~# cat belge2 | grep linux
-linux özet
root@kemal:~#
```

Yukarıdaki örnekte txt dosyası içinde linux geçen yerleri aradık. Fakat büyük harfle başlayan Linux'u göremedik. Büyük-küçük harf ayrımı yapılmaması için grep -i komutu kullanılmalıdır.

```
root@kemal:~# cat belge2
program:
-linux özet
-genel kavramlar
-neden Linux?
root@kemal:~# grep linux belge2
-linux özet
root@kemal:~# cat belge2 | grep linux
-linux özet
root@kemal:~#
```

Aradığımız ifadenin geçtiği satır numaralarını da görmek istersek **grep -ni** şeklinde komut kullanmalıyız.

```
root@kemal:~# cat belge2 | grep -ni linux
2:-linux özet
4:-neden Linux?
root@kemal:~#
```

Örneğin bulunduğumuz dizindeki tüm dosyalarda her hangi bir yerde geçen bir stringi aramak için **grep <string> * | more** gibi bir komut kullanabiliriz.

```
root@kemal:~# grep -r root * | more
Desktop/.~lock.Linux Komut Satırı.odt
/root/.libreoffice/3;
İkilik dosya Desktop/SQL_inj.odt eşle
İkilik dosya Desktop/kali tor kurulum
İkilik dosya Desktop/notlara eklenece
t eşleşir
İkilik dosya Desktop/notlara eklenece
leşir
```

Bu komutla ilgili küçük bir örnek daha verelim ve yeri geldikçe bu komuttan faydalanacağımızı belirtelim.

```
root@kemal:~# cat /etc/shadow | grep root
root:$6$RHxkvp3f$uMNlDynuT.ZuPV5zUb4TU2MOu5E1mQMPzLPz
/kiumtKLcHvhrNp8K1Lx60.:15777:0:99999:7:::
root@kemal:~#
```

Devam edelim. Şimdi de yine kullanışlı bir komut olan **cut** komutundan bahsedelim. Bu komutla da bazı satırların bazı alanlarını listeleyebiliriz. Yine bir örnek üzerinde inceleyelim.

```
root@kemal:~# cat /etc/passwd | head -n 5
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
root@kemal:~#
```

İlk önce örneğin passwd dosyasının ilk beş satırını görelim. Sonra mesela : ile ayrılmış alanlardan sadece birinci ve beşinci alandaki ifadeleri görelim. Bunun için **cut -d: -f 1,5 /etc/passwd | head -n 5** komutunu kullanmamız gerekir.

```
root@kemal:~# cat /etc/passwd | head -n 5
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
root@kemal:~# cut -d: -f 1,5 /etc/passwd | head -n 5
root:root
daemon:daemon
bin:bin
sys:sys
sync:sync
root@kemal:~#
```

Komutu incelerseniz **-d** ile alanların ayrıldığı yerleri, **-f** ile de kaçıncı alanları görmek istediğimizi belirttik.

Komutu farklı bir formatta da kullanabilirsiniz. Örneğin bu sefer **tail -n 6 /etc/passwd | cut -d : -f 3** komutunun çıktısı aşağıdaki gibi olacaktır.

```
root@kemal:~# tail -n 6 /etc/passwd | cut -d : -f 3
117
118
119
120
121
122
root@kemal:~#
```

İsterseniz **-c** ile gösterilecek karakter sayılarını da belirtebilirsiniz.

```
root@kemal:~# cat belge2
program:
-linux özet
-genel kavramlar
-neden Linux?

root@kemal:~# cut -c 1-5 belge2
progr
-linu
-gene
-nede

root@kemal:~#
```

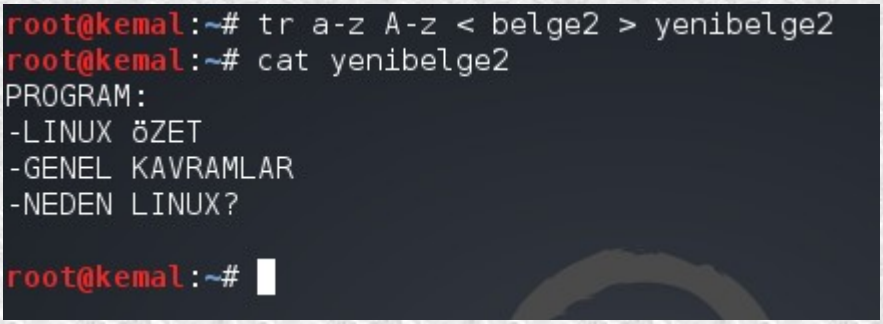
Şimdi de **tr** komutuna bakalım. Bu komut dosyalar içindeki karakterleri istediğimiz gibi değiştirmemizi sağlar. En temel kullanımı aşağıda görüldüğü gibidir.

```
root@kemal:~# cat belge2
program:
-linux özet
-genel kavramlar
-neden Linux?

root@kemal:~# cat belge2 |tr a-zA-Z A-Za-z
PROGRAM:
-LINUX öZET
-GENEL KAVRAMLAR
-NEDEN lINUX?

root@kemal:~#
```

Küçük harfleri büyük harflerle değiştirmiş olduk. Bunu daha kolay bir komutla yapabilirdik.

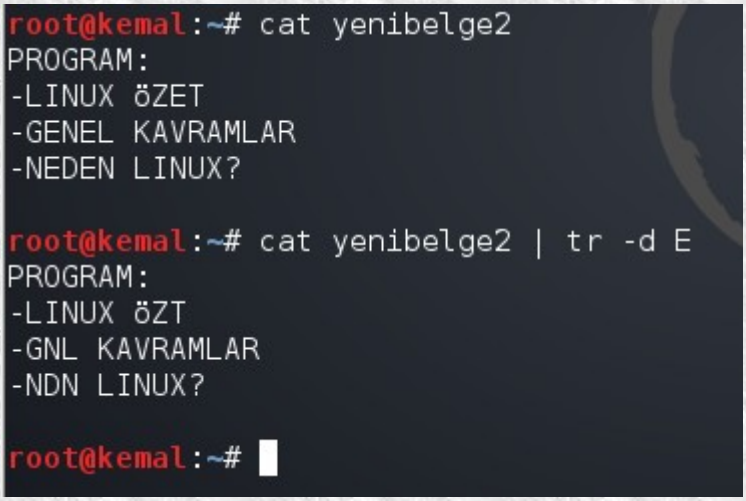
```
root@kemal:~# cat belge2
program:
-linux özet
-genel kavramlar
-neden Linux?

root@kemal:~# cat belge2 |tr a-z A-Z
PROGRAM:
-LINUX öZET
-GENEL KAVRAMLAR
-NEDEN LINUX?

root@kemal:~#
```

Daha önce gördüğümüz yönlendirme operatörlerine bir tanesini daha ekleyeli. Hatırlarsanız **>** ve **>>** operatörlerini yönlendirme işlemlerinde kullanıyorduk. Bunların haricinde bir de < operatörü var. Bu operatör diğerlerinden faklı olarak **dosyayı girdi** alıyor. Yani belirtilen dosyanın içeriğini alarak bu içeriğe göre değişiklik yapıyor. Şimdi hem **tr** komutunu hem de **>** ve **<** operatörlerini beraber kullanalım.



Yukarıda görmüş olduğunuz komutla şunu demek istedik: **belge2** dosyasını al, bu dosyadaki küçük harfleri büyük harf olarak değiştir ve **yenibelge2** adlı bir dosya oluştur ve bu dosyanın içine yaz.

Eğer **-d** parametresini kullanırsanız belirtilen karakteri yok sayacaktır. Örneğin **cat yenibelge2 | tr -d E** kullanımına bakalım.



Bu konunun devamı olarak **sed** ve **awk** kullanımından behsedeceğim fakat bu iki önemli konudan önce bahsetmemiz gereken bir kaç komut var. Bu komutlar **file**, **stat**, **find**, **locate**, **which** ve **whereis** komutları. Bunları da arada kaynamadan anlattıktan sonra **sed** ve **awk** kullanımına geçebiliriz.

Evet **file** komutuyla başlayalım. Bu komutla sorguladığımız dosyanın ne türde bir dosya olduğu hakkında bilgi alabiliriz. Bunun için **file <dosya>** komutunu kullanmamız yeterlidir. Şimdi daha önce işlemiş olduğumuz **ls** komutunu **ls -F** şeklinde F parametresiyle kullanarak istediğimiz bir dizin altındaki dosyaları görüntüleyelim.

```
root@kemal:~# ls -F /bin
bash*           fgrep*       nano*              setfacl*
bunzip2*        findmnt*     nc@                setupcon*
busybox*        fuser*       nc.traditional*    sh@
bzcat*          fusermount*  netcat@            sh.distrib@
bzcmp@          getfacl*     netstat*           sleep*
bzdiff*         grep*        nisdomainname*     ss*
bzegrep@        gunzip*      ntfs-3g*           stty*
bzexe*          gzexe*       ntfs-3g.probe*     su*
bzfgrep@        gzip*        ntfs-3g.secaudit*  sync*
bzgrep*         hostname*    ntfs-3g.usermap*   tailf*
bzip2*          ip*          ntfscat*           tar*
bzip2recover*   kill*        ntfsck*            tempfile*
bzless@         kmod*        ntfscluster*       touch*
bzmore*         less*        ntfscmp*           true*
```

Listede yanında * işareti olanlar çalıştırılabilir dosyalar, @ işareti olanlar da sembolik linkleri göstermektedir. Bu dosyalardan iki tanesine bakalım.

```
root@kemal:~# file /bin/dd
/bin/dd: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
inked (uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=0x58
94d398b12eea86c1c2d7c938, stripped
root@kemal:~# file /bin/lsmod
/bin/lsmod: symbolic link to `kmod'
root@kemal:~#
```

```
root@kemal:/usr/share/metasploit-framework# file msfrop
msfrop: Ruby script, ASCII text
root@kemal:/usr/share/metasploit-framework#
```

Çalıştırılabilir script dosyalarının da bilgisini verdiğini görüyoruz.

Diğer komutumuz olan **stat** komutu ise dosyanın/dizinin durumu hakkında bilgi almamızı sağlar. **stat <file>** şeklinde kullanılır.

```
root@kemal:/usr/share/metasploit-framework# stat modules
  File: `modules'
  Size: 4096            Blocks: 8          IO Block: 4096   dizin
Device: 801h/2049d      Inode: 1206760     Links: 8
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2013-08-30 17:31:11.270352174 +0300
Modify: 2013-06-11 17:12:04.911629203 +0300
Change: 2013-06-11 17:12:04.911629203 +0300
 Birth: -
root@kemal:/usr/share/metasploit-framework# cd
```

```
root@kemal:~# stat belge1
  File: `belge1'
  Size: 47              Blocks: 8          IO Block: 4096   normal dosya
Device: 801h/2049d      Inode: 1572394     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2013-08-31 14:31:44.630883411 +0300
Modify: 2013-08-31 14:15:54.854173722 +0300
Change: 2013-08-31 14:15:54.854173722 +0300
 Birth: -
root@kemal:~#
```

Diğer komutumuz **find** ile de istediğimiz kriterlerde arama yapabiliriz. Komutu **find <dizin> <parametre> <aranılan dosya/ifade>** şeklinde kullanıyoruz.

```
root@kemal:~# find /home -name zeynep.txt
/home/zeynep.txt
/home/python for hackers/zeynep.txt
root@kemal:~#
```

Yukarıdaki görüntüden de anlaşıldığı gibi **/home** dizininin altında **zeynep.txt** adlı dosya nerelerde varmış onu araştırdık. Biraz daha farklı bir şey yapalım ve **/sbin** altında bulunan dosyalardan hangilerinin izinleri **777** modunda onu arayalım.

```
root@kemal:~# find /sbin -perm 777 | more
/sbin/mkfs.ext4dev
/sbin/umount.hal
/sbin/mkfs.ext3
/sbin/mount.ntfs-3g
/sbin/mkfs.ntfs
/sbin/poweroff
/sbin/ip6tables-save
/sbin/insmod
/sbin/ip6tables-restore
/sbin/fsck.vfat
/sbin/iptables
```

Çok farklı parametrelerle kullanılabilecek **find** komutuna ilginç bir örnek verelim. Komutumuzu **-exec** ile beraber kullanacağız. İlk önce ev dizinimizde **zeynep.txt** dosyaları nerelerde bulunuyor ona bakalım.

```
root@kemal:~# find -name zeynep.txt
./zeynep.txt
./taslak/zeynep.txt
./Downloads/zeynep.txt
root@kemal:~#
```

Şimdi **find -name zeynep.txt -exec rm {} \;** kullanalım ve bu komutun ne yaptığına bakalım.

```
root@kemal:~# find -name zeynep.txt -exec rm {} \;
root@kemal:~# find -name zeynep.txt
root@kemal:~#
```

Evet bu komutla ev dizininmizdeki zeynep.txt dosyalarını bulmasını ve silmesini istedik. Komutu incelersek:

{} : Bulunan dosya ve dizinler bu parantez arasına parametre olarak yerleştirilecek ve **-exec** den hemen sonra belirtilen program buna göre çalıştırılacaktır. Yani dizinlerde bulunan her **zeynep.txt** dosyası için **rm** komutu çalıştırılarak silme işlemi yapılacaktır.

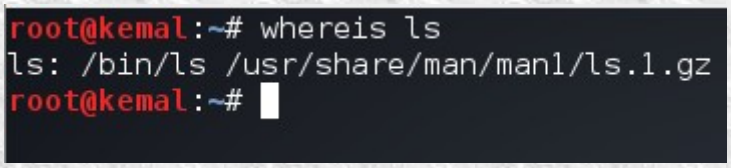
Eğer **find** komutu **-name "zey*"** ile beraber kullanılırsa bu sefer belirtilen dizin altındaki yerlerde **zey** ile başlayan dosyalar aranacaktır. Örneğin **apache** için deneyelim.

```
root@kemal:~# find /etc -name "apac*"
/etc/default/apache2
/etc/apache2
/etc/apache2/apache2.conf
/etc/logrotate.d/apache2
/etc/init.d/apache2
/etc/php5/apache2
/etc/cron.daily/apache2
/etc/bash_completion.d/apache2.2-common
root@kemal:~#
```

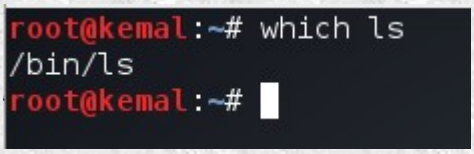
Şimdi de yine dosya aramaya yarayan **locate** komutunu görelim. Bu komutun **find** den farkı, devamlı güncellenen bir veritabanından arama yapmasıdır. Yani **find** bizim belirttiğimiz dizin altında arama yaparken, **locate** ile tüm olası lokasyonlarda arama yapılır. Aşağıdaki örneğe bakabilirsiniz.

```
root@kemal:~# locate mysql | more
/etc/mysql
/etc/init.d/mysql
/etc/logcheck/ignore.d.paranoid/mysql-server-5_5
/etc/logcheck/ignore.d.server/mysql-server-5_5
/etc/logcheck/ignore.d.workstation/mysql-server-5_5
/etc/logrotate.d/mysql-server
/etc/mysql/conf.d
/etc/mysql/debian-start
/etc/mysql/debian.cnf
/etc/mysql/my.cnf
/etc/mysql/conf.d/.keepme
/etc/mysql/conf.d/mysqld_safe_syslog.cnf
/etc/php5/conf.d/20-mysql.ini
/etc/php5/conf.d/20-mysqli.ini
```

Bir başka arama komutu da **whereis** komutudur. Bu komutla programların çalıştırılabilir dosyasının nerede olduğu ve man yardım sayfası yeri gibi bilgiler alabiliriz.



Son arama komutumuz olan **which** ise bir komuta/programa ilişkin çalıştırılabilir dosyanın hangi dizinde olduğunu gösterir. Yani daha önce de görmüş olduğumuz **PATH** ortam değişkenindeki yerlerin hangisinde olduğunu gösterir.



Artık iki önemli konu olan **sed** ve **awk** kullanımı konularına geçebiliriz. Bu iki önemli konuya başlamadan önce **düzgün deyimler** (**Regular Expressions**) kavramından bahsetmeliyiz.

Düzgün Deyimler

Düzenli ifadeler, programlamada ve bazı Linux shell işlemlerinde kullanılır. Bizlere bir çok alanda kolaylıklar sağlar. Verilerden ihtiyacımız olan bilgilerin çekilmesi, kullanıcı girdisinin denetlenmesi vs. gibi işlerde sıklıkla kullanılır. Örneğin verilen bir **şablon** (**pattern**) ifadenin aranması basit bir işlemken bu işlemi karmaşık ve büyük dosyalardan çekerken tam olarak istediğimiz şablonun aramasını yapmak için düzenli ifadeler kullanmamız gerekecektir. Sadece arama işlemlerinde değil, aranılan ifadelere uygulanacak işlemler için de düzenli ifadelerden faydalanılır.

Biz düzgün ifadeleri **sed** ve **awk** ile ilgili örneklerde de kullanacağız. Buna uygun düzgün ifadeler nelerdir şimdi onlardan bahsedelim. Uygun dedim çünkü örneğin **Perl** ve **Python** script dillerinin de kullandığı kendine uygun düzgün ifadeler bulunmaktadır. İşte biz de sed ve awk kullanırken işimize yarayacak Linux işletim sisteminde kullanılan düzenli ifadelerden bahsedeceğiz.

[] : Köşeli parantezin içindeki karakterlerin istenilen şablonda kullanılacağını belirtir. Aşağıda küçük bir örnek yer almaktadır.

```
root@kemal:~# cat belge1
kemal
ali
veli
maria
falanko
filanko
feşmekan
root@kemal:~# grep f[işn] belge1
filanko
root@kemal:~# grep f[iaşn] belge1
falanko
filanko
root@kemal:~#
```

Bizim verdiğimiz **f** harfinin yanına köşeli patantez içindeki karakterler sırasıyla ekleniyor sonra **fi** ve **fa** ile başlayan kelimeler bulunuyor. Araya aralık (**-**) işareti de konularak kullanılabilir.

```
root@kemal:~# grep f[a-i] belge1
falanko
filanko
feşmekan
root@kemal:~#
```

Nokta (**.**) : Nokta ile gösterilen yere herhangi bir karakterin gelebileceği belirtilmiş olur.

```
root@kemal:~# cat test
istanbul
izmir
ankara
adana
afyon
yozgat
konya
kayseri
bolu
bilecik
denizli
diyarbakır
root@kemal:~# grep a.a test
ankara
adana
root@kemal:~#
```

yıldız (*) : Her hangi sayıdaki her hangi bir karaktere karşılık gelir.

^ : Satır başına karşılık gelir.

```
root@kemal:~# cat /etc/shadow | grep ^daemon
daemon:*:15775:0:99999:7:::
root@kemal:~#
```

$: Satır sonunu ifade eder.

```
root@kemal:~# cat /etc/passwd | grep bash$
root:x:0:0:root:/root:/bin/bash
postgres:x:112:123:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
root@kemal:~#
```

[^..] : Kümenin içindeki karekterlerin haricindeki her hangi bir karaktere karşılık gelir.

```
root@kemal:~# cat test
istanbul
izmir
ankara
bolu
bilecik
denizli
diyarbakır
23377
099888
34355
root@kemal:~# cat test | grep [^0-9]
istanbul
izmir
ankara
bolu
bilecik
denizli
diyarbakır
root@kemal:~#
```

\{n\} : Kendisinden önceki karakterin n kez tekrar edildiğini gösterir.

\{n,m\} : Kendisinden önceki karakterin en az n kez en fazla m kez olduğunu gösterir.

\{n,\} : Kendisinden önceki karakterin en az n kez olduğunu gösterir.

\+ : Kendisinden önceki karakterin bir ya da daha fazla olduğunu gösterir.

\? : Kendisinden önceki karakterin sıfır ya da bir kez bulunduğunu gösterir.
\| : Kendisinden bir önceki veya bir sonraki karaktere karşılık gelir.
\(..\) : Grup olarak düzenli deyimleri tanımlar.
**** : Özel karakterlerin normal karakterler olarak algılanmasını sağlar.

Örnekler:
a\?b : b,ab,..
a[^a-z] : a0, a1, aZ,..
^a : satrı başında a karakteriyle başlayan sözcükler
^zzz veya **^z\{3\}** : satır başında 3 adet z karakteri bulunan sözükler
\{3\|5\}\+ : içinde 3 veya 5 sayılarından en az bir kez geçen sözcükler
\<Zey : Zey ile başlayan satırları bulur
yi\> : yi ile biten satırları bulur

```
root@kemal:~# grep "\<Ze" zeyno
Zeynocan bak güzel kitaP
root@kemal:~# grep "yi\>" zeyno
Kitap okumak eyidir eyi
root@kemal:~#
```

^Z.*P$: satır başlarında Z ile başlayıp satır sonunda da P ile biten sözcükler.

```
root@kemal:~# cat zeyno
Zeynocan bak güzel kitaP
Kitap okumak eyidir eyi
Yerli malı yurdun malı herkeşler kitap okumalı
root@kemal:~# grep "^Z.*P$" zeyno
Zeynocan bak güzel kitaP
root@kemal:~#
```

Evet, düzenli ifadeler hakkında biraz bilgimiz olduğuna göre artık **sed** ve **awk** kullanımına geçebiliriz. İlk önce sed konusuna bakalım.

sed (stream editor)

Sed programı, metin belgeleri üzerinde çeşitli komutlar kullanarak değişiklikler yapmamızı sağlar. Konsoldan direk komutları girerek ya da bir dosyadan komutların alınmasını isteyerek işlemler yapabiliriz. Kullanılan operatörler/parametreler şunlardır:

-n : sadece belirtilen satırlara uygulama yap
-e : bir sonraki komut bir düzenleme komutu
-f : bir sonraki bilgi bir dosya adı
p : print

```
root@kemal:~# cat zeyno
Zeynocan bak güzel kitaP
Kitap okumak eyidir eyi
Yerli malı yurdun malı herkeşler kitap okumalı
Zeytin de yurdun malı
root@kemal:~# sed -n "s/Zeytin/karpuz/p" zeyno
karpuz de yurdun malı
root@kemal:~#
```

İstenilen ifadeyi değiştirip **p** ile ekrana yazdırdık. Tabi ekrana değiştirdiğimiz ifadenin geçtiği satır yazdırıldı. (**-n** ile birlikte kullanılmalı)

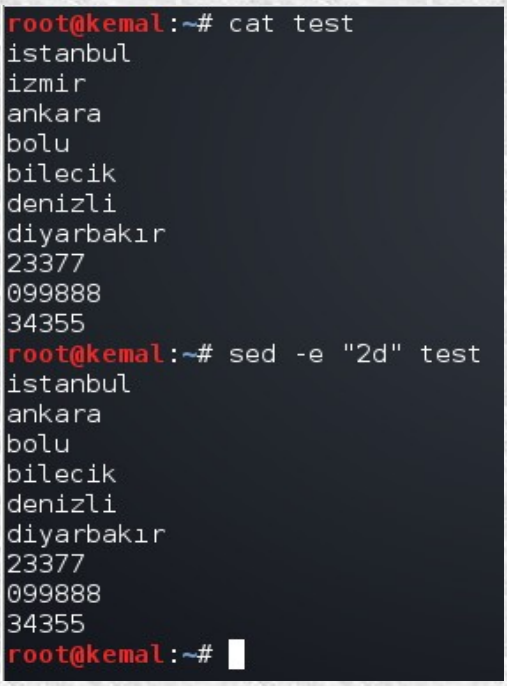
d : delete anlamına gelir

s/değişecek ifade/yerine gelecek ifade/g : bu kullanımla bir text içindeki kelimeler değiştirilebilir. Sondaki g eklenmezse dosyanın tamamı yerine her satırdaki ilk kalıp baz alınır.

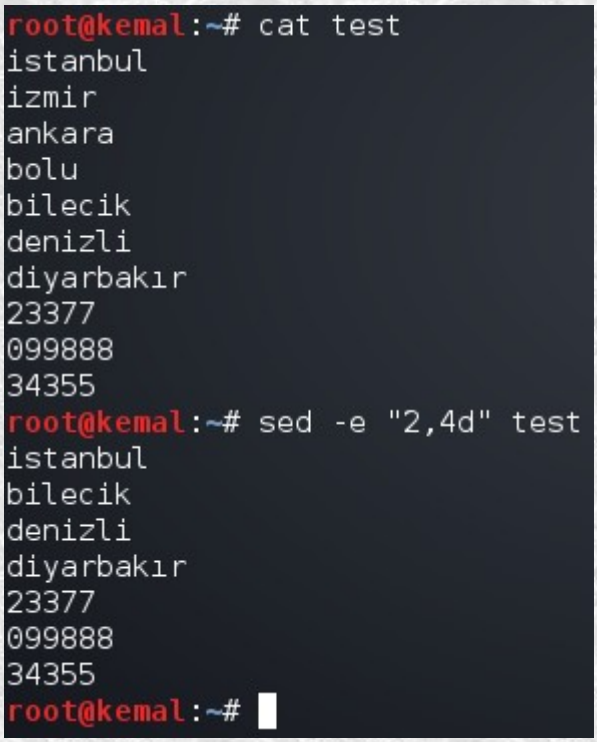
```
root@kemal:~# echo "heykır" | sed "s/heykır/fatihin fedaisi kara heykır/"
fatihin fedaisi kara heykır
root@kemal:~#
```

Tabi **sed** kullanırken düzenli ifadelerden de faydalanacağız. Şimdi bir kaç örnek ifadeyi inceleyelim:
5d : 5. satırı sil
/^$/d : tüm boş satırları sil
s/*$// : her satırın sonundaki tüm boşlukları sil
/zey/d : zey ifadesi olan tüm satırları siler
s/00*/0/g : ardışık sıfırların yerine tek sıfır yaz
sed -e "s/apple/elma/g" -e "s/prg/program/g" : aynı anda birden çok değiştirme yapmak için -e kullanılabilir.
s/\./&\/g : noktadan sonra gelen karakterleri yeni satıra kaydır.
10,20d : 10. satırla 20. satır arasını sil (10. ve 20. satır dahil)
/**$/d** : satırın tamamı ***** olan satırları sil
sed -e "s/[^|]*$//" <dosya> : belirtilen dosyadaki tab ve boşluk karakterlerini kaldır

Şimdi de bazı uygulamalar yapalım.



Yukarıdaki örneği incelerseniz 2. satırdaki izmir ifadesinin silinmiş olduğunu göreceksiniz.



Evet bu sefer de 2,3 ve 4. satırlar silinmiş.

Örneğin bir text dosyamız olsun. Bu dosyadaki **Zeynocan** kelimesini **Zeynep** kelimesiyle değiştirelim. Bunun için **sed "s/Zeynocan/Zeynep/" <dosya>** komutunu kullanmamız gerekir.

```
root@kemal:~# cat zeyno
Zeynocan bak güzel kitaP
Kitap okumak eyidir eyi
Yerli malı yurdun malı herkeşler kitap okumalı
Zeytin de yurdun malı
root@kemal:~# sed "s/Zeynocan/Zeynep/" zeyno
Zeynep bak güzel kitaP
Kitap okumak eyidir eyi
Yerli malı yurdun malı herkeşler kitap okumalı
Zeytin de yurdun malı
root@kemal:~#
```

Eğer Zeynocan kelimesi bir kaç yerde geçiyor olsaydı bu sefer **s/Zeynocan/Zeynep/g** şeklinde yani kalıbı **g** ile beraber kullanacaktık. Bununla birlikte değiştirilen ifadeyi başka bir dosyaya yazmak isteseydik **sed "s/Zeynocan/Zeynep/" zeyno > yenidosya** gibi bir komut kullanmamız gerekirdi.

Eğer dosyadaki bütün satırların ilk iki harfini silmek istersek aşağıdaki görüntüde kullanılan komutu kullanmamız gerekir.

```
root@kemal:~# sed "s/^..//" zeyno
ynocan bak güzel kitaP
tap okumak eyidir eyi
rli malı yurdun malı herkeşler kitap okumalı
ytin de yurdun malı
root@kemal:~#
```

Köşeli parantez kullanarak birden fazla seçim yapabiliriz. Aşağıda bununla ilgili bir örnek görüyorsunuz.

```
root@kemal:~# cat zeyno
Zeynocan bak güzel kitaP
Kitap okumak eyidir eyi
Yerli malı yurdun malı herkeşler kitap okumalı
Zeytin de yurdun malı
root@kemal:~# sed "s/[Kk]ita[Pp]/KiTAP/g" zeyno
Zeynocan bak güzel KiTAP
KiTAP okumak eyidir eyi
Yerli malı yurdun malı herkeşler KiTAP okumalı
Zeytin de yurdun malı
root@kemal:~#
```

Değiştirilen yerleri görmek için bir seçenek olabilir. Aşağıdaki örnek bununla ilgili.

```
root@kemal:~# cat zeyno
Zeynocan bak güzel kitaP
Kitap okumak eyidir eyi
Yerli malı yurdun malı herkeşler kitap okumalı
Zeytin de yurdun malı
root@kemal:~# sed "s/oku/(al)/" zeyno
Zeynocan bak güzel kitaP
Kitap (al)mak eyidir eyi
Yerli malı yurdun malı herkeşler kitap (al)malı
Zeytin de yurdun malı
root@kemal:~#
```

Şimdi de bir dosyadaki tüm büyük harf karakterlerini silelim. Bunu yapmak için **sed "s/[A-Z]//g" <dosya>** gibi bir komut kullanılmalıdır.

```
root@kemal:~# cat test
İSTANBUL, izmir, BURSA, ankara, Diyarbakır, bolU
123, 456, 000, 87654
root@kemal:~# sed "s/[A-Z]//g" test
, izmir, , ankara, iyarbakır, bol
123, 456, 000, 87654
root@kemal:~#
```

Rakamların ve küçük harflerin dışındaki her şeyi silmek için aşağıdaki gibi bir komut kullanmalısınız.

```
root@kemal:~# cat test
İSTANBUL, izmir, BURSA, ankara, Diyarbakır, bolU
123, 456, 000, 87654
root@kemal:~# sed "s/[^1-9a-z]//g" test
izmirankaraiyarbakırbol
12345687654
root@kemal:~#
```

Eğer text dosyamızda örneğin **/usr/share/bin** gibi bir ifade geçiyor ve biz bu ifadeyi **/usr** olarak değiştirmek istiyorsak ne yapmalıyız? Çünkü / karakteri özel karakter sınıfına giediği için bu karakterin özel anlamını yitirmesi için aşağıdakine benzer bir komut kullanılmalıdır.

```
root@kemal:~# cat test
 /usr/share/bin gibi bir
ifadeyi nasıl değiştireceğiz?
root@kemal:~# sed "s/\/usr\/share\/bin/\/usr/g" test
 /usr gibi bir
ifadeyi nasıl değiştireceğiz?
root@kemal:~#
```

Bu işlemi daha kolay yoldan yapmak için aşağıdaki gibi bir komut da kulanabilirsiniz.

```
root@kemal:~# cat test
 /usr/share/bin gibi bir
ifadeyi nasıl değiştireceğiz?
root@kemal:~# sed "s_/usr/share/bin_/usr_" test
 /usr gibi bir
ifadeyi nasıl değiştireceğiz?
root@kemal:~#
```

Bir de şuna bakın.

```
root@kemal:~# sed "s:/usr/share/bin:/usr:" test
 /usr gibi bir
ifadeyi nasıl değiştireceğiz?
root@kemal:~#
```

Bir metin dosyasındaki karakterlerden istediklerimizi değiştirelim. Örneğin i harfini d ile, f harfini u ile ve a harfini de r ile değiştirelim. Bunun için **sed "y/ifa/dur/" <dosya>** komutunu kullanmamız yeterlidir.

```
root@kemal:~# cat test
 /usr/share/bin gibi bir
ifadeyi nasıl değiştireceğiz?
root@kemal:~# sed "y/ifa/dur/" test
 /usr/shrre/bdn gdbd bdr
durdeyd nrsıl değdştdreceğdz?
root@kemal:~#
```

Bu konuyla ilgili daha bir çok örnek verilebilir. Fakat burada hepsiyle ilgili tek tek örnek vermem mümkün değil. İnternette araştırma yaparak sed aracının gelişmiş kullanımıyla ilgili bir çok örnek bulabilirsiniz. Son olarak bir dosyadan komutların alınıp işlenmesiyle ilgili küçük bir örnek verelim. Bir text dosyamıza şuna benzer ifadeleri yazıp **örnek.sed** adıyla kaydedelim.

s/izmir/NewYork/g
s/123/999/g

Sonra konsoldan bu dosyamızı **-f** ile belirterek kullanalım.

```
root@kemal:~# cat test
İSTANBUL, izmir, BURSA, ankara, Diyarbakır, bolU
123, 456, 000, 87654
root@kemal:~# sed -f örnek.sed test
İSTANBUL, NewYork, BURSA, ankara, Diyarbakır, bolU
999, 456, 000, 87654
root@kemal:~#
```

Siz de buna benzer olarak örneğin çalışacağınız dosya çok büyükse ve bir çok değişiklik yapmak istiyorsanız komutları bir dosya içine yazarak oradan da çağırabilirsiniz. Evet bu konu da bu kadar. Artık yeni konumuza geçebiliriz.

awk kullanımı

Awk, metin dosyalarını istediğimiz ölçülere göre işleyerek değişiklikler yapmamızı sağlayan bir script dilidir. Yani yorumlayıcı bir dildir. Eğer bir dosyadan komutlar alınıyorsa bu dosyadaki komutlar sırasıyla yorumlanarak çalıştırılır. Tabi sadece dosyadan veri alarak değil komut satırından da parametrelerle istenilen işlemler yaptırılabilir. Başta da belirttiğimiz gibi **awk** ile yapılacak en temel işlem dosyalarda istenilen satırları belirli kriterlere göre aramaktır.

Biz burada komutları program dosyasından almaktansa daha çok komut satırından girerek çalışma üzerinde duracağız. Aşağıda küçük bir örnek var.

```
root@kemal:~# awk "BEGIN {print \"ilk programım\"}"
ilk programım
root@kemal:~#
```

Bir başka örnekte konsoldan **awk "{print}"** komutunu verdikten sonra yazılan ifadeyi ekrana yazdırmayı görüyoruz.

```
root@kemal:~# awk "{print}"
kemal
kemal
root@kemal:~#
```

Komutları bir dosyadan alarak kullanmak isterseniz bunun için **awk -f <program dosyası>** şeklinde bir komut kullanmanız gerekir. Bir text dosyasına awk kodlarını yazarak kaydedip daha sonra **-f** parametresiyle bu dosyanın yerini belirterek kullanabilirsiniz. Dosyanın her hangi bir uzantısı olmasına gerek olmamasına rağmen **.awk** uzantısıyla kaydederseniz bunun bir awk script dosyası olduğunu anlarsınız. Küçük çaplı işlemler için komutlarımızı komut satırından girerek kullanmak daha mantıklıdır. Eğer büyük bir dosya içinde çok sayıda işlem yapmanız gerekirse bu durumda bir script dosyası kullanmak daha mantıklıdır. Şimdi dosyadan kullanıma küçük bir örnek verelim. Aşağıdaki ifadeyi bir metin dosyasına yazarak uzantısını .awk olacak şekilde kaydedelim.

```
BEGIN {
 print “deneme olsun gari”
      }
```

Ben örnek.awk diye kaydettim. Kaydedilen dosyayı kullanmak için **awk -f örnek.awk** komutunu kullanıyorum. Ben ev dizinime kaydettiğim için böyle kullanıyorum. Eğer **/home** dizini altına kaydetmiş olsaydım bu sefer **awk -f /home/örnek.awk** şeklinde kullanacaktım.

```
root@kemal:~# awk -f örnek.awk
deneme olsun gari
root@kemal:~#
```

Awk nın kullanılan değişik versiyonları var. Orijinali awk olmakla birlikte **gawk** (GNU awk), **nawk** kullanımlarının da olduğunu belirteyim. Ben örnekleri awk üzerinden vereceğim ama siz isterseniz nawk de kullanabilirsiniz.

```
root@kemal:~# which awk
/usr/bin/awk
root@kemal:~# which nawk
/usr/bin/nawk
root@kemal:~# nawk -f örnek.awk
deneme olsun gari
root@kemal:~#
```

Biz daha çok konsol kullanımından bahsedeceğimizden örnekler de **awk** nın konsol kullanımıyla yani komutları dosyadan almak yerine konsoldan girmeyle ilgili olacaktır. Konsoldan kullanımla ilgili bilinmesi gereken kurallar vardır. Bunlar:

' : komutlar tırnak işareti içinde yazılır, sonra işlem yapılacak dosya belirtilir.

$: işlenecek dosyadaki kolonları ifade eder. Her hengi bir değişiklik yapılmazsa kolonlar boşluklara ayrılmış kabul edilir. **$0** tüm satırlar(her bir satır için satırın tamamına karşılık gelen ifade), **$1** satırın birinci kolonu, **$2** satırın ikinci kolonu demektir.

NR : toplam kayıt sayısı(satır sayısı)

NF : sütun(kolon/alan) sayısı

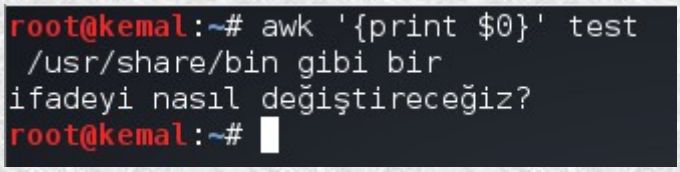
-F : alan ayracı işaretini belirtmek için

/../ : düzenli ifadeler iki / arasında kullanılır

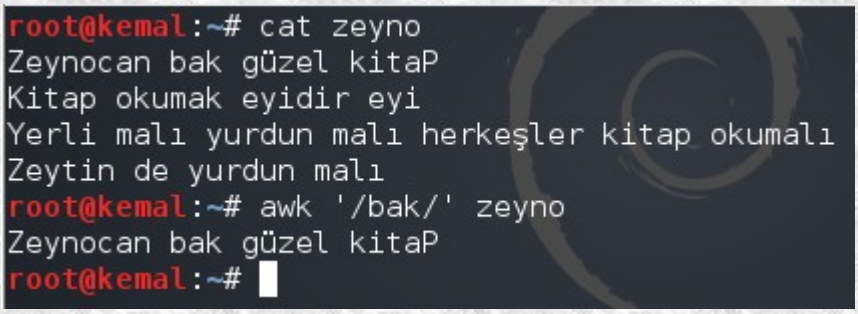
FS : alan ayracı

Ayrıca awk içerisinde matematiksel operatörler de kullanılabilir. (**+, +=, ++, -, -=, --, *, *=, /, /=**)
Karşılaştırma ifadeleri olarak da **<, <=, >, >=, ==, !=** gibi ifadeler kullanılabilir.

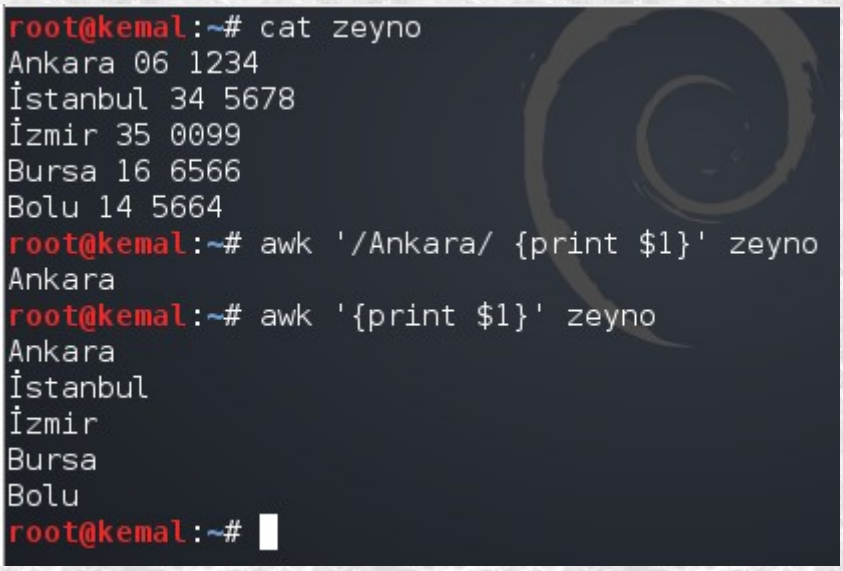
Şimdi bazı örnekler yapalım. Örneğin dosyanın içeriğini olduğu gibi göstermek için **awk '{print $0}' <dosya>** komutu kullanılır.



Dosyada geçen her hangi bir ifadeyi sorgulamak için '/../' kalıbını kullanmalıyız. Sorgulanan ifadeyle ilgili olan satır ya da satırlar görüntülenecektir.



Aradığımız ifade bir text dosyasının içindeki satırlarda birbiriyle ayrılmış bölümlerin en başında geçiyor mu öğrenmek için **awk '/Ankara/ {print $1}' zeyno** gibi bir komut kullanırız. Eğer metin dosyasındaki satırların en başında bulunan ayrılmış ifadelerin tamamını görüntülemek istersek bu sefer **awk '{print $1}' zeyno** komutunu kullanmalıyız. Tabi ben zeyno dosyasının içeriğini silip yeni şeyler yazmıştım. Yani bu zeyno yukarıdaki örneklerdeki zeyno değil :)



Dosya içindeki ifadeler arasında ayraç olarak boşluk değil de **:** işareti olsaydı o zaman **awk '{print $1}'** zeyno komutuyla istediğimiz sonucu alamayacaktık. İstediğimiz sonucu almak için **awk -F: '{print $1}' zeyno** komutunu kullanmamız gerekecekti. Eğer komutta **-F** ile alan tanımlaması yapılmamışsa alanların boşlukla ayrıldığı varsayılacaktır.

```
root@kemal:~# awk '{print $1}' zeyno
Ankara:06:1234
İstanbul:34:5678
İzmir:35:0099
Bursa:16:6566
Bolu:14:5664
root@kemal:~# awk -F: '{print $1}' zeyno
Ankara
İstanbul
İzmir
Bursa
Bolu
root@kemal:~#
```

Ayrılmış alanlarda (kolonlarda/sütunlarda) bulunan 1. ve 3. ifadeleri görmek içinse **awk -F: '{print $1,$3}' zeyno** komutu kullanılır. **$1** ve **$3** arasında **virgül** kullanmazsanız görüntülediğiniz alanlardaki ifadeler birbirine yapışık olarak çıkacaktır.

```
root@kemal:~# awk -F: '{print $1,$3}' zeyno
Ankara 1234
İstanbul 5678
İzmir 0099
Bursa 6566
Bolu 5664
root@kemal:~#
```

Ayraç işareti **:** yerine **,** olsaydı o zaman **-F:** yerine **-F”,”** kullanacaktık. **Alan ayracının** belirtilmesiyle ilgili olarak aşağıdaki örneği inceleyebilirsiniz. Komutla söylenen **:** ile ayrılmış alanlardan 3. alanda bulunan ifadeyi ekrana yazdırmasıdır. İki komut da aynı işlevi görmektedir.

```
root@kemal:~# awk -F: '{print $3}' zeyno
1234
5678
0099
6566
5664
root@kemal:~# awk '{print $3}' FS=":" zeyno
1234
5678
0099
6566
5664
root@kemal:~#
```

Aşağıdaki örnekte de **zeyno** dosyasının birinci alanında bulunan şehir isimleri önünde **Şehir:** ifadesi bulunduğu halde görüntülenir. Kullanılan komuta dikkat ettiğinizde **"Şehir: "** şeklinde bir kullanım görürsünüz. Eğer arada boşluk bırakılmazsa görüntülenen ifadede **:** işaretinden sonra **boşluk** bırakılmadan şehir isimleri yer alır.

```
root@kemal:~# cat zeyno
Ankara:06:1234
İstanbul:34:5678
İzmir:35:0099
Bursa:16:6566
Bolu:14:5664
root@kemal:~# awk -F: '{print "Şehir: " $1}' zeyno
Şehir: Ankara
Şehir: İstanbul
Şehir: İzmir
Şehir: Bursa
Şehir: Bolu
root@kemal:~#
```

Bir dosyadaki tüm kayıt içinden sadece istenilen ifadelerin geçtiği satırları görüntülemek için, örneğin test2 dosyamızdaki kayıtlardan İzmir ile ilgili olanları görmek için **awk -F: '$1 == "İzmir" {print $0}' test2** komutunu kullanıyoruz. Ayraç olarak **:** kullanıldığından **-F:** kullanmayı unutmuyoruz. Eğer boşluklarla ayrılsaydı o zaman ayraç işaretini belirtmeye gerek yoktu.

```
root@kemal:~# cat test2
Ankara:06:234:zurnacan
Ankara:006:176:davulcan
İstanbul:34:8998:tembelcan
İzmir:35:90998:heykırmen
İzmir:035:4647747:manyakcan
root@kemal:~# awk -F: '$1 == "İzmir" {print $0}' test2
İzmir:35:90998:heykırmen
İzmir:035:4647747:manyakcan
root@kemal:~#
```

Sadece İzmir ifadesi geçen satırların alanlarıyla ilgili (örneğin 1. alan, 2. alan) bilgi almak için aşağıdaki komutu kullanmalıyız.

```
root@kemal:~# awk -F: '$1 == "İzmir" {print $1}' test2
İzmir
İzmir
root@kemal:~# awk -F: '$1 == "İzmir" {print $2}' test2
35
035
root@kemal:~#
```

Örneğin verdiğimiz iki ifadenin de aynı satırda geçtiği kayıtları görüntülemek için aşağıdaki örnektekine benzer bir komut kullanılmalıdır.

```
root@kemal:~# awk -F: '/İzmir/ && /heykır/' test2
İzmir:35:90998:heykırmen
root@kemal:~#
```

Görüldüğü gibi **&&** operatörü "**ve**" anlamına gelmektedir.

Aynı dosyada, içinde İzmir **veya** İstanbul geçen kayıtları görüntülemek için **||** operatörünü kullanmak gerekir.

```
root@kemal:~# awk -F: '/İzmir/ || /İstanbul/' test2
İstanbul:34:8998:tembelcan
İzmir:35:90998:heykırmen
İzmir:035:4647747:manyakcan
root@kemal:~#
```

İçinde İzmir geçmeyen ifadeleri sorgulamak için de aşağıdaki komut işimize yarayacaktır.

```
root@kemal:~# awk -F: '!/İzmir/' test2
Ankara:06:234:zurnacan
Ankara:006:176:davulcan
İstanbul:34:8998:tembelcan
root@kemal:~#
```

Bir belgedeki karakterleri büyük karakter yapmak için aşağıdaki komutu kullanıyoruz.

```
root@kemal:~# cat test3
kakara kikiri kokoro
hebele hübele tebele
falan filan yalan
root@kemal:~# awk '{print toupper($0) }' test3
KAKARA KIKIRI KOKORO
HEBELE HüBELE TEBELE
FALAN FILAN YALAN
root@kemal:~#
```

Sadece dosyaları işlemede değil komut çıktılarını düzenlemede de **awk** kullanabiliriz. Aşağıdaki örnekte **ls -l** komutunun çıktısı istenilen kriterlere göre düzenleniyor.

```
root@kemal:~# ls -l | awk '{ print $5,$9 }'

47 belge1
627 deneme
604 deneme.o
233 deneme.s
4096 Desktop
4096 Downloads
4096 firefox-flash
57 kml
112 merhaba.py
38 örnek.awk
4096 taslak
```

Bir de passwd dosyasındaki **:** ile ayrılmış olan 1. ve 5. alanları görelim.

```
root@kemal:~# awk -F: '{print $1,$5 }' /etc/passwd
root root
daemon daemon
bin bin
sys sys
sync sync
games games
man man
lp lp
mail mail
news news
uucp uucp
proxy proxy
www-data www-data
```

Bir dosyadaki örneğin 3. satırdan sonraki satırları görmek için **NR>3** ifadesi kullanılır.

```
root@kemal:~# cat zeyno
Ankara:06:1234
İstanbul:34:5678
İzmir:35:0099
Bursa:16:6566
Bolu:14:5664
root@kemal:~# awk 'NR>3 {print}' zeyno
Bursa:16:6566
Bolu:14:5664
root@kemal:~#
```

Bir dosyadaki satır sayısını görmek için:

```
root@kemal:~# awk -F: 'END {print NR}' /etc/shadow
41
root@kemal:~#
```

İstenilen dosyanın içeriğini numaralayarak göstersin derseniz:

```
root@kemal:~# awk -F: '{print NR " " $0}' /etc/shadow
1 root:$6$RHxkvp3f$uMNlDynuT.ZuPV5zUb4TU2M0u5E1mQMPzLPz4sSpTlC
6F/kiumtKLcHvhrNp8K1Lx60.:15777:0:99999:7:::
2 daemon:*:15775:0:99999:7:::
3 bin:*:15775:0:99999:7:::
4 sys:*:15775:0:99999:7:::
5 sync:*:15775:0:99999:7:::
6 games:*:15775:0:99999:7:::
7 man:*:15775:0:99999:7:::
8 lp:*:15775:0:99999:7:::
9 mail:*:15775:0:99999:7:::
10 news:*:15775:0:99999:7:::
11 uucp:*:15775:0:99999:7:::
12 proxy:*:15775:0:99999:7:::
```

Bu konuyla ilgili çok fazla ve farklı örnekler verilebilir. Tabi burada hepsini anlatmak imkansız. Onun için ayrı bir çalışma yapmak lazım. İnternette **awk** nın çok farklı kullanımlarına bol miktarda örnekler bulabileceğiniz siteler var. Bu sitelerde hangi kalıbı kullanırsanız ne olur tarzında örnekler mevcut. Örneğin http://www.commandlinefu.com/ adresinde bir çok linux komutuyla ilgili çok güzel örnekler bulabilirsiniz.

Şimdiki konumuz **xargs** ile ilgili.

xargs

Kendisine girdi olarak verilen verileri tek tek kendisinden sonraki programa argüman olarak verir. Kullanımı çok basittir. Basit bir örnek üzerinden açıklayalım. Örneğin **önemli** adında bir dosyamız olsun ve dosyamızın içinde **/etc** ifadesi yer alsın. Şu komutu kullanalım: **cat önemli | xargs ls**

```
root@kemal:~# cat önemli
/etc
root@kemal:~# cat önemli | xargs ls
adduser.conf            ImageMagick         polipo
adjtime                 icedtea-web         polkit-1
aliases                 iceweasel           postgresql
alternatives            idmapd.conf         postgresql-common
amap                    ifplugd             ppp
apache2                 init                privoxy
apg.conf                init.d              profile
apm                     initramfs-tools     profile.d
apparmor.d              inittab             protocols
```

Komutun ne yaptığını açıklayacak olursak: İlk önce **cat** komutu ile dosyamızın içeriğini gördük ve **xargs** bu içeriği argüman olarak **ls** komutuna verdi. Sonuç olarak **/etc** dizininin içeriği listelendi.

Başka bir uygulama yapalım. Örneğin ev dizinimde bulunan **.txt** uzantılı dosyaları bulup silmek istiyorum diyelim. Bunun için **ls *.txt | xargs rm** komutunu kullanmalıyım.

```
root@kemal:~# ls *.txt
beş.txt  bir.txt  dört.txt  iki.txt  üç.txt
root@kemal:~# ls *.txt | xargs rm
root@kemal:~# ls *.txt
ls: *.txt'e erişilemedi: Böyle bir dosya ya da dizin yok
root@kemal:~#
```

Örneğin ev yani çalışma dizininizdeki **.log** uzantılı dosyaları silmek isterseniz **find -name "*.log" | xargs rm -f** komutunu kullanmanız gerekir.

```
root@kemal:~# find -name "*.log" | xargs rm -f
```

Bir başka örnek. Bu sefer /etc dizinimizdeki konfigürayon dosyalarını yani .conf uzantılı dosyaları listeleyelim. Tabi eğer sadece bulmak istersek **find /etc -name "*.conf"** komutunu kullanmamız yeterli. Fakat ls -l ile ayrıntılı liste almak için **find /etc -name "*.conf" | xargs ls -l** komutundan faydalanabiliriz.

```
root@kemal:~# find /etc -name "*.conf" | xargs ls -l
-rw-r--r-- 1 root     root       2981 Mar 11 22:01 /etc/adduser.conf
-rw-r--r-- 1 root     root       9640 Eki 30  2012 /etc/apache2/apache2.conf
lrwxrwxrwx 1 root     root         45 Mar 13 19:14 /etc/apache2/conf.d/javascrip
t-common.conf -> /etc/javascript-common/javascript-common.conf
-rw-r--r-- 1 root     root        332 Eki 30  2012 /etc/apache2/mods-available/a
ctions.conf
-rw-r--r-- 1 root     root        811 Eki 30  2012 /etc/apache2/mods-available/a
lias.conf
-rw-r--r-- 1 root     root       3265 Eki 30  2012 /etc/apache2/mods-available/a
utoindex.conf
-rw-r--r-- 1 root     root         69 Eki 30  2012 /etc/apache2/mods-available/c
gid.conf
-rw-r--r-- 1 root     root         37 Eki 30  2012 /etc/apache2/mods-available/d
av_fs.conf
```

Son örnek komutumuz da şu olsun: **find / -name *.jpg -type f -print | xargs tar -cvzf fotolar.tar.gz**

```
root@kemal:~# find / -name *.jpg -type f -print | xargs tar -cvzf fotolar.tar.gz
tar: Üye isimlerinden `/' kaldırılıyor
/usr/share/autopsy/pict/main_t_met_cur.jpg
/usr/share/autopsy/pict/menu_h_hdet.jpg
/usr/share/autopsy/pict/menu_b_help.jpg
/usr/share/autopsy/pict/file_h_mod_cur.jpg
/usr/share/autopsy/pict/but_alloc_list.jpg
/usr/share/autopsy/pict/file_h_acc_cur.jpg
/usr/share/autopsy/pict/file_h_uid_cur.jpg
/usr/share/autopsy/pict/tl_t_notes_link.jpg
/usr/share/autopsy/pict/but_report.jpg
```

Gerçi daha dosya sıkıştırma komutlarını işlemedik fakat bu komut şunu diyor: Kök dizinin altındaki dizinlerde uzantısı **jpg** olan dosyaları bul ekranda görüntüle ve bulduğun bu jpg dosyalarını sıkıştırarak fotolar isminde **arşiv** dosyası yap.

```
root@kemal:~# ls
belge1    deneme.s   firefox-flash
deneme    Desktop    fotolar.tar.gz
deneme.o  Downloads  kml
root@kemal:~#
```

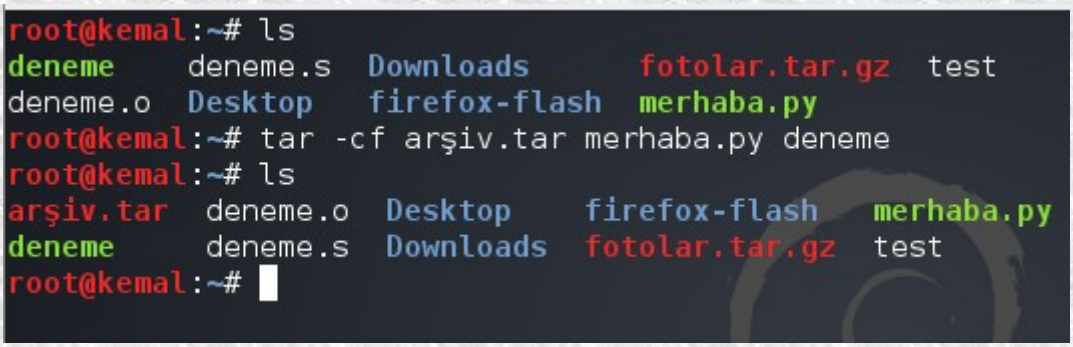
Evet istediğimiz gibi **fotolar.tar.gz** şeklinde arşiv dosyamız oluşturulmuş.

Bu konuyu da bitirdiğimize göre artık dosya sıkıştırma, açma, arşiv dosyası oluşturma konularından bahsedebiliriz.

Dosya Sıkıştırma-Açma ve Arşivleme İşlemleri

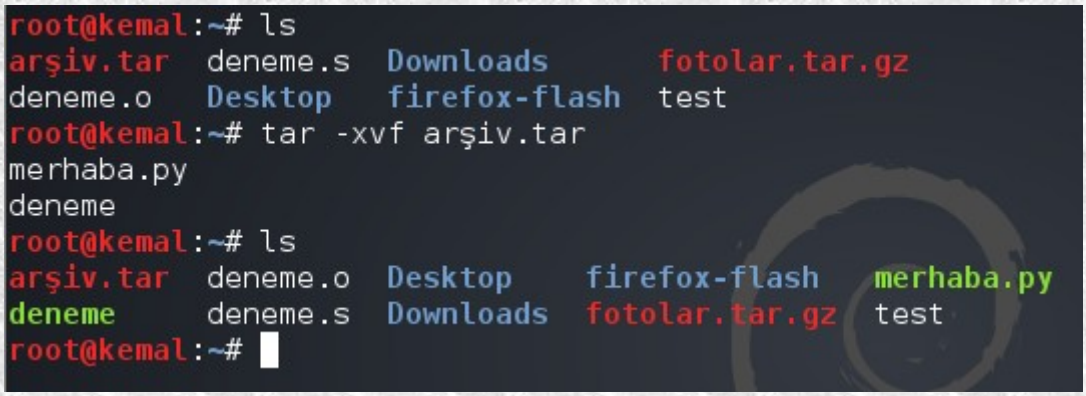
Linux sistemde istediğimiz dosyaları bir araya getirerek arşiv dosyası oluşturabiliriz. Ya da büyük boyutlu bir dosyayı arşiv dosyası yapabiliriz. Arşivleme işlemini ister normal olarak ister dosyaları sıkıştırarak arşivleme şeklinde tercih edebiliriz.

Tüm bu işlemler için de belli başlı komutlar/araçlar kullanmamız gerekir. En temel arşivleme işlemini **tar** komutuyla yaparız. Bu komut ile örneğin bir dizin ve altdizinlerini içinde bulunan dosyalarıyla birlikte bir araya toplayarak arşivleyebiliriz. Örneğin ev dizinimde bulunan **merhaba.py** ve **deneme** dosyalarını **tar** ile arşivlemek için **tar -cf arşiv.tar merhaba.py deneme** şeklinde bir komut kullanmam gerekir.



Yukarıdaki ekran görüntüsü şunu anlatmaktadır: İlk önce **ls** ile **ev dizinimdeki** (çalışma dizinimdeki) dosyalara baktım. Sonra bu dosyalardan **merhaba.py** ve **deneme** adındakileri **tar** komutuyla **arşiv** dosyası yaptım. Tekrar **ls** ile kontrol ettiğimde **arşiv.tar** dosyasının oluşturulduğunu gördüm. Komutta kullanılan **c** (create) parametresi oluştur anlamına gelirken, **f** parametresi ile oluşturulacak arşiv dosyasının ismi belirlenir.

Evet arşivi bu şekilde oluşturduk. Eğer arşiv dosyalarını açmak istersek bu sefer de **tar -xvf <dosya.tar>** komutunu kullanıyoruz. (bazıları **xvf** nin başında bulunan **–** işaretini kullanmıyor, bende alışkanlık olmuş o yüzden kullanıyorum) Komuttaki **x** parametresi extract(aç), **v** parametresi (verbose) açılan dosyaları görmek yani ayrıntılı komut çıktısı almak için kullanılır.



Evet görüldüğü gibi arşiv dosyamızın içinde bulunan dosyalar dışarı çıkartıldı. Yukarıda da söylediğimiz gibi arşivleme işlemini sıkıştırarak da yapabiliriz. Bunun için de iki farklı sıkıştırma aracını kullanabiliriz. Bunlar **gzip** ve **bzip2** araçlarıdır. Sıkıştırılmış arşiv dosyası oluşturulurken sıkıştırma işleminde **gzip** kullanılmışsa dosyanın uzantısı **tgz (ya**

da tar.gz), eğer **bzip2** kullanılmışsa **tar.bz2** olacaktır. Her ikisi ile de ilgili örnekler yapalım. Aşağıda tgz ile ilgili bir örnek görülmektedir.

```
root@kemal:~# tar -czvf arşiv2.tgz merhaba.py deneme
merhaba.py
deneme
root@kemal:~# ls
arşiv2.tgz  deneme    deneme.s  Downloads      fotolar.tar.gz  test
arşiv.tar   deneme.o  Desktop   firefox-flash  merhaba.py
root@kemal:~#
```

Görüntüden **arşiv2.tgz** dosyamızın oluşturulduğu görülüyor. Burada **z** parametresi, **gzip** ile sıkıştırarak arşiv dosyası oluşturmak istediğimizi gösteriyor.

Şimdi de **bzip2** kullanarak sıkıştırılmış arşiv dosyası oluşturalım. Oluşturulacak dosyanın uzantısı bu durumda **tar.bz2** olmalı.

```
root@kemal:~# tar -cjvf arşiv3.tar.bz2 merhaba.py deneme
merhaba.py
deneme
root@kemal:~# ls
arşiv2.tgz      arşiv.tar  deneme.o  Desktop    firefox-f
arşiv3.tar.bz2  deneme     deneme.s  Downloads  fotolar.t
root@kemal:~#
```

Yine görüleceği gibi **arşiv3.tar.bz2** dosyamız oluşturulmuş. Burada dikkatinizi çekmiş olmalı, eğer **gzip** ile sıkıştırarak **tgz(ya da tar.gz)** dosyası oluşturacaksak **czvf**, **bzip2** ile sıkıştırarak **tar.bz2** dosyası oluşturacaksak **cjvf** kullanıyoruz. Değişen sadece **z** ve **j** parametreleri.

Aynen sıkıştırarak arşivlemede olduğu gibi, dosyalarımızı açarken de aynı parametreleri kullanıyoruz. Yani **tgz(ya da tar.gz)** dosyasını açarken **z**, **tar.bz2** dosyasını açarken **j** kullanıyoruz. Fakat bu sefer **c** (create) değil **x** (extract) diyoruz. Aşağıdaki örneklere bakalım.

```
root@kemal:~# ls
arşiv2.tgz      deneme.o  Desktop    firefox-flash
arşiv3.tar.bz2  deneme.s  Downloads  test
root@kemal:~# tar -xzvf arşiv2.tgz
merhaba.py
deneme
root@kemal:~# ls
arşiv2.tgz      deneme    deneme.s  Downloads      merhaba.py
arşiv3.tar.bz2  deneme.o  Desktop   firefox-flash  test
root@kemal:~#
```

```
root@kemal:~# ls
arşiv2.tgz      deneme.o  Desktop    firefox-flash
arşiv3.tar.bz2  deneme.s  Downloads  test
root@kemal:~# tar -xjvf arşiv3.tar.bz2
merhaba.py
deneme
root@kemal:~# ls
arşiv2.tgz      deneme    deneme.s  Downloads      merhaba.py
arşiv3.tar.bz2  deneme.o  Desktop   firefox-flash  test
root@kemal:~#
```

Yukarıdaki örneklerde gayet açık şekilde **xzvf** ve **xjvf** kullanımları görülüyor. Siz de buna benzer denemeler yapabilirsiniz.

Arşiv dosyası oluşturmadan dosyalarımızı sadece sıkıştırmak istersek bunun için **gzip-gunzip**, **bzip2-bunzip2** araçlarından faydalanabiliriz. Sıkıştırmak için **gzip** kullandığınız dosyayı açarken **gunzip**, sıkıştırıken **bzip2** kullandığınız dosyayı açarken de **bunzip2** kullanmalısınız. Aşağıdaki örnekleri inceleyebilirsiniz.

```
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  test
root@kemal:~# gzip test
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  test.gz
root@kemal:~# gunzip test.gz
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  test
root@kemal:~#
```

Bu örnekte **test** isimli dosyamızı **gzip** ile sıkıştırıyoruz, daha sonra sıkıştırılmış **test.gz** dosyasını **gunzip** ile açıyoruz.

```
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  test
root@kemal:~# bzip2 test
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  test.bz2
root@kemal:~# bunzip2 test.bz2
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  test
root@kemal:~#
```

Bu örnekte de **test** isimli dosyamızı bu sefer **bzip2** ile şıkıştırıyoruz, daha sonra sıkıştırılmış **test.bz2** dosyasını **bunzip2** ile açıyoruz.

Linux sistemlerde **rar** ya da **zip** uzantılı sıkıştırılmış dosyaları da açabilirsiniz. Bunun için **unrar** ve **unzip** araçlarından faydalanabilirsiniz. Ben aşağıda rar dosyasının açılmasıyla ilgili bir örnek verdim.

```
root@kemal:~/Downloads# ls
1118362187.pdf
Android.rar
assembly
assembly_language_for_x86_processors.pdf
```

Çalışma dizinimdeki **Downloads** altında **Android.rar** diye bir dosya var. Bu dosyayı açmak için **unrar e Android.rar** komutunu kullanıyorum.

```
root@kemal:~/Downloads# unrar e Android.rar

UNRAR 4.10 freeware      Copyright (c) 1993-2012 Alexander Roshal


Extracting from Android.rar

Extracting  Android_Forensics_-_Andrew_Hoog.epub                      OK
Extracting  Android_Forensics_-_Andrew_Hoog.mobi                      OK
Extracting  readme.txt                                                OK
Extracting  sharebookfree.com.url                                     OK
All OK
root@kemal:~/Downloads#
```

Rarlı dosyamızın içindekileri çıkardık. Kullanımı bu kadar basit. Eğer rarlı dosya şifreliyse o vakit şifreyi girmenizi bekliyor ve şifre girildikten sonra açıyor. Yani farklı bir şey yok.

Dosya Kopyalama Taşıma Silme İşlemleri

Dosya kopyalamak için kullanılan komut **cp** komutudur. Bu komutun kullanım formu **cp <kopyalanacak dosya> <kopyalanan dosya>** şeklindedir. Yani dosya1 isimli dosyayı başka bir yere dosya2 ismiyle kopyalamak için **cp <dosya1><dosya2>** komutu kullanılır.

```
root@kemal:~# ls
deneme     deneme.s  Downloads      merhaba.py
deneme.o   Desktop   firefox-flash  test
root@kemal:~# cp deneme kopya
root@kemal:~# ls
deneme     deneme.s  Downloads      kopya        test
deneme.o   Desktop   firefox-flash  merhaba.py
root@kemal:~#
```

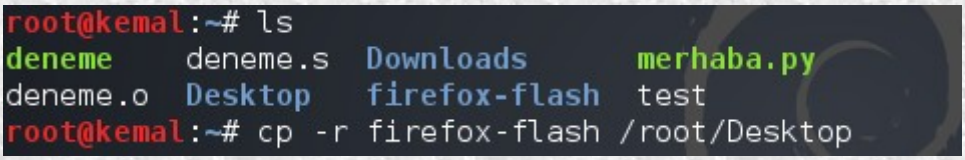
Yukarıdaki örnekte **deneme** adlı dosyayı **kopya** adıyla aynı dizine kopyaladığımızı görüyorsunuz.

Kopyalarken sorsun diyorsanız (yani kopyalanan yerde aynı adlı dosya olup içine yazması ihtimaline karşı) **-i** parametresiyle kullanmalısınız.

Örneğimizde görüldüğü gibi dosyayı kopyalamak istediğimiz yerde aynı isimde başka bir dosya olduğundan onayımızı sordu. (biz hayır anlamında **h** dedik ama ingilizler nayır anlamında **n** derler :)

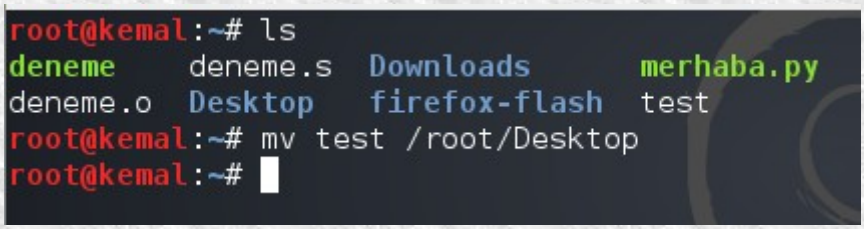
Eğer dosyamızı örneğin **/home** dizini altına kopyalamak isteseydik bu durumda **cp deneme /home/kopya** gibi bir komut kullanmamız gerekecekti. Tabi **/home/kopya** yerine **/home** deseydiniz **aynı adla** yani **deneme** adıyla **/home** dizininin altında kopya oluşturacaktı.

Dosya kopyalama işlemi bu şekilde oluyor. Dizin kopyalamada da yine aynı komutu kullanıyoruz fakat bu sefer **-r** parametresiyle birlikte. Bu parametre dizin altındaki dosyaları da kopyalamamız için.

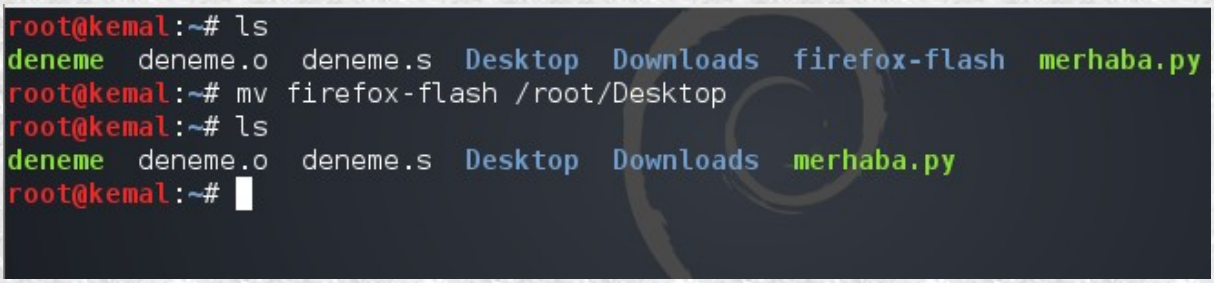


Örneğin çalışma dizinimizde bulunan firefox-flash dizinini masaüstüne kopyalamak için yukarıdaki komutu kullanabiliriz.

Dosya kopyalamak yerine eğer taşımak istersek bunun için de **mv** komutunu kullanıyoruz. Bu komutun da kullanımı cp gibidir fakat kopyalama yerine taşıma işlemi yapar.



Yukarıdaki komutla test isimli dosyayı masaüstüne taşıdık. Dizini de aynı komutu kullanarak taşıyabiliriz. Aşağıdaki örnekte olduğu gibi.



Taşıdığımız **firefox-flash** dizini/klasörü artık masaüstünde olduğundan çalışma dizinimiz altında görünmüyor.

Dosya ve dizin silmeye gelince, önceki konularda dizin silme işlemlerinden bahsetmiştik. Burada dosyaları silme işlemlerinden bahsedelim. Dosya ve dizin silme işlemlerinde **rm** komutunu kullanıyoruz. Tek bir dosya silebileceğimiz gibi bir çok dosyayı da yan yana belirtip silebiliriz. Aşağıda bir örnek görüyorsunuz.

Gördüğünüz gibi **rm** komutuyla aynı anda **bir.txt**, **iki.py** ve **üç.pl** dosyalarını sildik. Dizin silme konusundan daha önceki konularda bahsettik demiştim ama yine de bir hatırlatma yapalım. Dizin silmek için **rm -r** komutunu kullanıyoruz. Eğer dizin ve altındaki dosyalar silinmeden önce bize sorulsun diyorsak **rm -ir**, eğer sorgusuz sualsiz ne varsa silsin diyorsak **rm -rf** (tehlikeli bir komut) kullanıyoruz.

Link (Kısayol) Kavramı

Linux sistemlerde kısayol da diyebileceğimiz bir **link** kavramı vardır. Buna sembolik bağlantı denir. (**symlink**)

Sembolik bağlantı oluşturmak için **ln** komutu **s** parametresiyle birlikte **ln -s** şeklinde kullanılır. Bu kavramı daha iyi anlamak açısından bir örnek verelim. Ondan sonra sembolik link oluşturmayla ilgili bir kaç uygulama yaparız.

Benim kullandığım Linux dağıtımında (**Kali Linux**) web tarayıcı olarak **iceweasel** tarayıcı kullanılıyor. Bu tarayıcı aslında **firefox** temeli üzerine kurulu ve firefox tarayıcıya çok benzer. Ben konsoldan iceweasel komutunu da versem firefox komutunu da versem bu tarayıcı açılıyor. Sistemimde firefox kurulu olmamasına rağmen firefox komutuyla tarayıcıyı açabiliyorum fakat iceweasel tarayıcıyı açılıyor tabi.

```
root@kemal:~# which iceweasel
/usr/bin/iceweasel
root@kemal:~# which firefox
/usr/bin/firefox
root@kemal:~#
```

Gördüğünüz gibi ikisinin de çalıştırılabilir dosyası **/usr/bin/** altında görünüyor. Şimdi **file** komutuyla kontrol edelim.

```
root@kemal:~# file /usr/bin/iceweasel
/usr/bin/iceweasel: symbolic link to `../lib/iceweasel/iceweasel'
root@kemal:~# file /usr/bin/firefox
/usr/bin/firefox: POSIX shell script, ASCII text executable
root@kemal:~#
```

Anlaşılabileceği gibi birisi (firefox) çalıştırılabilir script, diğeri de (iceweasel) usr/lib/iceweasel/iceweasel konumunu gösteren bir **sembolik link**.

```
root@kemal:~# cat /usr/bin/firefox
#!/bin/sh

FIREFOX="$(which $0)"
[ -x "$FIREFOX.real" ] && exec "$FIREFOX.real" "$@"

exec iceweasel "$@"
root@kemal:~#
```

Firefox **bash scriptinin** içine baktığımızda, **firefox** komutu verildiğinde **iceweasel** komutunu çalıştırması için ayarlanmış. Dolayısıyla iceweasel komutunu kullanmış oluyoruz. Sonra da program **/usr/lib/iceweasel** altındaki **iceweasel** tarayıcı programını çalıştırıyor. Çünkü oraya link edilmiş.

```
root@kemal:~# cd /usr/lib/iceweasel
root@kemal:/usr/lib/iceweasel# ls
application.ini  chrome.manifest  extensions   icons     webapprt-stub
blocklist.xml    components       firefox-bin  modules   xulrunner
chrome           defaults         iceweasel    webapprt
root@kemal:/usr/lib/iceweasel# file iceweasel
iceweasel: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamic
ally linked (uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=0x4bd295
cc5824fca00ad79a74f0898da4728b7a56, stripped
root@kemal:/usr/lib/iceweasel#
```

Yukarıdaki görüntüden de anlaşılabileceği gibi **/usr/lib/iceweasel** dizinine gittiğimizde, dizin altında bulunan iceweaselin asıl program dosyamız olduğunu görüyoruz. Yani **/usr/bin** altındaki iceweasel aslında buraya **bağlantı** yapan bir **sembolik link**. Bunu **ls -l /usr/bin/iceweasel** komutuyla da görebiliriz.

```
root@kemal:~# ls -l /usr/bin/iceweasel
lrwxrwxrwx 1 root root 26 Mar 13 19:15 /usr/bin/iceweasel -> ../lib
/iceweasel/iceweasel
root@kemal:~#
```

Komut çıktısındaki **->** işaretine dikkat ederseniz asıl program yerini gösteriyor. Yani böyle bir çıktı görürseniz ok işaretinin sol tarafındaki yer sembolik bağlantıyı, sağ taraf ise bağlantı yapılan yeri göstermektedir.

Şimdi kendimiz bir sembolik link oluşturalım. Örneğin çalışma dizinimizde bulunan link.txt dosyasını yine aynı dizine (siz isterseniz başka bir dizin altına da link oluşturabilirsiniz) sembolik.txt adıyla link edelim. Bunun için **ln -s link.txt sembolik.txt** komutunu kullanmalıyız.

```
root@kemal:~# ls
deneme    deneme.s  Downloads      link.txt
deneme.o  Desktop   firefox-flash  merhaba.py
root@kemal:~# ln -s link.txt sembolik.txt
root@kemal:~# ls
deneme    deneme.s  Downloads      link.txt    sembolik.txt
deneme.o  Desktop   firefox-flash  merhaba.py
root@kemal:~#
```

Bir de kontrol edelim.

```
root@kemal:~# ls -l
toplam 28
-rwxr-xr-x  1 root root  627 Mar 17 12:54 deneme
-rw-r--r--  1 root root  604 Mar 17 12:54 deneme.o
-rw-r--r--  1 root root  233 Mar 17 16:13 deneme.s
drwxr-xr-x 11 root root 4096 Eyl  4 14:27 Desktop
drwx------  6 root root 4096 Eyl  4 11:11 Downloads
drwxr-xr-x  2 root root 4096 Ağu 29 19:16 firefox-flash
-rw-r--r--  1 root root    0 Eyl  4 14:25 link.txt
-rwxr-xr-x  1 root root  112 Ağu 20 17:50 merhaba.py
lrwxrwxrwx  1 root root    8 Eyl  4 14:29 sembolik.txt -> link.txt
root@kemal:~# file sembolik.txt
sembolik.txt: symbolic link to `link.txt'
root@kemal:~#
```

Hem **ok** işaretinden hem de **file** komutuyla gösterilen bilgiden sembolik linkin oluştuğunu anlıyoruz.

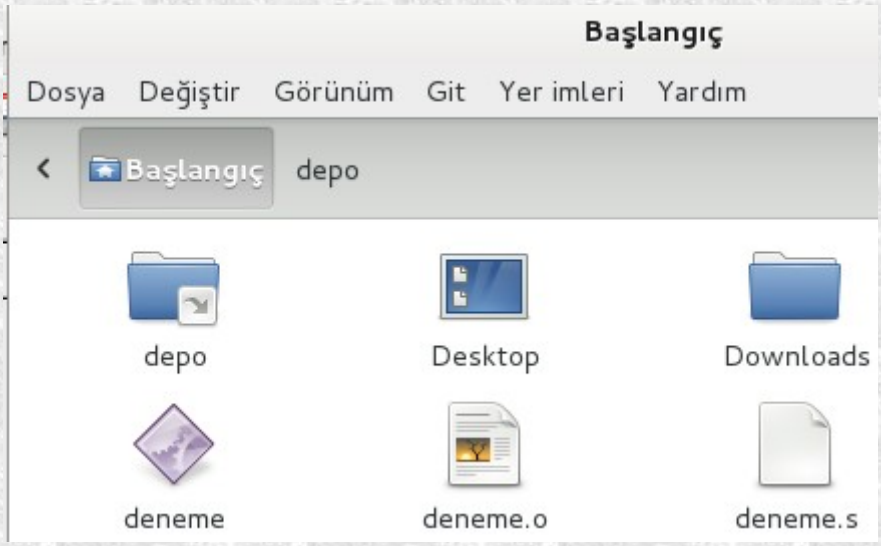
Sadece dosyalara değil **dizinlere** de sembolik link oluşturabiliriz. Örneğin **/home** dizinini **çalışma dizinime** sembolik linkle bağlayabilirim. Bunun için **ln -s /home** komutunu kullanmam yeterlidir. Ben çalışma dizinime bağlamak istediğimden bu komutu kullandım. Siz örneğin **/usr** altına bağlayacaksanız **ln -s /home /usr** komutunu kullanmalısınız.

```
root@kemal:~# ln -s /home
root@kemal:~# ls
deneme     deneme.s  Downloads      home       merhaba.py
deneme.o   Desktop   firefox-flash  link.txt
root@kemal:~# file home
home: symbolic link to `/home'
root@kemal:~#
```

Gördüğünüz gibi **/home** dizinime artık **kısayol** olarak **sembolik link** sayesinde ev dizinimden erişebilirim. Eğer home adıyla değil de örneğin depo adıyla çalışma (ev) dizininize sembolik link oluşturmak isterseniz bu sefer **ln -s /home depo** komutunu kullanmanız gerekir.

```
root@kemal:~# ln -s /home depo
root@kemal:~# ls
deneme     deneme.s  Desktop    firefox-flash  merhaba.py
deneme.o   depo      Downloads  link.txt
root@kemal:~# file depo
depo: symbolic link to `/home'
root@kemal:~#
```

Dosya yöneticisiyle ev dizinime baktığımda şöyle bir şey göreceğim.

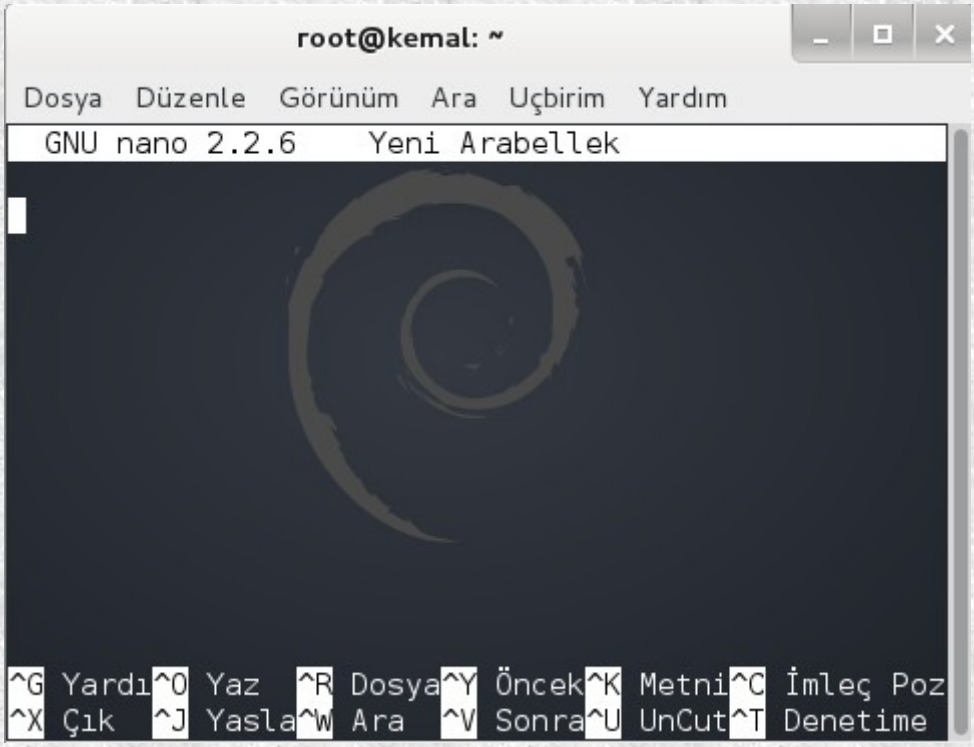


Üzerinde **ok** işareti olan **depo** dizinine/klasörüne tıkladığımda **/home** dizini açılacaktır.

Konsol Editörleri (Nano ve Vim)

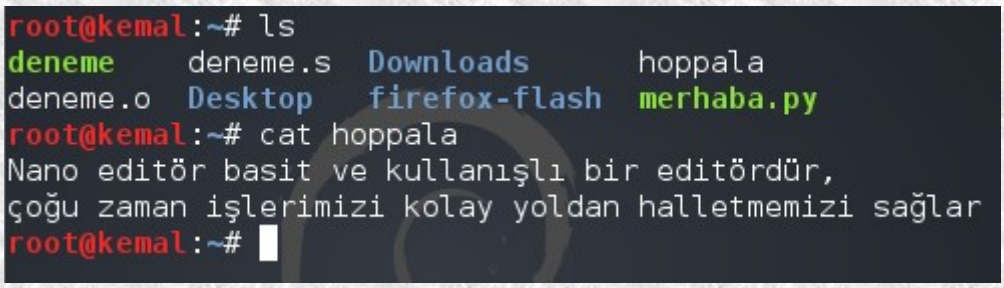
Linux işletim sisteminde kullanılan grafiksel metin editörleri (**gedit** vs.) yanında bir de konsoldan kullanılan editörler vardır. Bunlardan **nano** ve **vim** konsol editörlerinden bahsedeceğiz. Nano editör daha çok basit işler için kullanılırken Vim editör profesyonel kullanım için idealdir. Öncelikle nano editörden bahsettikten sonra diğer editörümüz olan vim den bahsedeceğiz.

Nano ile çalışmaya başlamak için konsoldan **nano** komutunu vermek yeterlidir. Konsol, aşağıdaki gibi bir editöre dönüşecektir.



Editörün alt tarafında yer alan bilgilendirmelere dikkat ederseniz **^X**, **^O** gibi ifadeler görürsünüz. **^X** ifadesi **ctrl+X** anlamına gelmektedir. Yani bu durumda, yanında **^** işareti olan karakterleri **ctrl** tuşuyla beraber kullanacaksınız demektir. Evet bu kısa bilgiden sonra örnek bir çalışma yaparak kaydetmeyi gösterelim.

Çalışmamızı bitirdikten sonra **ctrl+O** (ctrl ve O harfi) tuşlarına basarak dosyamızı hangi adla kaydedeceğimizi belirtiyoruz. Örneğin ben dosyama **hoppala** adını veriyorum. (eğer kaydetmeden dosya adını değiştirmeye karar verirseniz **ctrl+C** ile işlemi iptal edebilirsiniz). Dosya adını belirttikten sonra **Enter** tuşuna basıyoruz ve kaydedip çıkmak için de **ctrl+X** diyoruz. Her hangi bir dizin belirtmediğimizden dosyamız ev dizinimize kaydedildi.



Çalışmanız bittikten ve **ctrl+O** ile dosya adını verdikten sonra da isterseniz **Entera** basıp dosya üzerinde çalışmaya devam edebilirsiniz. İşiniz bitip de çıkmak istediğinizde (**ctrl+X**) size değişiklikleri kaydetmek isteyip istemediğinizi soracaktır. Eğer evet derseniz (E ya da Y) dosyanız son haliyle kaydedilecektir. Hayır derseniz ismi belirleyip Entera bastığınız zamana kadar yaptığınız çalışmalar dosyaya kaydedilecektir. Daha sonra yapılan çalışmalar kaydedilmeyecektir. En kolayı çalışmaya başlarken dosyaya isim vermektir. Yani **nano hoppala** derseniz direk hoppala adlı dosya üzerinde çalışmaya başlarsınız. Çalışmanız bittikten

sonra da **ctrl+X** ile değişiklikleri onaylayarak çıkarsınız.

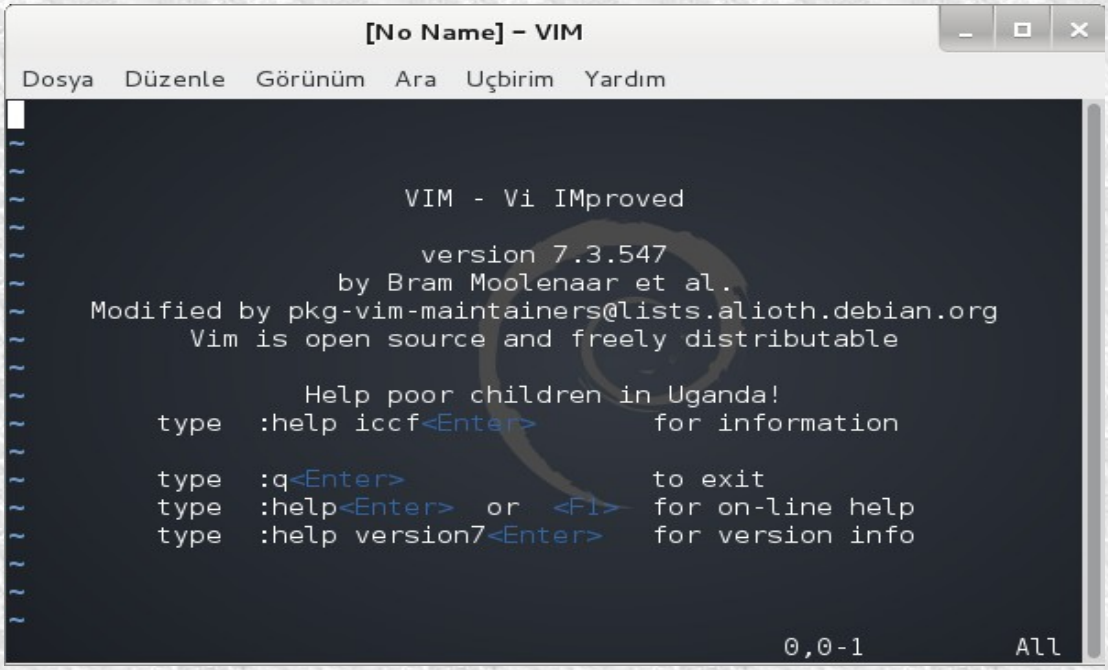
Her hangi bir **metin** ya da **script** dosyasını da nano editörle açabilirsiniz. Örneğin çalışma dizinimdeki muhteşem(!) bir programlama örneğini yani **merhaba.py** dosyasını nano editörle açmak için **nano merhaba.py** komutunu kullanmalıyım.

Python script dosyasındaki ifadeleri de renklendirmiş. Bu iyi bir şey.

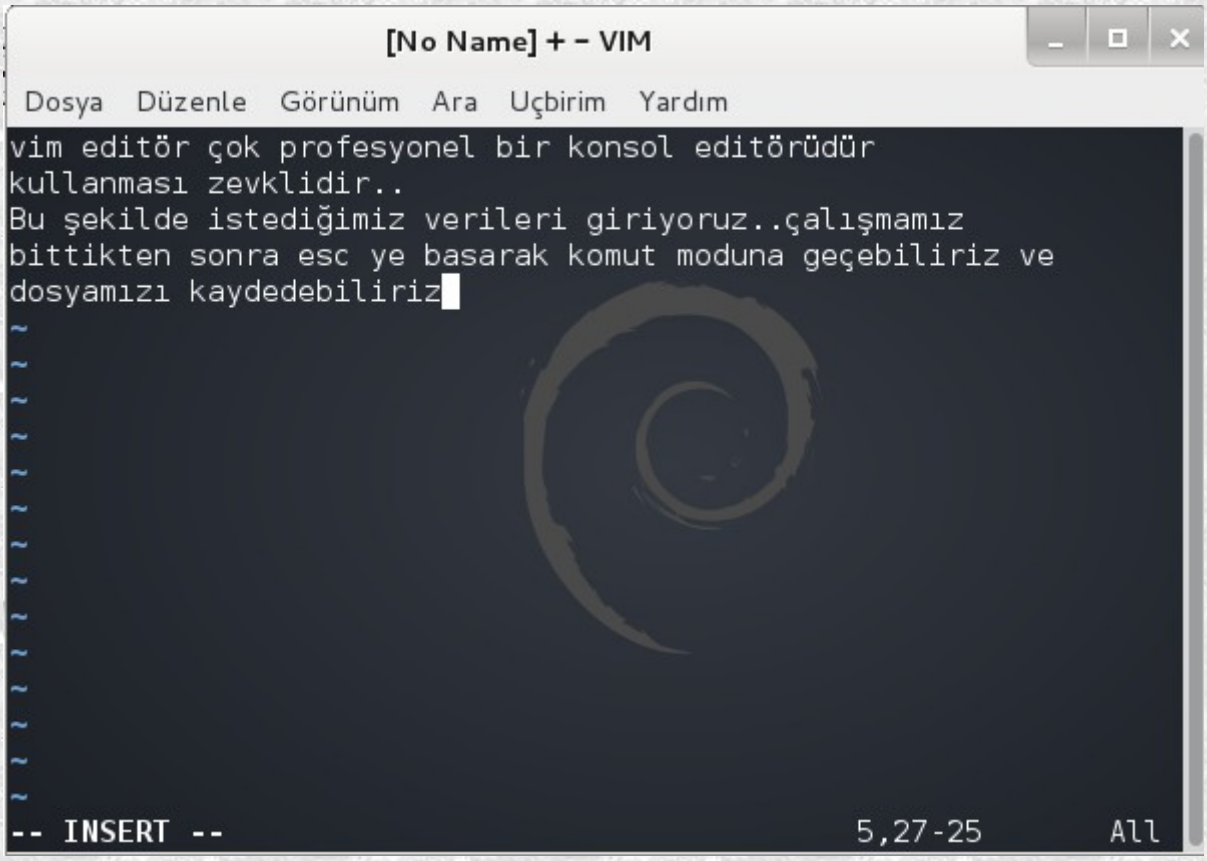
Nano editörle ilgili çok fazla örnek vermeye gerek yok. Zaten siz de denemeler yaparak ne kadar kolay kullanımı olduğunu göreceksiniz. Şimdi ikinci editörümüze, **Vim** editöre geçelim.

Profesyonel Linux kullanıcıları konsol editörü olarak eski adıyla **Vi**, yeni adıyla **Vim** editörü kullanmayı tercih ediyolar. Bu editörün çok fazla özelliği var fakat biz burada genel özelliklerinden bahsedeceğiz.

Konsoldan **vim** komutunu vererek vim editörü açabiliriz.



Konsol yukarıda görülen hale gelerek vim editör çalışmaya hazır bekliyor olacaktır. Çalışmaya başlamak için **i** (insert) tuşuna basabiliriz. Biz **i** tuşuna bastıktan sonra sol alt köşede **INSERT** uyarısı çıkarak gireceğimiz verileri almaya hazır olduğunu belirtecektir. Artık istediğimiz şekilde yazmaya başlayabiliriz.



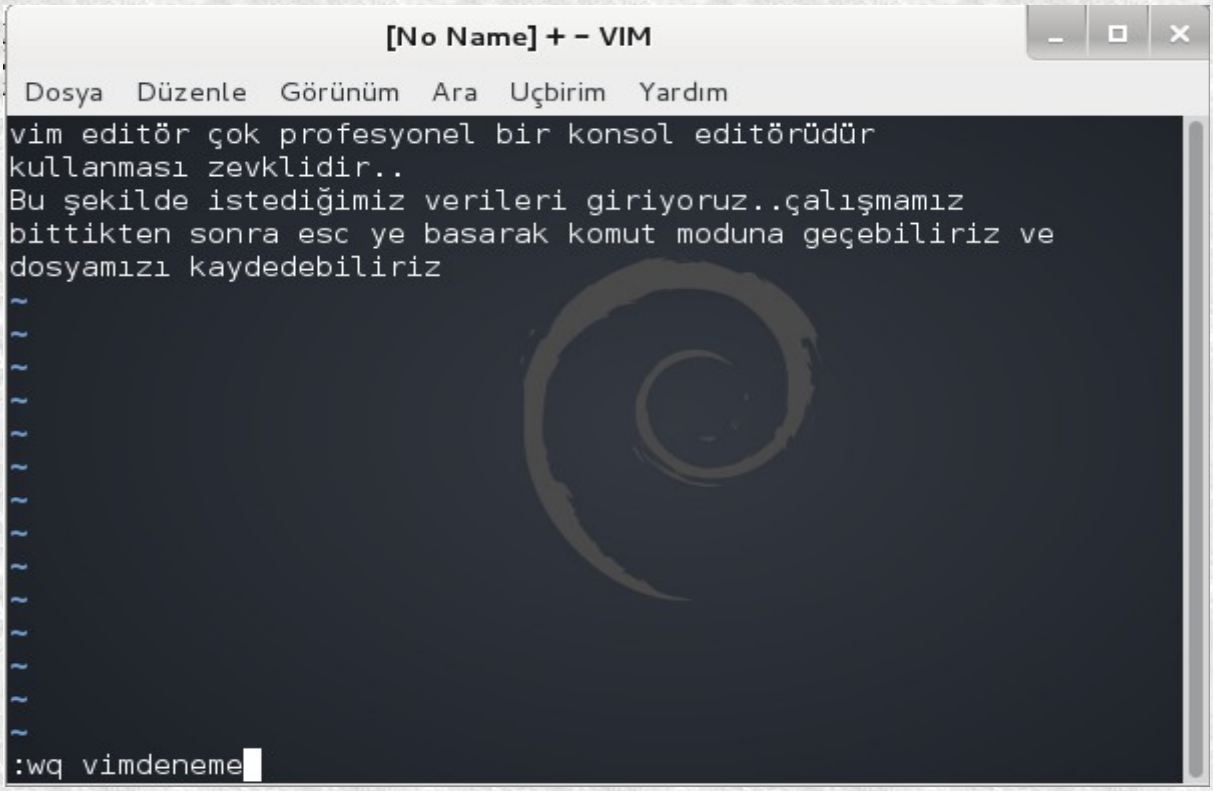
Yazma işlemini bitirdikten sonra **esc** tuşuna basarak **komut moduna** geçebilirsiniz. Siz esc ye basınca aşağıdaki INSERT uyarısı kaybolacaktır. Komut vermek için **:** kullanmalısınız. Verebileceğiniz komutlar:

:q --> vim editörden çıkmak için bu komut verilir, fakat çalışmayı kaydetmediyseniz uyarı verecektir.

:q! --> belgeyi kaydetmeden çıkmak için bu komutu kullanabilirsiniz

:w --> belgeyi kaydetmek için bu komut kullanılır(kaydeder ve vim editörde kalır, konsola dönmez)

:wq --> belgeyi kaydedip çıkmak için bu dosya kullanılır, eğer dosyaya isim vermediyseniz **:wq deneme** komutuyla dosyanızı deneme adıyla kaydederek çıkar.



Örneğin ben burada **:wq vimdeneme** komutuyla belgeyi **vimdeneme** adıyla kaydedip çıktım. Hemen bakalım kaydolmuş mu?

```
root@kemal:~# ls
deneme    deneme.s  Downloads      merhaba.py
deneme.o  Desktop   firefox-flash  vimdeneme
root@kemal:~# cat vimdeneme
vim editör çok profesyonel bir konsol editörüdür
kullanması zevklidir..
Bu şekilde istediğimiz verileri giriyoruz..çalışmamız
bittikten sonra esc ye basarak komut moduna geçebiliriz ve
dosyamızı kaydedebiliriz
root@kemal:~#
```

Evet her şey yolunda görünüyor.

Yukarıda verdiğim komutlar haricinde başka komutlar da var. Bunlar:

ZZ --> belgeyi o anki haliyle kaydet ve çık (esc ye bastıktan sonra :ZZ şeklinde değil direk ZZ şeklinde kullanılır)

:x --> vim den çık, eğer dosya değişmişse kaydet

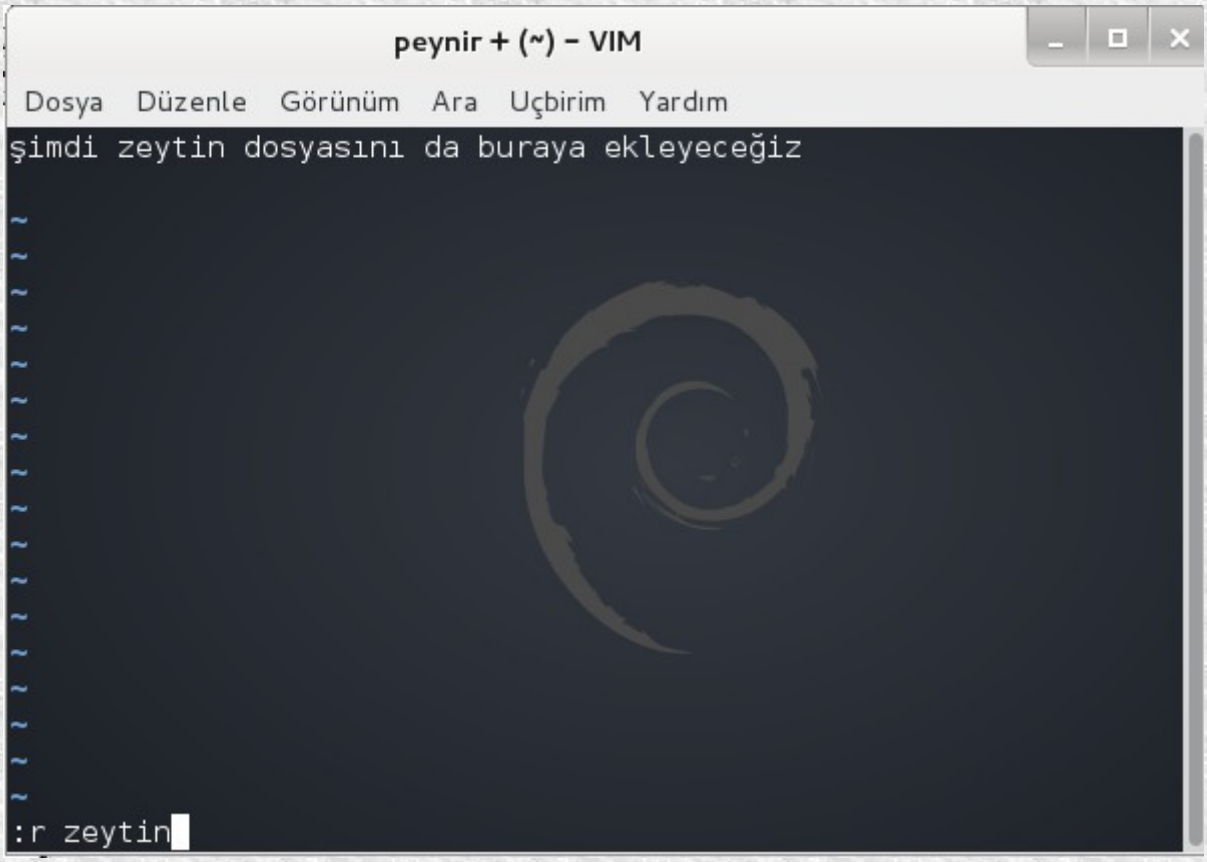
:r dosya --> belirtilen dosyayı oku ve imlecin bulunduğu noktadan başlayarak araya ekle

Son bahsettiğimiz komut (yani **:r dosya**) nasıl çalışır onunla ilgili örnek yapalım. Örneğin **zeytin** adında bir dosyamız olsun. Biz **vim** editörle yeni bir dosya üzerinde çalışmak için **vim peynir** komutuyla dosyamızın adını peynir vererek çalışmaya başlayalım.

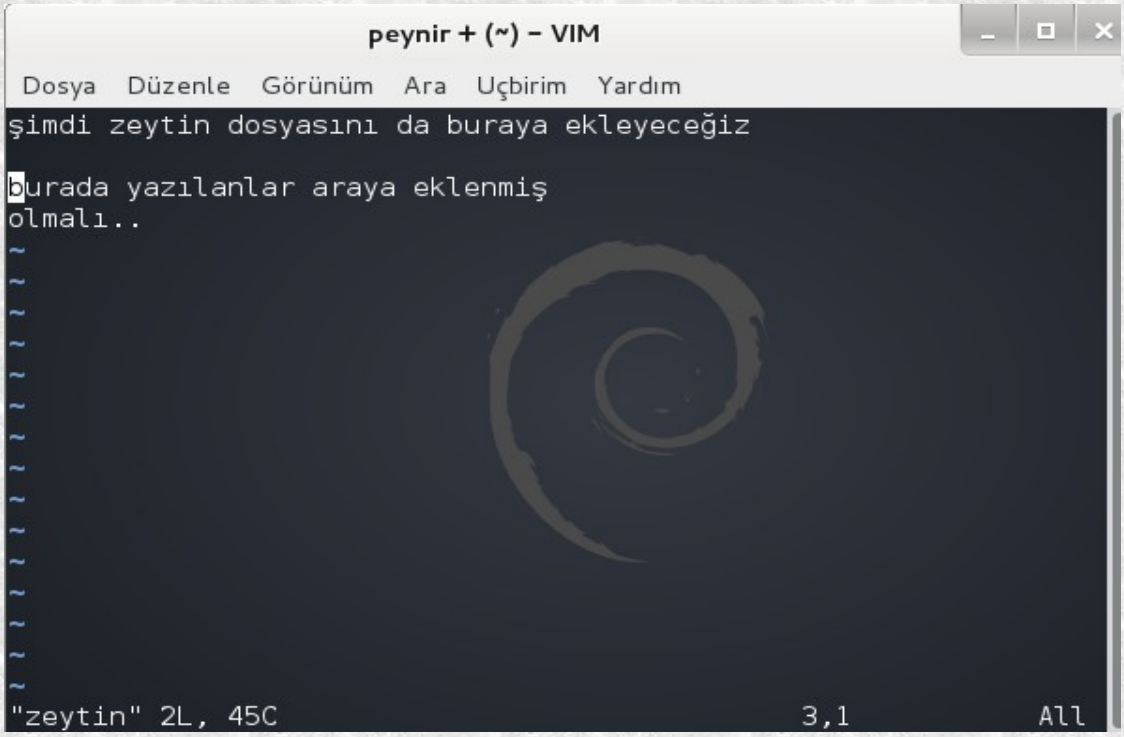
```
root@kemal:~# ls
deneme     deneme.s  Downloads      merhaba.py
deneme.o   Desktop   firefox-flash  zeytin
root@kemal:~# cat zeytin
burada yazılanlar araya eklenmiş
olmalı..
root@kemal:~#
```

Evet **zeytin** dosyamızın içeriği bu. Biz **vim peynir** komutuyla editörümüzü açalım.

```
root@kemal:~# vim peynir
```

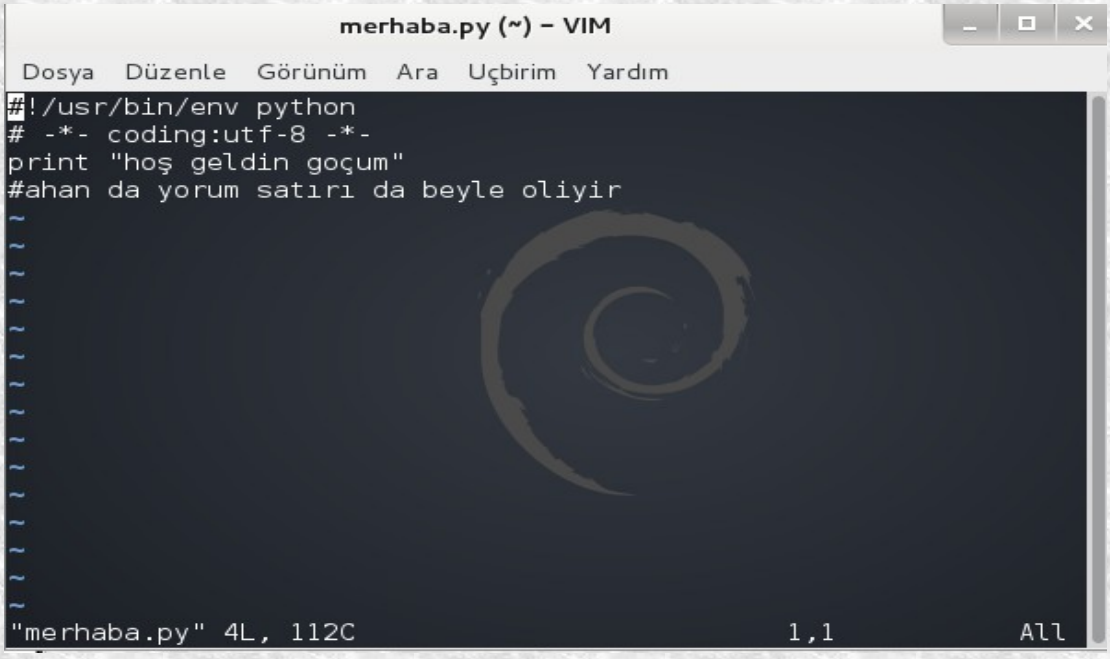


vim peynir komutunu verdikten sonra editörümüz açıldı. Arkasından **i** tuşuna bastık. Sonra çalışmaya başladık. Daha önce oluşturduğumuz **zeytin** dosyasını da bu üzerinde çalıştığımız **peynir** dosyasına eklemeye karar verdik. İlk önce **esc** ye bastık. Sonra **:r zeytin** komutunu verdik. **Enter** a bastıktan sonra üzerinde çalıştığımız peynir dosyasının içeriği aşağıda göründüğü gibi olacaktır.



Gördüğünüz üzere istediğimiz dosyayı o an çalıştığımız dosyaya ekleyebiliyoruz. Tabi **imleç** neredeyse o şekilde ekliyor. Yani araya ekleme yapılabiliyor.

Yeni metin dosyaları oluşturup çalışmak dışında varolan dosyaları da **vim** editörle açıp üzerinde çalışabiliriz. Örneğin bizim meşhur programlamacılık dehası(!) **Python** script dosyamız **merhaba.py** dosyasını vim editörle açmak için **vim merhaba.py** komutunu kullanabiliriz.

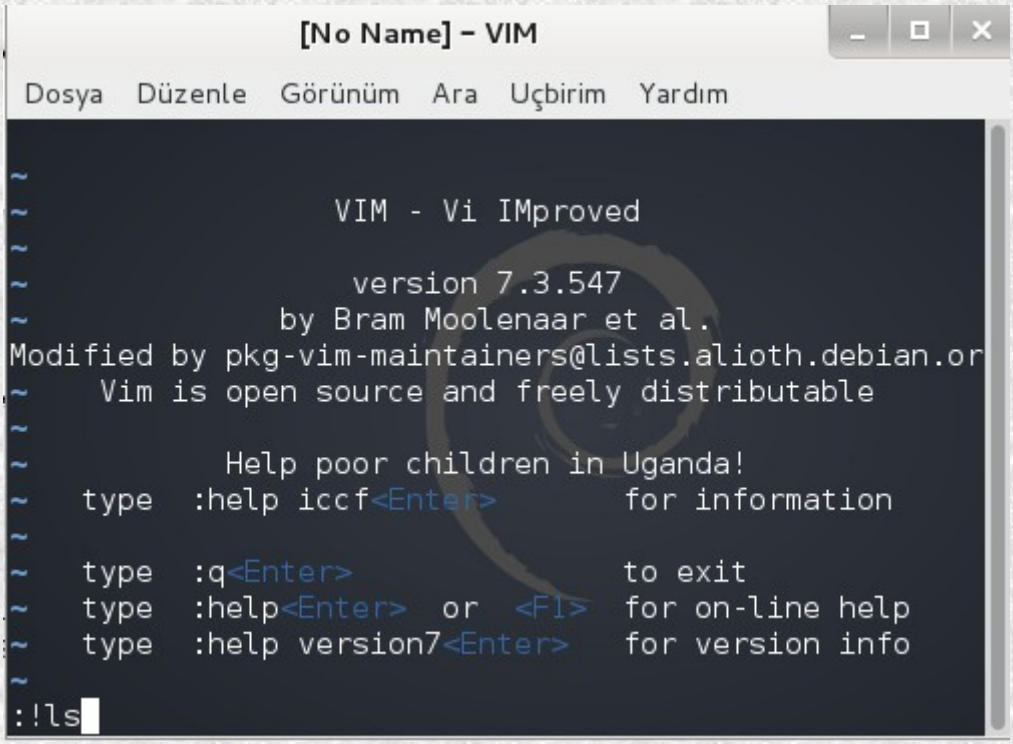


Alttaki açıklamaları incelersek:

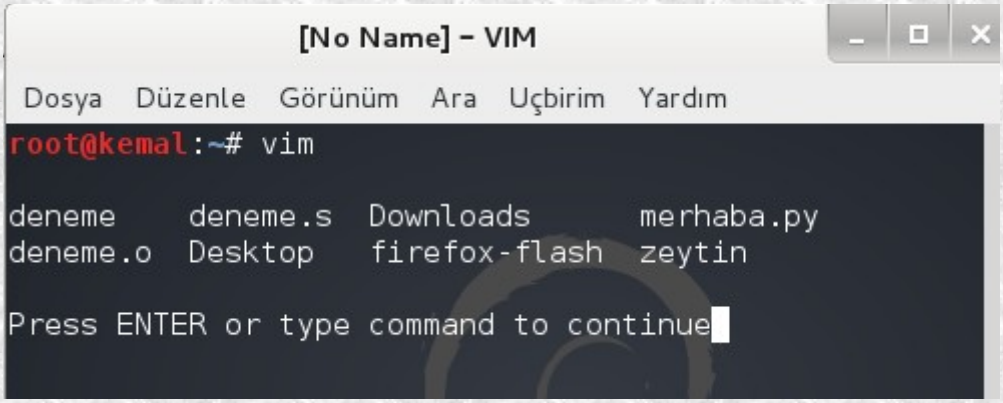
“**merhaba.py**” : üzerinde çalıştığımız dosya adı

4L : 4 satırdan oluşmuş
112C : 112 karakterden oluşmuş

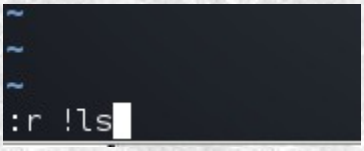
Vim editörle çalışırken Linux **konsol komutlarını** da kullanabiliriz. Örneğin vim editör açıkken **:!ls** komutunu vererek dosyalarımızı listeleyebiliriz.



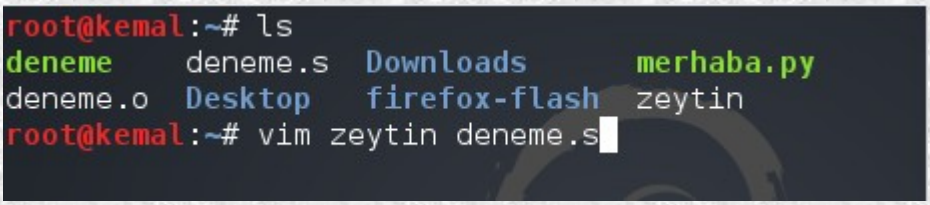
Yukarıda gördüğünüz gibi komutumuzu kullandıktan sonra şöyle bir ekran gelecektir:



Enter tuşuna basarak tekrar vim editöre dönebilirsiniz. İlginç bir özellik daha var. Örneğin :r !ls (ls yerine başka komut da kullanabilirsiniz) komutunu kullandığınızda ls komutunun çıktısı imlecin bulunduğu yere eklenecektir.

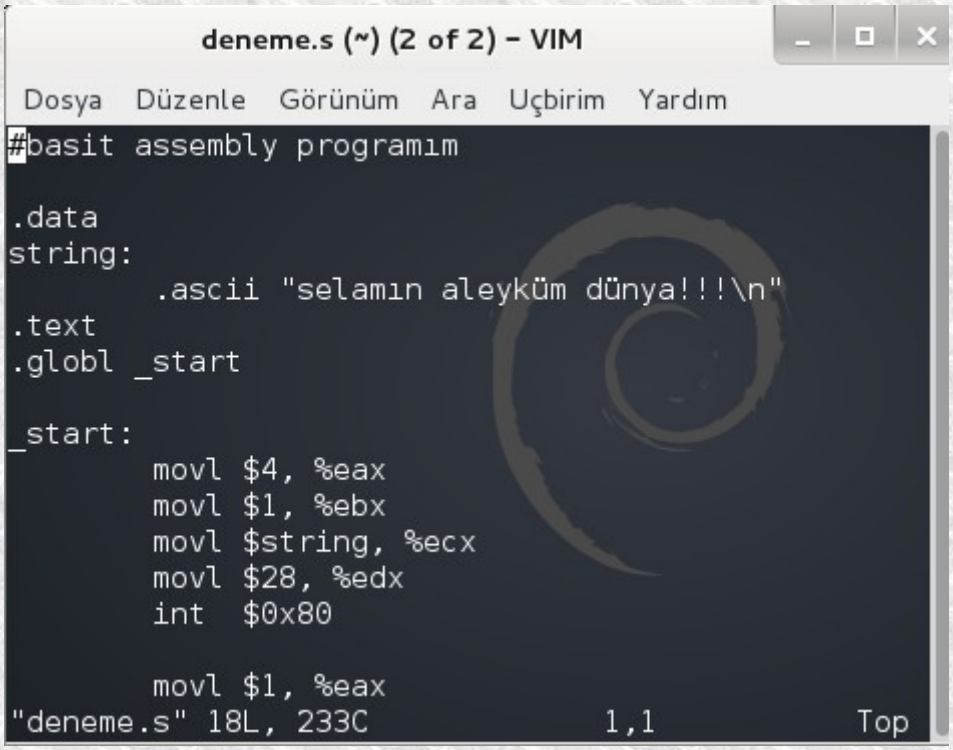


Vim editörde **iki dosyayla** da çalışabilirsiniz. Örneğin **vim <dosya1> <dosya2>** komutunu vererek, ilk önce **dosya1** adlı dosyayla çalışmaya başlarsınız. Bu dosyayla işiniz bittiğinde ya da diğer dosyayla çalışmak istediğinizde **:n** komutu ile diğer dosya üzerinde çalışabilirsiniz.



Yukarıdaki gibi bir komut kulandığınızda vim editör **zeytin** dosyasıyla çalışmaya başlayacak, siz **deneme.s** dosyasında çalışmak isterseniz **:n** komutuyla o dosyaya geçebileceksiniz.

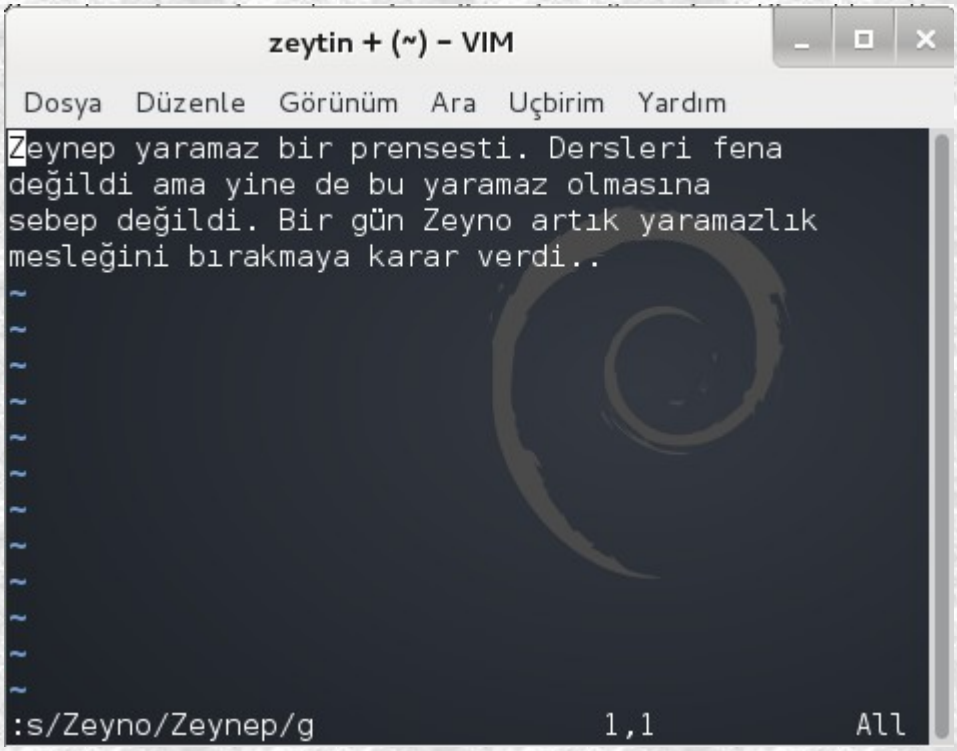
Sol aşağı köşede gördüğünüz şekilde :n komutunu verdikten sonra deneme.s dosyasında çalışmaya başlayacaksınız yani görüntü aşağıdaki gibi olacak.



Tabi birinci dosyada düzenleme işiniz bittikten sonra **:w** ile değişiklikleri kaydedip ondan sonra **:n** ile diğer dosyada çalışmaya başlayabilirsiniz. Böylece iki dosyayı da düzenlemiş olacaksınız.

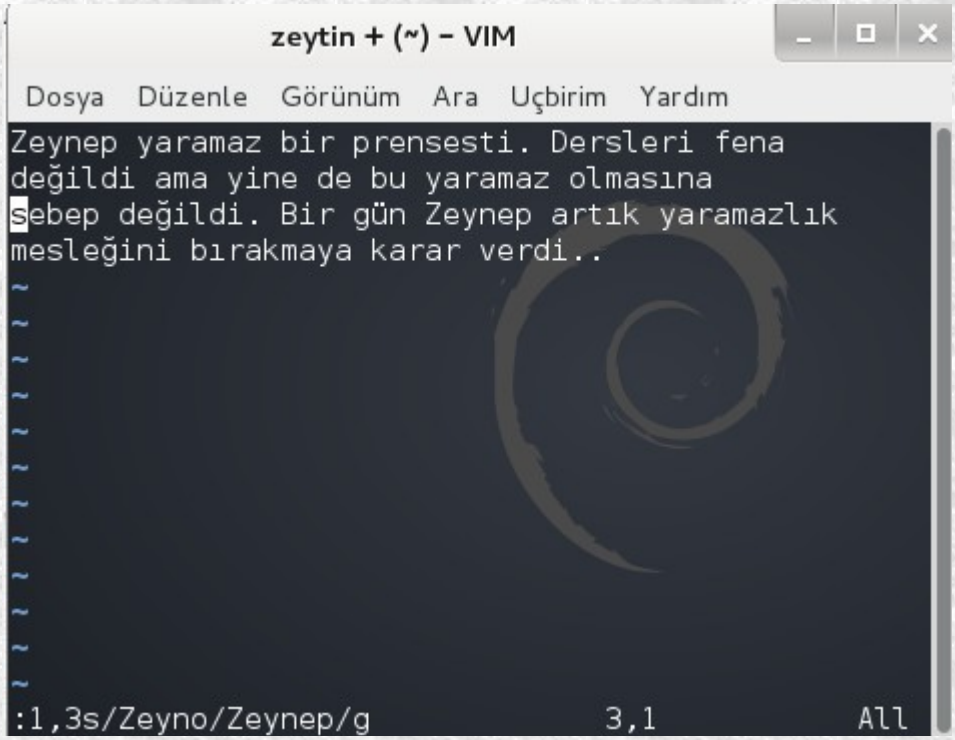
Vim editör, çalıştığınız dosyadaki karakter dizelerini değiştirmenize de imkan verir. Bununla ilgili olarak örneğin **zeytin** adlı metin dosyamızı açalım ve oradaki **Zeyno** ifadelerini **Zeynep** ile değiştirelim. Bunun için **:s/Zeyno/Zeytin/g** komutunu kullanmamız yeterli olacaktır. Bu komut sadece **bir satırdaki** tüm **Zeyno** ifadelerini değiştirecektir.

Komutumuzdan sonraki durum aşağıda göründüğü gibi olacaktır.



Gördüğünüz gibi bir satırdaki **Zeyno** ifadelerini **Zeynep** ile değiştirmiş. Biz hem **birinci** hem de **üçüncü** satırdaki ifadeleri değiştirmek istiyoruz. Bunun için **:1,3s/Zeyno/Zeynep/g** komutunu kullanmalıyız.

Komutumuzu bu şekilde kullandıktan sonra görüntü aşağıdaki gibi olacaktır.

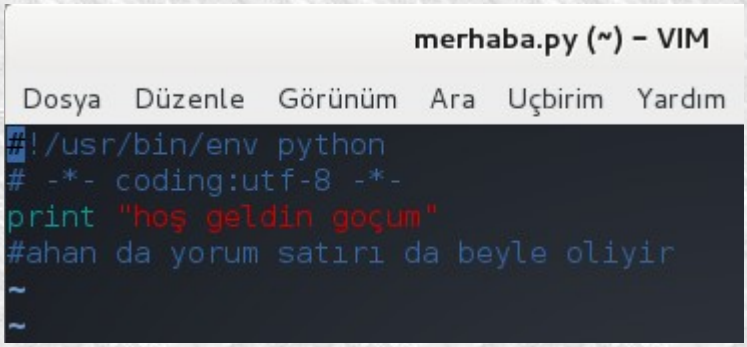


Bu kadar örnek bence yeterli olacaktır. Son olarak **/usr/share/vim/vimrc** dizinindeki **vimrc** ayar dosyasıyla vim editörü özelleştirmeyi görelim. Bu dosyanın içindeki ayarlar varsayılan ayarlardır. Bu ayarları isteğinize göre düzenleyerek vim editörü özelleştirebilirsiniz.

```
root@kemal:~# cat /usr/share/vim/vimrc
" All system-wide defaults are set in $VIMRUNTIME/debian
im (usually just
" /usr/share/vim/vimcurrent/debian.vim) and sourced by t
 call to :runtime
" you can find below.  If you wish to change any of thos
settings, you should
" do it in this file (/etc/vim/vimrc), since debian.vim
ll be overwritten
" everytime an upgrade of the vim packages is performed.
```

Vim editör ayar dosyasının (**vimrc**) içeriği yukarıda görüldüğü gibi. **“** işaretiyle başlayan satırların önündeki bu **“** işaretini kaldırarak istediğiniz ayarların aktif olmasını sağlayabilirsiniz. Örneğin **nano** editörle açıp isteğinize göre düzenleme yapabilirsiniz. Nano editörle açmak için **nano /usr/share/vim/vimrc** komutunu kullanabilirsiniz. İstediğiniz değişiklikleri yaptıktan sonra da **ctrl+X** ile onaylayıp kaydedebilirsiniz.

Örneğin **syntax on** önündeki **"** işaretini kaldırırsanız **karakter renklendirmesi** yapacaktır. Bu durumda daha önce vim ile açtığımız **merhaba.py** dosyasını açtığımızda artık şöyle görünecektir:



Siz de oradaki açıklamaları inceleyip istediğiniz düzenlemeleri yapabilirsiniz.

Network Komutları

Linux'ta kulanılan ağ komutlarıyla ilgili çok ayrıntıya girmeden küçük bazı örnekler vereceğim. Hemen başlayalım. Windows sistemlerde ip yapılandırmasıyla ilgili bilgiyi görmek için ipconfig komutu kullanıyorduk. Linux sistemlerde bunun için **ifconfig** komutunu kullanıyoruz.

```
root@kemal:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:60:b2:71:77
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255
.255.0
          inet6 addr: fe80::215:60ff:feb2:7177/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1910 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4849647 (4.6 MiB)  TX bytes:135612 (132.4 KiB)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:480 (480.0 B)  TX bytes:480 (480.0 B)

root@kemal:~#
```

Yukarıdaki görüntülenen bilgilerden **eth0** ifadesi bizim **ethetnet** kartımızı (**ağ arabirim kartımızı**) göstermektedir. Eğer sistemde başka ethernet kartları da varsa onlar da sırasıyla **eth1**, **eth2** şeklinde adlandırılacaktır. Diğer bilgiler zaten malum.

Eğer sadece ethernet kartı ile ilgili olan bilgiyi yani basit ağ yapılandırmasını görmek isterseniz **ifconfig eth0** komutunu kullanmalısınız.

```
root@kemal:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:15:60:b2:71:77
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255
.255.0
          inet6 addr: fe80::215:60ff:feb2:7177/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3298 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1910 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4849711 (4.6 MiB)  TX bytes:135612 (132.4 KiB)
          Interrupt:16

root@kemal:~#
```

Kablosuz ağ yapılandırmasıyla ilgiyi bilgiyi görüntülemek için de **iwconfig** komutu kullanılır.

```
root@kemal:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

root@kemal:~#
```

Eğer **IP** adresimizi elle yapılandırmak istersek, **ifconfig eth0 <ip_adresi> <alt_ağ_maskesi>** şeklinde bir komut kullanmalıyız. Aşağıda bir örnek var.

```
root@kemal:~# ifconfig eth0 192.168.1.200 netmask 255.255.255.0
root@kemal:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:15:60:b2:71:77
          inet addr:192.168.1.200  Bcast:192.168.1.255  Mask:255.255
.255.0
```

Yukarıdaki komutla **IP** adresimizi **192.168.1.200** yaptık.

Ping işlemi için yani bir adresin aktif olup olmadığını anlamaya yarayan sorgu paketleri gönderme işlemi için de **ping** komutu kullanılır. Bu komut parametresiz kullanıldığı zaman siz **ctrl+C** ile komutu kesene kadar hedef sisteme paket göndermeye devam eder. Bunun önlemek için **c** parametresi kullanılır. Örneğin **ping -c 3 <hedef>** komutu ile hedef adrese 3 sadece 3 sorgu paketi gönderir.

```
root@kemal:~# ping -c 3 facebook.com
PING facebook.com (173.252.110.27) 56(84) bytes of data.
64 bytes from edge-star-shv-13-frc1.facebook.com (173.252.110.27): i
cmp_req=1 ttl=75 time=166 ms
64 bytes from edge-star-shv-13-frc1.facebook.com (173.252.110.27): i
cmp_req=2 ttl=75 time=167 ms
64 bytes from edge-star-shv-13-frc1.facebook.com (173.252.110.27): i
cmp_req=3 ttl=75 time=167 ms

--- facebook.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 166.505/166.925/167.149/0.297 ms
root@kemal:~#
```

Yine Windows sistemlerde kullanılan ve paketin hedefe ulaşana kadar geçtiği hostları/yönlendiricileri görmemizi sağlayan **tracert** komutunun Linux sistemdeki karşılığı **traceroute** komutudur. Aşağıdaki örnekte olduğu gibi kulanılır.

```
root@kemal:~# traceroute facebook.com
traceroute to facebook.com (173.252.110.27), 30 hops max, 60 byte pa
ckets
 1  192.168.1.1 (192.168.1.1)  10.596 ms  10.568 ms  10.617 ms
 2  10.0.0.37 (10.0.0.37)  11.622 ms  12.336 ms  12.533 ms
 3  93.155.0.132 (93.155.0.132)  12.720 ms  14.970 ms  15.333 ms
 4  * * *
 5  195.175.168.100.static.turktelekom.com.tr (195.175.168.100)  22.
920 ms  29.523 ms  31.132 ms
 6  81.212.197.63.static.turktelekom.com.tr (81.212.197.63)  29.154
ms  7.586 ms  9.675 ms
 7  212.156.140.25.static.turktelekom.com.tr (212.156.140.25)  55.80
```

Bir sonraki komutumuz da **netstat** komutu. Bu komut ağ bağlantıları, yönlendirme tablosu vs. bilgisi almak için kullanılan bir komuttur ve farklı parametrelerle kullanılabilir fakat en temel kullanım amacı sistemdeki bağlantılarla ilgili bilgi almaktır. Ben **netstat -antp** şeklinde kullanımı tercih ettim.

```
root@kemal:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 127.0.0.1:8123          0.0.0.0:*
LISTEN      2315/polipo
tcp        0      0 192.168.1.106:45358     173.194.39.211:443
ESTABLISHED 4859/chromium --pas
tcp      295      0 192.168.1.106:47309     173.194.39.223:443
ESTABLISHED 4859/chromium --pas
tcp        0      0 192.168.1.106:46541     173.194.39.207:443
ESTABLISHED 4859/chromium --pas
tcp        0      0 192.168.1.106:50250     173.194.39.197:443
ESTABLISHED 4859/chromium --pas
tcp        0      0 192.168.1.106:58904     173.194.39.223:80
```

Şimdi de domain adları ve IP adresleriyle ilgili sorgu yapmamıza yarayan **nslookup** komutuna bakalım.

```
root@kemal:~# nslookup
> www.turkcell.com.tr
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   www.turkcell.com.tr
Address: 86.108.161.36
> www.facebook.com
Server:         8.8.8.8
Address:        8.8.8.8#53
```

Hedef sistemle ilgili **whois** bilgisi almak için de **whois** aracından faydalanabiliriz. Eğer kullandığınız dağıtımda bu araç yoksa **apt-get install whois** (**rpm** tabanlı sistemler için **yum install whois**) komutuyla sisteminize kurarak kullanabilirsiniz.

```
root@kemal:~# whois turkcell.com.tr
** Registrant:
   Turkcell Ileti@im Hizmetleri A.@.
   Turkcell Plaza Me@rutiyet Cad. No:153 Tepeba@
   34430
   @stanbul,
     T@rkiye
   hostmaster@turkcell.com.tr
   + 90-212-3131000-
   + 90-212-3131010


** Administrative Contact:
NIC Handle              : tih36-metu
Organization Name       : Turkcell Iletisim Hizmetleri A.S.
Address                 : Turkcell Maltepe Plaza
                          Yenimahalle Pamukkale Sok. No: 3 Soganlik
- Kartal
                          @stanbul,34880
                          T@rkiye
```

Yine ayrıntılı bilgi alabileceğimiz araçlardan bir tanesi de **dig** aracıdır.

```
root@kemal:~# dig www.turkcell.com.tr

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> www.turkcell.com.tr
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21047
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.turkcell.com.tr.           IN      A

;; ANSWER SECTION:
www.turkcell.com.tr.    44      IN      A       86.108.161.36

;; Query time: 63 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Sep  5 09:45:48 2013
;; MSG SIZE  rcvd: 53

root@kemal:~#
```

Sorguladığımız hedef sistemle ilgili bilgi alma araçlarından bir tanesi de **host** aracıdır. Komut **host -a <hedef>** şeklinde **a** paremetresiyle kullanılırsa aşağıdaki gibi bir çıktı alınacaktır.

```
root@kemal:~# host -a www.cyber-warrior.org
Trying "www.cyber-warrior.org"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19580
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.cyber-warrior.org.         IN      ANY

;; ANSWER SECTION:
www.cyber-warrior.org.  283     IN      A       141.101.120.121
www.cyber-warrior.org.  283     IN      A       141.101.120.120
www.cyber-warrior.org.  283     IN      CNAME   cyber-warrior.org.

Received 85 bytes from 8.8.8.8#53 in 61 ms
root@kemal:~#
```

Eğer komutu **C** parametresiyle kullanırsanız bu sefer çıktı şöyle olacaktır:

```
root@kemal:~# host -C www.cyber-warrior.org
Nameserver 173.245.58.105:
        cyber-warrior.org has SOA record dana.ns.cloudflare.com. dns
.cloudflare.com. 2013081709 10000 2400 604800 3600
Nameserver 173.245.59.148:
        cyber-warrior.org has SOA record dana.ns.cloudflare.com. dns
.cloudflare.com. 2013081709 10000 2400 604800 3600
;; connection timed out; no servers could be reached
root@kemal:~#
```

Her şeyi devletten beklemeyin :) parametreleri neye göre kullanıyoruz öğrenmek için **host** komutuyla bakın.

```
root@kemal:~# host
Usage: host [-aCdlriTwv] [-c class] [-N ndots] [-t type] [-W
            [-R number] [-m flag] hostname [server]
       -a is equivalent to -v -t ANY
       -c specifies query class for non-IN data
       -C compares SOA records on authoritative nameservers
```

Yönlendirme tablosuyla ilgili bilgi için de **route** komutunu kullanabilirsiniz.

```
root@kemal:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
Use Iface
default         192.168.1.1     0.0.0.0         UG    0      0
  0 eth0
192.168.1.0     *               255.255.255.0   U     0      0
  0 eth0
root@kemal:~#
```

Evet şimdi de sırada **arp** komutumuz var. Bu komutla **arp tablosu** görülür. Yani ağdaki hostların **ip-mac** adresleri incelenebilir.

```
root@kemal:~# arp -a
? (192.168.1.1) at 00:27:19:c2:30:86 [ether] on eth0
```

Sistemimize yapılan bağlantıları ve sistemimizden dışarıya yapılan bağlantıları canlı olarak izlemek için **tcpdump** komutunu kullanabiliriz. Bu komutu kullandıktan sonra bağlantı istekleri dinlenmeye başlanacak ve istekler konsolda görüntülenecektir. Yani ağdaki paketleri yakalayarak görüntüler. Ayrıca daha önceden kaydedilmiş paketler üzerinde de inceleme yapabilir. Adres çözümleme yapmadan paketleri yakalaması için **n** parametresiyle kullanım örneği aşağıda görülmektedir.

```
root@kemal:~# tcpdump -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 b
ytes
10:15:00.104785 IP 192.168.1.106.54907 > 173.194.39.216.443: Flags [
.], ack 1258679726, win 2133, options [nop,nop,TS val 690944 ecr 625
946286], length 0
10:15:00.134785 IP 173.194.39.216.443 > 192.168.1.106.54907: Flags [
.], ack 1, win 661, options [nop,nop,TS val 625991418 ecr 645839], l
ength 0
10:15:05.112766 ARP, Request who-has 192.168.1.1 tell 192.168.1.106,
 length 28
```

Sistemde tek ağ kartı varsa **tcpdump** komutuyla **tcpdump -i eth0** aynı anlama gelir. Eğer **eth1** adında ikinci bir ağ kartı varsa onunla ilgili bağlantıları dinlemek için **tcpdump -i eth1** komutu kullanılmalıdır.

Linux'ta önemli ve pratik bir indirme aracı olan **wget** kullanımından da bahsedelim. En sade haliyle **wget <indirilecek_dosya_adresi>** şeklinde bir komut kullanmak yeterlidir.

```
root@kemal:~# wget http://nmap.org/dist-old/nmap-1.51.tar.gz
--2013-09-05 10:08:40--  http://nmap.org/dist-old/nmap-1.51.tar.gz
nmap.org (nmap.org) çözümleniyor... 173.255.243.189, 2600:3c01::f03c
:91ff:fe70:d085
nmap.org (nmap.org)[173.255.243.189]:80 bağlanılıyor... bağlantı kur
uldu.
HTTP isteği gönderildi, yanıt bekleniyor... 200 OK
Uzunluk: 219371 (214K) [application/x-gzip]
Saving to: `nmap-1.51.tar.gz'

100%[==========================>] 219.371      231K/s   in 0,9s

2013-09-05 10:08:42 (231 KB/s) - `nmap-1.51.tar.gz' saved [219371/21
9371]

root@kemal:~#
```

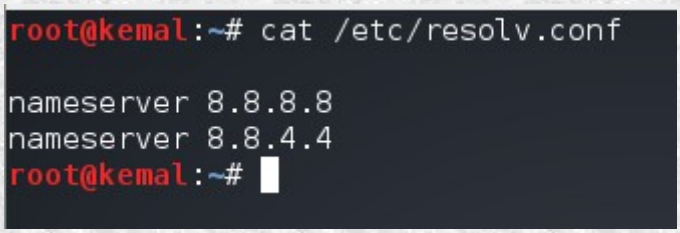
Yukarıdaki örnekte nmap.org adresinden bir dosya indirdik ve çalışma dizinimize kaydoldu.

```
root@kemal:~# ls
deneme    Desktop              firefox-flash
deneme.o  Downloads            merhaba.py
deneme.s  explore-2013-09-04   nmap-1.51.tar.gz
root@kemal:~#
```

Wget aracıyla isterseniz bir sitenin tamamını da indirebilirsiniz. Bunun için **wget -nrp <indirilecek_site_adresi>**

komutunu kullanmanız yeterlidir. Örneğin **falansite.com** sitesinin tamamını indirmek istiyorsanız **wget -nrp www.falansite.com** komutunu kullanmanız gerekir.

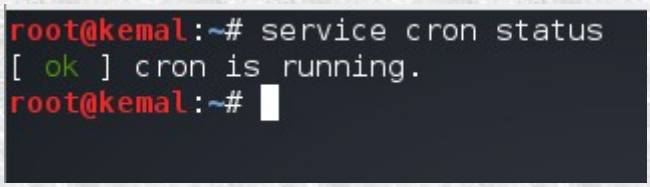
Şimdi de Linux sistemlerde **dns** ayarlarının nasıl yapılacağına bakalım. Dns ile ilgili kayıtlar **/etc/resolv.conf** dosyasında tutulur.



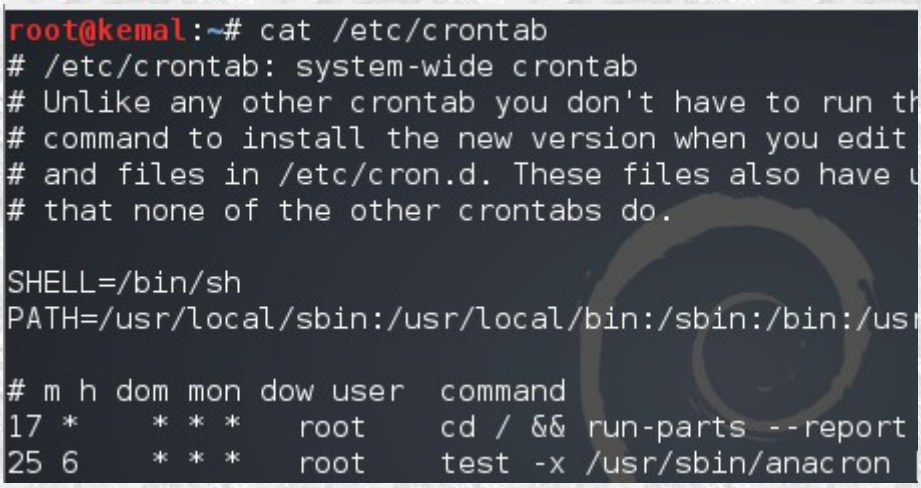
Bu dosyada gördüğünüz **nameserver** yazan yerlerin karşısına istediğiniz **dns** adreslerini yazarak kaydedin. Bundan sonra adres çözümlemesi sizin yazdığınız dns ler üzerinden yapılacaktır. Sistem yeniden başladığında ya da network servisi restart edildiğinde ayarların kaybolmasını istemiyorsanız daha önce de bahsetmiş olduğumuz **chattr** komutuyla **resolv.conf** dosyasını değiştirilemez yapabilirsiniz. Bunun için **chattr +i /etc/resolv.conf** komutunu kullanmalısınız. Yukarıda bahsedilen dns değişikliğini kısa yoldan **nano** editörle yapabilirsiniz. Yani **nano /etc/resolv.conf** komutunu vererek resolv.conf dosyasını açar ve istediğiniz değişiklikleri yaptıktan sonra **ctrl+X** ile çıkarsınız. Çıkmadan size değişiklikleri onaylıyor musunuz diye soracaktır. E ya da Y dersiniz :)

Zamanlanmış Görevler (Cron Servisi)

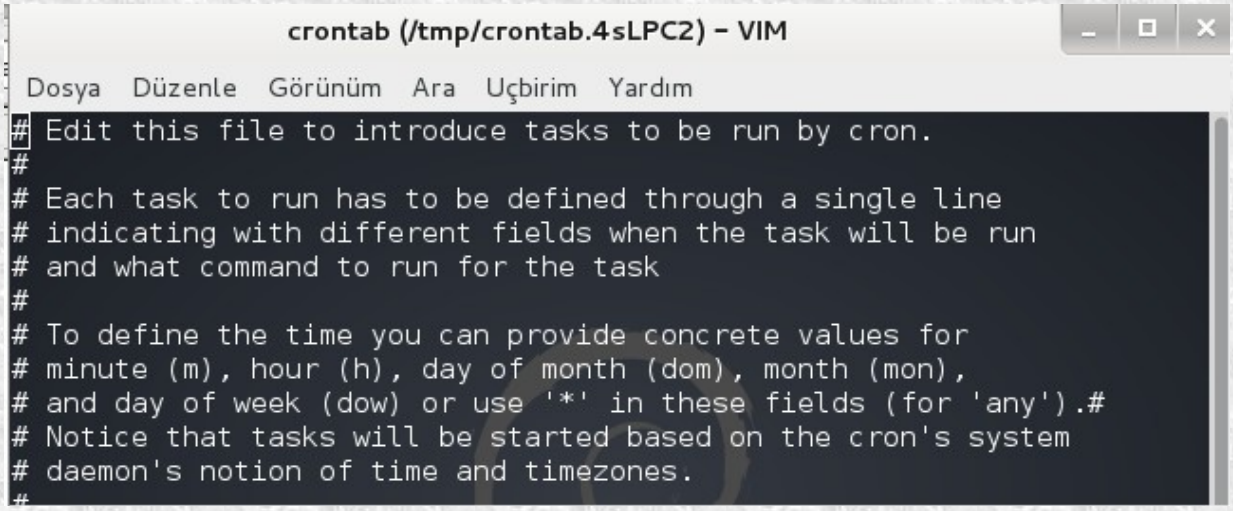
Linux sistemde bazı işlerin istediğiniz zamanlarda ya da zaman aralıklarında otomatik olarak yapılmasını sağlayabilirsiniz. Yani zamanlanmış görevler dediğimiz kavram. Bu işi **cron** servisi yapar.



Zamanlanmış işleri düzenlemek için **crontab** komutu kullanılır. Yani crontab komutuyla **cron** dosyaları düzenlenebilir. Cron uygulaması konfigürasyon dosyası **/etc/crontab** dosyasıdır.



Bu kısa bilgilerden sonra şimdi gelelim komutun kullanılması ve düzenlemelerin nasıl yapılacağına. Düzenleme yapmak için **crontab** komutu kullanıldığında **vim** editörle ilgili cron doyası açılır (yani hangi kullanıcı komutu verdiyse onunla ilgili olan cron dosyası) ve değişiklikler yapılıp editörden çıkıldığı anda cron servisi yapılan ayarlarla çalışmaya başlar. Düzenleme yapmak için **crontab -e** komutu vermemiz gerekir.



Açılan dosyaya zamanlanmış görevlerle ilgili bilgiler yazılır. Yani hangi tarihte ne yapılmasını istiyorsak ona göre ayarlamalar yapılır.

Düzenleme yapabilmek için öncelikle cron dosyalarının özelliklerini öğrenmemiz gerek. Cron dosyasında 6 tane bölüm vardır. İlk beş tanesi, yapılması istenen işle ilgili gün,saat, dakika gibi bilgiler için, son bölüm ise çalıtırılacak komut içindir.

1.bölüm : İşin seçilen saat başından kaç dakika sonra başlatılacağını belirler. (0-59)(örneğin 30 demek, saat başını 30 dk. Geçe demektir.

2.bölüm : İşin hangi saatte başlatılacağını belirler. (0-23) (sayı yerine * girilirse her saat anlamına gelir)

3.bölüm : İşin hangi günler başlatılacağını belirler. (1-31) (sayı yerine * girilirse her gün anşamına gelir. 7,14 ifadesi de

her ayın 7 si ve 14 ü anlamına gelir)

4.bölüm : İşin hangi aylarda yapılacağını belirler. (1-12) (sayı yerine * girilirse her ay anlamına gelir. 1,6 ifadesi Ocak ve Haziran demektir. 1-3 ifadesi de Ocak-Şubat-Mart demektir)

5.bölüm : İşin haftanın hangi günlerinde yapılacağını belirler. (0-7)(sayı yerine * girilirse her gün demektir. 0. ve 7. günler Pazar kabul edilir. 1,2 ifadesi P.tesi ve Salı anlamına, 1-3 ise P.tesi-Salı-Çarşamba anlamına gelir)

Yapılmasını istediğimiz işlerle ilgili satırları **crontab -e** ile açtığımız **crontab** dosyasının en altına ekleriz. Kaydedip çıktıktan sonra zamanlanmış görevimiz çalışmaya başlayacaktır. Basit bir örnek yapalım. Örneğin **0 19 * * * rm -rf /tmp** ifadesini, yani her gün saat 19:00 da /tmp dizininin silinmesini söyleyen ifadeyi crontab dosyasının en alt satırına aşağıda görüldüğü şekilde ekleyelim.

```
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups
#
# For more information see the ma
#
# m h  dom mon dow   command
0 19 * * * rm -rf /tmp
-- INSERT --
```

En alt satıra dikkat edin. Bu şekilde yazıyoruz. Sonra **esc** ye basıyoruz ve **:w** komutuyla kaydediyoruz. Çıkmak için de hatırların **:q** komutunu kullanmanız gerekiyor. Artık zamanlanmış görevimiz çalışmaya başladı. Zamanı geldiğinde kontrol edin, görevin çalıştığını göreceksiniz.

```
Prş Eyl 5, 19:02
root@kemal: ~
Dosya  Düzenle  Görünüm  Ara  Uçbirim  Yardım
root@kemal:~# cd /tmp
bash: cd: /tmp: Böyle bir dosya ya da dizin yok
root@kemal:~#
```

Zamanlanmış görevleri görmek için **crontab -l** komutu kullanılır.

```
#
# m h  dom mon dow   command
0 19 * * * rm -rf /tmp
root@kemal:~#
```

Görevleri silmek için de **crontab -r** komutunu kullanabiliriz.

```
root@kemal:~# crontab -r
root@kemal:~# crontab -l
no crontab for root
root@kemal:~#
```

Böylece bir konunun daha sonuna gelmiş olduk. Yeni konumuz Linux log dosyaları ile ilgili olacak.

Linux Log (Kayıt) Dosyaları

Log dosyaları bilindiği gibi sistemde neler yapıldığıyla ilgili bilgiler içeren kayıtlardır. Linux işletim sisteminde log dosyaları **/var/log** dizininin altında bulunur.

```
root@kemal:/var/log# ls
alternatives.log  dmesg.4.gz        mail.info          privoxy
apache2           dpkg.log          mail.log           pycentral.log
apt               exim4             mail.warn          samba
auth.log          faillog           messages           stunnel4
bootstrap.log     fontconfig.log    mysql              syslog
btmp              fsck              mysql.err          tor
chkrootkit        gdm3              mysql.log          user.log
ConsoleKit        guymager.log      news               wtmp
daemon.log        installer         nginx              Xorg.0.log
debug             kern.log          ntpstats           Xorg.0.log.old
dmesg             lastlog           openvas            Xorg.1.log
dmesg.0           lpr.log           pm-powersave.log   Xorg.1.log.old
dmesg.1.gz        lynis.log         pm-suspend.log     Xorg.2.log
dmesg.2.gz        lynis-report.dat  polipo
dmesg.3.gz        mail.err          postgresql
root@kemal:/var/log#
```

Gördüğünüz gibi log dosyaları bazı kategorilere ayrılmış. Dosya isimlerinden zaten hangi servisle ya da neyle ilgili oldukları anlaşılıyor. Ben temel konulardan bahsedeceğim. Hepsini tek tek anlatmaya gerek yok zannederim.

Örneğin **auth.log** dosyasında kullanıcılarla ilgili kayıtlar bulunmaktadır. Sisteme giriş tarihleri, oturum bilgileri vs. gibi bilgiler bu dosyada tutulmaktadır.

```
root@kemal:/var/log# cat auth.log | more
Mar 13 19:37:28 kemal gdm-welcome][2341]: pam_unix(
): session opened for user Debian-gdm by (uid=0)
Mar 13 19:37:28 kemal gdm-welcome][2341]: pam_ck_co
:session): nox11 mode, ignoring PAM_TTY :0
Mar 13 19:37:44 kemal polkitd(authority=local): Reg
```

Kullanıcıların süreçleriyle ilgili kayıtlar da **userlog** dosyasında bulunur.

```
root@kemal:/var/log# cat user.log | more
Mar 13 19:48:27 kemal mtp-probe: checking bus 1, de
es/pci0000:00/0000:00:1d.7/usb1/1-1"
Mar 13 19:48:27 kemal mtp-probe: bus: 1, device: 2
ce
Mar 13 20:05:58 kemal mtp-probe: checking bus 1, de
es/pci0000:00/0000:00:1d.7/usb1/1-1"
Mar 13 20:05:58 kemal mtp-probe: bus: 1, device: 3
ce
Mar 13 20:15:59 kemal mtp-probe: checking bus 1, de
es/pci0000:00/0000:00:1d.7/usb1/1-1"
Mar 13 20:15:59 kemal mtp-probe: bus: 1, device: 4
ce
Mar 13 21:19:07 kemal shutdown[6770]: shutting down
Mar 13 21:23:17 kemal shutdown[2917]: shutting down
Mar 13 21:53:00 kemal shutdown[3300]: shutting down
```

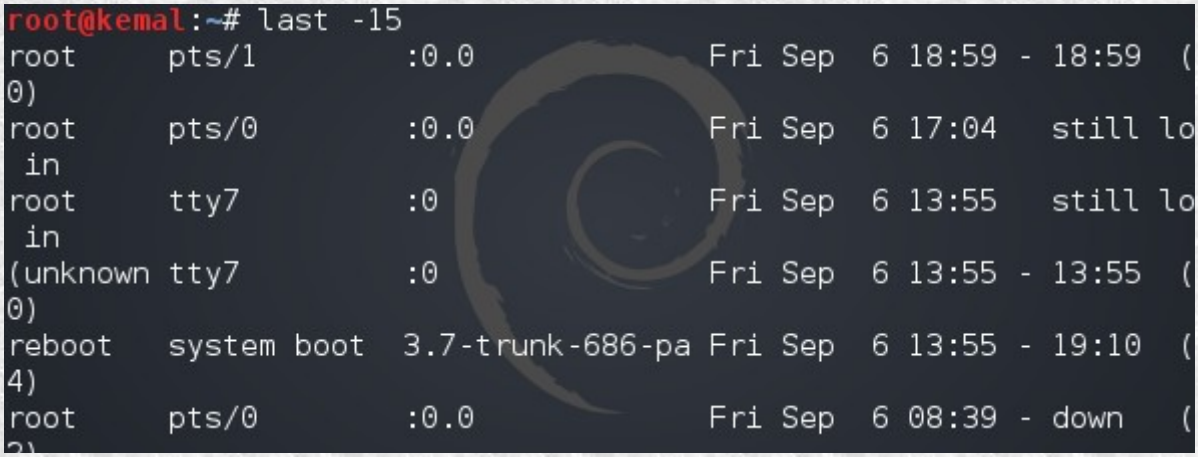
Mesela sistemle ilgili bir çok olayın kaydının tutulduğu dosya **messages** dosyasıdır. Bu dosyaya yapılan kayıtları konsoldan canlı olarak izlemek isterseniz **tail -f /var/log/messages** komutunu kullanmalısınız.

```
root@kemal:~# tail -f /var/log/messages
Sep  6 16:33:31 kemal mtp-probe: checking bus 1, devic
es/pci0000:00/0000:00:1d.7/usb1/1-2"
Sep  6 16:33:31 kemal mtp-probe: bus: 1, device: 3 was
ce
Sep  6 16:33:32 kemal kernel: [ 9519.889024] scsi 3:0:
ss     SanDisk  Cruzer           1.01 PQ: 0 ANSI: 2
Sep  6 16:33:32 kemal kernel: [ 9519.890425] sd 3:0:0:
 generic sg2 type 0
```

Bir başka log dosyası olan **dmesg** de ise sistem açılırken gerçekleşen olayların kaydı tutulur. Dmesg kayıtlarını görmek için konsoldan **dmesg | more** komutunu da kullanabilirsiniz.

```
root@kemal:~# dmesg | more
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 3.7-trunk-686-pae (debia
bian.org) (gcc version 4.7.2 (Debian 4.7.2-5) ) #1 SM
kali8
[    0.000000] Disabled fast string operations
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x0
```

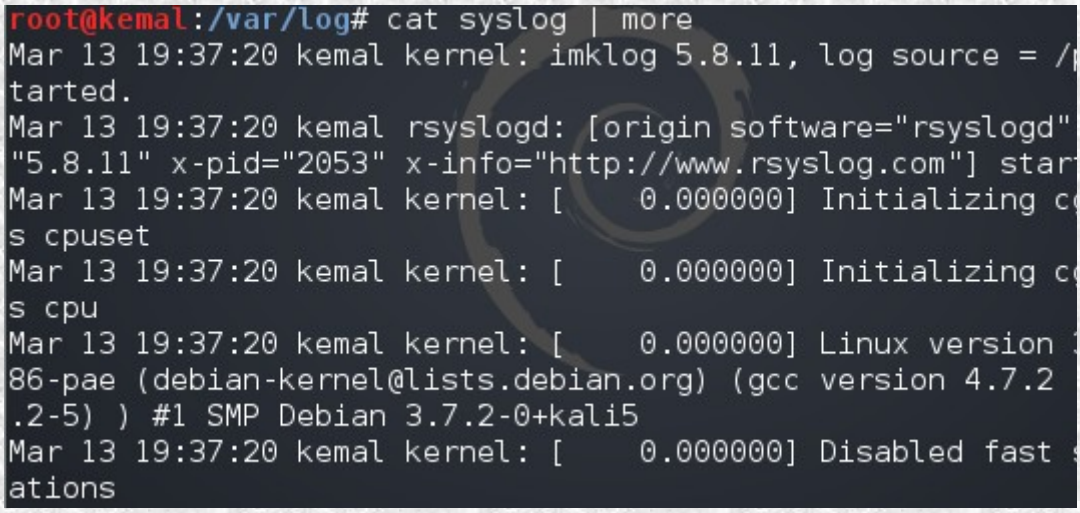
Sistemde en son oturum açan kişileri listelemek için **last** komutunu kullanabiliriz. Örneğin son 15 kullanıcı için **last -15** diyebiliriz.



Örneğin apache sunucu servisinin kayıtları bizim gibi kişisel kullanıcılar için olmasa da sistemini bir sunucu olarak kullananlar için önemlidir. Apache ile ilgili log dosyaları **/var/log/apache2/** altında bulunur. Bu dosyalardan **access.log** dosaysını inceleyerek kimler sunucumuza bağlantı yapmış görelim :)

Maalesef benden başka yüzüne bakan olmamış.

Sistemle ilgili kayıtların tutulduğu bir başka dosya da syslog dosyasıdır. Son olarak bu dosyayı da inceleyelim ve bu konuyu burada bitirelim.



Kaynaklar:

-Kim Korkar Linux'tan?
-Linux-101-Hacks
-İntro. Command Line
-Linux Pocket Guide
-Unix and Linux
-Shellintro
-Hacking Vim 7.2