

LİNX NOTLARI

Echo

İlk olarak, yazdırma yapmak istediğimiz metin için “**echo**” komutunu kullanıyoruz.

Clear

“**clear**” komutu ile mevcut terminaldeki tüm ekran temizlenir.

Calendar

Takvim seçeneğini çalıştırmak için “**cal**” komutu çalıştırılır.
(Paket yükleme daha sonradan ele alınacak)

```
sudo apt-get install ncal -y
```

Burada eğer takvimi yıl olarak almak istersek aşağıdaki şekilde alabiliriz.

```
cal -y
```

2018 yılı için takvimi getirmek istersek

```
cal 2018
```

Mevcut tarihi almak için “**date**” komutu kullanılır.

History

“**history**” komutu ile geçmişte terminalde çalıştırılan komutlar görüntülenebilir.

```
usrht@usrht-VirtualBox:~$ history
1  history
2  echo $PATH
3  ls
4  which date
5  which cal
6  which echo
7  cal 2023 12
8  cal 2023
9  cal 12 2023
10 cal -y
11 cal -B 2 08 2023
12 cal -BA 2 08 2023
13 echo $PATH
14 man which
15 man -k which
16 man which
17 man -k which
18 history
```

Eğer ki geçmiş içerisinde 5. sıradaki komutu çalıştırmak istersek “!5” komutu çalıştırılır.

Geçmişteki son komutu çalıştırmak istersek “!!” komutu çalıştırılır.

Geçmişi temizlemek istersek;

```
history -c
```

Komut Yapısı

```
komut_adi secenek girdi
```

- Seçenekler - ile yazılır.
- Seçeneklerin uzun hali ile yazılışında -- kullanılır.
- Girdiler ise düz şekilde yazılır

```
date -u
```

```
date --universal
```

```
cal -A 2 9 2023
```

Şimdi ise komutların nereden geldiğini anlayalım.

Program dosyalarının nereden geldiğini anlamak için önce tüm dosyaları listeleyelim.

```
echo $PATH
```

```
usrht@usrht-VirtualBox:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
```

Biz terminale bir komut girdiğimizde bu sırayla klasörlere bakarak komutu bulur.

Hangi klasörün nerede olduğunu anlamak için “**which**” komutunu çalıştırırız.

```
which date
```

```
usrht@usrht-VirtualBox:~$ which date
/usr/bin/date
```

```
which cal echo cd
```

```
usrht@usrht-VirtualBox:~$ which cal echo cd
/usr/bin/cal
/usr/bin/echo
```

MAN PAGES

Linux içerisindeki kodları ezberlememize gerek yoktur. Linux komutların ne işe yaradığını gösteren bir mekanizma sağlar. (<https://www.man7.org/linux/man-pages/index.html>)

Linux Man Pages içinde komutlar 8 kategoriye ayrılmıştır.

“**man**” komutu ile çalışır. Örneğin **which** komutu ile ***alakalı*** komutlara bakmak istersek aşağıdaki şekilde yapabiliriz. Burada -k ile komutun içinde veya açıklamasının içinde which geçenleri aradık.

```
man -k which
```

```
usrht@usrht-VirtualBox:~$ man -k which
ifcfg (8) - simplistic script which replaces ifconfig IP management
IO::AtomicFile (3pm) - write a file which is updated atomically
lcf (1) - Determine which of the historical versions of a config...
pam_exec (8) - PAM module which calls an external command
pam_warn (8) - PAM module which logs all PAM items if called
securetty (5) - list of terminals on which root is allowed to login
sol (6) - a collection of card games which are easy to play with...
URI::WithBase (3pm) - URIs which remember their base
which (1) - locate a command
which.debianutils (1) - locate a command
```

which komutunun ne işe yaradığını öğrenmek istersek

```
man which
```

```
WHICH(1)                                General Commands Manual                                WHICH(1)

NAME
    which - locate a command

SYNOPSIS
    which [-a] filename ...

DESCRIPTION
    which returns the pathnames of the files (or links) which would be executed in the current environment, had its arguments been given as commands in a strictly POSIX-conformant shell. It does this by searching the PATH for executable files matching the names of the arguments. It does not canonicalize path names.

OPTIONS
    -a      print all matching pathnames of each argument

EXIT STATUS
    0       if all specified commands are found and executable
    1       if one or more specified commands is nonexistent or not executable

Manual page which(1) line 1 (press h for help or q to quit)
```

Yukarıda dikkat edilmesi gereken bir diğer nokta [] köşeli parantez içinde kullanabileceğim opsiyonlar var.

Köşeli parantez bu opsiyonların zorunlu olmadığını gösterir.

Eğer zorunlu bir kullanım olursa < > ile gösterilir.

```
which [ -a ] <OPTION> filename ...
```

Eğer veya işareti kullanmak istersek bu opsiyonlar içerisinde | işareti kullanırız. Burada veya bildiğimiz lojikteki gibi değil de, bu ikisinden biri ile çalıştırabilirsiniz anlamında.

```
which [ -a | -b ] <OPTION> filename ...
```

Eğer birden fazla opsiyonu aynı anda kullanmak istersek;

```
which [ -a , -b, -c ] <OPTION> filename ...
```

Örneğin bir komut arıyoruz ve bu komutun klasörleri listeleme ile ilgili olduğunu biliyoruz.

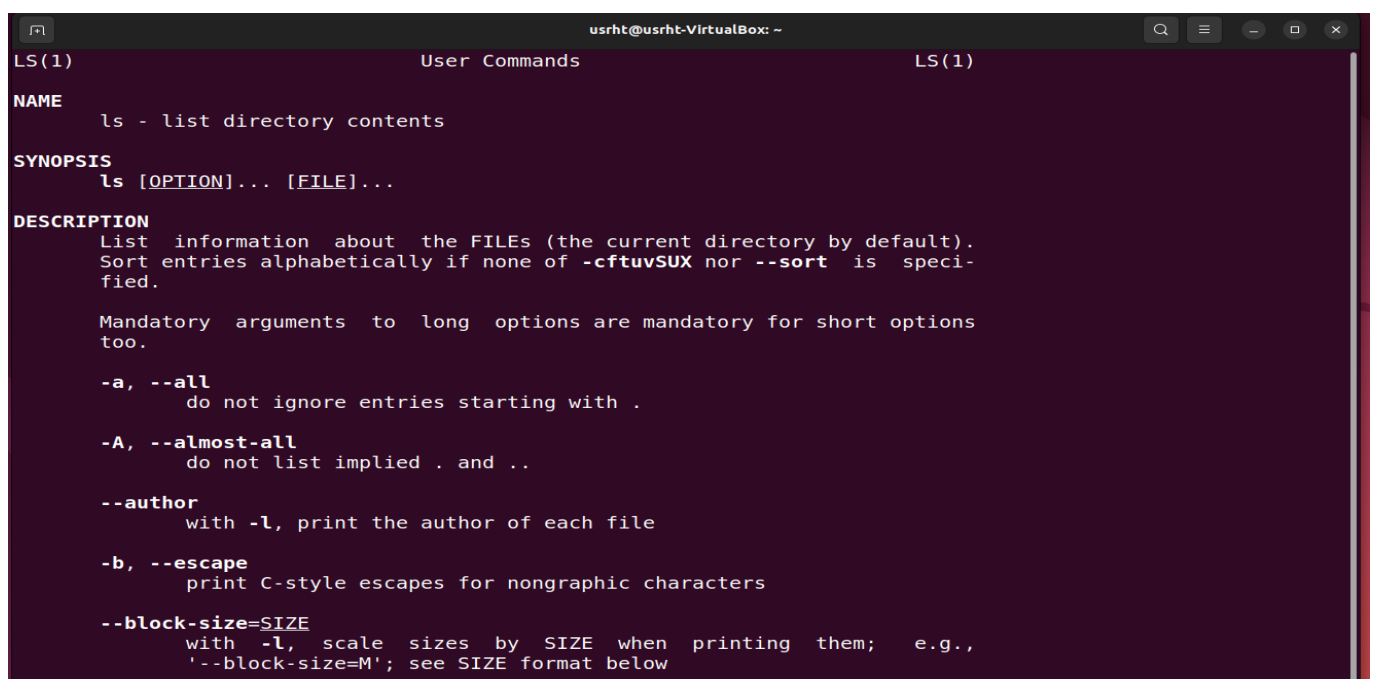
```
man -k "list directory"
```

```
usrht@usrht-VirtualBox:~$ man -k "list directory"
dir (1)          - list directory contents
ls (1)           - list directory contents
ntfsls (8)       - list directory contents on an NTFS filesystem
vdir (1)         - list directory contents
```

Burada görüyoruz ki “ls” komutu ile listeleme yapabiliyoruz.

Sadece ls komutu için arama yaparsak aşağıdaki gibi bir sonuç alırız.

```
man ls
```



```
usrht@usrht-VirtualBox: ~
LS(1) User Commands LS(1)

NAME
ls - list directory contents

SYNOPSIS
ls [OPTION]... [FILE]...

DESCRIPTION
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
fied.

Mandatory arguments to long options are mandatory for short options
too.

-a, --all
    do not ignore entries starting with .

-A, --almost-all
    do not list implied . and ..

--author
    with -l, print the author of each file

-b, --escape
    print C-style escapes for nongraphic characters

--block-size=SIZE
    with -l, scale sizes by SIZE when printing them; e.g.,
    '--block-size=M'; see SIZE format below
```

Şimdi yukarıdaki ls komutunu kullanalım.

```
ls
```

```
usrht@usrht-VirtualBox:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
```

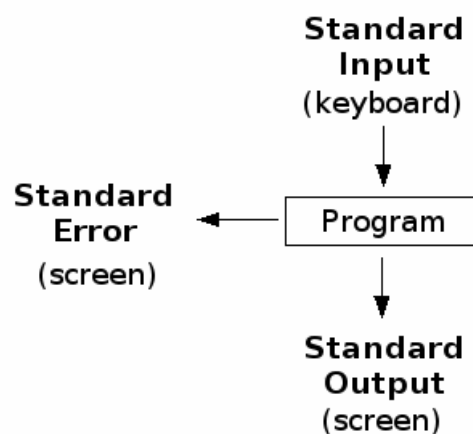
Detaylı bir listeleme yapmak istersek -l opsiyonunu kullanırız.

```
ls -l
```

```
usrht@usrht-VirtualBox:~$ ls -l
total 36
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Desktop
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Documents
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Downloads
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Music
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Pictures
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Public
drwx----- 3 usrht usrht 4096 Eki 11 23:14 snap
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Templates
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Videos
```

Dosyalar arasında aşağı yukarı gezinebilmemizi “cd” komutu ile sağlıyoruz. Ancak burada man komutunu cd komutu ile kullanamıyoruz. Bu durumda “help cd” ile bu bilgiye ulaşabiliriz. Man ile açıklaması bulunmayanlara bu şekilde ulaşılabilir.

Input & Output



Burada düzgün çalışan bir komut Standart Output olarak ekrana gelir, ancak hatalı girilen veya hatalı sonuç veren bir komut Standard Error olarak ekrana gelir.

Input & Output numaralandırmaları şu şekildedir

- Standard Input --> 0
- Standard Output --> 1
- Standard Error --> 2

Örnek ile ilerlersek;

Düzgün biçimde çalışan komut

```
cal
```

```
usrht@usrht-VirtualBox:~$ cal
      Ekim 2023
Pa Pz Sa Çr Pr Cu Ct
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Hatalı çalışan komut

```
usrht@usrht-VirtualBox:~$ cal -u
cal: invalid option -- 'u'
Usage: cal [general options] [-jy] [[month] year]
       cal [general options] [-j] [-m month] [year]
       ncal -C [general options] [-jy] [[month] year]
       ncal -C [general options] [-j] [-m month] [year]
       ncal [general options] [-bhJjpwySM] [-H yyyy-mm-dd] [-s country_code] [-W
number of days] [[month] year]
       ncal [general options] [-Jeo] [year]
General options: [-31] [-A months] [-B months] [-d yyyy-mm]
```

Yukarıda görüldüğü üzere “cal” komutu için “-u” opsiyonu bulunmamaktadır.

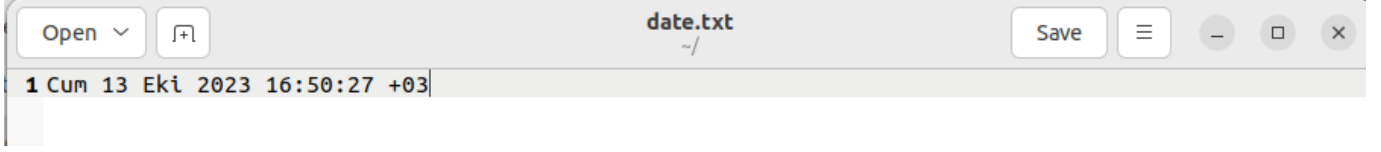
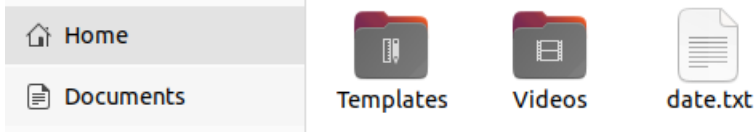
Komut Yönlendirme (Redirecting)

Input akışlarını gördükten sonra bu kanallardan aktarılan verileri bir dosyaya veya komuta aktarmayı görelim.

Mesela “date” komutu ile güncel tarihi elde ederiz. Peki ben bu komutu input olarak verdiğimde output olarak aldığım veriyi date.txt isimli bir dosyaya aktarmak istersem ne yapmalıyım ?

Bu sorunun cevabı şu şekilde;

```
date 1> date.txt
```



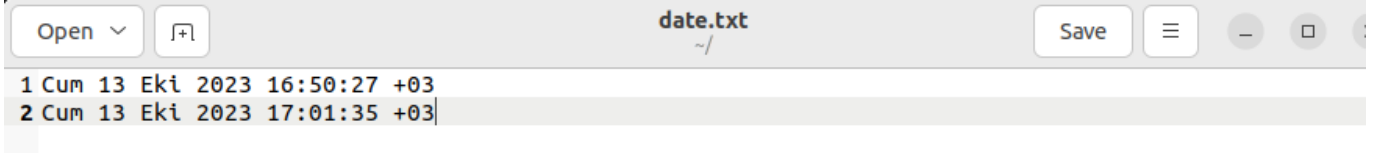
Yukarıda da görüldüğü üzere 1 numaralı kanaldan yani Standard Output cevabını date.txt içerisine aktardık.

Bu komutu tekrar çalıştırsak date.txt içerisindeki veriyi yeni komut ile ezer.

Peki ben yeniden çalıştırdığım komut çıktısını altına eklemek istersem ne yapmalıyım?

Bunun için aşağıdaki komutu kullanabiliriz.

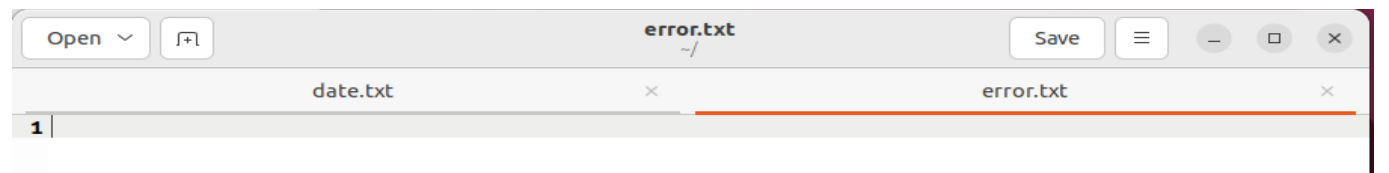
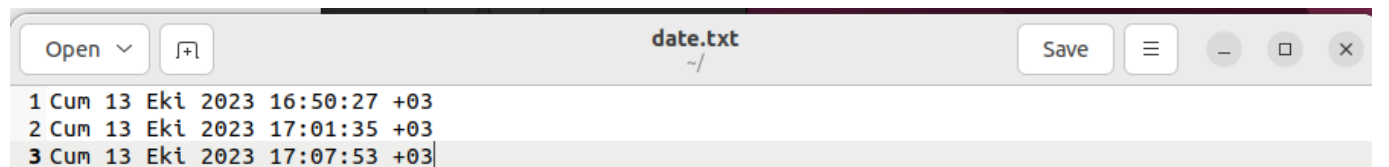
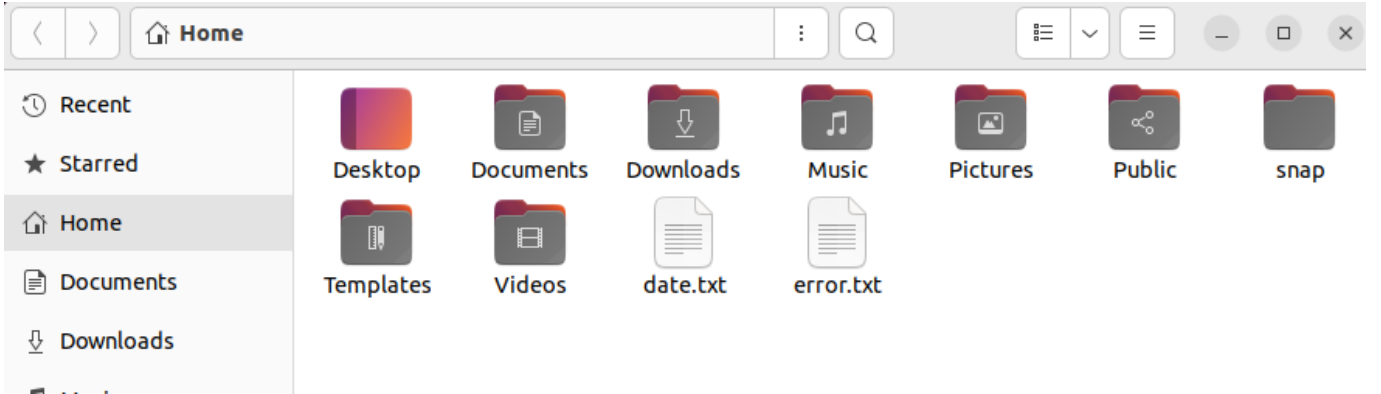
```
date 1>> date.txt
```



Sonuç yukarıdaki gibi olacaktır.

Burada 1 numaralı kanaldan gelenleri date.txt içine yönlendirdim. Peki 2 numaralı kanaldan gelenleri de ayrı bir dosyaya yönlendirmek istersek ne yapmalıyız ?

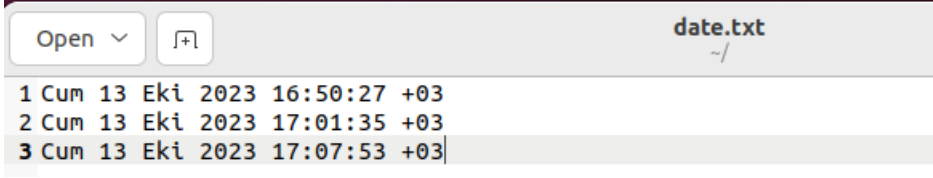
```
date 1>> date.txt 2>> error.txt
```



Yukarıda da görüldüğü üzere 2 numaralı kanaldan bir hata çıktısı olmadığı için herhangi bir veri yazılmamıştır.

Hata üretebilen bir komut girersek;

```
date shdhsldlshd 1>> date.txt 2>> error.txt
```



A screenshot of a text editor window titled 'date.txt'. The window has a toolbar with 'Open' and a file icon. The content of the file is as follows:

```
1 Cum 13 Eki 2023 16:50:27 +03
2 Cum 13 Eki 2023 17:01:35 +03
3 Cum 13 Eki 2023 17:07:53 +03
```



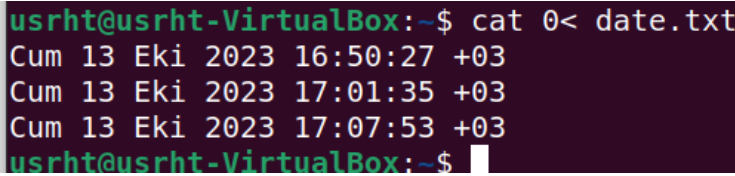
A screenshot of a text editor window titled 'error.txt'. The window has a toolbar with 'Open' and a file icon. The content of the file is as follows:

```
1 date: invalid date 'skhkshkdh'
```

Yukarıda görüldüğü üzere 2 numaralı standard error taraflı çıktudan gelen hatalı işlemde yeni bir satır eklendi fakat 1 numaralı standard output çıktısı olmadığı için date.txt içerisine herhangi bir şey eklenmedi.

Standard Input içinde bir veri aktarabiliriz. Bunun için daha sonradan göreceğimiz “cat” komutunu kullanalım.

```
cat 0< date.txt
```



A terminal window screenshot showing the execution of the 'cat' command. The prompt is 'usrht@usrht-VirtualBox:~\$'. The output is as follows:

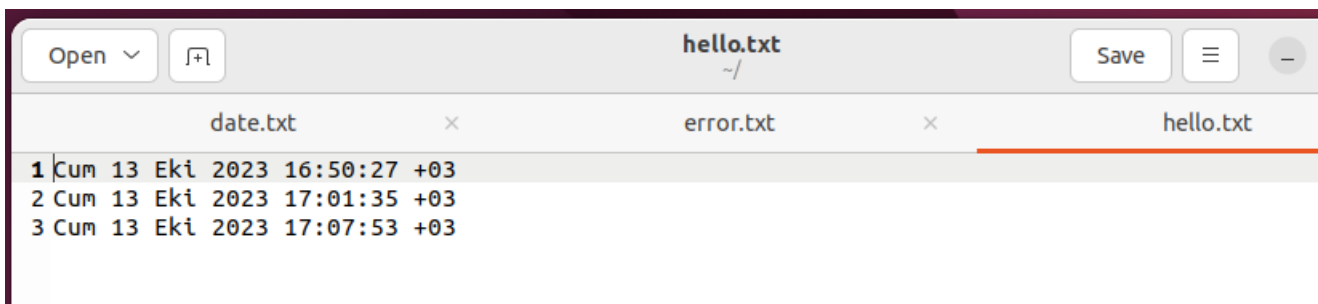
```
usrht@usrht-VirtualBox:~$ cat 0< date.txt
Cum 13 Eki 2023 16:50:27 +03
Cum 13 Eki 2023 17:01:35 +03
Cum 13 Eki 2023 17:07:53 +03
usrht@usrht-VirtualBox:~$
```

Görüldüğü üzere önceden date.txt içine eklenen 3 satırlık verileri bu şekilde okuyabiliyoruz.

Buradan “cat” komutu ile aldığımız verileri “hello.txt” isimli dosyaya yazdıralım. Eğer bir hata olursa da “error.txt” içerisine bunu yazdıralım.

```
cat 0< date.txt 1>> hello.txt 2>> error.txt
```

Başarılı şekilde çalıştığı için “hello.txt” dosyası şu şekilde olacaktır.

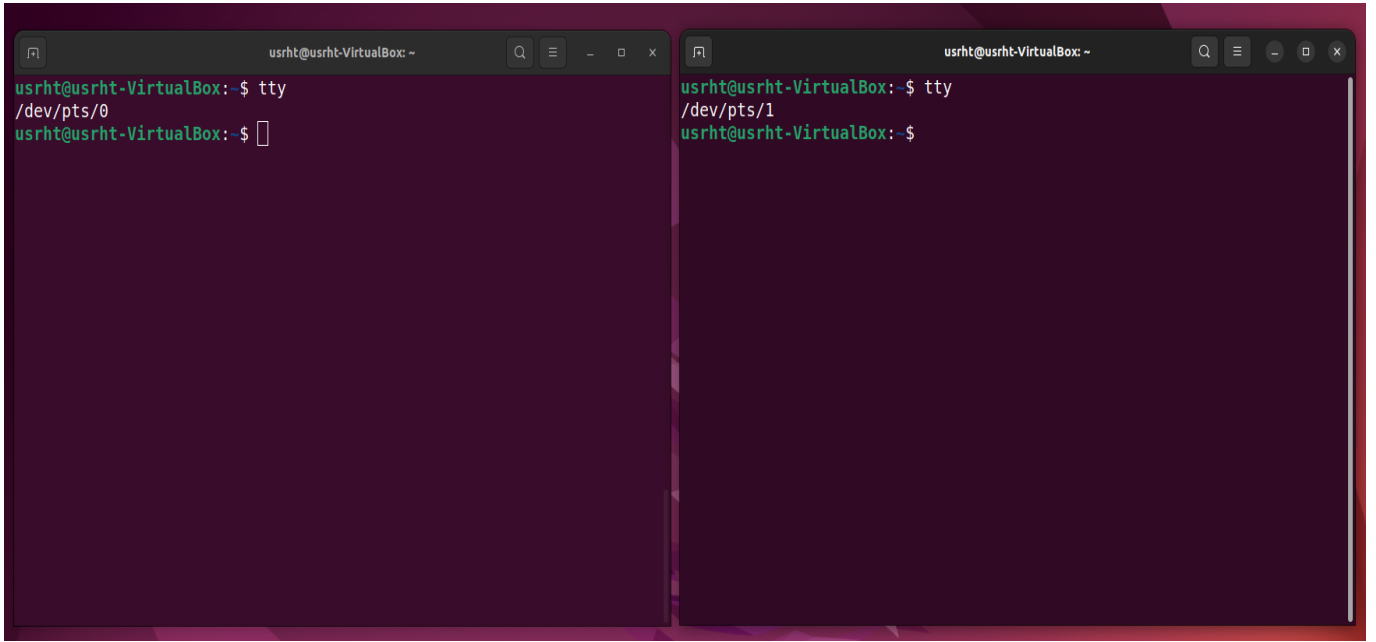


A screenshot of a text editor window titled 'hello.txt'. The window has a toolbar with 'Open', a file icon, 'Save', and window control buttons. The content of the file is as follows:

```
1 Cum 13 Eki 2023 16:50:27 +03
2 Cum 13 Eki 2023 17:01:35 +03
3 Cum 13 Eki 2023 17:07:53 +03
```


Kullandığımız terminaller de aslında birer dosyadan ibarettir.

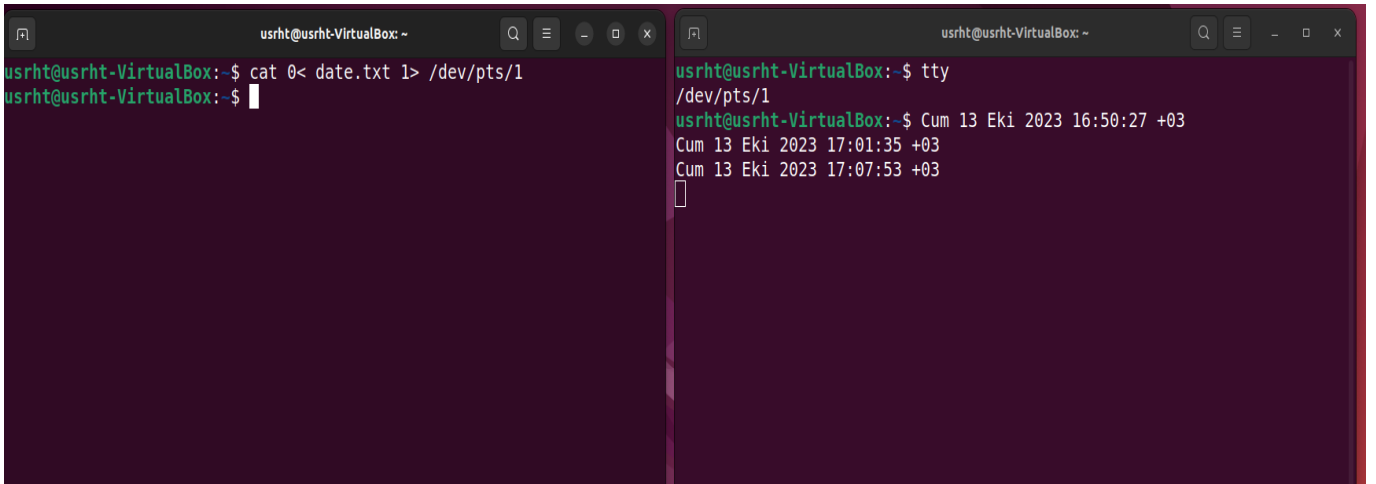
2 ayrı terminal açalım ve birinden diğerine bir input yollamayı deneyelim.



```
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ tty  
/dev/pts/0  
usrht@usrht-VirtualBox:~$  
  
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ tty  
/dev/pts/1  
usrht@usrht-VirtualBox:~$
```

Yukarıda görüldüğü üzere her terminal ayrı birer dosya konumuna işaret etmektedir.

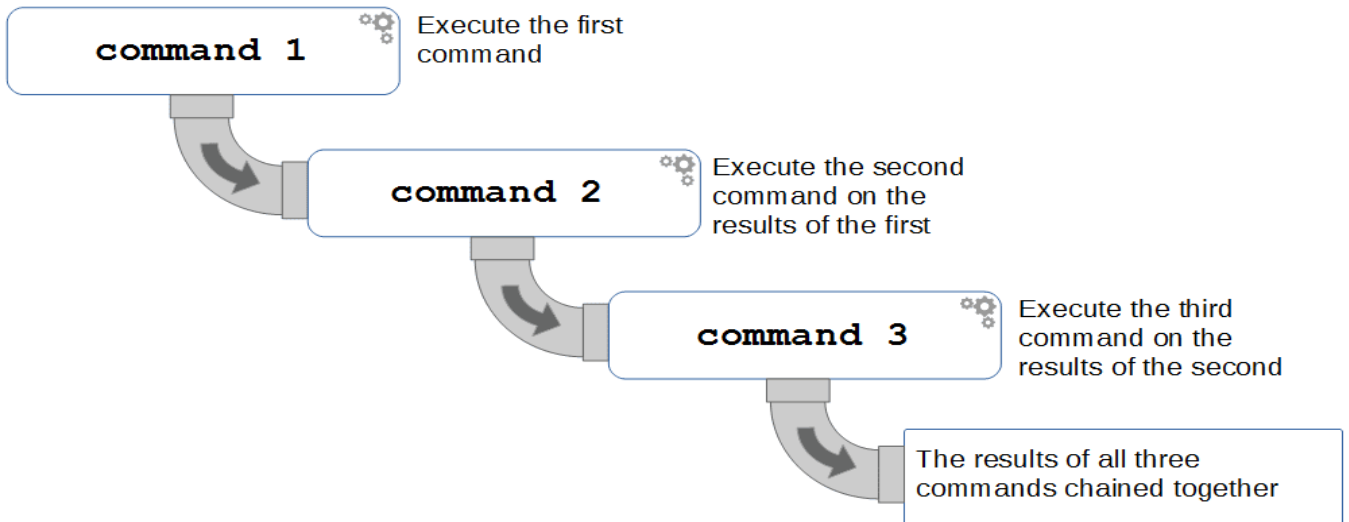
Şimdi 1. terminal içerisinde “**date.txt**” içerisinde yazanları input olarak 2. terminale aktaralım.



```
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ cat 0< date.txt 1> /dev/pts/1  
usrht@usrht-VirtualBox:~$  
  
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ tty  
/dev/pts/1  
usrht@usrht-VirtualBox:~$ Cum 13 Eki 2023 16:50:27 +03  
Cum 13 Eki 2023 17:01:35 +03  
Cum 13 Eki 2023 17:07:53 +03  
usrht@usrht-VirtualBox:~$
```

Yukarıdaki gibi bir çıktı alırız.Yani Linux içindeki herşey birer dosyadan oluşmaktadır.

Komut Birleştirme (Piping)



Yukarıda da anlaşıldığı üzere bir basamakta işlem yapıp çıktıyı diğer bir adıma aktararak devam eden süreçtir.

Burada “**cut**” komutunu da kullanacağız. Bu komutu “**man**” komutu ile incelersek;

```
man cut
```

```
usrht@usrht-VirtualBox: ~  
CUT(1) User Commands CUT(1)  
NAME  
    cut - remove sections from each line of files  
SYNOPSIS  
    cut OPTION... [FILE]...  
DESCRIPTION  
    Print selected parts of lines from each FILE to standard output.  
    With no FILE, or when FILE is -, read standard input.  
    Mandatory arguments to long options are mandatory for short options too.  
    -b, --bytes=LIST  
        select only these bytes  
    -c, --characters=LIST  
        select only these characters  
    -d, --delimiter=DELIM
```

Bu komut aslında bildiğimiz kopyalama komutudur. Yukarıda da ayrıntılı bir şekilde görebiliriz.

Bir örnek üzerinden gidecek olursak;

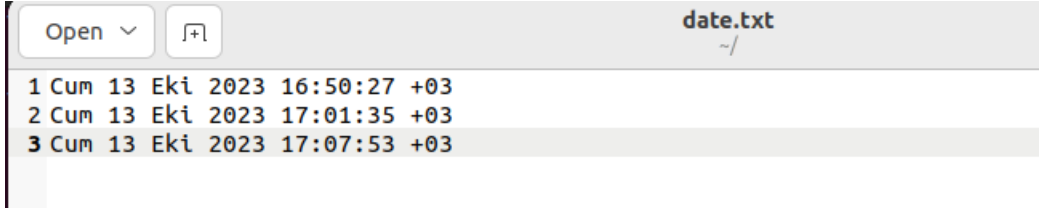
```
date 1>> date.txt
```

Önceki örneklerde olduğu gibi **date** komutu üzerinden ilerliyoruz ve mevcut zamanı **date.txt** dosyası üzerine ekliyoruz.

Şimdi ise **date.txt** dosyası içerisindeki bu veriyi keselim.

```
cut 0< date.txt --delimiter " " --fields 3
```

Yukarıda yaptığımız işlemi şu şekilde açıklayabiliriz.



```
Open  [icon] date.txt ~/
1 Cum 13 Eki 2023 16:50:27 +03
2 Cum 13 Eki 2023 17:01:35 +03
3 Cum 13 Eki 2023 17:07:53 +03
```

```
usrht@usrht-VirtualBox:~$ cut 0< date.txt --delimiter " " --fields 4
2023
2023
2023
usrht@usrht-VirtualBox:~$
```

İlk olarak **date.txt** içerisindeki verileri cut komutunun input verisi olarak aldık. Daha sonra bu veriyi boşluk ile ayırdık ve 4. sütunu almak istedik. Sonuç olarakta 2023 verisini elde ettik.

Peki bunu daha kısa nasıl yapabiliriz?

```
date | cut -d " " -f 4
```

Yukarıdaki işlemde aynı şekilde çalışacaktır. Burada “|” ile bir önceki komutun çıktısı sonraki kısma aktarılır.

Bu sonucu da “years.txt” klasörünün içine aktarırsak;

```
date | cut -d " " -f 4 1>> years.txt
```

bu şekilde çalıştırabiliriz.

Ancak burada bir problem var. O da aşağıdaki komutu çalıştırsak;

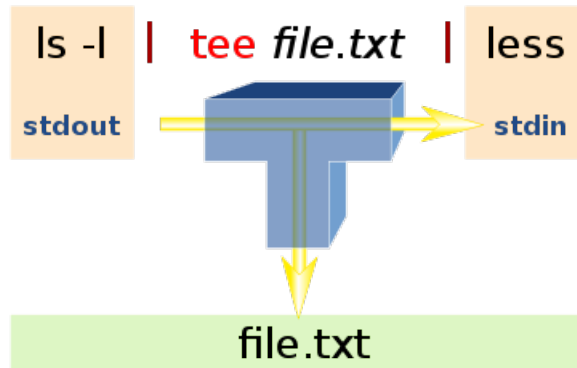
```
date > date.txt | cut -d " " -f 4
```

Bu komutu çalıştırsak;

```
usrht@usrht-VirtualBox:~$ date > date.txt | cut -d " " -f 4
usrht@usrht-VirtualBox:~$
```

Yukarıda görüldüğü üzere işlemime farklı bir input ile devam edemem çünkü bizim pipe sürecimize bir veri dönmüyor. Bunun için farklı bir çözüm yapacağız.

Tee Komutu



Burada tee komutu ile yukarı yapamadığımız file yazma işleminden sonra bir çıktı alamama sorununu bu şekilde çözüyoruz.

```
date | tee date_tee.txt | cut -d " " -f 1 > days.txt
```

Süreci uzatıp days.txt dosyasına yazdıktan sonra bunu bide terminale yazsın istersek;

```
date | tee date_tee.txt | cut -d " " -f 1 | tee days.txt | cat
```

```
usrht@usrht-VirtualBox:~$ date | tee date_tee.txt | cut -d " " -f 1 | tee days.txt | cat
Cum
usrht@usrht-VirtualBox:~$
```

Xarg Komutu

```
date | echo
```

Yukarıdaki piping mantığına göre burada ekrana date komutundan gelen output değeri yazılmalı fakat boş yazılıyor.

Burada echo terminal input olarak alınıyor ve aşağıdaki gibi yaparsak yazacaktır ama amacımıza ulaşmış olmayız bu şekilde.

```
echo hello
```

```
date | echo hello
```

Yukarıda pipe üzerinden gelen bir standart input kabul etmiyor.

```
date | xargs echo
```

```
usrht@usrht-VirtualBox:~$ date | xargs echo  
Cum 03 Kas 2023 16:15:10 +03
```

Yukarıda görüldüğü üzere “**xargs**” komutu ile date komutunun çıktısını echo için bir input olarak verebildik.

```
date | xargs echo "hello"
```

Yazarsak echo komutu öncelikle kendi input değerini daha sonra xargs ile geleni alır.

```
usrht@usrht-VirtualBox:~$ date | xargs echo "hello"  
hello Cum 03 Kas 2023 16:19:43 +03
```

Şimdi ilk olarak **new_file.txt** adında bir dosya oluşturalım.

Bu dosyanın içinde 3 adet

```
file1.txt  
file2.txt  
file3.txt
```

şeklinde satır bulunmaktadır.

Ben bu satırları alıp 3 ayrı dosya oluşturabilmek istiyorum

Oluşturma komutu “**touch**” olmaktadır.

```
usrht@usrht-VirtualBox:~$ cat new_file.txt  
file1.txt  
file2.txt  
file3.txt  
usrht@usrht-VirtualBox:~$ cat new_file.txt | touch  
touch: missing file operand  
Try 'touch --help' for more information.
```

Ancak hata almaktayız. Echo komutu gibi yapabiliriz.

```
cat new_file.txt | xargs touch
```

Yukarıdaki komut çalıştırılırsa 3 ayrı dosyanın oluştuğunu görürüz.

Şimdi bu oluşturulan dosyaları silmek istersek;

```
cat new_file.txt | xargs rm
```

ALIAS

Eğer elimizde devamlı kullandığımız komutlar varsa bunları kısaltarak bir alternatif komut ismi atanabilir.

Aşağıda ls komutu ile detaylı olarak dosyalarımız listeleyelim.

```
usrht@usrht-VirtualBox:~$ ls -la
total 104
drwxr-xr-x 14 usrht usrht 4096 Kas  3 16:22 .
drwxr-xr-x  3 root  root  4096 Eki 11 22:39 ..
-rw-r----- 1 usrht usrht   38 Eki 13 16:12 .bash_history
-rw-r--r--  1 usrht usrht  220 Eki 11 22:39 .bash_logout
-rw-r--r--  1 usrht usrht 3771 Eki 11 22:39 .bashrc
drwx----- 11 usrht usrht 4096 Eki 12 21:36 .cache
drwx----- 14 usrht usrht 4096 Eki 13 17:01 .config
-rw-rw-r--  1 usrht usrht   29 Kas  3 15:49 date_tee.txt
-rw-rw-r--  1 usrht usrht   62 Kas  1 13:35 date.txt
-rw-rw-r--  1 usrht usrht    4 Kas  3 15:49 days.txt
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Desktop
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Documents
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Downloads
-rw-rw-r--  1 usrht usrht  130 Kas  1 13:42 error.txt
-rw-rw-r--  1 usrht usrht  211 Kas  1 13:41 hello.txt
-rw-r----- 1 usrht usrht   20 Kas  1 13:44 .lessht
drwx-----  3 usrht usrht 4096 Eki 11 23:14 .local
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Music
-rw-rw-r--  1 usrht usrht   30 Kas  3 16:22 new_file.txt
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Pictures
-rw-r--r--  1 usrht usrht  807 Eki 11 22:39 .profile
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Public
drwx-----  3 usrht usrht 4096 Eki 11 23:14 snap
-rw-r--r--  1 usrht usrht    0 Eki 12 21:54 .sudo_as_admin_successful
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Templates
drwxr-xr-x  2 usrht usrht 4096 Eki 11 23:14 Videos
-rw-rw-r--  1 usrht usrht   15 Kas  1 14:01 years.txt
```

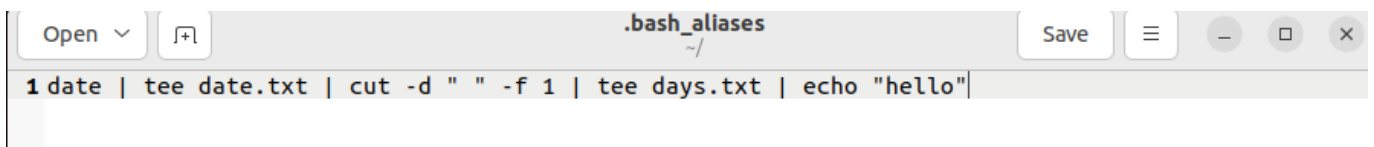
Yukarıda görüldüğü üzere “.” ile başlayanlar gizli dosyadır.

Alias oluşturmak için de gizli bir “**.bash_aliases**” isminde dosya oluşturmamız gerekiyor.

Mesela az önce kullandığımız bir komutun benzerini ele alalım.

```
date | tee date_tee.txt | cut -d " " -f 1 | tee days.txt | echo "hello"
```

Bu komutu sürekli kullandığımızı varsayalım.



Bunu her kullanmak istediğimde uzun uzun yazmak yerine bunu **tarihkaydet** isimli bir komuta atarsak neler olur görelim.

Burada **.bash_aliases** isimli dosyayı aşağıdaki şekilde değiştirmemiz gerekiyor.

```
.bash_aliases
1 alias tarih kaydet='date | tee date.txt | cut -d " " -f 1 | tee days.txt | xargs echo "hello"'
```

Bu şekilde kaydediyoruz ve tüm terminaller kapatıp tekrar açıyoruz.

```
usrht@usrht-VirtualBox:~$ tarih kaydet
hello Cum
usrht@usrht-VirtualBox:~$
```

```
date.txt
1 Cum 03 Kas 2023 16:59:05 +03
```

```
days.txt
1 Cum
```

Yukarıda görüldüğü gibi alias olarak atadığımız **tarihkaydet** komutu düzgün şekilde çalışıyor.

Yeni bir alias daha oluşturalım ve adı da **takvimkaydet** olsun.

```
.bash_aliases
1 alias tarih kaydet='date | tee date.txt | cut -d " " -f 1 | tee days.txt | xargs echo "hello"
2 alias takvim kaydet='xargs cal -A 1 -B 1 > takvim.txt'
```

Yukarıda görüldüğü üzere **xargs** komutu kullanıldı. Çünkü bunu bir argüman olarak kullanmak istiyorum.

```
echo "5 2023" | takvim kaydet
```

Yukarıdaki komut çalışınca mayıstan 1 ay önce ve sonrası arasını **takvim.txt** içerisine yazmasını bekliyoruz.

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

Linux Klasör ve Dosya Sistemi

```
usrht@usrht-VirtualBox:~$
```

Terminal açıldığında bizi karşılayan ve işlemleri üzerinden yaptığımız ifadenin anlamı;

- “usrht” kısmı terminalde işlem yapan kullanıcıyı
- @ ifadesinden sonra gelen “usrht-VirtualBox” ise hostname olmaktadır.
- “~” tilda işareti ise işlem yapılan konumu bize vermektedir.
- Aslında bu tilda ifadesi /home/usrht ifadesinin yerine geçmektedir. Kök dizindir yani en tepede yer alan klasördür.
- \$ işareti ise sıradan bir kullanıcının işlem yaptığı anlamına gelir
- # işareti olursa burada admin yetkileriyle donatılmış daha fazla yetkili kullanıcı olduğu anlamına gelir.

```
usrht@usrht-VirtualBox:~$ pwd
/home/usrht
usrht@usrht-VirtualBox:~$
```

Yukarıda yazdığımız pwd (print working directory) ile bize tilda ile gelen ifade gelmektedir.

“ls” komutu ile ise bu konumdaki klasörleri listeleyebiliriz. Bunu daha önce de incelemiştik.

Mavi renktekiler klasörü, beyazlar ise dosyaları simgeler.

```
usrht@usrht-VirtualBox:~$ ls
date_tee.txt  Desktop  error.txt  new_file.txt  snap  Videos
date.txt      Documents hello.txt  Pictures      takvim.txt  years.txt
days.txt     Downloads Music      Public        Templates
usrht@usrht-VirtualBox:~$
```

Eğer -l opsiyonu ile çalıştırsak

```
usrht@usrht-VirtualBox:~$ ls -l
total 68
-rw-rw-r-- 1 usrht usrht 29 Kas 3 15:49 date_tee.txt
-rw-rw-r-- 1 usrht usrht 29 Kas 3 16:59 date.txt
-rw-rw-r-- 1 usrht usrht 4 Kas 3 16:59 days.txt
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Desktop
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Documents
```

1. kolon yetkilendirmeyi (-rw-rw-r--)
2. kolon (hardlink) kısayol sayısını belirtir. (1)
3. kolon dosyanın sahibi olan kullanıcıyı (usrht)
4. kolon dosyanın sahibi olan grubu (usrht)
5. kolon dosyanın kapasitesini (29)
6. kolon dosyanın oluşturulma zamanını (Kas 3 15:49)

7. kolon ise dosya veya klasörün adını belirtir. (**date_tee.txt**)

eğer başında d ile başlıyorsa “directory” yani bir klasör olduğunu, bunun yerine sadece - varsa bunun bir dosya olduğunu belirtir.

Bu işareten sonra orada 9 haneli bir grup var.

9 haneli grubu 3 ana parçaya ayırabiliriz

rwX/rwx/rwx olarak ve

- ilki dosyanın sahibi olan kullanıcının bu dosya üzerindeki yetkisini,
- ikincisi bu dosyanın sahibi olan grubun bu dosya üzerindeki etkisi
- üçüncüsü ise dosya sahibinin ve sahibi olan grubun dışındaki kişilerin yetkilerini gösterir

Bu 9 haneli gruplardan

Örneğin “**drwxr-xr-x**”

d’den sonra gelen ilk 3 harf yani “**rwX**” dosya sahibi kullanıcının bu dosya üzerindeki yetkilerini

- *r - read*
- *w - write*
- *x - execute*

ikinci olarak gelen **r-x** ise dosya sahibi olan grubun yetkilerini

- *r - read*
- *x - execute*

üçüncü olarak gelen **r-x** ise aynı şekilde bu kişi ve grupların dışındaki kullanıcıların yetkilerini gösterir.

- *r - read*
- *x - execute*

cd Komutu

Bu komut ile klasörler arası gezinebiliriz.

Bildiğimiz gibi ~ tilde işareti ile kök dizine ulaşabiliriz. Burada bunu da kullanabiliriz.

```
usrht@usrht-VirtualBox: ~/Downloads
usrht@usrht-VirtualBox:~$ cd /home/usrht/Downloads
usrht@usrht-VirtualBox:~/Downloads$
```

```
usrht@usrht-VirtualBox: ~/Downloads
usrht@usrht-VirtualBox:~$ cd ~/Downloads
usrht@usrht-VirtualBox:~/Downloads$
```

Bildiğimiz üzere “.” bulunduğumuz konumu ifade eder. Yani bunu kullanarak ta bu konuma ulaşabiliriz. Şuan da zaten tilde (~) konumu olan **/home/usrht** konumunda bulunmaktayız.

Bir klasör geri gitmek istersek tanıdık bir kullanım olan iki nokta ile bunu yaparız.

```
usrht@usrht-VirtualBox:~$ pwd
/home/usrht
usrht@usrht-VirtualBox:~$ cd ..
usrht@usrht-VirtualBox:/home$ pwd
/home
```

Eğer ki 2 önceki klasör gitmek istersek;

```
usrht@usrht-VirtualBox:~$ pwd
/home/usrht
usrht@usrht-VirtualBox:~$ cd ../../
usrht@usrht-VirtualBox:/$ pwd
/
usrht@usrht-VirtualBox:/$
```

Yukarıda görüldüğü üzere 2 klasör öncesine yani kök dizine kadar indik.

Eğer sadece cd komutunu çalıştırsak bu bizi **/home/usrht** konumuna götürür.

WILDCARD

Arama yaparken kalıp olarak benzer isim taşıyan dosya ve klasörleri tek bir komut ile bulma, silme ve diğer işlemlerde kullanmaya fayda sağlayan karakterlerdir.

Eğer 3 ayrı klasörün altında dosyaları veya bulunduğum klasörün altındaki dosyaları listelemek istersem;

```
usrht@usrht-VirtualBox:~$ ls Desktop/ Downloads Documents/
Desktop/:

Documents/:

Downloads:
usrht@usrht-VirtualBox:~$ ls
date_tee.txt  Desktop      error.txt    new_file.txt  snap         Videos
date.txt      Documents    hello.txt    Pictures       takvim.txt   years.txt
days.txt     Downloads    Music        Public         Templates
```

ls komutundan sonra “*” ifadesi kullanılırsa bu klasörün altındaki herşey anlamına gelir.

```
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ ls *  
date_tee.txt  days.txt  hello.txt  takvim.txt  
date.txt      error.txt  new_file.txt  years.txt  
  
Desktop:  
  
Documents:  
  
Downloads:  
  
Music:  
  
Pictures:  
  
Public:  
  
snap:  
snapd-desktop-integration  
  
Templates:  
  
Videos:  
usrht@usrht-VirtualBox:~$
```

Mesela “D” harfi ile başlayan klasörleri listelemek istersek;

```
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ ls D*  
Desktop:  
  
Documents:  
  
Downloads:  
usrht@usrht-VirtualBox:~$
```

Eğer “file” ile başlayıp arada tek harfli bilinmeyen olup ve devamında “.txt” ile biten olursa;

```
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ ls file*  
file11.txt  file22.txt  file33.txt  fileAA.txt  
file1.txt   file2.txt   file3.txt   fileA.txt  
usrht@usrht-VirtualBox:~$ ls file?.txt  
file1.txt  file2.txt  file3.txt  fileA.txt  
usrht@usrht-VirtualBox:~$
```

Eğer “**file**” yazısından itibaren sadece A-Z arası büyük harf olmasını ve devamında “**.txt**” ile bitmesini istersek;

```
usrht@usrht-VirtualBox: ~$ ls
date_tee.txt  Downloads  file2.txt  hello.txt  snap
date.txt      error.txt  file33.txt Music      takvim.txt
days.txt     file11.txt file3.txt  new_file.txt Templates
Desktop       file1.txt  fileAA.txt Pictures    Videos
Documents     file22.txt fileA.txt  Public     years.txt
usrht@usrht-VirtualBox:~$ ls file[A-Z].txt
fileA.txt
usrht@usrht-VirtualBox:~$
```

Yukarıdaki örnekten farklı olarak A-Z arası değil de sadece ondalık bir adet sayı olsun istersek;

```
usrht@usrht-VirtualBox:~$ ls file[0-9].txt
file1.txt file2.txt file3.txt
usrht@usrht-VirtualBox:~$
```

Klasör ve Dosya Oluşturma Komutları

Dosya oluşturma komutu “**touch**”

file ve **file2** isiminde 2 dosya oluşturmak istersek;

```
touch file file2
```

Eğer ki “**Documents**” klasörü içerisinde “**file1.txt**” isiminde bir dosya oluşturmak istersek;

```
touch ~/Documents/file1.txt
```

Klasör oluşturma komutu “**mkdir**”

“**folder1**” isiminde bir klasör oluşturalım.

```
mkdir folder1
```

Eğer ki 2 ayrı klasör oluşturmak istersek;

```
mkdir folder1 folder2
```

Ancak şöyle de bir durum var, eğer ki komuttan sonra arasında boşluk bulunan bir klasör ismi yazmak istersek bunu kesme işaretiyle belirtmeliyiz aksi halde 2 ayrı klasör oluşturur.

```
mkdir "hello world"
```

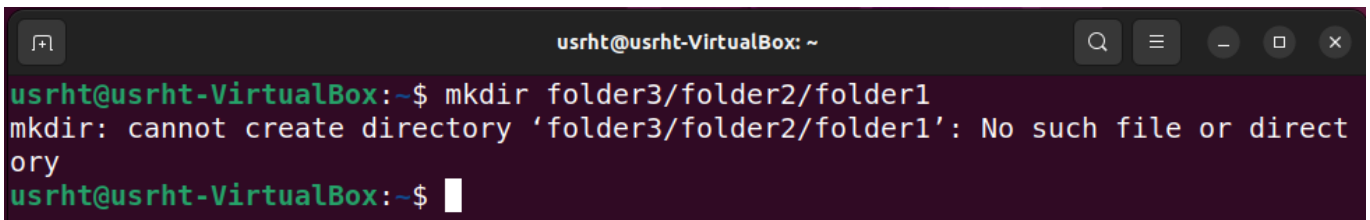
NOT !

- Dosya veya klasör isimlerinde boşluk bırakılmaması önerilir.

İç içe klasörler oluşturmak istersek;

```
mkdir folder3/folder2/folder1
```

Eğer yukarıdaki komut çalıştırılırsa;

A terminal window titled 'usrht@usrht-VirtualBox: ~' with search, menu, and window control icons in the title bar. The prompt is 'usrht@usrht-VirtualBox:~\$'. The command 'mkdir folder3/folder2/folder1' is entered. The output is 'mkdir: cannot create directory 'folder3/folder2/folder1': No such file or directory'. The prompt returns to 'usrht@usrht-VirtualBox:~\$' with a cursor.

```
usrht@usrht-VirtualBox:~$ mkdir folder3/folder2/folder1
mkdir: cannot create directory 'folder3/folder2/folder1': No such file or directory
usrht@usrht-VirtualBox:~$
```

Yani klasörleri oluştururken üstündeki klasörü oluşturamaz.

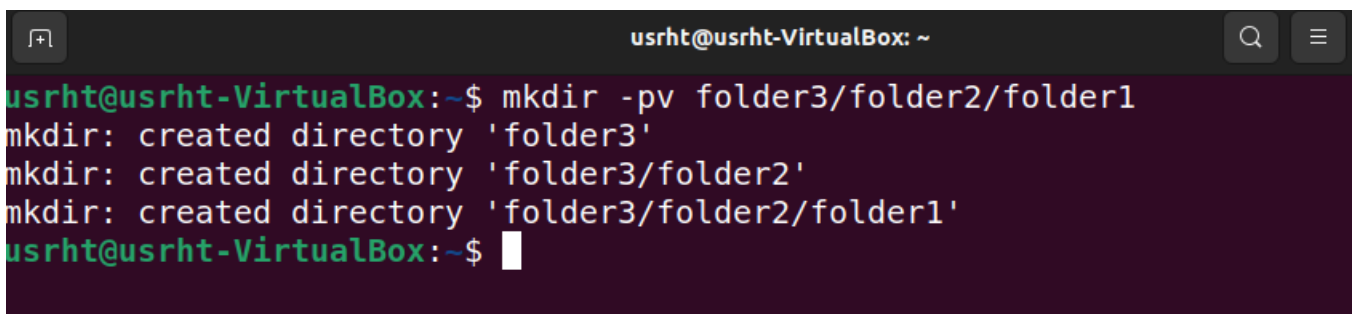
Bunun için -p (parent) opsiyonunu ekliyoruz.

```
mkdir -p folder3/folder2/folder1
```

Bunu yazınca klasörleri oluşturmadan önce olmayan klasörleri sırayla oluşturur.

Eğer oluşturulan dosya detaylarını output olarak almak istersek -v (verbose) opsiyonunu da ekleriz.

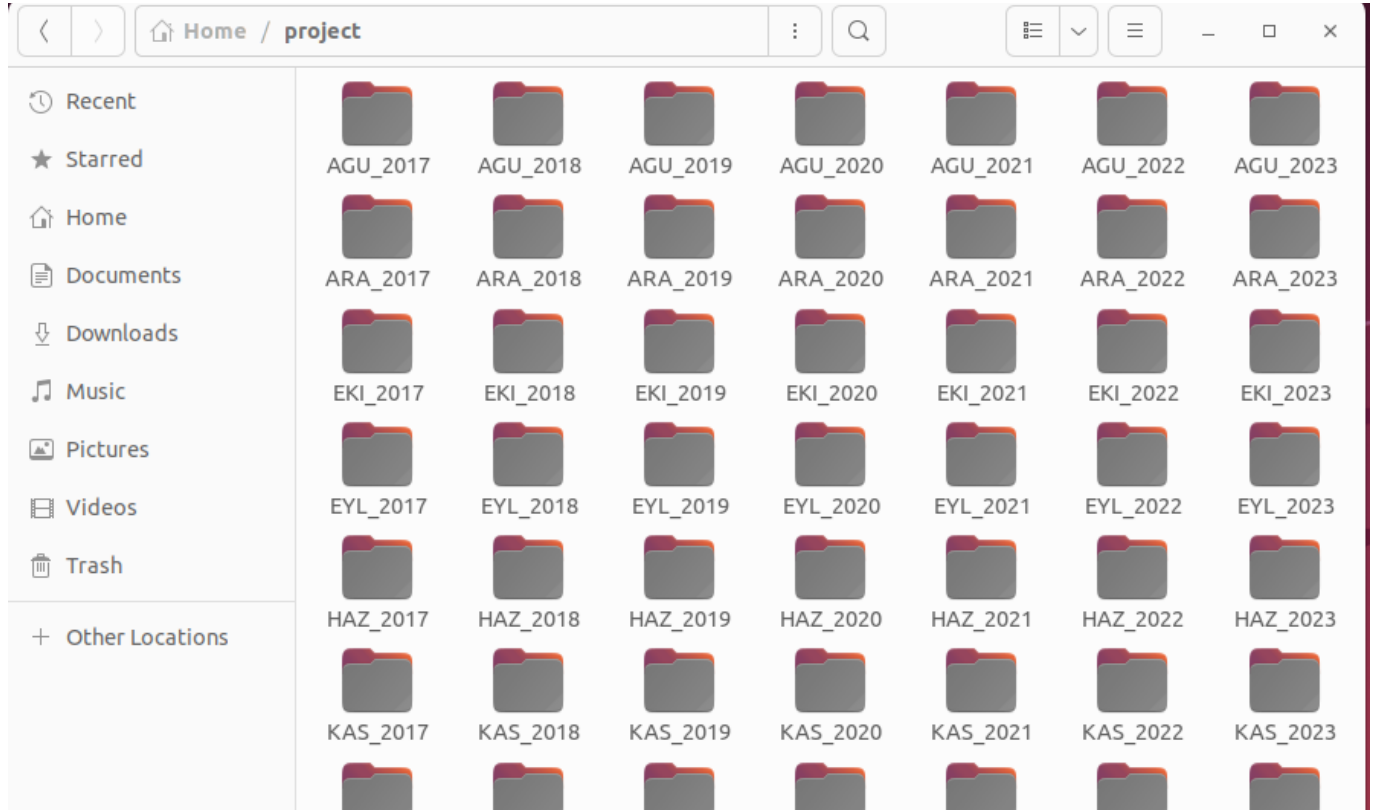
```
mkdir -pv folder3/folder2/folder1
```

A terminal window titled 'usrht@usrht-VirtualBox: ~' with search and menu icons in the title bar. The prompt is 'usrht@usrht-VirtualBox:~\$'. The command 'mkdir -pv folder3/folder2/folder1' is entered. The output is 'mkdir: created directory 'folder3'', 'mkdir: created directory 'folder3/folder2'', and 'mkdir: created directory 'folder3/folder2/folder1''. The prompt returns to 'usrht@usrht-VirtualBox:~\$' with a cursor.

```
usrht@usrht-VirtualBox:~$ mkdir -pv folder3/folder2/folder1
mkdir: created directory 'folder3'
mkdir: created directory 'folder3/folder2'
mkdir: created directory 'folder3/folder2/folder1'
usrht@usrht-VirtualBox:~$
```

Mesela bir klasör oluşturup daha sonra altında belirlediğimiz yıllar ve onlara göre tüm aylar için alt klasör oluşturursak;

```
usrht@usrht-VirtualBox:~$ mkdir project
usrht@usrht-VirtualBox:~$ cd project
usrht@usrht-VirtualBox:~/project$ ls
usrht@usrht-VirtualBox:~/project$ mkdir {OCA,SUB,MAR,NIS,MAY,HAZ,TEM,AGU,EYL,EKI,KAS,ARA}_{2017,2018,2019,2020,2021,2022,2023}
```



Belirtilen tüm yıllar için 12 ay için de bir klasör oluşturulmuştur.

Bu kod bloğunu daha kısa şekilde yazmak istersek;

```
mkdir {OCA,SUB,MAR,NIS,MAY,HAZ,TEM,AGU,EYL,EKI,KAS,ARA}_{2017..2023}
```

Bu klasörlerin içine dosyalar oluştursak ve bu dosya isimlerin 1'den 100'e kadar gitsin istersek;

```
mkdir {OCA,SUB,MAR,NIS,MAY,HAZ,TEM,AGU,EYL,EKI,KAS,ARA}_{2017..2023}/file{1..100}
```

Dosya Silme

“rm” komutu ile silme işlemini gerçekleştirebiliriz.

```
rm file1.txt
```

Mesela içerisinde “2” geçen dosyaları silmek istersek;

```
rm *2*
```

file ile başlayıp 0 ile 9 arasında bir rakam ile biten txt dosyalarını silmek istersek;

```
rm file[0-9].txt
```

Klasör Silme

Klasör silme işlemi için yukarıda olduğu gibi kullanırsak;

```
rm project
```

```
usrht@usrht-VirtualBox:~$ rm project
rm: cannot remove 'project': Is a directory
```

Yukarıdaki gibi bir hata ile karşılaşırız.

Klasör silme işlemi için “**rmdir**” komutunu kullanırız.

```
rmdir project
```

Eğer project klasörünün içi boş değilse silme işlemi yine gerçekleşmez.

Bunun için

```
rm -r project
```

içindekilerle birlikte silme işlemi gerçekleşti. “-r” opsiyonuyla kullanırken dikkat olmak gerekir. Çünkü geri dönüşüm kutusuna atmadan direkt olarak silme işlemini yapar.

Silme işleminde geri dönüşüm kutusunda bulunmadığını görürüz. İnteraktif şekilde silme işlemi de gerçekleştirebiliriz. Bu şekilde adım adım bize sorulan silme adımlarını onaylayabiliriz.

```
usrht@usrht-VirtualBox:~$ mkdir project1
usrht@usrht-VirtualBox:~$ cd project1
usrht@usrht-VirtualBox:~/project1$ touch file{1..3}.txt
usrht@usrht-VirtualBox:~/project1$ ls
file1.txt file2.txt file3.txt
usrht@usrht-VirtualBox:~/project1$ cd ..
usrht@usrht-VirtualBox:~$ rm -ri project1
rm: descend into directory 'project1'? Y
rm: remove regular empty file 'project1/file3.txt'? Y
rm: remove regular empty file 'project1/file2.txt'? Y
rm: remove regular empty file 'project1/file1.txt'? Y
rm: remove directory 'project1'? Y
usrht@usrht-VirtualBox:~$
```

Kopyalama İşlemleri

“**cp**” komutu kullanılır

Önce “**hedef**” ve “**kaynak**” olarak iki ayrı klasör oluşturalım.

```
mkdir hedef kaynak
```

Bunların dışında **file1** isimli bir dosya ve **kaynak** klasörü içinde ise 3 ayrı dosya oluşturalım.

```
touch file1 kaynak/file{2..4}
```

Şimdi ise **file1** dosyasını **hedef** klasörü içerisine kopyalamak istersek;

```
cp file1 hedef/
```

Eğer aynı dosyayı farklı bir isimle mevcut konum içine kopyalamak istersek;

```
cp file1 file2.txt
```

Eğer wildcard kullanmak istersek, mesela file ile başlayan tüm dosyaları hedef klasörü altına kopyalamak gibi;

```
cp file* hedef/
```

Şimdi ise **folder1** klasörü ve onun altında **folder2** isimli klasörler oluşturalım.

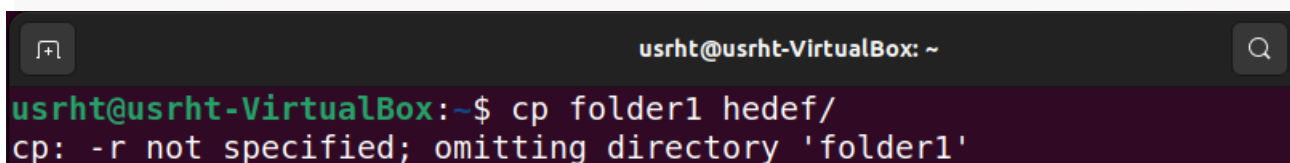
```
mkdir -p folder1/folder2
```

Bu oluşturduğumuz **folder2** klasörünün altında da **file5.txt** isimli bir dosya oluşturalım

```
mkdir -p folder1/folder2  
touch folder1/folder2/file5.txt
```

Eğer ki **folder1** klasörünü **hedef** klasörü içine taşımak istersek aşağıdaki komutu kullanırız. Ancak burada bir hata alırız.

```
cp folder1 hedef/
```

A terminal window with a dark background. The prompt is 'usrht@usrht-VirtualBox: ~'. The command entered is 'cp folder1 hedef/'. The output is 'cp: -r not specified; omitting directory 'folder1''.

```
usrht@usrht-VirtualBox: ~  
usrht@usrht-VirtualBox:~$ cp folder1 hedef/  
cp: -r not specified; omitting directory 'folder1'
```

Burada recursive (-r option) kullanılmalı.


```
cp -r folder1 hedef/
```

Bu opsiyonu kullandık çünkü tekrarlamalı bir işlem var burada.

Taşıma ve Yeniden Adlandırma

Taşıma işlemi için “**mv**” komutu kullanılır.

```
touch file1.txt  
mv file1.txt file_one.txt
```

Yukarıdaki komutlar çalıştırılırsa önce **file1.txt** dosyası oluşturulur. Daha sonra ise aynı konumda **file1.txt** dosyası **file_one.txt** ismiyle yer değişir. Yani **file1.txt** dosyası silinip **file_one.txt** dosyası onun yerine aynı şekilde oluşur.

Eğer ki bu **file_one.txt** dosyasını **Documents** klasörü altında taşımak istersek;

```
mv file_one.txt ~/Documents/
```

Eğer bir dosyayı Documents konumuna farklı bir isimle taşımak istersek;

```
mv file_one.txt ~/Documents/file_two.txt
```

Şimdi bir senaryo üzerinden gidelim.

new_folder klasörü oluşturalım.

Bu klasörün altında **folder1** isimli bir klasör daha oluşturalım.

Folder1 klasörü içinde 10 adet **file** ile başlayan dosya oluşturalım

ve bu **new_folder** klasörünü farklı isimle aynı yere kaydedelim.

```
mkdir -p new_folder/folder1  
touch new_folder/folder1/file{1..10}.txt  
mv new_folder/ new_folder_2
```

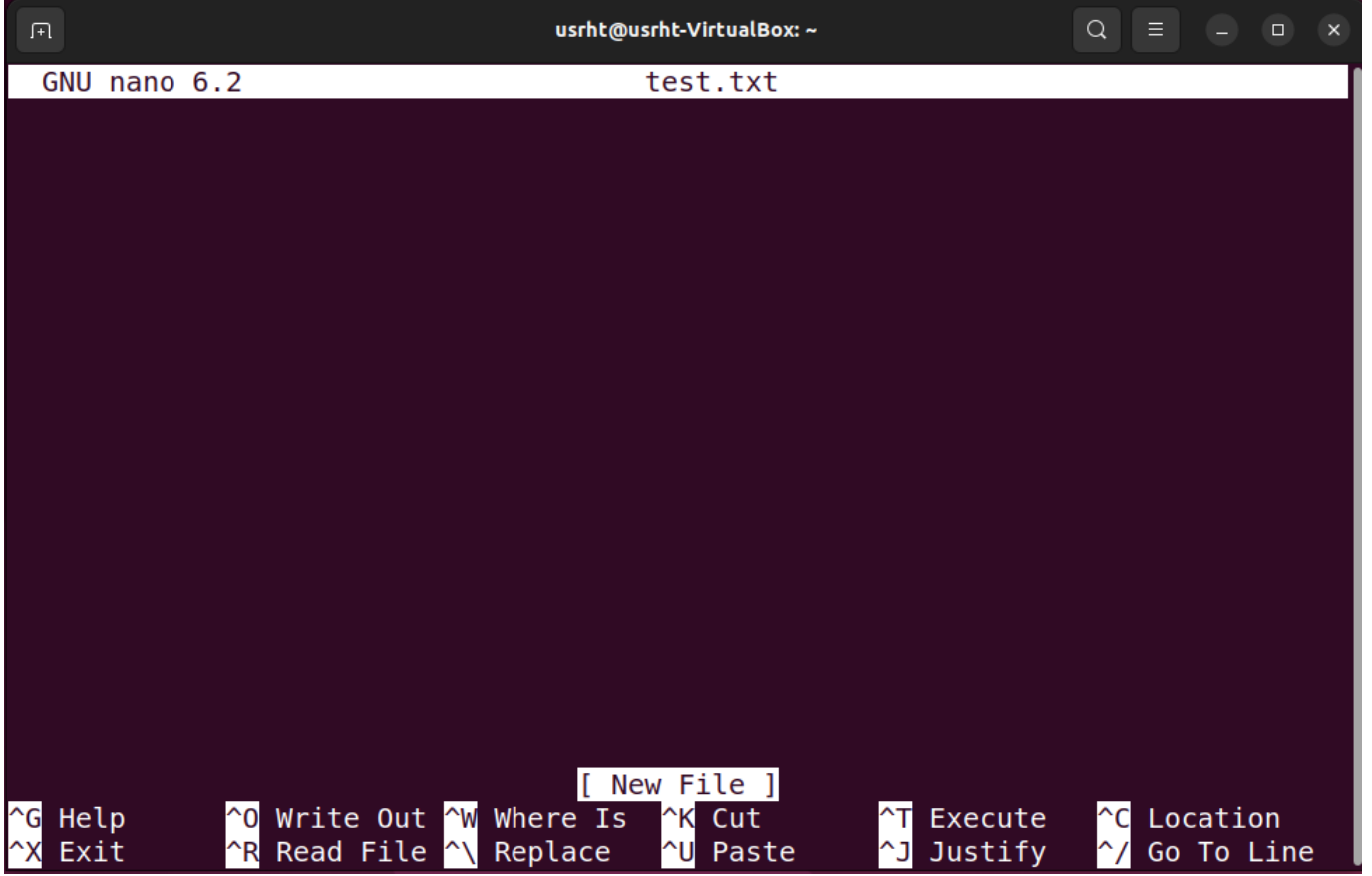
Yukarıda oluşturduğumuz bu **new_folder_2** klasörünü de farklı bir yere taşımak istersek;

```
Mv new_folder_2 ~/Documents/
```

Nano Editor ile Dosya Düzenleme

Var olmayan bir **test.txt** dosyası için;

```
nano test.txt
```



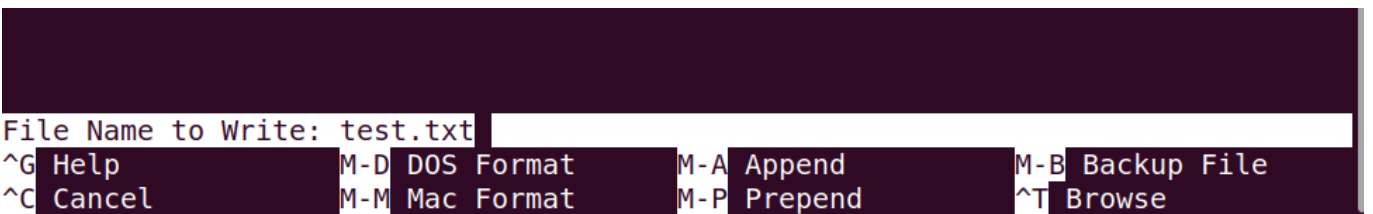
Burada “^” **CTRL** anlamına gelmektedir.

CTRL + G ile Nano kullanımıyla ilgili bir manuel içerisine erişiriz.

CTRL + O için tıklamadan önce birşeyler yazalım.



Daha sonra **CTRL + O** tıklandığında panelin altında bu şekilde bir ibare çıkacaktır. Buradan enter tıklanırsa aynı dosya üzerine bu bilgiler yazılır.



CTRL + R ile **date.txt** isimli dosyadan okuma yapmak istersek;

```
File to insert [from ./]: date.txt
^G Help          M-F New Buffer      ^X Execute Command
^C Cancel        M-N No Conversion  ^T Browse

GNU nano 6.2      test.txt *
Cum 03 Kas 2023 16:59:05 +03
|
```

date.txt içindeki tüm verileri ekrana aktarmış oluruz.

Eğer ekrandaki yazılan metinde arama yapmak istersek **CTRL + W** kullanılır. Burada **ALT + C** yaparsak Case-sensitive seçeneğini de aramaya eklemiş oluruz.

```
Search [Case Sensitive]:
^G Help          M-C Case Sens  M-B Backwards  ^P Older       ^T Go To Line
^C Cancel        M-R Reg.exp.   ^R Replace     ^N Newer
```

Replace işlemi için **CTRL + R** kullanılır.

Mesela nano ile python kodu yazmak istersek uzantısını **.py** yaparız.

```
nano test.py
```

Nano editör içerisinde **CTRL + K** ile kesme işlemi yapabilir, **CTRL + U** ile ise yapıştırma işlemi yapabiliriz.

Justify özelliği ile fare imleci nerede kaldıysa onun altındakileri iki yana yaslama işlemi yapılabilir. Bunun için **CTRL + J** kullanılır.

Mouse imlecimizin nerede kaldığını öğrenmek için ise **CTRL + C** kullanılabilir. Burada bize imlecimiz nerede kaldıysa orası hakkında bilgi verir. Aşağıdaki şekilde görebiliriz. Bunu çok yoğun dosyalarda spesifik bir yeri başka birinin bulabilmesi için kullanabiliriz.

```
GNU nano 6.2      test.txt *
I love Python and Linux
Hello world

[ line 3/4 (75%), col 8/12 ( 66%), char 32/37 (86%) ]
^G Help          ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit          ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

Tüm satırların %75 inde bulunduğu ve sütun olarak 8. sütunda bulunduğu, son olarak ise tüm karakterlerin %86'sı olan 32. karakter olduğunu belirtir.

Go To Line özelliği ise **CTRL + /** ile satır ve sütun bilgisini girerek o noktaya gidebiliriz.

M-U (ALT + U) ile ise yaptığımız bir işlem için undo (geri alma) işlemi yapabiliriz. Geri alınan işlemi tekrar yapmak istersek **M-E (ALT + E)** kullanılır.

M-A (ALT + A) ile ise highlight işlemi uygulanır. Mesela highlight edilen bu kısım **M-6 (ALT + 6)** ile kopyalanıp **CTRL + U** ile yapıştırılabilir.

Locate Komutu

Sistem üzerinde yer alan dosyaları arama, dosyaların hangi klasör içerisinde bulunduğunu bulmamızı sağlar.

Eğer yüklü değilse ilk olarak **plocate** paketini yüklüyoruz.

```
sudo apt install plocate
```

```
usrht@usrht-VirtualBox:~$ locate --help
Usage: plocate [OPTION]... PATTERN...

  -b, --basename          search only the file name portion of path names
  -c, --count             print number of matches instead of the matches
  -d, --database DBPATH  search for files in DBPATH
                        (default is /var/lib/plocate/plocate.db)
  -i, --ignore-case       search case-insensitively
  -l, --limit LIMIT       stop after LIMIT matches
  -0, --null              delimiter matches by NUL instead of newline
  -N, --literal           do not quote filenames, even if printing to a tty
  -r, --regexp            interpret patterns as basic regexps (slow)
  --regex                interpret patterns as extended regexps (slow)
  -w, --wholename         search the entire path name (default; see -b)
  --help                 print this help
  --version              print version information
```

İçerisinde “**fileA**” geçen dosyaları arattığımızda;

```
usrht@usrht-VirtualBox:~$ locate /fileA
/home/usrht/fileA.txt
/home/usrht/fileAA.txt
```

Sonu **.conf** uzantısı ile biten dosyaları getirmek istediğimiz;

```
locate *.conf
```

Yukarıda **--help** ile gelen kısım için “**-d, --database DBPATH**” kısmı için “**/var/lib/plocate/plocate.db**” içerisinde bir db dosyası bulunmaktadır.

Bu dosyada dosyaların konum bilgileri tutuluyor ve günlük güncelleniyor. Ama günlük güncellendiği için yeni eklenen bir şey dosyayı bulmak istediğimizde sonraki günü mü beklememiz gerekiyor hatalı almamak için?

Bunun önüne geçebilmek için bu dosya gün içinde silinmiş veya değişmiş mi yoksa hala mevcut mu kontrolü için “-e” opsiyonu (exist) kullanılır;

```
locate -e *.conf
```

Ama hala yeni oluşan klasörü göremiyoruz.

Locate komutu kullanımında case-sensitive şekilde çalışıyor, bunun önüne geçebilmek için “-i” opsiyonu kullanılır.

```
locate -i *.CONF
```

Dosya sayısını limitlemek istersek “--limit” veya “-l” opsiyonu kullanırız. İlk 10 sonucu göstermek istersek;

```
locate -limit 10 *.conf
```

Yukarıda da belirttiğimiz üzere günlük olarak locate DB içerisinde dosya bilgileri tutuluyordu ve günlük olarak yenileniyordu. Ancak burada yeni dosya eklenmesi durumu için bir çözüme ihtiyacımız var çünkü yeni dosya eklenmesi halinde bir sonraki günü beklememiz gerekir “**locate**” ile ulaşabilmek için.

Burada “**locate**” komutunu çalıştırmadan önce aşağıdaki güncelleme komutunu çalıştırsak bu sorunu çözmüş oluruz.

```
sudo updatedb
```

```
usrht@usrht-VirtualBox:~$ touch new_file_locate.txt
usrht@usrht-VirtualBox:~$ locate new_file_locate
usrht@usrht-VirtualBox:~$
usrht@usrht-VirtualBox:~$ sudo updatedb
[sudo] password for usrht:
usrht@usrht-VirtualBox:~$ locate new_file_locate
/home/usrht/new_file_locate.txt
usrht@usrht-VirtualBox:~$
```

- Yukarıda ilk olarak “new_file_locate.txt” isminde dosya oluştururuz.
- Daha sonra “**locate**” komutunu çalıştırınca herhangi bir sonuç elde etmeyiz.
- “**sudo updatedb**” komutunu çalıştırıp şifre girişi yaptıktan sonra tekrar “locate” komutu çalıştırsak istediğimiz sonucu elde ederiz.

Find Komutu

Dosya ve klasör arama yapmamızı sağlar. Herhangi bir parametre girmememiz durumunda bulunduğumuz konum altındaki tüm dosya ve klasörleri listelemeye yarar.

Eğer sadece **hedef** klasörü altındakileri almak istersem;

```
find ./hedef/
```

```
usrht@usrht-VirtualBox:~$ find ./hedef
./hedef
./hedef/folder1
./hedef/folder1/folder2
./hedef/folder1/folder2/file1.txt
usrht@usrht-VirtualBox:~$
```

Eğer maximum 3 klasör alta kadar göstermek istersek;

```
find ./hedef/ -maxdepth 2
```

```
usrht@usrht-VirtualBox:~$ find ./hedef/ -maxdepth 2
./hedef/
./hedef/folder1
./hedef/folder1/folder2
usrht@usrht-VirtualBox:~$
```

Eğer bu konum altında sadece dosyaları göstermek istersek;

```
find ./hedef/ -type f
```

Sadece klasörleri göstermek istersek;

```
find ./hedef/ -type d
```

İsim ile arama yapmak istersek “**hedef**” klasörü altında;

```
find ./hedef/ -name "file1.txt"
```

```
usrht@usrht-VirtualBox:~$ find ./hedef/ -name "file1.txt"
./hedef/folder1/folder2/file1.txt
usrht@usrht-VirtualBox:~$
```

Regex kullanarak tüm dosyaların içinde arama yapmak istersek;

```
usrht@usrht-VirtualBox:~$ find -name "file?.txt"
./folder1/folder2/file1.txt
./hedef/folder1/folder2/file1.txt
./file3.txt
./fileA.txt
./file2.txt
usrht@usrht-VirtualBox:~$
```

“file” ile başlayan tüm dosyaları listelemek istersek;

```
find -name "file*" -type f
```

Dosya boyutuna göre de arama yapılabilir. 100KB altındaki dosyaları listeleyelim.

```
find / -type f -size -100k
```

100MB'dan büyük dosyaları listelersek;

```
find / -type f -size +100M
```

10MB'dan büyük 20MB'dan küçük dosyaları listelersek;

```
find / -type f -size +10M -size -20M
```

Eğer default olarak size opsiyonları arasına bir operatör belirtmezsek AND olarak algılar. OR ile belirtmek istersek “-o” opsiyonu ile belirtiriz.

```
find / -type f -size +10M -o -size -20M
```

Burada yetki problemi ile karşılaşarsak “sudo” ile çalıştırabiliriz. Tabi burada çok fazla dosya olacak. Bunun yerine dosya sayısını öğrenmek istersek;

```
Sudo find / -type f -size +10M -o -size -20M | wc -l
```

```
usrht@usrht-VirtualBox:~$ sudo find / -type f -size +10M -o -size -10M | wc -l
[sudo] password for usrht:
find: '/proc/2243/task/2243/fd/6': No such file or directory
find: '/proc/2243/task/2243/fdinfo/6': No such file or directory
find: '/proc/2243/fd/5': No such file or directory
find: '/proc/2243/fdinfo/5': No such file or directory
find: '/run/user/1000/doc': Permission denied
find: '/run/user/1000/gvfs': Permission denied

594193
```

Bir klasör oluşturalım. 100kB üstü 5MB altı ve max derinliği 2 olan tüm dosyaları bu klasöre kopyalayalım.

İlk olarak dosyaları nasıl buluruz;

```
usrht@usrht-VirtualBox:~$ find / -maxdepth 2 -type f -size +100k -size -5M
find: '/lost+found': Permission denied
find: '/root': Permission denied
/boot/config-6.2.0-26-generic
/boot/memtest86+.elf
/boot/config-6.2.0-34-generic
/boot/memtest86+_multiboot.bin
/boot/memtest86+.bin
```

Find komutu içerisinde “**exec**” komutu ile bulduğumuz dosyalar üzerinde işlem yapabiliyoruz. Yine hatırlatma olarak yetki gerektiren dosyalar için bu komutu “**sudo**” ile çalıştırırız. Sonda kullanılan “**;**” komutu ile ise komutun bittiğini belirtiriz.

```
find / -maxdepth 2 -type f -size +100k -size -5M -exec cp {} ~/new_folder/ \;
```

Kritik yerlerde bul ve sil gibi bir işlem yapmak istersek problem olabilir.

Burada bul, benden onay al ve sil şeklinde bir yöntemle gideriz.

```
find / -maxdepth 2 -type f -size +100k -size -5M -ok {} ~/new_folder/ \;
```

```
usrht@usrht-VirtualBox:~$ find / -maxdepth 2 -type f -size +100k -size -5M -ok cp {} ~/new_folder/ \;
find: '/lost+found': Permission denied
find: '/root': Permission denied
< cp ... /boot/config-6.2.0-26-generic > ? Y
< cp ... /boot/memtest86+.elf > ? Y
< cp ... /boot/config-6.2.0-34-generic > ? N
< cp ... /boot/memtest86+_multiboot.bin > ? N
< cp ... /boot/memtest86+.bin > ? Y
usrht@usrht-VirtualBox:~$
```

Şimdi ise project klasörü ve altında “folder” ön ek ile başlayıp son kısım olarak 1’den 500’e kadar değer alan klasörler ekleyelim. Bu klasörlerin altında da “file” ön ekiyle başlayıp son kısım olarak 1’den 100’e kadar “txt” dosyaları ekleyelim.

```
mkdir -p project/folder{1..500}
touch project/folder{1..500}/file{1..100}.txt
```

Şimdi ise “**project**” klasörü içerisinde “**file1.txt**” dosyalarını arayalım.

```
find project/ -name file1.txt
```

Eğer 1 ile 500 arasında random bir sayı almak istersek “**shuf**” komutunu kullanabiliriz.

```
shuf -i 1-500 -n 1
```

```
usrht@usrht-VirtualBox:~$ shuf -i 1-500 -n 1
366
usrht@usrht-VirtualBox:~$ shuf -i 1-500 -n 1
147
```

Random belirlediğim bir dosya adı içerisinde “**gizli.txt**” isimli dosya oluşturalım. Burada “**command substitution**” yöntemi kullanacağız.

```
touch project/folder$(shuf -i 1-500 -n 1)/gizli.txt
```


Yukarıda rastgele bir klasör içinde oluşturulan “**gizli.txt**” isimli dosyayı bulmak istersek;

```
find project/ -name gizli.txt
```

```
usrht@usrht-VirtualBox:~$ touch project/folder$(shuf -i 1-500 -n 1)/gizli.txt
usrht@usrht-VirtualBox:~$ find project/ -name gizli.txt
project/folder18/gizli.txt
```

Eğer bu oluşturulan dosyayı da “**find**” komutu sonucunda mevcut konuma kopyalamak istersek;

```
find project/ -name gizli.txt -exec cp {} . \;
```

Dosya Görüntüleme

“**cat**” komutu ile dosya içeriklerini görebiliyorduk.

Eğer output için “**tac**” kullanırsak bu sıralamayı sondan başa alırız.

```
usrht@usrht-VirtualBox:~$ cat error.txt
date: invalid date 'skhkshkdh'
date: invalid date 'kks'
cat: invalid option -- 'a'
Try 'cat --help' for more information.
usrht@usrht-VirtualBox:~$ tac error.txt
Try 'cat --help' for more information.
cat: invalid option -- 'a'
date: invalid date 'kks'
date: invalid date 'skhkshkdh'
usrht@usrht-VirtualBox:~$
```

Burada satırları tersten yazdırmak istersek “**rev**” komutu kullanırız. İki şekilde de kullanabiliriz. Birincisi komutu direkt olarak yazmak, diğeri ise “**cat**” komutunun output değerine işlem uygulamaktır.

```
usrht@usrht-VirtualBox:~$ rev error.txt
'hdkhskhks' etad dilavni :etad
'skk' etad dilavni :etad
'a' -- noitpo dilavni :tac
.noitamrofni erom rof 'pleh-- tac' yrT
usrht@usrht-VirtualBox:~$ cat error.txt | rev
'hdkhskhks' etad dilavni :etad
'skk' etad dilavni :etad
'a' -- noitpo dilavni :tac
.noitamrofni erom rof 'pleh-- tac' yrT
usrht@usrht-VirtualBox:~$
```

“**head**” komutu ile bir dosyayı baştan itibaren ele alabiliriz. Örneğin çok fazla verinin olduğu bir dosyayı ele alırsak, bu veri dosyasında bu dosyada ilk 50 satırı getirmek istersek (default 10);

```
head -n 50 error.txt
```

“**tail**” komutu ile bir dosyayı sondan itibaren ele alabiliriz.

```
tail -n 50 error.txt
```

“**less**” komutu ile ise daha kullanışlı biçimde bir sayfalama ile ilerleyebiliriz.

```
less error.txt
```

Bir dosyayı “**cat**” komutu ile görüntüleyip, çıktısının “**head**” komutu ile ilk 30 satırını daha sonra bunun da çıktısından “**tail**” komutu ile son 5 satırını alabiliriz. Yani kısacası tüm verinin 25 ile 30. satırları arasını ele alabiliriz.

```
cat error.txt | head -n 30 | tail -5
```

Veri Düzenleme İşlemleri

“**sort**” komutu ile alfabetik veya nümerik olarak sıralama yapabiliriz.

```
cat error.txt  
sort error.txt
```

Yukarıda görüldüğü üzere önce dosyayı görüntülüyoruz. Daha sonra ise sıralama işlemi yapıyoruz. Sonuç aşağıdaki gibi olacaktır.

```
usrht@usrht-VirtualBox:~$ cat error.txt  
date: invalid date 'skhkshkdh'  
date: invalid date 'kks'  
cat: invalid option -- 'a'  
Try 'cat --help' for more information.  
usrht@usrht-VirtualBox:~$ sort error.txt  
cat: invalid option -- 'a'  
date: invalid date 'kks'  
date: invalid date 'skhkshkdh'  
Try 'cat --help' for more information.
```

Sıralamayı tersten yapmak istersek “**-r**” (veya “**--reverse**”) opsiyonunu kullanırız.

```
sort -r error.txt
```

Eğer bir dosyada tekrar eden objeleri ayırmak istersek “**-u**” opsiyonu (veya “**--unique**”) kullanılabilir;

```
sort -u error.txt
```

Eğer “**numbers.txt**” dosyası içerisinde rastgele yazılmış sayıları sıralamak istersek aynı şekilde ilerleriz.

```
sort numbers.txt
```

```
usrht@usrht-VirtualBox:~$ sort numbers.txt
```

```
1
1
1
11
2
2
3
3
33
4
4
5
5
6
78
```

Yukarıda görüldüğü üzere bir yanlışlık oluştu. Burada ilk harfleri baz aldı ve nümerik olarak sıralamadı. Eğer nümerik olarak sıralanmasını istersek “-n” opsiyonunu (veya “—**numeric-sort**”) kullanırız.

```
sort -n numbers.txt
```

```
usrht@usrht-VirtualBox:~$ sort -n numbers.txt
```

```
1
1
1
2
3
3
4
4
5
5
6
11
33
78
```

Görüldüğü üzere nümerik olarak sıralama gerçekleşti..

Sütun veya satırlara göre işlem yapmak istersek;

İlk olarak bir örnek üzerinden gidelim;

```
usrht@usrht-VirtualBox:~$ ls -l /var | head -n 20
total 48
drwxr-xr-x  2 root root    4096 Kas 24 11:06 backups
drwxr-xr-x 18 root root    4096 Kas 14 15:01 cache
drwxrwsrwt  2 root whoopsie 4096 Ağu 8 01:55 crash
drwxr-xr-x 69 root root    4096 Kas 23 17:22 lib
drwxrwsr-x  2 root staff   4096 Nis 18 2022 local
lrwxrwxrwx  1 root root      9 Eki 11 22:36 lock -> /run/lock
drwxrwxr-x 13 root syslog   4096 Kas 29 10:06 log
drwxrwsr-x  2 root mail    4096 Ağu 8 01:52 mail
drwxrwsrwt  2 root whoopsie 4096 Ağu 8 01:55 metrics
drwxr-xr-x  2 root root    4096 Ağu 8 01:52 opt
lrwxrwxrwx  1 root root      4 Eki 11 22:36 run -> /run
drwxr-xr-x 12 root root    4096 Ağu 8 01:59 snap
drwxr-xr-x  7 root root    4096 Ağu 8 01:54 spool
drwxrwsrwt 11 root root    4096 Kas 29 10:57 tmp
usrht@usrht-VirtualBox:~$
```

Burada 20 adet dosya aldık örnek olarak;

Bu sütunların ne anlama geldiğini daha önceden yazmıştık. Burada **4096** bayt ile başlayan ve dosya boyutunu veren kısma göre sıralama yapmak ve daha sonrasında ise “reverse” işlemi uygulayıp tersten sıralamak istersem. Burada “Human readable” kısmını da gözden kaçırmayalım ve opsiyonlarımıza “-h” opsiyonunu da ekleyelim.

```
usrht@usrht-VirtualBox:~$ ls -lh /var | head -n 20 | sort -k 5hr
drwxrwsrwt  2 root whoopsie 4,0K Ağu 8 01:55 crash
drwxrwsrwt  2 root whoopsie 4,0K Ağu 8 01:55 metrics
drwxrwsr-x  2 root mail    4,0K Ağu 8 01:52 mail
drwxrwsr-x  2 root staff   4,0K Nis 18 2022 local
drwxrwxrwt 11 root root    4,0K Kas 29 10:57 tmp
drwxrwxr-x 13 root syslog   4,0K Kas 29 10:06 log
drwxr-xr-x 12 root root    4,0K Ağu 8 01:59 snap
drwxr-xr-x 18 root root    4,0K Kas 14 15:01 cache
drwxr-xr-x  2 root root    4,0K Ağu 8 01:52 opt
drwxr-xr-x  2 root root    4,0K Kas 24 11:06 backups
drwxr-xr-x 69 root root    4,0K Kas 23 17:22 lib
drwxr-xr-x  7 root root    4,0K Ağu 8 01:54 spool
lrwxrwxrwx  1 root root      9 Eki 11 22:36 lock -> /run/lock
lrwxrwxrwx  1 root root      4 Eki 11 22:36 run -> /run
total 48K
usrht@usrht-VirtualBox:~$
```

Eğer ay bilgisine göre sıralamak istersek (“M” ay bilgisine göre);

```
ls -lh /var | head -n 20 | sort -k 6M
```

Dosya Ve Klasör Arşivleme

Dosya zipleme işlemi ile başlayalım. Linux taraflı isimlendirmesi “tar” olarak geçiyor.

```
tar -cvf test_arhive.tar file3 file4
```

“-c” (create) opsiyonu ile yeni bir arşiv oluşturuyoruz. “-v” ile (verbose) bilgi verme işlemi uygulanır. “-f” ile ise isim verme işlemi uygulanır.

```
usrht@usrht-VirtualBox:~/kaynak$ tar -cvf test_archive.tar file3 file4
file3
file4
usrht@usrht-VirtualBox:~/kaynak$
```



file3



file4



test_
archive.tar

Eğer bu TAR dosyasının içinde ne olduğunu görmek istersek “-t” (--test-label) opsiyonunu ve “-f” ile de dosyanın adını yazabiliriz.

```
tar -tf test_archive.tar
```

Eğer TAR dosyasının içindekileri çıkartmak istersek “-x” (extract) opsiyonunu kullanırız.

```
tar -xvf test_archive.tar
```

Şimdi ise TAR dosyası **sıkıştırma** işlemini görelim.

İlk olarak yeni bir TAR dosyası oluşturalım.

```
tar -cvf archive.tar file[1-4].txt
```

Burada arşiv dosyası oluşturduk ancak henüz zipleme işlemi yapmadık. Burada 2 komuttan yardım alıyoruz. Bunlar “**gzip**” ve “**bzip2**” komutlarıdır.

Şimdi zipleme işlemini uygulayalım. Ancak ilk halini aşağıda görelim ve ondan sonra ilerleyelim.

```
usrht@usrht-VirtualBox:~$ ls -lh
total 124K
-rw-rw-r-- 1 usrht usrht 10K Ara 1 10:53 archive.tar
-rw-rw-r-- 1 usrht usrht 29 Kas 3 15:49 date_tee.txt
-rw-rw-r-- 1 usrht usrht 29 Kas 3 16:59 date.txt
-rw-rw-r-- 1 usrht usrht 4 Kas 3 16:59 days.txt
```

```
usrht@usrht-VirtualBox:~$ gzip archive.tar
usrht@usrht-VirtualBox:~$ ls -lh
total 116K
-rw-rw-r-- 1 usrht usrht 138 Ara 1 10:53 archive.tar.gz
-rw-rw-r-- 1 usrht usrht 29 Kas 3 15:49 date_tee.txt
-rw-rw-r-- 1 usrht usrht 29 Kas 3 16:59 date.txt
-rw-rw-r-- 1 usrht usrht 4 Kas 3 16:59 days.txt
```

Yukarıda görüldüğü üzere dosyamız sıkıştırılmış haldedir ve 10K’dan 138B kadar düşüş oldu.

Yukarıda yaptığımız zipleme işlemini geri almak için;

```
gunzip archive.tar.gz
```

Aynı komut kullanımı “**bzip2**” içinde geçerlidir.

Eğer 2 adımlı şekilde gitmek istemeyip direkt olarak zipleme işlemini de yapmak istersek;

```
tar -cvzf archive.tar.gz file[1-4].txt
```

“**bzip**” için bu işlem uygulanırsa;

```
tar -cvjf archive.tar.gz file[1-4].txt
```

bu kullanım ile unzip işlemi uygulanmak istenirse;

```
tar -xvzf archive.tar.gz
```

```
tar -xvjf archive.tar.gz
```

Klasör Ve Dosya İzin İşlemleri

Daha önceden dosya ve klasörler ile ilgili izin türleri ve kullanıcılarından bahsetmiştik. Örneğin “**-rw-rw-r--**”, burada read, write execute gibi izin tipleri vardır.

```
usrht@usrht-VirtualBox:~$ ls -l
total 116
-rw-rw-r-- 1 usrht usrht 138 Ara 1 10:53 archive.tar.gz
-rw-rw-r-- 1 usrht usrht 29 Kas 3 15:49 date_tee.txt
-rw-rw-r-- 1 usrht usrht 29 Kas 3 16:59 date.txt
-rw-rw-r-- 1 usrht usrht 4 Kas 3 16:59 days.txt
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Desktop
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Documents
drwxr-xr-x 2 usrht usrht 4096 Eki 11 23:14 Downloads
-rw-rw-r-- 1 usrht usrht 130 Kas 1 13:42 error.txt
```

- “**r**” = read olarak yani okuma izni için kullanılır
- “**w**” = write olarak yani yazma izni için kullanılır
- “**d**” = directory olarak yani bunun bir klasör olduğunu belirtir
- “**x**” = execute olarak yani eğer bu işlenebilir bir yapı olursa bunu işleyebilme iznini belirtir.

İlk harf olan “**d**” veya “**-**” kısmını geçecek olursak kalan kısmı 3 parçaya ayırabiliriz.

rwX – rwX – rwX

İlk üçlü sahibin kullanma yetkisini, ikinci üçlü dosyanın veya klasörün grubunun kullanım yetkilerini, sonuncu ise diğerlerin kullanım yetkisini belirtir.

Eğer yetki değişimi yapmak istersek aşağıdaki tabloyu kullanabiliriz.

7	rwX	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wX	011
2	-w-	010
1	--x	001
0	---	000

Yukarı görüldüğü üzere bu tabloyu yetkilendirme için kullanırız. Aşağıdaki gibi “chmod” (**changemod**) ile kullanılabilir.

```
chmod 777
```

Şimdi “**file2.txt**” dosyası üzerinden tüm yetkileri kaldıralım. Aşağıda yapılan işlemde aslında 0-0-0 olarak düşünebiliriz. Yani tablodaki en alt satırdan 3 adet kullandık.

```
sudo chmod 000 file2.txt
```

```
-rw-rw-r-- 1 usrht usrht 0 Kas 13 13:05 file2
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file22.txt
----- 1 usrht usrht 0 Kas 10 17:39 file2.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file33.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file3.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 fileAA.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 fileA.txt
```

Şimdi bu file2.txt dosyasını görüntülemek istersek yetki hatası alırız, çünkü tüm yetkileri kaldırdık.

```
usrht@usrht-VirtualBox:~$ cat file2.txt
cat: file2.txt: Permission denied
usrht@usrht-VirtualBox:~$
```

Eğer tüm yetkileri vermek istersek;

```
sudo chmod 777 file2.txt
```

```
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file11.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 13 13:05 file2
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file22.txt
-rwxrwxrwx 1 usrht usrht 0 Kas 10 17:39 file2.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file33.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file3.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 fileAA.txt
```

Yukarıda görüldüğü üzere tüm yetkiler verilmiştir.

Mevcut kullanıcının tüm yetkileri olsun, grup sadece okuma yetkisi olsun ve diğerlerinin hiçbir yetkisi olmasın istersek;

```
sudo chmod 704 file2.txt
```

```
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file11.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 13 13:05 file2
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file22.txt
-rwxr----- 1 usrht usrht 0 Kas 10 17:39 file2.txt
-rw-rw-r-- 1 usrht usrht 0 Kas 10 17:39 file33.txt
```

Mesela kendimize herhangi bir yetki olmadığı senaryoyu düşünelim. Yani aşağıdaki komutu çalıştırdık varsayabiliriz.

```
sudo chmod 000 file2.txt
```

Bu durumda herhangi bir işlem yapamayız bu dosya için. Ancak burada kendi kullanıcıma “**Executable**” yetkisi eklemek istersem;

```
sudo chmod +x file2.txt
```

Grep Komutu İle Filtreleme

Bir veriyi filtrelemek için “**grep**” komutu kullanılır.

hello.txt” dosyası içerisinde işlem yapabiliriz.

```
usrht@usrht-VirtualBox:~$ cat hello.txt
Cum 13 Eki 2023 16:50:27 +03
Cum 13 Eki 2023 17:01:35 +03
Cum 13 Eki 2023 17:07:53 +03
Çrş 01 Kas 2023 13:34:34 +03
Çrş 01 Kas 2023 13:35:55 +03
Çrş 01 Kas 2023 13:34:34 +03
Çrş 01 Kas 2023 13:35:55 +03
```

```
cat hello.txt | grep 17
grep 17 hello.txt
```

```
usrht@usrht-VirtualBox:~$ cat hello.txt | grep 17
Cum 13 Eki 2023 17:01:35 +03
Cum 13 Eki 2023 17:07:53 +03
```

Eğer hangi satırda eşleştiği bilgisini de almak istersek “**-n**” opsiyonunu kullanırız.

```
grep 17 -n hello.txt
```

```
usrht@usrht-VirtualBox:~$ grep 17 -n hello.txt
2:Cum 13 Eki 2023 17:01:35 +03
3:Cum 13 Eki 2023 17:07:53 +03
```


Eğer büyük-küçük harf duyarlılığını kaldırmak istersek “-i” opsiyonunu kullanırız.

```
grep test -i hello.txt
```

Filtrelerde RegEx kullanılabilir.

Satır “**web**” sözcüğü ile başlasın istersek;

```
grep ^web hello.txt
```

Eğer “**web**” ile bitsin istersek;

```
grep web$ hello.txt
```

“**grep**” komutunun biraz daha gelişmiş hali olan “**egrep**” ile yapacak olursak ve “**web**” ile veya “**mobil**” ile başlayan satırları görmek istersek;

```
egrep '^[web|mobil]' hello.txt
```

İlk karakter “a” veya “A” ve 2. karakter “s” veya “S” olanları getirmek istersek;

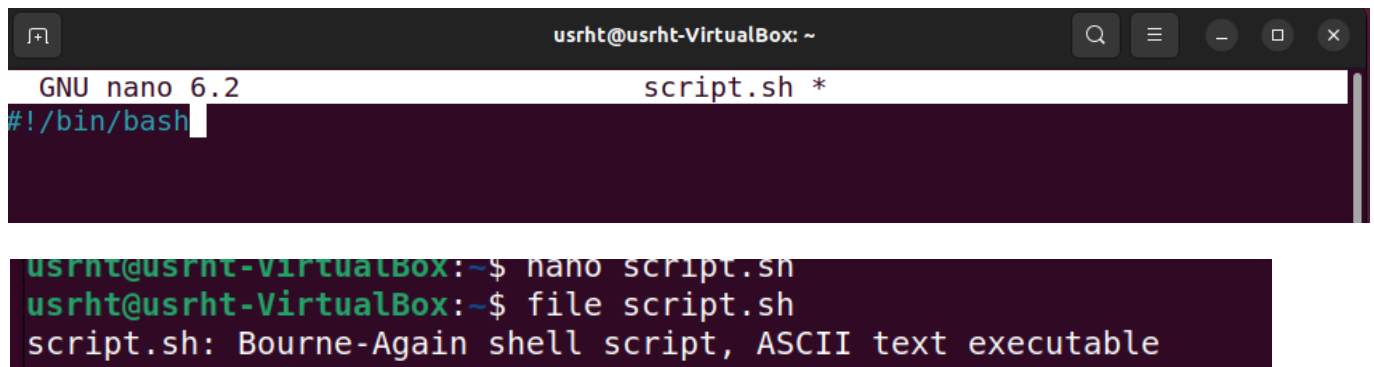
```
egrep '^[a,A][s,S]' hello.txt
```

Bash Script Oluşturma (Temel Seviyede)

İlk olarak **nano** ile **.sh (shell script)** uzantılı bir dosya oluşturalım.

NOT : *sh* uzantılı bir dosya, unix kabuğu tarafından çalıştırılacak bilgisayar programını içeren bir betik dili komut dosyasıdır.

Bu şekilde çalıştırılan dosyalar açıldığında en başta “**interpreter**” belirtiriz. Burada bir araya getirdiğimiz komutları hangi derleyici derleyecek bunu belirtiriz. Biz burada **bash** kullanıyoruz (Bourne-again shell script)



```
usrht@usrht-VirtualBox: ~  
GNU nano 6.2 script.sh *  
#!/bin/bash  
  
usrht@usrht-VirtualBox:~$ nano script.sh  
usrht@usrht-VirtualBox:~$ file script.sh  
script.sh: Bourne-Again shell script, ASCII text executable
```

Şimdi işlemlere geçelim;

```
usrht@usrht-VirtualBox: ~  
GNU nano 6.2 script.sh  
#!/bin/bash  
  
echo "This is first command for shell script"  
  
mkdir nfolder{1..5}  
sleep 2  
echo "Folder have been created"  
  
touch nfolder{1..5}/file{1..5}  
sleep 2  
echo "Files have been created"
```

Bu dosyayı çalıştırmadan önce aşağıdaki gibi yetki kontrolü yapalım.

```
drwxrwxr-x  360 usrht  usrht  20480 Kas  27  17:50 project  
drwxr-xr-x   2 usrht  usrht   4096 Eki  11  23:14 Public  
-rw-rw-r--   1 usrht  usrht    191 Ara   5  14:54 script.sh  
drwx-----   3 usrht  usrht   4096 Eki  11  23:14 snap  
-rw-rw-r--   1 usrht  usrht    573 Kas   3  17:05 takvim.txt
```

Görüldüğü üzere **script.sh** dosyası için **execute** yetkisi bulunmamaktadır.

Bu yetkisi vermek istersek kendi kullanıcımız için aşağıdaki şekilde ilerleyebiliriz.

```
sudo chmod +x script.sh  
sudo chmod 754 script.sh
```

Şimdi bu dosyayı çalıştıralım.

```
usrht@usrht-VirtualBox:~$ ./script.sh  
This is first command for shell script  
Folder have been created  
Files have been created  
usrht@usrht-VirtualBox:~$
```

Bu işlemten sonra ise **backup script** oluşturma işlemine geçelim.

Nano ile **backup.sh** dosyasını oluşturalım. İçinde ise sıkıştırma işlemini gerçekleştirelim. Bu komutları daha önceden de görmüştük.

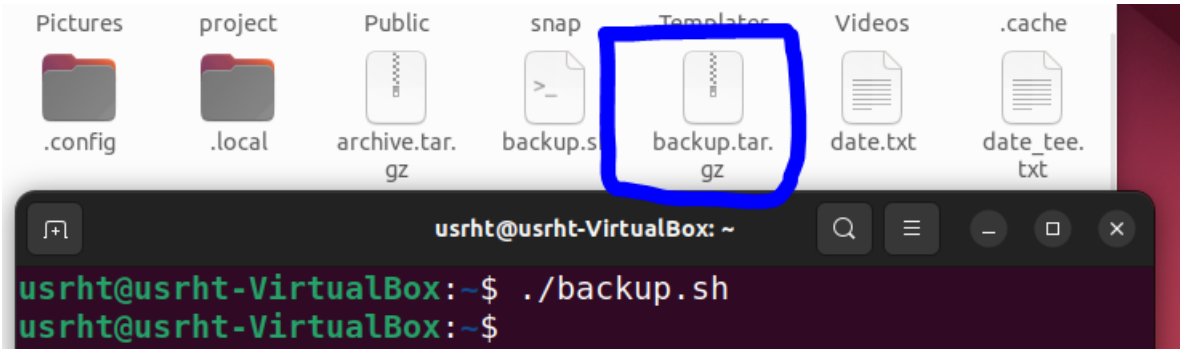
```
usrht@usrht-VirtualBox: ~  
GNU nano 6.2 backup.sh *  
#!/bin/bash  
  
tar -czf backup.tar.gz nfolder[1-5] 2>/dev/null
```

Yukarıda hatalı çıktı alması durumunda bunu bunu yazmaması için bir komutta ekledik.

Yine yetki kontrolü ve eklenmesi işlemini yapalım.

```
usrht@usrht-VirtualBox:~$ ls -l
total 144
-rw-rw-r-- 1 usrht usrht 138 Ara 1 10:53 archive.tar.gz
-rw-rw-r-- 1 usrht usrht 61 Ara 5 15:12 backup.sh
-rw-rw-r-- 1 usrht usrht 29 Kas 3 15:49 date_tee.txt
-rw-rw-r-- 1 usrht usrht 29 Kas 3 16:59 date.txt
```

```
sudo chmod +x backup.sh
sudo chmod 754 backup.sh
```



Hatırlarsak bir komut çalıştıracağımız zaman bu komutu “**echo \$PATH**” içerisindeki klasör yollarında arıyordu.

```
usrht@usrht-VirtualBox:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
```

Yukarıda **.sh** uzantılı backup dosyası oluşturmuştuk. Bunu da bu klasör yollarının içine eklemek isteyebiliriz.

Bunun için ilk olarak **bin** adında bir klasör oluşturalım. Daha sonra ise backup dosyamızı bu klasörün içine taşıyalım. En son olarak ise PATH değişkenini güncelleyelim.

```
mkdir bin
mv backup.sh bin/backup
export PATH="$PATH:~/bin"
```

Şimdi terminalden “**backup**” şeklinde bir komut çalıştırsak otomatik olarak “**bin/backup**” dosyasını çalıştıracaktır ve akabinde **tar.gz** dosyası oluşacaktır.

Cron İle Script Planlama Ve Çalıştırma

Bazı görevleri, çalıştırmak istediğimiz komutları veya scriptleri belirli plan dahilinde makineye yaptırmaya çalışacağız.

Örneğin sunucunun bazı kayıtlarının düzenli aralıklarla alınması veya bazı paketlerin güncellenmesi gibi durumlarda kullanılır. Bunun için “**Cron**” kullanılır.

Burada “**crontab**” komutunu kullanacağız.

Bu komutu kendi konfigürasyon dosyasında inceleyecek olursak;

```
GNU nano 6.2 /tmp/crontab.gJHB6E/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
```

Örneğin yukarıdaki örnek üzerinden gidersek;

```
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
```

Burada “**tar -zcf**” kısmını daha önce de gördük, zipleme işlemini burada yapıyoruz.

Burada “**0 5**” kısmı sabah **05:00** saatini temsil etmektedir.

Format olarak ise “**m h dom mon dow command**” şeklindedir

m : Dakika bilgisini verir (minute “0-59”)

h : Saat bilgisini verir (hour “0-23”)

dom : Ayın hangi günleri (day of month, tümü için * yazdık yukarıda “1-31”)

mon : Hangi ay bilgisi (month, yukarı 1 yazdık yani Ocak ayı “1-12”)

dow : Haftanın hangi günleri (day of week, tümü için * yazdık yukarıda “0-6” Pazar = 0 veya 7)

command : Çalışacak komut bilgisi

Örnek yapacak olursak, aşağıda 12. ay olan aralığın

```
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
10 13 3 12 6  backup
```

Her dakika date bilgisinin bir dosyaya yazılmasını istersek aşağıdaki şekilde kullanabiliriz.

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * date >> ~/cron date.txt
```

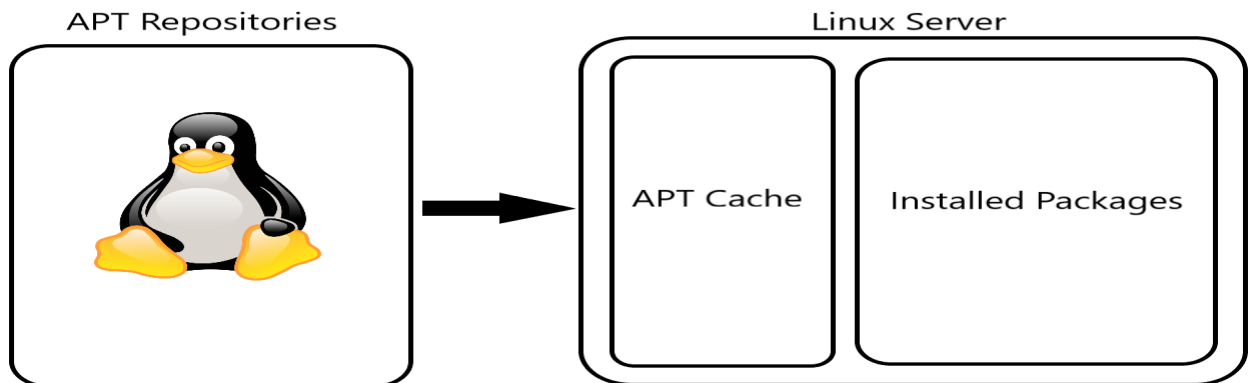
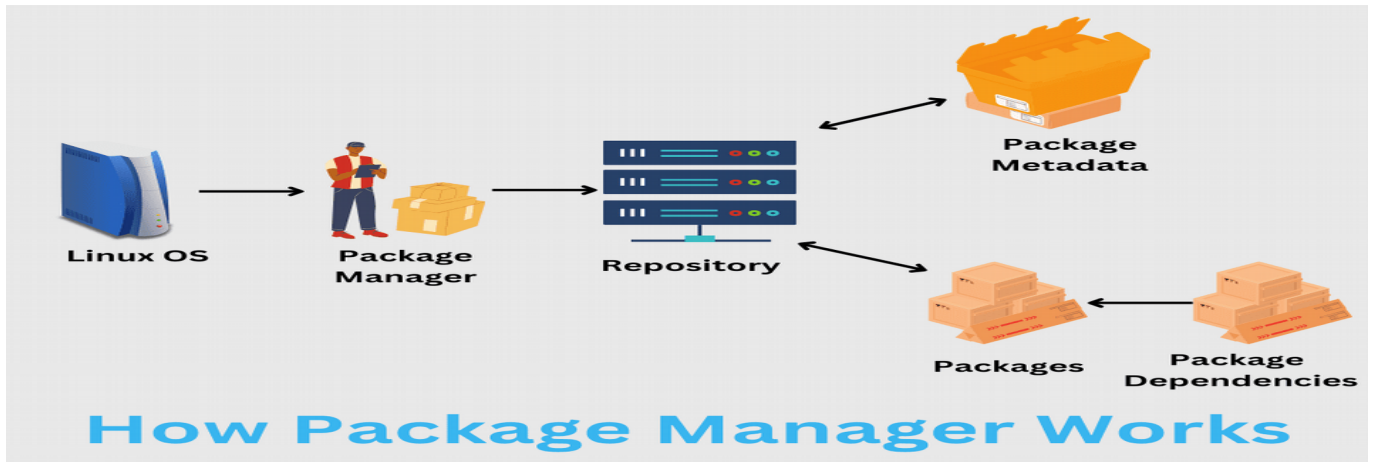
Bazı örnek senaryolar şu şekilde alınabilir;

- Her gün öğlen 12 ve 18 saatlerinde çalışacak
00 12,18 * * * date >> ~/cron_date.txt
- Sabah 9 akşam 18 arasında her saat çalışacak
00 09-18 * * * date >> ~/cron_date.txt
- Hafta için günler sabah 9 akşam 18 arası her saat çalışacak
00 09-18 * * 1-5 date >> ~/cron_date.txt

Linux Paket Yönetimi, Apt-cache

İnternet üzerinde paketlerin versiyonlarına ait listelerin tutulduğu bir depo bulunmaktadır. Yeni bir sistem kurduğumuzda bazı paketler kendi içerisinde yüklü şekilde gelmektedir.

İşletim sistemi içerisinde “**cache**” olarak isimlendirdiğimiz önbellek içerisinde bu paketlere ait tar dosyaları ve sistem üzerinde yüklü paketlere ait versiyonlara ait liste bulunmaktadır.



Yukarıda da görülen APT cache içerisinde yüklü paketlerin dışında diğer tüm paketler liste halinde bulunmaktadır. Eğer internet bağlantısı kesilirse buradan yine paket araması yapılabilir ama indirilemez. Tabi burada mevcutta yüklü olanlar için yükleme zip dosyaları bulunmaktadır.

Bu cache belleği güncelleyebiliriz. Yukarıda gördüğümüz cron ile de belirli aralıklarla bu listeyi güncelletebiliriz.

İlk olarak cache bellek komutlarına bakalım.

Aşağıdaki komut ile cache bellek üzerinde arama yapabiliriz. Eğer Burada internetle herhangi bir bağlantı olmadan sadece kendi belleğinde arama yapacaktır.

apt-cache search docx

```
usrht@usrht-VirtualBox:~$ apt-cache search docx
antiword - Converts MS Word files to text, PS, PDF and XML
diffoscope - in-depth visual diff tool for files, archives and directories
diffoscope-minimal - in-depth visual diff tool for files, archives and directories (minimal package)
docx2txt - Convert Microsoft OOXML files to plain text
libapache-poi-java - Apache POI - Java API for Microsoft Documents
libapache-poi-java-doc - Apache POI - Java API for Microsoft Documents (Documentation)
libghc-pandoc-dev - general markup converter - libraries
libghc-pandoc-doc - general markup converter - library documentation
libghc-pandoc-prof - general markup converter - profiling libraries
libimage-exiftool-perl - library and program to read and write meta information in multimedia files
mat2 - Metadata anonymisation toolkit v2
okular-backend-odt - Okular backend for ODT documents
pandoc - general markup converter
pandoc-data - general markup converter - data files
python3-docx - library for creating and updating Microsoft Word files (Python 3)
python3-docxcompose - concatenate/append Microsoft Word (.docx) files
python3-docxtpl - Python DOCX template engine
python3-knipy - report generation tool with Python
python3-python-docx - transitional package
r-cran-pander - GNU R 'Pandoc' writer
tea - graphical text editor with syntax highlighting
ugrep - faster grep with an interactive query UI
usrht@usrht-VirtualBox:~$
```

Burada **python3-docx** incelemesi yapalım.

```
usrht@usrht-VirtualBox:~$ apt-cache show python3-docx
Package: python3-docx
Architecture: all
Version: 0.8.11+dfsg1-3
Priority: optional
Section: universe/python
Source: python-docx
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Debian Python Team <team+python@tracker.debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 604
Depends: python3-lxml, python3:any
Breaks: python3-python-docx (< 0.8.11+dfsg1-2~)
Replaces: python3-python-docx (< 0.8.11+dfsg1-2~)
Filename: pool/universe/p/python-docx/python3-docx_0.8.11+dfsg1-3_all.deb
Size: 130140
MD5sum: 40906617475441973049cd0ee506507a
SHA1: e0eb505e03b15e5ca50d07718b490a81816604
SHA256: 979a064919ef159adca6da1bbe1ce6655173a6012e1b6a0c2f50344271f7ddf9
SHA512: 99856b0f5abb8158be1e076b7ac4a28a3c86cdf0bd11711ad02e89b2ae215f31532519d8cdf188640f8d573e0310ae26a6590ac91287f6f15eeb7a80d25d2b5
Homepage: https://github.com/python-openxml/python-docx
Description-en: library for creating and updating Microsoft Word files (Python 3)
 python-docx is a Python library for creating and updating Microsoft Word (.docx) files.
.
This package installs the library for Python 3.
Description-md5: 6e906b57e01a8d5076fb9de4c3654bee
```

Bu bilgiler cache üzerinden gelmektedir. Bu cache dosyaları ise aşağıdaki dosya konumunda bulunur. Aşağıdaki konuma gittikten sonra “ls” ile listeleme yapabiliriz.

```
cd /var/lib/apt/lists/
```

Eğer bu cache bellekteki paketlerin güncellemesini yapmak istersek aşağıdaki komutu kullanabiliriz.

```
sudo apt-get update -y
```

Yukarıdaki güncellemeler ile biz sadece cache kısmındaki paketleri güncelledik, yüklenen paketlerde herhangi bir değişiklik yapılmadı. Eğer mevcutta yüklenen paketleri de güncellemek istersek aşağıdaki komutu kullanırız.

Burada eğer mevcuttaki paket ile cache içerisinde bulunan paket içerisinde versiyon farkı bulunursa bu işleme girecektir.

```
sudo apt-get upgrade -y
```

Yeni Paket Yükleme Ve Kaldırma

Yukarıda yaptığımız işlemlerden sonra paket yükleme işlemini görelim.

İlk olarak “tree” komutunu çalıştırmak istiyorum. Bu komut ile dosya ve klasörlerimizi dallandırılmış biçimde gösterebiliriz.

```
usrht@usrht-VirtualBox:~$ tree
Command 'tree' not found, but can be installed with:
sudo snap install tree # version 1.8.0+pkg-3fd6, or
sudo apt install tree # version 2.0.2-1
See 'snap info tree' for additional versions.
usrht@usrht-VirtualBox:~$
```

Yukarıda görüldüğü üzere “tree” paketi yüklü olmadığından dolayı komut çalışmıyor. Ancak alt kısmında bizden bu paketi nasıl yüklememizi istediği gösterilmiş. Önceki konumuz olan **apt-cache** komutu ile cache bellek içerisinde bu komut paketinin olup olmadığını kontrol edelim.

```
usrht@usrht-VirtualBox:~$ apt-cache show tree
Package: tree
Architecture: amd64
Version: 2.0.2-1
Priority: optional
Section: universe/utils
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Florian Ernst <florian@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 113
Depends: libc6 (>= 2.34)
Filename: pool/universe/t/tree/tree_2.0.2-1_amd64.deb
Size: 47942
MD5sum: 56c0a1b14219269a7e731e146da46ec7
SHA1: d67408db0df34d6688473ccba6edaf9431e72dc4
SHA256: 08dee87871da761246a8dd9223c094e1c9bf75baf4f24028558426229ada7ab5
SHA512: b445dc879b4fecc97af4f676e1e2447cc8a6d3fa7feb02c67072b73482e7c4b5d8d80529382cd3d6244f9d55a26eb32f5c3d57410b81ac59ee7947992c28d91
Homepage: http://mama.indstate.edu/users/ice/tree/
Description-en: displays an indented directory tree, in color
Tree is a recursive directory listing command that produces a depth indented
listing of files, which is colorized ala dircolors if the LS_COLORS environment
variable is set and output is to tty.
Description-md5: 9b53b68087a50d4cd859ac0117aecc08
Task: xubuntu-desktop, lubuntu-desktop, ubuntu-mate-core, ubuntu-mate-desktop, ubuntu-budgie-desktop, ubuntu-budgie-desktop-raspi
```


Yukarıda görüldüğü üzere “**tree**” paketi cache bellek içerisinde bulunmaktadır.

```
sudo apt-get install tree -y
```

```
usrht@usrht-VirtualBox:~/hedef$ tree
.
├── folder1
│   └── folder2
│       └── file1.txt
└──
2 directories, 1 file
```

Şimdi ise “**xeyes**” paketini cache bellekte arayalım.

```
usrht@usrht-VirtualBox:~$ sudo apt-cache search xeyes
x11-apps - X applications
xfce4-eyes-plugin - eyes that follow your mouse for the Xfce4 panel
```

Bu paketi **x11-apps** paketi içinde bulabilirim. Bu paketi yüklersek **xeyes** kullanılabilir hale gelecektir.

```
usrht@usrht-VirtualBox:~$ sudo apt-get install x11-apps -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
x11-apps is already the newest version (7.7+8build2).
x11-apps set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Paket görüldüğü üzere mevcutta bulunmaktadır. Bu paketi yükledik. Artık “**xeyes**” komutu ile çalıştırabiliriz.

Eğer paket silmek istersek;

```
sudo apt-get remove tree
```

Yukarıdaki komut ile silme işlemi gerçekleşti ancak geriye bazı konfigürasyon dosyaları kalmış olabilir. Tamamını silme işlemi gerçekleştirmek için aşağıdaki komutu kullanabiliriz.

```
sudo apt-get purge tree
```

Eğer bu paketin tamamını silip daha sonra bağlı olduğu tüm paketleri de silmek istersek;

```
sudo apt-get autoremove tree
```

Bu komutlarla sistemden bu paketi sildik. Ancak hala cache bellek içerisinde bu paket bulunmaktadır. Bu listeyi de güncellemek istersek bu komut ile kullanılmayan paket zip dosyalarını sileriz.

```
sudo apt-get clean
```