



TEMEL LİNX KOMUTLARINA GİRİŞ (3)

Find

Dosya ve dizin arama için kullanılır.

Kullanımı: *find* **dizin** **seçenekler**

Seçenekler:

- name isim**: Aranılacak dosyanın ismi.
- iname isim**: Aranılacak dosya ismi büyük/küçük harf duyarlıdır
- type tip**: Aranılan dosyanın tipini belirler. Tip aşağıdaki değerlerden biri olabilir.
 - f*: Normal Dosya
 - d*: Dizin
 - b*: Blok dosyası
 - c*: Karakter dosyası
 - l*: Sembolik bağlantışeklindedir.

❖ *Bu komutun ayrıca daha detaylı arama yapmak için daha başka alt parametreleri de vardır.*

❖ Bir dosya aramak için

```
root@bilmuh:~# find /var -name boot.log  
/var/log/boot.log
```

❖ Var dizini altında **'log'** geçen dosyaları bulmak için

```
root@bilmuh:~# find /var/ -name "*log"  
/var/cache/swcatalog  
/var/log  
/var/log/apt/term.log  
/var/log/apt/history.log  
/var/log/dpkg.log  
/var/log/boot.log  
/var/log/faillog  
/var/log/installer/syslog  
/var/log/installer/Xorg.0.log  
/var/log/lastlog  
/var/log/alternatives.log  
/var/log/fontconfig.log  
/var/log/cups/access_log  
/var/lib/gdm3/.local/share/gvfs-metadata/root-b3eed2dd.log  
/var/lib/systemd/catalog  
/var/lib/sgml-base/supercatalog  
/var/lib/swcatalog  
/var/lib/dpkg/triggers/update-sgmlcatalog  
/var/lib/xml-core/catalog
```

❖ Var dizini altında içerisinde **'cups'** geçen dizinleri bulmak için

```
root@bilmuh:~# find /var/ -type d -name "*cups"  
/var/spool/cups  
/var/cache/cups  
/var/log/cups
```

exec

- ❖ Bulunan dosyalar ile ilgili işlem yapmak için **find komutu ile birlikte kullanılır.**

```
root@bilmuh:~# find /root -name ders.txt -exec rm {} \;
```

- ❖ Burada {} ile **bulunan dosyalar**, bu parantez arasına **yerleştirilir** ve **exec'ten hemen sonra belirtilen program buna göre çalışır.**

```
root@bilmuh:~# ls
123      a      a2.txt  index.html  not.txt      wget-log  yeni_dosya
1.txt    a1.txt  a34.txt  LISTE.txt   program.sh   YEDEK
```

- ❖ Aşağıda verilen örnekte **a1.txt** ve **a2.txt** dosyalarının silinmesi gösterilmiştir.

```
root@bilmuh:~# find . -name 'a[0-9].txt' -exec rm {} \;
root@bilmuh:~# ls
123      a      index.html  not.txt      wget-log  yeni_dosya
1.txt    a34.txt  LISTE.txt   program.sh   YEDEK
```

shred

- ❖ Dosyaları güvenli bir şekilde silmek için kullanılır.
- ❖ Bu komutla silmek istediğimiz **hedef dosya rastgele bitler yazılarak kurtarılması önlenmeye çalışılır.**
- ❖ Parametresiz kullanımda hedef dosyaya 3 defa rastgele bitler yazılır.
- n parametresi ile bunun kaç kez yapılacağı belirtilebilir.
- u parametresi, hedef dosyanın işlem sonunda silinmesi anlamına gelir.
- v parametresi yapılan işlemi ayrıntılı çıktı şeklinde görmek için kullanılır.

ÖRNEK

```
root@bilmuh:~# shred -n 10 -v index.html -u
shred: index.html: Geçiş 1/10 (random)...
shred: index.html: Geçiş 2/10 (000000)...
shred: index.html: Geçiş 3/10 (555555)...
shred: index.html: Geçiş 4/10 (924924)...
shred: index.html: Geçiş 5/10 (ffffff)...
shred: index.html: Geçiş 6/10 (random)...
shred: index.html: Geçiş 7/10 (aaaaaa)...
shred: index.html: Geçiş 8/10 (492492)...
shred: index.html: Geçiş 9/10 (db6db6)...
shred: index.html: Geçiş 10/10 (random)...
shred: index.html: Kaldırılıyor
shred: index.html: 0000000000 olarak yeniden adlandırıldı
shred: 0000000000: 0000000000 olarak yeniden adlandırıldı
shred: 0000000000: 0000000000 olarak yeniden adlandırıldı
shred: 000000000: 0000000000 olarak yeniden adlandırıldı
shred: 00000000: 00000000 olarak yeniden adlandırıldı
shred: 0000000: 00000000 olarak yeniden adlandırıldı
shred: 000000: 000000 olarak yeniden adlandırıldı
shred: 00000: 000000 olarak yeniden adlandırıldı
shred: 0000: 0000 olarak yeniden adlandırıldı
shred: 000: 00 olarak yeniden adlandırıldı
shred: 00: 0 olarak yeniden adlandırıldı
shred: index.html: Kaldırıldı
```

Dosya İsmi Örüntüleri

❖ *Linux altında dosya isimleri açıkça yazılabildiği gibi **belirli kurala uyan dosyaların tamamını ifade etmek üzere örüntüler de kullanılabilir.***

? Herhangi bir tek karakteri gösterir.

hd?, hda, hdb, hdc, hdd gibi hd ile başlayıp devamın tek karakter olan

** bütün dosya isimleriyle eşleşir.*

** Hiç veya herhangi sayıdaki karakteri gösterir.*

**1.jpg, dosyaadı 1 olan veya 1 ile biten tüm jpg uzantılı dosyalarla eşleşir.*

[] Aralık gösterir.

a[1-9], a1, a2...a9 isimleriyle eşleşir.

a[6,8], a6 ve a8 isimleriyle eşleşir.

{ } Virgülle ayrılmış seçenekleri gösterir.

{.doc,*.pdf} .doc ve .pdf dosyaları gösterir.*

\ Özel karakterleri korumak için kullanılır.

❖ Tek bir karakteri için listeleme

```
root@bilmuh:~# ls
a not.txt x.txt y.txt
root@bilmuh:~# ls ?.txt
x.txt y.txt
```

❖ Birden fazla karakteri için listeleme

```
root@bilmuh:~# ls *.txt
not.txt x.txt y.txt
```

❖ Rakam ile başlamayan dosyaları silme

```
root@bilmuh:~# ls
123 a LISTE.txt not.txt x.txt YEDEK y.txt

root@bilmuh:~# rm [a-z]*.txt
root@bilmuh:~# ls
123 a LISTE.txt YEDEK
```

❖ tmp dizini altında a ile başlayıp 2. karakteri bilinmeyen ve rakam ile devam eden metin belgelerinin listelenmesi

```
root@bilmuh:~# ls /tmp
123.txt
a1.txt
a2.txt
a34.txt
abc123.txt
systemd-private-818b299893fd462d92ab21603939ece6-
```

```
root@bilmuh:~# find /tmp -name "a?[0-9].txt"
/tmp/a34.txt
```

echo

- ❖ Normalde bu komut, istenilen ifadeleri terminal ekranına çıktı olarak gönderir.

```
root@bilmuh:~# echo "Merhaba"  
Merhaba
```

```
root@bilmuh:~# echo merhaba  
merhaba
```

- ❖ Özel karakterler (', " vb.) ekrana yazılmak istenirse \ karakteri kullanılmalıdır.

```
root@bilmuh:~# echo linux'ta dizin yapısı  
>
```

```
root@bilmuh:~# echo linux\'ta dizin yapısı  
linux'ta dizin yapısı
```

HATA!!!

Çevre Değişkenleri

- ❖ Çevre değişkenleri, **programlama dillerindeki değişkenler (variable)** gibi **belirli bilgileri taşıyan özel stringler'dir.**
- ❖ **Uygulamalar**, bu çevre değişkenlerinin **değerlerine göre davranışlarını değiştirebilir.**
- ❖ **Çevre değişkenlerine**, değişken adının **başına \$ eklenerek ulaşılır.**
- ❖ **env** komutu ile o anda tanımlı tüm çevre değişkenleri listelenir.

```
root@bilmuh:~# env
SHELL=/bin/bash
PWD=/root
LOGNAME=root
HOME=/root
LANG=tr_TR.UTF-8
TERM=xterm-256color
USER=root
SHLVL=1
XDG_DATA_DIRS=/usr/share/gnome:/usr/local/share/:/usr/share/
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
MAIL=/var/mail/root
_=/usr/bin/env
```

❖Aşağıda görüldüğü gibi çevre değişkenlerinin değerleri ekrana yazdırılmıştır.

```
root@bilmuh:~# echo $SHELL
/bin/bash
```

```
root@bilmuh:~# echo $PWD
/root
```

❖ Bir değişken, sadece içinde bulunulan kabuk içinde kullanılacaksa şu şekilde tanımlanır:

```
root@bilmuh:~# x=15
root@bilmuh:~# echo $x
15
```

❖ Tanımlı olan bir çevre değişkenini silmek için ise **unset** komutu kullanılır.

❖ Bu komut parametre olarak değişken adını **başında \$ işareti olmadan alır.**

```
root@bilmuh:~# unset x
```

```
root@bilmuh:~# echo $x
```

x değişkeninin içi boşaltıldığı için ekrana bir şey yazmamıştır.

❖ **PATH:** Verilen komutların **hangi dizinlerde aranacağını belirler.**

❖ Bu değişkende kolon ayracı ':' ile ayrılmış birden fazla dizin yolu tanımlanabilir.

```
root@bilmuh:~# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

❖ **LANG:** Sisteminde tanımlı dil ayarlarını gösterir.

```
root@bilmuh:~# echo $LANG  
tr_TR.UTF-8
```

EDITOR: Aksi belirtilmedikçe dosya işlemlerinde kullanılacak ön tanımlı dosya düzenleyiciyi belirler.

Örneğin **crontab -e** ile zamanlanmış **görevler açılırken bu değişkene bakılır.**

PS1: Kullanıcının *birincil komut satırını şekillendirir.*

PS2: Kullanıcının *ikincil komut satırını şekillendirir.*

❖ **Kabuk programın kullanıcıya sunduğu iki komut satırı vardır:**

❖ **Esas komut satırı ve yardımcı komut satırıdır.**

❖ Bunlar **PS1** ve **PS2** değişkenlerinde tutulur.

```
root@bilmuh:~# echo $PS1  
${debian_chroot:+($debian_chroot)}\u@\h:\w\$_
```

❖ Burada **\u** kullanıcı (user),

❖ **\h** host,

❖ **\W** ise çalışma dizini anlamındadır.

❖ Böyle bir prompt'ta user1 kullanıcısında \$ ile biterken root kullanıcısında # ile biter.

```
root@bilmuh:~# echo $PS2
```

```
>
```

❖ **Yardımcı prompt (>), tamamlanmamış komut satırları için çıkar.**

❖ Komutun devam ettiğini ve hala girdi yapılması gerektiğini bildirir.

Program Çalıştırma

- ❖ Linux altında bir **programı**, **komutu** veya **betiği** çalıştırabilmek için programa ait dosyanın **çalıştırma yetkisinin olması gerekmektedir**.
- ❖ Yetkiler '**ls -l**' komutuyla görülebilir.

```
root@bilmuh:~# ls -l /bin/cat  
-rwxr-xr-x 1 root root 44016 Eyl 20 2022 /bin/cat
```

- ❖ Bir komut eğer çalışma yolunda(PATH) ise herhangi bir dizinden çalıştırılabilir.
- ❖ PATH kabukta komut tam yol verilmeden yazıldığında **komutun sıra ile hangi dizinlerde aranacağını belirleyen özel bir çevre değişkenidir**.
- ❖ **Sistem komutları** çalışma PATH'ine yerleştirildiği için **tam yolunu vermeden çalıştırılabilir**.
- ❖ Ancak kendi programlarımız veya betikler genelde standart **sistem PATH'inde olmadığı için tam yolu verilerek çalıştırılmalıdır**.

Örnek

❖ Ekranı 10 defa Merhaba yazan betik.

```
GNU nano 7.2 program.sh
#!/bin/bash
i=1
while [ $i -lt 10 ]
do
i=$((i+1))
echo "Merhaba"
done
```

```
root@bilmuh:~# ls -l
toplam 12
drwxr-xr-x 2 root root 4096 Kas 11 22:42 123
-rw-r--r-- 1 root root 0 Kas 11 22:50 1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:38 a
-rw-r--r-- 1 root root 0 Kas 11 22:50 a1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a2.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a34.txt
-rw-r--r-- 1 root root 0 Kas 11 22:44 LISTE.txt
-rw-r--r-- 1 root root 71 Kas 12 01:35 program.sh
drwxr-xr-x 2 root root 4096 Kas 11 22:42 YEDEK
```

❑ Bu betik kullanıcının bulunduğu dizin ile

- Aynı dizindeyse ./program.sh
- Eğer bir üst dizindeyse ../program.sh
- Eğer başka bir dizindeyse tam yol yazılır /root/program.sh

yazarak çalıştırılır.

❖ Bu betiğin **sağda görüldüğü gibi çalıştırma yetkisi yoktur!**

❖ **chmod +x program.sh** diyerek çalıştırma yetkisi verilir.

```
root@bilmuh:~# ./program.sh
-bash: ./program.sh: Erişim engellendi
```

Örnek (Devam)

```
root@bilmuh:~# chmod +x program.sh
```

❖ Yukarıda belirtildiği gibi çalışma yetkisi verildikten sonra betik aşağıdaki gibi çalıştırılır.

```
root@bilmuh:~# ./program.sh
```

Merhaba

Merhaba

Merhaba

Merhaba

Merhaba

Merhaba

Merhaba

Merhaba

Merhaba

Çıkışı Yönlendirme

- ❑ Bir komutun çıktıları bir dosyaya yönlendirilebilir.
- ❑ Bu sayede komut çıktıları, kalıcı olarak diske yazılıp daha sonra üzerlerinde çalışılabilir.
- ❑ Linux'de 3 adet ön tanımlı dosya tanımlayıcısı (file descriptor) vardır:
 - Standart Girdi (stdin)
 - Standart Çıktı (stdout)
 - Standart Hata (stderr)

Standart Girdi (stdin) :

- ❖ Çalışan programın dosya vb kaynakları açmadan **veri okumak için kullanacağı kaynağı belirtir.**
- ❖ Ön tanımlı olarak **stdin** klavyeden veri alır.
- ❖ **Stdin** dosya numarası **0**'dır.
- ❖ **Stdin** yönlendirmesi **<** karakteri ile yapılır.

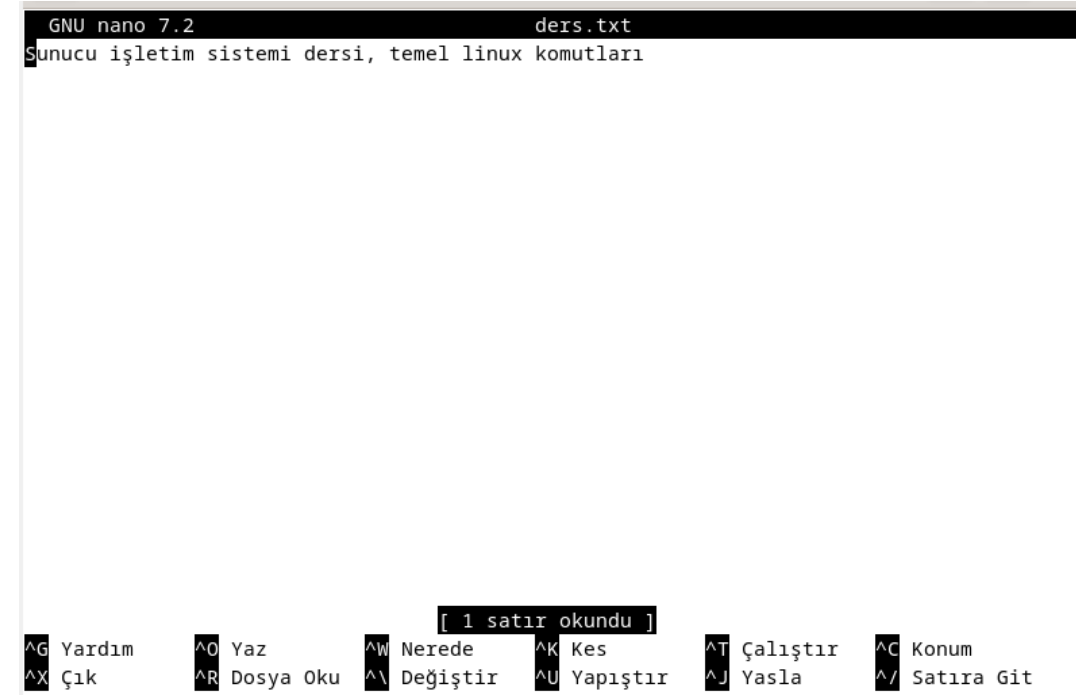
```
root@bilmuh:~# sort < program.sh
#!/bin/bash
do
done
echo "Merhaba"
i=$((i+1))
i=1
while [ $i -lt 10 ]
```

```
root@bilmuh:~# sort 0< program.sh
#!/bin/bash
do
done
echo "Merhaba"
i=$((i+1))
i=1
while [ $i -lt 10 ]
```

Standart Çıktı (stdout) :

- ❖ Çalışan programın **çıktılarını göndereceği hedefi belirtir.**
- ❖ Ön tanımlı olarak çıktı terminale basılır.
- ❖ Stdout dosya tanımlama numarası **1**'dir.
- ❖ Ön tanımlı olarak **stdin klavyeden veri alır.**
- ❖ Stdout yönlendirilmesi > veya >> karakteri ile yapılır.
- ❖ Örnek:

```
root@bilmuh:~# echo "Sunucu işletim sistemi dersi, temel linux komutları" >ders.txt
```



```
GNU nano 7.2                                ders.txt
Sunucu işletim sistemi dersi, temel linux komutları

[ 1 satır okundu ]
^G Yardım      ^O Yaz          ^W Nerede       ^K Kes          ^T Çalıştır    ^C Konum
^X Çık          ^R Dosya Oku   ^\ Değiştir    ^U Yapıştır    ^J Yasla      ^_ Satıra Git
```

❖> karakteri **mevcut dosya içeriğini silip** yeniden yazılması için kullanılır.

❖ >> karakterleri ise **mevcut dosyanın sonuna ekleme** yapar.

Örnek : **echo “yeni içerik” >> ders.txt**

```
GNU nano 7.2      ders.txt
sunucu işletim sistemi dersi, temel linux komutları
yeni içerik
```

Örnek : **echo “yeni içerik” > ders.txt**

```
GNU nano 7.2      ders.txt
yeni içerik
```

Standart Hata (stderr) :

- ❖ Çalışan programın **hata çıktılarını göndereceği hedefi belirtir.**
- ❖ Ön tanımlı olarak hata terminale basılır.
- ❖ Hata dosya tanımlama numarası **2**'dir.
- ❖ **Stderr** yönlendirmesi için **stdout**'dan farklı olarak 2 nolu standart hata numarası belirtilerek **2>** şeklinde yapılmalıdır.

Örnek: **cat dosya.txt 2> /tmp/err.log**

- ❖ **Hem STDOUT hem de STDERR'yi aynı dosyaya yönlendirmek için:**

Örnek: **cat dosya.txt > /tmp/err.log 2>&1**

Örnekler:

❖ **> /dev/null 2>&1**

Stdout'u **/dev/null**'a yönlendirir.

Stderr'yi de **stdout**'a yönlendirir.

Sonuç olarak Stdout /dev/null'a yönlendirildiği için çıktılar, **/dev/null'a yönlendirilmiştir** ve ekranda bir şey gözükmez.

/dev/null, linux'ta bir kara delik olarak görülebilir.

❖ **2>&1 > /dev/null**

Stderr, **stdout**'a yönlendirilir.

Sonra sadece **stdout** **/dev/null**'a yönlendirilir.

❖ **1>&2 > /dev/null**

stdout, **stderr**'ye yönlendirilir.

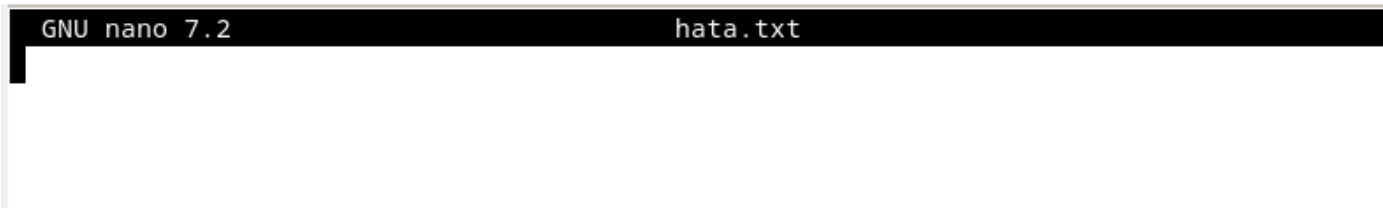
Sonra **stdout** **/dev/null**'a yönlendirilir. **Stderr'a dokunulmaz.**

❖ **dmesg** komutu ile çekirdeğin işletim sistemi açılığında yürüttüğü işlemler ekrana bastırılır.

❖ Çekirdek tarafından tespit edilen hataları ***hata.txt*** adında bir dosya yazdırılmasını sağlayınız?

```
root@bilmuh:~# dmesg 2>hata.txt
[    0.000000] Linux version 6.1.0-13-amd64 (debian-kernel@lists.debian.org) (gcc-12
ebian 12.2.0-14) 12.2.0, GNU ld (GNU Binutils for Debian) 2.40) #1 SMP PREEMPT_DYNAM
Debian 6.1.55-1 (2023-09-29)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-6.1.0-13-amd64 root=UUID=fbfc0
-eead-4890-ad9b-24ed7576c745 ro quiet
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x0000000000009fc00-0x0000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x000000000000f0000-0x000000000000ffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000100000-0x00000000000dfffff] usable
```

❖ Aşağıda görüldüğü hiçbir hata ile karşılaşmadığı için hata.txt dosyası boştur.



❖ Peki **dmesg 2>hata.txt** komutu çalıştırıldığında **ekranda bir şey gözükmemesi** nasıl sağlanır?

❖ **Peki dmesg 2>hata.txt komutu çalıştırıldığında ekranda bir şey gözükmemesi nasıl sağlanır?**

```
root@bilmuh:~# dmesg 2>hata.txt 1>/dev/null
```

❖ Burada hatalar (**stderr**), **hata.txt** dosyasına **yönlendirilmiştir**.

❖ dmesg komutunun çıktısı (**stdout**), **/dev/null'a yönlendirilmiştir**.

❖ **Sonuç olarak ekranda bir şey yazmamıştır.**

Ardışık Komutlar – Pipe (|)

- ❖ Bash altında bir komutun çıktısı, bir dosyaya yönlendirilebildiği gibi **bir başka komuta gönderilebilir.**
- ❖ Pipe denen bu işlem şu biçimdedir:
- ❖ `komut1 | komut2 | komut3 ...`
- ❖ Burada **komut2** ve **komut3** 'ün girdileri **standart girişten alan programlar olması gerekir.**
- ❖ Örneğin bir sunucuda kaç kullanıcının tanımlı olduğunu bulduran komut aşağıda verilmiştir.

```
root@bilmuh:~# ls /home | wc -l  
2
```

tee

- ❖ Komut çıktısını **hem dosyaya** hem de **Stdout'a** yazar.
- ❖ **-a**: Dosyanın **sonuna ekleme yapar**, **mevcut dosyayı sıfırlamaz**.

```
root@bilmuh:~# ls -l | tee not.txt
toplamlam 220
drwxr-xr-x 2 root root 4096 Kas 11 22:42 123
-rw-r--r-- 1 root root 0 Kas 11 22:50 1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:38 a
-rw-r--r-- 1 root root 0 Kas 11 22:50 a1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a2.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a34.txt
-rw-r--r-- 1 root root 109 Kas 12 23:06 ders.txt
-rw-r--r-- 1 root root 201565 Kas 12 22:53 index.html
-rw-r--r-- 1 root root 0 Kas 11 22:44 LISTE.txt
-rw-r--r-- 1 root root 0 Kas 12 23:07 not.txt
-rwxr-xr-x 1 root root 71 Kas 12 01:35 program.sh
-rw-r--r-- 1 root root 937 Kas 12 22:53 wget-log
drwxr-xr-x 2 root root 4096 Kas 11 22:42 YEDEK
-rw-r--r-- 1 root root 0 Kas 12 22:39 yeni_dosya
```

```
GNU nano 7.2 not.txt
toplamlam 220
drwxr-xr-x 2 root root 4096 Kas 11 22:42 123
-rw-r--r-- 1 root root 0 Kas 11 22:50 1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:38 a
-rw-r--r-- 1 root root 0 Kas 11 22:50 a1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a2.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a34.txt
-rw-r--r-- 1 root root 109 Kas 12 23:06 ders.txt
-rw-r--r-- 1 root root 201565 Kas 12 22:53 index.html
-rw-r--r-- 1 root root 0 Kas 11 22:44 LISTE.txt
-rw-r--r-- 1 root root 0 Kas 12 23:07 not.txt
-rwxr-xr-x 1 root root 71 Kas 12 01:35 program.sh
-rw-r--r-- 1 root root 937 Kas 12 22:53 wget-log
drwxr-xr-x 2 root root 4096 Kas 11 22:42 YEDEK
-rw-r--r-- 1 root root 0 Kas 12 22:39 yeni_dosya
```

- ❖ Komut satırında gözüken

- ❖ Çıktı not.txt dosyasına aktarılmıştır.

❖Çıktı birden fazla dosyaya da yazılabilir.

Örnek: `ls | tee file1.txt file2.txt`

❖Çıktı **hem dosyaya yazılır** hem de **bir komuta (sort) girdi** olarak **verilebilir.**

```
root@bilmuh:~# ls -l | tee not.txt | sort
drwxr-xr-x 2 root root 4096 Kas 11 22:42 123
drwxr-xr-x 2 root root 4096 Kas 11 22:42 YEDEK
-rw-r--r-- 1 root root 0 Kas 11 22:38 a
-rw-r--r-- 1 root root 0 Kas 11 22:44 LISTE.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a1.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a2.txt
-rw-r--r-- 1 root root 0 Kas 11 22:50 a34.txt
-rw-r--r-- 1 root root 0 Kas 12 22:39 yeni_dosya
-rw-r--r-- 1 root root 0 Kas 12 23:11 not.txt
-rw-r--r-- 1 root root 109 Kas 12 23:06 ders.txt
-rw-r--r-- 1 root root 201565 Kas 12 22:53 index.html
-rw-r--r-- 1 root root 937 Kas 12 22:53 wget-log
-rwxr-xr-x 1 root root 71 Kas 12 01:35 program.sh
toplam 220
```

Grep

❖ Bir dosya içerisinde ya da bir komut çıktısı içerisinde **bir ifade(string) aramak için kullanılır.**

Kullanımı: **grep** [parametre...] [kelime veya örüntü ...] [dosya...]

Bu komutun aldığı başlıca parametreler:

- v: komutun **davranışını tersine çevirir.** Bir başka ifadeyle aranılan kelimeyi içermeyen satırları listeler.
- i: arama sırasında **büyük küçük harf ayrımı** yapmaz.
- r: verilen dizinin **alt dizini içerisinde arama** yapar.
- n: aranan kelimenin geçtiği **satır numarasını** verir.
- c: belirtilen dizinde aranılan **kelimenin kaç kez geçtiğini** gösterir.
- l: şablona/pattern'e uygun satırların **bulunduğu dosya adlarını** listeler.

Örnek

- ❖ Passwd dosyası kullanıcı bilgilerinin tutulduğu dosyadır.
- ❖ Sistemde bir kullanıcı adının tanımlanıp tanımlanmadığı **grep komutu ile aşağıdaki gibi öğrenilebilir.**

```
root@bilmuh:~# grep seckin /etc/passwd  
seckin:x:1000:1001:seckin,,,:/home/seckin:/bin/bash
```

veya

```
root@bilmuh:~# cat /etc/passwd | grep seckin  
seckin:x:1000:1001:seckin,,,:/home/seckin:/bin/bash
```

xargs

- ❖ Bir komutu parametrelerini **standart girdiden(stdin)** alıp **başka komutlara geçebilecek** şekilde çalıştırır.
- ❖ Böylece bir defada işleyebileceği parametreden daha fazlası komuta aktarılabilir.
- ❖ Genelde **ls** veya **find** komutu tarafından **üretilmiş uzun dosya listesini**, **bu dosyaların her birinde tek tek işlem yapacak** bir başka komuta parametre olarak geçmek için kullanılır.
- ❖ Örneğin bir dizinde bulunan tüm txt dosyalarının silinmesini sağlayan komut aşağıda verilmiştir.

```
root@bilmuh:~# ls
123  a      index.html  not.txt      wget-log  yeni_dosya
1.txt a34.txt LISTE.txt    program.sh  YEDEK
root@bilmuh:~# ls *.txt | xargs rm
root@bilmuh:~# ls
123  a  index.html  program.sh  wget-log  YEDEK  yeni_dosya
```

Linux'ta birden fazla komut çalıştırılması

- ❖ Linux'ta birden fazla komut tek bir satırda çalıştırılabilir.
- ❖ Bunun 3 farklı yolu vardır.

ÖZEL KARAKTER	KULLANIMI	AÇIKLAMA
&&	Komut 1 && Komut 2	Komut 2'yi yalnızca komut 1 başarıyla biterse çalıştırır.
	Komut 1 Komut 2	Komut 2'yi yalnızca komut 1 başarısız olursa çalıştırır.
;	Komut 1 ; Komut 2	Önce komut 1 'i ve ardından komut 2'yi çalıştırır.

ÖRNEKLER

- ❖ **Komut 1 && Komut 2** şeklinde komut yazılırsa bunu **bağlı komutlar şeklinde değerlendirmek gerekir.**
- ❖ Örneğin “make” komutu başarılı olursa “make install” komutu çalıştırılması mantıklı olacaktır.
make && make install şeklinde komutu çalıştırmak gerekir.

ÖRNEKLER

```
root@bilmuh:~# asadsad && echo "Merhaba"  
-bash: asadsad: komut yok
```

- ❖ Burada ilk kısımda hata olunca (linux'ta böyle bir komut yok) ve ikinci kısımdaki komut çıktı vermemiştir.

```
root@bilmuh:~# ls || echo "Merhaba"  
123  1.txt  a  _a1.txt  a2.txt  a34.txt  ders.txt  LISTE.txt  program.sh  YEDEK
```

- ❖ Burada ilk kısım hata vermeden çalıştı ve ekrana *Merhaba* yazmadı.

ÖRNEKLER

```
root@bilmuh:~# touch yeni_dosya; ls
123      a          a2.txt    ders.txt  program.sh yeni_dosya
1.txt    a1.txt    a34.txt  LISTE.txt YEDEK
```

❖ Burada yeni bir dosya oluşturup sonrasında tüm dizinin içerisi listelenmiştir.

```
root@bilmuh:~# pwd; touch yeni_dosya; ls
/root
123      a          a2.txt    ders.txt  program.sh yeni_dosya
1.txt    a1.txt    a34.txt  LISTE.txt YEDEK
```

❖ Yukarıda görüldüğü gibi 3 tane farklı komut verilmiştir.

❖ Her biri sırasıyla çalıştırılmıştır.

❖ Linux'ta bir komutu veya işlemi arkaplanda yaptırmak

❖ Bu tarz işlemleri gerçekleştirmek için & (ampersand) karakteri kullanılır.

❖ Kullanımı: **[komut] &**

❖ Örneğin *sleep saniye* komutu sistemi belirtilen sürede uyku moduna (beklemeye) geçirir. Bu süre zarfında işlem yapılmaz.

```
root@bilmuh:~# sleep 10 ; echo "Merhaba"
```

❖ Yukarıda görüldüğü gibi **10 saniye dolmadığı için ekrana Merhaba yazmamıştır.**

```
root@bilmuh:~# sleep 220 &
```

```
[1] 2992
```

```
root@bilmuh:~# jobs -l
```

```
[1]+ 2992 çalışıyor
```

```
sleep 220 &
```

❖ Yukarıda verildiği gibi **bu komut arkaplanda yürütülmektedir.**

Arşiv ve Yedekleme

❑ **Linux'ta arşivleme ve yedekleme için çeşitli komutlar vardır.**

❑ **Bunlar:**

- tar
- gzip
- gunzip
- zcat,
- bzip2
- dd
- Cpio

şeklinde özetlenebilir.

tar

- ❖ Linux'ta tar (Tape ARchive) ile arşivleme yapmayı destekler.
- ❖ Tar aslında teyp birimlerine arşivleme amacıyla geliştirilmiştir.
- ❖ Teyp üniteleri gelişen yeni teknoloji ile birlikte kullanımdan yavaş yavaş kalkmaktadır.
- ❖ Fakat tar programı halen daha değişik amaçlarla dosya sisteminde kullanılabilmektedir.
- ❖ **Tar, kendisi veri sıkıştırması yapmaz.**
- ❖ Veri sıkıştırması için “**gzip**” veya “**bzip2**” biçimlerini kullanabilmektedir.

Örnek 1:

```
❖ tar -zcvf yedek.tar.gz db.sql mbox  
db.sql  
Mbox
```

Burada **z** parametresi **gzip** sıkıştırması **algoritmasının kullanılacağını**

- '**v**' ekrana ne yapıldığına dair **bilgi yazdırılacağını**
- '**c**' parametresi **arşiv oluşturma**,
- '**x**' parametresi **arşiv açmayı**
- '**f**' parametresi ise arşivleme sonucu oluşacak veya mevcut **arşiv dosyanın adını**

belirtmek kullanılmıştır.

Bzip2 biçiminde sıkıştırma yapmak için **z** yerine **j** kullanmak gerekir.

```
❖ ls -lh yedek.tar.gz
```

```
-rw-r--r-- 1 ahmet ahmet 176 Ara 13 23:22 yedek.tar.gz
```

cpio

- ❖ Dosyaları, **arşive kopyalamak, arşivdeki dosyaları çıkartmak** için kullanılır.
- ❖ Arşivlenecek dosyaların listesini **STDIN**'den alır ve arşivi **STDOUT**'a yazar.
- ❖ Bu nedenle dosya listesinin oluşturulması ve çıkışın yönlendirilmesi gerekmektedir.

Örnek: ls | cpio -ov > directory.cpio

Burada **o** seçeneği **arşiv oluşturmak**,

-v seçeneği ise **yapılan işlemler hakkında bilgi vermek** için kullanılır.

Örnek:

- ❖ Bir dizini alt klasörleriyle tamamen yedeklemek için dosya isimleri **find** komutuyla oluşturulabilir.

```
find . -print -depth | cpio -ov > tree.cpio
```

- ❖ Arşivi açmak için:

```
cpio -idv < tree.cpio
```

Burada **-l** seçeneği **arşivi açmak**,

-d seçeneği ise **arşivin içerdiği klasörleri oluşturmak** için kullanılır.

Normalde cpio klasörleri oluşturmaz.

Örnek:

- ❖ Cpio'nun bir kullanım şekli de **bir dizini diğer bir dizine kopyalamasıdır.**

find . -depth -print0 | cpio --null -pvd new-dir

- ❖ cpio komutu kullanılarak **tar arşiv oluşturulabilir.**

ls | cpio -ov -H tar -F arşiv.tar

Burada **-F** seçeneği ile **çıktının yazılacağı arşiv dosyasının ismi,**

-H ile arşiv biçimi belirtilir.

- ❖ **Tar arşivi açmak için:**

cpio -idv -F ornek.tar

- ❖ **Tar arşivin içindeki dosyaları görüntülemek için:**

cpio -it -F ornek.tar

dd

❖ Öntanımlı olarak **Stdin**'den **Stdout**'a belirtilen **blok boyutlarında dosya kopyalar**.

- Genel kullanımı aşağıdaki gibidir:

dd if=girdi of=cikti bs=blok-boyutu count=blok-sayısı

if: Input File, girdi

of: Output File, çıktı

count: bs ile boyutları belirtilen bloklardan kaç adet kopyalanacağı

bs: Bir blogun boyutu (ön tanımlı: 512 byte)

dd

- ❖ dd komutu genel olarak **disk imajı almak için yedekleme amaçlarıyla kullanılır.**
- ❖ Aşağıdaki komutla *sda* diski *sdb* diskine **dump** edilmektedir.
`dd if=/dev/sda of=/dev/sdb`
- ❖ *Bir diskin imajını alıp dosya olarak saklamak için*
`dd if=/dev/hda of=haddisk.img`
- ❖ *Alınmış imajı herhangi bir diske (örnekte hdb) dump etmek için:*
`dd if=hdadisk.img of=/dev/hdb`

dd

- ❖ Bir disk tamamen dump edilebildiği gibi içindeki bir disk bölümü yalnız başına dump edilebilir.

dd if=/dev/hda1 of=hda1.img

- ❖ Bir CD'den ISO dosyası oluşturulabilir:

dd if=/dev/cdrom of=debian.iso bs=2048

- ❖ MBR'nin yedeğini almak için:

dd if=/dev/hdx of=/path/to/image count=1 bs=512

Not: MBR ilk 512 byte olduğundan 512 byte'lık tek bir blok kopyalanmalıdır.

gzip

- ❖ Verilen dosyayı **Lempel-Ziv (LZ77)** kodlamasını kullanarak **sıkıştırır** ve **dosya boyutunu küçültür**.
- ❖ Sıkıştırılan **dosyanın uzantısı otomatik olarak .gz** yapılır.
- ❖ Ön tanımlı olarak **orijinal dosya ismini** ve **dosya değiştirilme zamanını** **muhafaza eder**.
- ❖ **Örnek:**

gzip test.c , gzip -r directory

Yukarıda verilen örneklerde **dosyanın veya klasörün eski hali diskten silinip yerine sıkıştırılmış hali yazılır.**

gzip -c archive.txt > arshiv.txt.gz

- ❖ **-c parametresi sayesinde çıktı STDOUT'a yazılır ve orjinal dosya muhafaza edilir.**
- ❖ **Arşiv dosyasının saklanması için STDOUT'un bir dosyaya yönlendirilmesi gerekmektedir.**

gunzip

❖ **gunzip** komutu, **gzip**, **zip**, **compress** ve **pack** sıkıştırma biçimlerini otomatik olarak tanıyarak açar.

gunzip test.c.gz

❖ **gunzip -c** parametresiyle kullanılırsa çıktıyı STDOUT'a yazar.

❖ **zcat** komutu, **gunzip -c** ile aynı işlevi görür.

bzip2

- ❖ **bzip2** daha iyi sıkıştırma yapabilen *Burrows-Wheeler* metin sıkıştırma algoritmasını kullanan bir sıkıştırma komutudur.
- ❖ **Kullanımı gzip ile aynıdır.**
- ❖ Sadece dosya uzantısı **bz2** şeklinde oluşturulur.
- ❖ **Örnek: `bzip2 -c readme.txt > readme.txt.bz2`**

KAYNAKLAR

- ❖ Pardus LPI-Sertifikasyon-Kitabı
- ❖ Linux Komut Satırı, Kemal DEMİREZ, Abaküs Kitap Yayın Dağıtım Hizmetleri, 2019

UYGULAMA SORULARI

Aşağıdaki soruları okuldaki sunucuya bağlanarak yapınız.

1. Sistemde kullanıcı adınızda (öğrenci numaranız) bir klasörü aratıp yolunu ekrana yazdırınız.
2. Bir dosya oluşturup içerisine **/home** dizini altındaki klasör adlarını yazdırınız.
3. Bu dosyayı gzip algoritması kullanarak sıkıştırıp arşivleyiniz ve yedek adı altında kendi bulunduğunuz dizinde saklayınız.
4. Kendi bulunduğunuz dizinde a1.txt, a2.txt ...a10.txt veya rast gele isimlerden oluşacak 10 tane dosyayı otomatik oluşturacak betiği (script) yazıp çalıştırınız.
5. Kendi bulunduğunuz dizinde a1.txt, ...a5.txt dosyalarını (5 tane dosyayı) silecek komutu yazınız.

UYGULAMA SORULARI

6. Kullandığınız kabuğu (Shell) öğreniniz.
 7. `metin_belgeler` adında bir klasör oluşturunuz. Bu klasöre `a6.txt,...a10.txt` dosyalarını komut yazarak taşıyınız.
 8. `Cpio` komutu kullanarak `metin_belgeler` klasörünü yedekleyiniz. Yedeğiniz sunucudan indirip Windows işletim sisteminde açmaya çalışınız.
 9. `metin_belgeler` klasörünü sadece arşivleyiniz.
 10. Bu arşivi kalıcı olarak siliniz. (5 kez tekrarlama yapılacak şekilde)
 10. Terminal ekranına `sleep 120` yazdıktan sonra `/tmp` dizini altındaki dosyaları listeleyiniz.
- Daha sonra `sleep 120` komutunu arkaplanda çalışacak şekilde tekrar uygulayınız.
- Son olarak `jobs -l` komutu ile yürütülen süreçlere bakınız.

UYGULAMA SORULARI

Aşağıdaki sorular kendi bilgisayarınız üzerinde deneyiniz.

1. Kurduğunuz sunucuda sistemsel bir hata olup olmadığını bakınız.
dmesg komutundan faydalanarak hataları başka bir dosyaya aktarıp okuyunuz.
2. dd komutundan faydalanarak sunucuyu kurduğunuz diskin imajını alınız.
3. Programcı, bir web sayfası hazırlayıp sunucuya yüklediğinde bu web sayfasındaki dosyalara (html, php, resim vb.) ait haklar ne olmalıdır?
4. Bir web sayfasında dosya upload etme kısmı vardır.
Bu sayfada eğer bir kullanıcı, normal dosyalar yerine (metin belgesi, resim vb.) yerine bir betik yüklemek isterse bunu nasıl engelleriz?
Burada sistemciye ve programcıya düşen görevler nelerdir?
Örneğin dosyaların upload edildiği dizinde kullanıcı, grup ve diğerleri için haklar ne olmalıdır?
Bunu programcı da denetleyebilir mi?