KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Linux Ağ Yönetimi

6. Hafta – Birleşik Komutlar

Yrd. Doç. Dr. A. Burak İNNER

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr

LİNUX KOMUT FELSEFESİ

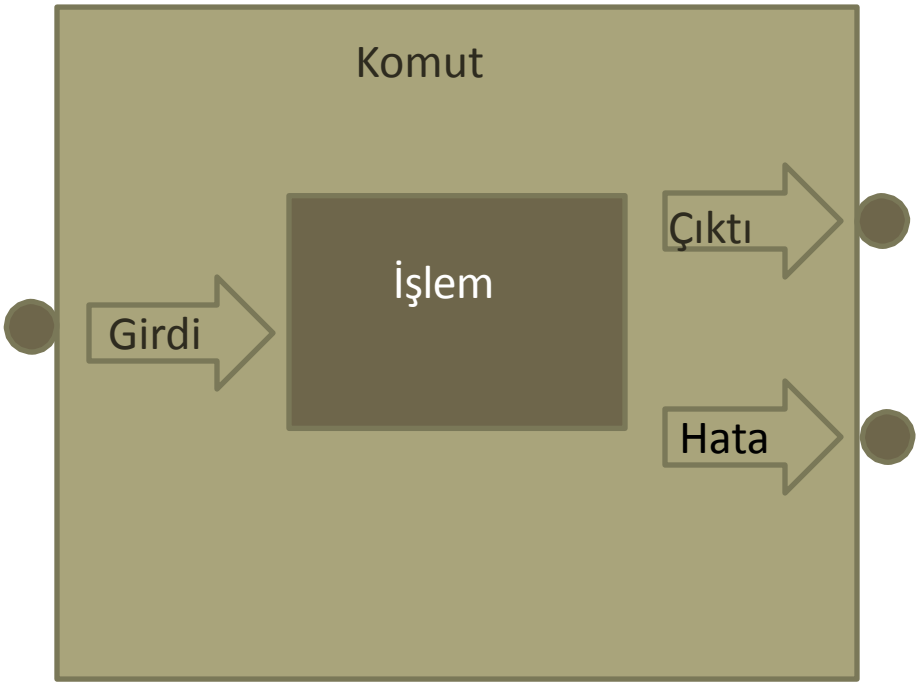
- Linux komutları aşağıdaki felsefeleri takip eder;

-Bir görevi iyi gerçekleştiren küçük, taşınabilir, uzmanlaşmış programlar,

- Bu programlar diğer programlardan girdi alabilir ve diğer programlara yönlendirebilir.

- Linux Komutları LOGO gibidir.
- Bu felsefe böyle küçük sonuçlardan oluşur fakat birleştiğinde çok güçlü olurlar.

Komut Modeli



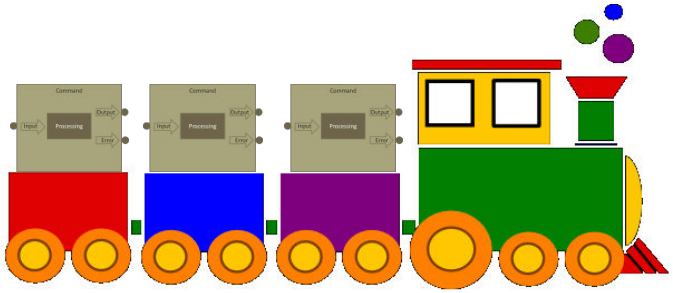
I/O Akışları

Fonksiyon	Akış İsmi	Akış açıklayıcı	Varsayılan Aygıt
Giriş	stdin	0	Klavye
Çıkış	stdout	1	Ekran
Hata	stderr	2	Ekran

Birleşik Komutlar

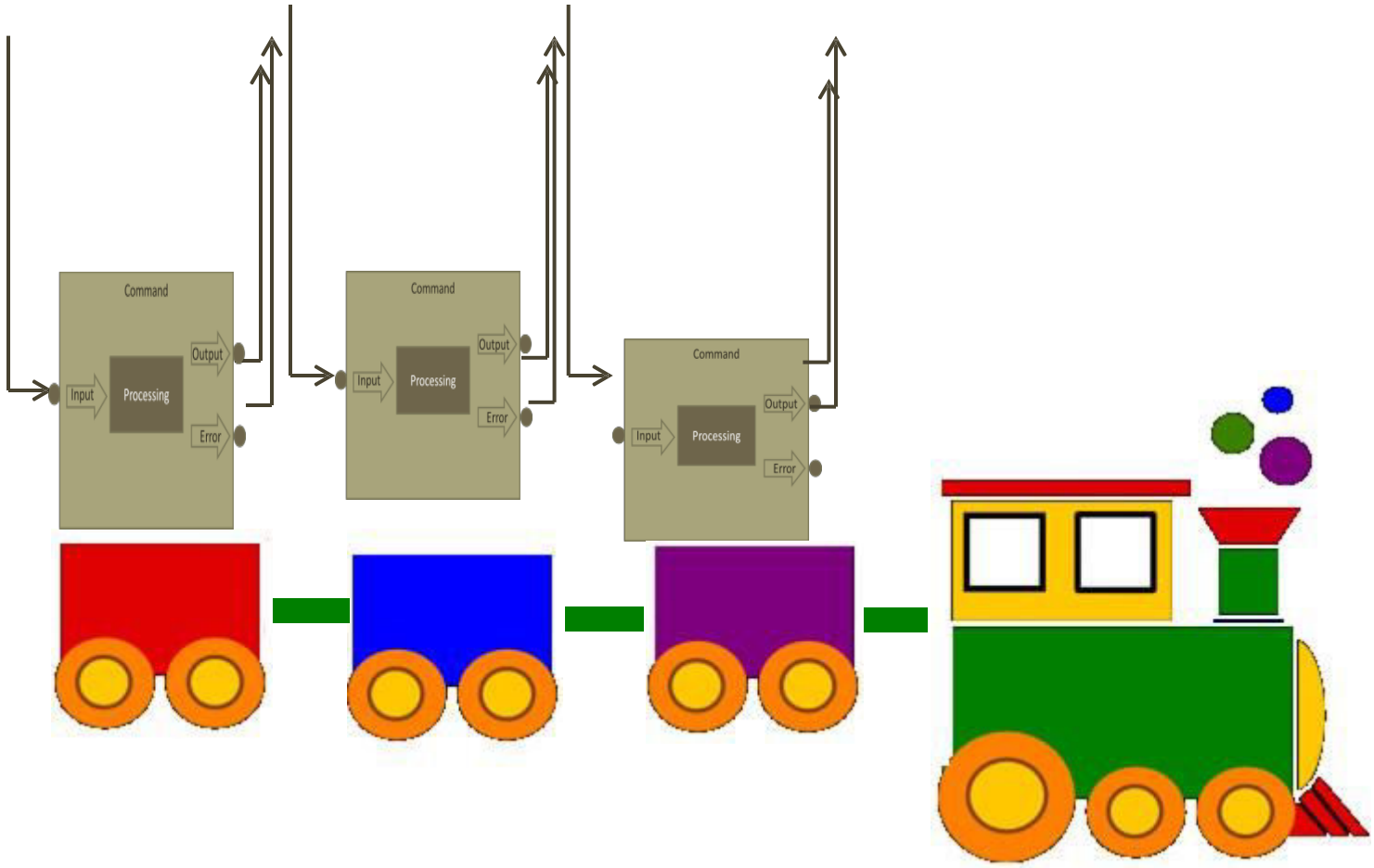
Bileşik Komutlar alt başlıkları;

- Sıralı Komutlar
- Şartlı Komutlar
- Komut Döngüleri
- Giriş/Çıkış Yönlendirme
- Borular(Pipes)
- Komut Argümanının Genişletilmesi
- Komut Argümanı Alıntılama



SIRALI Komutlar
(SEQUENTIAL) COMMANDS

Bağımsız Giriş - Çıkışlar



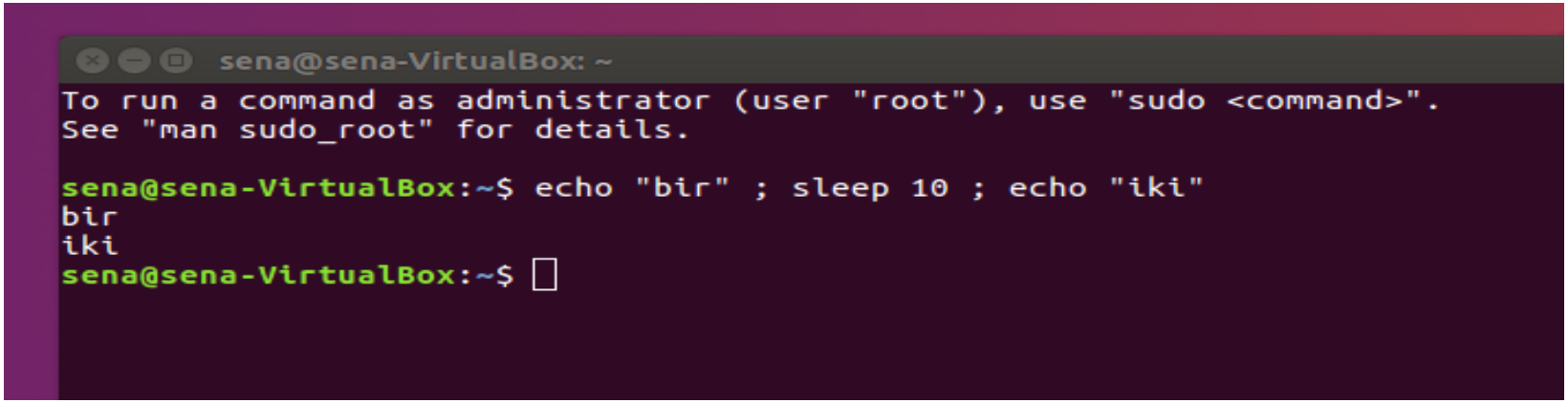
Sıralı Komutlar

Aynı satırda birden fazla komut gerçekleştirebiliriz.

$ <ilk komut> ; <ikinci komut> ; <üçüncü komut>

$ echo "Bir" ; sleep 10; echo "İki"

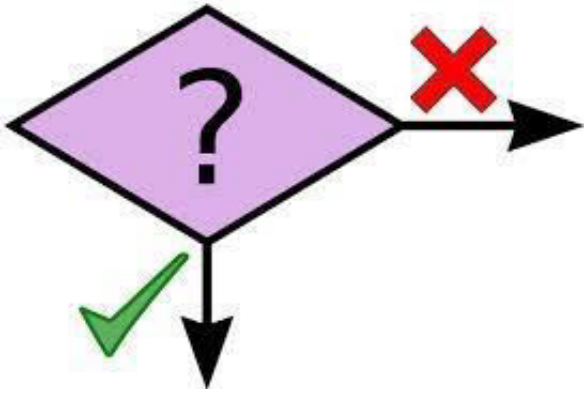
-Ekrana 'Bir' yazar, sistem 10 saniye bekler, ekrana 'iki' yazar.



Sıralı Komutlar

- Sıralı komutlar, ilk komutun yürütülmesi uzun sürdüğü ve tamamlanıncaya kadar beklemek istemediğimizde kullanılınca avantajlıdır.
- ***$ make ; sudo make install***
- ***-Derleme işlemi yapılır, uygulama kurulur.***

Not: Ardıl Komutlar birbiri ardına çalışır ve bağımsız Girdi/Çıktıları vardır.



ŞARTLI (CONDITIONAL) KOMUTLAR

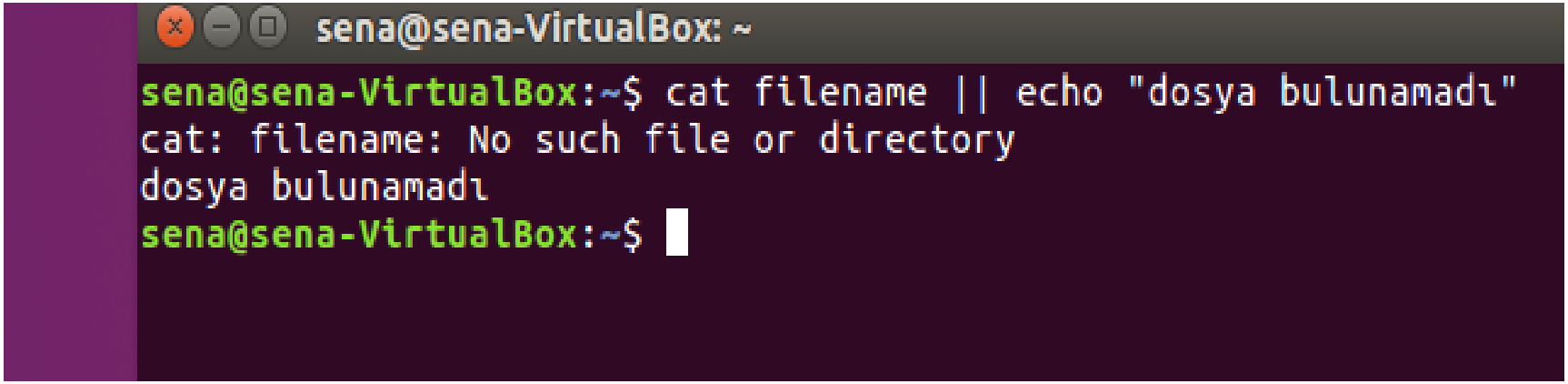
Şartlı Komutlar

OR (“||”)

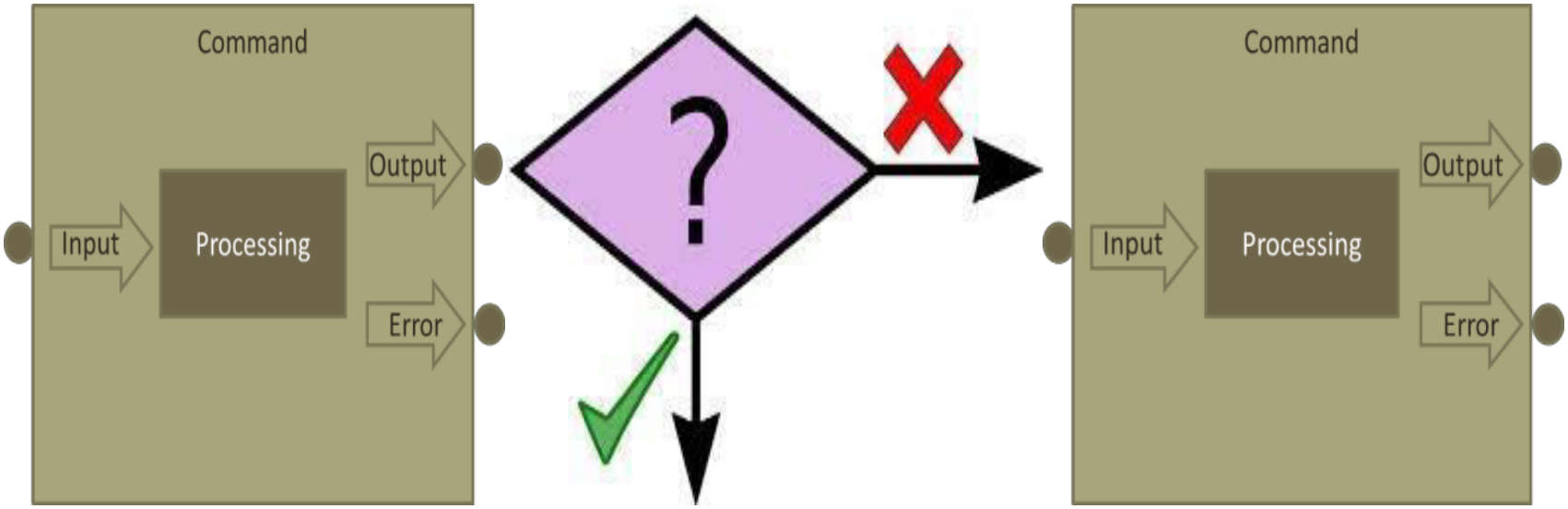
İkinci komut sadece ilk komutta hata dönerse çalışacaktır.

$ cat <filename> || echo “File Not Found”

-Dosya içeriğini göstermede hata alınırsa ikinci komut çalışır.

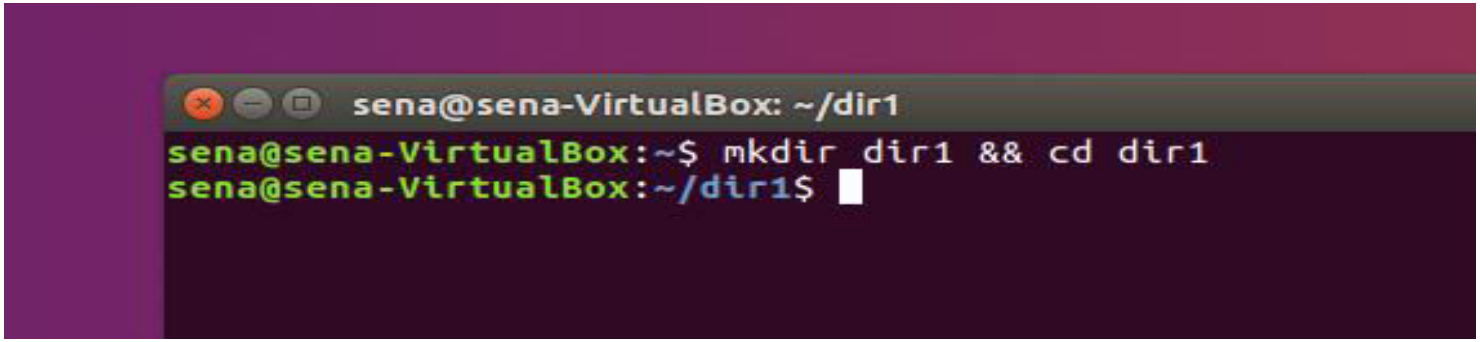


Şartlı Komutlar

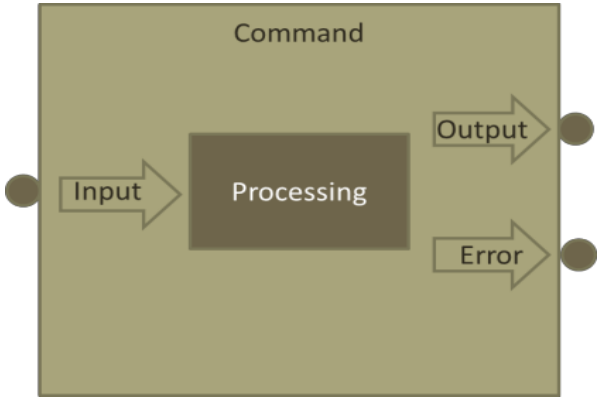


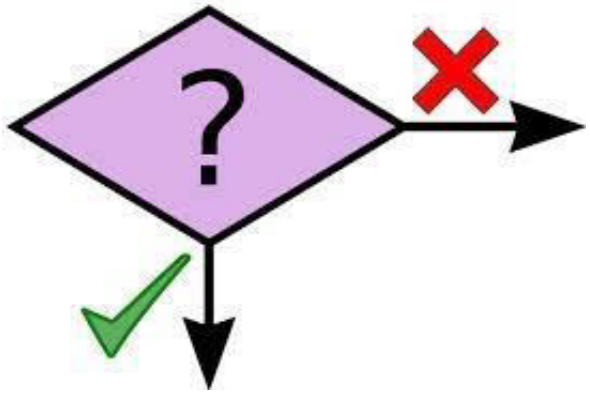
Şartlı Komutlar

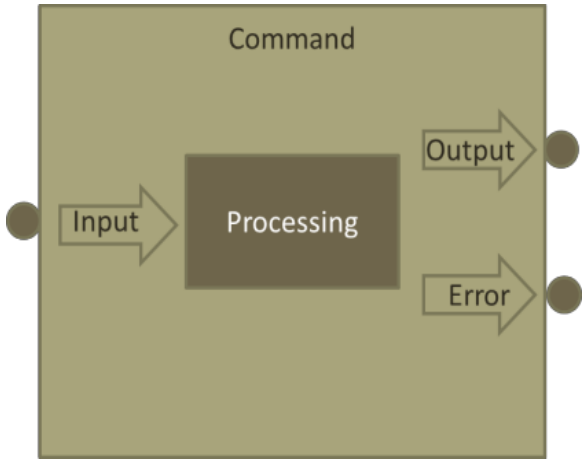
- AND (“**&&**”)
- İkinci komut sadece ilk komut başarılı olarak çalıştırılıp dönerse çalışır.
  - **$ mkdir dir1 && cd dir1**
    - **dir1 klasörü oluşturulur, sonra o dizinde cd komutu ile gidilir.**
- ‘’Başarılı olduğu sürece devam et’’



Şartlı Komutlar







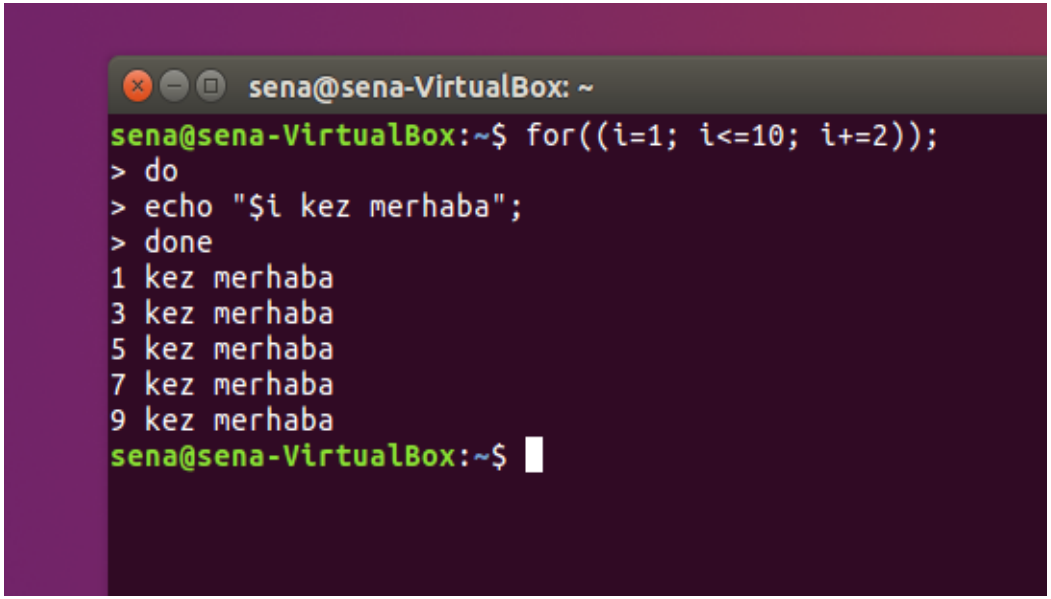
KOMUT DÖNGÜLERİ

Komut Döngüleri

Komutun birden çok kez çalıştırılmasını istediğimizde bir döngü inşa edebiliriz
Örnek:

```
for((i=1;i<=10;i+=2));
>do
>echo "$i kez merhaba";
>done
```

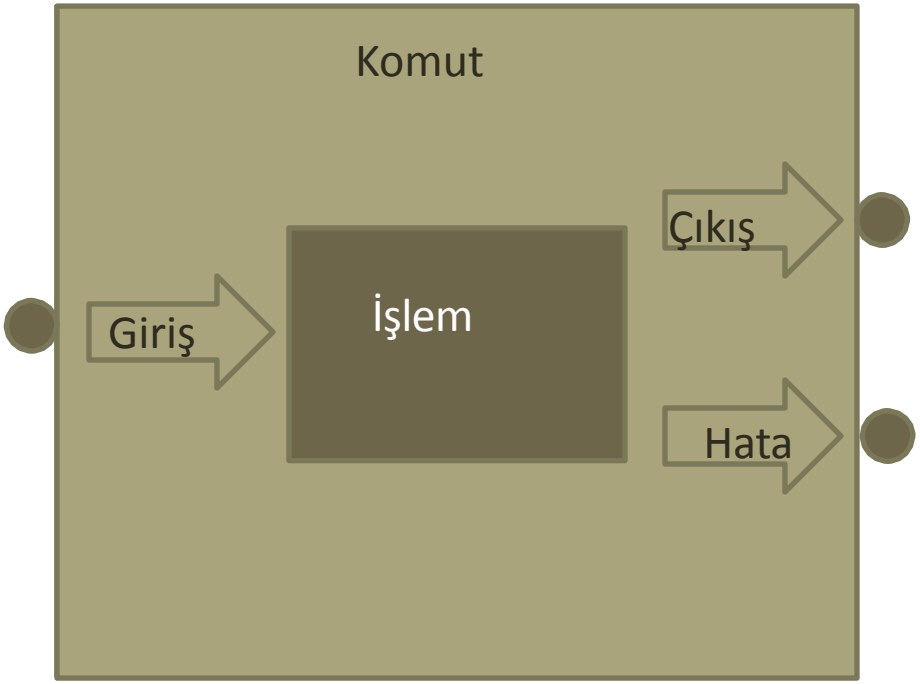
- i=1 den ikişer artarak 10 kadar Merhaba yaz



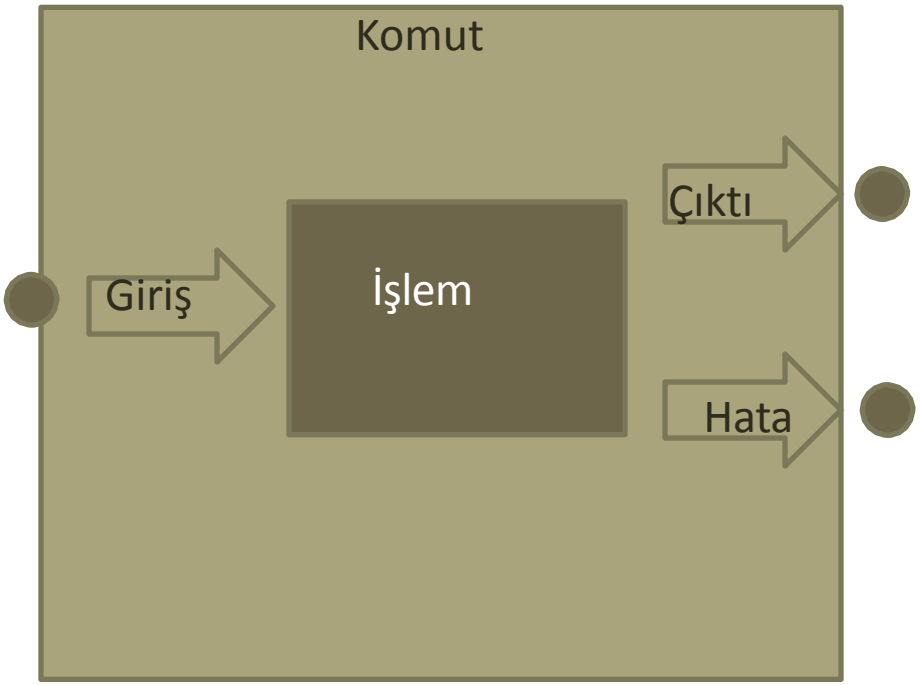
Döngüler normal olarak komut dosyalarında kullanılır, ancak bazen komut satırında da kullanılır.

GİRİŞ/ÇIKIŞ YÖNLENDİRME

I/O Yönlendirme

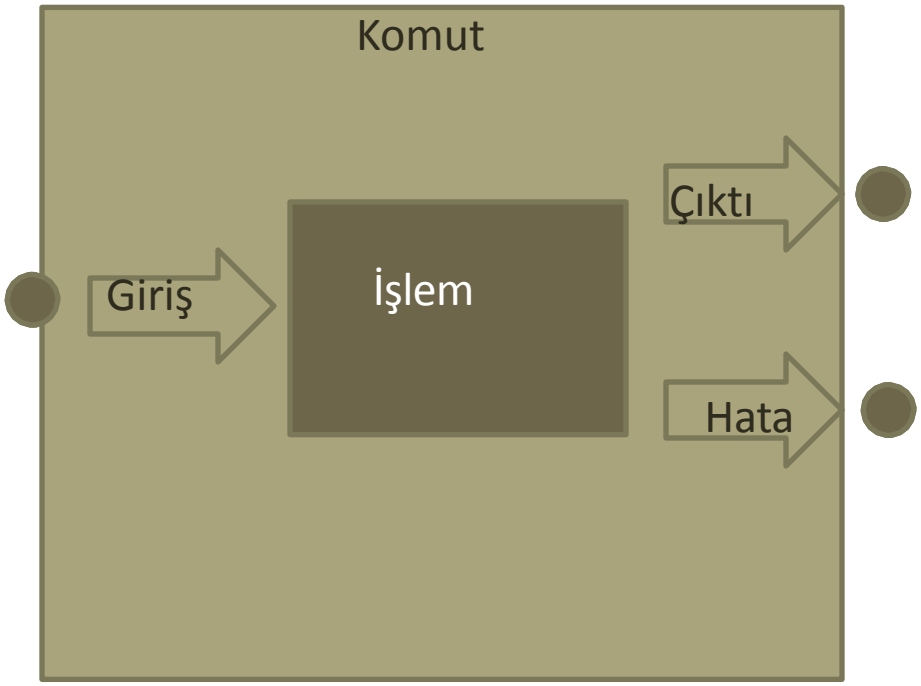


I/O Yönlendirme



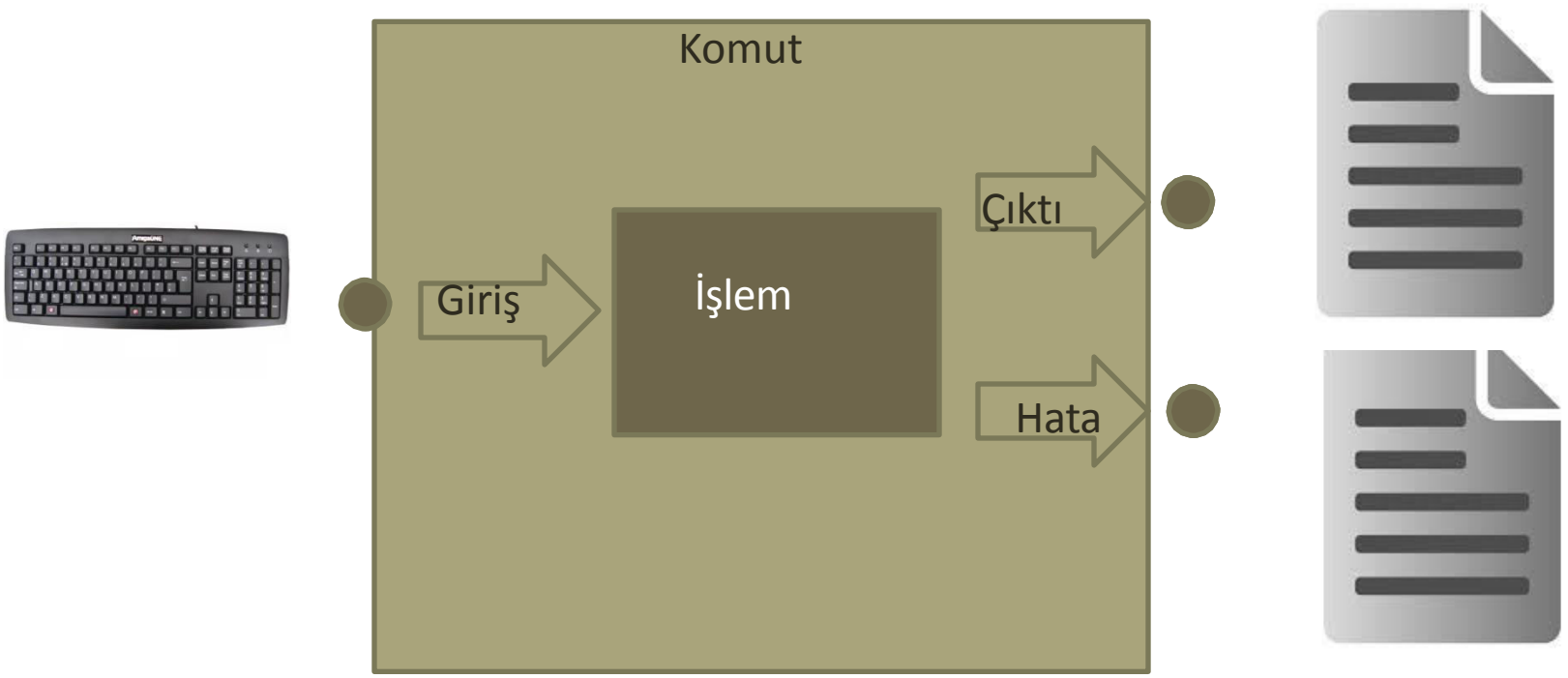
Çıktıyı Dosyaya Yönlendirme

I/O Yönlendirme



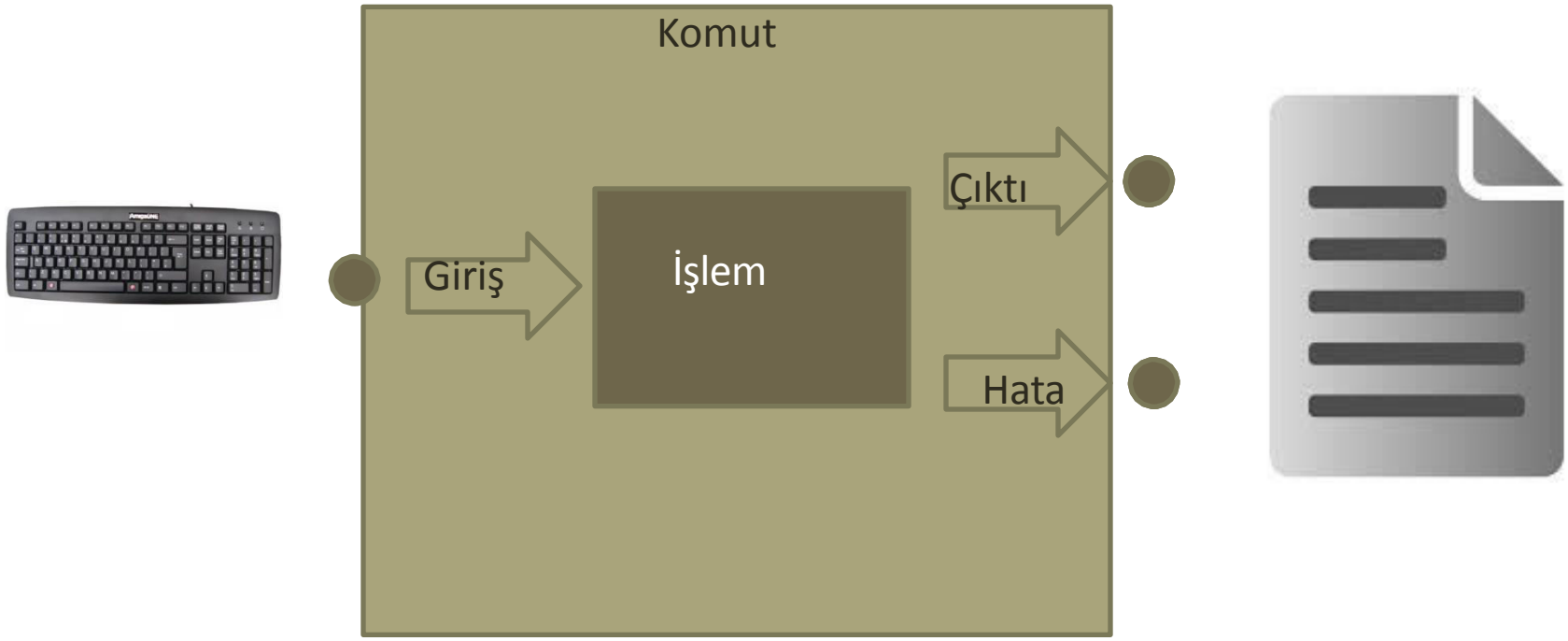
Hatayı Dosyaya Yönlendirme

I/O Yönlendirme



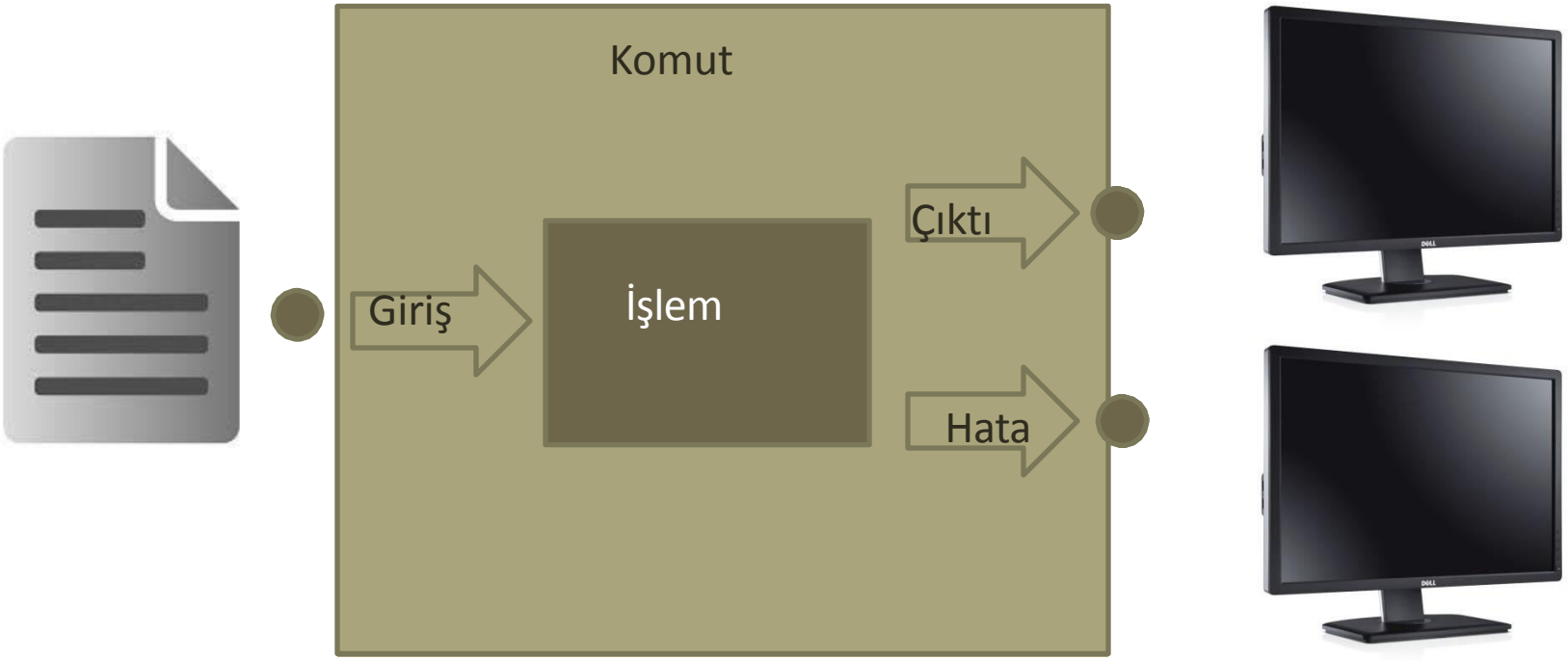
Çıktı ve hatayı Dosyaya yönlendirme

I/O Yönlendirme



Çıktı ve hatayı aynı dosyaya yönlendirme

I/O Yönlendirme



Girişi dosyadan yeniden yönlendirme

Standart Çıkış Yönlendirme

$ <Command> > file (Çıktıyı dosyaya yönlendir, üzerine yaz)
$ <Command> >> file (Çıktıyı dosyaya yönlendir, üstüne ekle)

$ echo "Hello world" (Çıktı ekrana gelir)

$ echo "Hello world" > greeting.txt (Çıktı dosyanın üzerine yazılır)

$ ls -al /usr/bin >> file-listing.txt (Çıktı dosyanın üstüne eklenerek yazılır)

$ cat file1 file2 > combined-file.txt

Not:

1. Hata mesajları ekrana gelir.
2. "***>***" -> "***1>***" anlamına gelir.
   (Çıktı yönlendirme)

Standart Hata Yönlendirme

$ <Command> 2> file (Hatayı dosyaya yönlendirir, üstüne yazar)

$ <Command> 2>> file (Hatayı dosyaya yönlendirir, üstüne ekler)

$ make (Çıktı ve hata ekrana yazar)

$ make 2> log (Çıktı ekrana yazılır;
Hata mesajı dosyaya yazılır, üstüne yazar)

$ make 2>> log (Çıktı ekrana yazılır;
Hata mesajı dosyaya yazılır, üstüne eklenir)

Hata & Çıkış Yönlendirme

- Çıktı ve hata farklı dosyalara gider,

  $ <command> >file1 2>file2 (çıktı file1, hata file2)

  $ <command> >>file1 2>>file2 (çıktı file1, hata file2)

- Çıktı ve hata aynı dosyaya gider,

  $ <command> >file 2>&1 (file)

  $ <command> >>file 2>>&1 (file)

- Aynı komutun kısa versiyonu,

  $ <command> &>file (file)

  $ <command> &>>file (file)

Standart Giriş Yönlendirme

$ command < file (komutun girdisi file dosyasından okunur)

Örnekler:

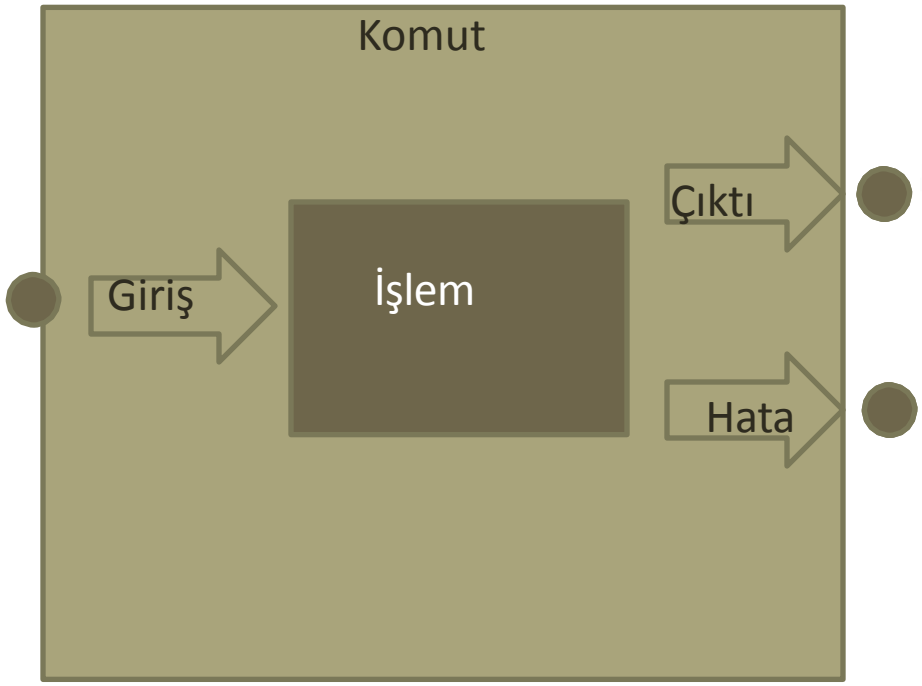
$ wc -l < log-file.log

$ sort < log-file > sorted-log-file

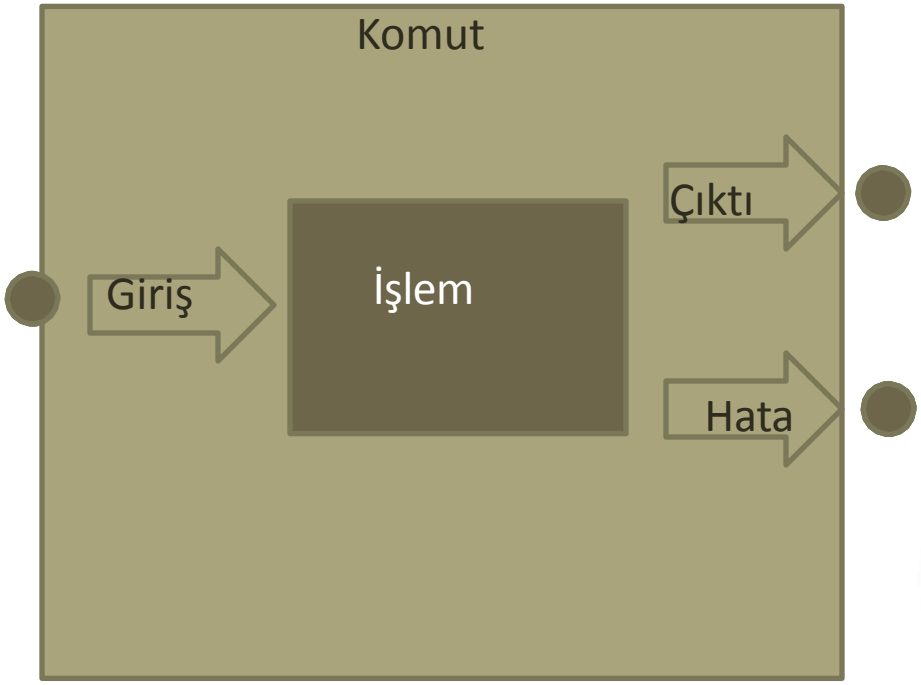
$ mail ceo@company.com < resume

$ spell < report.txt > error.log

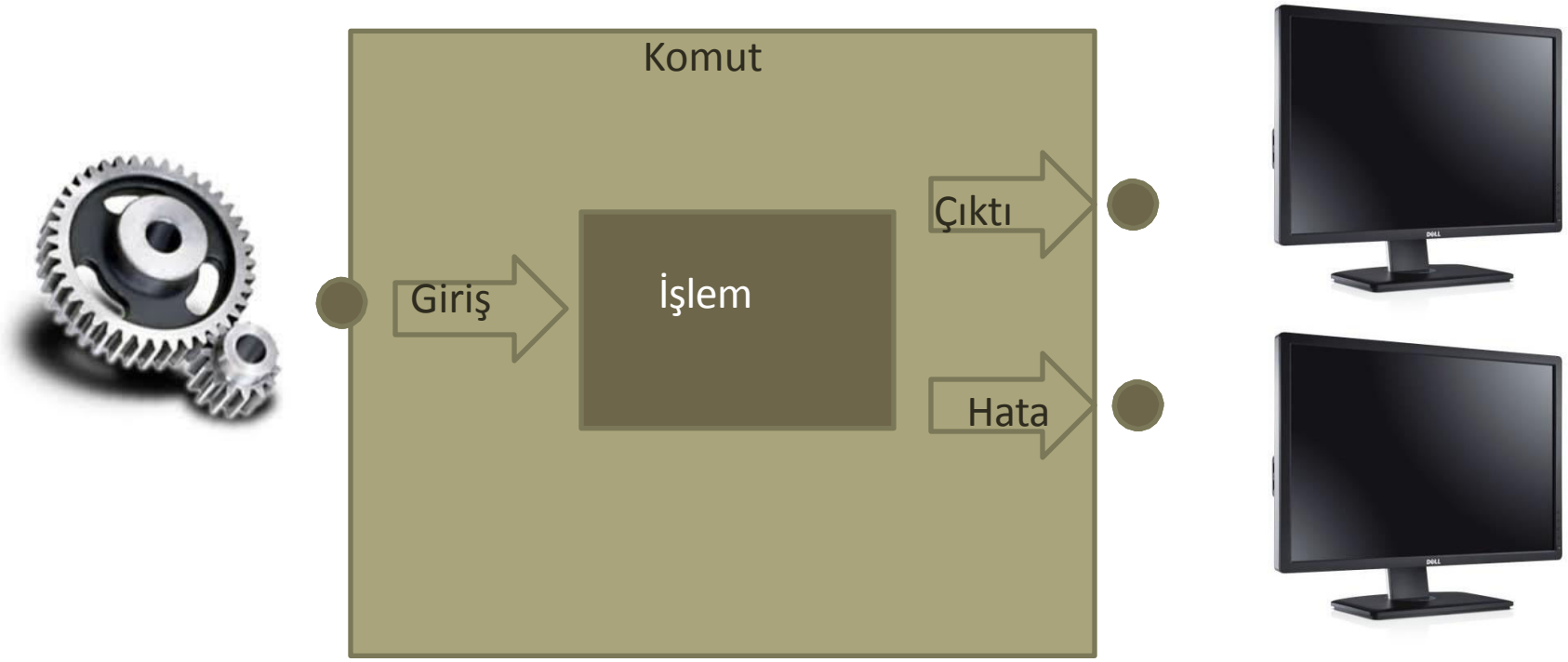
Aygıtlar ile I/O Yönlendirme



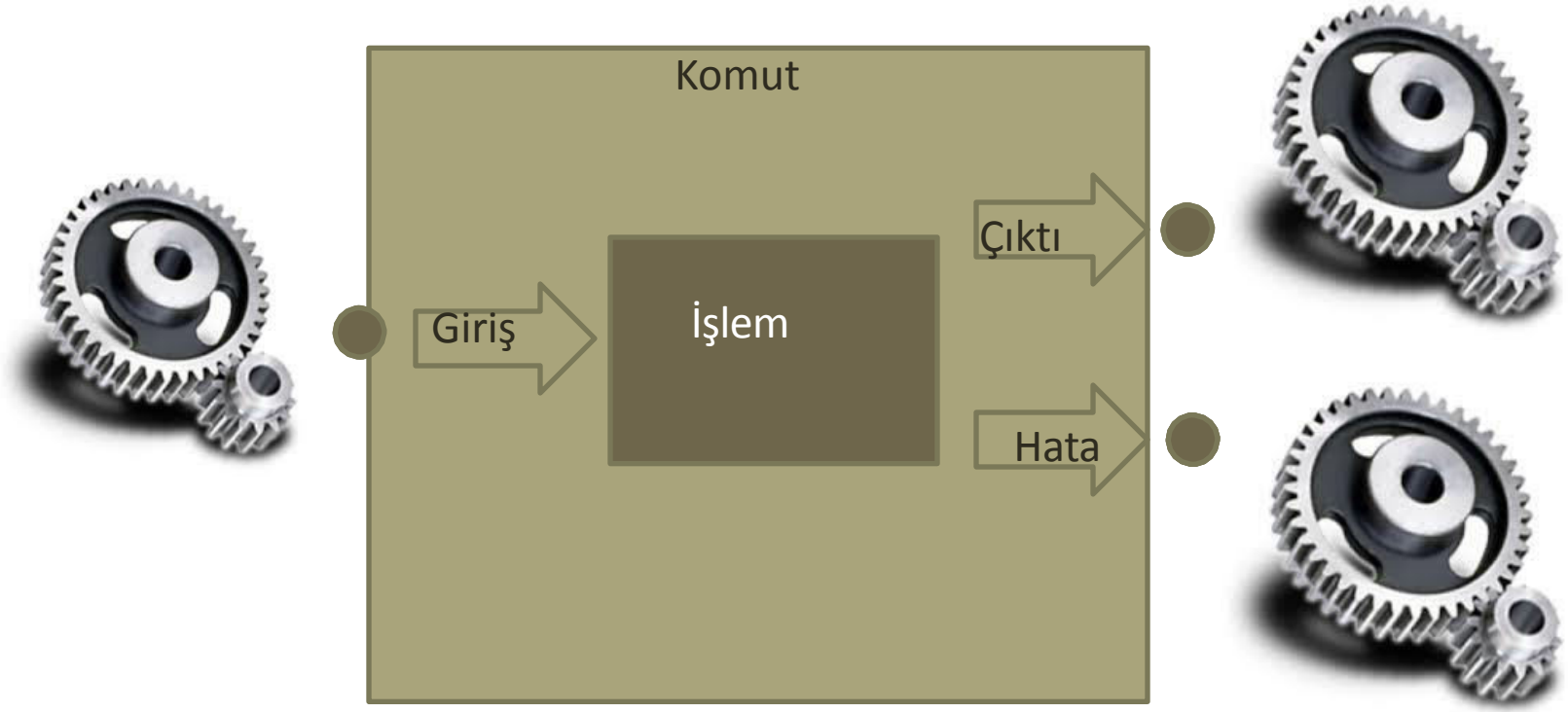
Aygıtlar ile I/O Yönlendirme



Aygıtlar ile I/O Yönlendirme



Aygıtlar ile I/O Yönlendirme



Yönlendirmeler için ortak aygıtlar

1. ***/dev/stdout*** : Standard çıktı aygıtı
2. ***/dev/stderr***: Standard hata aygıtı
3. ***/dev/stdin***: Standard giriş aygıtı

Varsayılan

4. ***/dev/null*** :

   Bu cihaz, veri biriktirmek için kullanışlıdır. Bu, komut çıktısını atmak istediğimizde yararlıdır.(uzun çıktılar için)
5. ***/dev/zero***

   Bu cihaz sıfır üretmek ve sonsuz akışı sağlamak için bir giriş cihazı olarak kullanışlıdır.

   /dev/random

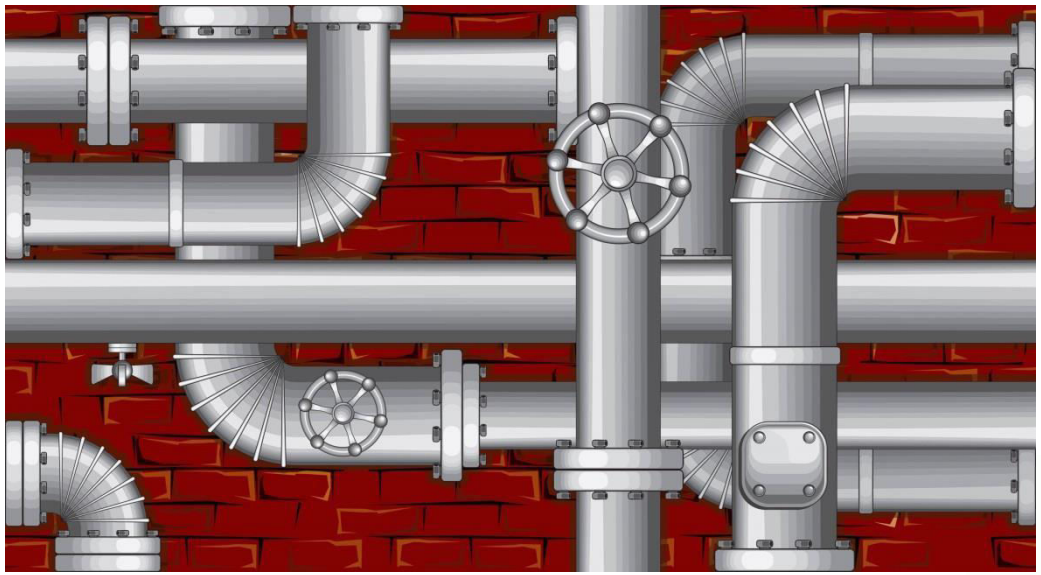
   Bu cihaz, rastgele baytlar üretmek için bir giriş cihazı olarak kullanışlıdır. Bloke edebilir.

   /dev/urandom

   Bu cihaz, yarı rasgele bayt üretmek için bir giriş cihazı olarak kullanışlıdır. Bloke edilmez.

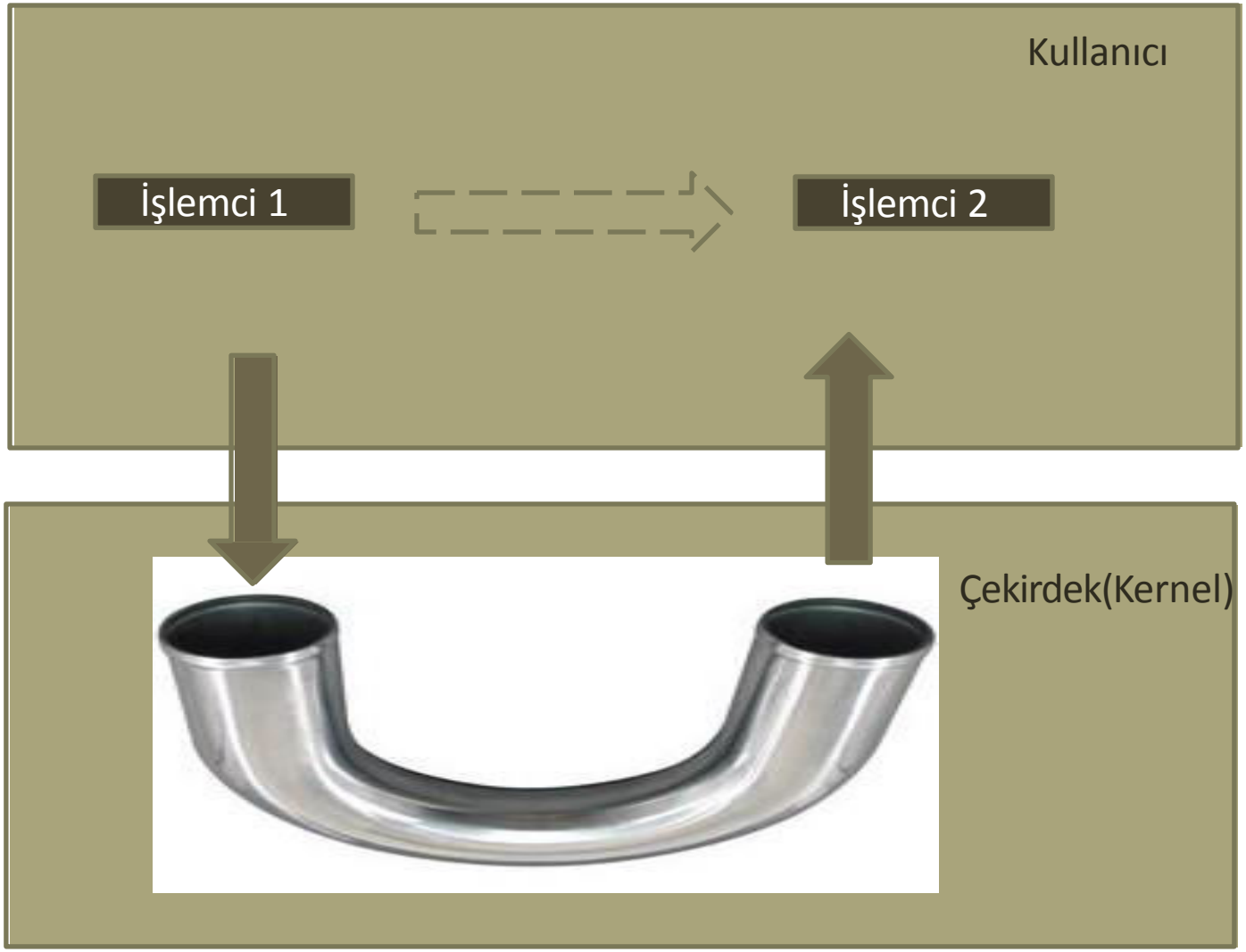
   /dev/full

   Bu cihaz, tam bir dosyayı taklit etmek için bir çıkış cihazı olarak kullanılır. Test amaçlı kullanılır.



BORULAR(PIPES)

Boru(Pipe) Nedir?



Boru(Pipe) nedir?

- Boru, Linux Çekirdeğinde, bir işlemin diğer işleme bilgi göndermesine olanak tanıyan bir mekanizmadır.
- IPC de denir.(Inter Process Communication)

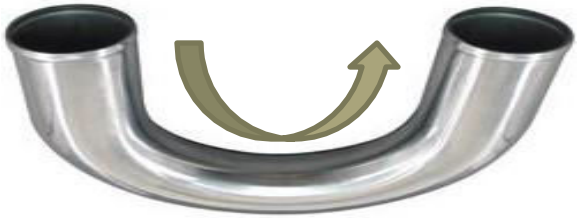
- Boru tek yönlü bir mekanizmadır, bu nedenle her iki yönde de akışa ihtiyaç duyarsanız, iki boru kullanmanız gerekir. (Her biri için bir tane)

Pipe Kullanımı

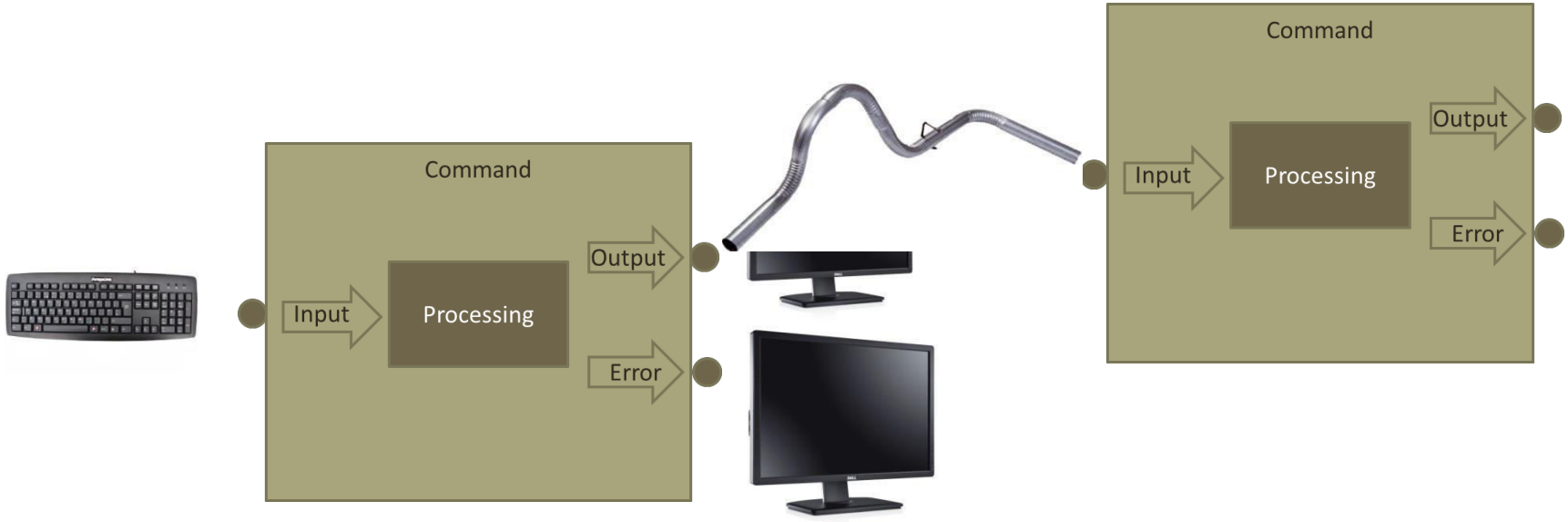
```
sena@sena-VirtualBox: ~
sena@sena-VirtualBox:~$ ls -al | grep "a"
total 124
drwxr-xr-x 16 sena sena 4096 May 21 13:07 .
drwxr-xr-x  3 root root 4096 Mar 18 23:08 ..
-rw-------  1 sena sena 3912 Nis 14 07:05 .bash_history
-rw-r--r--  1 sena sena  220 Mar 18 23:08 .bash_logout
-rw-r--r--  1 sena sena 3771 Mar 18 23:08 .bashrc
drwx------ 14 sena sena 4096 Nis  2 10:54 .cache
drwx------ 16 sena sena 4096 Mar 21 22:54 .config
```

```
sena@sena-VirtualBox: ~
sena@sena-VirtualBox:~$ ls -al
total 124
drwxr-xr-x 16 sena sena 4096 May 21 13:07 .
drwxr-xr-x  3 root root 4096 Mar 18 23:08 ..
-rw-------  1 sena sena 3912 Nis 14 07:05 .bash_history
-rw-r--r--  1 sena sena  220 Mar 18 23:08 .bash_logout
-rw-r--r--  1 sena sena 3771 Mar 18 23:08 .bashrc
drwx------ 14 sena sena 4096 Nis  2 10:54 .cache
drwx------ 16 sena sena 4096 Mar 21 22:54 .config
drwxr-xr-x  2 sena sena 4096 Nis  2 10:56 Desktop
```

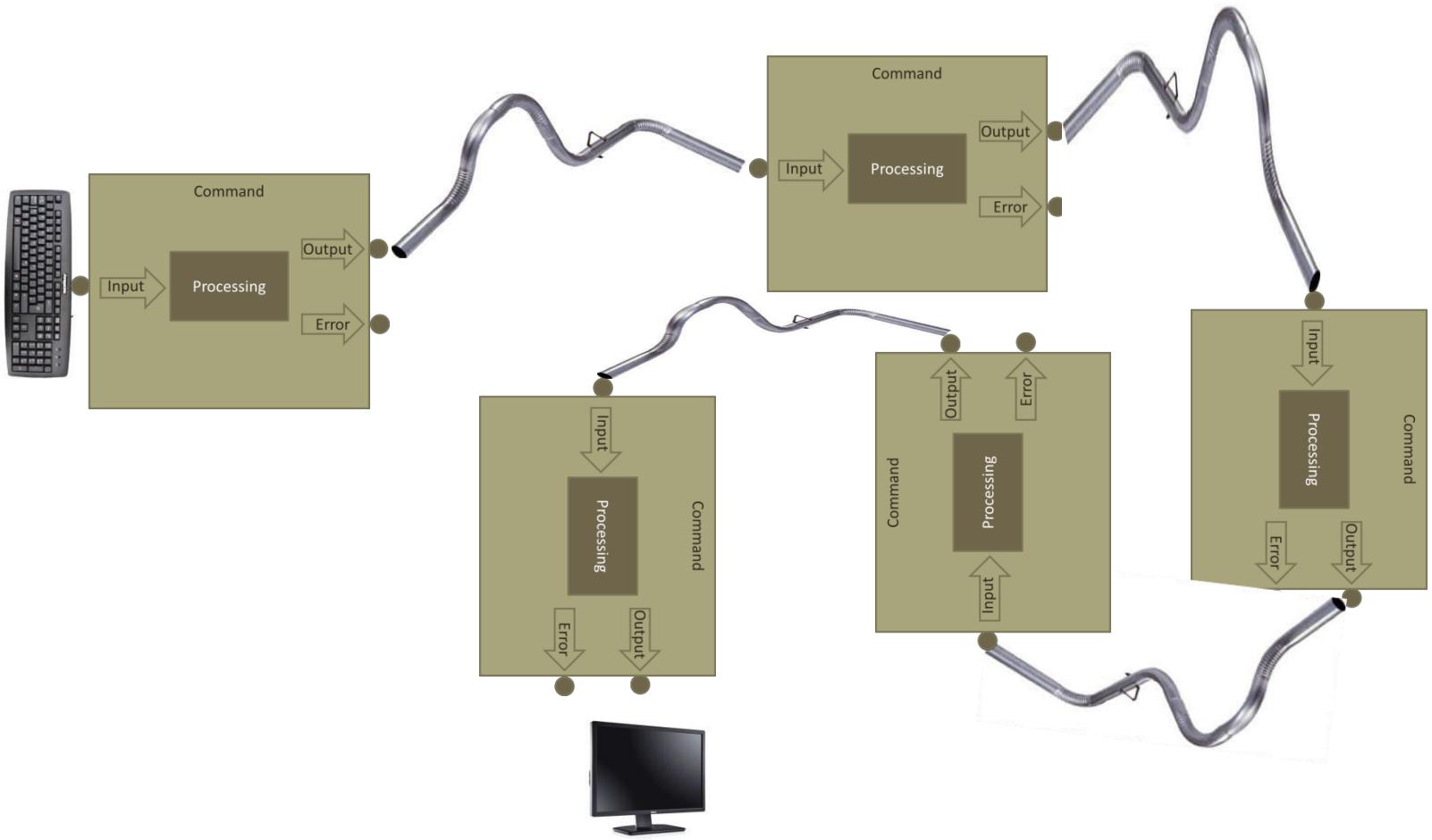
```
sena@sena-VirtualBox: ~
sena@sena-VirtualBox:~$ grep "a"
```



Pipe Kullanımı



Pipe Kullanımı



Pipe Kullanımı

- Linux'taki borular farklı süreçler arasındaki iletişim için bir yöntemdir. (Inter-Process Communication)
- Komut satırı Arayüzü'nde iki komut çalıştırabilir ve bunları bir boruyla bağlayabiliriz, böylece bir komutun çıktısı bir başka komutun girdisi olur.
- Her komutun, kabuğun bir alt kabuğunda olduğunu unutmayın(child Shell-alt kabuk)

Pipe Kullanımı

$ <command1 > | <command2> | <command3>

- Örnekler:

$ cat log-file.log | grep "Error"

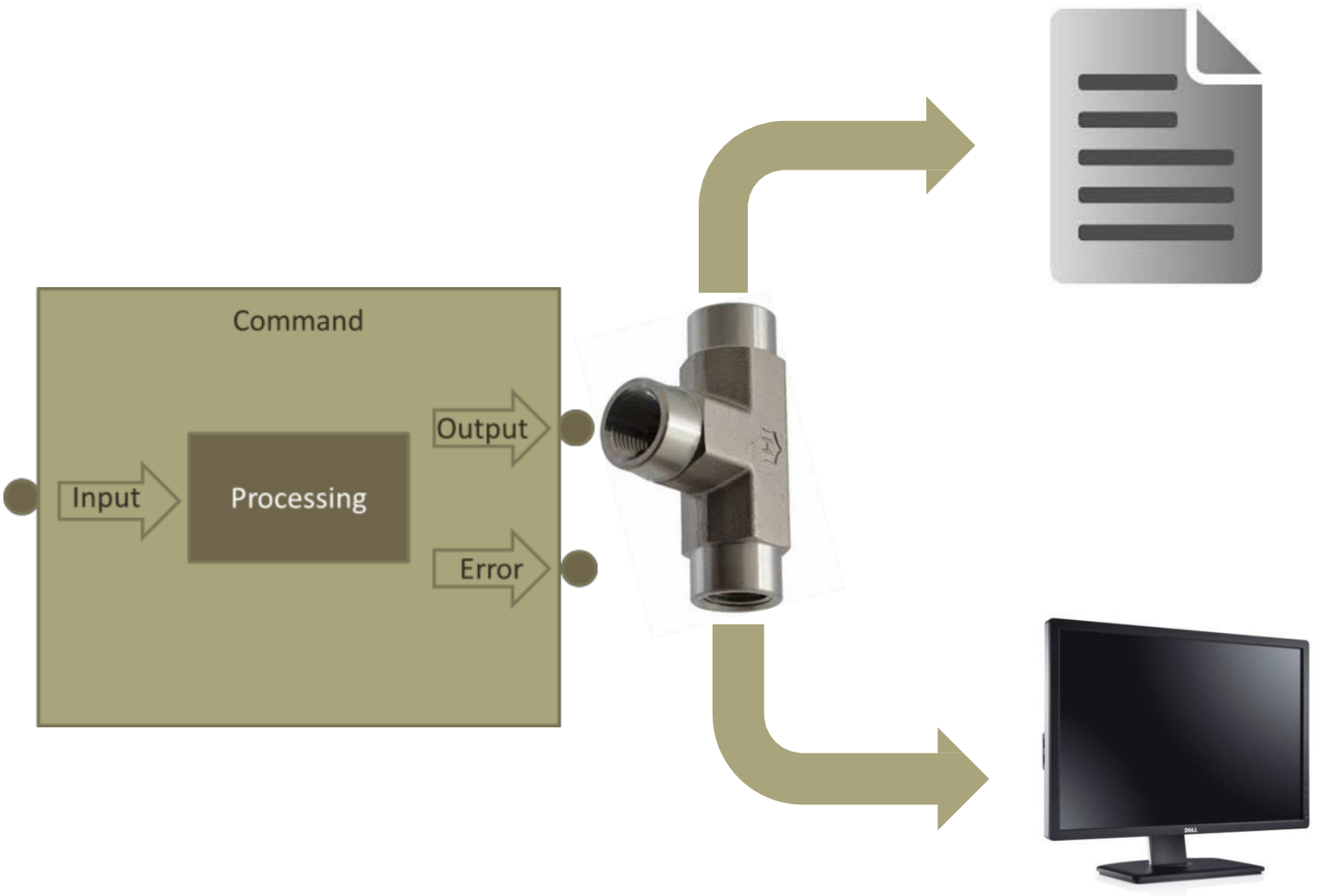
$ man gzip | grep -i "compress"

$ cat name-list.txt | sort

$ cat * | grep -i "error" | grep -v "severe" | sort > file.log

$ cat resume | mail bill-gates@microsoft.com

Pipe Kullanımı (tee) komutu



tee Komutu

<Command > | tee <list of Sinks>

Tee komutu çıktıları bir dosyaya ve standart çıkış'a gönderir.

- Örnekler:

  $ make |tee make.out.txt (Çıktıyı ekrana ve dosyaya gönderir)

  $make |tee -a make.out.txt (aynı fakat üstüne ekler)

  $ date | tee file1 file2 file3 file4 (Tarih bilgisi hem ekrana hem dosyaya gönderilir)

  $ make |tee make.out.txt >> file2 (çıktı make/out.txt‘ e gider
  Ve dosya2 nin üstüne ekler)

  $ make | tee make.out.txt | grep “error” (çıktı make/out.txt‘ e gider ve onun içindende grep komutu ile error kelimesi aranır.)

Yes komutu

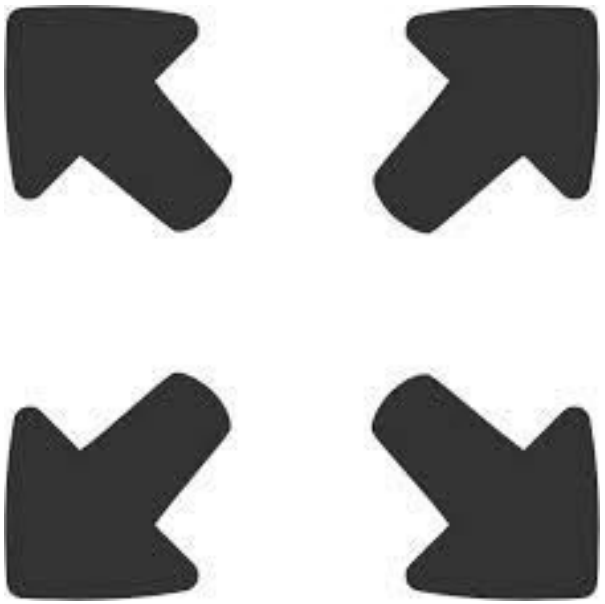
$ yes <string> |<command>

- "Evet" komutu bir dize komut istemine gönderir.
- Varsayılan string = “***y***”

Örnekler:

$ yes |rm -r ~/project-folder/

$ yes “ “ |make config (Varsayılanları kabul etmek için boş dizgiyi iletir)



KOMUT ARGUMANI GENİŞLETME

Özel karakterler (*, ?, ...)

Dosya ve dizin adları için geçerlidir.

Örnekler:

$ cp * ../project/

$ cat * > log-files

$ echo *

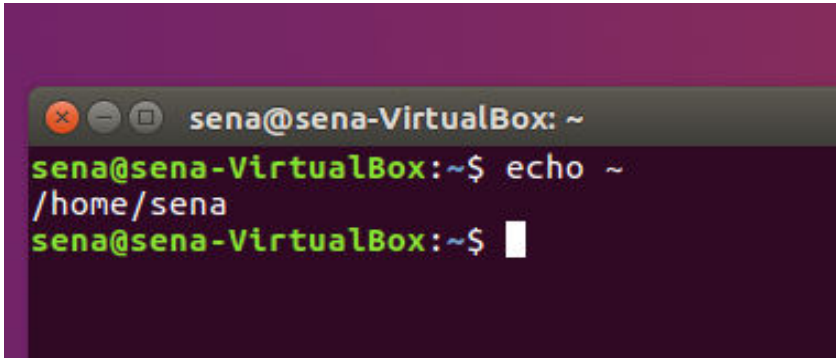
Tilde İşareti (~)

- Tilde , kullanıcı ev yoluna genişletir.

/home/<username>

Örnek:

$ echo ~ ➔



Komut Argümanı Genişletme
Parametre ($ ve ${ })

Parametre Genişletme , kabuk veya ortam değişkenlerini değerlendirmek için kullanılır.

Örnekler:

$ echo $PATH

$ echo ${PATH}

$ MY_NAME=TOM

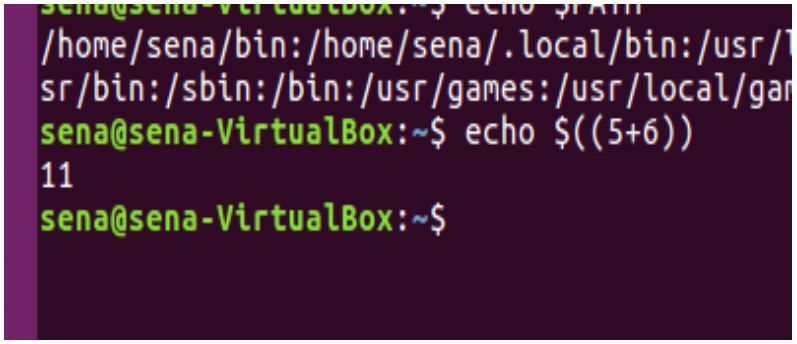
$ echo "My Name Is $MY_NAME"

Komut Argümanı Genişletme Aritmetik($(( )))

Aritmetik Genişleme, değerlendirilecek aritmetik ifadeleri koymak için kullanılır.

Örnek:

- ***$ echo $((5 + 6))*** ➔ 11
- ***$ echo $((((5**2)) * 3))*** ➔ 75
- ***$ echo $((5 % 2))*** ➔ 1
- ***$echo $((5/2))*** ➔ 2



Komut Argümanı Genişletme
Bağ ({})

Bağ Genişletme, köşe parantezlerin içinde açıklandığı gibi bir değerler seti ile sonuçlanır.

Örnekler:

$ echo abc-{A,B,C}def ➔ abc-Adef abc-Bdef abc-Cdef

$ echo a{A{1,2} , B{3,4} }b ➔ aA1b aA2b aB3b aB4b

Benzer şekilde açılımlar kullanılabilir.

{1..5}

{A..Z}

{Z..A}

Komut Argümanı Alıntılama

Çift Tırnak Alıntılama(“ ”)

Çift tırnakla alıntılama,

- Stringleri korumak için kullanılır (spaces, wild cards, tilde, ….)

  $ echo “*” ➔ *

- Yedek komutu koruma

  $ echo “ls” ➔ ls

- Aritmetik genişlemeyi koruma

  $ echo “ $((5 + 6)) “ ➔ $((5 +6))

- Değişken geri alımı durdurmaz

  $ echo “My Name Is $MY_NAME” ➔ My Name Is Tom

Örnekler:

$ echo “cp *.*” ➔ cp *.*

$ echo “$HOME” ➔ /home/sena

Tek tırnak Alıntılama (‘ ’)

Tek tırnakla alıntılama,

- Tırnaklar arasına yazılan ifadenin string olarak alınmasını sağlar.
- Çift tırnak kullanıldığında tırnaklar içinde bulunan değişkenin içindeki veriler kullanılırken tek tırnak kullanıldığında değişkenin adı kullanılmış olur.

```
linuxagyonetimi@ubuntu:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
linuxagyonetimi@ubuntu:~$ echo '$PATH'
$PATH
linuxagyonetimi@ubuntu:~$
```

- Örnekler:

  $ echo ‘$PATH’ ➔ $PATH

  $ echo ‘My Name Is $MY_NAME’ ➔ My Name Is $MY_NAME

  Yani echo $PATH olarak kullanıldığında $PATH içerisnde yer alan bilgiyi ekrana yazacakken echo ‘$PATH’ şeklinde kullandığımızda ekrana $PATH olarak yazacaktır.

Geri Alıntılama (‘ ‘)

Geri Alıntılar , komutlar / dizeler içerisinde komutlar yapmak için kullanılır . Tırnaklar arasına yazılmış komutun geri döndüreceği cevabın yerine yazılmış olur. Yani örnekle açıklamak gerekirse adı bugünün tarihi olan bir dosya oluşturmak istediğinde

$ touch `date +%m-%d-%Y`

kullanılırsa otomatik olarak dosya adı bugünün tarihini olarak ayarlanmış olarak oluşturulucaktır.

Örnekler:

$ cd /lib/modules/`uname -r`

ile

$ cd /lib/modules/3.11.0-15-generic

Komutları eşdeğerdir.

$(<command>) , ***`command`*** ya da ***$(command)*** ***ile değiştirilebilir***

Örnek:

$ cd /lib/modules/$(uname -r)

$ pushd `pwd`

Kaynakça

- Ahmed ElArabawy, Linux for Embedded Systems for Arabs

Teşekkürler.

Dersin Sonu

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr/