



ANKARA UNIVERSITY CYBER CLUB

BASH PROGRAMMING



~ KONU BAŞLIKLARI

- ~ BASH NEDİR ?
- ~ BASH SCRIPT NEDİR ?
- ~ ÖZEL KARAKTERLER
- ~ DEĞİŞKENLER VE PARAMETRELER
- ~ DÖNÜŞ DEĞERLERİ



~ KONU BAŞLIKLARI

- ~ SABİT DEĞİŞKENLER
- ~ DÖNGÜLER
- ~ KOMUTLAR
- ~ DÜZENLİ İFADELER
- ~ GİRDİ VE ÇIKTI YÖNLENDİRME



~ KONU BAŞLIKLARI

~ FONKSİYONLAR

~ DİZİLER

~ LİSTELER

~ HATA AYIKLAMA (DEBUGGING)

~ SORULAR



No programming language is perfect. There is not even a single best language; there are only languages well suited or perhaps poorly suited for particular purposes.

Herbert MAYER

~ BASH NEDİR ?

Bash GNU işletim sistemi için bir kabuk ya da başka bir deyişle komut dili yorumlayıcısıdır. **Bourne-Again Shell** sözcüklerinden türetilmiş bir kısaltmadır. Bell Araştırma Laboratuari'nın Unix'inin yedinci sürümündeki Unix kabuğu olan sh'ın atasının yazarı Stephen Bourne'a atfen bu isim verilmiştir.





~ BASH NEDİR ?

Bash, sh'ın hemen hemen tüm özelliklerini ve Korn kabuğu olan ksh ile C kabuğu olarak bilinen csh'ın kullanışlı özelliklerini bir araya getirir. Aynı zamanda sh'ın hem etkileşimli hem de programlama için kullanımını işlevsel olarak artıran geliştirmeler içerir.



~ BASH NEDİR ?

GNU işletim sistemi, csh'ın bir sürümü de dahil olmak üzere başka kabuklarla da uyumlu olsa da Bash öntanımlı kabuktur. Diğer GNU yazılımları gibi Bash de bir çok işletim sistemine uyarlanabilir. MS-DOS, OS/2 ve Windows platformları için bağımsız olarak desteklenen sürümleri vardır.



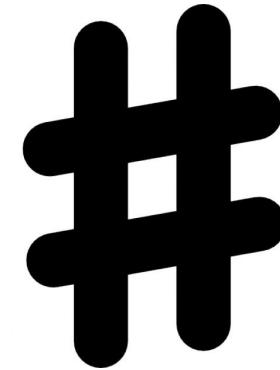
~ BASH SCRIPT NEDİR ?

LINUX sistemlerde defalarca yazmanın zor olacağı karmaşık işlemleri çok hızlı gerçekleştirmek için döngüler ve case deyimleri gibi programlama yapılarını kullanarak komutları bir dosyada bir araya getirebiliriz. Bu şekilde bir dosyada toplanan ve çalıştırılan komutları içeren programlara **bash scriptleri** denir.



~ ÖZEL KARAKTERLER (HASH)

Python'da da kullandığımız gibi **#** (
HASH) karakteri kendinden sonra
gelenleri satır sonuna kadar yorum satırı
haline getirir.



~ ÖZEL KARAKTERLER (HASH)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo "Örnek Metin" #Yorum Satırı.....  
Örnek Metin  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (NOKTALI VİRGÜL)

Satırı sonlandırır. Bu sayede bize bir satırda birden fazla kod çalıştırabilme imkanı tanır.

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo Örnek Metin ; ls  
Örnek Metin  
Desktop Documents Downloads Music Pictures Public Templates Videos  
[aid3n@localhost ~]$ █
```

A screenshot of a terminal window titled "aid3n@localhost:~". The window shows a command-line interface with the following session:
- The user types "[aid3n@localhost ~]\$ echo Örnek Metin ; ls" and presses enter.
- The terminal displays the output: "Örnek Metin" followed by a list of directory names: Desktop, Documents, Downloads, Music, Pictures, Public, Templates, and Videos.
- The user then types "[aid3n@localhost ~]\$" and presses enter again.

~ ÖZEL KARAKTERLER (NOKTA)

- Eğer tek bir nokta kullanırsak, üzerinde çalıştığımız dizini temsil eder.
- İki nokta (..) kullanırsak, üzerinde çalıştığımız dizinin bir üst dizinini ifade eder.
- Bir döngüde başta ve sonda verdığımız rakamlar arasındaki sayıları temsil eder.



~ ÖZEL KARAKTERLER (NOKTA)

```
aid3n@localhost:~/Documents
File Edit View Search Terminal Help
[aid3n@localhost Documents]$ ls .
örnek1 örnek2 örnek3
[aid3n@localhost Documents]$ ls ..
Desktop Documents Downloads Music Pictures Public Templates Videos
[aid3n@localhost Documents]$ █
```

~ ÖZEL KARAKTERLER (NOKTA)

```
File Edit View Search Terminal Help
#!/bin/bash

for i in {0..9}
do
    echo $i
done
~
```

```
aid3n@localhost:~
File Edit View Search Terminal Help
[aid3n@localhost ~]$ echo {0..9}
0 1 2 3 4 5 6 7 8 9
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (TEK TIRNAK)

Karakter tanımlaması yapmaya yarar.

‘

```
aid3n@localhost:~/Documents
File Edit View Search Terminal Help
[aid3n@localhost Documents]$ degisken='örnek'
[aid3n@localhost Documents]$ echo $degisken
örnek
[aid3n@localhost Documents]$ █
```

~ ÖZEL KARAKTERLER (VİRGÜL)

- Bash'te aritmetik operasyon yaparken işlemler arasında kullanılır.
- Verilen bir stringin ilk karakterini veya tüm karakterlerini küçük harfe çevirmek için kullanılır.

~ ÖZEL KARAKTERLER (VİRGÜL)

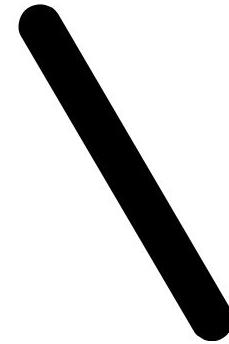
```
aid3n@localhost:~/Documents
File Edit View Search Terminal Help
[aid3n@localhost Documents]$ let "x=3+2, y=5+x"
[aid3n@localhost Documents]$ echo "x= $x and y= $y"
x= 5 and y= 10
[aid3n@localhost Documents]$ █
```

~ ÖZEL KARAKTERLER (VİRGÜL)

```
aid3n@localhost:~/Documents
File Edit View Search Terminal Help
[aid3n@localhost Documents]$ degisen="TeSt"
[aid3n@localhost Documents]$ echo ${degisen,}
teSt
[aid3n@localhost Documents]$ echo ${degisen,,}
test
[aid3n@localhost Documents]$ █
```

~ ÖZEL KARAKTERLER (Kaçış Karakteri)

Bash'te ön tanımlı gelen bazı karakterler vardır ki bunların ifade ettiği şeyler vardır. Ama bizim bazen bu karakterlerin ifade ettiği şeyleri değil de direkt kendilerini kullanmamız gerekebilir. İşte böyle durumlarda \ (kaçış karakteri) kullanırız.



~ ÖZEL KARAKTERLER (Kaçış Karakteri)

```
aid3n@localhost:~/Documents
File Edit View Search Terminal Help
[aid3n@localhost Documents]$ echo "Örnek Metin \"rockabye\""
Örnek Metin "rockabye"
[aid3n@localhost Documents]$ █
```

~ ÖZEL KARAKTERLER (Kaçış Karakteri)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ degisken=aid3n
[aid3n@localhost Desktop]$ echo \$degisken=$degisken
$degisken=aid3n
[aid3n@localhost Desktop]$ █
```

~ ÖZEL KARAKTERLER (SLASH)

- Dizinleri ayırmak için kullanılır.
- Aritmetik operasyonlarda bölme işlemi yapmak için kullanılır.





~ ÖZEL KARAKTERLER (SLASH)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Documents]$ cd /home/aid3n/Desktop/
[aid3n@localhost Desktop]$ let "v=10/2"
[aid3n@localhost Desktop]$ echo $v
5
[aid3n@localhost Desktop]$ █
```

~ ÖZEL KARAKTERLER (TERS TIRNAK)

Ters tırnak işaretinde kalan komutu çalıştırıp kendi bulunduğu yere çıktıyı yazdırır.

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ let "deger=10/2"
[aid3n@localhost Desktop]$ deger2=`echo $deger`
[aid3n@localhost Desktop]$ echo deger=$deger deger2=$deger2
deger=5 deger2=5
[aid3n@localhost Desktop]$ █
```

~ ÖZEL KARAKTERLER (İKİ NOKTA)



- Null Command olarak bilinir. Scriptin içerisinde bir yeri hiçbir şey yapmadan atlamak istersek veya bir şey yapmadan diğer komuta geçme gibi ihtiyacımız olduğu durumlarda kullanılabilir.
- İçi dolu olan bir text dosyasının içini boşaltmak için kullanılabilir.



~ ÖZEL KARAKTERLER (İKİ NOKTA)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
#!/bin/bash

for i in {1..10}
do
    if [ $i -eq 2 ] || [ $i -eq 7 ]
    then
        :
    else
        echo "$i "
    fi
done
~
~
```

~ ÖZEL KARAKTERLER (İKİ NOKTA)

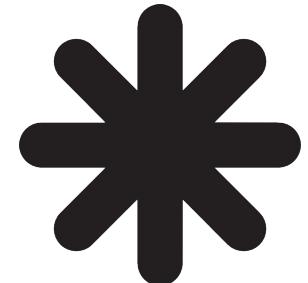
```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ ./ornek.sh
1
3
4
5
6
8
9
10
[aid3n@localhost Desktop]$
```

~ ÖZEL KARAKTERLER (İKİ NOKTA)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ cat deneme.txt  
Örnek Metin  
[aid3n@localhost ~]$ : > deneme.txt; cat deneme.txt  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (ASTERISK)

- Tam olarak “HER ŞEY” in karşılığıdır.
- Bütün karakterleri temsil eder.
- Aritmetik operasyonlarda hem çarpmacı hem de üs simgesi olarak kullanılır.



~ ÖZEL KARAKTERLER (ASTERISK)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ ls -1
test1
test10
test2
test3
test4
test5
test6
test7
test8
test9
[aid3n@localhost Desktop]$ rm test*
[aid3n@localhost Desktop]$ ls
[aid3n@localhost Desktop]$ █
```

~ ÖZEL KARAKTERLER (ASTERISK)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ let "degisken=2*3"
[aid3n@localhost Desktop]$ echo $degisken
6
[aid3n@localhost Desktop]$ let "degisken=2**3"
[aid3n@localhost Desktop]$ echo $degisken
8
[aid3n@localhost Desktop]$ █
```

~ ÖZEL KARAKTERLER (SORU İŞARETİ)

- Tek satırda if şartı yazmamızı sağlar.
- Herhangi bir karakterin yerini tutar.



~ ÖZEL KARAKTERLER (SORU İŞARETİ)

```
aid3n@localhost
File Edit View Search Terminal Help
#!/bin/bash

degisken=10

echo $(( degisken2 = degisken<20?1:0))
```

```
ai
File Edit View Search Terminal Help
[aid3n@localhost ~]$ ./ornek.sh
1
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER

- Parantez içerisindeki initialize işlemi interpreter tarafından okunmaz.
- Array tanımlarken kullanılır.
- Fonksiyon tanımlarken kullanılır.

()

~ ÖZEL KARAKTERLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ (degisen=10)  
[aid3n@localhost ~]$ echo $degisen  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ renkler=(sarı kırmızı turuncu beyaz)  
[aid3n@localhost ~]$ echo ${renkler[*]}  
sarı kırmızı turuncu beyaz  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (SÜSLÜ PARANTEZ)

- Fonksiyon tanımlarken kullanılır.
- Döngülerde, değişkenlerde kullanılır.
- Basit bir döngü oluşturmak için kullanılabilir.

{ }

~ ÖZEL KARAKTERLER (SÜSLÜ PARANTEZ)

{

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
  
fonksiyon1 ()  
{  
    echo "Deneme fonksiyonu oluştururdum."  
}
```

}

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo \"{test1,test2,test3}\\"  
"test1" "test2" "test3"  
[aid3n@localhost ~]$
```

~ ÖZEL KARAKTERLER (SÜSLÜ PARANTEZ)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ echo {1..10}
1 2 3 4 5 6 7 8 9 10
[aid3n@localhost Desktop]$
```



~ ÖZEL KARAKTERLER (AMPERSANT)

Herhangi bir komuttan sonra kullanıldığında yapılacak işi arka planda çalıştırır.

İki tane yan yana kullanıldığında “VE” anlamına gelir.

&

~ ÖZEL KARAKTERLER (AMPERSANT)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ sleep 100
^C
[aid3n@localhost Desktop]$ sleep 100 &
[1] 2319
[aid3n@localhost Desktop]$
```

~ ÖZEL KARAKTERLER (PIPE)

- Bir komutun çıkışını diğer komuta yönlendirmeye yarar.
- İki tane yan yana kullanıldığında “VEYA” anlamına gelir.



~ ÖZEL KARAKTERLER (PIPE)

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ cat innovera.txt | grep -i we
WE ENABLE
[aid3n@localhost Desktop]$
```

~ ÖZEL KARAKTERLER (PIPE)

```
aid3n@localhost:~ x
File Edit View Search Terminal Help
#!/bin/bash

degisken=1

if [ "$degisken" -gt 0 ] || [ "$degisken" -eq 0 ]
then
    echo "Birinin doğru olması halinde bu satır gözükür."
fi

if [ "$degisken" -gt 0 ] && [ "$degisken" -lt 10 ]
then
    echo "Her ikisinin de doğru olması halinde bu satır gözükür."
fi
```



~ ÖZEL KARAKTERLER (PIPE)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
Birinin doğru olması halinde bu satır gözükür.  
Her ikisinin de doğru olması halinde bu satır gözükür.  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (KISA ÇİZGİ)

- Komutlara argüman vermek için kullanılır.
- Aritmetik operasyonlarda çıkarma işlemi görevini görür.



~ ÖZEL KARAKTERLER (KISA ÇİZGİ)

```
aid3n@localhost:~
```

File Edit View Search Terminal Help

```
[aid3n@localhost ~]$ ls -l
deneme.txt
Desktop
Documents
Downloads
Music
ornek.sh
Pictures
Public
Templates
Videos
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (KISA ÇİZGİ)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ let "x=30-4"  
[aid3n@localhost ~]$ echo $x  
26  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (YÜZDE SEMBOLÜ)

Aritmetik operasyonlarda mod işlemi yapmak için kullanılır.



~ ÖZEL KARAKTERLER (YÜZDE SEMBOLÜ)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ let "slash=10/5"  
[aid3n@localhost ~]$ let "yuzde=10%5"  
[aid3n@localhost ~]$ echo slash=$slash ;echo yuzde=$yuzde  
slash=2  
yuzde=0  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (TİLDA)

Çalıştığımız kullanıcının /home dizinini temsil eder.



~ ÖZEL KARAKTERLER (TİLDA)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost share]$ ls ~  
Desktop Downloads Pictures Templates  
Documents Music Public Videos  
[aid3n@localhost share]$ cd ~  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (ÜST SEMBOLÜ)

- Virgülün tam tersi olarak verilen bir stringin ilk harfini veya tüm harflerini büyük harfe çevirmek için kullanılır.
- Verilen stringin başlangıcını temsil eder.

~ ÖZEL KARAKTERLER (ÜST SEMBOLÜ)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ degisken="DeNEmE"  
[aid3n@localhost ~]$ echo $degisken  
DeNEmE  
[aid3n@localhost ~]$ echo ${degisken^}  
DeNEmE  
[aid3n@localhost ~]$ echo ${degisken^^}  
DENEME  
[aid3n@localhost ~]$ █
```

~ ÖZEL KARAKTERLER (DOLAR SEMBOLÜ)

- Değişkeni çağrıırken kullanılır.
- Bir stringin sonunu temsil eder.

~ ÖZEL KARAKTERLER (DOLAR SEMBOLÜ)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ degisken=26  
[aid3n@localhost ~]$ echo degisken  
degisken  
[aid3n@localhost ~]$ echo $degisken  
26  
[aid3n@localhost ~]$
```

~ ÖZEL KARAKTERLER (DOLAR SEMBOLÜ)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ cat isimler.txt | grep an$  
canan  
gürhan  
hasan  
kaan  
[aid3n@localhost ~]$
```

~ DEĞİŞKENLER TANIMLAMA

- Bash'te değişken tanımlarken C, C++ dillerinde olduğu gibi değişken için bir tip belirtmek zorunda değiliz. Direk değişken ismi yazıp **=** koyduktan sonra (**arada boşluk olmamasına dikkat edelim**) istediğimiz tipte bir değişken oluşturabiliriz.
- Değişkene vereceğimiz değer, bir integer, string, char hatta bir dosya yolunu bile olabilir.

~ DEĞİŞKENLER TANIMLAMA

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ degisen1=10
[aid3n@localhost Desktop]$ degisen2:string
[aid3n@localhost Desktop]$ degisen3='char'
[aid3n@localhost Desktop]$ degisen4=/usr/share/local
[aid3n@localhost Desktop]$ █
```

~ DEĞİŞKENİ ÇAĞIRMA

- Tanımladığımız bir değişkeni ekrana bastırmak için \$ simgesini kullanırız. \$ simgelerinden sonra arada boşluk olmadan istediğimiz değişkenin ismini vererek o değişkeni çağrılabılırız.
- \$degisken_ismi

~ DEĞİŞKENİ ÇAĞIRMA

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ degisken=10
[aid3n@localhost Desktop]$ echo $degisken
10
[aid3n@localhost Desktop]$ █
```

~ DEĞİŞKENİ ÇAĞIRMA

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ degisken=26
[aid3n@localhost Desktop]$ let "sonuc=$degisken+10"
[aid3n@localhost Desktop]$ echo $sonuc
36
[aid3n@localhost Desktop]$ █
```



~ TANIMLANMIŞ DEĞİŞKENİ DEĞİŞTİRME

Bir scriptte önceden tanımladığımız bir değişkene sonradan müdahale edebiliriz. İstediğimiz kısmı script devam ederken alt satırlarda değiştirerek kullanabiliriz.

~ TANIMLANMIŞ DEĞİŞKENİ DEĞİŞTİRME

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
degisken=hey1100  
echo $degisken  
  
sayi=${degisken/hey/20}  
echo $sayi  
~  
~
```

~ TANIMLANMIŞ DEĞİŞKENİ DEĞİŞTİRME

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
hey1100  
201100  
[aid3n@localhost ~]$ █
```

~ ARİTMETİK OPERASYONLAR

- Tanımladığımız bir değişken üzerinde aritmetik operasyonlar yapmak için `let` komutunu kullanırız.
- Eğer tanımladığımız değişken bir char ile başlıyorsa kabuk yorumlayıcısı onu “0” olarak alacaktır.

~ ARİTMETİK OPERASYONLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ degisken1=10+16  
[aid3n@localhost ~]$ let "degisken2=10+16"  
[aid3n@localhost ~]$ echo $degisken1 ;echo $degisken2  
10+16  
26  
[aid3n@localhost ~]$ █
```

~ ARİTMETİK OPERASYONLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ degisken=a10  
[aid3n@localhost ~]$ let "x=$degisken+10"  
[aid3n@localhost ~]$ echo $x  
10  
[aid3n@localhost ~]$ █
```

~ FLOAT DEĞİŞKEN TANIMLAMA

- Bash ön tanımlı olarak float sayıları tanıtmaz. İşlem yapıldığında otomatik olarak sonucu integer döndürür.
- Bash scriptlerinde float sayı kullanmanın bir çok yolu vardır ben burda `awk` ve `bc` komutları üzerinden göstereceğim.

~ FLOAT DEĞİŞKEN TANIMLAMA

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ printf "%.3f" $(echo "26/5" | bc -l)  
5.200[aid3n@localhost ~]$
```

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo "26/5" | bc -l  
5.200000000000000000000000  
[aid3n@localhost ~]$
```

~ FLOAT DEĞİŞKEN TANIMLAMA

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo "scale=3; 26/5" | bc -l  
5.200  
[aid3n@localhost ~]$
```

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ awk 'BEGIN {print 26/5}'  
5.2  
[aid3n@localhost ~]$ awk 'BEGIN {print 100/3}'  
33.3333  
[aid3n@localhost ~]$
```

~ DÖNÜŞ DEĞERLERİ

\$# sembolü script çalıştırılırken scripte gönderilen argümanların toplam sayısını tutar.

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
if [ $# -lt 2 ]  
then  
    echo "2'den az argüman girdiniz."  
else  
    echo "2'den fazla argüman girdiniz."  
fi  
~
```

~ DÖNÜŞ DEĞERLERİ

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
2'den az argüman girdiniz.  
[aid3n@localhost ~]$ ./ornek.sh selam  
2'den az argüman girdiniz.  
[aid3n@localhost ~]$ ./ornek.sh selam deneme tahtası  
2'den fazla argüman girdiniz.  
[aid3n@localhost ~]$ █
```

~ DÖNÜŞ DEĞERLERİ

Kabukta çalışan her bir kod doğru da çalışsa hata da verse bir değer döndürür. **\$?** sembolü de dönen bu değer ne ise onu tutar.

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ asdfgh  
bash: asdfgh: command not found...  
[aid3n@localhost ~]$ echo $?  
127  
[aid3n@localhost ~]$ █
```



~ ŞARTLI İFADELER (if)

Her programlama dilinde olduğu gibi kabuk scriptlerinde de “if” ile şartlı durum kontrolü yapabiliriz.



~ ŞARTLI İFADELER (if)

```
if [ condition ]
then
    commands...
else
    commands...
fi
```



~ ŞARTLI İFADELER (if)

if [condition]

then

commands...

elif [condition]

then

commands...

fi



~ ŞARTLI İFADELER (if)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
if [ $# -lt 2 ]  
then  
    echo "2'den az argüman girdiniz."  
else  
    echo "2'den fazla argüman girdiniz."  
fi  
~
```

~ ŞARTLI İFADELER (if)

```
aid3n@localhost:~$ ./ornek.sh
2'den az argüman girdiniz.
[aid3n@localhost ~]$ ./ornek.sh selam
2'den az argüman girdiniz.
[aid3n@localhost ~]$ ./ornek.sh selam deneme tahtası
2'den fazla argüman girdiniz.
[aid3n@localhost ~]$ █
```

~ ŞARTLI İFADELER (if)

```
aid3n@localhost:~
```

```
File Edit View Search Terminal Help
#!/bin/bash

num=4

if [ "$num" -gt 0 ]
then
    if [ "$num" -lt 5 ]
    then
        if [ "$num" -gt 3 ]
        then
            echo "num>0 num<5 num>3"
        fi
    fi
elif [ "$num" -eq 0 ]
then
    echo "Girdiğin sayı 0"
else
    echo "Hiçbir fikrim yok."
fi
~"ornek.sh" 19L, 230C written
```

~ ŞARTLI İFADELER (if)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
num>0 num<5 num>3  
[aid3n@localhost ~]$ █
```

~ ŞARTLI İFADELER (if'te seçenekler)

- **-f** → Verilen dosya sıradan bir dosya ise true döner.
- **-s** → Verilen dosyanın boyutu 0'dan büyükse true döner.
- **-r** → Kullanıcının verilen dosyayı okuma izni varsa true döner.

~ ŞARTLI İFADELER (if'te seçenekler)

- **-x** → Kullanıcının verilen dosyayı çalışma izni varsa true döner.
- **-w** → Kullanıcının verilen dosyaya yazma izni varsa true döner.
- **!** → Verilen şartı tam tersine çevirir.

~ ŞARTLI İFADELER (if'te seçenekler)

- **-e** → Verilen dosya var ise true döner.
- **-z** → Bildiklerimizin tersine bir mantıkla çalışır. Argüman verilmemişse true döner.
- **\$#** → Verilen argüman sayısını tutar.

~ ŞARTLI İFADELER (if'te seçenekler)

- **-lt** → İngilizce “lesser than” in kısaltmasıdır < işaretini yerine geçer.
- **-gt** → İngilizce “greater than” in kısaltmasıdır > işaretini yerine geçer.

~ ŞARTLI İFADELER (if'te seçenekler)

- **-eq** → İngilizce “equal”ın kısaltmasıdır. == yerine geçer. Eşitlik kontrolü yapmamızı sağlar. (Yalnızca integer kontrol edilir.)
- **-ne** → İngilizce “not equal”ın kısaltmasıdır. != yerine geçer -eq’ın tam tersidir. (Yalnızca integer kontrol edilir.)

~ ŞARTLI İFADELER (if'te seçenekler)

- **-le** → İngilizce “lesser or equal”ın kısaltmasıdır. <= in yerine geçer.
- **-ge** → İngilizce “greater or equal”ın kısaltmasıdır. >= in yerine geçer.

~ SABİT DEĞİŞKENLER

\$\$ sembolü çalışan işlemin pid (process id) numarasını verir.

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $$  
1987  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$BASH_VERSINFO()

- Kullandığımız kabuğun bilgilerini tutan değişkendir.
- Bilgileri array olarak tutar

~ SABİT DEĞİŞKENLER

- Array olarak elemanları tek tek yazdırırsak elemanlar sırasıyla şunlardır:
 - 1) Majör Versiyon
 - 2) Minör Versiyon
 - 3) Patch Level
 - 4) Yapı Numarası
 - 5) Release Durumu
 - 6) Sistemin Mimari Yapısı

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
for i in {0..5}  
do  
    echo ${BASH_VERSINFO[$i]}  
done
```

```
aid3n@localhost:~$ ./ornek.sh  
4  
3  
43  
1  
release  
x86_64-redhat-linux-gnu  
[aid3n@localhost ~]$ █
```



~ SABİT DEĞİŞKENLER

\$PATH

İşletim sisteminizin komut satırından veya Terminal penceresinden gerekli çalıştırılabilir dosyaların yerini belirlemek için kullandığı sistem değişkenidir.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $PATH  
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$UID

- Açılımı User-id'dir.
- Aktif kullanıcının sistemde tanımlanmış id numarasını tutar.
- \$UID değişkeni her sistemde farklı değerler gösterebilir ama Linux sistemlerdeki en yetkili kullanıcı olan root kullanıcısının \$UID değişkeni her zaman 0'dır.

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $UID  
1000  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$EUID

\$UID'ye benzer olarak bu da kullanıcı numarası tutar. Ama bu değişkenin tuttuğu kullanıcı numarası sistemde yetkili olan kullanıcının id'si (numarası) dır.

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $EUID  
1000  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$FUNCNAME

- Script yazarken tanımladığımız fonksiyonların isimlerini tutar.
- Hangi fonksiyonun ismini öğrenmek istiyorsak o fonksiyonun içinde kullanmamız gereklidir.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
fonksiyon_1 ()  
{  
    echo "Bu fonksiyonun adı: \"\$FUNCNAME\""  
}  
  
fonksiyon_1  
  
echo "Fonksiyonun adı: \$FUNCNAME"  
~
```

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
Bu fonksiyonun adı: "fonksiyon_1"  
Fonksiyonun adı:  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$GROUPS

- Aktif olan kullanıcının ait olduğu grubun numarasını tutar.
- Bu değişken bir dizidir. Farklı kullanıcıların grup numaralarına da bu şekilde erişebiliriz.

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $GROUPS  
1000  
[aid3n@localhost ~]$ █
```



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ echo ${GROUPS[*]}
1000 10
[aid3n@localhost Desktop]$
```



~ SABİT DEĞİŞKENLER

\$HOME

Aktif kullanıcının /home dizininin yolunu tutar.

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $HOME  
/home/aid3n  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$HOSTNAME

Adından da anlaşılacağı üzere bilgisayarın ismini (HOSTNAME) tutan değişkendir.

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $HOSTNAME  
localhost.localdomain  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$HOSTTYPE

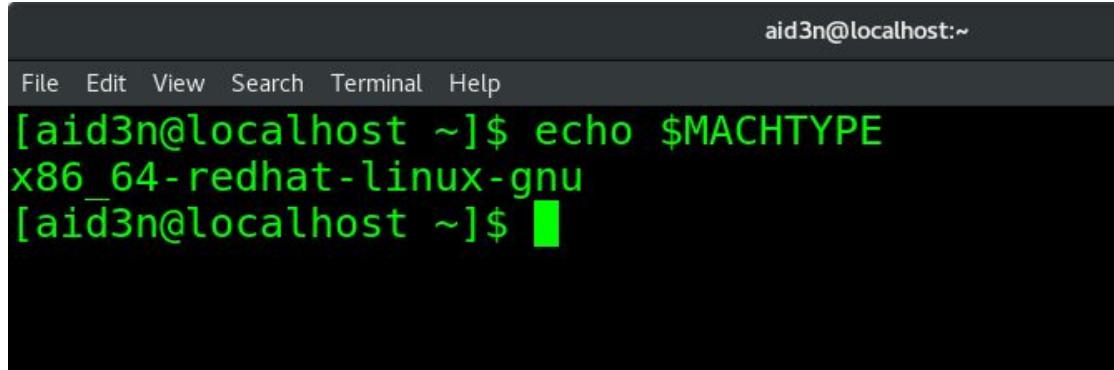
Bu değişken bilgisayarınızın sistem donanımı hakkında bilgi tutar.

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo $HOSTTYPE  
x86_64  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

\$MACHTYPE

\$HOSTTYPE değişkeninden daha ayrıntılı olarak sistem bilgilerini tutar.

A screenshot of a terminal window showing the output of the command 'echo \$MACHTYPE'. The terminal has a dark theme with white text. The prompt is 'aid3n@localhost:~'. The command entered is '[aid3n@localhost ~]\$ echo \$MACHTYPE'. The output is 'x86_64-redhat-linux-gnu'. The cursor is at the end of the line '[aid3n@localhost ~]\$'.



~ SABİT DEĞİŞKENLER

\$LINENO

Script'te yazıldığı satırın satır numarasını tutar.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
1 #!/bin/bash  
2  
3  
4  
5  
6  
7 echo "Bu yazı scriptimde $LINENO. satırdadır."
```

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
Bu yazı scriptimde 7. satırdadır.  
[aid3n@localhost ~]$
```

~ SABİT DEĞİŞKENLER

\$PWD

- Üzerinde bulunulan dizinin yolunu tutar.
- `pwd` terminal komutuyla aynı işlevi görür.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ echo $PWD
/home/aid3n/Desktop
[aid3n@localhost Desktop]$ pwd
/home/aid3n/Desktop
[aid3n@localhost Desktop]$ █
```

~ SABİT DEĞİŞKENLER

\$OLDPWD

Bir önceki dizinin yolunu verir.

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ echo $OLDPWD
/home/aid3n
[aid3n@localhost Desktop]$ █
```

~ SABİT DEĞİŞKENLER

\$OSTYPE

İşletim sistemi bilgisini tutar.

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ echo $OSTYPE
linux-gnu
[aid3n@localhost Desktop]$ █
```

~ SABİT DEĞİŞKENLER

IFS (INTERNATIONAL FIELD SEPERATOR)

- Shell tarafından kelimelerin nasıl birbirinden ayırt edileceğini gösterir.
- Ön tanımlı değeri boşluktur.

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
renkler1="sarı:::::kırmızı:::::beyaz"  
renkler2="sarı:kırmızı::beyaz"  
  
echo "IFS ayarlamadan önce:"  
echo $renkler1  
echo  
  
IFS=:  
  
echo "IFS ayarladıkten sonra:"  
echo $renkler1  
echo $renkler2
```

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
IFS ayarlamadan önce:  
sarı:::::kırmızı::::::beyaz  
  
IFS ayarladıkten sonra:  
sarı      kırmızı      beyaz  
sarı  kırmızı  beyaz  
[aid3n@localhost ~]$ █
```



~ SABİT DEĞİŞKENLER

\$SECONDS

Scriptin çalışmasından itibaren geçen saniyeyi tutar.

~ SABİT DEĞİŞKENLER

\$RANDOM

- 0 ile 32767 arasında rastgele bir sayı tutar.
- Adından da anlaşılacağı üzere her çağrılığında farklı bir sayı tutar.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
let "sayi1=$RANDOM%10 + 1"  
let "sayi2=$RANDOM%10 + 1"  
  
echo "Oluşturulan rastgele sayılar:"  
echo "$sayi1"  
echo "$sayi2"  
~
```

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~$ ./ornek.sh
Oluşturulan rastgele sayılar:
2
5
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

`sleep`

Bir fonksiyondur ve verilen değer kadar süre boyunca scripti uyutur, çalışmaz.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
sleep 12  
echo "Bu script $SECONDS saniye boyunca çalıştı."  
~
```

~ SABİT DEĞİŞKENLER

```
aid3n@localhost: ~]$ ./ornek.sh
Bu script 12 saniye boyunca çalıştı.
[aid3n@localhost ~]$ █
```



~ SABİT DEĞİŞKENLER

read

Kullanıcıdan değer almak için kullanılır.



~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
echo "En sevdığınız renk nedir?"  
read renk  
  
echo "En sevdığınız renk:"  
echo "$renk"
```

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
En sevdiğiniz renk nedir?  
yeşil  
En sevdiğiniz renk:  
yeşil  
[aid3n@localhost ~]$ █
```

~ SABİT DEĞİŞKENLER

declare

Tanımlayacağımız değişkene bir tip tanımlamak istediğimizde veya sadece okunabilir, sadece çalıştırılabilir gibi özellikler eklemek istediğimizde kullanılır.

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
declare -r degisken1=5  
echo "Okunabilir değişken=$degisken1"  
  
declare -i degisken2=10  
echo "Tamsayı değişkeni=$degisken2"  
  
degisken2=yeşil  
echo $degisken2  
~
```

~ SABİT DEĞİŞKENLER

```
aid3n@localhost:~ x
File Edit View Search Terminal Help
[aid3n@localhost ~]$ ./ornek.sh
Okunabilir değişken=5
Tamsayı değişkeni=10
./ornek.sh: line 9: yeşil: syntax error: invalid arithmetic operator (e
rror token is "şil")
10
[aid3n@localhost ~]$ █
```

~ DÖNGÜLER (FOR)

- Her programlama dilinde olduğu gibi bash scriptlerinde de arka arkaya sürekli işlem yapacağımız yerlerde aynı kodları tekrar tekrar yazmak yerine bir döngüyle işimizi kolayca halledebiliriz.
- Bash'te de Python'daki for yapısına benzer bir for yapısı bulunmaktadır.



~ DÖNGÜLER (FOR)

```
for i in 1 2 3 4 5  
do  
    codes...  
done
```



~ DÖNGÜLER (FOR)

```
for (( i=0 ; i<10; i++ ))  
do  
    codes...  
done
```

~ DÖNGÜLER (FOR)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
for i in {1..10}  
do  
    echo $i  
done
```

~ DÖNGÜLER (FOR)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
for i in 1 2 3 4 5  
do  
    echo $i  
done
```



~ DÖNGÜLER (FOR)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
for (( i=0; i<10; i++ ))  
do  
    echo $i  
done
```



~ DÖNGÜLER (WHILE)

for yapısına benzer şekilde while da Python'daki yapıya benzer.



~ DÖNGÜLER (WHILE)

while [i -lt 10]

do

codes...

done



~ DÖNGÜLER (WHILE)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
i=0  
  
while [ $i -lt 10 ]  
do  
    echo $i  
    ((i++))  
done
```

~ DÖNGÜLER (UNTIL)

- until döngüsü while yapısının tam tersidir. While'da belirttiğimiz şart doğru olursa döngüye girer yoksa girmeden devam eder. Until'de olay tam tersidir. Şart yanlış olduğu sürece döngüye girer doğru olduğunda çıkar.
- Yani verilen şart sağlanana kadar gibi bir anlamı vardır.



~ DÖNGÜLER (UNTIL)

until [\$1 == “son”]

do

codes...

done



~ DÖNGÜLER (UNTIL)

```
aid3n@localhost:~ x
File Edit View Search Terminal Help
#!/bin/bash

until [ "$n" == son ]
do
    echo "Devam etmek için son dışında bir kelime girin."
    read n
    echo $n
done
```

~ DÖNGÜLER (UNTIL)

```
aid3n@localhost:~
```

```
File Edit View Search Terminal Help
```

```
[aid3n@localhost ~]$ ./ornek.sh
Devam etmek için son dışında bir kelime girin.
selam
selam
Devam etmek için son dışında bir kelime girin.
merhaba
merhaba
Devam etmek için son dışında bir kelime girin.
son
son
[aid3n@localhost ~]$ █
```

~ DÖNGÜLER (CONTINUE VE BREAK)

- Scriptimizde kullandığımız döngülerin içinde işlem yapmak istemediğimiz değişkenler olabilir. Böyle durumlarda continue yapısını kullanabiliriz.
- Belli bir yerden sonra da döngüden çıkış geri kalan komutları çalıştırırmak isteyebiliriz. Bu durumda da break yapısını kullanırız.

~ DÖNGÜLER (CONTINUE VE BREAK)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
for i in {1..10}  
do  
    if [ "$i" -eq 5 ] || [ "$i" -eq 7 ]  
    then  
        continue  
    fi  
    echo $i  
done
```



~ DÖNGÜLER (CONTINUE VE BREAK)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
1  
2  
3  
4  
6  
8  
9  
10  
[aid3n@localhost ~]$ █
```

~ DÖNGÜLER (CONTINUE VE BREAK)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
for i in {1..10}  
do  
    if [ "$i" -eq 5 ] || [ "$i" -eq 7 ]  
    then  
        break  
    fi  
    echo $i  
done
```



~ DÖNGÜLER (CONTINUE VE BREAK)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
1  
2  
3  
4  
[aid3n@localhost ~]$ █
```



~ DÖNGÜLER (CASE)

Tek tek kontrol etmek yerine elimizdeki bir değer üzerinden kontrol yapmak istersek case yapısını kullanabiliriz.



~ DÖNGÜLER (CASE)

```
case "$degisken" in
[0-9]      ) echo "sayı" ;;
esac
```

~ DÖNGÜLER (CASE)

```
aid3n@localhost:~ x
File Edit View Search Terminal Help
#!/bin/bash
echo "Bir harf veya sayı giriniz."
read girdi

case "$girdi" in
    [[:lower:]] ) echo "$girdi küçük harftir.";;
    [[:upper:]] ) echo "$girdi büyük harftir.";;
    [0-9]        ) echo "$girdi bir sayıdır.";;
    *            ) echo "$girdi bir özel karakterdir.";;
esac
~
```

~ DÖNGÜLER (CASE)

```
aid3n@localhost:~ x
File Edit View Search Terminal Help
[aid3n@localhost ~]$ ./ornek.sh
Bir harf veya sayı giriniz.
a
a küçük harftir.
[aid3n@localhost ~]$ ./ornek.sh
Bir harf veya sayı giriniz.
F
F büyük harftir.
[aid3n@localhost ~]$ ./ornek.sh
Bir harf veya sayı giriniz.
5
5 bir sayıdır.
[aid3n@localhost ~]$ ./ornek.sh
Bir harf veya sayı giriniz.
*
* bir özel karakterdir.
[aid3n@localhost ~]$ █
```

~ DÖNGÜLER (SELECT)

- Kullanıcıya seçmesi için seçenekler vermeye yarayan bir yapıdır.
- Normalde elle yapılması uzun sürecek bir seçme işlemini bir kaç satırda yapmamızı sağlayan efektif bir yapıdır.



~ DÖNGÜLER (SELECT)

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
PS3='Bir renk seçin:'  
  
select renk in "sarı" "kırmızı" "beyaz" "siyah" "yeşil"  
do  
    echo "Seçtiğin renk: $renk"  
    break  
done
```



~ DÖNGÜLER (SELECT)

```
aid3n@localhost:~
```

```
File Edit View Search Terminal Help
```

```
[aid3n@localhost ~]$ ./ornek.sh
```

```
1) sarı
2) kırmızı
3) beyaz
4) siyah
5) yeşil
```

```
Bir renk seçin:5
```

```
Seçtiğin renk: yeşil
```

```
[aid3n@localhost ~]$ █
```

~ KOMUTLAR

echo

- Bash scriptlerinde en çok kullanılan komutlardan biridir.
- Ekrana yazdırma komutudur.

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ echo "Örnek Metin"  
Örnek Metin  
[aid3n@localhost ~]$
```

~ KOMUTLAR

printf

- C dilinin ekrana yazdırma komutudur.
- Gördüğümüz `echo` komutunun aynısıdır. Yani ekrana yazdırma komutudur. Farkı daha ayrıntılı çıktılar verebiliyor olmasıdır.

~ KOMUTLAR

```
vi /home/aid3n
File Edit View Search Terminal Help
#!/bin/bash

declare -r PI=3.1415926

printf "Noktadan sonra sadece iki basamak yazdır %1.2f\n" $PI
printf "Noktadan sonra sadece beş basamak yazdır %1.5f\n" $PI
printf "Verilen sayının tamamını yaz %f\n" $PI
~
```

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
Noktadan sonra sadece iki basamak yazdır 3.14  
Noktadan sonra sadece beş basamak yazdır 3.14159  
Verilen sayının tamamını yaz 3.141593  
[aid3n@localhost ~]$
```

~ KOMUTLAR

eval

- Bu komut içine yazdığımız komutların tamamını tek bir komutta birleştirir ve o komutu yorumlar. Dönüş değeri de eval değeri olarak döner.
- Eval tehlikeli argümanlar alabilir. Bu yüzden kullanımı güvenlik açısından sıkıntı oluşturabilir.

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
ornek=26 x=ornek  
  
y='$'$x  
echo $y  
echo $ornek  
  
eval y='$'$x  
echo $y
```



~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
$ornek  
26  
26  
[aid3n@localhost ~]$
```



~ KOMUTLAR

set

set komutu ile scripti çalıştırırken göndereceğimiz argümanı scriptin içinden de gönderebiliriz.

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
echo "Argüman göndermeden önce:"  
  
echo $1  
echo $2  
  
echo "Argümanları gönderdikten sonra:"  
  
set ` echo "J0hn d0e" `  
  
echo "Birinci argüman: $1"  
echo "İkinci argüman: $2"
```



~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
Argüman göndermeden önce:  
  
Argümanları gönderdikten sonra:  
Birinci argüman: J0hn  
İkinci argüman: d0e  
[aid3n@localhost ~]$
```



~ KOMUTLAR

unset

unset komutu script içerisinde daha önceden oluşturduğumuz bir değişkenin yerine NULL karakteri basarak siler.

~ KOMUTLAR

```
File Edit View Search Terminal Help
#!/bin/bash

degisken=26
echo $degisken

unset degisken
echo $degisken
~
```

```
aid3n@localhost:~
File Edit View Search Terminal Help
[aid3n@localhost ~]$ ./ornek.sh
26

[aid3n@localhost ~]$
```



~ KOMUTLAR

getopts

getopts argümanlar ile çalışacak bir script hazırlamamızı sağlar.



~ KOMUTLAR

```
aid3n@localhost:~ x
File Edit View Search Terminal Help
#!/bin/bash

while getopts :ga secenek
do
    case $secenek in
        g) g_option=1 ;;
        a) a_option=1 ;;
        *) echo "Kullanım: -ga"
    esac
done

day=`date | awk '{print $1 " " $2}'`
if [ ! -z $g_option ]
then
    echo "Gün: $day"
fi

month=`date | awk '{print $3}'`
if [ ! -z $a_option ]
then
    echo "Ay: $month"
fi

shift $((OPTIND - 1))
```

~ KOMUTLAR

```
aid3n@localhost:~
```

```
File Edit View Search Terminal Help
```

```
[aid3n@localhost ~]$ ./ornek.sh
[aid3n@localhost ~]$ ./ornek.sh -g
Gün: Thu May
[aid3n@localhost ~]$ ./ornek.sh -a
Ay: 18
[aid3n@localhost ~]$ ./ornek.sh -ga
Gün: Thu May
Ay: 18
[aid3n@localhost ~]$ ./ornek.sh -ag
Gün: Thu May
Ay: 18
[aid3n@localhost ~]$ ./ornek.sh -f
Kullanım: -ga
[aid3n@localhost ~]$
```

~ KOMUTLAR

shopt

- Shell Options'ın kısaltmasıdır.
- Shell özelliklerini açıp kapatmamızı sağlayan komuttur.

~ KOMUTLAR

```
File Edit View Search Terminal Help
aid3n@localhost:~
```

```
globasciiranges off
gnu_errfmt off
histappend on
histredit off
histverify off
hostcomplete off
huponexit off
interactive_comments on
lastpipe off
lithist off
login_shell off
mailwarn off
no_empty_cmd_completion off
nocaseglob off
nocasematch off
nullglob off
progcomp on
promptvars on
restricted_shell off
shift_verbose off
sourcepath on
xpg_echo off
[aid3n@localhost ~]$
```

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ shopt -s cdspell  
[aid3n@localhost ~]$ cd Docunents  
Documents  
[aid3n@localhost Documents]$ shopt -u cdspell  
[aid3n@localhost Documents]$ cd ..  
[aid3n@localhost ~]$ cd Docunents  
bash: cd: Docunents: No such file or directory  
[aid3n@localhost ~]$
```



~ KOMUTLAR

type

Komutun tipi hakkında bilgi gösterir.

~ KOMUTLAR

```
aid3n@localhost:~
```

File Edit View Search Terminal Help

```
[aid3n@localhost ~]$ type tar  
tar is /usr/bin/tar  
[aid3n@localhost ~]$ type cd  
cd is a shell builtin  
[aid3n@localhost ~]$ type ls  
ls is aliased to `ls --color=auto'  
[aid3n@localhost ~]$ type mv  
mv is /usr/bin/mv  
[aid3n@localhost ~]$
```



~ KOMUTLAR

jobs

O anda arka planda yapılan işleri gösterir.

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ sleep 1000 &  
[1] 7172  
[aid3n@localhost ~]$ sleep 1001 &  
[2] 7179  
[aid3n@localhost ~]$ jobs  
[1]- Running sleep 1000 &  
[2]+ Running sleep 1001 &  
[aid3n@localhost ~]$
```

~ KOMUTLAR

disown

- Arka planda yapılan işleri bitirmek için kullanılır.
- Direk `disown` yazarak en son çalışan işlem kapatılır.
- -a parametresiyle arka planda çalışan tüm uygulamalar kapatılır.

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ sleep 1000 &  
[1] 7339  
[aid3n@localhost ~]$ sleep 1001 &  
[2] 7346  
[aid3n@localhost ~]$ jobs  
[1]- Running sleep 1000 &  
[2]+ Running sleep 1001 &  
[aid3n@localhost ~]$ disown  
[aid3n@localhost ~]$ jobs  
[1]+ Running sleep 1000 &  
[aid3n@localhost ~]$
```

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ sleep 1000 &  
[1] 7471  
[aid3n@localhost ~]$ sleep 1001 &  
[2] 7478  
[aid3n@localhost ~]$ sleep 1002 &  
[3] 7485  
[aid3n@localhost ~]$ jobs  
[1]  Running                  sleep 1000 &  
[2]- Running                  sleep 1001 &  
[3]+ Running                  sleep 1002 &  
[aid3n@localhost ~]$ disown -a  
[aid3n@localhost ~]$ jobs  
[aid3n@localhost ~]$
```

~ KOMUTLAR

fg

- Foreground'un kısaltmasıdır.
- Adından anlaşılacağı üzere bu komutla arka planda çalışan bir uygulamayı ön plana alabiliriz.
- Bu komut argüman olarak bir id bekler. Bu yüzden jobs komutuyla arka planda çalışan uygulamalara baktıktan sonra argüman olarak çıkan çıktıdaki numaralardan birini göndermemiz gerekiyor.



~ KOMUTLAR

```
aid3n@localhost:~
```

File Edit View Search Terminal Help

```
[aid3n@localhost ~]$ sleep 1000&
[1] 7541
[aid3n@localhost ~]$ sleep 1001&
[2] 7548
[aid3n@localhost ~]$ sleep 1002&
[3] 7555
[aid3n@localhost ~]$ sleep 1003&
[4] 7562
[aid3n@localhost ~]$ jobs
[1]  Running                  sleep 1000 &
[2]  Running                  sleep 1001 &
[3]- Running                  sleep 1002 &
[4]+ Running                  sleep 1003 &
[aid3n@localhost ~]$ fg 2
sleep 1001
```

~ KOMUTLAR

xkill

- Bu komutu terminalde çalıştırduğumuz zaman imleç siyah bir çarpı işaretine dönüşür ve hangi pencerenin üzerine basarsak o processi sonlandırır.
- Root olarak bu komut verildiğinde hangi pencereye basarsak basalım sorgusuz sualsız direkt kapatacağı için pek kullanışlı değildir.

~ KOMUTLAR

kill

- Kill komutu verilen pid numarasındaki işlem ne ise o işlemi sonlandırmaya yarayan komuttur.
- Yukarıda da belirttiğim gibi `kill` komutu argüman olarak bir pid numarası bekler. Bu yüzden hangi çalışan işlemi sonlandırmak istiyorsak ilk önce o işlemin pid numarasını öğrenmemiz gerek.



~ KOMUTLAR

```
aid3n@localhost:~ top - 18:15:46 up 8 min, 1 user, load average: 0.16, 0.26, 0.17
Tasks: 189 total, 3 running, 186 sleeping, 0 stopped, 0 zombie
%Cpu(s): 27.8 us, 14.4 sy, 0.0 ni, 55.2 id, 0.4 wa, 1.7 hi, 0.4 si, 0.0 st
KiB Mem : 2048572 total, 648380 free, 828176 used, 572016 buff/cache
KiB Swap: 839676 total, 839676 free, 0 used. 1015260 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
1435 aid3n    20   0 2163148 237508 85400 R 101.0 11.6 0:40.70 gnome-shell
2149 aid3n    20   0 651736 30172 23988 S 12.0  1.5 0:00.36 gnome-scre+
1913 aid3n    20   0 734956 39336 30720 S 1.3  1.9 0:01.22 gnome-term+
 700 root      20   0    4416   696   628 S 0.7  0.0 0:00.14 rngd
1695 aid3n    20   0 221424 2780 2412 S 0.7  0.1 0:00.53 VBoxClient
    7 root      20   0      0     0     0 S 0.3  0.0 0:00.21 rcu_sched
1446 aid3n    20   0 212024 29976 22308 S 0.3  1.5 0:00.09 Xwayland
1633 aid3n    20   0 537972 17176 12168 S 0.3  0.8 0:00.08 tracker-st+
    1 root      20   0 149336 10396 7400 S 0.0  0.5 0:01.38 systemd
    2 root      20   0      0     0     0 S 0.0  0.0 0:00.00 kthreadd
    3 root      20   0      0     0     0 S 0.0  0.0 0:00.01 ksoftirqd/0
    5 root      0 -20      0     0     0 S 0.0  0.0 0:00.00 kworker/0:+
    8 root      20   0      0     0     0 S 0.0  0.0 0:00.00 rcu_bh
    9 root      20   0      0     0     0 S 0.0  0.0 0:00.11 rcuos/0
   10 root      20   0      0     0     0 S 0.0  0.0 0:00.00 rcuob/0
   11 root      rt  0      0     0     0 S 0.0  0.0 0:00.00 migration/0
   12 root      0 -20      0     0     0 S 0.0  0.0 0:00.00 lru-add-dr+
```

~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ps -e | grep nautilus  
2622 ? 00:00:00 nautilus  
[aid3n@localhost ~]$ kill 2622  
[aid3n@localhost ~]$
```

~ KOMUTLAR

command

Yazdığımız script içerisinde bash'te ön tanımlı komutlardan birinin adını fonksiyon adı olarak kullanmış olabiliriz. Bu noktada fonksiyonu çağrırmamız gerekiğinde ön tanımlı komutu çalıştırmak mı istiyoruz yoksa fonksiyonu mu çağrırmak istiyoruz bu konuda bir karışıklık çıkabilir.

~ KOMUTLAR

command

İşte bu noktada imdadımıza `command` komutu yetişir. Direk fonksiyon adıyla çağrırsak fonksiyonumuz çağrılmış olur.

`command <komut>` şeklinde yazarsak ön tanımlı komutu çalıştırılmış oluruz.



~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
#!/bin/bash  
  
ls ()  
{  
    echo "Bu fonksiyon normalde listeleme komutudur."  
    echo;echo  
}  
  
ls  
  
command ls
```



~ KOMUTLAR

```
aid3n@localhost:~  
File Edit View Search Terminal Help  
[aid3n@localhost ~]$ ./ornek.sh  
Bu fonksiyon normalde listeleme komutudur.  
  
Desktop Downloads ornek.sh Public Videos  
Documents Music Pictures Templates  
[aid3n@localhost ~]$
```



~ DÜZENLİ İFADELER

Soru İşareti (?)

Herhangi bir karakterin yerini tutmaya yarayan karakterdir.



~ DÜZENLİ İFADELER

```
aid3n@localhost:~/Desktop
File Edit View Search Terminal Help
[aid3n@localhost Desktop]$ ls -l dene?e.txt
-rw-rw-r--. 1 aid3n aid3n 7 May 19 15:15 deneme.txt
[aid3n@localhost Desktop]$
```



~ DÜZENLİ İFADELER

Üs İşareti (^)

Herhangi bir karakterin yerini tutmaya yarayan karakterdir.



~ DÜZENLİ İFADELER



~ DÜZENLİ İFADELER