

# BİLGİ İŞLEM

MSGSU FİZİK BÖLÜMÜ

## DERS 5

Yrd Doç Dr. Ferhat ÖZOK  
MSGSU FİZİK BÖLÜMÜ

# Kabuk Programlamaya Giriş

Her kabuğun kendine özgü programlama dili yapısı vardır. Bash kabuğu güçlü programlama özellikleriyle karmaşık programların rahatca yazılmasına izin verir. Mantıksal operatörler, döngüler, değişkenler ve modern programlama dillerinde bulunan pek çok özellik bash kabuğunda da vardır ve işleyiş tarzlarında hemen hemen aynıdır. Genellikle, bir program oluşturacak olan komutlar bir dosyaya yazılırlar ve ardından bu dosya çalıştırılır. Herhangi bir editor yardımıyla yazılan program, daha sonra kabuk altında çalıştırılır. Bir kabuk programı başka kabuk programlarında çalıştırabilir. Bundan sonraki derslerimizde kabuk programlama konusunda duracağız.

Kabuk Programları , bir veya birden fazla linux komutunu tutan dosyalardır. Bu dosya Hazırlandıktan sonra çalıştırılabilir. Bir kabuk programı, istenen kullanıcılar için çalıştırılabilir hale getirilebilir

\$ chmod +x komut-ismi

Kabuk programlar yazarken dosyanın işlevini ve her satırdaki komutun veya komut kumesi açıklamak için açıklama satırları kullanmak işe yarar. Bir açıklama eklemek için satır başına (veya boş satıra) # isareti eklenir ve ardından istenilen cümle girilir. # isaretinden sonraki tüm satır kabuk tarafından göz ardı edilir.

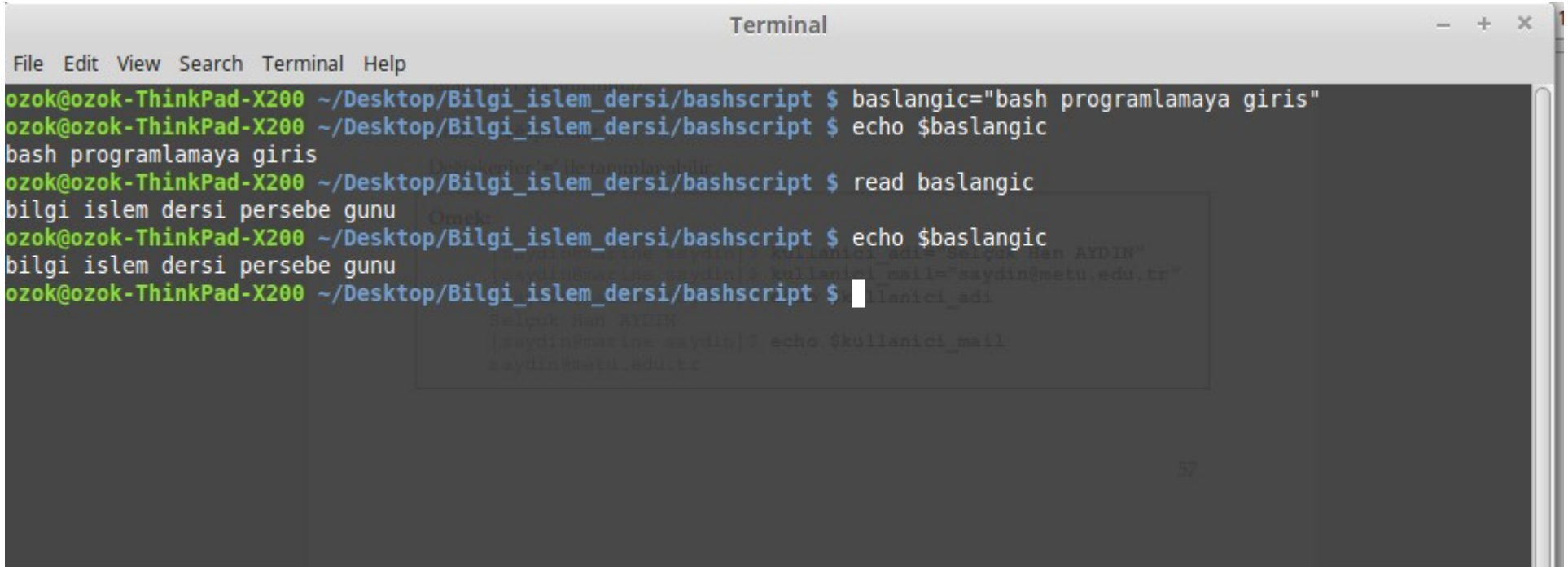
```
Terminal
File Edit View Search Terminal Help
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ cat calistir
echo -n "Tarih : "
date
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ chmod +x calistir
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ls
calistir
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./calistir
Tarih : Wed Oct 12 23:45:21 EEST 2016
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $
```

## Değişkenlerin Kullanımı

Bir değişkene değer atandığı anda sistem tarafından tanınır. Değişkenler alfabetik veya nümerik karakterlerden oluşabilirler fakat bir değişken sayısal bir değer ile başlayamaz. Bunların dışında değişken isminin içinde " " karakteri de bulunabilir. Bir değişkene değer ataması "=" isareti yardımıyla yapılır.

### Değer Okuma

Değişkenlere read komutu ile çalışma sırasında dışarıdan değer atanabilir



```
Terminal
File Edit View Search Terminal Help
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ baslangic="bash programlamaya giris"
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $baslangic
bash programlamaya giris
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ read baslangic
bilgi islem dersi persebe gunu
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $baslangic
bilgi islem dersi persebe gunu
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ |
bilgi islem dersi persebe gunu
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $kullanici_mail
bilgi islem dersi persebe gunu
bilgi islem dersi persebe gunu
```

# Aritmetik İşlemler

bash kabuğunda matematiksel işlemlere büyük sınırlamalar getirilmiştir. Tamsayı değişkenler dışında matematiksel değişken kullanmak için bu işlemler için geliştirilmiş ve kolaylıklar sağlayan awk veya bc gibi komutlar kullanılabilir. Aritmetik işlemler için eval komutunu veya bash kabuğu altında yerleşik (builtin) komut olan let komutunu kullanabilirsiniz.

```
let "degisken=aritmetik islem"
```

Aritmetik değişken tanımlamanın diğer bir yolu da typeset komutudur. Herhangi bir tanım yapmadan (( )) ile de aritmetik işlemler gerçekleştirilebilir.

```
Terminal
File Edit View Search Terminal Help
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ let "toplam=5+8"
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $toplam
13
[seydin@marine seydin]$ echo "Girdiginiz Kullanici"
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ typeset -i toplam2
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ a=5;b=8
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ toplam2=a+b
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $toplam2
13
Armetik islemler için let, typeset komulları kullanılabilir.
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ (( c= $a - $b ))
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $c
-3
[seydin@marine seydin]$ a=5
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ (( c= $a + $b ))
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ echo $c
13
[seydin@marine seydin]$ typeset -i sonuc
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $
[seydin@marine seydin]$ echo $sonuc
[seydin@marine seydin]$ let "sonuc=$a+$b"
[seydin@marine seydin]$ echo $sonuc
[seydin@marine seydin]$ let "sonuc=$a*$b"
[seydin@marine seydin]$ echo $sonuc
13
Herhangi bir tanım yapmadan (( )) ile de armetik islemler gerçekleştirilebilir.
```

Bash kabuk programında \$? ile en son çalışan komutun sonucu öğrenilebilir. Bu özellikten yararlanarak komutların sonuçları ile ilgili bilgi edinilebilir. Eğer komutlar başarılı bir şekilde sonuçlanmışsa 0 sonucunu, diğer durumlarda 0'dan farklı bir değer üretirler.

```
ozok@ozok-ThinkPad-X200 ~ $ test 5 -gt 3
ozok@ozok-ThinkPad-X200 ~ $ echo $?
0
ozok@ozok-ThinkPad-X200 ~ $ test 3 -gt 3
ozok@ozok-ThinkPad-X200 ~ $ echo $?
1
ozok@ozok-ThinkPad-X200 ~ $
```

# If-Else Kalıbı ve Kontrol İşlemleri

Aritmetik karşılaştırmalarda, karakter dizileri (string) ve dosya karşılaştırmalarında aşağıdaki seçenekler kullanılabilir:

Aritmetik		Karakter dizileri		Dosya	
<b>eq</b>	eşit	<b>-z</b>	Boş	<b>-f</b>	Dosya var
<b>gt</b>	büyük	<b>-n</b>	Tanımlı	<b>-s</b>	Dosya boş değil
<b>lt</b>	küçük	<b>=</b>	Eşit	<b>-r</b>	Dosya okunabilir
<b>ge</b>	büyük eşit	<b>!=</b>	Farklı	<b>-w</b>	Dosya yazılabilir
<b>le</b>	küçük eşit			<b>-x</b>	Dosya çalıştırılabilir
				<b>-h</b>	Sembolik bağlantı
				<b>-c</b>	Karakter dosyası
				<b>-b</b>	Blok dosyası



Hemen her programlama dilinde olan if kalıbı bir Linux komutunun çalışmasını kontrol ( test ) eder. if komutu yerlesik bir komuttur ve her if , bir fi komutuyla bitmelidir. if komutunun ardından gelen Linux komutu çalıştırılır ve komutun çıkış durumu (exit status) gözönüne alı narak ardından gelen then deyimiyle birlikte devamı işletilir. Genellikle komutun iki türlü çıkış durumu olacağından else komutunun ardından gelen komut zinciri, diğer çıkış durumunda çalıştırılır

İki mantıksal sonuç && (ve), || (ya da) ile karşılaştırılabilir

```
if linux komutu
then
komut1
komut2
...
else
komut1
komut2
...
fi
```

```
Terminal
File Edit View Search Terminal Help
GNU nano 2.2.6 File: ifkalibi
echo "bir sayi girin"
read sayi
if [ $sayi -gt 10 ]; then
  echo "Girdiginiz sayi 10 dan buyuk";
elif [ $sayi -eq 10 ]; then
  echo "Sayi 10"
else
  echo "Sayi 10 dan kucuk"
fi
[ Read 10 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

```
Terminal
File Edit View Search Terminal Help
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ pico ifkalibi
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./ifkalibi
bir sayi girin
3
Sayi 10 dan kucuk
```

# Case Döngüsü

Birkaç alternatif arasında seçim yapılması gerekiyorsa case kalıbı kullanılır. Komutun genel kullanımını şu biçimdedir:

```
case aranacak_kelime in  
seçenek1)  
komutlar  
;;  
seçenek2)  
komutlar  
;;  
...  
*)  
komutlar  
;;  
esac
```

Terminal



File Edit View Search Terminal Help

GNU nano 2.2.6

File: casedongusu

```
#Dosya işlemleri
echo "Dosya işlem menüsü"
echo "1-Kopyalama"
echo "2-Taşıma"
echo "3-Silme"
echo "Seçek giriniz"
read secenek
case $secenek in
1)
cp dosya /tmp
echo "Dosya kopyalandı"
;;
2)
mv dosya /tmp
echo "Dosya taşındı"
;;
3)
rm dosya
echo "Dosya silindi"
;;
*)
```

[ Read 23 lines ]

^G Get Help

^O WriteOut

^R Read File

^Y Prev Page

^K Cut Text

^C Cur Pos

^X Exit

^J Justify

^W Where Is

^V Next Page

^U UnCut Text

^T To Spell

```
Terminal
File Edit View Search Terminal Help
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ls
calistir casedongusu dosya ifkalibi komutlar
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./casedongusu
"Dosya işlem menüsü"
"1-Kopyalama"
"2-Taşıma"
"3-Silme"
"Seçek giriniz"
5
"Yanlış bilgi girişi"
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./casedongusu
"Dosya işlem menüsü"
"1-Kopyalama"
"2-Taşıma"
"3-Silme"
"Seçek giriniz"
1
"Dosya kopyalandı"
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ls /tmp/
dosya icedteaplugin-root-VJ52vi MathLink
hsperfdata_mdm kde-ozok mintUpdate
hsperfdata_ozok ksocket-ozok OSL_PIPE_1000_SingleOfficeIPC_8dc3b8df23cb7a490fd786e34aba479
hsperfdata_root lu212847c8ji4.tmp pulse-PKdhtXMmr18n
icedteaplugin-mdm-fpBgvf m000002191301 ssh-Is100uHqx1D7
icedteaplugin-root-NrVdA9 m00000285851
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $
```

# Döngüler

Bash kabuk programında aşağıdaki döngü türleri kullanılabilir:

while kontrol ifadesi

do

komutlar

Done

ya da;

for değişken in değerler

do

komutlar

done

```
Terminal
File Edit View Search Terminal Help
GNU nano 2.2.6 File: dowhile
toplam=0
i=1
while [ $i -le 10 ]
do
  (( toplam = $toplam +$i))
  (( i = $i + 1 ))
done
echo $toplam

[ Read 9 lines ]
^G Get Help      ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit          ^J Justify      ^W Where Is    ^V Next Page    ^U UnCut Text   ^T To Spell
```

```
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./dowhile
55
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $
```

Terminal



File Edit View Search Terminal Help

GNU nano 2.2.6

File: fortest

```
#!/bin/bash
for i in {1..5}
do
    echo "Welcome $i times"
done

# Bash version
# echo "Bash version ${BASH_VERSION}..."
# for i in {0..10..2}
# do
#     echo "Welcome $i times"
# done

# v4.0+ has inbuilt support for setting up a step value using
# for i in 1 2 3 4 5
# do
#     echo "Welcome $i times"
# done

# Bash version 3(BASH VERSION) ...
# in {0..10..2}

# echo "Welcome $i times"
```

[ Read 17 lines ]

^G Get Help  
^X Exit

^O WriteOut  
^J Justify

^R Read File  
^W Where Is

^Y Prev Page  
^V Next Page

^K Cut Text  
^U UnCut Text

^C Cur Pos  
^T To Spell

Top 10 Open Source Web-Based Project Management Software

Top 5 Email Client For Linux, Mac OS X, and Windows Users

The Novice Guide To Buying A Linux Laptop

How to run sudo command without a password on a...

Cheap \$199 Linux Computer from Shuttle

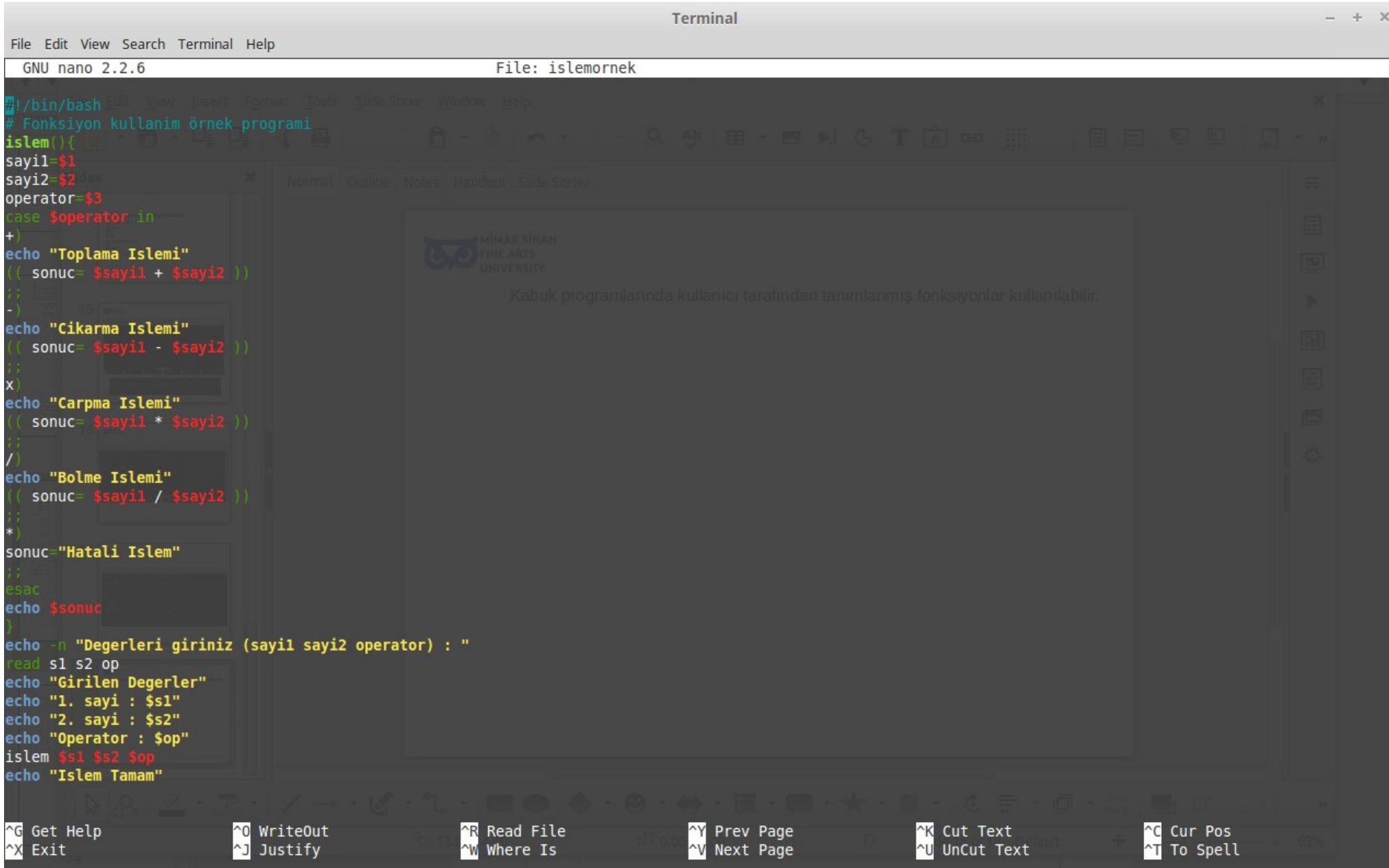


## Terminal

File Edit View Search Terminal Help

```
^[[Aozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ pico forttest
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./forttest
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
Welcome 5 times
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ pico forttest
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./forttest
Bash version 4.3.11(1)-release...
Welcome 0 times
Welcome 2 times
Welcome 4 times
Welcome 6 times
Welcome 8 times
Welcome 10 times
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $
```

```
Terminal
File Edit View Search Terminal Help
GNU nano 2.2.6 File: islemornek
#!/bin/bash
# Fonksiyon kullanım örnek programı
islem(){
sayi1=$1
sayi2=$2
operator=$3
case $operator in
+)
echo "Toplama Islemi"
(( sonuc= $sayi1 + $sayi2 ))
;;
-)
echo "Çikarma Islemi"
(( sonuc= $sayi1 - $sayi2 ))
;;
x)
echo "Çarpma Islemi"
(( sonuc= $sayi1 * $sayi2 ))
;;
/)
echo "Bölme Islemi"
(( sonuc= $sayi1 / $sayi2 ))
;;
*)
sonuc="Hatali Islem"
;;
esac
echo $sonuc
}
echo -n "Değerleri giriniz (sayi1 sayi2 operator) : "
read s1 s2 op
echo "Girilen Değerler"
echo "1. sayi : $s1"
echo "2. sayi : $s2"
echo "Operator : $op"
islem $s1 $s2 $op
echo "Islem Tamam"
```



^G Get Help  
^X Exit  
^O WriteOut  
^J Justify  
^R Read File  
^W Where Is  
^Y Prev Page  
^V Next Page  
^K Cut Text  
^U UnCut Text  
^C Cur Pos  
^T To Spell

## Terminal

File Edit View Search Terminal Help

```
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./islemornek
```

```
Degerleri giriniz (sayi1 sayi2 operator) : 9 3 /
```

```
Girilen Degerler
```

```
1. sayi : 9
```

```
2. sayi : 3
```

```
Operator : /
```

```
Bolme Islemi
```

```
3
```

```
Islem Tamam
```

```
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./islemornek
```

```
Degerleri giriniz (sayi1 sayi2 operator) : 6 4 -
```

```
Girilen Degerler
```

```
1. sayi : 6
```

```
2. sayi : 4
```

```
Operator : -
```

```
Cikarma Islemi
```

```
2
```

```
Islem Tamam
```

```
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./islemornek
```

```
Degerleri giriniz (sayi1 sayi2 operator) : 8 4 x
```

```
Girilen Degerler
```

```
1. sayi : 8
```

```
2. sayi : 4
```

```
Operator : x
```

```
Carpma Islemi
```

```
32
```

```
Islem Tamam
```

```
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $ ./islemornek
```

```
Degerleri giriniz (sayi1 sayi2 operator) : 4 7 :
```

```
Girilen Degerler
```

```
1. sayi : 4
```

```
2. sayi : 7
```

```
Operator : :
```

```
Hatali Islem
```

```
Islem Tamam
```

```
ozok@ozok-ThinkPad-X200 ~/Desktop/Bilgi_islem_dersi/bashscript $
```