Kabuk Programlama (Bash) - 4

Erkan Esmer

Nisan, 2016

Merhaba,
Hatırlarsınız if komutu ile bir kontrol gerçekleştirmiştik ve bu kontroldeki sonuca göre de uygulamada başka bir satıra zıplayarak çalışmaya oradan devam ediyorduk. Aslında bir nevi döngü oluşturmuş olup gerektiği zamanda uygulamayı istediğimiz yerden devam ettiriyorduk.

Özgür Yazılım Günleri-2013'ün etkinliklerine baktıysanız görmüşsünüzdür. "GoTo Zararlıdır" isimli bir konferans vardı. Özellikle bu tür konuların gördüğüm kadarıyla önemli bir ders notu ve tartışma noktası aslında GoTo. Konumuz tabii ki bu değil yalnız bizim jump komutumuz GoTo gibi. Gelin biz GoTo gibi kullandığımız jump komutunu döngüye döndürelim.

İlk önce uygulamamızdaki if kontrol deyimlerimizi bir hatırlayalım:

```
if [ $sayi1 -gt $sayi2 ]
   then
        typeset -i sonuc
        sonuc=$sayi1-$sayi2
        echo $sonuc > $klasor1/$dosya1
        echo "İşlemin sonucu:" $sonuc " Dosya yolu:" $klasor1/$dosya1
   else
        echo "1. sayı 2. sayıdan küçük. Lütfen tekrar giriniz."
jumpto islembasi
fi
```

Şekil 1:

Eğer girilen sayi1 diğer girilen sayi2'den küçükse jumpto islembasi komutu ile "islembasi noktasına dön." diyoruz.
If kontrolü genel itibarıyla yukarıda kullandığımız gibidir.
If kontrolü gördüğünüz gibi fi ile son ermekte.

Gelelim jumpto komutuna;
Betik, jumpto gördüğü yerde yanında yazan değerin noktası varsa (yoksa uygulama zaten çalışmaz) o noktaya zıplar ve oradan çalışmaya devam eder.

```
islembasi:
echo "Lütfen birinci sayıyı giriniz"
read sayi1
echo "Lütfen ikinci sayıyı giriniz"
read sayi2
```

Şekil 2:

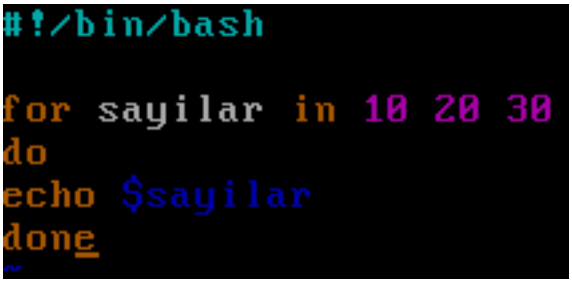
Resimde zıpladığımız nokta olan islembasi: noktasını görüyoruz. Uygulamanın satırlarından bunu hatırlamaktayız.

Şimdi buraya kadar bahsettiğimiz aralığı döngü ile kontrol edip tasvip edilmeyen zıplama mantığını kaldıracağız.

> Yazdığımız bu kadar küçük bir uygulamada jumpto 'nun hiçbir sakıncası yoktur. Hatta bir noktada işimizi çok kolaylaştırabilir. GoTo'yu sadece örnek olarak verdik. Bu tür yorumlar tabii ki büyük projeler için geçerlidir.

Döngüler:
Döngüler, komut veya komutları tekrarlatmak amacıyla kullanılır. Belli bir sayıya ulaşana kadar tekrar edebileceği gibi bir koşula bağlı olarak da çalışabilir.

Bizim uygulamamızda koşula bağlı olan bir döngü söz konusu. Buna değinmeden önce örnek bir döngü yapıp sonra uygulamamızdaki ile pekiştirelim.



Şekil 3:

İşte yukarıda koşula bağlı olmayan, elemanları kadar dönecek bir döngü örneği görmekteyiz.
Bu döngü ekrana sırayla 10 20 30 elemanlarını basar. Eleman sayıları kadar çalışır ve ardından sonlanır. Bu döngü sabit sayıda çalışır-döner (eleman sayısı kadar). Ayrıca bu örnekte For-do-done döngüsünü kullandık.

Uygulamamız için ihtiyacımızdan bahsedecek olursak; örneğimizdeki döngümüz koşula bağlı olacak. Peki koşulumuz ne? Hemen örneğimizi hatırlıyoruz.
sayi1, sayi2'den küçük olamaz. Döngü, sayi1 sayi2'den küçük olduğu müddetçe çalışacak.
Buna göre döngünün kaç kez çalışacağı önceden belli olamaz. Yani döngü ilk örneğimize göre sabit çalışma sayılı değildir.
Sayi1-lt küçük kontrolü yapar.\
-gt büyük kontrolü yapar.\
-ge büyük eşit kontrolü yapar.\
-le küçük eşit kontrolü yapar.\
-eq eşit kontrolü yapar.\
-ne eşit değil kontrolü yapar.\

Anlaşılacağı üzere yapacağımız kontrole göre kullanacağımız komutları belirliyoruz.
Biz -lt 'yi kullandık. Bu şu demek oluyor. Sayi1 -lt sayi2 Yani sayi1, sayi2'den küçükse.
Aynı döngümüzde Sayi2'yi başa yazıp -gt komutunu kullanarak sayi2, sayi1'den büyükse sorusunun kontrolünü yaparak da aynı sonuca ulaşabilirdik.

Şimdi tam döngümüzü yazalım.

```
1 While [ sayi1 -lt sayi2]
2 do
3 echo "Lütfen 1. sayıyı giriniz."
4 read sayi1
5 echo "Lütfen 2. sayiyi giriniz"
6 read sayi2
7 done
```

Şimdi de bu döngümüzü uygulamamıza yerleştiriyoruz.

Eski uygulamamızı hatırlarsak;

Buradan jumpto isimli fonksiyonu kaldırıyoruz.

Ardından islembasi: satırını da kaldırıyoruz.

Ve son orlak da if kontrol komutlarımızı kaldırıyoruz.
Yalnız buradaki typeset -i sonuc ile başlayan hesaplamamız kalıyor. Yani then ve else arasını koruyarak resimdeki alanı kaldırıyoruz. Bu hesaplamamızı döngü sonrasına koyacağız.

Birinci ve ikinci sayılarımızı istediğimiz read sayi2 ile biten satırın sornrasına da döngümüzü yazıyoruz.

```
#!/bin/bash
function jumpto
{
    label=$1
    cmd=$(sed -n "/$label:/{:a;n;p;ba};" $0 | grep -v ':$')
    eval "$cmd"
    exit
}
echo "Lütfen birinci klasörün ismini giriniz."
read klasor1
echo "Lütfen ikinci klasörün ismini giriniz."
read klasor2

mkdir $klasor1
mkdir $klasor2
echo $klasor1 " ve " $klasor2 " isminde klasörler oluşturuldu."

echo "Lütfen oluşturulacak dosyanın ismini giriniz."
read dosya1
touch $klasor1/$dosya1
echo  $dosya1 " isimli dosya oluşturuldu."

islembasi:
echo "Lütfen birinci sayıyı giriniz"
read sayi1
echo "Lütfen ikinci sayıyı giriniz"
read sayi2
if [ $sayi1 -gt $sayi2 ]
   then
        typeset -i sonuc
        sonuc=$sayi1-$sayi2
        echo $sonuc > $klasor1/$dosya1
        echo "İşlemin sonucu:" $sonuc " Dosya yolu:" $klasor1/$dosya1
   else
        echo "1. sayı 2. sayıdan küçük. Lütfen tekrar giriniz."
jumpto islembasi
fi
```

Şekil 4:

```
function jumpto
{
    label=$1
    cmd=$(sed -n "/$label:/{:a;n;p;ba};" $0 | grep -v ':$')
    eval "$cmd"
    exit
}
```

Şekil 5:

```
islembasi:
```

Şekil 6:

```
if [ $sayi1 -gt $sayi2 ]
   then
        typeset -i sonuc
        sonuc=$sayi1-$sayi2
        echo $sonuc > $klasor1/$dosya1
        echo "İşlemin sonucu:" $sonuc " Dosya yolu:" $klasor1/$dosya1
   else
        echo "1. sayı 2. sayıdan küçük. Lütfen tekrar giriniz."
jumpto islembasi
fi
```

Şekil 7:

```
while [ $sayi1 -lt $sayi2 ];
do
echo "Birinci sayi, ikinci sayidan kucuk. Lutfen birinci sayiyi tekrar giriniz."
read sayi1
echo "Lutfen ikinci sayiyi giriniz"
read sayi2
done
```

Şekil 8:

Döngümüzün sonrasına da typeset -i hesaplama satırlarımızı giriyoruz.

Uygulamamızın son hâli aşağıdaki gibi olacak.

Böylece uygulamamızdan bir fonksiyonu ve kontrolü kaldırıp bunların yerine bir döngü yazmış olduk. Uygulamamız daha efektif bir hâle gelmiş oldu.

Tabii ki asıl önemli olan neyi neye alternatif olarak kullanabileceğimizi öğrenerek kavramları benimsememizdir. Bu da o kavramları mantıklarıyla anlamayı gerektirir. Bash yazımızın başından bu yazıya kadar uygulamamızı önceki yazılarla bağlı yürüterek temel seviyede bir noktaya kadar getirmeye çalıştık. Döngülere, toparlamak amacıyla bir kez daha bakabiliriz. Bir de önemsediğimiz başka bir konu fonksiyonlar. Bu ay yeteri kadar yer kapladık. :) Fonksiyonlar ve dizilere önümüzdeki yazımızda değineceğiz.

Yine uygulamamıza uygun bir senaryo ile bu konuları işleyeceğiz.

Bu ayki yazımıza katkılarından dolayı Semetey'e hepiniz ve kendi adıma çok teşekkür ediyorum.

Tekrar görüşmek üzere…

```
#!/bin/bash

echo "Lufen birinci klasorun ismini giriniz."
read klasor1
echo "Lutfen ikinci klasorun ismini giriniz."
read klasor2

mkdir $klasor1
mkdir $klasor2
echo $klasor1 " ve " $klasor2 " isminde klasorler olusturuldu. "

echo "Lutfen olusturulacak dosyanin ismini giriniz."
read dosya1
touch $klasor1/$dosya1
echo $dosya1 " isimli dosya olusturuldu."

echo "Lutfen birinci sayiyi giriniz."
read sayi1
echo "Lutfen ikinci sayiyi giriniz."
read sayi2

while [ $sayi1 -lt $sayi2 ];
do
echo "Birinci sayi, ikinci sayidan kucuk. Lutfen birinci sayiyi tekrar giriniz."
read sayi1
echo "Lutfen ikinci sayiyi giriniz"
read sayi2
done

typeset -i sonuc
sonuc=$sayi1-$sayi2
echo $sonuc > $klasor1/$dosya1
echo "Islemin sonucu:" $sonuc " Dosya yolu:" $klasor1/$dosya1
```

Şekil 9: