Linux/Unix Sistem Programlama

2023-2024 Bahar Dönemi

8-Hafta

Dr. Öğr. Üyesi Nurullah ÖZTÜRK

Bash Kabuk Programı ile Çalışmak

- Bash Nedir?
  - Bash (Bourne Again Shell), UNIX ve UNIX benzeri işletim sistemlerinde kullanılan bir kabuk (shell) programıdır.
  - Komut yorumlayıcı ve betik dili olarak işlev görür.

Temel Özellikler

- Komut Satırı Arabirimi:
  - Kullanıcıya komutları girmesi ve sistemle etkileşimde bulunması için bir arayüz sağlar.
- Betik Dili:
  - İleri düzeyde betik oluşturma ve otomasyon için kullanılabilir.
- Genişletilebilirlik:
  - Değişkenler, döngüler, koşullu ifadeler ve fonksiyonlar gibi yapılarla geniş bir özellik yelpazesi sunar.

Özel Tanımlamalar

- Kabuk Değişkenleri:
- **$HOME:** Kullanıcının ana dizinini temsil eder.
- **$PATH:** Çalıştırılabilir dosyaların bulunduğu dizinleri içeren bir listedir.
- **$USER:** Mevcut kullanıcı adını temsil eder.
- **$PWD:** Şu anki çalışma dizinini temsil eder.
- **$SHELL:** Kullanıcının varsayılan kabuk programını temsil eder

Özel Kabuk Değişkenleri

- **$?:** En son çalıştırılan komutun çıkış durumunu temsil eder.
- **$$:** Şu anki kabuk işlemi PID (Process ID) numarasını temsil eder.
- **$!:** Arka planda çalışan en son komutun PID numarasını temsil eder.
- **$#:** Bir komut dosyasında veya işlevde kullanılan argüman sayısını temsil eder.
- **$0, $1, $2, ...:** Bir komut dosyasında veya işlevde kullanılan argümanların değerlerini temsil eder.

Kabuk Programlamaya Giriş

- Her kabuğun kendine özgü programlama dili yapısı vardır.
- Bash kabuğu ise güçlü programlama özellikleriyle karmaşık programların rahatça yazılmasına izin verir.
- Mantıksal operatörler, döngüler, değişkenler ve modern programlama dillerinde bulunan pek çok özellik bash kabuğunda da vardır ve işleyiş tarzları da hemen hemen aynıdır.
- Herhangi bir editör yardımıyla yazılan program, daha sonra kabuk altında çalıştırılır. Bir kabuk programı diğerlerini çalıştırabilir.

Kabuk Programları

- Kabuk programları, bir veya birden fazla linux komutunu tutan dosyalardır.
- Bu dosya yaratıldıktan sonra doğrudan dosyanın ismi girilerek veya dosya isminden önce '.' karakteri getirerek çalıştırılabilir.
- chmod komutu yardımıyla bir programı çalıştırılabilir yapmak için ,
  - $ chmod +x komut-ismi
- Bundan sonra programın ismi yazılıp enter tuşuna basıldığı zaman bir program Linux komutuymuş gibi çalışacaktır.

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 "=" işareti yardımıyla yapılır.
  - $ mesaj="linux dersi bugun«
- echo komutu yardımıyla bir değişkenin içeriği ekrana basılıyor
  - $ echo $mesaj

Giriş/Çıkış İşlemleri

- Bir kabuk programı çalışırken kullanıcıdan klavye yardımıyla bilgi girmesi sağlanabilir.
- Bu tür işlemler için tanımlanan read komutu klavyeyi okur ve aldığı bilgiyi bir değişkene atar.
  - echo Bir sayi giriniz..
  - read sayi
  - echo Girilen sayi : $sayi

Aritmetik İşlemler

- bash kabuğunda matematiksel işlemlere büyük sınırlamalar getirilmiştir. Tamsayı değişkeni dışında matematiksel değişken kullanmak için bu işlemler için geliştirilmiş ve kolaylıklar sağlayan **awk** veya **bc** kullanabilirsiniz.
- Aritmetik işlemler için **eval** komutunu veya bash kabuğu altında yerleşik (builtin) komut olan **let** komutunu kullanabilirsiniz.
- let "degisken=aritmetik islem«
  - $ let "carpim=2*7"
  - $ echo $carpim

if-else Kalıbı ve Kontrol İşlemleri

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

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

if-else Kalıbı ve Kontrol İşlemleri

- Aritmetik karşılaştırmalar
  - -gt büyük
    -lt küçük
    -ge büyük eşit
    -le küçük eşit
    -eq eşit
    -ne eşit değil
- Dizisel karşılaştırma
  - -z boş dizi
    -n tanımlı dizi
    = eşit diziler
    != farklı diziler
- Dosya karşılaştırması
  - -f dosya var
    -s dosya boş değil
    -r dosya okunabilir
    -w dosyaya yazılabilir
    -x çalıştırılabilir dosya
    -h sembolik bağlantı
    -c karakter aygıt
    -b blok aygıt
- Mantıksal karşılaştırmalar
  - -a VE
    -o VEYA
    ! DEĞİL

if-else Kalıbı ve Kontrol İşlemleri

- if komutu genellikle kendine test komutu ile birlikte kullanım bulur.
- Bu komut yardımıyla mantıksal işlemler yapılabilir, sayılar ve hatta diziler karşılaştırılabilir.
- Anahtar sözcük olan test'ten sonra opsiyonlar ve/veya karşılaştırılacak olan değerler yazılır. Her opsiyon bir mantıksal işleme karşılık gelir. Örneğin -lt opsiyonu ilk girilen aritmetik değişkenin ikinci değerden küçük olup olmadığını denetler. Benzer şekilde = opsiyonu da iki karakter kümesinin eşitliğini kontrol eder.

if-else Kalıbı ve Kontrol İşlemleri

- Son çalıştırılan tüm Linux komutlarının çıkış değeri $? değişkeninde tutulur. test komutunun çıkış değeri de bu yolla öğrenilebilir.

  ```
  $ sayi=4
  $ test $sayi -eq 4
  $ echo $?
  0
  $ test $sayi -lt 2
  $ echo $?
  1
  ```

- test komutu yerine parantezler de kullanılabilir. Yukarıdaki iki örnek, parantez kullanılarak şu şekilde yazılabilir:

  ```
  $ [ $sayi -eq 4 ]
  $ [ $sayi -lt 12 ]
  ```

- Dikkat edilmesi gereken bir nokta, köşeli parantez kullanırken araya boşlukların eklenmesidir. Parantezler başlı başına bir komut olarak görüldüklerinden sağında ve solunda en az bir boşluk bırakılmalıdır.

if-else Kalıbı ve Kontrol İşlemleri

- Örnek

```
#!/bin/bash
echo "0 ile 20 arasinda bir sayi secin"
read sec
if [ $sec -lt 10 ]
  then
      echo "Secilen sayi tek basamakli"
  else
      echo "Secilen sayi cift basamakli"
fi
```

case Kalıbı

- Birkaç alternatif arasından seçim yapmak için kullanılan bir komut olan case, bir eşleştirme gördüğü anda belirli bir komut kümesini işleme sokar. case yapısı case komutu ile başlar, eşleştirilecek olan anahtar sözcük yazılır ve seçenekler alt alta, her seçeneğe ait olan komutlarla birlikte belirtilir. Tüm yapı esac komutu ile son bulur.

```
case anahtar-sozcuk in
        secenek1)
                komutlar
                ;;
        secenek2)
                komutlar
                ;;
        *)
                komutlar
                ;;
esac
```

case Kalıbı

- Örnek

```
#!/bin/bash
clear
echo    "1. ekrani temizle"
echo    "2. sistemdekileri goruntule"
echo    "3. dizindeki dosyalari goster"

echo -n "Secenegi giriniz : "
read secenek
```

```
case $secenek in
        1)
          clear
          ;;
        2)
          w
          ;;
        3)
          ls -al
          ;;
        *)
          echo Hatali secenek
esac
```

Döngüler

- **while-do Döngüsü**
  - Döngü bloğu while anahtar kelimesiyle başlar, ardından gelen koşul sağlandığı sürece döngü işletilir. Önce koşulun sağlanıp sağlanmadığına bakılır. Döngüden çıkabilmek için mutlaka döngü içindeki koşul ifadesinin değerini yanlış yapacak bir durum oluşmalıdır, aksi halde sonsuz döngü oluşur.

```
while kosul ifadesi
   do
        komutlar
   done
```

while-do Döngüsü

- if komutuyla birlikte kullanılan test komutu, while döngüsünde koşul ifadesi olarak da yer alabilir. Aşağıda 1'den 100'e kadar sayan ve ekrana basan bir döngü görülüyor.

```
#!/bin/bash
deger=0
while [ $deger -lt 100 ]
  do
      deger=$((deger+1))
      echo $deger
  done
```

- Yukarıda kullanılan ((ve)) karakterleri arasına matematiksel bir işlem getirilebilir.

for-do döngüsü

- Bir liste dahilindeki tüm değerlere sırayla erişimi sağlar. for komutundan sonra yeralan liste sırayla kullanılır ve herbirisi için döngü çalıştırılır. Listenin sonuna gelindiğinde ise döngüden çıkılır.

```
for degisken1 in deger1 deger2 ... degerX
   do
        komutlar
   done
```

Örnek Kabuk Programı