# LINUX SHELL PROGRAMLAMA

- **Kernel:** İşletim sistemi çekirdeğini oluşturan programdır. Sorumlu olduğu alanlar şunlardır:
  1) File, 2) Process, 3) I/O, 4) Memory ve 5) Device Management.
- **Shell:** İşletim sistemi servislerine erişmek için kullanılan programdır.
  BASH: Bourne Again Shell, CSH: C Shell, KSH: Korn Shell.

- **Temel Shell Komutları**
  - **clear:** Terminaldeki tüm yazıları temizler.
  - **ls:** Dosyaları listelemek için kullanılır.
    - ls: Bulunulan dizindeki dosyaları listeler.
    - ls -l: Bulunulan dizindeki dosyaları listeler ve dosya izinleri ile ilgili bilgiler verir.
    - ls -a: Bulunulan dizindeki dosyaları, üst ve alt dizinleri listeler.
    - ls dizin: Belirtilen dizindeki dosyaları listeler.
  - **cd:** Dizin değiştirmek için kullanılır.
    - cd: /home klasörüne (temel dizine) gider.
    - cd -: Bir önceki bulunulan dizine gider.
    - cd ..: Bir üst dizine gidebilmeyi sağlar.
  - **pwd:** Terminalde o an bulunduğumuz dizini gösterir.
  - **nano:** Dosyaların içeriğini düzenlemek için kullanılan yardımcı programı çalıştırır.
    - nano dosya: "dosya"'yı düzenlemek üzere açar.
  - **cat:** Dosya içeriğini okuma ve çeşitli dosya kaydetme işlemleri için kullanılır.
    - cat dosya: "dosya"'nın içeriğini okur ve o içeriği terminale döker.
    - cat > dosya: "dosya" isminde yeni bir dosya oluşturur ve terminalden içeriğini doldurmamızı bekler.
    - cat dosya1 dosya2 > dosya3: "dosya1" ve "dosya2"'yi okur, ikisinin içeriğini "dosya3"'e ekler.
  - **touch:** Yeni ve boş bir dosya oluşturur.
    - touch dosya: "dosya" isminde yeni ve boş bir dosya oluşturur.
  - **cp:** Kopyalama işlemi için kullanılır.
    - cp dosya1 dosya2: "dosya1" dosyasının içeriğini kopyalar ve "dosya2" adıyla kaydeder.
  - **mv:** Taşıma işlemi için kullanılır.
    - mv dosya dizin: "dosya" dosyasını, belirtilen "dizin"'e taşır.
  - **mkdir:** Klasör oluşturmak için kullanılır.
    - mkdir klasör: "klasör" isminde yeni bir klasör oluşturur.
    - mkdir -p klasör/altklasör: "klasör" içinde istenilen miktarda alt klasör oluşturur.
  - **rmdir:** Klasör silmek için kullanılır.
    - rmdir klasör: Boş bir klasörü silmek için kullanılır. Yalnızca boş klasörü siler.
  - **rm:** Dosya ve Klasör silme işlemleri için kullanılır.
    - rm dosya: "dosya"'yı siler.
    - rm -r klasör: "klasör"'ü ve alt dizinlerinde bulunan tüm dosya ve klasörleri siler.
  - **echo:** Terminale yazı yazdırmak için çift tırnakla kullanılır.
    - echo "$USER": Anlık olarak terminali kullanan kullanıcıyı ekrana yazdırır.
    - echo -e "deneme \a": Terminale basıldığında bir uyarı sesi çıkartır.
    - echo -e "dene\bme": Yazı arasında kullanıldığında, kendinden bir önceki karakteri siler. "denme"
    - echo -e "deneme \c": Terminale basılan çıktının sonundaki yeni satırı siler.
    - echo -e "deneme \n": Terminale basılan çıktının sonuna satır başı ekler.
    - echo -e "deneme \n\r": Terminale basılan çıktının sonuna yeni bir satır ekler.
    - echo -e "deneme\tdeneme": Kullanıldığı yerde bir TAB tuşu kadar boşluk bırakır.
    - echo -e "deneme\\deneme": Ters slaş "\" karakterinin yazılabilmesini sağlar.
    - echo "yaz beni1" > dosya.txt: "dosya.txt" dosyasına "yaz beni1" cümlesini baştan yazar, dosya yoksa oluşturur.
    - echo "yaz beni2" >> dosya.txt: "dosya.txt" dosyasına "yaz beni2" cümlesini dosya sonuna ekler, dosya yoksa oluşturur.
  - **read:** Terminale input açar. Kullanıcının veri girmesini sağlar.
    - read kullanici_adi: Kullanıcıdan "kullanici_adi" değişkenine veri girmesini sağlar.
      Daha sonra $kullanici_adi şeklinde, girilen veri kullanılabilir.
  - **sort:** Sıralama işlemi yapmak için kullanılır.
    - sort dosya: "dosya" içerisindeki satırları a-z veya 0-9'a sıralar, dosya güncellenmez, sadece içeriği terminale dökülür.
    - sort -r dosya: "dosya" içerisindeki satırları z-a veya 9-0'a sıralar, dosya güncellenmez, sadece içeriği terminale dökülür.
    - sort dosya > yenidosya: "dosya" içeriği a-z veya 0-9'a sıralanır ve "yenidosya" dosyasına kaydedilir.
    - sort -k2 kolonludosya: Boşlukla birbirinden ayrılan kolonlar tablosunda 2. kolona göre alfabetik sıralama yapar.
    - sort -k2 -n kolonludosya: Boşlukla birbirinden ayrılan kolonlar tablosunda 2. kolona göre nümerik sıralama yapar.
    - sort -k2 -n -t',' kolonludosya: Virgüller ile birbirinden ayrılan kolonlar tablosunda 2. kolona göre nümerik sıralama yapar.

- **expr:** Aritmetik işlemler yapmak için kullanılır.
  - expr 1 + 2: Terminale 3 yazar. Bu komutla ilgili tüm işlemlerde operatörün yanlarında mutlaka boşluk olmalıdır.
  - expr 2 - 1: Çıkartma işlemi.
  - expr 3 * 3: Çarpma işlemi. * operatörünü kullanmak için \ karakteri de mutlaka kullanılmalıdır.
  - expr 10 % 3: Mod alma işlemi. % operatörü \% şeklinde de kullanılabilir.
  - echo `**expr** 5 + 3`: Bu şekilde işlem sonucu terminale çıktı olarak verilebilir. ` karakteri klavyede **~** karakterinin altındadır.
- **awk:** Filtreleme işlemleri için kullanılır.
  - awk '**int($1)>3**' kolonludosya: Boşlukla birbirinden ayrılan kolonlar tablosunda, 1. kolonda değeri 3'ten büyük olan satırları döndürür.
  - awk -F',' '**int($1)>3 && int($1)<7**' kolonludosya: Virgüller ile birbirinden ayrılan kolonlar tablosunda, 1. kolonda değeri 3'ten büyük ve 7'den küçük olan satırları döndürür.
- **sudo:** Kullanıcı yönetimi için kullanılır.
  - sudo useradd -m kullaniciadi -p şifre: "kullaniciadi" ve "şifre" bilgisiyle linux'ta /home klasöründe yeni kullanıcı oluşturulur.
  - sudo useradd -M kullaniciadi -p şifre: Üsttekinden tek farkı /home klasöründe oluşturulmamasıdır.
  - sudo userdel -r kullaniciadi: "kullaniciadi" kullancısını siler.
  - sudo userdel -r -f kullaniciadi: "kullaniciadi" kullancısını, sahip olduğu tüm dosyalarla birlikte siler.
  - sudo groupadd grupadi: "grupadi" adında yeni bir kullanıcı grubu oluşturur.
  - sudo usermod -a -G grupadi kullaniciadlari: "kullanıcıadlari"'ni "grupadi" adlı gruba ekler.
  - sudo groupdel grupadi: "grupadi" adlı kullanıcı grubunu siler.
- **chown:** Dosyanın sahipliğini değiştirir.
  - chown :kullaniciadi dosya: "dosya"'nın yeni sahibi "kullaniciadi" adlı kullanıcı olur.
    Kullanıcı adı yerine Grup adı da yazılabilir.
- **chmod**: Dosyanın izinlerini değiştirir.
  - chmod izinkodu dosya: "dosya"'nın izinlerini verilen izinkodu ile düzenler.
    - İzinkodu 3 haneli bir sayıdır ve dosya üzerinde gruplara göre okuma, yazma ve çalıştırma izinlerini düzenler.
    - Okuma izni: **4**, Yazma izni: **2**, Çalıştırma izni: **1** ve hiçbir işleme izin verilmeme durumu da **0** ile ifade edilir.
    - Her bir hane, 0-7 arası rakam alabilir. Her bir rakam, söz konusu izinlerin toplamıdır. (Tüm izinler için, 4+2+1=7)
    - Birinci hane, dosya sahibinin yapabileceği işlemleri;
      İkinci hane, dosya grubunun yapabileceği işlemleri;
      Üçüncü hane ise diğer kullanıcıların yapabilecekleri işlemleri ifade eder.
    - **ls** -l komutu ile terminalde dosyaların izinleri ile ilgili bilgiler de çıkar. Bu bilgilerde r,w,x ve – karakterleri bulunur.
    - **r:** okuma, **w:** yazma, **x:** çalıştırma ve **-** karakteri ise izinsizlik durumunu açıklar.
    - Örneğin, **drwxrwxrwx** şeklindeki bir ifadede en baştaki **d** karakteri dosya tipini belirtir. Sonraki karakterleri üçer üçer alırsak, ilk üç karakter izinkodundaki birinci haneyi, ortadaki üç karakter ikinci haneyi, son üç karakter de izinkodundaki üçüncü haneyi belirtmiş olur. Böylece **drwxrwxrwx** ifadesi 777 izinkoduna karşılık gelmiş olur.
    - Aynı şekilde **d-wx-w-rwx** ifadesi **327**, **drwxr-xr-x** ifadesi **755**, **drwx------** ifadesi de **700** olmuş olur.
    - **Örnek olarak 755 kodu**;
      dosya sahibinin okuma, yazma, çalıştırma gibi tüm izinlere sahip olduğunu (4+2+1=**7**),
      dosya grubunun okuma ve çalıştırma yapabileceğini (4+0+1=**5**),
      diğer kullanıcıların da sadece okuma ve çalıştırma yapabileceğini (4+0+1=**5**) belirtmiş olur.
- **egrep:** Bir dosyada veya klasörde bulunan belli metinleri filtrelemek için kullanılabilir.
  - egrep aranan dosya: "dosya" dosyasındaki metinlerde aranan kelimeye göre filtreleme yapar. Aranan ifade regex formatında da verilebilmektedir.

- **Shell Değişkenleri:** Sistem değişkenleri büyük harflerden oluşur. Kullanıcı tarafından oluşturulan değişkenlerin de genellikle küçük harflerden oluşturulması beklenir. Bazı sistem değişkenleri aşağıdaki gibidir. Bu değişkenlerin değerlerini görüntülemek için terminalde echo **$**DEĞİŞKENADI yazmak yeterlidir.
  - **SHELL:** Shell adını verir.
  - **COLUMNS:** Terminal ekranının sütun sayısını verir.
  - **LINES:** Terminal ekranının satır sayısını verir.
  - **HOME:** /home klasörünün dizinini verir.
  - **USER:** Anlık olarak terminali kullanan kullanıcının, kullanıcı adını verir.
  - **OSTYPE:** İşletim sistemi tipini verir.
  - **PATH:** /path klasörünün dizinini verir.
  - **PWD:** Anlık olarak kullanıcının bulunduğu klasörün dizinini verir.

- **Değişken Oluşturma ve Değişken Kullanımı:** Değişken isimleri alt çizgi (_) veya harf ile başlar. Eşittir işaretinin sağında ve solunda boşluk olmamalıdır. Nümerik değişkenler direkt eşittir yazarak, String değişkenler ise "çift tırnak" kullanılarak oluşturulur. Eşittir işaretinden sonra hiçbir değer yazılmazsa, değişken NULL değerini alır. Kullanıcı değişkenlerine istenirse, sistem değişkenleri de atanabilir.
    - degisken=; echo $degisken
    - degisken2=789; echo $degisken2
    - degisken3="text metin"; echo $degisken3
    - degisken4=$PWD; echo $degisken4

- **Pipe ile Komutları Birbirine Bağlamak:** Pipe karakteri "|" birden fazla komutu birbirine bağlayarak çalıştırabilir. Bir komutun çıktısı, sonraki komuta parametre olarak verilir. Örneğin, dosya.txt içerisinde isimler listesi olsun. Terminale cat dosya.txt yazdığımız zaman, dosya.txt içerisindeki isimler alt alta terminale dökülür. Eğer cat dosya.txt | sort yazarsak, dosya verileri alfabetik olarak sıralanmış bir şekilde terminale dökülür. Yani bir sonraki komut olan, sort komutuna parametre olur ve sanki tek başına sort dosya.txt komutu yazılmış gibi çalıştırılır. Bu şekilde istediğimiz kadar pipe karakteri kullanarak, çıkan yeni sonuca yeni işlemler yaptırabiliriz.
    - **tr** komutu pipe ile kullanılabilen bir transform komutudur. Aşağıdaki kodda ; karakterlerini - karakterine çevirir.
      echo "ali;ayşe;mehmet;canan" > birdosya.txt; cat birdosya.txt | tr ';' '-'
    - Aşağıdaki tr komutu küçük harfleri büyük harfe dönüştürür.
      echo "ankara konya antalya" > yenidosya.txt; cat yenidosya.txt | tr a-z A-Z

- **Koşullu İfadelerin Kullanımı:** Koşullu ifade komutlarını .sh uzantılı bir dosya içerisine yazıp, onu sh dosya.sh gibi çağırarak çalıştırabiliriz. Koşullu ifadeleri yazarken özellikle boşluklara çok dikkat edilmelidir. Dosya çağırırken kullanacağımız bazı değişkenler şunlardır:
    - $0: Çalıştırılan dosyanın ismi.
    - $1: Çalıştırılan dosyaya komut satırında (dosya isminden sonra) verilen birinci parametre.
    - $N: Çalıştırılan dosyaya komut satırında (dosya isminden sonra) verilen N'inci parametre.
    - $#: Çalıştırılan dosyaya komut satırında (dosya isminden sonra) verilen toplam parametre sayısı.

**Örnek 1:** if-else kodlarımızı yazdığımız dosyanın ismi ifelse.sh olsun ve sh ifelse.sh yazıp çalıştıralım.

```
if [ $0 == "ifelse.sh" ]
then echo "Dosyamızın ismi $0"
fi
```

**Örnek 2:** Bu sefer dosyamıza aşağıdaki kodu yazalım ve terminale sh ifelse.sh test yazıp çalıştıralım, sonrasında ise sh ifelse.sh 1 yazalım. Burada == operatörü ile metinsel karşılaştırma yapıyoruz.

```
if [ $1 == "test" ]
then echo "yazdığınız birinci parametre test"
else echo "yazdığınız birinci parametre test DEĞİL!"
fi
```

**Örnek 3:** Eğer sayısal olarak karşılaştırma yapmak istersek **-eq** komutunu kullanmamız gerekir.
Önce sh ifelse.sh 11 11 daha sonra sh ifelse.sh 11 12 yazıp deniyoruz.

```
if [ $1 -eq $2 ]
then echo "ilk iki parametre birbirine EŞİT"
else echo "birinci ve ikinci parametreler birbirine EŞİT DEĞİL!"
fi

if [ $1 -gt $2 ]
then echo "ilk parametre ikincisinden BÜYÜK"
elif [ $1 -lt $1 ]
then echo "ilk parametre ikincisinden KÜÇÜK"
else echo "birinci ve ikinci parametreler birbirine EŞİT!"
fi
```

**Örnek 4:** Dosyanın yazdırılabilir veya okunabilir olup olmadığını kontrol edelim. sh ifelse.sh birdosya.txt

```
if [ -w $1 ]
then echo "$1 dosyası yazdırılabilir bir dosya"
elif [ -r $1 ]
then echo "$1 dosyası okunabilir bir dosya"
else echo "$1 dosyası ne yazdırılabilir, ne okunabilir!"
fi
```

- **Döngülerin Kullanımı:** For döngüsünde, for i in 1 2 3 4 5 ile for i in {1..5} ve for ((i=1; i<=5; i++)) aynı anlama gelir. Şimdi donguler.sh dosyasını oluşturalım ve bu sefer dosyamızı çalıştırırken, **./donguler.sh** komutunu uygulayalım. "Permission Denied" hatası verirse, **chmod 777 donguler.sh** yaparak, sorunu düzeltebiliriz.

  **Örnek 1:** Bu örnekte 1'den 5'e kadar terminale sayı yazdıracağız. ./donguler.sh
  ```
  for i in 1 2 3 4 5
  do echo $i
  done
  ```

  **Örnek 2:** Bu sefer sayıları 1'den 10'a kadar 2'şer 2'şer yazdıracağız. ./donguler.sh
  ```
  for i in {1..10..2}
  do echo $i
  done
  ```

  **Örnek 3:** Bir shell komutunun çıktısını, döngü için kullanabiliriz.
  Aşağıdaki örnekte, ls komutundan gelen tüm değerleri, sırayla döngüye sokup kullanacağız. ./donguler.sh
  ```
  for i in $(ls)
  do echo "Dosya: $i"
  done
  ```

  **Örnek 4:** Liste döndüren her komutun sonucunu, döngü için kullanabiliriz. Aşağıdaki örnekte, dosya ve klasörleri listeleyen komutun sonucunu döngüye sokup, listeleme yaparken, dosya ve klasör ayırımı da yapabileceğiz. ./donguler.sh
  ```
  for i in ~/*
  do if [ -f $i ]
  then echo "Dosya: $i "
  elif [ -d $i ]
  then echo "Klasör: $i"
  else echo "Bilinmeyen: $i "
  fi
  done
  ```

  **Örnek 5:** While döngüsü. ./donguler.sh
  ```
  i=10;
  while [ $i -gt 0 ]
  do echo "sayaç: $i";  i = `expr $i - 1`
  done
  ```

  **Örnek 6:** Until do döngüsü. Koşul sağlanmadığı sürece devam eden döngüdür. ./donguler.sh
  ```
  c=6
  until [ $c -lt 3 ]
  do let c-=1
  echo $c
  done
  ```

- **Case Komutu Kullanımı:** if-else alternatifi olan switch-case'in shell'deki versiyonudur. Case durumları son karakterde tek parantezle ifade edilir, en sonuna da iki adet **;** konulur.

  **Örnek 1:**
  ```
  read ariyorum
  case $ariyorum in
  "araba") echo "araba arıyorsun";;
  "ev") echo "ev arıyorsun";;
  *) echo "ne aradığınız belli değil";;
  esac
  ```