TEMEL LİNUX KOMUTLARI

Linux Dosya Yapısı

```
/ -> Kök dizini
├── bin/ -> Temel kullanıcı komutları (ls, cp, mv, cat, …)
├── boot/ -> Çekirdek ve önyükleme dosyaları (kernel, grub)
├── dev/ -> Donanım aygıt dosyaları (/dev/sda, /dev/null)
├── etc/ -> Sistem yapılandırma dosyaları/konfigürasyon (/etc/passwd, /etc/hosts)
├── home/ -> Kullanıcıların ev dizinleri (/home/ali, /home/bugra)
├── lib/ -> Paylaşımlı kütüphaneler (.so dosyaları)
├── media/ -> Çıkarılabilir ortamlar (USB, CD, DVD)
├── mnt/ -> Geçici bağlama noktaları (mount edilen dosya sistemleri)
├── opt/ -> Opsiyonel yazılımlar (3. parti paketler)
├── proc/ -> Çalışan süreçler ve sistem bilgileri (sanal dosya sistemi)
├── root/ -> Root kullanıcısının ev dizini
├── run/ -> Geçici çalışma zamanı dosyaları (PID dosyaları vs.)
├── sbin/ -> Sistem yönetim komutları (ifconfig, shutdown, …)
├── srv/ -> Sunucu servis verileri (örn: web, ftp)
├── sys/ -> Donanım ve çekirdek bilgileri
├── tmp/ -> Geçici dosyalar (yeniden başlatınca silinir)
├── usr/ -> Kullanıcıya ait uygulamalar ve kütüphaneler
│ ├── bin/ -> Kullanıcı uygulamaları (ls, grep, awk, …)
│ ├── sbin/ -> Yönetim uygulamaları
│ ├── lib/ -> Kütüphaneler
│ └── share/ -> Paylaşılan dosyalar (dökümantasyon, ikonlar)
└── var/ -> Değişken dosyalar (loglar, spool, cache)
├── log/ -> Sistem log dosyaları
├── mail/ -> E-posta verileri
├── spool/ -> Yazıcı ve e-posta kuyrukları
└── tmp/ -> Geçici dosyalar
```

Semboller

- **$** : Sizin sıradan (normal) kullanıcı olduğunuzu belirtir
- **#** : Süper kullanıcı (root) olduğunuzu belirtir
- **~** : Kullanıcının ev dizininde bulunduğunuzu belirtir
- **/** : Kök dizin (en üst seviye)
- **.** : Bulunduğun mevcut dizin
- **..** : Bir üst dizin
- **-** : Bir önce bulunduğun dizin
- **~kullanici** : Belirtilen kullanıcının home dizini (örn: ~root → /root)

Linux Komut Yapısı

```
komut [seçenekler] [argümanlar]
```

- **Komut:** Komutun kendisidir (işlev)

- **Seçenekler (Options):** Komutun işlevini değiştirmeye yarar. (-) işareti ile başlar
- **Uzun yazılış:** Çift tire (--) tamamı yazarız
- **Argümanlar (Arguments):** Komutun üzerinde işlem yapacağı nesnedir

Örnek: `-v --verbose` İşlemin aşamalarını ayrıntılı gösterir.

Temel Seçenekler

Seçenek	Açıklama
`-h, --help`	Komutun yardım ekranını gösterir
`-v, --version`	Komutun sürümünü gösterir
`-r`	Recursive (özyinelemeli işlem)
`-f`	Force (zorla çalıştırma)
`-i`	Interactive (işlemden önce sorar)
`-q`	Quiet (sessiz mod)
`-n`	Satır/limit belirtme (örn: head -n 5)

📂 ls (listeleme)

Komut	Açıklama
`ls`	Bulunduğun dizindeki dosyaları listele
`ls -l`	Ayrıntılı listeleme (izinler, boyut vb.)
`ls -a`	Gizli dosyaları da göster
`ls -la`	Gizli dosyaları ayrıntılı listele
`ls -lh`	Boyutları okunabilir formatta
`ls -R`	Alt dizinleri de listele
`ls -lt`	Dosyaları/dizinleri son değişiklik tarihine göre sıralar

📂 mkdir/rmdir (Klasör oluşturma/silme)

Komut	Açıklama
`mkdir -p`	Tek seferde iç içe birden fazla dizin oluşturur
`mkdir --mode`	İstediğin erişim haklarına sahip dizinler yaratır
`rmdir`	İçi boş bir dizini silmek için kullanılır

📄 cp (dosya kopyalama)

Komut	Açıklama
`cp dosya hedef/`	Dosyayı hedefe kopyala
`cp -r dizin hedef/`	Dizini içindekilerle kopyala
`cp -i dosya hedef/`	Üzerine yazmadan önce sor

`cp -f dosya hedef/`	Zorla kopyala
`cp -v dosya hedef/`	Kopyalanan dosyaları göster

✂️ rm (dosya/dizin silme)

Komut	Açıklama
`rm dosya.txt`	Dosya sil
`rm -i dosya.txt`	Silmeden önce sor
`rm -f dosya.txt`	Onay sormadan sil
`rm -r dizin/`	Dizini ve içindekileri sil
`rm -rf dizin/`	Onaysız, dizini komple sil

📖 cat / head / tail (dosya görüntüleme)

Komut	Açıklama
`cat dosya.txt`	Dosyanın tamamını göster
`cat -n dosya.txt`	Satırları numaralar
`cat > dosya.txt`	Dosya içindeki metni sil yeni metni ekle
`cat >> dosya.txt`	Dosya içindeki metnin sonuna ekle
`head dosya.txt`	İlk 10 satırı göster
`head -n 5 dosya`	İlk 5 satırı göster
`tail dosya.txt`	Son 10 satırı göster
`tail -f dosya.txt`	Sürekli güncellenen log dosyalarını izlemek için
`tac`	Dosya içeriğini ekrana tersten yazdırır

🔍 grep (arama)

Komut	Açıklama
`grep kelime dosya.txt`	Dosya içinde kelime ara
`grep -i kelime dosya.txt`	Büyük/küçük harf duyarsız arama
`grep -r kelime /dizin/`	Dizin içinde özyinelemeli arama
`grep -n kelime dosya.txt`	Satır numarasıyla göster
`grep -v "error" log.txt`	Eşleşmeyen satırları göster
`grep -c "error" log.txt`	Eşleşen satırların sayısını göster
`grep -o "error" log.txt`	Sadece eşleşen kısmı yazdırır

echo (ekrana yazdırma)

echo komutu Linux'ta ekrana çıktı vermek için kullanılır.

```
echo Merhaba Dünya
```

Değişkenlerle Kullanım:

```
NAME="Bugra" echo Merhaba $NAME
```

Tık tırnak - Çift tırnak meselesi

Tek tırnak: İçindeki her şeyi olduğu gibi yazar. Değişkenler, $, \n, \t gibi özel karakterler çalışmaz.

Çift Tırnak: İçindeki değişkenleri ve özel karakterleri işler. $ ile başlayan değişkenler yerine değerini koyar.

Linux Kontrol Karakterleri

Kontrol Karakter	Tuş Kombinasyonu	Açıklama
`^C`	Ctrl + C	Çalışan işlemi durdurur
`^Z`	Ctrl + Z	Çalışan işlemi arka plana gönderir (suspend)
`^D`	Ctrl + D	Girdi sonu (EOF), terminali kapatır
`^S`	Ctrl + S	Terminali durdurur (flow control)
`^Q`	Ctrl + Q	Terminali yeniden başlatır
`^U`	Ctrl + U	Satırın başına kadar siler
`^K`	Ctrl + K	İmleçten satır sonuna kadar siler
`^W`	Ctrl + W	Son kelimeyi siler
`^L`	Ctrl + L	Terminal ekranını temizler (clear)
`^R`	Ctrl + R	Komut geçmişinde arama yapar
`^A`	Ctrl + A	Satır başına gider
`^E`	Ctrl + E	Satır sonuna gider

Man Sayfasında Gezinme

- **spacebar** - man sayfasının bir sonraki ekranını gösterir
- **return** - her basışta bir satır gösterir
- **b / PgUp** - bir ekran öncesine döner
- **f / PgDown** - bir ekran sonrasına gider
- **q** - man sayfasından çıkar
- **/kelime** - yazılan kelimeyi bulunulan yerden itibaren ileriye doğru arar
- **n** - aranan kelimenin bir sonraki geçtiği yeri gösterir
- **h** - tüm bu işlemler için yardım sunar

less Komutu

Dosya içeriğini sayfa sayfa veya satır satır gösterir.

- **spacebar** - Bir sayfa ileri
- **enter** - Bir satır ileri
- **b** - Bir sayfa geri
- **q** - Çık
- **/kelime** - Belirtilen kelimeyi ara
- **n** - Son aramayı yinele

Dosya Yolları

- **Mutlak (Tam):** Kök klasöründen (/) başlayarak
- **Bağıl:** Bulunulan dizine göre
- **.** - İçinde bulunulan klasör
- **..** - İçinde bulunulan klasörün bir üst klasörü
- **-** - Bir önceki klasör
- **~** - Mevcut kullanıcının ev klasörü
- **~kullanıcı** - Belirtilen kullanıcının ev klasörü
- **/** - Kök klasörü (dizini)

Birkaç Örnek Komut

```
ls -la ~/Desktop/kabuk # masaüstündeki kabuk klasörünün içini detaylı listelerken gizli dosyaları da göster ls -lt # dosyaları/dizinleri son değişiklik tarihine göre sıralar pwd # normal kullanıcının home dizinin yolunu göster cd /var/log # kök dizin → var dizini → log dizini cd ../.. # bulunduğun dizinin bir üstünün bir üst dizinine geçer cd ~ # Direkt olarak kendi ev dizininize gidebilirsiniz cd ~bugor # bugor kullanıcısının kök dizini cd - # En son bulunduğunuz dizine geri dön head -n 3 dosyaAdi # dosyanın ilk 3 satırını gösterir tail -n 3 dosyaAdi # dosyanın son 3 satırını gösterir mv deneme.txt ~/Desktop # deneme.txt'yi desktop'a taşı
```

Erişim İzinleri

İzin Türleri

İzin	Açıklama
r (read)	Okuma izni / içeriğinin görüntülenmesine ve kopyalanmasına izin verilmesi
w (write)	Yazma / değiştirme izni / içeriğinin değiştirilebilmesine ve silinebilmesine izin verilmesi
x (execute)	Çalıştırma izni

Kullanıcı Grupları

Grup	Açıklama
Owner (kullanıcı / u)	Dosyanın sahibi
Group (g)	Dosyanın ait olduğu grup
Others (o)	Diğer tüm kullanıcılar

Örnek: -rwxr-xr--

- **-** → dosya (d olsaydı klasör)
- **rwx** → sahibi (user) okuma, yazma, çalıştırma yapabilir
- **r-x** → gruptakiler okur ve çalıştırır ama yazamaz
- **r--** → diğer kullanıcılar sadece okuyabilir

Erişim hakkı değiştirme

```
chmod u+x dosya.sh # Kullanıcıya çalıştırma izni ekler chmod g-w dosya.txt # Gruptan yazma iznini alır chmod o=r dosya.txt # Diğerleri sadece okuma yapabilir
```

Sayısal yöntem

İzin	Sayı
r (read)	4
w (write)	2
x (execute)	1

```
chmod 755 dosya.sh # 7 = rwx (4+2+1) → kullanıcı # 5 = r-x (4+1) → grup # 5 = r-x (4+1) → diğerleri
```

Sahiplik Değiştirme

```
# 1 Kullanıcı değiştirme chown ali dosya.txt # 2 Kullanıcı + Grup değiştirme chown ali:ogrenci dosya.txt # 3 Sadece grup değiştirme chown :ogrenci dosya.txt # 4 Klasör ve içindekiler için (recursive) chown -R ali:ogrenci /home/proje # 5 Numara ile değiştirme chown 620:100 deneme.txt
```

Wildcards

Wildcard	Açıklama	Örnek
*	Sıfır veya daha fazla karakter	`ls *.txt` # txt dosyalarını listele
?	Tek karakter	`ls ?.txt` # Sadece tek karakterli dosya isimleri
[]	Köşeli parantez içerisindeki karakterlerden herhangi biri	`ls file[1-3].txt` # file1.txt, file2.txt, file3.txt
[!]	Hariç tutma	`ls file[!0-9].txt` # Son karakteri rakam olmayan
{}	Küme parantezi ile seçenekler	`ls {a,b,c}.txt` # a.txt, b.txt, c.txt

```
# Örnekler ls [A-Z]* # Büyük harfle başlayan dizinler ls D* # "D" ile başlayan her şey ls -d D* # Sadece dizin isimlerini listeler ls -d */ # Tüm dizinleri listele ls *.txt 2>/dev/null # Hata mesajlarını gizle
```

expr (evaluate expressions) Komutu

Dikkat: Boşluklara dikkat edilerek kullanılmalı

```
# Toplama expr 5 + 3 # Çıktı: 8 # Çıkarma expr 10 - 4 # Çıktı: 6 # Çarpma (dikkat: * escape edilmeli) expr 6 \* 7 # Çıktı: 42 # Bölme (integer division) expr 15 / 3 # Çıktı: 5 # Mod (kalan) expr 17 % 3 # Çıktı: 2 # Değişkenlerle a=10 b=3 expr $a + $b # String işlemleri expr length "Bugra" # Çıktı: 5 expr substr "BugraChat" 2 3 # Çıktı: ugr expr index "Linux" n # Çıktı: 3
```

Girdilerin ve Çıktıların Yönlendirilmesi

Yönlendirme	Temsil	Açıklama
stdin	0	Standart girdi (klavyeden gelen veri)
stdout	1	Standart çıktı (ekrana yazılan normal çıktı)
stderr	2	Standart hata (hata mesajları)

```
# Çıktıyı dosyaya yönlendirme ls > cikti.txt # Dosyayı oluştur/üzerine yaz echo "Merhaba" >> cikti2.txt # Dosyanın sonuna ekle # Girdiyi dosyadan alma wc -l < metin.txt # Birden çok dosyayı birleştirme cat cikti.txt cikti2.txt >> yenicikti.txt
```

REGEX (Düzenli Deyimler)

Karakter	Açıklama
^	Satır başı
$	Satır sonu
.	Herhangi bir karakter
*****	Önceki karakterden sıfır veya daha fazla
+	Önceki karakterden bir veya daha fazla
?	Önceki karakterden sıfır veya bir tane
[abc]	a, b veya c karakterlerinden biri
[^abc]	a, b, c dışındaki karakterler
\{n\}	Tam n tane
\{n,\}	En az n tane

```
# Örnekler ^a # Satır başında a ile başlayan y$ # Satır sonunda y ile biten ^K.*M$ # K ile başlayıp M ile biten ti?ren # "tren" veya "tiren" ^zzz # Satır başında 3 adet z ^k\{2,\} # Satır başında en az 2 adet k
```

grep Komutu (Global Regular Expressions Print)

```
# Temel kullanım grep "^A" notlar.txt # A ile başlayan satırlar grep -i "linux" dosya.txt # Büyük/küçük harf duyarsız grep -v "error" log.txt # "error" geçmeyen satırlar grep -n "main" program.c # Satır numarası ile grep -o "error" log.txt # Sadece eşleşen kısmı grep -c "error" log.txt # Eşleşen satır sayısı # Extended grep (egrep veya grep -E) grep -E "+" # + operatörü için egrep "linux|unix" # VEYA operatörü
```

cut Komutu

```
# Sütun ayırıcı ile cut -d',' -f1,3 data.csv # Virgülle ayrılmış 1. ve 3. alanlar cut -d':' -f1 /etc/passwd # : ile ayrılmış 1. alan cut -d',' -f2-4 file.csv # 2'den 4'e kadar cut -d',' -f3- file.csv # 3'ten sonra tüm alanlar # Karakter pozisyonu ile cut -c1-5 file.txt # İlk 5 karakter cut -c3-10 file.txt # 3. karakterden 10. karaktere cut -c1,3,5 file.txt # 1, 3, 5. karakterler
```

sort Komutu

```
sort file.txt # Alfabetik sıralama sort -r names.txt # Ters sıralama (Z'den A'ya) sort -n file.txt # Sayısal sıralama sort -k2 data.txt # 2. alana göre sırala # ASCII tablosu kullanmak için export LC_ALL=C sort file.txt
```

find Komutu

```
# Temel arama find . # Geçerli dizinde tüm dosyaları listele find /home # /home dizininde tüm dosyaları ara find . -name "test.txt" # Belirli isimde dosya ara find /home /var /tmp -name "*.log" # Birden fazla dizinde ara # İsim araması find . -name "config.txt" # Tam isim eşleşmesi (duyarlı) find . -iname "CONFIG.TXT" # Büyük/küçük harf duyarsız find . -name "*.txt" # Wildcard kullanımı # Tip bazlı arama find . -type f -name deneme.txt # Dosya ara find . -type d -name Desktop # Dizin ara # Boyut bazlı arama find . -size 100 # Tam olarak 100 byte
```

wc (word count) Komutu

```
wc dosya.txt # Satır, kelime, karakter sayısı wc -l dosya.txt # Sadece satır sayısı wc -w dosya.txt # Sadece kelime sayısı wc -c dosya.txt # Sadece karakter sayısı
```

Girdi Yöntemi Farkları:

- `wc -l deneme.txt` → 16 deneme.txt (dosya adı gösterir)
- `wc -l < deneme.txt` → 16 (sadece sayı)
- `cat deneme.txt | wc -l` → 16 (sadece sayı)

Boru (Pipe) İşlemleri

Bir komutun çıktısını başka bir komuta yönlendirmek için **|** kullanılır.

```
komut1 | komut2 | komut3 # Örnekler ls | grep "txt" # txt geçen dosyaları filtreler cat dosya.txt | grep "error" | wc -l # error geçen satır sayısı ls -l | awk '{ print $9, $5 }' # Dosya adı ve boyutu
```

Editör (Metin Düzenleyicisi)

vim

```
vim deneme.cpp # Dosyayı aç/oluştur
```

vim Modları

- **Escape mod (Normal mod):** Çıkmak için Esc tuşu
- **Insert mod (Düzenleme modu):** Metin yazma modu

vim Komutları

Komut	Açıklama
`:q`	Çık
`:w`	Kaydet
`:wq`	Kaydet ve çık
`:q!`	Kaydetmeden çık
`:x`	Değişiklik varsa kaydet ve çık
`yy`	Satırı kopyala
`p`	Yapıştır

`/kelime`	Kelime ara (ileri)
`?kelime`	Kelime ara (geri)
`n`	Aynı yönde ara
`N`	Zıt yönde ara
`:set nu`	Satır numaralarını göster
`:set nonu`	Satır numaralarını gizle
`Ctrl+G`	Dosya hakkında bilgi

vim Dosya Açma Seçenekleri

```
vim dosyaAdi # Basit dosya açma vim +satirNo dosyaAdi # İmleci belirtilen satırda konumlandır vim +/kelime dosyaAdi # İmleci kelime geçen ilk satırda konumlandır
```

nano

^ klavyedeki Ctrl tuşunu temsil eder.

Tuş	Açıklama
`^O`	Dosyayı kaydet
`^X`	Programdan çık
`^G`	Yardım ekranı aç

Kabuk Programlama

Basit Bash Script

```
#!/bin/bash # Bu bir yorum satırı echo "Merhaba Dünya!"
```

Script Çalıştırma

```
# Çalıştırma izni ver chmod u+x deneme.sh # Çalıştır ./deneme.sh # veya bash deneme.sh # Farklı dizinden /home/bugor/scripts/deneme.sh
```

Değişkenler

```
#!/bin/bash # Değişken tanımlarken atama ifadesinde boşluk olmamalıdır isim="Ahmet" yas=25 echo "Merhaba $isim, yaşınız $yas"
```

Klavyeden Veri Girme

```
#!/bin/bash echo "Adınızı giriniz:" read isim echo "Merhaba $isim!"
```

printf Komutu

```
#!/bin/bash # Format string ile printf "Merhaba %s!\n" "Ahmet" # Çıktı: Merhaba Ahmet! printf "Yaş: %d\n" 25 # Çıktı: Yaş: 25 printf "%f\n" 5 # Çıktı: 5.000000 printf "%10d\n" 11 # 10 karakterlik alanda sağa yasla printf "%-10d %-10d\n" 11 12 # 10 karakterlik alanda sola yasla printf "%10.5f\n" 5.3 # 10 alan, 5 ondalık basamak
```

Belirteç	Anlamı
%d	Tam sayı (decimal)
%f	Ondalık sayı
%.2f	Ondalık sayı, 2 basamak
%s	String
%c	Tek karakter
%e	Bilimsel gösterim (exponential)
%x	Onaltılık (hexadecimal)

Aritmetik İşlemler

```
#!/bin/bash a=10 b=3 ((toplam=a + b)) # 1. atama işlemi fark=$((a - b)) # 2. atama işlemi carpim=$((a * b)) let bolum=a/b; # 3. atama işlemi kalan=$((a % b)) # Mod (kalan) us=$((a ** b)) # Üs alma printf "Toplam: %d\n" $toplam # 13 printf "Fark: %d\n" $fark # 7 printf "Çarpım: %d\n" $carpim # 30 printf "Bölüm: %d\n" $bolum # 3 printf "Kalan: %d\n" $kalan # 1 printf "Üs: %d\n" $us # 1000
```

bc Komutu (Basic Calculator)

Komut	Çıktı	Açıklama
`echo "57+43" | bc`	100	Toplama işlemi
`echo "57-43" | bc`	14	Çıkarma işlemi
`echo "57*43" | bc`	2451	Çarpma işlemi
`echo "scale=25;57/43" | bc`	1.3255813953488372093023255	scale → virgülden sonra basamak sayısı
`echo "scale=30;sqrt(2)" | bc`	1.414213562373095048801688724029	sqrt() → karekök hesaplar
`echo "6^6^6" | bc`	Çok büyük sayı	^ → üs alma (power)
`echo "obase=16;255" | bc`	FF	obase → çıktı tabanı (16'lık)
`echo "obase=2;12" | bc`	1100	obase=2 → binary çıktı
`echo "ibase=2;obase=A;10" | bc`	2	ibase → girdi tabanı

test Komutu

test komutu bir karşılaştırma ifadesinin sonucunu öğrenmek için kullanılır. $? ile son komutun sonucu öğrenilebilir.

```
#!/bin/bash # test komutu ile test 5 -eq 5 echo $? # 0 (başarılı/doğru) test 5 -eq 3 echo $? # 1 (başarısız/yanlış) # Köşeli parantez ile (aynı işlev) [ 5 -eq 5 ] echo $? # 0 [ 5 -eq 3 ] echo $? # 1
```

Karşılaştırma Operatörleri

Aritmetik İşlemler	Açıklama	Karakter Dizileri	Açıklama	Dosya/Dizin İşlemleri	Açıklama
`-eq`	Eşit Mi?	`=`	Eşit Mi?	`-e`	Dosya/Dizin Mevcut Mu?
`-ne`	Eşit Değil Mi?	`!=`	Eşit Değil Mi?	`-f`	Dosya Mı?
`-gt`	Büyük Mü?			`-d`	Dizin Mi?
`-ge`	Büyük Eşit Mi?			`-r`	Dosya Okunabilir Mi?
`-lt`	Küçük Mü?			`-w`	Dosya Yazılabilir Mi?
`-le`	Küçük Eşit Mi?			`-x`	Dosya Çalıştırılabilir Mi?

Şartlı Deyimler (if, elif ve else)

```
if [ koşul ]; then # koşul doğruysa çalışacak komutlar elif [ başka_koşul ]; then # ilk koşul yanlış, bu koşul doğruysa çalışır else # yukarıdaki hiçbir koşul doğru değilse burası çalışır fi
```

```
#!/bin/bash sayi=10 if [ $sayi -gt 0 ]; then echo "Sayı pozitiftir." elif [ $sayi -lt 0 ]; then echo "Sayı negatiftir." else echo "Sayı sıfırdır." fi
```

Mantıksal Operatörler

```
# Mantıksal VE if [ condition1 -a condition2 ] if [ condition1 ] && [ condition2 ] if [[ condition1 && condition2 ]] # Mantıksal VEYA if [ condition1 -o condition2 ] if [ condition1 ] || [ condition2 ] if [[ condition1 || condition2 ]]
```

Case Yapısı

```
case değişken in pattern1) # komutlar ;; pattern2) # komutlar ;; *) # varsayılan durum ;; esac
```

```
#!/bin/bash echo "Bir harf girin (a, b, c): " read harf case $harf in a) echo "A harfini seçtiniz" ;; b) echo "B harfini seçtiniz" ;; c) echo "C harfini seçtiniz" ;; *) echo "Geçersiz seçim!" ;; esac
```

Döngüler

While Döngüsü

```
while [ koşul ]; do # komutlar done
```

For Döngüsü

```
# Kullanım 1 for değişken in liste; do # komutlar done # Kullanım 2
for((ifade1;ifade2;ifade3)) do komutlar done
```

```
# Örnekler - Hepsi aynı for i in 1 2 3 4 5 6 7 8 9 10 for((i=1;i<=10;i=i+1)) for i in
{1..10} # Brace Expansion for i in `seq 1 10` # sequence of numbers for i in $(seq 1 10) #
sequence of numbers do echo $i done
```

Kabuk Programı Argümanları

Parametre	Açıklama
`$0`	Script adı
`$1`	Kabuk programına aktarılan 1. argüman
`$2`	Kabuk programına aktarılan 2. argüman
`$#`	Kaç argüman gönderildi
`$@`	Tüm argümanlar (her biri ayrı)
`$*`	Tüm argümanlar (tek string olarak)
`$`	Script'in PID'si (process ID)
`$?`	Son çalıştırılan komutun çıkış durumu

```
./Argumanlar.sh Bugor Etkingames # Çıktı: # Kabuk Programinizin Adi: ./Argumanlar.sh #
Kabuk Programinizin Tum Argumanlari: Bugor Etkingames # Kabuk Programinizin Arguman Sayisi:
2 # Kabuk Programinizin 1. Argumani: Bugor # Kabuk Programinizin 2. Argumani: Etkingames
```

Sayısı Bilinmeyen Argümanlara Erişim

```
#!/bin/bash # Yöntem 1 for i in "$@" do echo "$i" done # Yöntem 2 for((i=1;i<=$#;i++)) do
echo "${!i}" # indirect expansion done
```

"$*" ve "$@" Farkı:

- **"$*"** → Tüm argümanları tek bir string haline getirir
- **"$@"** → Tüm argümanları ayrı ayrı stringler halinde korur

Bir Dosyayı Satır Satır Okumak

```
#!/bin/bash # Yöntem 1 cat $1 | while read line do echo "$line" done # Yöntem 2 while read line do echo "$line" done < $1
```

Kabuk Fonksiyonları

```
# Yöntem 1 function_name() { # komutlar } # Yöntem 2 function function_name { # komutlar }
```

```
#!/bin/bash selamla() { echo "Merhaba $1" # $1 ilk parametre } selamla "Deniz" selamla "Ahmet"
```

```
#!/bin/bash kontrolEt() { if [ -f "$1" ]; then echo "Dosya bulundu: $1" return 0 # 0 → başarılı else echo "Dosya yok: $1" return 1 # 1 → hata fi } kontrolEt "deneme.txt" if [ $? -eq 0 ]; then echo "İşlem başarılı." else echo "İşlem başarısız." fi
```

IFS (Input/Internal Field Separator)

Kabuk bu değişkeni kullanarak girdiyi nasıl parçalayacağını belirler.

Varsayılan IFS içeriği:

- Boşluk (space)
- Tab (\t)
- Yeni satır (\n)

```
# String'i kelimelere ayırma text="apple banana cherry" for word in $text; do echo "Kelime: $word" done # Virgülle ayrılmış değerler için IFS ayarlama IFS=',' data="elma,armut,kiraz,üzüm" for item in $data; do echo "Meyve: $item" done # IFS'yi eski haline getir IFS= \t\n'
```

```
# IP adresi parse etme örneği parse_ip() { local ip="192.168.1.100" local old_ifs="$IFS" IFS='.' set -- $ip # Pozisyonel parametrelere ata echo "1. oktet: $1" echo "2. oktet: $2" echo "3. oktet: $3" echo "4. oktet: $4" IFS="$old_ifs" }
```

DİZİLER

Not: Dizi indisi 0'dan başlar ve boyutlarla ilgili maksimum kısıtlama yoktur.

```
# Yöntem 1: Doğrudan atama fruits[0]="elma" fruits[1]="armut" fruits[2]="kiraz" # Yöntem 2: Parantez ile fruits=("elma" "armut" "kiraz" "üzüm") # Boş dizi deklarasyonu fruits=()
```

Diziye Erişim

```
# Tek elemana erişim echo ${fruits[0]} # elma echo $fruits # elma (ilk element) echo ${fruits} # elma echo ${fruits[1]} # armut # Tüm elemanlara erişim echo ${fruits[@]} # elma armut kiraz echo ${fruits[*]} # elma armut kiraz # Son element (Bash 4.3+) echo ${fruits[-1]} # son element
```

```
Linux=("Debian" "Red Hat" "Ubuntu" "Suse" "Fedora") # Tırnak kullanmadan for i in ${Linux[@]}; do echo $i # Red ve Hat ayrı olarak alır done echo "BU DA İKİNCİ SATIR----------" # Tırnak kullanarak for i in "${Linux[@]}"; do echo "$i" # "Red Hat" aynen korunur done
```

Dizi Dilimleme (Array Slicing)

```
numbers=(1 2 3 4 5 6 7 8 9 10) # İndeks 2'den başlayarak 3 eleman echo ${numbers[@]:2:3} # 3 4 5 # İndeks 5'ten sonuna kadar echo ${numbers[@]:5} # 6 7 8 9 10 # Sondan 3 eleman echo ${numbers[@]: -3} # 8 9 10 # Belirli elemandan substring Linux=("Debian" "Red Hat" "Ubuntu" "Suse" "Fedora") echo ${Linux[2]:0:4} # Ubun (2. indeksteki elemanın ilk 4 karakteri)
```

Dizi Bilgileri

```
# Bir dizinin eleman sayısı (boyutu) echo ${#Linux[@]} # 5 # Belirli elemanın boyutu echo ${#Linux[2]} # 6 (Ubuntu'nun karakter sayısı)
```

Diziye Eleman Ekleme/Silme

```
fruits=("elma" "armut") # Sona ekleme fruits+=("kiraz") fruits[${#fruits[@]}]="üzüm" Linux=("${Linux[@]}" "Knoppix") # Başa ekleme Linux=("Kali Linux" "${Linux[@]}") # Belirli indekse ekleme fruits[10]="muz" # İlk elemanı silme Linux=("${Linux[@]:1}") # Son elemanı silme Linux=("${Linux[@]:0:$((${#Linux[@]}-1))}") # Aradan bir elemanı silme pos=3 Linux=("${Linux[@]:0:$pos}" "${Linux[@]:$((pos + 1))}") # Dizinin kopyalanması LinuxYedek=("${Linux[@]}")
```

Dizilerde Döngüler

```
fruits=("elma" "armut" "kiraz" "üzüm") # Yöntem 1: Elemanları döngüde for fruit in "${fruits[@]}"; do echo "Meyve: $fruit" done # Yöntem 2: İndeks ile döngü for i in "${!fruits[@]}"; do echo "İndeks $i: ${fruits[i]}" done # Yöntem 3: C tarzı döngü for ((i=0; i<${#fruits[@]}; i++)); do echo "Eleman $i: ${fruits[i]}" done
```

Karakter Dizileri ile İlgili İşlemler

String'in Uzunluğunun Hesabı

```
# Yöntem 1 isim="Bugor" echo ${#isim} # Çıktı: 5 # Yöntem 2 echo "$isim" | wc -c # Çıktı: 6 (newline dahil) echo -n "$isim" | wc -c # Çıktı: 5 (newline hariç) # Yöntem 3 echo `expr length "$isim"` # Çıktı: 5
```

Dosya Adı ve Uzantı İşlemleri

```
# basename komutu ile dosyaAdi="Uygulama1.sh" uzantisizDosyaAdi=$(basename $dosyaAdi .sh) echo $uzantisizDosyaAdi # Çıktı: Uygulama1 # cut komutu ile (tek nokta varsa) dosyaAdi="Uygulama1.sh" uzantisizDosyaAdi=$(echo "$dosyaAdi"|cut -d '.' -f 1) dosyaUzantisi=`echo "$dosyaAdi"|cut -d '.' -f 2` echo $uzantisizDosyaAdi # Çıktı: Uygulama1 echo $dosyaUzantisi # Çıktı: sh
```

rev (reverse) Komutu

```
string1="ey edip adanada pide ye" string2=$(echo "$string1"|rev) echo $string2 # ey edip adanada pide ye test "$string1" = "$string2"; echo $? # Palindrom testi: 0
```

Çoklu Nokta Karakteri İçeren Dosyalar

```
dosyaAdi="Uygulama1.sh.txt.jpeg" uzantisizDosyaAdi=$(echo "$dosyaAdi"|rev) echo $uzantisizDosyaAdi # gepj.txt.hs.1amalugyU uzantisizDosyaAdi=$(echo "$dosyaAdi"|rev|cut -d '.' -f 2-) echo $uzantisizDosyaAdi # txt.hs.1amalugyU uzantisizDosyaAdi=$(echo "$dosyaAdi"|rev|cut -d '.' -f 2-|rev) echo $uzantisizDosyaAdi # Uygulama1.sh.txt
```

AWK UTILITY

AWK kabuk programlamada çok güçlü bir metin işleme aracıdır.

AWK Temel Yapısı

```
awk 'pattern { action }' dosya
```

AWK Kullanım Şekilleri

```
# 1. Pattern + Action awk '/error/ {print $1, $2}' logfile # "error" geçen satırların 1. ve 2. sütunu # 2. Sadece Action awk '{print $1}' data.txt # Tüm satırların sadece 1. sütunu # 3. Sadece Pattern awk '/error/' logfile # "error" geçen satırların tamamı
```

AWK Alan (Field) Yapısı

AWK girdiyi kayıtlara ve alanlara böler:

- Her satır bir kayıttır (varsayılan olarak)
- Her kayıt, özel bir karakterle (varsayılan olarak boşluk) ayrılmış alanlara bölünür
- $0 → tüm satır

- $1, $2, $3... → 1., 2., 3. alan

```
# Örnek data: # George Jones Admin # Anthony Smith Accounting awk '{ print $1, $3 }'
dosya.txt # Çıktı: # George Admin # Anthony Accounting
```

AWK Alan Ayırıcısı

```
# Varsayılan boşluk yerine : kullan awk -F':' '/small000/{print $5}' /etc/passwd # Virgül
ayırıcısı ile awk -F',' '{print $1, $3}' data.csv # Script dosyasından çalıştır awk -f
awkprog.awk students
```

AWK İlişkisel Operatörler

```
# 1. sütunu kontrol et, 10'dan büyük olan satırları yazdır awk '$1 > 10 { print $0 }'
dosya.txt # 2. sütun "apple" olan satırları yazdır awk '$2 == "apple" { print $0 }'
dosya.txt # 3. sütun 50 ile 100 arasında olan satırları yazdır awk '$3 >= 50 && $3 <= 100 {
print $0 }' dosya.txt # 'Win32' İÇERMEYEN kayıtları yazdır awk '!/Win32/' log.txt # 4. alan
6'dan küçükse awk '$4<6{print $1,$3,$4,$5}' kim
```

AWK Pattern Matching

```
/^$/{print "Bu satır boştur"} # Boş satırlar $1 ~ /num/{print "Satır $1'de num içeriyor"} #
$1'de num geçenler $2 !~ /num/{print "Satır $2'de num içermiyor"} # $2'de num geçmeyenler
!/num/{print "Satır num içermiyor"} # Satırda num geçmeyenler /[0-9]+$/{print "Tam sayı
son:",$0} # Rakamla bitenler /^[A-Z]/{print "Büyük harfle başlar"} # Büyük harfle
başlayanlar
```

AWK Değişkenleri

Değişken	Anlamı
`NF`	Mevcut satırdaki alan (kolon) sayısı
`NR`	Şu ana kadar işlenen kayıt (satır) sayısı
`FILENAME`	Şu anda işlenen dosyanın adı
`FS`	Alan ayracı (default: boşluk veya tab)
`OFS`	Çıktı alan ayracı (default: boşluk)
`RS`	Kayıt ayracı (default: yeni satır \n)
`ORS`	Çıktı kayıt ayracı (default: \n)

AWK Fonksiyonları

Fonksiyon	Açıklama
`sqrt(x)`	x sayısının karekökünü döndürür
`sin(x)`	x radyan cinsinden sinüs değerini döndürür
`index(s1,s2)`	s1 string'i içinde s2'nin pozisyonu

`substr(s,m,n)`	s'nin m'den başlayan n karakterli alt dizisi
`tolower(s)`	Dizeyi küçük harfe dönüştürür
`log(x)`	x sayısının doğal logaritmasını döndürür
`int(x)`	x sayısını tam sayıya yuvarlar (keser)
`rand()`	0 ile 1 arasında rastgele sayı üretir
`printf`	C gibi biçimlendirilmiş printf yazdır

AWK printf Format Belirteçleri

Belirteç	Anlamı
%d	Tam sayı (decimal)
%f	Ondalık sayı
%.2f	Ondalık sayı, 2 basamak
%s	String
%c	Tek karakter
%e	Bilimsel gösterim (exponential)
%x	Onaltılık (hexadecimal)

```
# Üçüncü alanı iki ondalık basamaklı göster awk '{printf("%.2f\n",$3)}' file # Locale ayarı export LC_NUMERIC="en_US.UTF-8"
```

AWK Programları

```
BEGIN{ # Dosya okunmadan önce çalışır # Değişken başlatma, FS tanımlama, başlık yazdırma } { # Her satır için çalışır } pattern{ # Pattern eşleşen satırlar için çalışır } END{ # Dosya tamamen okunduktan sonra çalışır # Toplam, ortalama, sayma, özet raporlar }
```

```
# Örnek AWK programı awk ' BEGIN { print "Ad | Yaş" print "-----------" } { print $1, $2 } END { print "Toplam kayıt sayısı:", NR }' dosya.txt
```

AWK Program Blokları:

- **BEGIN bloğu** → başlık yazdırılır
- **Her satır** → $1 ve $2 yazdırılır
- **END bloğu** → toplam kayıt sayısı yazdırılır (NR ile)

```
# Toplama örneği awk '{ sum += $1 } END { print "Toplam:", sum }' dosya.txt # Science öğrencilerini filtrele awk ' /Science/{print "Science stu:",$1,$2} /CompSci/{print "Computing stu:",$1,$2} ' students
```

Linux Komutları Referansı

Bu PDF referans belgesi, temel Linux komutlarından ileri kabuk programlamaya kadar kapsamlı bir rehber sunar.

Tarayıcınızın yazdır işlevini kullanarak PDF olarak kaydedin (Ctrl+P → PDF olarak kaydet)