

# Linux



## Linux İşletim Sistemi

Son güncelleme : 12/2025

### İçindekiler

- [Komut Satırı](#)
- [Metin İşlemleri](#)
- [Gelişmiş Metin İşlemleri](#)
- 

July  
17

### Tarihçe

Linux'un nasıl ortaya çıktığını öğrenmek için, 1969'a, Ken Thompson ve Dennis Ritchie'nin Bell Laboratuvarlarında UNIX işletim sistemini geliştirdikleri zamana dönelim. Daha sonra taşınabilirliği artırmak için C dilinde yeniden yazıldı ve sonunda yaygın olarak kullanılan bir işletim sistemi haline geldi.

Yaklaşık on yıl sonra, Richard Stallman, GNU (GNU, UNIX Değildir) projesi üzerinde çalışmaya başladı. Bu proje kapsamında Hurd adında bir GNU çekirdeği geliştirildi, ancak maalesef asla tamamlanmadı. Bunun sonucu olarak, özgür yazılım lisansı olan GNU Genel Kamu Lisansı (GPL) de oluşturuldu.

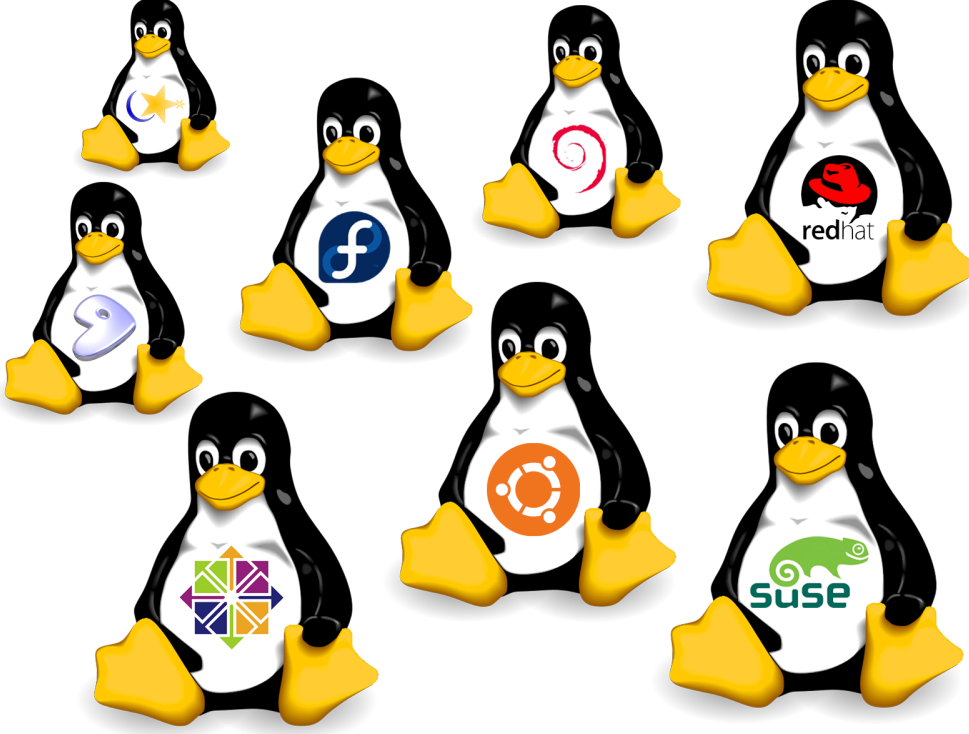
Çekirdek, işletim sisteminin en önemli parçasıdır. Donanımın yazılımla iletişim kurmasını sağlar. Biz kullanıcılar sistemde bulunan yazılımlar ile çekirdeğe emirler veririz, çekirdekde donanıma o işi yaptırır. Çekirdeğin sisteminizde olup biten her şeyi kontrol eder.

Bu dönemde BSD, MINIX vb. gibi diğer UNIX benzeri sistemler geliştirildi. Ancak, tüm bu UNIX benzeri sistemlerin ortak noktası, tek bir çekirdek eksikliği idi.

Ardından 1991'de Linus Torvalds adında genç bir adam, bugün bildiğimiz Linux çekirdeğini geliştirmeye başladı.

Sonuç olarak GNU yazılımları ve linux çekirdeğinin birleşimi ile ortaya açık kaynak, özgür bir işletim sistemi çıkmıştır.

# 🔥 Linux Dağıtımları



Bir Linux sistemi üç ana bölümden oluşur:

- **Donanım:** Bu, sisteminizin çalıştığı tüm donanımları, bellek, CPU, diskler vb. içerir.
- **Linux Çekirdeği:** Yukarıda belirttiğimiz gibi, çekirdek işletim sisteminin merkezidir. Donanımı yönetir ve sistemle nasıl etkileşim kuracağını söyler.
- **Kullanıcı Alanı:** Bu, bizler gibi kullanıcıların çeşitli yazımlar ile doğrudan sistemle etkileşim kuracağı yerdir.

Seçilebilecek birçok Linux dağıtımı vardır, sadece en popüler seçeneklere göz atacağız.

## ► Debian Dağıtımı

### Genel Bakış

Debian, tamamen özgür ve açık kaynaklı yazılımlardan oluşan bir işletim sistemidir. Geniş çapta bilinen ve 20 yılı aşkın süredir geliştirilmektedir. Kullanabileceğiniz üç ana sürümü vardır: Stable ( kararlı), Testing (test ) ve Unstable (kararsız).

### Sürümler

**Stable:** Genel olarak kullanılması iyi olan bir sürüm. Sisteminizde kararlılık ve güvenlik öncelikliyse bu sürümü tercih edebilirsiniz.

**Testing ve Unstable:** Sürekli güncelleme (rolling release) alan dallardır. Bu, bu dallardaki aşamalı değişikliklerin sonunda Stable sürümüne dahil olacağı anlamına gelir. Örneğin, Windows XP'den Windows 10'a yükseltme yapmak istiyorsanız, tam bir Windows 10 kurulumu yapmanız gerekir. Ancak Testing sürümünü kullanıyorsanız, tam bir kurulum yapmadan bir sonraki işletim sistemi sürümü olana kadar otomatik olarak güncellemeleri alacaksınız.

### Paket Yönetimi

Debian, kendi paket yönetim (APT) araçlarını kullanır. Her Linux dağıtımı paketleri farklı şekilde kurar, yönetir ve farklı paket yönetim araçları kullanır.

### Yapılandırılabilirlik

Debian en son güncellemeleri almasa da son derece kararlıdır. İyi bir "temel" işletim sistemi arıyorsanız, bu sizin için doğru tercih olabilir.

### **Kullanım Alanları**

Debian, her platform için genel olarak harika bir işletim sistemidir.

## **► Red Hat Enterprise Linux Dağıtımı**

### **Genel Bakış**

Red Hat Enterprise Linux, genellikle RHEL olarak adlandırılır ve Red Hat tarafından geliştirilir. RHEL, ücretsiz yeniden dağıtımı kısıtlamak için katı kurallara sahiptir, ancak yine de kaynak kodunu ücretsiz olarak sağlar.

### **Paket Yönetimi**

RHEL, Debian'dan farklı bir paket yöneticisi olan RPM paket yöneticisini kullanır.

### **Yapılandırılabilirlik**

RHEL tabanlı işletim sistemleri, Debian tabanlı işletim sistemlerinden biraz farklılık gösterecek, özellikle paket yönetiminde daha belirgin olacaktır.

### **Kullanım Alanları**

Adından da anlaşılacağı gibi, çoğunlukla kurumsal alanda kullanılır, bu nedenle sağlam bir sunucu işletim sistemine ihtiyacınız varsa bu iyi bir tercih olacaktır.

## **► Ubuntu Dağıtımı**

### **Genel Bakış**

Kişisel bilgisayarlar için en popüler Linux dağıtımlarından biri Ubuntu'dur. Ubuntu ayrıca varsayılan olarak kendi masaüstü ortamı yöneticisi Unity'yi yayınlar.

### **Paket Yönetimi**

Ubuntu, Canonical tarafından geliştirilen Debian tabanlı bir işletim sistemidir. Dolayısıyla temel olarak Debian paket yönetim sistemini kullanır.

### **Yapılandırılabilirlik**

Linux'a başlamak isteyen yeni başlayanlar için Ubuntu harika bir seçimdir. Ubuntu, kullanıcı dostu arayüzü ve geniş çapta benimsenilmesine yol açan kullanım kolaylığı sunar. Yaygın olarak kullanılmakta ve desteklenmektedir ve kullanılabilirlik açısından diğer işletim sistemleri gibi OSX ve Windows'a en çok benzerlik gösterir.

### **Kullanım Alanları**

Masaüstü, dizüstü bilgisayar ve sunucu dahil olmak üzere her platform için uygundur.

## **► Fedora Dağıtımı**

### **Genel Bakış**

Red Hat tarafından desteklenen Fedora Projesi, açık kaynaklı ve ücretsiz yazılımları içeren, topluluk odaklı bir projedir. Red Hat Enterprise Linux, Fedora'dan dallanarak geliştirilir, bu nedenle Fedora'yı bir upstream RHEL işletim sistemi olarak düşünebilirsiniz. Sonuç olarak, Red Hat Enterprise Linux, kapsamlı test ve kalite güvencesinden sonra Fedora'dan güncellemeler alacaktır. Fedora'yı, Debian yerine Red Hat altyapısı kullanan bir Ubuntu eşdeğeri olarak düşünebilirsiniz.

### **Paket Yönetimi**

Fedora, Red Hat paket yöneticisini kullanır.

### **Yapılandırılabilirlik**

Red Hat tabanlı bir işletim sistemi kullanmak istiyorsanız, bu kullanıcı dostu bir versiyondur.

### **Kullanım Alanları**

Fedora, Red Hat tabanlı bir işletim sistemini fiyat etiketi olmadan kullanmak istiyorsanız harika bir seçimdir. Masaüstü ve dizüstü bilgisayarlar için önerilir.

## **► Linux Mint Dağıtımı**

### **Genel Bakış**

Linux Mint, Ubuntu tabanlı bir işletim sistemidir. Ubuntu'nun yazılım depolarını kullanır, böylece her iki dağıtımda da aynı paketler kullanılabilir. Ubuntu'dan daha hafif bir dağıtım tercih ediyorsanız, Linux Mint ilginizi çekebilir.

### **Paket Yönetimi**

Linux Mint, Ubuntu tabanlı olduğundan Debian paket yöneticisini kullanır.

### **Yapılandırılabilirlik**

Harika bir kullanıcı arayüzü sunar, yeni başlayanlar için uygundur ve Ubuntu'dan daha az gereksiz yazılım içerir.

### **Kullanım Alanları**

Masaüstü ve dizüstü bilgisayarlar için uygundur.

## **► Arch Linux Dağıtımı**

### **Genel Bakış**

Arch Linux, %100 topluluk tarafından yönetilen, hafif ve esnek bir Linux dağıtımdır. Debian'a benzer şekilde, Arch da sürekli güncelleme modelini (rolling release) kullanır, bu nedenle kademeli güncellemeler sonunda Stable (kararlı) sürüm haline gelir. Sistemi ve işlevlerini anlamak için gerçekten uygulamalı olarak öğrenmeniz gerekir, ancak karşılığında sisteminiz üzerinde tam ve eksiksiz kontrol elde edersiniz.

### **Paket Yönetimi**

Paketleri kurmak, güncellemek ve yönetmek için kendi paket yöneticisi Pacman'ı kullanır.

### **Yapılandırılabilirlik**

Hafif bir işletim sistemi istiyor ve Linux'u gerçekten anlamak istiyorsanız Arch'ı kullanın! Biraz öğrenme eğrisi olsa da, hardcore Linux kullanıcıları için harika bir seçimdir.

### **Kullanım Alanları**

Masaüstü ve dizüstü bilgisayarlar için uygundur. Ayrıca Raspberry Pi gibi küçük bir cihazınız varsa ve üzerine hafif bir işletim sistemi kurmanız gerekiyorsa, Arch'ı tercih edebilirsiniz.

## **► openSUSE Dağıtımı**

### **Genel Bakış**

openSUSE Linux, tüm dünyadaki Özgür ve Açık Kaynaklı Yazılım topluluğunun bir parçası olarak açık, şeffaf ve dostça bir şekilde birlikte çalışan openSUSE Projesi tarafından yaratılmıştır. openSUSE, halen çalışmakta olan ikinci en eski Linux dağıtımdır ve ödüllü SUSE Linux Enterprise ürünleriyle taban sistemini paylaşır.

### **Paket Yönetimi**

RPM paket yöneticisini kullanır.

### **Kullanılabilirlik**

openSUSE, yeni bir Linux kullanıcısı için harika bir seçimdir. Kullanımı kolay bir grafiksel kurulum/yönetim uygulaması (YaST) ve düzenli bir temel sistem sunar, kurcalamaya kolay açıktır. openSUSE, ister fotoğraflarınız, videolarınız, müzikleriniz ister kodunuz olsun, İnternet'in keyfini virüslerden/casus yazılımlardan endişe duymadan çıkarmanız ve yaratıcılığınızı ortaya koymanız için ihtiyacınız olan her şeyi içerir.

### Kullanım Alanları

openSUSE Leap, masaüstü PC ve dizüstü bilgisayarda kullanıma tamamen uygundur.

## Komut Satırı

### [Başa Dön](#)

## Kabuk (Shell)

Kabuk, temelde klavyenizden komutlarınızı alıp bunları işletim sistemine göndererek gerçekleştirilmesini sağlayan bir programdır. Daha önce bir GUI (grafiksel arayüz) kullandıysanız, "Terminal" veya "Konsol" gibi progralları görmüşsünüzdür. Bunlar sizin için bir kabuk başlatan programlardır.

Bu belgede kabuk programı bash (Bourne Again SHell) kullanacağız, hemen hemen tüm Linux dağıtımları varsayılan olarak bash kabuğunu kullanır. Ksh, zsh, tsch gibi başka kabuklar da mevcuttur, ancak en çok kullanılan kabuk programı bash'dir.

Genel görünümü (prompt) aşağıdaki gibidir.

```
kullanıcı_adı@bilgisayar_adı:su_anki_dizin  
ali@pc:/home/ali $
```

Promptun sonunda \$ sembolünü gördünüz mü? Farklı kabukların farklı promptları olacaktır, bizim durumumuzda \$ sembolü Bash, Bourne veya Korn kabuğunu kullanan normal bir kullanıcı içindir, komutu yazarken bu sembolü eklemeyin.

Basit bir komut olan echo ile başlayalım. echo komutu, metin argümanlarını ekrana yazdırır.

```
$ echo Merhaba Dünya
```

## pwd (Print Working Directory / Çalışma Dizini Yazdır)

Linux'ta her şey bir dosyadır, Linux'u derinlemesine öğrendikçe bunu anlayacaksınız, ancak şimdilik sadece bunu aklınızda bulundurun. Her dosya, hiyerarşik bir dizin ağacında organize edilir. Dosya sistemindeki ilk dizin, kök dizin olarak adlandırılır. Kök dizinde, daha fazla klasör ve dosya depolayabileceğiniz birçok klasör ve dosya bulunur.

Bu dosya ve dizinlerin konumları yollar olarak adlandırılır.

Dosya sisteminde gezinmek, tıpkı gerçek hayatta olduğu gibi, nerede olduğunuzu ve nereye gideceğinizi bilmeniz yararlıdır. Nerede olduğunuzu görmek için pwd komutunu kullanabilirsiniz, bu komut "çalışma dizinini yazdır" anlamına gelir ve yalnızca hangi dizinde olduğunuzu gösterir, yolun kök dizinden geldiğini unutmayın.

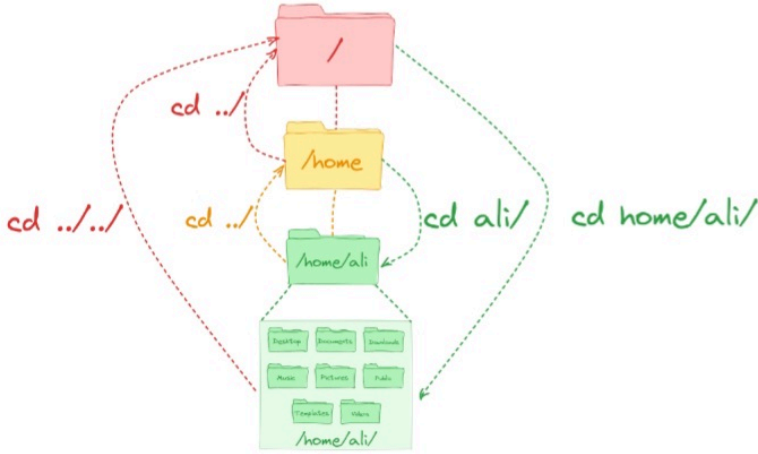
```
$ pwd
```

## cd (Change Directory / Dizin Değiştir)

Şimdi nerede olduğunuzu öğrendiğinize göre, dosya sisteminde biraz dolaşabileceğimize bakalım. Dosya sisteminde gezinmek için yolları kullanmamız gerektiğini unutmayın. Yol belirtmenin mutlak ve görelî olmak üzere iki farklı yolu vardır.

**Mutlak yol:** Bu, kök dizinden itibaren olan yoldur. Kök dizin en önemli dizindir. Kök dizin genellikle bir eğik çizgi "/" olarak gösterilir. Yolunuz her zaman "/" ile başladığında, kök dizinden başladığınız anlamına gelir. Örneğin, `/home/ali/Masaüstü`

**Görelî yol:** Bu, dosya sistemindeki bulunduğunuz konumdan itibaren olan yoldur. Eğer `/home/ali/Dökümanlar` konumunda olsaydım ve Dökümanlar içinde vergiler adında bir dizine gitmek isteseydim, `/home/ali/Dökümanlar/vergiler` gibi kök dizinden tüm yolu belirtmeme gerek yok, bunun yerine sadece `vergiler/` dizinine gidebilirim.



Artık yolların nasıl çalıştığını bildiğimize göre, istediğimiz dizine geçmemize yardımcı olacak bir şeye ihtiyacımız var. Neyse ki, bunu yapmak için `cd` "dizin değiştir" komutu kullanılır.

```
$ cd /home/ali/Pictures
```

Böylece şimdi dizin konumumu `/home/ali/Pictures` olarak değiştirdim.

Şimdi bu dizinden Hawaii adında bir klasörüm var, şu şekilde o klasöre gidebilirim:

```
ali@pc:/home/ali/Pictures $ cd Hawaii
```

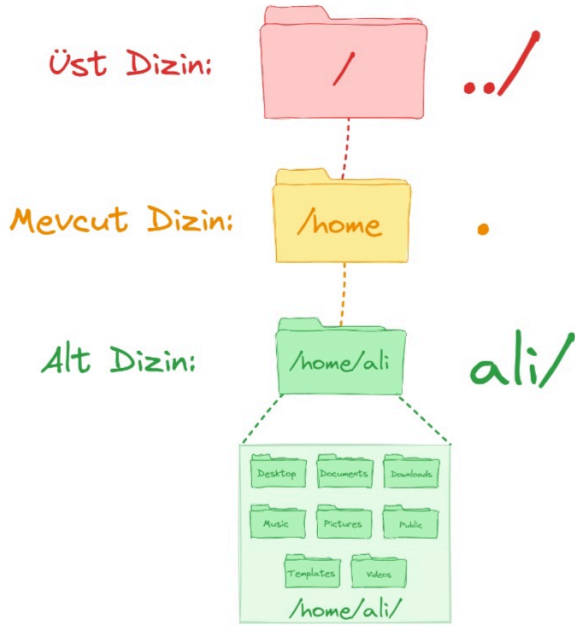
Sadece klasörün adını nasıl kullandığımı fark ettiniz mi? Çünkü zaten `/home/ali/Pictures` konumundaydım.

Her zaman mutlak ve görelî yollarla gezinmek oldukça yorucu olabilir, Neyse ki, size yardımcı olacak bazı kısayollar var.

- **.(geçerli dizin):** Şu anda bulunduğunuz dizindir.
- **..(üst dizin):** Sizi şu anki konumunuzun bir üst dizinine götürür.
- **~(ana dizin):** Bu dizin varsayılan olarak "ana dizininize" (`~/home/kullanıcı_adı`) gider.
- **-(önceki dizin):** Bu sizi az önce bulunduğunuz önceki dizine götürür.

Örnekler:

```
$ cd . # geçerli dizinde kal
$ cd .. # bir üst dizine git
$ cd ~ # ana dizine git
$ cd - # önceki dizine git
```



## ls (List Directories)

Dizin içeriklerini listelemek `ls` komutunu kullanabiliriz. `ls` komutu varsayılan olarak geçerli dizindeki dizinleri ve dosyaları listeler, ancak hangi dizinin dizinlerini listelemek istediğinizi belirtebilirsiniz.

```
$ ls
```

```
$ ls /home/ali
```

Ayrıca bir dizindeki tüm dosyaların görünmeyeceğini de unutmayın. Nokta ile başlayan dosya adları gizlidir, ancak bunları `ls` komutuyla görebilirsiniz ve `-a` (tümü için a) işaretini ekleyebilirsiniz.

```
$ ls -a
```

Bir başka `ls` işareti, `-l` uzun formatta ayrıntılı bir dosya listesi gösterir. Bu size ayrıntılı bilgi gösterecektir, soldan başlayarak: dosya izinleri, bağlantı sayısı, sahip adı, sahip grubu, dosya boyutu, son değişiklik zaman damgası ve dosya/dizin adı.

```
ali@pc:~$ ls -l

total 80

drwxr-x--- 7 ali ali 4096 Nov 20 16:37 Desktop
drwxr-x--- 2 ali ali 4096 Oct 19 10:46 Documents
```

```
drwxr-x--- 4 ali ali 4096 Nov 20 09:30 Downloads
drwxr-x--- 2 ali ali 4096 Oct 7 13:13 Music
drwxr-x--- 2 ali ali 4096 Sep 21 14:02 Pictures
drwxr-x--- 2 ali ali 4096 Jul 27 12:41 Public
drwxr-x--- 2 ali ali 4096 Jul 27 12:41 Templates
drwxr-x--- 2 ali ali 4096 Jul 27 12:41 Videos
```

## Sık kullanılan argümanlar

### -l

Uzun listeleme formatı (izinler, sahip, grup, boyut, tarih).

```
ls -l
```

### -h

Dosya boyutlarını insan okunabilir biçimde gösterir (KB, MB, GB). Genellikle `-l` ile birlikte kullanılır.

```
ls -lh
```

### -a

Gizli dosyaları da listeler (`.` ile başlayanlar).

```
ls -a
```

### -A

Gizli dosyaları listeler ancak `.` ve `..` hariç tutar.

```
ls -A
```

## Sıralama Seçenekleri

### -t

Dosyaları son değiştirilme zamanına göre sıralar.

```
ls -lt
```

### -S

Dosyaları boyutlarına göre sıralar.

```
ls -ls
```

### -r

Ters sıralama yapar.

```
ls -ltr
```

---

## Dosya Türleri ve Ayırt Etme

- **-F**

Dosya türünü sonuna ek işaretle belirtir:

- `/` → izin
- `*` → çalıştırılabilir dosya
- `@` → sembolik link

```
ls -F
```

### **--color=auto**

Dosya türlerine göre renklendirerek gösterir (çoğu dağıtımda varsayılan).

```
ls --color=auto
```

---

## Dizin ve Alt Dizin İşlemleri

### **-d**

Dizinin içeriğini değil, dizinin kendisini listeler.

```
ls -ld /etc
```

### **-R**

Alt dizinlerle birlikte recursive (özyinelemeli) listeleme yapar.

```
ls -R
```

---

## Zaman Bilgileri

### **-u**

Son erişim zamanına göre listeler.

```
ls -lu
```

### **-c**

Son durum değişikliği zamanına göre listeler.

```
ls -lc
```

---

## Yaygın Kullanım Kombinasyonları

- **ls -lah**  
Tüm dosyalar, detaylı liste, okunabilir boyutlar.
- **ls -ltrh**  
En eski dosyalar üstte olacak şekilde detaylı ve okunabilir liste.
- **ls -ld \*/**  
Sadece dizinleri uzun formatta listeler.

## Kısa Özet Tablosu

Argüman	Açıklama
-l	Detaylı liste
-a	Gizli dosyalar
-h	Okunabilir boyut
-t	Zamana göre sıralama
-r	Ters sıralama
-S	Boyuta göre sıralama
-R	Alt dizinlerle
-F	Dosya türü işareti
-d	Dizinin kendisini gösterir

## touch

Touch, yeni boş dosyalar oluşturmanıza olanak tanır.

```
$ touch <dosya>
```

Touch ayrıca mevcut dosya ve dizinlerde zaman damgalarını değiştirmek için kullanılır. Bir dosyada **ls -l** komutunu kullanın ve zaman damgasını not edin, ardından o dosyaya **touch** komutunu uygulayın, zaman damgası güncellenecektir.

## file

Linux'ta, dosya adlarının dosyanın içeriğini temsil etmesi gerekmez. Aslında GIF olmayan **komik.gif** adında bir dosya oluşturabilirsiniz. Bu onun bi GIF dosyası olduğu anlamına gelmez.

Bir dosyanın ne tür bir dosya olduğunu bulmak için **file** komutunu kullanabilirsiniz. Bu komut, dosyanın içeriğinin bir açıklamasını size gösterecektir.

```
$ file komik.gif
```

## cat

Dosya okumak için kullanılır. Bu komut, concatenate (birleştirmek) kelimesinin kısaltmasıdır, yalnızca dosya içeriğini görüntüleyemez, aynı zamanda birden fazla dosyayı birleştirebilir ve size çıktıları gösterebilir.

```
$ cat dogfile birdfile > newfile
```

Ancak büyük dosyaları görüntülemek için pek uygun değildir ve yalnızca kısa içerikler için kullanılır.

## less

Basit çıktılardan daha büyük metin dosyaları görüntüleyecekseniz, "az daha fazladır" (aslında benzer bir şey yapan `more` adında bir komut vardır). Metin, sayfa sayfa görüntülenir, böylece bir metin dosyasında sayfa sayfa gezinebilirsiniz.

Devam edin ve bir dosyanın içeriğine `less` komutu ile bakın. `less` komutundayken, dosyada gezinmek için diğer klavye komutlarını kullanabilirsiniz.

```
$ less /home/ali/Dökümanlar/metin1
```

`less` içinde gezinmek için aşağıdaki komutları kullanın:

- **q** - `less` programından çıkıp komut satırına geri dönmek için kullanılır.
- **Sayfa yukarı, Sayfa aşağı, Yukarı ve Aşağı okları** - Ok tuşları ve sayfa tuşlarını kullanarak gezinir.
- **g** - Metin dosyasının başına gitmek için kullanılır.
- **G** - Metin dosyasının sonuna gitmek için kullanılır.
- **/arama** - Metin belgesinin içinde belirli metinleri arayabilirsiniz. Aramak istediğiniz kelimelerin öncesine / işareti ekleyin.
- **h** - `less` programını kullanırken nasıl kullanılacağı hakkında biraz yardıma ihtiyacınız varsa, `h` komutunu kullanarak yardım ekranına erişebilirsiniz.

## history

Kabukta, daha önce girdiğiniz komutların bir geçmişi vardır, aslında bu komutlara göz atabilirsiniz. Bu, daha önce kullandığınız bir komutu yeniden yazmadan bulup çalıştırmak istediğinizde oldukça faydalıdır.

- **Komut geçmişinizi görmek:**

```
$ history
```

- **Önceki komutu tekrar çalıştırmak:** Yukarı ok tuşuna basın.
- **Son komutu tekrar çalıştırmak:**

```
!!
```

Örneğin, `cat dosya1` yazdıysanız ve tekrar çalıştırmak istiyorsanız, sadece `!!` yazıp Enter'a basabilirsiniz. Bu, en son çalıştırdığınız komutu çalıştıracaktır.

- **Ters arama:** `Ctrl-R` tuşlarına birlikte basın. Bu, ters arama komutudur. `Ctrl-R`'ye basıp aradığınız komutun bir kısmını yazmaya başlarsanız, size eşleşmeleri gösterecektir. `Ctrl-R` tuşuna tekrar basarak

bunlar arasında gezinebilirsiniz. Kullanmak istediğiniz komutu bulduktan sonra Enter tuşuna basmanız yeterlidir.

- **Ekranı temizleme:**

```
$ clear
```

- **Tab tuşu ile tamamlama:** Komut satırı ortamında en kullanışlı özelliklerden biri tab tuşu ile tamamlamadır. Bir komutun, dosyanın, dizinin vb. başlangıcını yazmaya başlarsanız ve Tab tuşuna basarsanız, arama yaptığınız dizinde bulunduğu şeye göre otomatik tamamlama yapacaktır. Örneğin, `chrome` komutunu çalıştırmaya çalışıyorsanız, `chr` yazıp Tab tuşuna basabilirsiniz, otomatik olarak `chrome` tamamlanacaktır.

## cp (Copy)

Dosyaları diğer işletim sistemlerinde kopyalayıp yapıştırmaya benzer şekilde, kabuk bize bunu yapmanın daha da basit bir yolunu sunar.

- **Tek bir dosya kopyalama:**

```
$ cp <kopyalanacak_dosya> <hedef_konum>
```

`kopyalanacak_dosya` kopyalamak istediğiniz dosyadır ve `hedef_konum` dosyayı kopyaladığınız yerdir.

Örnek:

```
$ cp metin.txt /home/ali/Dökümanlar/
```

Bu komut, `metin.txt` adlı dosyayı `/home/ali/Dökümanlar/` dizinine kopyalar.

- **Çoklu dosya ve dizin kopyalama:**

Birden fazla dosya ve dizini kopyalayabilirsiniz ve ayrıca joker karakterleri de kullanabilirsiniz. Joker karakter, arama daha fazla esneklik kazandıran bir desen tabanlı seçimi temsil eden bir karakterdir. Daha fazla esneklik için her komutta joker karakterleri kullanabilirsiniz.

### Joker karakterler:

- `*`: Tüm tek karakterleri veya herhangi bir dizeyi temsil eder.
- `?`: Tek bir karakteri temsil eder.
- `[]`: Köşeli parantez içinde yer alan herhangi bir karakteri temsil eder.

Örnek:

```
$ cp *.jpg /home/ali/Resimler
```

Bu komut, geçerli dizininizdeki tüm `.jpg` uzantılı dosyaları `Resimler` dizinine kopyalar.

- **Yinelenen dizin kopyalama:**

Yararlı bir komut, `-r` (recursive, yinelenen) işaretini kullanmaktır. Bu, bir dizin içindeki dosyaları ve dizinleri yinelenmeli olarak kopyalar.

Örnek:

```
$ cp -r Kabak /home/ali/Dökümanlar
```

Not: Aynı ada sahip bir dosyayı bir dizine kopyalarsanız, kopyaladığınız şey her neyse, var olan dosya üzerine yazılır. Bu, yanlışlıkla üzerine yazılmasını istemediğiniz bir dosyanız varsa iyi değildir. Dosyayı üzerine yazmadan önce size sormak için `-i` (interactive, etkileşimli) işaretini kullanabilirsiniz.

Örnek:

```
$ cp -i süperdosyam /home/pete/Resimler
```

## mv (Move)

`mv` komutu, dosyaları taşımak ve yeniden adlandırmak için kullanılır. `cp` komutuna benzer şekilde çalışır ancak dosyaları kopyalamak yerine taşır.

### Dosya Yeniden Adlandırma

Dosyaları şu şekilde yeniden adlandırabilirsiniz:

```
$ mv <eski_dosya> <yeni_dosya>
```

### Dosya Taşıma

Bir dosyayı farklı bir dizine şu şekilde taşıyabilirsiniz:

```
$ mv dosya2 /home/pete/Dökümanlar
```

### Çoklu Dosya Taşıma

Birden fazla dosyayı şu şekilde taşıyabilirsiniz:

```
$ mv dosya_1 dosya_2 /bir_dizin
```

### Dizin Yeniden Adlandırma

Dizinleri de şu şekilde yeniden adlandırabilirsiniz:

```
$ mv <dizin1> <dizin2>
```

### Üzerine Yazma

Bir dosyayı veya dizini `mv` ile taşırsanız, aynı dizindeki herhangi bir şeyin üzerine yazar. Bu davranışı değiştirmek için `-i` işaretini kullanabilirsiniz.

```
$ mv -i dizin1 dizin2
```

### Yedek Oluşturma

Taşıma işlemi gerçekleştirilmek ve üzerine yazmak istediğinizi varsayalım. Ayrıca o dosyanın bir yedeğini oluşturabilir ve eski sürümünü yalnızca bir `~` ile yeniden adlandırabilirsiniz.

```
$ mv -b dizin1 dizin2
```

## mkdir (Make Directory)

Oluşturduğumuz tüm dosyaları depolamak için dizinlere ihtiyacımız olacak. `mkdir` (Make Directory) komutu bunun için kullanılır, var olmayan bir dizin oluşturur. Aynı anda birden fazla dizin bile oluşturabilirsiniz.

```
$ mkdir kitaplar resimler
```

Ayrıca `-p` (parent, üst dizin) işareti ile aynı anda alt dizinler de oluşturabilirsiniz.

```
$ mkdir -p kitaplar/hemingway/favoriler
```

## rm (Remove)

Birçok dosya oluşturduk, şimdi bazılarını silelim. Dosyaları silmek için `rm` komutunu kullanabilirsiniz. `rm` (remove) komutu, dosya ve dizinleri silmek için kullanılır.

```
$ rm dosya1
```

**Dikkat:** `rm` komutunu kullanırken dikkatli olun. Silinen dosyaları geri getirmek için sihirli bir çöp kutusu yoktur. Silindikten sonra sonsuza kadar kaybolurlar. Bu yüzden dikkatli olun.

Neyse ki, ortalama bir kullanıcının önemli dosyaları kolayca silmesini önlemek için bazı güvenlik önlemleri alınmıştır. Yazma korumalı dosyalar, silinmeden önce sizden onay ister. Bir dizin yazma korumalıysa, kolayca silinemez.

Linux'ta `chattr` (Change Attribute) komutu, dosyaların ve dizinlerin **özniteliklerini** (attributes) değiştirmek için kullanılır. Bu komut, standart `chmod` (izinler) komutundan farklıdır; çünkü dosya izinleri yazma yetkisi verse bile, `chattr` ile korunan bir dosya silinemez veya değiştirilemez.

Özellikle sistem güvenliğini sağlamak ve kritik dosyaların yanlışlıkla silinmesini önlemek için çok güçlü bir araçtır.

### Temel Kullanım Sözdizimi

```
chattr [operatör] [öznitelik] [dosya_adı]
```

- `+`: Belirtilen özniteliği ekler.
- `-`: Belirtilen özniteliği kaldırır.
- `=`: Dosyanın sadece belirtilen özniteliklere sahip olmasını sağlar.

### En Çok Kullanılan Öznitelikler

Aşağıdaki tabloda en yaygın kullanılan `chattr` parametrelerini bulabilirsiniz:

Öznitelik	Açıklama
<b>i</b> (immutable)	Dosya <b>değiştirilemez, silinemez</b> , ismi değiştirilemez ve bağ oluşturulamaz. Root kullanıcısı bile bu korumayı kaldırmadan dosyayı silemez.
<b>a</b> (append-only)	Dosya silinemez veya içeriği değiştirilemez; ancak sonuna <b>yeni veri eklenebilir</b> (Log dosyaları için idealdir).

Öznitelik	Açıklama
<b>c</b> (compressed)	Dosyanın disk üzerinde kernel tarafından otomatik olarak sıkıştırılmasını sağlar.
<b>u</b> (undeletable)	Dosya silindiğinde verileri saklanır, böylece geri getirilmesi (undelete) kolaylaşır.

## Örnek Senaryolar

### 1. Dosyayı Tamamen Korumaya Almak (Silinemez/Değiştirilemez)

Bir dosyayı root dahil kimsenin silmemesi veya düzenleyememesi için **i** özniteliğini kullanırız:

Bash

```
sudo chattr +i onemli_dosya.txt
```

Bu aşamadan sonra **rm** veya **nano** ile dosyaya müdahale edilemez.

### 2. Sadece Veri Eklenmesine İzin Vermek

Bir log dosyasının geçmişinin silinmesini istemiyor, sadece yeni satırlar eklenmesini istiyorsanız:

Bash

```
sudo chattr +a sistem.log
```

### 3. Korumayı Kaldırmak

Özniteliği devre dışı bırakmak için **-** operatörü kullanılır:

```
sudo chattr -i onemli_dosya.txt
```

## Öznitelikleri Nasıl Kontrol Edilir? (**lsattr**)

Bir dosyanın hangi özniteliklere sahip olduğunu görmek için standart **ls** komutu işe yaramaz. Bunun yerine **lsattr** komutunu kullanmalısınız:

Bash

```
lsattr onemli_dosya.txt
```

Çıktı örneği:

----i-----e---- onemli\_dosya.txt (Buradaki i, dosyanın kilitli olduğunu gösterir.)

## Dikkat Edilmesi Gerekenler

- chattr** komutunu kullanmak için genellikle **root** veya **sudo** yetkisi gerekir.
- Bu komut genellikle **ext2**, **ext3**, **ext4**, **XFS** gibi Linux dosya sistemlerinde çalışır.
  - i** özniteliği atanmış bir dosyayı düzenlemeye çalışırsanız, "Permission Denied" (Erişim Engellendi) hatası alırsınız; bu hata dosya izinlerinden (**chmod**) değil, öznitelikten kaynaklıdır.

Dizinlerde kullanım için iki temel yöntem vardır:

## 1. Sadece Dizinin Kendisini Korumak

Eğer komutu doğrudan izin ismiyle çalıştırırsanız, öznitelik sadece o klasörün kendisine uygulanır.

```
sudo chattr +i /home/kullanici/ozel_dizin
```

### Bu ne sağlar?

- Klasörün adı değiştirilemez.
- Klasör silinemez.
- Klasörün içine **yeni dosya eklenemez** ve içindeki mevcut **dosyalar silinemez**.
- Ancak:* Klasörün içindeki mevcut bir dosyanın içeriği (eğer dosyanın kendi **i** özniteliği yoksa) hala değiştirilebilir.

## 2. Alt Dosya ve Dizinlerle Birlikte Korumak (Rekürsif)

Eğer klasörün içindeki her şeyin (tüm alt dosyalar ve klasörler) aynı korumaya sahip olmasını istiyorsanız **-R** (recursive) parametresini kullanmalısınız.

```
sudo chattr -R +i /home/kullanici/ozel_dizin
```

### Bu ne sağlar?

- Ana klasör kilitlenir.
- İçindeki tüm mevcut dosyalar ve alt klasörler de tek tek **+i** özniteliğini alır. Artık ne klasör ne de içindeki herhangi bir dosya silinebilir veya içeriği değiştirilebilir.

## Önemli Bir Fark: **i** ve **a** Öznitelikleri

Dizinler söz konusu olduğunda şu farkı bilmek çok faydalıdır:

Komut	Klasör İçindeki Etkisi
<code>chattr +i izin/</code>	İçine yeni dosya eklenemez, mevcut dosyalar silinemez.
<code>chattr +a izin/</code>	Mevcut dosyalar silinemez ama <b>yeni dosyalar oluşturulabilir</b> .

## Kontrol Etmek İçin

Dizine uygulanan özniteliği görmek için `lsattr` komutuna **-d** (directory) parametresini eklemeniz gerekir:

```
lsattr -d /home/kullanici/ozel_dizin
```

- f** veya **force** seçeneği, `rm` komutuna tüm dosyaları silmesini (yazma korumalı olsalar bile) kullanıcıya sormadan silmesini söyler (tabii ki gerekli izinlere sahipseniz).

```
$ rm -f dosya1
```

- Diğer birçok komutta olduğu gibi **-i** işaretini eklemek, dosyaları veya dizinleri gerçekten silmek isteyip istemediğinizi soran bir uyarı görüntüler.

```
$ rm -i dosya
```

- Varsayılan olarak `rm` ile bir dizini silemezsiniz. İçerdiği tüm dosyaları ve alt dizinleri silmek için `-r` (recursive, yinelemeli) işaretini eklemeniz gerekir.

```
$ rm -r dizin
```

- `rmdir` komutuyla boş bir dizini silebilirsiniz.

```
$ rmdir dizin
```

## find

Sistemde bu kadar çok dosya varken, belirli bir dosyayı bulmaya çalışmak biraz zor olabilir. Neyse ki, bunun için kullanabileceğimiz bir komut var: `find`

`find` komutunu kullanarak hangi dizinde arama yapacağınızı ve ne aradığınızı belirtmeniz gerekir. Bu örnekte, `puppies.jpg` adlı bir dosya aramaya çalışıyoruz.

- Dosya adına göre arama:**

```
$ find /home -name puppies.jpg
```

- Dosya türüne göre arama:**

Aradığınız dosyanın türünü de belirtebilirsiniz. Örneğin, bir klasör aramak için `-type d` seçeneğini kullanabilirsiniz.

```
$ find /home -type d -name MyFolder
```

Bu komutta, aradığımız dosya türünü `(d)` (dizin) olarak ayarladık ve yine `MyFolder` adına göre arama yapıyoruz.

**Önemli Not:** `find` komutu yalnızca aradığınız dizinde arama yapmaz, aynı zamanda o dizinin içinde olabilecek alt dizinlerin içine de bakar.

## help

Linux, bir komutu nasıl kullanacağınızı öğrenmenize veya bir komut için hangi işaretlerin (flag) mevcut olduğunu denetlemenize yardımcı olacak yerleşik araçlara sahiptir.

- help komutu:**

`help` komutu, diğer bash komutları (`echo`, `logout`, `pwd`, vb.) hakkında yardım sağlayan yerleşik bir bash komutudur. Kullanmak istediğiniz komut hakkında bilgi almak için aşağıdaki gibi yazabilirsiniz:

```
$ help echo
```

Bu komut, `echo` komutunu çalıştırmak istediğinizde kullanabileceğiniz açıklamayı ve seçenekleri size gösterecektir.

- help seçeneği:**

```
$ ls --help
```

## man

Linux programları hakkında daha fazla bilgi edinmek istiyorsanız, `man` komutunu kullanarak man sayfalarına erişebilirsiniz. Man sayfaları, komutların ayrıntılı açıklamalarını, seçeneklerini ve kullanım örneklerini içerir.

Örneğin, `ls` komutu hakkında daha fazla bilgi edinmek için:

```
$ man ls
```

Man sayfaları bazen teknik olabilir, ancak bir komutun tüm özelliklerini öğrenmek için harika bir kaynaktır.

## whatis

Bir komutun ne işe yaradığından şüphe duyuyorsanız, `whatis` komutunu kullanarak kısa bir açıklama alabilirsiniz. `whatis` komutu, komut satırı programları hakkında özlü bilgiler sağlar.

### Kullanım:

```
$ whatis <komut_adı>
```

### Örnek:

```
$ whatis cat
```

Bu örnekte, `cat` komutunun ne işe yaradığı hakkında kısa bir açıklama görürsünüz. Açıklama, komutun man sayfasından alınır.

## alias

Uzun komutları yazmaktan yoruldunuz mu? Belki de aynı komutu tekrar tekrar mı kullanıyorsunuz?

Linux size, sık kullandığınız komutlar için takma adlar oluşturma imkanı sunar. Bu takma adlar sayesinde komutları daha kısa ve yazması daha kolay hale getirebilirsiniz.

### Takma Ad Oluşturma:

Bir takma ad oluşturmak için `alias` komutunu kullanın. Takma adın ismini istediğiniz gibi seçebilirsiniz, ardından eşittir işareti (=) yazın ve takma adın hangi komutu çalıştırmasını istediğinizi belirtin.

Örneğin, `ls -la` komutunu sık sık kullanıyorsanız, bunun için `la` adında bir takma ad oluşturabilirsiniz:

```
$ alias la='ls -la'
```

Bundan sonra, `ls -la` yazmak yerine `la` yazabilirsiniz. `la` yazdığınızda, aslında `ls -la` komutu çalıştırılacaktır.

### Kalıcı Takma Adlar:

Bu komutla oluşturduğunuz takma adlar, terminal oturumunu kapattığınızda kaybolur. Eğer takma adın sürekli olarak kullanılabilir olmasını istiyorsanız, onu konfigürasyon dosyalarından birine eklemeniz gerekir.

Genellikle bash kullanıcıları için takma adları `/home/` dizinindeki `.bashrc` dosyasına eklenir. Bu dosyayı bir metin editörü ile açıp, takma adınızı şu şekilde ekleyebilirsiniz:

```
alias la='ls -la'
```

Daha sonra dosyayı kaydedin. Artık terminal oturumunu kapatıp açsanız bile `la` takma adını kullanmaya devam edebilirsiniz.

### Takma Ad Silme:

Oluşturduğunuz bir takma ada artık ihtiyacınız yoksa, `unalias` komutunu kullanarak silebilirsiniz.

```
$ unalias la
```

Bu komuttan sonra `la` takma adını kullanamazsınız.

## exit

Shell'den çıkmak için aşağıdaki komutlardan birini kullanabilirsiniz:

- `exit`: Bu en yaygın çıkış komutudur.
- `logout`: `exit` komutuyla aynı işlevi görür.

Eğer terminal emülatörü kullanıyorsanız, pencereyi kapatarak da çıkabilirsiniz.



## Metin İşlemleri

[Başa Dön](#)

## stdout (Standard Out)

Komutların nasıl çalıştığını ve çıktı ürettiklerini öğrendik. Şimdi bir sonraki konuya, yani **girdi/çıkış akışları (I/O)** konusuna geçelim. Aşağıdaki komutu çalıştırarak nasıl işlediğini görelim:

```
$ echo Hello World > peanuts.txt
```

### Ne Oldu?

Bu komutu çalıştırdığınız dizine gidin ve orada `peanuts.txt` adında bir dosya göreceksiniz. Dosyayı açtığınızda içinde "Hello World" yazısını göreceksiniz. Tek bir komutta birçok şey oldu, hadi gelin bunları parçalara ayıralım.

### echo Komutu

İlk olarak komutun ilk kısmını ele alalım:

```
$ echo Hello World
```

Bu komutun "Hello World" yazısını ekrana yazdırdığını biliyoruz. Peki nasıl oluyor? İşlemler, giriş almak ve çıktı döndürmek için **girdi/çıkış akışları (I/O)** kullanır. Varsayılan olarak, `echo` komutu klavyeden **standart girdi (stdin)** alır ve **standart çıktı (stdout)** olarak ekrana yazdırır. Bu nedenle, `echo Hello World` yazdığınızda ekranda "Hello World" görürsünüz.

### Yönlendirme Operatörü

Ancak I/O yönlendirme, bize daha fazla esneklik sağlayarak bu varsayılan davranışı değiştirmemize izin verir.

> sembolü, standart çıktının nereye gideceğini değiştirmemizi sağlayan bir **yönlendirme operatörüdür**. `echo Hello World` komutunun çıktısını ekrana yazdırmak yerine bir dosyaya göndermemizi sağlar. Dosya zaten yoksa, bizim için oluşturur. Ancak, dosya zaten varsa, üzerine yazar (kullandığınız shell'e bağlı olarak bunu önlemek için bir shell işareti ekleyebilirsiniz).

## Standart Çıktı Yönlendirme

Yani standart çıktı yönlendirme böyle çalışır!

### Dosyaya Ekleme

Peki ya `peanuts.txt` dosyasının üzerine yazmak istemezsek? Neyse ki, bunun için de bir yönlendirme operatörü var: >>

```
$ echo Hello World >> peanuts.txt
```

Bu komut, "Hello World" yazısını `peanuts.txt` dosyasının sonuna ekler. Dosya zaten yoksa, tıpkı > yönlendiricisi gibi bizim için oluşturur.

## stdın (Standard In)

Standart giriş (stdın) akışlarını da farklı kaynaklardan kullanabiliriz. Klavyeden gelen veriler varsayılan standart giriş kaynağı olsa da, dosyaları, diğer işlemlerin çıktılarını ve terminali de stdın olarak kullanabiliriz.

### Örnek: stdın Yönlendirme ile Dosya Kopyalama

Önceki derste oluşturduğumuz `peanuts.txt` dosyasını kullanalım. Bu dosyanın içinde "Hello World" yazısı olduğunu hatırlayın.

```
$ cat < peanuts.txt > banana.txt
```

Standart çıktı yönlendirmede > sembolünü nasıl kullandık, aynı şekilde standart giriş yönlendirmede de < sembolünü kullanıyoruz.

Normalde `cat` komutunda, bir dosya ismi verirsiniz ve bu dosya standart giriş (stdın) haline gelir. Bu örnekte, `peanuts.txt` dosyasını standart giriş olarak kullanmak için yönlendirdik. Daha sonra, `cat peanuts.txt` komutunun çıktısı olan "Hello World" metni, `banana.txt` adında yeni bir dosyaya yönlendirildi.

### Açıklama:

- `cat` komutu, varsayılan olarak standart girişten (stdın) okuyup standart çıktıyı (stdout) ekrana yazar.
- `< peanuts.txt` kısmı, `peanuts.txt` dosyasının içeriğini standart giriş akışına yönlendirir. Yani, `cat` komutu sanki klavyeden "Hello World" yazmışız gibi davranır.
- `> banana.txt` kısmı ise standart çıktı akışını `banana.txt` dosyasına yönlendirir. Böylece, `cat` komutunun "Hello World" çıktısı bu dosyaya yazılır.

### Sonuç:

Bu komutu çalıştırdığınızda, `banana.txt` adında yeni bir dosya oluşur ve içinde "Hello World" yazısı yer alır. Özetle, bu komut `peanuts.txt` dosyasının içeriğini `banana.txt` dosyasına kopyalamış olur.

## stderr (Standard Error)

Şimdi biraz farklı bir şey deneyelim. Sisteminizde olmayan bir dizinin içeriğini listelemeye çalışalım ve çıktıyı yine `peanuts.txt` dosyasına yönlendirelim.

```
$ ls /fake/directory > peanuts.txt
```

Bu komutu çalıştırdığınızda ekranda aşağıdaki gibi bir mesaj görmelisiniz:

```
ls: cannot access /fake/directory: No such file or directory
```

Muhtemelen şu anda, bu mesajın dosyaya yazdırılması gerektiğini düşünüyorsunuz. Aslında burada devreye giren başka bir I/O akışı var: **standart hata (stderr)**. Standart çıktı (stdout) akışından tamamen farklı olan standart hata akışı, varsayılan olarak çıktısını da ekrana gönderir. Yani, standart hata çıktısını farklı bir şekilde yönlendirmeniz gerekir.

Ne yazık ki, standart hata yönlendirme sembolleri (< veya >) kadar kolay değildir, ancak dosya tanımlayıcıları kullanılarak yapılabilir. Bir **dosya tanımlayıcısı**, bir dosyaya veya akışa erişmek için kullanılan negatif olmayan bir sayıdır. Dosya tanımlayıcıları hakkında daha sonra daha ayrıntılı bilgi edineceğiz, ancak şimdilik standart giriş (stdin), standart çıktı (stdout) ve standart hata (stderr) için dosya tanımlayıcılarının sırasıyla 0, 1 ve 2 olduğunu bilmeniz yeterli.

Şimdi standart hata çıktısını dosyaya yönlendirmek istiyorsak şöyle yapabiliriz:

```
$ ls /fake/directory 2> peanuts.txt
```

Bu komutta, standart hata mesajlarını `peanuts.txt` dosyasına yazdırmış olduk.

Peki hem standart hata hem de standart çıktıyı `peanuts.txt` dosyasına yazdırmak istersek ne yapabiliriz? Bunu da dosya tanımlayıcıları ile yapabiliriz:

```
$ ls /fake/directory > peanuts.txt 2>&1
```

Bu komut, `ls /fake/directory` komutunun sonuçlarını `peanuts.txt` dosyasına gönderir ve ardından `2>&1` ile standart hatayı standart çıktının yönlendirildiği yere yönlendirir. İşlem sırası burada önemlidir. `2>&1`, standart hatayı standart çıktının işaret ettiği yere gönderir. Bu durumda standart çıktı bir dosyaya işaret ettiğinden, `2>&1` de standart hatayı bir dosyaya gönderir. Yani `peanuts.txt` dosyasını açarsanız, hem standart hata hem de standart çıktı mesajlarını görmelisiniz. Yukarıdaki komut yalnızca standart hata çıktısı ürettiği için her ikisini de görmeyebilirsiniz.

Hem standart hata hem de standart çıktıyı bir dosyaya yönlendirmenin daha kısa bir yolu vardır:

```
$ ls /fake/directory &> peanuts.txt
```

Peki tüm bu gereksiz hata mesajlarından kurtulmak ve standart hata mesajlarını tamamen yok saymak istersek ne yapabiliriz? Çıktıyı `/dev/null` adlı özel bir dosyaya yönlendirebilirsiniz. Bu dosya, herhangi bir girişi yok sayar.

```
$ ls /fake/directory 2> /dev/null
```

## pipe ve tee

```
$ ls -la /etc
```

Çok uzun bir öğeler listesi göreceksiniz, aslında okuması biraz zor. Bu çıktıyı bir dosyaya yönlendirmek yerine, çıktıyı `less` gibi başka bir komutta görebilsek harika olmaz mı? Evet yapabiliriz!

```
$ ls -la /etc | less
```

Dikey çubukla temsil edilen pipe operatörü `|`, bir komutun standart çıktı (`stdout`) verisini alıp başka bir işlemin standart girdi (`stdin`) verisi haline getirmemizi sağlar. Bu durumda, `ls -la /etc` komutunun standart çıktısını alıp `less` komutuna aktardık.

Peki ya komut çıktıyı iki farklı akışa yazmak istersem? Bu, `tee` komutu ile mümkündür:

```
$ ls | tee f1st1k.txt
```

Ekranda `ls` komutunun çıktısını görmelisiniz ve `f1st1k.txt` dosyasını açarsanız aynı bilgileri görmelisiniz!

## env (Environment)

Aşağıdaki komutu çalıştırın:

```
$ echo $HOME
```

Ana dizininize giden yolu görmelisiniz, benimki `/home/kullanıcı` gibi görünüyor.

Peki ya şu komut:

```
$ echo $USER
```

Kullanıcı adınızı görmelisiniz!

Bu bilgiler nereden geliyor? Bunlar ortam değişkenlerinizden geliyor. Bunları yazarak görebilirsiniz:

```
$ env
```

Bu komut, şu anda ayarladığınız ortam değişkenleri hakkında bir sürü bilgi verir. Bu değişkenler, kabuğun ve diğer işlemlerin kullanabileceği faydalı bilgiler içerir.

İşte kısa bir örnek:

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/bin
```

```
PWD=/home/kullanıcı
```

```
USER=kullanıcı
```

Özellikle önemli bir değişken `PATH` değişkenidir. Bu değişkenlere, değişken adının önüne bir `$` işareti koyarak erişebilirsiniz:

```
$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/bin
```

Bu sisteminiz bir komut çalıştırdığında aradığı yolların, iki nokta ile ayrılmış listesini döndürür. İnternette manuel olarak bir paket indirip yüklediğinizi ve standart olmayan bir dizine koyduğunuzu ve bu komutu çalıştırmak istediğinizi varsayalım, `komut` yazıyorsunuz ve komut bulunamadı diyor. İkili dosyayı bir klasörde görüyorsunuz ve var olduğunu biliyorsunuz. Olan şey, `$PATH` değişkeninin bu ikili dosyayı aramak için o dizini kontrol etmemesi ve bu nedenle bir hata vermesidir.

Çalıştırmak istediğiniz birçok ikili dosyanın bulunduğu bir dizininiz olduğunu varsayalım, `PATH` ortam değişkeninizi bu dizini içerecek şekilde değiştirmeniz yeterlidir.

## cut

Metin işlemek için kullanabileceğiniz birkaç faydalı komut öğreneceğiz. Başlamadan önce, üzerinde çalışacağımız bir dosya oluşturalım. Aşağıdaki komutu kopyalayıp yapıştırın, bunu yaptıktan sonra "lazy" ve "dog" arasına bir TAB ekleyin (Ctrl-v + TAB tuşlarına basılı tutun).

```
$ echo 'The quick brown; fox jumps over the lazy dog' > sample.txt
```

Öğreneceğimiz ilk komut `cut` komutudur. Bu komut, bir dosyadan metin parçalarını ayıklar.

Karakter listesine göre içerik çıkarmak için:

```
$ cut -c 5 sample.txt
```

Bu, dosyadaki her satırın 5. karakterini çıktı olarak verir. Bu durumda "q" dır, boşluk da bir karakter olarak sayılır.

İçeriği bir alana göre çıkarmak için biraz değişiklik yapmamız gerekiyor:

```
$ cut -f 2 sample.txt
```

`-f` veya alan bayrağı, metni alanlara göre ayıklar, varsayılan olarak ayırıcı olarak TAB'ları kullanır, bu nedenle TAB ile ayrılmış her şey bir alan olarak kabul edilir. Çıktı olarak "dog" görmelisiniz.

Alan bayrağını, ayırıcı bayrağıyla birlikte kullanarak içeriği özel bir ayırıcıya göre ayırabilirsiniz:

```
$ cut -f 1 -d ";" sample.txt
```

Bu, TAB ayırıcıyı ";" ayırıcıya değiştirecek ve ilk alanı kestiğimiz için sonuç "The quick brown" olmalıdır.

## paste

`paste` komutu, `cat` komutuna benzer şekilde bir dosyadaki satırları birleştirir. Aşağıdaki içerikle yeni bir dosya oluşturalım:

sample2.txt

The

quick

brown

fox

Tüm bu satırları tek bir satırda birleştirelim:

```
$ paste -s sample2.txt
```

`paste` komutu için varsayılan ayırıcı TAB'dır, bu nedenle şimdi her kelimeyi ayıran TAB'lar içeren tek bir satır var.

Hadi bu ayırıcıyı ( `-d` ) biraz daha okunaklı bir şeyle değiştirelim:

```
$ paste -d ' ' -s sample2.txt
```

Şimdi her şey tek bir satırda olmalı ve boşluklarla ayrılmalıdır.

## head

Metin dosyalarında, özellikle sistem günlükleri gibi çok uzun dosyalarda, genellikle yalnızca ilk birkaç satıra bakmak istersiniz. Bunu yapmak için `head` komutunu kullanabilirsiniz.

`head` komutu varsayılan olarak bir dosyanın ilk 10 satırını görüntüler. Aşağıdaki komut, sistem günlüklerinden (`/var/log/syslog`) ilk 10 satırı görüntüleyecektir:

```
$ head /var/log/syslog
```

İlk kaç satırı görmek istediğinizi belirtmek için `-n` bayrağını kullanabilirsiniz. Örneğin, ilk 15 satırı görmek istiyorsanız:

```
$ head -n 15 /var/log/syslog
```

`-n` bayrağı ile birlikte satır sayısını belirterek, uzun dosyalarda hızlı bir şekilde özet bilgi edinebilirsiniz.

## tail

`head` komutuna benzer şekilde, `tail` komutu da varsayılan olarak bir dosyanın son 10 satırını görüntülemenizi sağlar. Sistem günlüklerine ( `/var/log/syslog` ) bakalım:

```
$ tail /var/log/syslog
```

`head` komutunda olduğu gibi, görmek istediğiniz satır sayısını da değiştirebilirsiniz:

```
$ tail -n 10 /var/log/syslog
```

`tail` komutunun gerçekten faydalı bir özelliği de, dosya içeriği güncellendikçe onu takip edebilmesidir. Bunu yapmak için `-f` (takip) bayrağını kullanabilirsiniz. Deneyin ve neler olduğunu görün:

```
$ tail -f /var/log/syslog
```

Sisteminizle etkileşim kurarken syslog dosyanız sürekli değişecektir. `tail -f` kullanarak, bu dosyaya eklenen her şeyi görebilirsiniz. Bu, sisteminizde neler olup bittiğini gerçek zamanlı olarak takip etmek için kullanışlıdır.

## expand ve unexpand

Önceki derste kullandığımız `sample.txt` dosyası bir tab içeriyordu. Normalde tablalar genellikle boşluk bırakır ancak bazı metin editörleri bunu net göstermeyebilir. Bir metin dosyasındaki tablalar istediğiniz aralığı sağlamayabilir. Sekmeleri boşluklara dönüştürmek için `expand` komutunu kullanabilirsiniz.

```
$ expand sample.txt
```

Bu komut, her taksimi bir grup boşluğa dönüştürerek çıktıyı yazdıracaktır. Bu çıktıyı bir dosyaya kaydetmek için aşağıdaki gibi çıktıyı yönlendirmeyi kullanın.

```
$ expand sample.txt > sonuc.txt
```

`expand` komutunun tersi olarak, boşluk gruplarını `unexpand` komutuyla tek bir tab'a dönüştürebilirsiniz:

```
$ unexpand -a sonuc.txt
```

Bu, özellikle metin dosyaları farklı programlar arasında paylaşılırken veya bir metin dosyasının biçimini korumak istediğinizde kullanışlıdır.

## join ve split

Birleştirme ve ayırma işlemleri için kullanışlı komutlar vardır:

Bu komut, ortak bir alanı temel alan birden fazla dosyayı birleştirebilir.

Örneğin, iki dosyayı birleştirmek istediğinizi varsayalım:

```
dosya1.txt
```

```
1 John
2 Jane
3 Mary
```

```
dosya2.txt
```

```
1 Doe
2 Doe
3 Sue
```

```
$ join dosya1.txt dosya2.txt
```

```
1 John Doe
2 Jane Doe
3 Mary Sue
```

Gördüğünüz gibi, dosyalar varsayılan olarak ilk alana göre birleştirilir ve alanların aynı olması gerekir. Aksi halde dosyaları sıralayabilirsiniz. Bu örnekte dosyalar 1, 2, 3 üzerinden birleştirildi.

Farklı alanları birleştirmek için hangi alanları kullanacağınızı belirtmeniz gerekir. Örneğin, `dosya1.txt` 'de 2. alanı ve `dosya2.txt` 'de 1. alanı birleştirmek istiyorsanız, komut şöyle görünür:

```
$ join -1 2 -2 1 dosya1.txt dosya2.txt
```

```
1 John Doe
2 Jane Doe
3 Mary Sue
```

`-1` `dosya1.txt` 'yi, `-2` ise `dosya2.txt` 'yi temsil eder.

- **split:** Bu komut, tek bir dosyayı birden fazla dosyaya böler.

```
$ split bazıdosya
```

Bu komut, satır sayısı 1000'e ulaştığında dosyayı birden fazla dosyaya böler. Oluşan dosyalar varsayılan olarak `x**` şeklinde adlandırılır.

## sort

Bu komut, bir dosyadaki satırları alfabetik sıraya göre sıralar.

Örneğin, `dosya1.txt` adlı bir dosyanız olduğunu varsayalım:

### **dosya1.txt**

köpek inek kedi fil kuş

```
$ sort dosya1.txt
```

```
fil
inek
kedi
köpek
kuş
```

Gördüğünüz gibi, `sort` komutu satırları alfabetik sıraya göre sıraladı.

- **Ters Sıralama:**

Ters sıralama yapmak için `-r` seçeneğini kullanabilirsiniz:

```
$ sort -r dosya1.txt
```

```
kuş
köpek
fil
inek
kedi
```

- **Sayısal Sıralama:**

Sayısal değer içeren metinleri sıralamak için `-n` seçeneğini kullanabilirsiniz:

```
$ sort -n dosya1.txt
```

```
kuş  
kedi  
inek  
fil  
köpek
```

Bu örnekte, sayılar metin içinde yer almasına rağmen, `sort` komutu `-n` seçeneği sayesinde sayısal olarak sıraladı.

## tr (Translate)

Bu komut, bir metin içindeki karakterleri başka karakterlere dönüştürmek için kullanılır.

Örneğin, tüm küçük harfleri büyük harflere dönüştürmek için:

```
$ tr a-z A-Z
```

```
hello
```

```
HELLO
```

Komutta `a-z` küçük harflerin aralığını, `A-Z` ise büyük harflerin aralığını belirtir. Böylece, `tr` komutu yazdığınız tüm küçük harfleri büyük harfe dönüştürür.

## uniq (Unique)

`uniq` (unique) komutu, metin ayırtırmak için kullanışlı bir başka araçtır.

Çok sayıda yinelenen öge içeren bir dosyanız olduğunu varsayalım:

```
reading.txt  
kitap  
kitap  
kağıt  
kağıt  
makale  
makale  
dergi
```

Yinelenen öğeleri kaldırmak istiyorsanız, `uniq` komutunu kullanabilirsiniz:

```
$ uniq reading.txt
```

```
kitap  
kağıt  
makale  
dergi
```

Bir satırın kaç kez tekrar ettiğini görelim:

```
$ uniq -c reading.txt
2 kitap
2 kağıt
2 makale
1 dergi
```

Yalnızca tekil değerleri görelim:

```
$ uniq -u reading.txt
dergi
```

Yalnızca yinelenen değerleri görelim:

```
$ uniq -d reading.txt
kitap
kağıt
makale
```

**Dikkat:** uniq komutu, yan yana olmayan yinelenen satırları algılamaz. Örneğin, reading.txt dosyanız aşağıdaki gibi olsun:

```
reading.txt
kitap
kağıt
kitap
kağıt
makale
dergi
makale
```

Bu durumda `uniq reading.txt` komutu tüm satırları döndürür.

uniq komutunun bu sınırlamasını aşmak için `sort` komutuyla birlikte kullanabilirsiniz:

```
$ sort reading.txt | uniq
makale
kitap
dergi
kağıt
```

Bu şekilde, tüm yinelenen satırlar, konumlarından bağımsız olarak kaldırılır.

## wc ve nl

Bu komut, bir dosyadaki toplam kelime sayısını görüntüler.

```
$ wc /etc/passwd
96  265  5925 /etc/passwd
```

Bu çıktı satırları, sırasıyla kelime sayısı, karakter sayısı ve dosya boyutunu göstermektedir.

- Belirli bir alanı saydırmak için `-l`, `-w` veya `-c` seçeneklerini kullanabilirsiniz.

```
$ wc -l /etc/passwd
96
```

- Bir dosyadaki satır sayısını görmek için nl (satır numaralandırma) komutunu da kullanabilirsiniz.

```
dosya1.txt
ben
adana'yı
seviyorum
```

```
$ nl dosya1.txt
1. ben
2. istanbul'u
3. seviyorum
```

## grep

grep, muhtemelen en sık kullanacağınız metin işleme komutlarından biridir. Belirli bir kalıpla eşleşen karakterleri dosyalarda aramanıza olanak tanır.

Bir dizinde belirli bir dosyanın olup olmadığını veya bir metnin bir dosyada bulunup bulunmadığını öğrenmek isterseniz? Elbette her satırı tek tek incelemezsiz, grep kullanırsınız!

Örnek olarak `sample.txt` dosyamızı kullanalım:

```
$ grep fox sample.txt
```

grep komutu, sample.txt dosyasında "fox" kelimesini bulduğunu göstermelidir.

- **Büyük/Küçük Harfe Duyarlı Olmayan Arama:**

-i bayrağı ile büyük/küçük harfe duyarlı olmayan aramalar yapabilirsiniz:

```
$ grep -i somepattern somefile
```

- **Diğer Komutlarla Kombinasyon:**

grep komutunu, | sembolü ile diğer komutlarla birleştirebilirsiniz. Bu sayede daha esnek aramalar yapabilirsiniz:

```
$ env | grep -i User
```

Gördüğünüz gibi, grep oldukça çok yönlüdür. Kalıplarınızda hatta **düzenli ifadeler** bile kullanabilirsiniz:

```
$ ls /somedir | grep '.txt$'
```

Bu komut, /somedir dizinindeki tüm ".txt" ile biten dosyaları döndürmelidir.



# Gelişmiş Metin İşlemleri

## regex (Regular Expressions)

Düzenli ifadeler, daha önce karşılaştığımız yıldız (\*) gibi özel semboller kullanarak desenlere göre metin seçimi yapan güçlü bir araçtır. Bu ifadeler hemen hemen tüm programlama dillerinde kullanılabilir.

Örnek metnimiz olarak şunu ele alalım:

```
alican deniz kabukları satıyor
sahile göre
```

### 1. ^ ile Satırın Başı

```
^sahile
ifadesi sadece "sahile göre" satırını seçer.
```

### 2. \$ ile Satırın Sonu

```
sahile$
ifadesi sadece "sahile göre" satırını seçer.
```

### 3. . ile Tek Karakter Eşleşmesi

```
s.
ifadesi "sahile" ile eşleşir.
```

### 4. [] ile Köşeli Ayraç Kullanımı

Bu biraz kafa karıştırıcı olabilir. Köşeli ayraçlar, içinde belirtilen karakterlerden herhangi biriyle eşleşmeyi sağlar.

```
k[ıaö]z
Bu ifade "kız", "kaz" ve "köz" ile eşleşir.
```

Daha önce gördüğümüz ^ sembolü, köşeli ayraç içinde kullanıldığında ayraç içindeki karakterler HÂRİÇ herhangi bir karakteri temsil eder.

```
k[^ı]z
Bu ifade "kaz" ve "köz" ile eşleşir ancak "kız" ile eşleşmez.
```

Köşeli ayraçlar ayrıca aralıklarla birden fazla karakteri temsil edebilir.

```
k[a-c]z
Bu ifade "kaz", "kbz" ve "kcz" gibi desenlerle eşleşir.
```

Dikkatli olun, köşeli ayraçlar büyük/küçük harfe duyarlıdır:

k[A-C]z

Bu ifade "kAz", "kBz" ve "kCz" ile eşleşir ancak "kaz", "kbz" ve "kcz" ile eşleşmez.

İşte bazı temel düzenli ifade örnekleri böyledir.



Referans ve Katkılar: Bu belgedeki bilgiler [Türkçe Linux](#) üzerinden referans alınarak derlenmiştir.