macOS Terminal Komutları Rehberi

Başlangıçtan İleri Seviyeye — Kısaltmalar, Açıklamalar ve İpuçlarıyla

Her komutun ne anlama geldiğini öğren • 300+ komut • Karanlık tema

Claude by Anthropic · 2025

Rehber İçeriği

1. Terminal ve Kabuk Temelleri

Terminal Nedir?

Terminal (İng. *Terminal Emulator*), işletim sisteminin çekirdeğiyle doğrudan iletişim kurmanızı sağlayan metin tabanlı bir arayüzdür. macOS'ta varsayılan kabuk **Zsh** (Z Shell) olup Catalina sürümünden itibaren Bash'in yerini almıştır. Kabuk (shell), komutlarınızı okuyup çekirdeğe ileten bir yorumlayıcıdır.

Terminal'i açmanın yolları:

Komut	Açılımı	Ne yapar?
`Cmd + Boşluk`	*Spotlight Search*	Spotlight'ı aç, 'Terminal' yaz
`open -a Terminal`	*open = aç, -a = uygulama*	Terminal'i komutla başlat
`Finder → Uygulamalar → Yardımcılar`	—	Finder üzerinden manuel açma

Komut Anatomisi

Her komut üç parçadan oluşur:

```
komut [seçenekler / flag'ler] [argümanlar]
```

Komut: çalıştırılacak program | **Seçenek (flag):** davranışı değiştiren - veya -- ile başlayan parametre | **Argüman:** komutun üzerinde çalışacağı dosya, dizin veya değer

Komut	Açılımı	Ne yapar?
`man komut`	*man = manual (kılavuz)*	Komutun resmi kılavuz sayfasını göster (q ile çık)
`komut --help`	*--help = yardım bayrağı*	Kısa yardım ve seçenek listesini göster
`which komut`	*which = hangisi*	Komutun diskteki tam yolunu göster
`type komut`	*type = tür*	Komutun türünü göster (builtin/alias/function/file)
`whatis komut`	*whatis = bu nedir*	Komut hakkında tek satır özet
`apropos kelime`	*apropos = ilgili (Fransızca)*	Kılavuzlarda kelimeyi ara, ilgili komutları listele
`history`	*history = geçmiş*	Daha önce çalıştırılan komutları listele
`history | tail -20`	*tail = kuyruk*	Son 20 komutu göster
`!!`	*!! = önceki komut*	Son komutu tekrar çalıştır
`!ls`	*! = geçmiş operatörü*	'ls' ile başlayan son komutu çalıştır
`clear`	*clear = temizle*	Terminali temizle (Ctrl+L ile aynı)
`exit`	*exit = çıkış*	Terminal oturumunu kapat
`echo 'metin'`	*echo = yankı/yazdır*	Metni veya değişken değerini ekrana yazdır
`printf 'fmt'`	*printf = print formatted*	Biçimlendirilmiş metin yazdır

 man sayfasında / tuşuyla arama yapabilir, n ile sonraki sonuca, q ile çıkabilirsiniz.

Kabuk Değişkenleri ve Özel Karakterler

Komut	Açılımı	Ne yapar?
`$HOME`	*HOME = ev dizini değişkeni*	Kullanıcının ana dizini (~'ye eşdeğer)
`$USER`	*USER = kullanıcı adı*	Mevcut kullanıcı adını tutar
`$PATH`	*PATH = yol listesi*	Kabuğun komutları aradığı dizinlerin listesi
`$SHELL`	*SHELL = kabuk yolu*	Aktif kabuğun tam yolunu gösterir
`$?`	*? = son çıkış kodu*	Son komutun çıkış durumu (0=başarılı)
`$$`	*$$ = süreç ID'si*	Mevcut kabuğun PID numarası
`~`	*~ = tilde (ev dizini)*	$HOME ile aynıdır
`.`	*. = nokta (mevcut dizin)*	Bulunulan dizini temsil eder
`..`	*.. = çift nokta (üst dizin)*	Bir üst dizini temsil eder
`*`	** = joker karakter (glob)*	Sıfır veya daha fazla karakterle eşleşir
`?`	*? = tek joker*	Tek bir karakterle eşleşir
`[abc]`	*[] = karakter sınıfı*	Parantez içindeki karakterlerden biriyle eşleşir

2. Dosya ve Dizin Yönetimi

Gezinme (Navigation)

Komut	Açılımı	Ne yapar?
`pwd`	*pwd = Print Working Directory*	Bulunduğunuz dizinin tam yolunu gösterir
`cd ~`	*cd = Change Directory, ~ = home*	Kullanıcının ana dizinine gider
`cd /`	*cd = Change Directory, / = root*	Sistemin kök dizinine gider
`cd ..`	*cd .. = bir üst dizine git*	Bir üst seviyeye çıkar
`cd -`	*cd - = önceki dizine dön*	Bir önceki çalışma dizinine döner
`cd ~/Desktop`	*cd + tam yol*	Masaüstü dizinine gider
`cd 'Ad Boşluklu'`	*tırnak = boşluklu ad*	İsimde boşluk varsa tırnak kullanılır
`pushd /yol`	*pushd = push directory*	Dizin yığınına ekler ve oraya gider
`popd`	*popd = pop directory*	Yığından önceki dizini geri alır
`dirs`	*dirs = directories*	Dizin yığınını listeler

 cd tek başına girildiğinde her zaman $HOME'a döner. cd - ise son iki dizin arasında geçiş yapmanızı sağlar.

Listeleme — ls (List Segments)

ls = *List Segments* — dizin içeriğini listeler. Seçenekler birleştirilebilir: **ls -lah** gibi.

Komut	Açılımı	Ne yapar?
`ls`	*ls = list (listele)*	Mevcut dizindeki dosya ve klasörleri göster
`ls -l`	*-l = long format (uzun biçim)*	İzin, sahip, boyut ve tarih bilgisiyle listele
`ls -a`	*-a = all (tümü)*	Gizli dosyaları da göster (. ile başlayanlar)
`ls -la`	*-l + -a birleşimi*	Gizli dahil tüm dosyaları ayrıntılı listele
`ls -lh`	*-h = human-readable (okunabilir)*	Boyutları KB/MB/GB olarak göster
`ls -lt`	*-t = time (zaman)*	Son değiştirilme tarihine göre sırala
`ls -lS`	*-S = Size (boyut)*	Boyuta göre büyükten küçüğe sırala
`ls -R`	*-R = Recursive (özyinelemeli)*	Alt dizinlerle birlikte hepsini listele
`ls -d */`	*-d = directory, */ = klasörler*	Sadece klasörleri listele
`ls -1`	*-1 = tek sütun*	Her dosyayı tek satırda listele
`ls *.txt`	**.txt = .txt uzantılı joker*	Sadece .txt dosyalarını listele
`ls -lai`	*-i = inode numarası*	Dosyaların inode numaralarını göster

Dosya ve Klasör Oluşturma

Komut	Açılımı	Ne yapar?
`touch dosya.txt`	*touch = dokunmak*	Boş dosya oluşturur; varsa erişim tarihini günceller
`touch {a,b,c}.txt`	*touch + brace expansion*	a.txt b.txt c.txt oluşturur (brace expansion)
`mkdir klasor`	*mkdir = Make Directory*	Yeni bir klasör oluşturur
`mkdir -p a/b/c`	*-p = parents (üst dizinler)*	İç içe tüm dizinleri tek seferde oluşturur
`mkdir -p proje/{src,test,docs}`	*brace expansion ile dizin ağacı*	src/ test/ docs/ alt klasörlerini oluşturur
`install -d /hedef/klasor`	*install = kur*	İzinlerle birlikte dizin oluşturur
`mktemp`	*mktemp = Make Temporary*	Geçici benzersiz bir dosya oluşturur
`mktemp -d`	*mktemp -d = directory*	Geçici benzersiz bir dizin oluşturur

Kopyalama — cp (CoPy)

cp = *CoPy* — dosya veya dizin kopyalama.

Komut	Açılımı	Ne yapar?
`cp kaynak.txt hedef.txt`	*cp = copy*	Dosyayı yeni adla kopyalar
`cp kaynak.txt /hedef/dizin/`	*cp + hedef dizin*	Dosyayı başka dizine kopyalar
`cp -r klasor/ yedek/`	*-r = recursive*	Klasörü içindekilerle birlikte kopyalar
`cp -i kaynak.txt hedef/`	*-i = interactive*	Üzerine yazmadan önce onay ister
`cp -n kaynak.txt hedef/`	*-n = no clobber*	Hedef varsa üzerine YAZMAZ
`cp -v kaynak.txt hedef/`	*-v = verbose (ayrıntılı)*	Kopyalanan dosyaları ekrana yazdırır
`cp -p kaynak.txt hedef/`	*-p = preserve*	İzin ve tarih bilgisini korur
`cp -u kaynak.txt hedef/`	*-u = update*	Sadece daha yeni dosyaları kopyalar
`cp -a klasor/ yedek/`	*-a = archive (arşiv modu)*	-rp'yi birleştirir; tam yedek için idealdir

Taşıma ve Yeniden Adlandırma — mv (MoVe)

mv = *MoVe* — taşıma ve yeniden adlandırma için kullanılır.

Komut	Açılımı	Ne yapar?
`mv eski.txt yeni.txt`	*mv = move*	Dosyayı yeniden adlandırır
`mv dosya.txt ~/Desktop/`	*mv + hedef dizin*	Dosyayı masaüstüne taşır
`mv klasor/ /yeni/yol/`	*mv + klasör taşıma*	Tüm klasörü başka konuma taşır
`mv -i dosya.txt hedef/`	*-i = interactive*	Üzerine yazmadan önce onay ister
`mv -n dosya.txt hedef/`	*-n = no clobber*	Hedef varsa taşımaz
`mv -v dosya.txt hedef/`	*-v = verbose*	Taşınan dosyaları ekrana yazdırır
`mv *.txt /arsiv/`	*joker ile toplu taşıma*	Tüm .txt dosyalarını arşive taşır

Silme — rm / rmdir

⚠ **rm ile silinen dosyalar Çöp Kutusu'na GİTMEZ, kalıcı olarak silinir!**

Komut	Açılımı	Ne yapar?
`rm dosya.txt`	*rm = ReMove*	Dosyayı kalıcı olarak siler
`rm -i dosya.txt`	*-i = interactive*	Silmeden önce onay ister (güvenli)
`rm -f dosya.txt`	*-f = force*	Onay istemeden zorla siler
`rm -r klasor/`	*-r = recursive*	Klasörü içindekilerle birlikte siler
`rm -rf klasor/`	*-r + -f = zorla recursive sil*	Onay istemeden tüm klasörü siler (ÇOK DİKKATLİ!)
`rm -v dosya.txt`	*-v = verbose*	Silinen her dosyayı ekrana yazdırır
`rmdir bos_klasor`	*rmdir = Remove Directory*	Sadece BOŞ klasörü siler
`rmdir -p a/b/c`	*-p = parents*	İç içe boş dizinleri üstten siler
`trash dosya.txt`	*trash = çöpe taşı*	Çöp Kutusu'na gönderir (brew install trash)

Dosya İçeriği Görüntüleme

Komut	Açılımı	Ne yapar?
`cat dosya.txt`	*cat = conCATenate*	Dosyayı okur ve terminale basar
`cat -n dosya.txt`	*-n = number (numaralandır)*	Satır numaraları ile gösterir
`cat -A dosya.txt`	*-A = All (tüm karakterler)*	Görünmez karakterleri de gösterir
`cat f1.txt f2.txt`	*cat ile birleştirme*	İki dosyayı arka arkaya basar
`less dosya.txt`	*less = daha az (less > more)*	Sayfalı görüntüler; / ara, q çık
`more dosya.txt`	*more = daha fazla*	Sayfalı görüntüler; Boşluk=ileri
`head dosya.txt`	*head = baş (ilk kısım)*	İlk 10 satırı gösterir
`head -n 20 dosya.txt`	*-n = number of lines*	İlk 20 satırı gösterir
`tail dosya.txt`	*tail = kuyruk (son kısım)*	Son 10 satırı gösterir
`tail -n 30 dosya.txt`	*-n = satır sayısı*	Son 30 satırı gösterir
`tail -f gunluk.log`	*-f = follow (takip et)*	Dosyaya eklenen satırları canlı gösterir
`tail -F gunluk.log`	*-F = Follow (dosya silinse de)*	Dosya silinip yeniden oluşsa bile takip eder
`wc -l dosya.txt`	*wc = Word Count, -l = lines*	Satır sayısını gösterir
`wc -w dosya.txt`	*wc = Word Count, -w = words*	Kelime sayısını gösterir
`wc -c dosya.txt`	*wc = Word Count, -c = chars*	Bayt (karakter) sayısını gösterir
`stat dosya.txt`	*stat = statistics (istatistik)*	Dosyanın tüm meta verilerini gösterir
`file dosya.txt`	*file = dosya türü*	Dosyanın türünü tahmin eder
`xxd dosya.bin`	*xxd = hex dump*	Dosyayı hexadecimal olarak gösterir

 tail -f uygulamalar/error.log ile canlı hata takibi yapabilirsiniz. Ctrl+C ile çıkılır.

Sembolik Bağlantılar — ln (LiNk)

Sembolik link (symlink): Başka bir dosyaya veya dizine işaret eden kısayol. **Hard link**: Aynı inode'u paylaşan ikinci bir isim.

Komut	Açılımı	Ne yapar?
`ln -s /orijinal/yol /link`	*ln = link, -s = symbolic*	Sembolik link oluşturur
`ln -sf kaynak hedef`	*-f = force*	Hedef varsa üzerine yazar
`ln kaynak hedef`	*ln (s'siz) = hard link*	Hard link oluşturur
`ls -la`	*ls -l ile link görme*	Link -> hedef şeklinde görünür
`readlink -f link`	*readlink = link oku*	Linkin gösterdiği gerçek yolu verir
`unlink link`	*unlink = bağı kopar*	Sembolik linki siler

3. Metin İşleme ve Arama

grep — Global Regular Expression Print

grep = *Global Regular Expression Print* — dosyalarda kalıp eşleştirme ve arama.

Komut	Açılımı	Ne yapar?
`grep 'kelime' dosya.txt`	*grep = ara*	Dosyada kelimeyi içeren satırları bulur
`grep -i 'kelime' dosya.txt`	*-i = ignore case*	Büyük/küçük harf duyarsız arar
`grep -r 'kelime' klasor/`	*-r = recursive*	Klasör ve alt dizinlerin tamamında arar
`grep -n 'kelime' dosya.txt`	*-n = number*	Eşleşen satır numarasını gösterir
`grep -v 'kelime' dosya.txt`	*-v = invert match*	Eşleşmeyen satırları gösterir
`grep -c 'kelime' dosya.txt`	*-c = count*	Kaç satırda geçtiğini sayar
`grep -l 'kelime' *.txt`	*-l = files with matches*	Kelimeyi içeren dosya adlarını listeler
`grep -L 'kelime' *.txt`	*-L = files without match*	Kelimeyi İÇERMEYEN dosyaları listeler
`grep -o 'kelime' dosya.txt`	*-o = only matching*	Sadece eşleşen kısmı yazdırır
`grep -A 3 'kelime' dosya.txt`	*-A = After (sonra)*	Eşleşmeden sonra 3 satır daha gösterir
`grep -B 3 'kelime' dosya.txt`	*-B = Before (önce)*	Eşleşmeden önce 3 satır daha gösterir
`grep -C 3 'kelime' dosya.txt`	*-C = Context (bağlam)*	Öncesi ve sonrasıyla 3 satır bağlam gösterir
`grep -E 'hata|uyari' log.txt`	*-E = Extended regex*	Gelişmiş regex ile birden fazla kalıp arar
`grep -rn 'TODO' .`	*-r + -n birlikte*	Tüm projedeki TODO yorumlarını satır no. ile bulur
`grep '^Basla' dosya.txt`	*^ = satır başı anchoru*	Satır başında aranan kelimeyi bulur
`grep 'biter$' dosya.txt`	*$ = satır sonu anchoru*	Satır sonunda aranan kelimeyi bulur
`grep -P '\d{3}' dosya.txt`	*-P = Perl regex*	Perl uyumlu regex kullanır (macOS'ta ggrep gerekir)

 grep -rn 'TODO\|FIXME' . komutu ile projenizin tüm yapılacaklar listesini çıkarabilirsiniz.

find — Dosya Sistemi Arama

find, dosya sistemi üzerinde çok güçlü sorgular çalıştırmanızı sağlar.

Komut	Açılımı	Ne yapar?
`find . -name '*.log'`	*find = bul, . = buradan*	Mevcut dizinde .log dosyalarını arar
`find / -name 'dosya.txt' 2>/dev/null`	*2>/dev/null = hataları gizle*	Tüm sistemde arar, izin hatalarını gizler
`find . -type f`	*-type f = file (dosya)*	Sadece dosyaları bulur
`find . -type d`	*-type d = directory*	Sadece dizinleri bulur
`find . -type l`	*-type l = link*	Sadece sembolik linkleri bulur
`find . -name '*.txt' -type f`	*name + type birlikte*	.txt uzantılı sadece dosyaları bulur
`find . -size +10M`	*-size = boyut, M = megabyte*	10 MB'dan büyük dosyaları bulur
`find . -size -1k`	*-size -1k = 1 KB'dan küçük*	1 kilobayttan küçük dosyaları bulur
`find . -mtime -7`	*-mtime = modification time*	Son 7 günde değiştirilen dosyalar
`find . -atime -1`	*-atime = access time*	Son 24 saatte erişilen dosyalar
`find . -newer referans.txt`	*-newer = daha yeni*	referans.txt'den daha yeni dosyaları bulur
`find . -empty`	*-empty = boş*	Boş dosya ve dizinleri bulur
`find . -name '*.tmp' -delete`	*-delete = sil*	Bulunan dosyaları otomatik siler
`find . -name '*.txt' -exec wc -l {} \;`	*-exec = komut çalıştır*	Her bulunan dosyaya komut uygular
`find . -name '*.py' -exec grep -l 'def' {} +`	*-exec + toplu işlem*	Tüm sonuçları tek seferde işler
`find . -perm 777`	*-perm = permissions*	Belirli izindeki dosyaları bulur

sed — Stream EDitor

sed = *Stream EDitor* — metin akışı üzerinde bul-değiştir ve dönüştürme işlemleri.

Komut	Açılımı	Ne yapar?
`sed 's/eski/yeni/g' d.txt`	*s = substitute, g = global*	Tüm eşleşmeleri değiştirir (terminale basar)
`sed -i '' 's/eski/yeni/g' d.txt`	*-i = in-place, '' = yedeksiz*	Dosyayı doğrudan düzenler (macOS)
`sed -i.bak 's/eski/yeni/g' d.txt`	*-i.bak = yedekle*	.bak yedek alarak dosyayı düzenler
`sed -n '5,10p' dosya.txt`	*-n = no print, p = print*	5 ile 10. satırları yazdırır
`sed '5d' dosya.txt`	*d = delete*	5. satırı siler
`sed '/yorum/d' dosya.txt`	*/kalıp/d = kalıba uyan satırı sil*	'yorum' içeren satırları siler
`sed 's/^/ /' dosya.txt`	*^ = satır başı*	Her satırın başına 2 boşluk ekler
`sed 's/ *$//' dosya.txt`	*$ = satır sonu*	Satır sonundaki boşlukları temizler
`sed -n '/BASLANGIC/,/BITIS/p' d.txt`	*kalıplar arası yazdırma*	İki kalıp arasındaki satırları yazdırır

awk — Aho, Weinberger, Kernighan

awk — yaratıcılarının soyadlarının baş harflerinden oluşur (Aho, Weinberger, Kernighan). Sütun tabanlı metin işleme için kullanılır.

Komut	Açılımı	Ne yapar?
`awk '{print $1}' dosya.txt`	*$1 = 1. alan*	Her satırın 1. sütununu yazdırır
`awk '{print $NF}' dosya.txt`	*NF = Number of Fields*	Her satırın son sütununu yazdırır
`awk -F: '{print $1}' /etc/passwd`	*-F = Field separator*	':' ayırıcıyla 1. alanı yazdırır
`awk -F, '{print $2,$3}' veri.csv`	*-F, = virgül ayırıcı*	CSV'nin 2. ve 3. sütunlarını yazdırır
`awk 'NR==5' dosya.txt`	*NR = Number of Record*	5. satırı yazdırır
`awk 'NR>=3 && NR<=7' dosya.txt`	*NR aralığı*	3-7. satırları yazdırır
`awk '{sum+=$1} END{print sum}' d.txt`	*END = dosya bittikten sonra*	1. sütunun toplamını hesaplar
`awk '$3 > 100' dosya.txt`	*koşullu filtreleme*	3. alanı 100'den büyük satırları gösterir
`awk '/kelime/{print $0}' dosya.txt`	*/kelime/ = regex filtre*	Kelimeyi içeren satırların tamamını basar
`awk '{print NR, $0}' dosya.txt`	*NR + $0 = satır no. + satır*	Her satıra numara ekler

sort, uniq, cut, tr

Komut	Açılımı	Ne yapar?
`sort dosya.txt`	*sort = sırala*	Satırları alfabetik sıralar
`sort -n dosya.txt`	*-n = numeric sort*	Sayısal sıralar
`sort -r dosya.txt`	*-r = reverse*	Ters sıralar
`sort -k2 dosya.txt`	*-k = key (alan)*	2. sütuna göre sıralar
`sort -u dosya.txt`	*-u = unique*	Sıralar ve tekrarları kaldırır
`sort -t',' -k2n veri.csv`	*-t = field separator*	CSV'yi 2. sütuna göre sayısal sıralar
`uniq dosya.txt`	*uniq = unique (tekrarları kaldır)*	Ardışık tekrar satırları kaldırır
`uniq -c dosya.txt`	*-c = count*	Her satırın tekrar sayısını gösterir
`uniq -d dosya.txt`	*-d = duplicates only*	Sadece tekrar eden satırları gösterir
`sort dosya.txt | uniq`	*sort + uniq birlikte*	Tüm tekrarları kaldırır (sort önce şart)
`cut -d',' -f1 veri.csv`	*cut = kes, -d = delimiter*	CSV'nin 1. sütununu alır
`cut -f2,4 sekme.txt`	*-f = field*	Sekme ayrılı 2. ve 4. alanları alır
`cut -c1-10 dosya.txt`	*-c = characters*	Her satırın 1-10. karakterlerini alır
`tr 'a-z' 'A-Z' < d.txt`	*tr = translate*	Küçük harfleri büyüğe çevirir
`tr -d '\n' < d.txt`	*-d = delete*	Yeni satır karakterlerini siler
`tr -s ' ' < d.txt`	*-s = squeeze*	Birden fazla boşluğu teke indirir

Pipe ve Yönlendirme Operatörleri

Yönlendirme operatörleri komutların girdi/çıktılarını birbirine veya dosyalara bağlar.

Komut	Açılımı	Ne yapar?
`cmd1 | cmd2`	*| = pipe (boru hattı)*	cmd1'in çıktısını cmd2'ye girdi olarak verir
`cmd > dosya.txt`	*> = stdout yönlendirme*	Standart çıktıyı dosyaya yazar (üzerine yazar)
`cmd >> dosya.txt`	*>> = append (ekle)*	Standart çıktıyı dosyanın sonuna ekler
`cmd < girdi.txt`	*< = stdin yönlendirme*	Dosyayı komuta girdi olarak verir
`cmd 2> hata.txt`	*2 = stderr, > = yönlendir*	Sadece hata mesajlarını dosyaya yazar
`cmd 2>> hata.txt`	*2>> = stderr ekle*	Hata mesajlarını dosyanın sonuna ekler
`cmd &> tumu.txt`	*&> = stdout + stderr*	Hem çıktı hem hatayı aynı dosyaya yazar
`cmd 2>/dev/null`	*/dev/null = kara delik*	Hata mesajlarını tamamen yok sayar
`cmd1 | tee d.txt | cmd2`	*tee = T boru*	Hem ekrana basar hem dosyaya yazar
`cmd1 && cmd2`	*&& = ve (başarılıysa devam)*	cmd1 başarılıysa cmd2'yi çalıştırır
`cmd1 || cmd2`	*|| = veya (başarısızsa devam)*	cmd1 başarısızsa cmd2'yi çalıştırır
`cmd1 ; cmd2`	*; = sıralı çalıştırma*	cmd1 bitmeden cmd2'yi çalıştırır
`(cmd1 && cmd2) || cmd3`	*parantez = gruplama*	Komutları gruplar, öncelik belirler
`xargs cmd`	*xargs = extended arguments*	Stdin'den okuduğu satırları argüman olarak verir

 *ls *.log | xargs grep 'ERROR' | sort | uniq -c | sort -rn ile en sık hataları bulabilirsiniz.*

4. Süreç ve Sistem Yönetimi

Sistem Bilgisi

Komut	Açılımı	Ne yapar?
`uname -a`	*uname = Unix Name, -a = all*	Kernel adı, sürümü ve mimari bilgisi
`sw_vers`	*sw = software, vers = version*	macOS sürümü, build numarası
`hostname`	*hostname = bilgisayar adı*	Bilgisayarın ağ adını gösterir
`whoami`	*whoami = ben kimim*	Oturum açmış kullanıcı adını gösterir
`id`	*id = kimlik*	Kullanıcı ve grup ID numaralarını gösterir
`uptime`	*uptime = çalışma süresi*	Sistemin ne kadar süredir açık olduğunu gösterir
`date`	*date = tarih*	Tarih ve saati gösterir
`date '+%Y-%m-%d %H:%M:%S'`	*format dizisi*	Özel formatta tarih gösterir
`cal`	*cal = calendar*	Ay takvimini gösterir
`df -h`	*df = Disk Free, -h = human*	Bağlı disklerin kullanım bilgisi
`df -H`	*-H = Human (1000 tabanlı)*	1000 tabanlı birimlerle gösterir
`du -sh klasor/`	*du = Disk Usage, -s = summary*	Klasörün toplam boyutunu gösterir
`du -sh * | sort -rh`	*du + sort büyükten küçüğe*	Mevcut dizindeki ögeleri boyuta göre sıralar
`vm_stat`	*vm = virtual memory*	Sanal bellek istatistiklerini gösterir
`sysctl hw.memsize`	*sysctl = system control*	Toplam RAM miktarını bayt cinsinden gösterir
`sysctl hw.ncpu`	*hw = hardware, ncpu = cpu sayısı*	İşlemci çekirdeği sayısını gösterir
`system_profiler SPHardwareDataType`	*SPHardwareDataType = HW profili*	CPU, RAM, seri no. dahil detaylı bilgi
`top`	*top = en üst (süreçler)*	Canlı süreç ve kaynak kullanımı ekranı
`top -o cpu`	*-o = order, cpu = CPU'ya göre*	En çok CPU kullanan süreçleri listeler
`top -o mem`	*-o mem = belleğe göre*	En çok RAM kullanan süreçleri listeler

Süreç Yönetimi

Komut	Açılımı	Ne yapar?
`ps aux`	*ps = Process Status, aux = tüm*	Sistemdeki tüm aktif süreçleri listeler
`ps aux | grep uygulama`	*grep ile filtrele*	Belirli bir uygulamanın süreç bilgisini bulur
`ps -ef`	*-e = all, -f = full format*	Tüm süreçleri tam formatta listeler
`pgrep Safari`	*pgrep = Process GREP*	Uygulamanın PID numarasını bulur
`pgrep -l Safari`	*-l = list names*	PID ve uygulama adını birlikte gösterir
`kill PID`	*kill = sonlandır*	Sürece SIGTERM sinyali gönderir (nazik kapatma)
`kill -9 PID`	*kill -9 = SIGKILL*	Süreci anında zorla sonlandırır
`kill -15 PID`	*kill -15 = SIGTERM*	Sürece temiz kapatma sinyali gönderir
`killall Safari`	*killall = adına göre hepsini öldür*	Adıyla eşleşen tüm süreçleri sonlandırır
`pkill -f 'python script.py'`	*pkill = process kill*	Komut satırı kalıbıyla süreç sonlandırır
`nohup komut &`	*nohup = No HangUP*	Terminal kapansa bile çalışmaya devam eder
`komut &`	*& = arka plan*	Komutu arka planda çalıştırır
`jobs`	*jobs = işler*	Mevcut kabukta arka plandaki işleri listeler
`fg %1`	*fg = foreground, %1 = 1. iş*	1. arka plan işini öne alır
`bg %1`	*bg = background*	1. askıya alınan işi arka plana gönderir
`wait`	*wait = bekle*	Tüm arka plan işleri tamamlanana dek bekler
`time komut`	*time = süre ölç*	Komutun çalışma süresini ölçer
`Ctrl + C`	*SIGINT = Interrupt*	Çalışan komutu keser
`Ctrl + Z`	*SIGTSTP = Stop*	Komutu askıya alır
`Ctrl + \`	*SIGQUIT = Quit*	Komutu çekirdek dosyasıyla sonlandırır

Disk ve Depolama

Komut	Açılımı	Ne yapar?
`diskutil list`	*diskutil = disk yöneticisi*	Tüm diskler ve bölümleri listeler
`diskutil info /dev/disk0`	*info = bilgi*	Disk hakkında ayrıntılı bilgi verir
`diskutil mountDisk /dev/disk2`	*mountDisk = diski bağla*	Diski sisteme bağlar
`diskutil unmountDisk /dev/disk2`	*unmountDisk = diski çıkar*	Diski güvenli şekilde çıkarır
`diskutil eraseDisk APFS Ad /dev/disk2`	*eraseDisk = diski formatla*	Diski silip yeni formatta oluşturur
`hdiutil attach dosya.dmg`	*hdiutil = HD Image UTility*	DMG disk imajını sisteme bağlar
`hdiutil detach /Volumes/Ad`	*detach = ayır*	Bağlı disk imajını kaldırır
`hdiutil create -size 100m -fs HFS+ d.dmg`	*create = oluştur*	Yeni bir disk imajı oluşturur
`fsck_apfs /dev/disk1s1`	*fsck = File System CheckK*	APFS disk bütünlüğünü kontrol eder

5. Ağ ve Bağlantı Komutları

Bağlantı Tanı ve Test

Komut	Açılımı	Ne yapar?
`ping google.com`	*ping = PING (İnternet Groper)*	ICMP paketleri göndererek bağlantıyı test eder
`ping -c 5 google.com`	*-c = count (paket sayısı)*	5 paket gönderip durur
`ping -i 0.5 google.com`	*-i = interval (aralık)*	0.5 sn aralıkla paket gönderir
`traceroute google.com`	*traceroute = yolu izle*	Paketin izlediği ağ yolunu gösterir
`traceroute -n google.com`	*-n = numeric (sayısal IP)*	DNS çözümlemesi yapmadan IP gösterir
`nslookup google.com`	*nslookup = Name Server LOOKUP*	Alan adının DNS kayıtlarını sorgular
`dig google.com`	*dig = Domain Information Groper*	Ayrıntılı DNS sorgusu yapar
`dig google.com MX`	*MX = Mail eXchange*	Mail sunucu kayıtlarını sorgular
`dig +short google.com`	*+short = kısa çıktı*	Sadece IP adresini gösterir
`host google.com`	*host = IP/isim dönüştürme*	Alan adını IP'ye veya tersine çevirir
`whois google.com`	*whois = kim bu?*	Alan adı kayıt bilgilerini sorgular
`curl ifconfig.me`	*curl = veri aktar*	Dış (genel) IP adresinizi gösterir
`curl ipinfo.io`	*ipinfo.io = IP bilgi servisi*	IP, konum, ISP bilgilerini JSON döner
`ifconfig`	*ifconfig = Interface CONFIGure*	Tüm ağ arayüzlerini listeler
`ifconfig en0`	*en0 = Ethernet port 0 (Wi-Fi)*	Wi-Fi arayüzünün detaylarını gösterir
`networksetup -getinfo Wi-Fi`	*networksetup = ağ ayarları*	Wi-Fi ağ yapılandırmasını gösterir
`networksetup -listallnetworkservices`	*listallnetworkservices = listele*	Tüm ağ servislerini listeler
`netstat -an`	*netstat = NETwork STATistics*	Ağ bağlantılarını ve portları listeler
`netstat -an | grep LISTEN`	*LISTEN = dinleyen portlar*	Açık ve dinleyen portları filtreler
`lsof -i :8080`	*lsof = List Open Files, -i = IP*	8080 portunu kullanan süreci gösterir
`lsof -i tcp`	*-i tcp = TCP bağlantıları*	Tüm TCP bağlantılarını listeler

curl — Client URL

curl = *Client URL* — HTTP/HTTPS/FTP ve daha birçok protokolü destekleyen veri transfer aracı.

Komut	Açılımı	Ne yapar?
`curl https://site.com`	*curl = client url*	GET isteği gönderir
`curl -o dosya.zip URL`	*-o = output (çıktı dosyası)*	Dosyayı indirir, isim verir
`curl -O URL`	*-O = uzak adı kullan*	URL'deki dosya adıyla indirir
`curl -I https://site.com`	*-I = head (sadece başlıklar)*	HTTP başlıklarını gösterir
`curl -X POST -d 'veri' URL`	*-X = method, -d = data*	POST isteği gönderir
`curl -X DELETE URL`	*-X DELETE*	DELETE isteği gönderir
`curl -H 'Auth: token' URL`	*-H = Header (başlık)*	Özel HTTP başlığı ekler
`curl -H 'Content-Type: application/json' -d '{"k":"v"}' URL`	*JSON POST*	JSON ile POST isteği gönderir
`curl -u kullanici:sifre URL`	*-u = user (kullanıcı)*	HTTP Basic Auth ile istek yapar
`curl -L URL`	*-L = location (yönlendirme)*	HTTP yönlendirmelerini takip eder
`curl -v URL`	*-v = verbose (ayrıntılı)*	İstek/yanıt tüm detaylarını gösterir
`curl -s URL`	*-s = silent (sessiz)*	İlerleme çubuğunu gizler
`curl -k URL`	*-k = insecure*	SSL sertifika doğrulamasını atlar
`curl --max-time 10 URL`	*--max-time = zaman aşımı*	10 saniyede yanıt gelmezse keser
`curl -C - -O URL`	*-C - = continue (devam et)*	Yarıda kalan indirmeyi sürdürür
`curl -w '%{http_code}' URL`	*-w = write-out formatı*	HTTP durum kodunu yazdırır

ssh — Secure SHell

ssh = *Secure SHell* — uzak sunuculara şifreli bağlantı kurar.

Komut	Açılımı	Ne yapar?
`ssh kullanici@sunucu.com`	*ssh = secure shell*	Uzak sunucuya bağlanır
`ssh -p 2222 kullanici@sunucu.com`	*-p = port*	Belirtilen port üzerinden bağlanır
`ssh -i ~/.ssh/anahtar kullanici@sunucu`	*-i = identity file*	Özel anahtar dosyasıyla bağlanır
`ssh-keygen -t ed25519 -C 'yorum'`	*ssh-keygen = anahtar üretici*	Ed25519 algoritmasıyla anahtar çifti oluşturur
`ssh-keygen -t rsa -b 4096`	*-b = bit sayısı*	4096 bit RSA anahtar çifti oluşturur
`ssh-copy-id kullanici@sunucu.com`	*ssh-copy-id = anahtarı kopyala*	Açık anahtarı sunucuya yükler
`ssh-add ~/.ssh/id_ed25519`	*ssh-add = anahtar ekle*	Anahtarı SSH agent'a ekler
`ssh-agent bash`	*ssh-agent = anahtar yöneticisi*	SSH agent başlatır
`scp dosya.txt kullanici@sunucu:/yol/`	*scp = Secure CoPy*	Dosyayı SSH üzerinden kopyalar
`scp -r klasor/ kullanici@sunucu:/yol/`	*scp -r = recursive*	Klasörü recursive kopyalar
`rsync -avz kaynak/ kullanici@sunucu:/hedef/`	*rsync = Remote SYNC*	Hızlı ve verimli dosya senkronizasyonu
`rsync -avz --delete kaynak/ hedef/`	*--delete = silinenleri yansıt*	Kaynakta silinenleri hedefte de siler
`ssh -N -L 8080:localhost:80 kullanici@s`	*-N = no command, -L = local port*	Yerel port tüneli açar
`ssh -N -R 9090:localhost:3000 kullanici@s`	*-R = remote port forward*	Uzak port tüneli açar
`ssh -D 1080 kullanici@sunucu`	*-D = dynamic (SOCKS proxy)*	SOCKS5 proxy tüneli açar

 ~/.ssh/config dosyasına: Host myserver HostName 1.2.3.4 User kullanici Port 2222 ekleyerek 'ssh myserver' ile bağlanabilirsiniz.

6. İzin ve Kullanıcı Yönetimi

chmod — CHange MODe

chmod = *CHange MODe* — dosya ve dizin izinlerini değiştirir.

İzin sistemi 3 grup × 3 bit: **rwx rwx rwx** → sahip | grup | diğerleri r=4 (read/okuma) w=2 (write/yazma) x=1 (execute/çalıştırma)

Komut	Açılımı	Ne yapar?
`chmod 755 dosya`	*7=rwx 5=r-x 5=r-x*	Sahip tam, grup ve diğerleri okuma+çalıştırma
`chmod 644 dosya.txt`	*6=rw- 4=r-- 4=r--*	Sahip okuma+yazma, diğerleri sadece okuma
`chmod 600 gizli.txt`	*6=rw- 0=--- 0=---*	Sadece sahip okuyup yazabilir (SSH key için)
`chmod 777 dosya`	*7=rwx 7=rwx 7=rwx*	Herkese tam yetki (güvenlik riski!)
`chmod 400 anahtar.pem`	*4=r-- 0=--- 0=---*	Sadece sahip okuyabilir (EC2 anahtarları için)
`chmod +x script.sh`	*+ = ekle, x = execute*	Çalıştırma izni ekler
`chmod -w dosya.txt`	*- = kaldır, w = write*	Yazma iznini kaldırır
`chmod u+x dosya`	*u = user (sahip)*	Sadece sahibine çalıştırma izni ekler
`chmod g-w dosya`	*g = group, - = kaldır*	Grup yazma iznini kaldırır
`chmod o=r dosya`	*o = others, = = tam ayarla*	Diğerlerine sadece okuma izni verir
`chmod a+r dosya`	*a = all (herkes)*	Herkese okuma izni ekler
`chmod -R 755 klasor/`	*-R = recursive*	Tüm alt dosya ve dizinlere uygular
`ls -la`	*izin sütununu oku*	-rwxr-xr-x şeklinde 10 karakter gösterir

chown — CHange OWNer & sudo

Komut	Açılımı	Ne yapar?
`chown kullanici dosya`	*chown = change owner*	Dosyanın sahibini değiştirir
`chown kullanici:grup dosya`	*kullanici:grup = sahip:grup*	Sahip ve grubu birlikte değiştirir
`chown -R kullanici klasor/`	*-R = recursive*	Tüm alt ögelerin sahibini değiştirir
`chgrp gelistirici dosya`	*chgrp = CHange GRouP*	Sadece grup sahipliğini değiştirir
`sudo komut`	*sudo = SuperUser DO*	Komutu yönetici (root) yetkileriyle çalıştırır
`sudo su`	*su = Switch User*	Root kabuğuna geçer
`sudo su - kullanici`	*su - = login shell*	Başka kullanıcının ortamıyla oturumu açar
`sudo -l`	*-l = list*	Kullanıcının sudo yapabileceği komutları listeler
`sudo -u kullanici komut`	*-u = user*	Komutu belirtilen kullanıcı adına çalıştırır
`sudo !!`	*!! = önceki komut*	Son komutu sudo ile yeniden çalıştırır
`su kullanici_adi`	*su = Switch User*	Başka kullanıcıya geçer
`groups`	*groups = gruplar*	Mevcut kullanıcının üye olduğu grupları listeler
`id kullanici`	*id = kimlik bilgisi*	Kullanıcının UID, GID ve grup listesini gösterir
`dscl . list /Users`	*dscl = Directory Service CL*	Sistemdeki tüm kullanıcıları listeler

 sudo !! yazarak 'permission denied' alan son komutu hızlıca yönetici olarak yeniden çalıştırabilirsiniz.

7. Arşivleme ve Sıkıştırma

tar — Tape ARchive

tar = *Tape ARchive* — başlangıçta manyetik bant yedeklemesi için tasarlanmıştır. Seçenekleri şöyle okuyun: **c**=create, **x**=extract, **t**=list, **v**=verbose, **f**=file, **z**=gzip, **j**=bzip2, **J**=xz.

Komut	Açılımı	Ne yapar?
`tar -cvf arsiv.tar klasor/`	*c=create v=verbose f=file*	Arşiv oluşturur (sıkıştırmasız)
`tar -czvf arsiv.tar.gz klasor/`	*z = gzip sıkıştırma*	gzip ile sıkıştırarak arşivler
`tar -cjvf arsiv.tar.bz2 klasor/`	*j = bzip2 sıkıştırma*	bzip2 ile sıkıştırarak arşivler (daha iyi oran)
`tar -cJvf arsiv.tar.xz klasor/`	*J = xz sıkıştırma*	xz ile sıkıştırır (en iyi sıkıştırma oranı)
`tar -xvf arsiv.tar`	*x = extract (çıkar)*	Arşivi mevcut dizine açar
`tar -xzvf arsiv.tar.gz`	*x + z = gzip'li arşivi aç*	gzip arşivini açar
`tar -xzvf arsiv.tar.gz -C /hedef/`	*-C = change directory*	Belirtilen dizine açar
`tar -tvf arsiv.tar`	*t = list (listele)*	Arşiv içeriğini dosyayı açmadan gösterir
`tar -xzvf arsiv.tar.gz dosya.txt`	*belirli dosya adı ile*	Arşivden sadece tek dosya çıkarır
`tar -czvf - klasor/ | ssh u@s 'cat > arsiv.tar.gz'`	*pipe ile uzak aktarım*	Arşivi SSH üzerinden uzak sunucuya gönderir
`tar --exclude='*.log' -czvf a.tar.gz klasor/`	*--exclude = dışla*	Belirtilen kalıpla eşleşen dosyaları arşive dahil etmez

zip, gzip, bzip2

Komut	Açılımı	Ne yapar?
`zip arsiv.zip dosya1 dosya2`	*zip = sıkıştır ve arşivle*	ZIP arşivi oluşturur
`zip -r arsiv.zip klasor/`	*-r = recursive*	Klasörü alt ögeleriyle ZIP'ler
`zip -9 arsiv.zip dosya.txt`	*-9 = maksimum sıkıştırma*	En yüksek sıkıştırma oranıyla ZIP'ler
`zip -e arsiv.zip dosya.txt`	*-e = encrypt (şifrele)*	Parola korumalı ZIP oluşturur
`zip -u arsiv.zip yeni.txt`	*-u = update*	Mevcut ZIP'e yeni dosya ekler
`unzip arsiv.zip`	*unzip = aç*	ZIP dosyasını mevcut dizine açar
`unzip arsiv.zip -d /hedef/`	*-d = destination*	ZIP'i belirtilen dizine açar
`unzip -l arsiv.zip`	*-l = list*	ZIP içeriğini açmadan listeler
`unzip -t arsiv.zip`	*-t = test*	ZIP bütünlüğünü test eder
`gzip dosya.txt`	*gzip = GNU zip*	Dosyayı sıkıştırır (orijinali siler)
`gzip -k dosya.txt`	*-k = keep (orijinali sakla)*	Sıkıştırır ama orijinali silmez
`gzip -d dosya.txt.gz`	*-d = decompress*	gzip dosyasını açar
`gunzip dosya.txt.gz`	*gunzip = gzip aç*	gzip dosyasını açar (gzip -d ile aynı)
`bzip2 dosya.txt`	*bzip2 = blok tabanlı sıkıştırma*	Dosyayı bzip2 ile sıkıştırır
`bunzip2 dosya.txt.bz2`	*bunzip2 = bzip2 aç*	bzip2 dosyasını açar
`zcat dosya.txt.gz`	*zcat = zip cat*	gzip dosyasını açmadan okur

8. Git Versiyon Kontrolü

Temel Git Komutları

Komut	Açılımı	Ne yapar?
`git init`	*init = initialize (başlat)*	Mevcut klasörü Git reposu olarak başlatır
`git clone URL`	*clone = klonla*	Uzak repoyu yerel makineye kopyalar
`git clone URL klasor`	*clone + hedef klasör*	Belirtilen klasöre klonlar
`git clone --depth 1 URL`	*--depth = yüzeysel klon*	Sadece son commit geçmişiyle klonlar (hızlı)
`git status`	*status = durum*	Değiştirilmiş, sahnelenmiş, izlenmeyen dosyaları gösterir
`git status -s`	*-s = short*	Kısa format durum raporu
`git add dosya.txt`	*add = ekle (sahneye al)*	Dosyayı commit için sahneye (index) ekler
`git add .`	*. = mevcut dizindeki her şey*	Tüm değişiklikler sahneye alınır
`git add -p`	*-p = patch (parça parça)*	Değişiklikleri seçerek sahneye alır
`git add -u`	*-u = update tracked files*	Sadece izlenen dosyaların değişikliklerini ekler
`git commit -m 'mesaj'`	*commit = taahhüt, -m = message*	Sahneyi kalıcı commit olarak kaydeder
`git commit -am 'mesaj'`	*-a = all, -m = message*	İzlenen dosyaları doğrudan commit eder
`git commit --amend`	*--amend = düzelt*	Son commit mesajını veya içeriğini düzenler
`git log`	*log = günlük*	Commit geçmişini gösterir
`git log --oneline`	*--oneline = tek satır*	Her commit'i tek satırda özetler
`git log --oneline --graph --all`	*--graph = dal ağacı*	Tüm dalları görsel ağaçla gösterir
`git log -p`	*-p = patch*	Her commit'in fark (diff) içeriğini gösterir
`git log --author='Ad'`	*--author = yazar*	Belirli yazara ait commit'leri filtreler
`git log --since='2 weeks ago'`	*--since = tarihten beri*	Belirli tarihten bu yana olan commit'ler
`git diff`	*diff = fark*	Sahneye alınmamış değişiklikleri gösterir
`git diff --staged`	*--staged = sahnelenmiş*	Sahneye alınmış değişiklikleri gösterir
`git diff HEAD`	*HEAD = son commit*	Son commit ile mevcut durumu karşılaştırır
`git show HASH`	*show = göster*	Belirli commit'in içeriğini gösterir
`git blame dosya.txt`	*blame = sorumluyu bul*	Her satırın kim tarafından yazıldığını gösterir

Dal (Branch) ve Birleştirme

Komut	Açılımı	Ne yapar?
`git branch`	*branch = dal*	Yerel dalları listeler (* aktif dal)
`git branch -a`	*-a = all (uzak dahil)*	Yerel ve uzak tüm dalları listeler
`git branch yeni-dal`	*branch + ad*	Yeni dal oluşturur
`git checkout yeni-dal`	*checkout = geçiş yap*	Belirtilen dala geçer
`git checkout -b yeni-dal`	*-b = branch (oluştur ve geç)*	Dal oluşturup hemen geçer
`git switch yeni-dal`	*switch = değiştir (modern yol)*	Dala geçer (Git 2.23+)
`git switch -c yeni-dal`	*-c = create*	Dal oluşturup geçer (modern)
`git branch -d dal`	*-d = delete*	Birleştirilmiş dalı siler
`git branch -D dal`	*-D = force delete*	Birleştirilmemiş olsa bile dalı siler
`git merge hedef-dal`	*merge = birleştir*	Hedef dalı mevcut dala birleştirir
`git merge --no-ff dal`	*--no-ff = no fast forward*	Merge commit oluşturarak birleştirir
`git merge --squash dal`	*--squash = ezdirerek birleştir*	Tüm commit'leri tek commit'e sıkıştırır
`git rebase main`	*rebase = yeniden temellendir*	Dalın tabanını main üzerine taşır
`git rebase -i HEAD~3`	*-i = interactive, HEAD~3 = 3 geri*	Son 3 commit'i etkileşimli düzenler
`git cherry-pick HASH`	*cherry-pick = seçici alma*	Başka daldaki commit'i mevcut dala ekler

Uzak Repo (Remote) ve Senkronizasyon

Komut	Açılımı	Ne yapar?
`git remote -v`	*remote = uzak, -v = verbose*	Uzak repo adreslerini listeler
`git remote add origin URL`	*origin = varsayılan uzak ad*	Uzak repo bağlantısı ekler
`git remote set-url origin URL`	*set-url = adresi güncelle*	Uzak repo adresini değiştirir
`git remote remove origin`	*remove = kaldır*	Uzak repo bağlantısını siler
`git push origin main`	*push = it, origin = uzak sunucu*	Commit'leri uzak sunucuya gönderir
`git push -u origin main`	*-u = upstream (takip edilecek)*	Takip bağlantısı kurarak push yapar
`git push --force-with-lease`	*güvenli zorla push*	Başkası push etmediyse zorla push yapar
`git pull`	*pull = çek*	Uzak değişiklikleri çekip birleştirir
`git pull --rebase`	*--rebase = rebase ile çek*	Fetch + rebase yapar (temiz geçmiş)
`git fetch`	*fetch = getir (birleştirmeden)*	Uzak değişiklikleri indirir, birleştirmez
`git fetch --prune`	*--prune = budama*	Silinmiş uzak dalları temizler
`git stash`	*stash = gizle*	Sahneye alınmamış değişiklikleri saklar
`git stash push -m 'mesaj'`	*push + mesaj*	Stash'e açıklama ekleyerek saklar
`git stash list`	*list = listele*	Saklanan tüm değişiklikleri listeler
`git stash pop`	*pop = al ve sil*	Son stash'i uygular ve listeden siler
`git stash apply stash@{1}`	*apply = uygula*	Belirli stash'i listeden silmeden uygular
`git tag v1.0.0`	*tag = etiket*	Mevcut commit'e etiket ekler
`git tag -a v1.0.0 -m 'mesaj'`	*-a = annotated tag*	Açıklamalı (ağır) etiket ekler
`git push origin --tags`	*--tags = etiketleri gönder*	Tüm etiketleri uzak sunucuya gönderir
`git reset --soft HEAD~1`	*--soft = soft reset*	Son commit'i geri alır, değişiklikler kalır
`git reset --hard HEAD~1`	*--hard = sert sıfırlama*	Son commit + değişiklikleri tamamen siler
`git revert HASH`	*revert = geri al*	Commit'i yeni bir commit ile geri alır
`git clean -fd`	*clean = temizle, -f force -d dir*	İzlenmeyen dosya ve klasörleri siler

 git log --oneline --graph --all --decorate ile tüm dal geçmişini görsel olarak izleyebilirsiniz.

9. Homebrew Paket Yöneticisi

Homebrew Kurulum

Homebrew, macOS için en yaygın kullanılan açık kaynaklı paket yöneticisidir. Terminale şu komutu girerek kurabilirsiniz:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Temel Paket İşlemleri

Komut	Açılımı	Ne yapar?
`brew install paket`	*install = kur*	CLI aracını veya kütüphaneyi kurar
`brew install --cask uygulama`	*--cask = paket (GUI uygulamalar)*	Grafik arayüzlü macOS uygulaması kurar
`brew uninstall paket`	*uninstall = kaldır*	Paketi sistemden kaldırır
`brew reinstall paket`	*reinstall = yeniden kur*	Paketi kaldırıp yeniden kurar
`brew update`	*update = Homebrew'u güncelle*	Paket listelerini ve Homebrew'u günceller
`brew upgrade`	*upgrade = paketleri güncelle*	Kurulu tüm paketleri en son sürüme günceller
`brew upgrade paket`	*upgrade + paket adı*	Belirli bir paketi günceller
`brew list`	*list = listele*	Kurulu tüm CLI paketlerini listeler
`brew list --cask`	*--cask = GUI uygulamalar*	Kurulu tüm GUI uygulamalarını listeler
`brew search kelime`	*search = ara*	Homebrew deposunda paket arar
`brew info paket`	*info = bilgi*	Paketin sürüm, bağımlılık ve web adresi bilgisi
`brew deps paket`	*deps = dependencies*	Paketin bağımlılıklarını listeler
`brew doctor`	*doctor = tanı*	Homebrew kurulumundaki sorunları tespit eder
`brew cleanup`	*cleanup = temizlik*	Eski sürümleri ve önbelleği siler
`brew cleanup -n`	*-n = dry run (simüle et)*	Silinecekleri gösterir, silmez
`brew pin paket`	*pin = sabitle*	Paketi belirli sürümde dondurur
`brew unpin paket`	*unpin = sabitlemeyi kaldır*	Dondurulmuş paketi güncellemeye açar
`brew leaves`	*leaves = yapraklar*	Başka paket tarafından kullanılmayan paketler
`brew autoremove`	*autoremove = gereksizleri sil*	Artık gerekmeyen bağımlılıkları kaldırır
`brew tap kullanici/repo`	*tap = ek depo ekle*	Üçüncü taraf Homebrew deposu ekler
`brew untap kullanici/repo`	*untap = depoyu kaldır*	Eklenmiş depoyu kaldırır
`brew services list`	*services = servisler*	Yönetilen servisleri (nginx, mysql vb.) listeler
`brew services start nginx`	*start = başlat*	Servisi başlatır ve sistem açılışında başlatılmak üzere ayarlar
`brew services stop nginx`	*stop = durdur*	Servisi durdurur
`brew services restart nginx`	*restart = yeniden başlat*	Servisi durdurup yeniden başlatır

 brew install wget htop tree bat ripgrep fzf ile geliştirme ortamını tek seferde hazırlayabilirsiniz.

10. Kabuk, Ortam Değişkenleri ve Kısayollar

Ortam Değişkenleri

Komut	Açılımı	Ne yapar?
`printenv`	*printenv = ortam değişkenlerini yazdır*	Tüm ortam değişkenlerini listeler
`printenv HOME`	*belirli değişkeni yazdır*	HOME değişkeninin değerini gösterir
`echo $HOME`	*echo = yazdır, $ = değişken*	HOME'un değerini terminale basar
`echo $PATH`	*$PATH = program arama yolları*	Kabuğun program aradığı yolları gösterir
`export AD=deger`	*export = dışa aktar*	Kabuk değişkeni oluşturup alt süreçlere aktarır
`export PATH="$PATH:/yeni/yol"`	*$PATH = mevcut yol + eklenen*	PATH'e yeni dizin ekler
`unset AD`	*unset = kaldır*	Ortam değişkenini siler
`env`	*env = environment*	Tüm ortam değişkenlerini listeler
`env VAR=deger komut`	*env ile geçici değişken*	Sadece o komut için değişken tanımlar
`source ~/.zshrc`	*source = dosyayı yükle ve çalıştır*	Zsh ayar dosyasını yeniden yükler
`. ~/.zshrc`	*. = source'un kısa hali*	source ile aynı işlevi görür

~/.zshrc Özelleştirme

~/.zshrc dosyası, Zsh kabuğu her başladığında otomatik çalıştırılan yapılandırma dosyasıdır.

Komut	Açılımı	Ne yapar?
`nano ~/.zshrc`	*nano = metin editörü*	.zshrc dosyasını düzenler
`alias ll='ls -la'`	*alias = takma ad*	'll' yazınca 'ls -la' çalışır
`alias gs='git status'`	*alias + git kısayolu*	'gs' ile git durumunu gösterir
`alias ip='curl ifconfig.me'`	*ip komut takma adı*	'ip' yazınca dış IP adresini gösterir
`alias ..='cd ..'`	*üst dizin kısayolu*	'..' yazınca bir üst dizine gider
`alias grep='grep --color=auto'`	*renkli grep*	grep çıktısını renklendirir
`export EDITOR='nano'`	*EDITOR = varsayılan editör*	Varsayılan metin editörünü ayarlar
`export HISTSIZE=10000`	*HISTSIZE = geçmiş boyutu*	Komut geçmişi boyutunu 10.000'e çıkarır
`export HISTFILESIZE=20000`	*HISTFILESIZE = dosya boyutu*	Geçmiş dosyasının satır limitini ayarlar
`setopt HIST_IGNORE_DUPS`	*setopt = Zsh seçenek ayarla*	Tekrar eden komutları geçmişe kaydetmez
`function mkcd() { mkdir -p $1 && cd $1; }`	*function = fonksiyon tanımla*	Oluştur ve gir: mkcd yeni_klasor
`autoload -U compinit && compinit`	*compinit = tamamlama başlat*	Gelişmiş Tab tamamlamayı etkinleştirir
`source ~/.zshrc`	*değişiklikleri uygula*	Düzenlemeleri mevcut oturuma yükler

 Oh My Zsh ile çok daha güçlü bir terminal deneyimi: sh -c '$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)'

Klavye Kısayolları (Zsh / Bash)

Komut	Açılımı	Ne yapar?
`Ctrl + A`	*A = Beginning*	İmleci satırın en başına taşır
`Ctrl + E`	*E = End*	İmleci satırın en sonuna taşır
`Ctrl + U`	*U = clear to beginning*	İmleçten satır başına kadar siler
`Ctrl + K`	*K = Kill to end*	İmleçten satır sonuna kadar siler
`Ctrl + W`	*W = Word (geri bir kelime sil)*	İmleçten önceki kelimeyi siler
`Alt + F`	*F = Forward one word*	İmleci bir kelime ileri taşır
`Alt + B`	*B = Backward one word*	İmleci bir kelime geri taşır
`Ctrl + R`	*R = Reverse search*	Komut geçmişinde geriye doğru arar
`Ctrl + G`	*G = Get out*	Reverse search'den çıkar
`Ctrl + L`	*L = cLear*	Ekranı temizler (clear ile aynı)
`Ctrl + C`	*C = Cancel (SIGINT)*	Çalışan komutu durdurur
`Ctrl + Z`	*Z = Suspend (SIGTSTP)*	Komutu askıya alır (fg ile devam)
`Ctrl + D`	*D = Delete / EOF*	Satır boşsa terminali kapatır
`Tab`	*Otomatik tamamlama*	Komut, dosya ve dizin adlarını tamamlar
`Tab Tab`	*Çift Tab*	Olası tamamlamaları listeler
`Yukari ok`	*önceki komut*	Komut geçmişinde geriye gider
`Asagi ok`	*sonraki komut*	Komut geçmişinde ileriye gider
`Ctrl + P`	*P = Previous*	Önceki komuta gider (yukarı ok)
`Ctrl + N`	*N = Next*	Sonraki komuta gider (aşağı ok)

macOS'a Özel Komutlar

Komut	Açılımı	Ne yapar?
`open .`	*open = aç*	Finder'da mevcut dizini açar
`open dosya.pdf`	*open + dosya*	Dosyayı varsayılan uygulamayla açar
`open -a 'Visual Studio Code' .`	*-a = application*	Belirtilen uygulama ile açar
`open -R dosya.txt`	*-R = Reveal*	Finder'da dosyayı seçili gösterir
`pbcopy < dosya.txt`	*pb = PasteBoard (pano) copy*	Dosya içeriğini panoya kopyalar
`pbpaste > dosya.txt`	*pb = PasteBoard paste*	Pano içeriğini dosyaya yazar
`echo 'metin' | pbcopy`	*pipe + pbcopy*	Komut çıktısını panoya kopyalar
`say 'Merhaba Dunya'`	*say = söyle (TTS)*	Metni sesli okur (Text To Speech)
`say -v 'Samantha' 'Hello'`	*-v = voice (ses)*	Belirli sesle okur
`screencapture -i ekran.png`	*-i = interactive*	Kullanıcı seçimi ile ekran görüntüsü alır
`screencapture -x ekran.png`	*-x = no sound*	Seccionsuz ekran görüntüsü alır
`mdfind 'kelime'`	*mdfind = Metadata FIND*	Spotlight veri tabanında arar
`mdfind -name 'dosya.txt'`	*-name = ada göre ara*	Dosya adıyla Spotlight araması
`mdls dosya.txt`	*mdls = MetaData LS*	Dosyanın tüm Spotlight meta verilerini gösterir
`defaults write com.apple.finder AppleShowAllFiles YES`	*defaults = sistem tercihleri*	Finder'da gizli dosyaları gösterir
`killall Finder`	*killall = tüm örnekleri sonlandır*	Finder'ı yeniden başlatır
`caffeinate -t 3600`	*caffeinate = kafein (uyuma!)*	3600 saniye (1 saat) uyku modunu engeller
`caffeinate -di`	*-d = disk, -i = idle*	Disk ve boşta uyumasını engeller
`pmset -g`	*pmset = Power Management SET*	Enerji tasarrufu ayarlarını gösterir
`softwareupdate -l`	*softwareupdate = yazılım güncelle*	Mevcut sistem güncellemelerini listeler
`softwareupdate -ia`	*-i = install, -a = all*	Tüm güncellemeleri sessizce yükler
`launchctl list`	*launchctl = Launch Control*	Çalışan Launch Agents ve Daemon'ları listeler
`osascript -e 'script'`	*osascript = Open Scripting Arch.*	AppleScript komutunu terminalde çalıştırır
`networksetup -getinfo Wi-Fi`	*networksetup = ağ ayarları*	Wi-Fi ağ bilgilerini gösterir
`networksetup -setairportpower en0 off`	*setairportpower = Wi-Fi gücü*	Wi-Fi'yi kapatır
`airport -s`	*airport = Wi-Fi taraması*	Çevredeki Wi-Fi ağlarını tarar

11. Hızlı Başvuru Tablosu

Kategori	Sık Kullanılan Komutlar
Gezinme	`pwd` • `cd ~` • `cd ..` • `cd -` • `pushd` • `popd`
Listeleme	`ls -la` • `ls -lh` • `ls -lt` • `ls -R` • `ls *.txt`
Dosya Oluştur	`touch` • `mkdir -p` • `mkdir -p {a,b,c}`
Kopyala/Taşı	`cp -r` • `cp -a` • `mv` • `mv -i` • `mv *.txt /hedef`
Sil	`rm -i` • `rm -r` • `rmdir` • `trash (brew)`
Görüntüle	`cat -n` • `less` • `head -n` • `tail -f` • `wc -l`
Arama	`grep -rni` • `find . -name` • `find . -size` • `mdfind`
Metin İşleme	`sed 's/eski/yeni/g'` • `awk '{print $1}'` • `sort -u` • `uniq -c`
Yönlendirme	`pipe |` • `> dosya` • `>> ekle` • `2>/dev/null` • `tee`
Sistem Bilgi	`uname -a` • `sw_vers` • `df -h` • `du -sh` • `vm_stat`
Süreç	`ps aux` • `kill -9` • `killall` • `pgrep` • `nohup &`
Ağ	`ping -c` • `curl -O` • `ssh` • `scp` • `netstat` • `lsof -i`
İzin	`chmod 755` • `chmod +x` • `chown` • `sudo !!`
Arşiv	`tar -czvf` • `tar -xzvf` • `zip -r` • `unzip -d`
Git	`git add .` • `git commit -m` • `git push` • `git pull` • `git stash`
Homebrew	`brew install` • `brew update` • `brew upgrade` • `brew cleanup`
macOS Özel	`open .` • `pbcopy` • `pbpaste` • `say` • `caffeinate` • `mdfind`
Yardım	`man komut` • `komut --help` • `whatis` • `apropos`

Bu rehberi faydalı buldunuz mu? Terminal her gün kullandıkça daha güçlü bir araç haline gelir. man sayfalarını okuyun, kendi alias'larınızı oluşturun ve düzenli pratik yapın!