✩ Bu repo shell komutlarının türkçe açıklamalarını içerir. Ekleme için pull request'lere tamamiyle açıktır.

Temel Komutlar

Dosya Ve Dizin İşlemleri

`pwd` : Neredeyiz? Şu an bulunduğumuz dizini gösterir. ("print working directory")

`ls` : Bulunduğumuz dizinin içindeki dosyaları ve klasörleri listeler. ("listing")

`cd directory_adı` : Başka bir dizine gitmemizi sağlar. ("change directory")

`touch dosya_adı` : Bulunduğumuz dizinde yeni dosya açar ve bununla beraber uzantısını da belirtebiliriz.

`cd ..` : Bir üst klasöre çıkmaya yarar. Bir diğer açıdan şuanda bulunduğunuz klasörün bulunduğu klasörü belirtir.

`~` : home directory anlamına gelir.

`cd ~/../.` : 'home', 'bir üst klasör', 'burası' anlamına gelir.

`cp klasör/dosya.csv backup/dosya.bck` : backup dizininde dosya.csv'nin kopyasını oluşturup ismini duplicate.txt yapar.

`cp dosya1.txt dosya2.txt backup` : İki tane dosyayı backup'ta kopyalar.

`mv` : Dosyaları taşır. (move)

`mv dosya_adi.txt yeni_dosya_adi.txt` : mv aynı şekilde dosyaların ve dizinlerin ismini değiştirmek için de kullanılır.

`rm` : Dosyaları siler. (remove)

`rm dosya.txt backup/dosya-2.txt` : Dizinde ismi verilen iki dosyayı siler.

`mkdir directory_ismi` : Yeni boş bir dizin açar.

`mkdir -p gecersiz_directory/hedef_directory` : gecersiz_directory'nin varolmaması durumunda hedef_directory ile birlikte onu da açar.

`rm -rf directory_ismi` : Dizini siler.

`pbcopy` : Sağına yazılanı panoya kopyalar / hafızaya alır.

`cat dosya_ismi` : Dosyanın içeriğini görmemizi sağlar.

`df` : Dosya sisteminizdeki disk alanı hakkında bilgi edinmenizi sağlar.

`ps` : Sisteminizde hangi işlemlerin çalıştığını görmenizi sağlar.

`wget` : Bir adresteki dosyayı terminal üzerinden doğruca kendi makinemize indirmek için kullanılır.

`grep` : Bu komut dosyaların içinde arama yapmamızı sağlar. Log incelerken oldukça işe yarayan bir komuttur.

Dosya Okuma / Görüntüleme

`less` : Daha büyük dosyalara bakmak için kullanılır.

`less`'e birden fazla dosya ismi verirsek `:n`'le bir sonraki dosyaya, `:p`'yla bir önceki dosyaya gideriz, `:q`'la çıkarız.

`head` : Dosyanın başına bakmaya yarar.

Command Line Flags

`head -n 3 directory/dosya_ismi.csv` : Dosyadaki ilk üç satırı getirir. ("number of lines")

`ls -R` : Bütün klasörleri ve içindekileri gösterir.

`man` : Yanına yazılan komut hakkında yardım eder. (manual)

`cut` : Csv dosyalarında sütunları seçer.

`cut -f 2-5,8 -d , dosya.csv` : 2. ve 5. sütun arasındakileri seçer. (flag'ler: -f (fields, sütunları belirtir), -d (delimiter, ayırıcı))

`grep` : Dosyada arama yapar.

`grep kelime directory/dosya.csv` : "kelime"yi içeren satırları getirir.

`grep -Hirn kelime directory` : Belirtilen dizin altında "kelime"yi içeren satırları getirir.

Grep komutunun Flag'leri

- `-c` : Kelimenin bulunduğu satır sayısı döndürür.
- `-h` : Birden fazla dosyada arama yapıyorsa dosya isimlerini döndürmez.
- `-i` : Büyük/küçük harf farkını yok sayar. ("kelime" ve "Kelime"yi arar)
- `-l` : Eşleşme bulunan dosya isimlerini getirir, eşleşmeleri getirmez.
- `-n` : Eşleşen satırların satır sıralarını döndürür.
- `-v` : Eşleşme bulunmayan satırları döndürür.

`paste` : İki tabloyu tek tablo yapar.

Komutların Çıktılarını Nasıl Saklarız?

> **Örnek:** `head -n 5 directory/dosya.csv > dosya2.csv` : çıktı dosya2.csv'ye gider.

Çıkardığımız dosyanın içeriklerine `cat dosya2.csv` 'yle bakabiliriz.

> **Örnek:** `head -n 5 directory/dosya.csv >> dosya2.csv` : çıktı dosya2.csv'ye gider.

Fakat buradaki farklılık, `>` yerine `>>` kullanmamız. eğer `>>` kullanırsak, yazdığımız veriler dosyaya eklenir, dosyada halihazırda bulunan veriler silinmez.

Yine aynı şekilde çıkardığımız dosyanın içeriklerine `cat dosya2.csv` 'yle bakabiliriz.

> **Örnek:** `head -n 5 directory/dosya.csv 2> dosya2.csv` : oluşan hata dosya2.csv'ye gider.

Buradaki farklılık ise `>` yerine `2>` kullanmamız. eğer `2>` kullanırsak, kullandığımız komutta oluşan hata çıktılarını dosyaya yazmış oluruz.

> **Örnek:** `head -n 5 directory/dosya.csv > dosya2.csv 2> err.txt`

Buradaki komutta hata oluşmazsa çıktı dosya2.csv'ye, hata oluşursa oluşan hata çıktısı err.txt'ye yazılır.

pipe | operatörü:

Pipe soldaki komutun çıktısını alıp sağa gönderir.

> **Örnek:** `head -n 5 directory/dosya.csv | tail -n 3` : Beşinci satıra kadar aldı, sonra sondan üçüncüyü aldı. (Tatil dosyanın sonundan alması gerektiğini belirtir.)
>
> **Örnek:** `cut -d , -f 2 directory/dosyacsv | grep -v anahtar_kelime` : `cut` 'la dosya.csv'deki 2. kolonu seçti, `pipe` 'la sağ tarafa `grep` 'le "anahtar_kelime" kelimesinin geçmediği durumların aramasını yaptı.
>
> **Örnek:** `cut -d , -f 1 directory/dosya.csv | grep -v Tarih | head -n 10` : dosya.csv verisinin ilk kolonunu seçtik, "Tarih" yazan header line'ı sildik ve ardından ilk on satırı aldık. (`head` en baştaki satırları alması gerektiğini belirtir.)

`wc` : Kaç tane karakter, kelime, ya da satır varsa sayar. ("word count")

Wildcards

Komutlarda birden fazla dosya ismi verirsek birden fazla dosya üstüne çalışır. Bütün dosyaların hepsini tek tek yazmaktansa * operatörünü kullanırız.

```
cut -d , -f 1 directory/*.csv
```

`cut -d , -f 1 directory/ka*.csv` : Klasörde kalem.csv ve kagit.csv isimli iki tane csv dosyası olduğunu varsayarsak, ikisini de seçmek için kullanılır.

`?` : Tek bir karakteri eşler, örneğin `201?.txt` 2017.txt ve 2018.txt'yi eşler, ama 2017-01.txt'yi eşlemez.

`[…]` : Köşeli parantezin içindeki bir karakteri eşler, mesela `201[78].txt` 2017.txt ya da 2018.txt'yi eşler.

`{…}` : `{*.txt, *.csv}` .txt ya da .csv'yle biten dosyaları eşler.

Sıralama İşlemleri

- `sort` komutuyla yapılabilir.
- `-n` flag'i: numerik olarak sıralar.
- `-r` : Tersten sıralar.
- `-b` : Boşlukları yok saymamızı sağlar.
- `-f` : Büyük/küçük harfi yok saymamızı sağlar. (folding)

> **Örnek:** `cut -d , -f 2 directory/dosya.csv | grep -v "anahtar_kelime" | sort -r` : aramayı tersten sıralar.

`uniq` : Çift yazılmış satırları siler.

`cut -d , -f 2 directory/dosya.csv | grep -v "anahtar_kelime" | sort | uniq -c` : İkinci kolonu ve içinde anahtar_kelime geçmeyen çıktıları alır, hepsini sıralar, her eşleşme bir kez çıkacak şekilde gösterir ve kaç kez gözüktüğü yazar.

Dosya İzinleri

Bazen kullandığımız dosyalar üzerinde yeterli iznimiz olmayabilir veya önemli dosyaların başkaları tarafından okunup yazılmasını istemeyebiliriz. böyle durumlarda dosya izinlerini kullanabiliriz.

`chmod` komutu ile yapılabilir.

`chmod` sekizlik (octal) sayı sistemi kullanır, bu sistemde `read` izni 4, `write` izni 2, `execute` izni ise 1 sayısı ile gösterilir.

Bir sistemde ise temel olarak 3 kullanıcı vardır, bunlar sırasıyla: - `owner` dosyanın sahibi - `group` bilgisayar üzerinde kayıtlı olan kullanıcılar - `other` geri kalan herkes

> **Örnek:** `chmod 644 file.txt` : Bu komut sonrasında file.txt'yi dosyanın sahibi okuyabilir/yazabilir. Bilgisayar üzerinde kayıtlı olan kullanıcılar ve geri kalan herkes bu dosyayı sadece okuyabilir.

Buradaki 6 sayısı dosya sahibinin iznini, 4 sayısı bilgisayar üzerinde kayıtlı olan kullanıcıların iznini ve diğer 4 sayısı ise geri kalan herkesin iznini temsil eder.

İzinleri hesaplamak basittir. mesela dosya sahibine okuma ve yazma izni vermek için 4+2=6, okuma ve çalıştırma izni vermek için ise 4+1, yani 5 yazabiliriz. hiçbir izin vermek istemiyorsak 0 yazabiliriz. sıralama ise daima **owner-group-other** şeklindedir.

> Written with StackEdit.