Git Bash / Shell / Ubuntu Terminal Komutları ve İşlevleri (Başlangıç Seviyesi)

NOT: Eğer komutlar bitişik yazılmamışsa arada bir boşluk bırakmayı unutmayınız. Bazı komutlarda yazan;
folder : sizin oluşturduğunuz veya oluşturacağınız "klasör" adını temsil etmektedir.
file : sizin oluşturduğunuz veya oluşturacağınız "dosya adını" temsil etmektedir.

KOMUTLAR :

$: Terminalde normal kullanıcı modunu gösteren işarettir. (terminal açıldığında komut satırında otomatik olarak görünür)

% : $ ile aynı işlevdedir. Farklı terminal uygulamalarında karşımıza çıkabilir.

: Administrator kullanıcı modunu gösteren işarettir. Root modu olarak da adlandırılır. Bu yetki açıkken yapılan işlemlerde dikkatli olunmalıdır. Öğrenme aşamasında veya alıştırma yaparken kalıcı değişikliklere ve yanlış silmelere neden olabilir, bu yüzden alıştırmalarda $ normal kullanıcı modu kullanılmalıdır.

pwd : Bulunduğumuz dosyanın relative path'ini gösterir. Yani nerede olduğumuzu root directory'den başlayarak bulunduğumuz klasöre kadar gösterir. Hangi klasör içerisinde olduğumuzu ve bunun yolunu/dizinini görmek istiyorsak bu komut kullanılır.

~ : Tilde işareti. Home Directory adı verilen konumu gösterir. Komut satırında yalın olarak bu işareti görüyorsak Home Directory'de olduğumuzu gösterir.

cd : sadece cd yazarak Home Directory'e gideriz. Home Directory'den iki veya daha fazla alt/üst klasör içerisinde olsak da cd komutu ile direkt olarak Home Directory'e gidebiliriz.

cd ~ : bu komut sadece "cd" komutu ile aynı işlevdedir. Yukarıdaki ile aynıdır.

. : komut verirken kullanılan "tek nokta" içerisinde bulunduğumuz klasörü temsil eder.

.. : komut verirken kullanılan "yan yana iki nokta" bir üs klasörü temsil eder.(Parent Directory)

cd / : cd yazıp bir boşluk bıraktıktan sonra / işareti kullanarak Root Directory'ye gideriz.

cd .. : bu komut ile içerisinde bulunduğumuz klasörün bir üst klasörüne gideriz.

cd ../.. : 2 klasör birden daha üst klasöre gitmek istersek bu komut ile gideriz.

cd folder : cd yazdıktan sonra gitmek istediğimiz klasör adını yazarak o klasör içerisine gideriz. cd klasör adı ile klasörler arasında gezebiliriz fakat sadece bir alt veya bir üst klasöre gidebiliriz. Yani 2 klasör altta veya üstte olan klasöre doğrudan gidemeyiz. Bu komut ile adım adım gidebiliriz.

ls : bu komut ile içerisinde bulunduğumuz klasörde bulunan dosya ve klasörleri listeleriz. "Ayrıntısız" bir şekilde sadece isimleri ile listelenir.
Mavi renkliler : Klasörleri
Yeşil/Turkuaz renkliler : Yürütülebilir Dosyaları
Beyaz renkliler : Dosyaları temsil eder.

ls -l : bu komut ile içerisinde bulunduğumuz klasörde bulunan dosya ve klasörler "ayrıntıları" ile listelenir. Dosyaların yetkileri, oluşturulma tarihleri vs. bu komut ile listelenebilir.

ls -a : bu komut ile bulunduğumuz klasör içerisindeki "gizli" dosya ve klasörler "dahil" olarak listelenir. "Ayrıntısız" bir şekilde sadece isimleri listelenir.

ls -al : bu komut ile bulunduğumuz klasör içerisindeki "gizli" dosya ve klasörler "dahil" olarak listelenir. "Ayrıntılı" olarak listelenir.

mkdir folder : bu komut ile folder adında yeni boş bir "klasör" oluştururuz.

mkdir -p folder0/folder1/folder2 : bu komut ile iç içe klasör oluştururuz.

mkdir -pv folder0/ folder1/ folder2 : bu komut ile iç içe klasör oluştururuz ve ekranda geri bildirim alırız.

rmdir folder : bu komut ile varolan "boş" bir klasörü sileriz.

rm file : bu komut ile varolan "boş" bir dosyayı sileriz.

rm –r file : bu komut ile "dolu bir dosya"yı silebiliriz.

rm –rf folder veya file: bu komut ile içi dolu bir klasör veya dosyayı silebiliriz. (recursive force/zorla sil komutu denebilir)

touch file : dosya oluşturmak için kullanılan komuttur. file.txt, file.js, file.py vb. dosya oluştururken uzantısı ile birlikte oluşturmalıyız.

cp : klasör veya dosya kopyalamak için kullanılan komuttur. aynı klasör içerisinde bir dosyayı farklı bir adla kopyalamak için komut örneği;

cp file newfile => aynı klasör içerisinde file ve newfile şeklinde iki dosya oluşur

bir klasör içinden başka bir klasör içerisine dosya kopyalamak için komut örneği;

cp ./file ./hedefklasöradı => bulunduğunuz klasör "içinde bulunan" başka bir klasöre file adında bir dosya kopyalar

cp ./file ../hedefklasöradı => bulunduğunuz klasörün "dışında" bulunan bir hedef klasörde file adında bir dosya kopyalar.

mv : bu komutun 2 fonksiyonu vardır. klasör veya dosyanın "adını değiştirir". klasör veya dosyayı "taşır".

mv file newfile => file isimli dosyanın ismini newfile olarak değiştirir.

mv ./file ./hedefklasöradı => bulunduğunuz klasör "içinde bulunan" başka bir klasöre file dosyasını taşır.

mv ./file ../hedefklasöradı => bulunduğunuz klasörün "dışında" bulunan bir hedef klasöre file dosyasını taşır.

mv ./file ../hedefklasöradı/newfile => bulunduğunuz klasörün "dışında" bulunan bir hedef klasöre file dosyasının ismini değiştirerek newfile olarak taşır.

mv ./file ../newfile => bulunduğunuz klasörün parent/üst klasörüne file dosyasının ismini newfile olarak değiştirir ve taşır.

NOT : cp ve **mv** komutları kullanılırken, hangi klasör içerisinde bulunduğumuza, hangi klasörden hangi klasöre taşıma veya kopyalama yapacağımıza göre komutlar değişiklik göstermektedir. Yani iç içe olan klasörlerde mi taşıma işlemi yapıyoruz, yoksa aynı dizinde bulunan ama iç içe olmayan farklı iki klasör arasında mı kopyalama veya taşıma yapıyoruz, hedef klasörü belirtirken bu konuya dikkat etmek gerekiyor.

echo : bu komutun da farklı kullanımları vardır; dosya içerisinde text/metin yazdırırız.

echo "Hello World" => yazarsak sadece terminal ekranına Hello World çıktısı üretir.

echo "Hello World" > file.txt => yazarsak file.txt dosyasının içerisine Hello World yazar. zaten file.txt isminde bir dosyamız varsa Hello World'ü bunun içerisine yazar fakat böyle bir dosya yoksa otomatik yeni bir file.txt isminde dosya oluşturur ve içerisine yazar.

echo "Hello World İki" >> file.txt => yazarsak file.txt dosyasının en alt satırına Hello World İki yazar. eğer içerisi dolu bir dosyamız varsa ve iki tane >> büyüktür işareti yerine > tek büyüktür işareti kullanarak komut gönderirsek tüm içeriğin üzerine yazarız, bu yüzden ekleme yapmak ve eski veriyi kaybetmek istemiyorsak >> iki adet büyüktür işareti kullanmamız gerekir. Aksi takdirde mevcut dosya içeriğini kaybederiz.

cat file.txt : bu komut ile dosyalar içerisinde bulunan text/metin'leri terminal ekranında görebiliriz. dosya içerisinin dolu olup olmadığını cat komutu ile kontrol edebilir veya görebiliriz.

cat > file.txt : bu komutu kullandığımızda terminal ekranı "editör ekranı" gibi çalışır ve yazığımız her kelime, metin vs file.txt dosyasının içerisine satır satır yazılır. Her yazdığımız metinden sonra kaydetmesi için Enter yaparız. Bu moddan kaydedip çıkmak için ise CTRL+C yaparak çıkarız.

cat >> file.txt : eğer iki adet büyüktür işareti olan bu komutu kullanırsak içeriği dolu olan bir file.txt dosyasının mevcut içeriğine bir değişiklik yapmadan dosyanın altına ekleme yaparız. > tek büyüktür işareti kullanırsak dosyanın eski içeriğinin silineceği unutulmamalıdır. Kaydet ve çıkış yukarıdaki komut ile aynıdır.

Bazı Sembollerin Kullanımları

> sembolü : > sembolü ile değişik işlemler için kullanabiliriz. Örneğin; bir komutun ne içeriğini bir dosya içerisine yazdırmak istiyorsak;

pwd > pwd.txt : yazarsak pwd içeriğini bir pwd.txt dosyası oluşturur ve onun içerisine yazar.

history > file.txt : yazarsak history içeriğini bir file.txt dosyası oluşturur ve onun içerisine yazar.

*** sembolü** : * işareti kullanarak çoklu işlemler gerçekleştirebiliriz. Örneğin; file1, file2, file3, file4 adında 4 adet dosyamız olsun. Bu dört dosyayı da aynı anda silmek istersek * sembolünden şöyle faydalanırız.

rm -r f* : f ile "başlayan" tüm dosyaları silecektir.

rm -r fi* : fi ile "başlayan" tüm dosyaları silecektir.

rm -r *fi : varsa fi ile "biten" tüm dosyaları silecektir.

rm -r *fi* : varsa "içerisinde" fi yazan tüm dosyaları silecektir.

{} sembolü : {} kullanarak yine çoklu işlem gerçekleştirebiliriz. Örneğin; aynı anda standart bir isimle başlayan 10 dosya veya klasör oluşturmak istediğimizi varsayalım. Komut;

touch file{1..10} : komutu ile 1'den 10'a kadar 10 adet file1, file2, ….., file9, file10 klasörü oluşturabiliriz.

{} parantez içerisindeki iki adet **..** kullandığımıza dikkat edelim.

$ sembolü : $ sembolünün "echo" komutu ile birlikte kullanımı. Önceden tanımladığımız bir değişkeni terminal ekranına çağırmak ve yazdırmak için echo komutu ile birlikte kullanılır. Örneğin;

selam="Blockchain Path'ine Selamlar" => yazarak "selam" kelimesine bir değişken tanımlayalım. Bu değişkene tanımladığımız değeri terminal ekranına yazdırmak için $ sembolünü şu şekilde kullanırız;

echo $selam : bu komutu yazdığımızda önceden tanımlanmış "Blockchain Path'ine Selamlar" cümlesi terminal ekranında gösterilir.

" " sembolü : " " sembolü tırnak içinde yer alan çoğu karakterin gerçek değerini korur.

' ' sembolü : ' ' sembolü tırnak işaretleri arasında kalan her şeyi korur.

\ sembolü : \ Ters eğik çizgi Karakterlerin özel anlamlarını değiştirmek veya metin içindeki tırnak işaretleri gibi özel karakterlerden kaçmak için kullanılır.

**$, " ", ' ', ** sembollerinin kullanımı gerektiğinde daha detaylı bilgi için araştırma yapmak gerekebilir. Burada kısaca değinilmiştir.

Araştırılabilecek diğer semboller : ? , [] , ^ , | sembolleri.

whoami : bu komut sistemde oturum açmış olan kullanıcı adını görüntüler.

Linux'te Dosya İzinleri

Linux sistemindeki her dosya ve dizin için **3 tip sahip** atanmıştır,

User : Bir kullanıcı dosyanın sahibidir.

Group : Bir kullanıcı grubu birden fazla kullanıcı içerebilir.

Other/All : Bir dosyaya erişimi olan diğer herhangi bir kullanıcı.

ugo = user/group/other

Linux sisteminizdeki her dosya ve dizin, tüm **3 sahip için tanımlanmış 3 izne** sahiptir.

Read : Okuma izni size bir dosyayı açma ve okuma yetkisi verir.

Write : Yazma izni size bir dosyanın içeriğini değiştirme yetkisi verir.

Execute : Linux'ta, yürütme izni ayarlanmadıkça bir programı çalıştıramazsınız.

rwx = read/write/execute

- işareti : no permission / izin yok anlamına gelir.

chmod : bu komut ile dosya üzerinde hangi kullanıcının hangi yetkiye (rwx) sahip olacağını değiştirebiliriz.

chmod u=rwx, g=rw-, o=r-- file.txt => bu komutu $ normal user olarak kullanamadığım için internetten araştırılması gerekiyor. doğru komut şeklini yazamamış olabilirim veya $ normal user olduğum için sistem yetki vermiyor olabilir. **chmod u+rwx,g-rwx file.txt veya chmod 777 file.txt, chmod 444 file.txt**

vim editör

vim editörü bir dosyanın "içerisine girerek" değişiklik yapmamıza yarayan bir komuttur. vim editörü açıldığında geri çıkması biraz sorun olabilir bu yüzden aşağıdaki adımları unutmamak gereklidir.

vim file.txt : komutunu yazarak yeni bir ekran açar ve file.txt dosyasının içerisine gireriz. Aşağıdaki adımlar çok önemlidir yoksa bu ekranı kullanamayız veya geri çıkamayabiliriz. Yeni ekran açıldığında;

İ' harfine basarız. Bu "insert" anlamına gelir ve file.txt içerisinde değişiklik yapmamıza izin verir. yapmak istediğimiz değişiklikleri yapar ve enter'a basarız. Vim editör içerisindeyken

d siler,

u : silme işlemini geri alır,

p : yapıştır,

gg : en başa götürür,

G en sona götürür,

Değişiklik işlemimiz bittiyse vim editörden çıkmak için;

ESC'ye basarız, **:** yaparız ve **q**, **wq** veya **q!** seçeneklerinden birini kullanarak **enter** yapar ve çıkarız.

q : quit, düz kaydetmeden çıkış

wq : kaydet ve çık

q! : zorla çık

Diğer editör seçenekleri için (nano vb) internetten araştırma yapılması gerekebilir.

Bazı Komutlar ve Notlar

Komut satırındaki imleç "kilitlenir ve yanıt vermez" ise CTRL+C yaparak imleç çözülebilir. İşe yaramaz ise CTRL+D veya CTRL+X denenebilir.

CTRL+L ekranı temizler. Aslında ekranı silmez yukarı kaydırır ve eski işlemleri görebiliriz.

clear : ekranı temizler ve siler.

history : terminal ekranına girdiğimiz komutları ekranda gösterir.

history > file.txt : terminal ekranına girdiğimiz komutları file.txt dosyası oluşturur ve dosyanın içerisine yazar.