

Linux Nedir ?

Başlamadan.. Şunu bilin ki Linux tek başına işletim sistemi değil çekirdektir!

UNIX

Burada sizlere çok tarihi veya çok teknik olmayan bir üslupla, linux hakkında genel bilgi sahibi olmanızı sağlayacak şekilde Linux'un ortaya çıkış hikayesini anlatacağım. Zamanında Bell Laboratuvarı'nda geliştirilen UNIX adında bir işletim sistemi vardı. Bu sistem kendi bünyesinde bulundurduğu araçları sayesinde üniversite ve diğer ihtiyaç duyulan yerlerde kullanılıyordu. Buraya kadar her şey gayet güzel ancak Unix lisans ücreti istiyordu ve bu durum çoğu kullanıcının hoşuna gitmedi. Bunun sonucunda "para ödemek yerine bende kendi işletim sistemimi yazarım" diye düşünenler ortaya çıktı ve bu da UNIX'e mecbur olunmadığı eğer istenirse yeni bir sistemin yazılabileceği düşüncesini ortaya koydu ve bir akımı başlattı. Bunun sonucunda birçok işletim sistemi geliştirme girişimleri oldu ancak sonuçta geliştirilen sistemler stabil şekilde çalışmıyorlardı. Tabi bu durum sonsuza kadar böyle kalmayacaktı..

Linus Torvalds

Bir efsanenin doğuşu..

Buraları her ne kadar istemesem de kısaca geçmek durumundayım ama eğer "Linux kullanıyorum" demek istiyorsanız kesinlikle bu konuları araştırın ve ayrıca [Revolution OS](#) belgeselini de izlemeyin derim.

İnsanların lisans ücretinden kaçmak için kendi işletim sistemini yazma akımı devam ediyorken bu durum [Linus Torvalds](#)'ı da etkilemiş olacak ki Linus,UNIX'ten esinlenerek Helsinki Üniversitesindeyken kendi işletim sistemi çekirdeğini yazdı. UNIX'ten esinlendi ancak bunu hiçbir kod alıntısı yapmadan sıfırdan yazarak başardı. Ve bu geliştirdiği çekirdeğin insanlara(topluluklar vs..) ulaşmasını sağladı ve insanların verdiği dönütlerle düzenlemeler yaparak çekirdeğin daha da kullanışlı hale gelmesini sağladı. Ve bu geliştirdiği çekirdeğe(kernel) "[Linux](#)" adını verdi. Ancak Linux bir çekirdekti ve içerisinde UNIX gibi araçları bulundurmuyordu.

Linux cephesinde bunlar yaşanırken [Richard Stallman](#) , GNU genel kamu lisansını ortaya atar ve [özgür yazılımı](#) savunur.Nedir bu genel kamu lisansı ve özgür yazılım diyecek olursanız lütfen okumaya devam edin.

GPL

GNU GPL (General Public kısıyollar - Genel Kamu Lisansı) açık kaynağı destekleyici bir lisans türü olup, ilk defa açık kaynak kodlu yazılımların kapalı kaynak kodlu hale getirilerek satılmasından rahatsız olunmasıyla öneri olarak geliştirilmiştir.

GPL 4 temel özgürlük üzerine kurulmuştur bunlar:

- **Özgürlük 1:** Programı sınırsız kullanma özgürlüğü.
- **Özgürlük 2:** Programın nasıl çalıştığını inceleme ve amaçlara uygun değiştirme özgürlüğü.
- **Özgürlük 3:** Programın kopyalarını sınırsız dağıtma özgürlüğü.
- **Özgürlük 4:** Programın değiştirilmiş halini dağıtma özgürlüğü.

GPL Richard Stallman tarafından geliştirilmiş çok akıllıca detaylarla bağlayıcılığı bulunan lisans türüdür. Burda herkes

kazançlıdır.[Kaynak kodları](#) paylaşmak zorunda olduğu için her şey şeffaf ve oldukça berraktır .Burada insani bir hizmet vardır. Yani savunulan şey daha fazla rant değil, insanların gönül rahatlığı ile sınır olmadan ihtiyaçlarını karşılamasıdır. Bu konu hakkında ayrıca avantajları ve dezavantajları sıralayabiliriz ancak konuyu çok genişletmemek ve sizlere araştırma kendi kendine bulup öğrenme alışkanlığı kazandırmak adına bu konuyu burada kesiyorum. Ben yazmaktan üşenmiyorum ancak benim burada yazıyı genişletmem sizin araştırıp kendin kendinize öğrenmenizden ve bu araştırma alışkanlığını kazanmanızdan daha yararınıza olmayacak lütfen buna inanın. Birde benim konu içinde veya dışında belirtmiş olduğum kaynaklarla sakın ola sınırlı kalmayın. Ben bir başlangıç noktası ve referans olması adına sizlerle bunları paylaşıyorum. Benden bu yüzden nefret edebilirsiniz ancak, araştırma ve kendi kendine öğrenme ateşi sizi sardığında hak vereceğinizi umuyorum. Sözümlü o çok bilindik bir çin ata sözü ile noktalyorum.

"Bir kişiye iyilik yapmak istiyorsan ona balık verme, balık tutmayı öğret." Konfüçyüs

GNU

GNU, açık kaynak hareketinin doğrultusunda geliştirilen, araçların içinde bulunduğu ücretsiz ve herkes için ulaşılabilir bir işletim sistemidir. Ancak GNU için her şey her daim güllük gülistanlık değildi. GNU içinde kendine ait güçlü araçlarını bulundurmasının yanı sıra kararlı bir çekirdeğe sahip değildi. Çekirdek için denemeler yapılmış ancak kararlı bir çekirdek oluşturulamamıştı.

Tam da bu noktada Linus Torvalds'ın geliştirip topluluğa sunduğu ve topluluk doğrultusunda geliştirdiği çekirdek yazılımı, GNU ile tencere kapak uyumu olacak şekilde bir araya geldi ve ortaya bir GNU/Linux efsanesi çıktı. İşte GNU/Linux devrimi kısaca bu şekilde oldu. Hem GNU'nun hem de Linux'un topluluğa bağlı dönütlerle geliştirilmesi doğrultusunda herkese açık, ücretsiz ve sınırsız geliştirilme potansiyeline sahip gerçek anlamda insana hizmet eden bir işletim sistemi ortaya çıkmış oldu.

Sonuç Olarak

Linux bir işletim sistemi değil çekirdektir(kernel) GNU/Linux bütünü bir işletim sistemidir. Ancak genellikle söylenmesi uzun veya zor geldiği için zamanla sadece Linux olarak geçmeye başlamıştır. Ben de yazımda GNU/Linux yerine Linux kullanıyor olacağım. Ancak emin olun çekirdekten(kernel) yani Linux'tan bahsettiğimde ve GNU/Linux(işletim sistemi)'tan bahsettiğimde hangisini kastettiğimi anlıyor olacaksınız.

Gerekli Ortamın Kurulması

Linux kurmak veya kurmadan kullanmak için çok fazla metod bulunuyor. Ben sadece içlerinde bilmediğiniz bir alternatif metod olması ihtimaline karşı genel kullanımları aşağıda listeliyorum. Şimdi neden kurulum detaylarını anlatmıyorsun diyecek olursanız, burada izahı dökümantasyonu uzatacak ve resim kullanmadığım için çok da verimli olmayacaktır. Siz en iyisi hem bana destek olmak hemde de kurulumları videolu şekilde takip etmek için [buradan](#) kursumu satın alarak devam edin. Ancak hemen önyargıya kapılıp sinirlenmeyin lütfen sizler bu kursu almadan da internet üzerinden araştırarak çok fazla kaynağa ulaşabilirsiniz. Fakat benim videolarımın artısı buraya bağlantılı ve açıklık getirici şekilde ilerliyor olması. Yani tamamen sizlerin isteğine bağlı bir durum.

Kurulum ve Kullanım Metodları

- Sanal olarak kurulum (Vmware & Virtualbox)
- İkincil işletim sistemi olarak kurmak (Dualboot)
- Live versiyon olarak kullanmak.
- Linux VPS aracılığı ile kullanmak.

Komut Satırı

Esaslı bir giriş yapabilmemiz için öğrenmemiz gereken ilk kavram Linux Komut Satırı diğer bir adıyla Linux Terminali (konsol) olacaktır. Ancak bundan önce Linux çekirdeğini (kernel) ele almalıyız. Linux çekirdeği yani kernel Linux'un kalbi kabul edilir. En önemli ana görevinin tanımı kısaca yazılımla donanımın haberleşmesini sağlamaktır. Ayrıca Linux'un yani çekirdeğin İngilizce karşılığı "kernel"dir ben de gerektiğinde bu isim ile kullanacağım. Yabancı terimleri kullanma nedenim eninde sonunda bu terimlere alışmamız gerektiği ve bu durumun ne kadar erken olursa bizim için o kadar iyi olacaktır.

Kernel (çekirdek)'i biraz daha açıklayacak olursak:

Kernel

Biz ister grafiksel arayüzü kullanalım istersek de yalnız komut satırını kullanacak olalım; örneğin bir dosyayı bir yerden başka bir yere sürükleyerek taşırsak ya da komut satırından komutlar yardımı ile bu işi gerçekleştirecek de yapılan işlem aslında arka tarafta komutların yorumlanarak çalıştırılması ile gerçekleşmektedir. Bu işlemleri gerçekleştirmekle görevli bazı yapılar vardır. Çekirdek(kernel) de kullanıcıdan gelen girdilerle birlikte sistemin işleyebilmesi (process) için donanıma iş yaptırmakla görevlidir. Ancak direkt olarak kullanıcıdan alınan komutlar Kernel'e geçmez. Bundan önce komut satırı dediğimiz bir kabuk(shell) programını temel alarak çalışan bir yapı, kullanıcı ile çekirdek arasında aracı bir katman görevi görür.

Shell (kabuk) programını açıklayacak olursak:

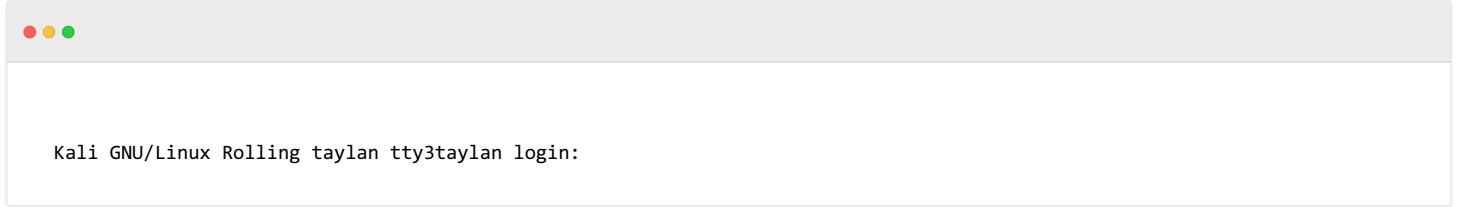
Shell

Mantığını oturtmanız ve kavramların daha kolay yerleşmesi açısından bir fısıktı düşünün. Bu fısıktının dış katmanı Kabuk (Shell) iç kısmı ise Çekirdek (Kernel)'olacaktır. Bu benzetme sayesinde isimlerin de nereden geldiği az çok açıklığa kavuşmuş oldu. Shell'in görevi bir üst kısımda belirttiğimiz gibi kullanıcı ile kernel arasında aracı bir katman olarak kullanıcıdan gelen girdileri kernel'e uygun şekilde iletmektir. Sistemde yapmak istediklerimiz ve yapabileceklerimiz Shell'in esnek ve kullanışlı yapısına yani yeteneklerine bağlı olarak değişmektedir. Bu bağlamda Shell'in sisteme hükmetmekteki anahtarlardan olduğunu söyleyebiliriz. Shell hakkındaki biraz önceki açıklamalara("yeteneklerine bağlı olarak değişmektedir") bakarak birden fazla Shell programının olduğunu tahmin etmiş olabilirsiniz. Tebrik ederim çok doğru, Linux sistemlerinde BASH dışında (ksh,tcsh,zsh,fish...) gibi birçok Shell (kabuk) programı mevcuttur. Ancak yetenekleri dolayısıyla olsa gerek en çok tercih edilen kabuk programı BASH kabuk programıdır. İnanmıyorsanız bir test edelim sizin sisteminizde hangi kabuk programı kullanılıyor.

Bunun için konsolunuzu açın ve aşağıdaki komutu girin diyecektim ki... "Hayda.. daha konsol nedir açıklamadın ki !" diyeceksiniz haklısınız temeli oturtmak adına buralara kadar geldik ama direkt konsol şudur demedik. Ancak zaten bu kısma kadar Konsol'un ne olduğunu dolaylı olarak da olsa açıklamış olduk.

Konsol, kullanıcı ile Shell arasından yer alarak kullanıcının komut girmesini sağlayan grafiksel ve komut satırı arayüzüne sahip bir araçtır. Biz komutlarımızı bu araç aracılığı ile Shell'e ulaştırırız Shell ise kullanıcıdan gelen girdileri yorumlayarak Kernel'e aktarır.

Konsolu daha yakından incelemek adına bir konsol ekranı açalım. Yeni bir konsol ekranı açmak için kısayol tuşları olduğu gibi grafik arayüz aracılığı ile de açmak mümkün. Linux bize çok sayıda konsol açma imkanı tanır. Aynı anda 6 ayrı konsol açıp hepsinde ayrı işlem yapabiliriz. Bunu kanıtlamak istersek..(Anlatımlar Kali linux işletim sistemi üzerinden gerçekleştirilmektedir. Dolayısıyla anlatım sırasında anlatılanların uygulanması noktasına gelindiğinde sizlerden kali linux işletim sistemini halihazırda açık ve kullanıma hazır şekilde bekletiyor olmanız beklenmektedir.) Grafiksel arayüzden komut satırına geçmek için **Ctrl + Alt + (f1,f2,f3,f4,f5,f6)** tuş kombinasyonunu kullanabiliriz. Demiştim ya 6 farklı konsol açılabilir işte açtığınız konsolun numarası da **tty1,tty2,tty3,tty4,tty5,tty6** gibi "tty_konsol_numarası" şeklinde konsolda görülüyor. Örneğin ben **Ctrl + Alt + f3** tuş kombinasyonunu yaptığımda karşıma aşağıdaki gibi benzer bir komut satırı geliyor ve benden login olmamı yani kullanıcı adı ve sonrasında şifre yazarak giriş yapmamı bekliyor.

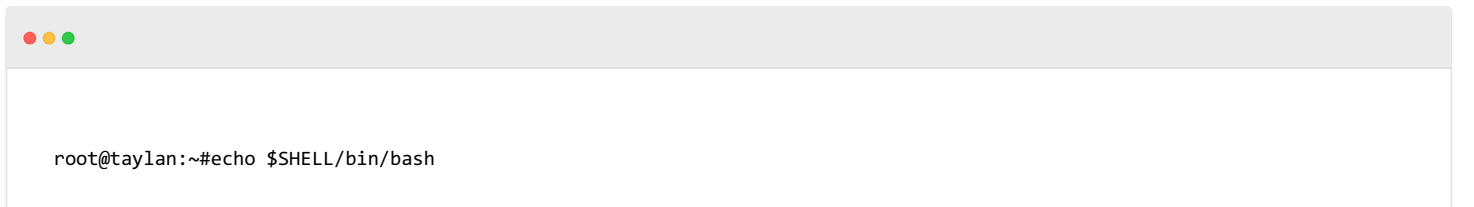


```
Kali GNU/Linux Rolling taylan tty3taylan login:
```

Eğer bu komut satırından çıkıp tekrar kullanıcı arayüzüne yani grafiksel arayüze geçmek isterseniz **Ctrl + Alt + (f7,f8,f9,f10,f11,f12)** kombinasyonlarından herhangi birini kullanarak bunu başarabilirsiniz. Ancak kimi Kali Linux sürümlerinde durum böyle olamayabiliyor yani eğer **Ctrl + Alt + f7** ve sonrası tuş kombinasyonu grafik arayüze dönmeyi sağlamaz ise bu sefer **Ctrl+Alt+f1** veya **Ctrl + Alt + f2** şeklinde dönmeye çalışın. Grafik arayüzden Terminal(konsol) ile çalışmak isterseniz masaüstünde bulunan Terminal simgesine tıklamanız yeterli olacaktır.(Genelde görev çubuğunda sabit şekilde yer alır.) Eğer masaüstünde yok bulamıyorum diyorsanız arama çubuğu ve benzeri yardımcı araçları ile lütfen "konsol" veya "terminal" şeklinde aratarak kendiniz bulun. Ayrıca kısayollardan bahsetmiştim örneğin bazı dağıtımlarda **Ctrl + Alt + T** tuş kombinasyonu direkt olarak terminal ekranını açıyor. Ancak bu kısayollar ayarlar aracılığı ile ve ayrıca da dağıtımdan dağıtıma farklılık gösterdiği için bu kısayolları sizin kendi kullandığınız dağıtıma göre araştırmanız gerek. Korkmayın inanılmaz kolay bir işlem, hem böylelikle yavaş yavaş Linux öğrenmenin aslında araştırmak ve kendi kendine öğrenmek olduğunu öğreniyoruz. Zaten ileride Kali Linux için [kısayollar](#) konularına ayrıca değineceğim.

Şimdilik konumuza dönecek olursak Shell'in ne olduğunu açıklamıştık ve en çok tercih edilen kabuk programının BASH kabuk programı olduğunu söylemiştik.Bunu teyit etmek için komut satırına **echo \$SHELL** komutumuzu giriyoruz.

Girdiğimiz komutu açıklayacak olursak ' **echo** ' komut satırında ekrana yazdırma işlevini görür örneğin ben komut satırına **echo "Merhabalar"** yazarsam komut satırı ' Merhabalar ' çıktısını verecektir.' **\$** ' işareti bir parametre ve tanımlanan değişkene normal bir kullanıcı olarak erişebilmeyi sağlıyor. Şimdilik bu komut bütünü hakkında bu kadar bilgi yeter. Eğer komutun açıklamasını anlamadıysanız sorun yok. Okumaya devam ettikçe parçalar yerine oturmaya başlayacak. Zaten ileride bu konuların her birine tek tek değineceğiz ve sizler de anlamış olacaksınız. Şimdi devam edecek olursak:



```
root@taylan:~#echo $SHELL/bin/bash
```

Konsola girmiş olduğumuz komut yorumlandı ve neticesinde bize **/bin/bash** şeklinde bir çıktı verdi. Bu çıktı bize mevcut sistemde kullanılan ana kabuk programının BASH olduğunu bildirmiş oldu. Burada komut yorumlandı dedik ancak neye göre oldu bu iş ? İşte bu noktada devreye ortam değişkenleri(çevre değişkenleri) giriyor. Bu kısımları fazlaca incelediğimi düşünenler olabilir ancak emin

olun öğrendiğiniz her şey birbiri ile bağlı ve bütünü anlamak için bilmek gerekli. İleride yeri geldikçe bunları daha net kavrayacağız zaten.

Anlatımları mümkün oldukça teknik dilden uzak sade anlaşılır ve sizlerin de sürekli etkileşimde olacağı şekilde aktarmaya çalışıyorum. Yani burada yazanları kuru kuruya ezberlemeyecek, ilerleyiş sırasında uygulamalar yaparak konuları anlamış olacağız.

Tamda bu noktada uygulama yaparak öğrenmenin önemini belirten Konfüçyüs'ün bir sözünü hatırlatmak istiyorum:

"Bana anlat unuturum, bana göster hatırlarım, beni dahil et, anlarım." Konfüçyüs

İlerleyişimize gelin ortam değişkenlerini tanıyarak devam edelim.

Ortam Değişkenleri (Çevre Değişkenleri)

Ortam değişkenlerini anlamak için bir örnek üzerinden gidelim. Örneğin grafiksel arayüzde komut satırına `leafpad` yazdığımızda karşımıza leafpad(metin editörü) programı gelir. (Özellikle grafiksel arayüz diye belirttim çünkü leafpad uygulaması grafiksel arayüz ile çalışmaktadır yani tek başına komut satırının olduğu durumda çalışmayacaktır.)

Fark ettiyseniz bu komutu yazarken programın bulunduğu konumu belirtmeme gerek kalmadı. Yalnızca programın ismini Terminale girmemiz sonucu leafpad programı açılmış oldu. Bunun nedeni programın dosya dizinlerinin PATH(yol) ortam değişkeni üzerinde ekli olmasıdır. Sonuç olarak konsoldan girilen `leafpad` komutu bu yol üzerinde arandı ve yol üzerinde var olan leafpad uygulaması çalıştırıldı.

Daha iyi anlayabilmek adına bize PATH(yol) bilgisini veren komutumuzu kullanalım ve terminalin bize vereceği tepkiye göre açıklamaya devam edelim. Komut satırımıza `echo $PATH` komutunu verelim:

```
root@taylan:~# echo $PATH/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Çıktının bize gösterdiği yani konsolun demek istediği şu:

Konsoldan girilen komutu çalıştırabilmem için sırasıyla bu `/usr/local/sbin: /usr/local/bin: /usr/sbin: /usr/bin: /sbin: /bin` dizinlere bakmam gerek.Eğer verilen komutun çalıştırılabilir dosyası bu dizinlerin içerisinde ise çalıştırırım, yoksa çalıştıramam.

Çıktıda görülen iki nokta üst üste (:) işareti ile ayrılmış dizinlere PATH(yol) ortam değişkeni deniyor.Ben yinede emin olmadım diyenler için gelin PATH yoluna ekli olmayan bir programı kendimiz ekleyelim ve konsoldan verdiğiniz bir komutla direk olarak çalışmasını sağlayalım. Adım adım ilerleyelim.

Ben masaüstünde "yeni_dizin" adında bir klasör oluşturdum daha sonra içerisine "yeni" diye başka bir klasör daha oluşturdum son olarak bu klasörün de içerisine "en_yeni" şeklinde bir klasör daha oluşturdum. Yani oluşturduğum dizinin tam adresi `yeni_dizin/yeni/en_yeni` şeklinde oluşmuş oldu.

Şimdi oluşturduğum dizinin en alt klasörünün(en_yeni) içine girerek leafpad programı aracılığı ile metin belgesi oluşturuyorum ve belgenin içine `echo "Program Çalıştı"` yazarak oluşturmuş olduğum dosya dizininin en alt dosyaya yani "en_yeni" ismindeki klasöre dosyayı "komut.sh" ismiyle kaydediyorum. Dosya sonuna eklediğimiz `.sh` eki ile bir betik dosyası halini aldı. Betik dosyası genel tanımı ile konsol komutlarını içerisine kaydettikten sonra tek seferde bir bütün halinde komutları çalıştırabildiğimiz dosya

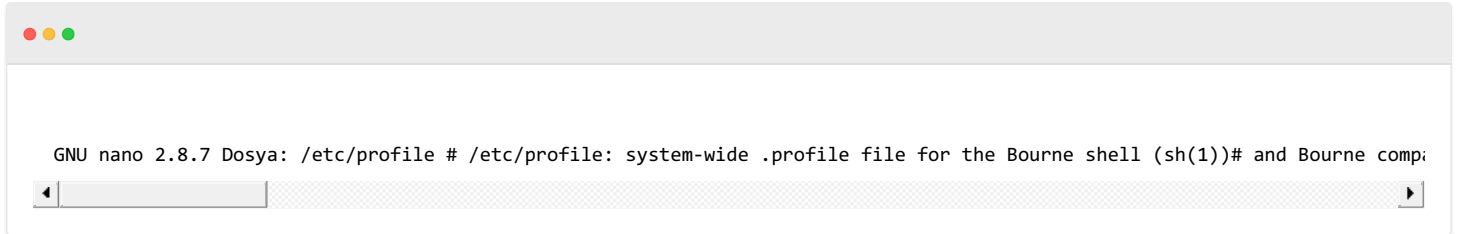
genel anlamıyla komut dosyalarının içeriğini kaydedtikten sonra tek olarak en alt dizinle komut dosyası çalıştıracağımız dosya türüdür. Zaten fark ettiyseniz dosyamızın içerisine daha önce de konsolda verdiğimiz komutlardan birini olan `echo` komutunu ekledik. Yani bu oluşturmuş olduğumuz betik dosyası Terminal üzerinden çalıştığında komut satırına "Program Çalıştı" şeklinde bir çıktı verecek. Betik dosyasının dizin içerisindeki tam konumu aşağıdaki şekildedir.



Programımızın dizin adresi belli olduğuna göre artık bu dizini PATH ortam değişkenine ekleyip istediğimiz zaman, istediğimiz yerden programımızı(komut.sh) çalıştırabiliriz.

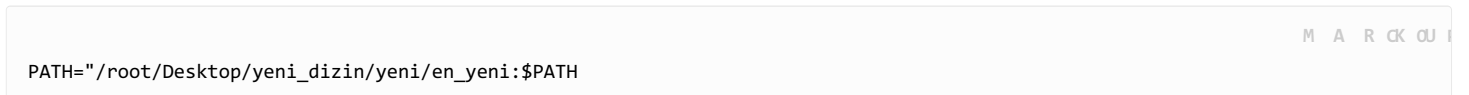
PATH(yola)Dizin Ekleme

PATH(yola) yeni dizin eklemek için öncelikle her defasında oturum başlatılırken okunan `profile` isimli dosyada değişiklik yapmalıyız ki eklediğimiz dizin her daim biz silene kadar geçerli olsun. `profile` dosyasında değişiklik yapmak için komut satırına dosyamızın bulunduğu tam konumu belirtmek üzere `nano -w /etc/profile` komutunu giriyoruz.



Ve komut satırında karşımıza yukarıdaki gibi `profile` dosyasının içeriği geliyor. Şimdi yapmamız gereken programımızın (komut.sh) bulunduğu tam dizin konumunu bu dosyaya uygun şekilde eklemek.

Bunun için dosyanın en alt satıra inerek eklemek istediğim dosyanın tam dizin adresini aşağıdaki şekilde dosyaya ekliyorum.



Yazma işlemi bittikten sonra dosyada yaptığımız değişiklikleri kaydedip dosyayı kapatmak için `Ctrl + X` tuş kombinasyonu uyguluyoruz. Bana, "Değiştirilen tampon kaydedilsin mi? ("Hayır" demek değişiklikleri SİLECEK.)" şeklinde bir uyarı veriyor. Dosyada yaptığım değişikliği kaydetmek için `e` tuşuna basarak devam ediyorum. Son olarak bana, "Yazılacak Dosya Adı: /etc/profile" diye belirtiyor, buradaki ismi değiştirmeden kaydediyoruz. Eğer ismi değiştirirsek sistem bu dosyayı okuyamacağı için problem çıkacaktır.

Bu adımları geçtikten sonra istediğimiz dizin PATH(yol) ortam değişkenine eklenmiş oldu. Ancak işlemin geçerli olması için oturumun kapatılıp tekrar açılması gerekiyor çünkü `profile` dosyası oturum açılırken okunuyor.

Oturumu kapatıp tekrar giriş yaptım. Şimdi sırada eklediğimiz dizinin PATH yolunda ekli olup olmadığını kontrol etmek var bunun için konsola `echo $PATH` komutunu veriyoruz.

```
root@taylan:~# echo $PATH/root/Desktop/yeni_dizin/yeni/en_yeni:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Gördüğümüz gibi dizin yola(PATH) eklenmiş bulunuyor.

Artık dosyayı çalıştırmadan önce son bir ayar daha yapmamız gerek. Daha sonra "komut.sh" dosyamızı aynı leafpad programında olduğu gibi istediğimiz zaman komut satırından ismi ile çalıştırabiliyor olacağız. Son işlemimiz yetkilendirme işlemi. Bu neden gerekli diyecek olursanız işlemin gerekliliğini kendi gözlerinizle görmek için bu kısmı atlayarak dosyayı çalıştırmaya çalışın. Bunun için komut satırına `komut.sh` yazalım.

```
root@taylan:~/Desktop/yeni_dizin/yeni/en_yeni# komut.shbash: /root/Desktop/yeni_dizin/yeni/en_yeni/komut.sh: Erişim engelli
```

Gördüğümüz gibi konsol bize "Erişim engellendi" diye bildirdi ve dosyayı bulmasına rağmen çalıştırmadı. İşte bu durumu yaşamamak için "komut.sh" dosyamızın bulunduğu konuma giderek (`cd Desktop/yeni_dizin/yeni/en_yeni/`) `chmod +rwx komut.sh` komutunu vermemiz gerekiyor. Böylelikle dosyayı çalıştırma izni de elde etmiş olacağız. Yani artık konsola `komut.sh` yazdığımız anda bizim oluşturmuş olduğumuz "komut.sh" betik dosyası çalışarak "Program Çalıştı" yazısını konsola basacak.

```
root@taylan:~# chmod +rwx komut.shroot@taylan:~# komut.shProgram Çalıştı
```

Sizler de bu yolla sıklıkla kullandığınız dosyaları kolay erişilebilir kılabilirsiniz. Burada vermiş olduğum komutları ayrıntılı şekilde ileride açıklayacağım siz şimdilik konunun temeline ve ne neden oluyor ona hakim olsanız yeter. Ancak kafalarda soru işareti ile devam etmemek adına ben yine de tek tek izah edeyim hangi komutu neden kullandık.

`nano -w /etc/profile` bu komutta; `nano` komutu, konsol üzerinden dosya içeriğini okumamıza olanak sağlayan bir araç ve `nano` komutunun yanındaki `-w` karakteri ise dosyada değişiklik yapmamıza olanak sağlayan parametredir. Biz bu parametre sayesinde nano aracı ile belgeyi yazma(write) kipinde açmış oluyoruz. Geri kalan `/etc/profile` kısmı ise `profile` dosyasının bulunduğu konumu belirtiyor. Bu sayede nano aracı ile yazma kipinde `/etc/profile` konumundaki `profile` dosyasını komut satırı üzerinden görebiliyor ve değişiklik yapabiliyoruz.

`PATH="$PATH:/root/Desktop/yeni_dizin/yeni/en_yeni:$PATH"` ifadesi ise mevcut PATH(yola) yeni dizin eklememize olanak sağlayan

PATH = /root/Desktop/yeni_dizin/yeni_ ~~en_yeni~~.txt ifadesi ise moved PATH (yeni) yeni dizin sistemimize olanak sağlayan bir bütündür. Burada açıklanacak özel bir durum yok kullanım şekli itibari ile PATH yoluna dizin ekleme işlemi için gereken ifade bütünüdür.

Mesela bende `/home/taylan/Desktop` olan masaüstü dizini sizde `/home/burak/Desktop` şeklinde olabilir. Hatta işletim sisteminin dilinin Türkçe veya İngilizce (diğer tüm diller de dahil..) olmasına göre de masaüstünün yolu farklı olabilir. Yani demem o ki bu kadar basit bir işlem için bile tahmin edilemez bu yolun bilinmesi veya bulunması gerek. Ama masaüstüne kısayol oluşturma çok olağan bir durum bir imkansız değil. İşte tam bu noktada neden ortam değişkenlerinin var olduğunu biraz daha net kavırıyoruz. Ortam değişkenleri, kullandığımız işletim sisteminde belli değerlere daha sonra kolaylıkla ve kararlılıkla ulaşabilmemiz ve işleyebilmemiz için oluşturulan ifadelerdir. Bahsettiğimiz konuda örneğin masaüstüne herhangi bir yerden ulaşmak istersek, komut satırına ev dizini yolunu tutan değişken olan HOME değişkenini `cd $HOME` şeklide girmemiz gerekiyor. Bu sayede ev dizini içerisinde yer alan Desktop konumuna da ulaşabiliyor olacağız.

Ayrıca burada ekstra bir bilgi olsun diye burada belirtmek isterim ki ev dizinine gitmek için iki farklı yol daha bulunmaktadır. Bunlardan ilki yalnızca `cd` komutunu kullanmak diğeri ise `cd ~` komutunu kullanmaktır. (Burada yer alan `~` işareti Türkçe klavyede **Alt Gr + ü** tuş kombinasyonu ile oluşturuluyor.)

```
root@taylan:/# cd $HOMEroot@taylan:~# lsDesktop Documents Downloads Music Pictures Public Templates Videos
```

Komutumuzu girdikten sonra ulaştığımız konumda hangi dosyaların olduğunu görmek için konsola `ls` komutunu verdik. Konsol bize Desktop(masaüstü) ve ev dizininde bulunan diğer dosyalarında çıktılarını verdi. Yani bu demek oluyor ki ev dizinine kullanıcı ismini bilmemize gerek olmadan kolayca ulaştık. Örnek durumda açıkladığımız program kısayolu oluşturma işlemi ana dizini tutan HOME ortam değişkeni sayesinde kolayca gerçekleştirilebilir oluyor.

Komutu biraz açıklayacak olursak `cd` dizinler arası geçiş yapmamızı sağlayan bir komut `$` (dolar işareti) tanımlanmış olan değişkene erişme işlemini yapan parametremiz, HOME ise ev dizinini tutan ortam değişkeni. Son olarak `ls` komutu bulunduğumuz konumdaki dosyaları listeliyor. Burada bahsi geçen tüm komutları ileride ele alacağız şimdilik bu açıklama yeterli.

Temel ve sık kullanılan değişkellerden birkaçını da kısaca açıklayacak olursak:

SHELL: Çalışmakta olan kabuk programının adını ve yeri.

PATH: Konsola komut girildiğinde, komut için gereken ilgili dosyaların aranacağı dizinler diğeri bir adıyla yol.

HOME: Kullanıcının ev dizinini içeren değişken.

TERM: Komut satırı uygulamalarının hangi terminalde çalışacağını belirtir. Birçok çeşidi vardır ancak xterm yaygın şekilde karşımıza çıkmaktadır.

Ortam değişkenlerinin değerlerini tek tek kontrol etmek istersek, konsola `echo $ORTAM_DEĞİŞKENİ` şeklinde komut vererek ilgili bilgilere ulaşabiliriz. Örneğin TERM değişkenine bir bakalım.

```
root@taylan:~# echo $TERMxterm-256color
```

Gördüğünüz üzere konsol TERM değişkeninin değerini xterm olduğunu göstermiş oldu. Zaten daha önce yaygın olarak

kullanıldığından söz ettiğimiz için bu çıktı bizi şaşırtmadı. Eğer bu çıktıyı teyit etmek istersek konsola `stty -a` komutunu verdiğimizde

kullanılabilir. Bu özelliği kullanmak için bu çıktıyı teyit etmek istersek konsola `set` komutunu verdiğimizde xterm açılacak ve çıktı teyit edilmiş olacaktır.

İstersek diğer ortam değişkenleri için de aynı şekilde tek tek bilgi alabiliriz. Ancak ortam değişkenleri sabit ve sınırlı değildir. Sistemde var olanların haricinde bizde kendimiz ortam değişkenleri tanımlayabiliriz. Tanımlamaları üç farklı kategoriye ayırabiliriz;

- **konsola(terminal) özel** :yalnızca geçerli, açık olan terminaldeki uygulamalar için kullanılabilen
- **kullanıcı** :yalnızca tek kullanıcı için geçerli ve o kullanıcının her oturum açtığında kullanabildiği
- **sistem geneli** :sistemde öntanımlı olarak ayarlanmış sürekli kullanılabilir

Gelin şimdi teker teker kullanımlarına değinelim.

Mevcut Konsola Özel

Sadece o an kullanılmakta olduğumuz terminale(konsol) özel olan ve terminali kapattıktan sonra sıfırlanan ortam değişkenidir. Aşağıdaki örnek ile adım adım açıklayalım bu durumu:

Terminali açtım ve komut satırına " asdf " yazdım. Konsol yanıt olarak " bash: asdf: komut yok " yazısını ekrana bastı.

```
root@taylan:~# asdfbash: asdf: komut yok
```

Çıktı görüldüğü gibi konsol çıktısı Türkçe oldu ancak ben bu çıktıları İngilizceye çevirmek istiyorum diyelim. Bunun için konsola `export LANG=C` komutunu veriyorum. Ve sonrasında tekrar komut satırına `asdf` yazıp dilin değişip değişmediğini kontrol ediyorum.

```
root@taylan:~# export LANG=Croot@taylan:~# asdfbash: asdf: command not found
```

Çıktıda da görüldüğü gibi dil değişmiş ve yazdığımız `asdf` komutunun neticesinde konsol, ekrana "bash: asdf: command not found" yazısını basmış oldu. Sonuç olarak geçerli terminal için dil İngilizceye çevrilmiş oldu.

Bütün bu işlemler sadece bu komut penceresi için sınırlı. Yani açmış olduğumuz terminal ekranını(konsol/komut pencersi) kapatırsak yeni bir tane açtığımızda başlangıçta olduğu gibi konsol bize Türkçe yanıt verecektir. Yani bu şekilde yapılan tüm değişiklikler sadece yapıldığı konsol için geçerli. Bunu kendiniz de deneyerek görebilirsiniz.

Kullanıcıya (Oturuma) Özel

Daha önce BASH kabuğundan bahsetmiştik ve mevcut sistemimizde bulunup bulunmadığını da kontrol etmiştik. Bash her oturum açtığımızda tüm ayarlarını ve davranışlarını " .bashrc " isimli gizli bir dosyadan okur. Ufak bir bilgi daha başında (nokta) olan

dosyalar gizli dosya görevindedir. Konuya dönecek olursak bizim mevcut oturumumuzda ortam değişkenlerinde kalıcı değişiklik yapabilmemiz için yapmak istediğimiz değişiklikleri " .bashrc " isimli dosyaya eklememiz gerekiyor ki oturum açtığımızda sistem burada yaptığımız değişiklikleri her seferinde görebilsin.

Bulduğumuz oturumda kalıcı değişiklik yapmak için birisi grafiksel diğeri ise sadece komut satırı arayüzü olmak üzere iki farklı yoldan nasıl değişiklik yaparız onu görelim. İsterseniz ilk olarak grafiksel arayüz ile başlayalım. Öncelikle `.bashrc` dosyasını açmalıyız. Dosyayı açmak için konsola `leafpad ~/.bashrc` komutunu veriyoruz.

```
root@taylan:~# leafpad ~/.bashrc
```

Kodu kısaca açıklayacak olursak `leafpad` sistemde mevcut bulunan basit metin düzenleyicisinin adıdır, `~` (Alt Gr + ü kombinasyonu ile oluşturulan "tilde" karakteri) karakteri ise ev dizinini temsil ediyor `/.bashrc` ise düzenleme yapacağımız dosyanın adıdır. Bu kısa açıklama sonrası komutları tam olarak anlamamış olabilirsiniz. Ancak yakında her birine değineceğiz ve bu kısımlar da tam anlamıyla oturmuş olacak. Yani şimdilik bu kodlara çok takılmadan asıl anlatılmak istenilene odaklanın lütfen.

`leafpad ~/.bashrc` komutunu verdikten sonra karşınıza `.bashrc` dosyasının açılmış olması gerek. Şimdi yapmak istediğimiz değişikliği dosyanın en alt satırına yani dosyanın sonuna eklemeliyiz. Bu eklemeyi `export DEĞİŞKEN_ADI=değeri` şeklinde yapıyoruz. Ben örnek olması açısından dil değişikliğini ele aldım. Bu sebepten dil değişimi(ingilizce olan dili Türkçeye çevirmek) için gerekli olan yazı dizisini `export LANG=TR` şeklinde dosyanın sonuna ekledim ve dosyayı kaydederek kapattım.

Geldik diğer yöntem olan yalnızca konsol ekranını kullanarak değişiklik yapmaya. Bunun için konsola `nano -w ~/.bashrc` komutumuzu veriyoruz.

```
root@taylan:~# nano -w ~/.bashrc
```

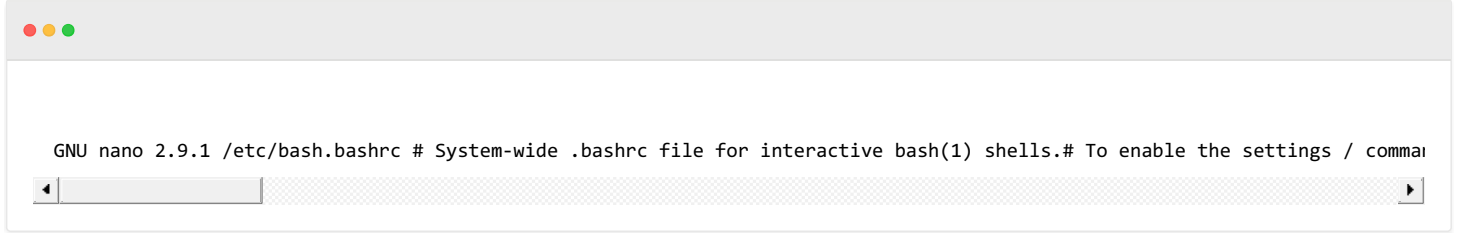
Komutumuzu girdikten sonra karşımıza konsol ekranı içerisinde `.bashrc` dosyasının içeriği geliyor. (`.bashrc` dosyasının içeriği çok uzun olduğu için yukarıdaki çıktı vermek istemedim.) Klavyedeki yön tuşlarını kullanarak en alt satıra iniyoruz ve oraya yapmak istediğimiz değişikliği giriyoruz. Ben dilde değişiklik yapacağım için satırın sonuna `export LANG=TR` şeklinde eklemeyi yaptım ve `Ctrl + X` tuş kombinasyonunu uyguladım. Daha sonra konsol bana çıkmadan önce dosyada yaptığım değişikliği kaydetmek isteyip istemediğimi sordu bende "yes" diyerek dosyanın adını herhangi bir değişime uğratmadan kaydettim ve işte bu kadar işlem tamamdır.

Yalnızca komut satırından işlem yapmak sanki daha uzun ve zor gibi gelmiş olabilir ancak kesinlikle grafiksel olarak yapmaktan bir farkı yok sadece belki izahı biraz uzun sürmüş olabilir.

Artık böylelikle değişiklik yaptığımız bu kullanıcı oturumunu ne zaman açarsak, yaptığımız dil değişikliği geçerli olacak. Ancak değişikliği yaptığımız anda etki etmiyor. Bunun nedeni `.bashrc` dosyasının oturum açılırken okunmasıdır. Yani yaptığımız değişikliklerin geçerli olabilmesi için oturumun kapatılıp tekrar açılması gerekmektedir. Ya da konsoldan vereceğimiz `source ~/.bashrc` komutu da dosyanın tıpkı oturum yeniden açılıyomuşcasına `.bashrc` dosyasının tekrar okunup yapılan değişikliklerin geçerli olmasını sağlar.

Sistem Genelinde

Eğer yaptığımız değişiklik bütün kullanıcı oturumlarında aynı şekilde geçerli olsun istiyorsak değişilen sistemde her oturum açıldığında okunan bir dosyada bulunması gerekmektedir. İşte bizim de istediğimiz şey tüm kullanıcılar için geçerli olsun istiyorsak, yapacağımız değişikliği `bash.bashrc` dosyasına eklemeliyiz. Dosyamızın tam konumu `/etc/bash.bashrc` dizininde yer alıyor. Bu dosyada ilgili değişikliği nano aracı ile yapmak için konsola `nano -w /etc/bash.bashrc` komutunu veriyoruz. (Çıktı fazlaca uzun olduğundan üç nokta (...) ile aradaki karakterler kısaltılmıştır.)



Örnek olması açısından dosyamızın en son satırına dil değişikliği işlevindeki `"export LANG="fr_FR"` ifadesini ekliyorum. Bu sayede terminal dili bütün sistemdeki kullanıcılar için fransızca olacaktır.

İlgili ifadeyi `bash.bashrc` dosyamıza ekleyip dosyamızı kaydettikten sonra sıra geldi değişikliklerin sistem tarafından tanınmasına. Yapılan değişikliğin sistem bütününde geçerli olabilmesi için daha önce de bahsi geçen `source` komutunu `bash.bashrc` dosyası için `source /etc/bash.bashrc` şeklinde kullanıyoruz. Ve değişikliklerin geçerli olup olmadığını denemek için konsola olmayan geçersiz bir komut verelim, örneğin konsola `asdf` yazalım bakalım konsolun tepkisi hangi dilde olacak.



Çıktımız fransızca olduğuna göre başarmışız demektir. Yaptığımız bu değişiklik bütün kullanıcılar için yani sistem geneli için geçerlidir. Bu durumu başka bir hesap oluşturarak kendiniz de gözlemleyebilirsiniz.

Ayrıca değişiklikleri geri almak isterseniz eklediğiniz ifadeyi ilgili dosyadan silin ve sistemi `source ilgili_dosya_adi` şeklindeki komut bütünü ile yeniden konfigüre edin, bütün değişimler düzelmiş olacaktır.

Kısayollar

Bu bölümde birlikte sistem üzerinde gerekli hakimiyeti sağlamak için bize kolaylıklar sağlayan bazı kısayolları ve bazı önemli bilgileri öğrenip uygulayacağız. Kısayolları şimdi öğrenmemizin sebebi ileride komutları uygularken bizlere kolaylıklar sağlayarak bize hız katacak olmalarıdır. Hem şimdi öğrenirsek eğitim boyunca da sürekli pratik yapma imkanı bulmuş oluruz. Benim tecrübeme göre eğitim asla ezber üzerine kurulamaz. (aman M.E.B duymasın!) Öğrenme denilen olgu uygulama yani sürekli olarak yapılan pratik ile

gerçekleşir. Bu durumda öğreneceğiniz bilgiler beki başlangıç için taziaca gozukebilir. Ancak ner kısayolu ve bilgii direk ve surekli olarak kullanmasak bile burada olduğunu bilmeli ve gerektiğinde dönüp tekrar bakabilmeliyiz. O yüzden gözünüze fazla gözükürse endişe etmenize gerek yok. Zaten bunlar hayati olarak sayılmazlar ancak zamanla göreceksiniz ki işlerimizi kolaylaştırarak Linux sistemini verimli şekilde kullanmamızı sağlıyor olacaklar. Neyse bu kadar açıklama yeter gelin konumuza Terminal için kısayol atama işlemleri ile başlayalım.

Terminal Kısayolları

Genelde başka bir işletim sisteminden geçiş yapan arkadaşlar konsoldan bir komutu kopyalamak üzere, alıştıkları gibi **Ctrl + C** ve kopyalanan komutu yapıştırmak için ise **Ctrl + V** tuş kombinasyonunu denerler. Ancak komut satırında görürler ki kopyalamak için bastıkları **Ctrl + C** tuş kombinasyonu **^C** yapıştırmak için bastıkları **Ctrl + V** tuş kombinasyonu ise **^V** şeklinde konsola çıktı basmış. Tabi yanlış basmış olma ihtimaline karşı aynı işlemi genellikle tekrar deniyen arkadaşlar olur ancak yine aynı çıktıları alınca anlaşılır ki o işler öyle olmuyormuş :) İşin şakası bir yana insanın zamanla edindiği alışkanlıklarını değiştirmesi zor olabiliyor. O yüzden Terminalde **Ctrl + Shift + C** ve **Ctrl + Shift + V** olan kopyala-yapıştır kısayolunu daha önceden almış olduğumuz gibi değiştirmek bizim en doğal hakkımız. Bunun için eğer isterseniz yalnızca kopyala-yapıştır kısayolunu değiştirebilir yada diğer mevcut tüm kısayolları istediğiniz ve alıştığınız şekilde düzenleyebilirsiniz. Düzenlemek için, yeni Kali Linux versiyonlarında Terminal'in üst kısmında yer alan sekmelerden sırasıyla **Düzenle > Tercihler > Kısayollar** sekmelerini takip ederek açılan pencerede yer alan kısayolları dilediğiniz şekilde değiştirebilirsiniz. Eğer eski bir versiyon kullanıyorsanız dediğim adımlarla Terminal kısayollarına ulaşamamış olabilirsiniz. Ancak Terminal'in üst kısmında yer alan sekmelere göz atarak "kısayollar" penceresine ulaşabilirsiniz. Sonuçta sürekli yenilikler olduğu için her versiyona özel bu tarz şeyleri yazmak gereksiz olacaktır. Kendiniz de biraz kurcalama ile çok rahat keşfedebilirsiniz.

Terminalin kısayollar penceresine ulaştığımıza göre artık istediğimiz şekilde değişiklik yapabiliriz. Buradaki kısayol seçeneklerine göz atın hangisi size uymuyorsa dilediğiniz şekilde düzenleyin. Ayrıca bilmediğiniz kısayollar varsa onlara da göz atmayı ihmal etmeyin.

Sistem Kısayolları

Şimdi ise sistem üzerindeki kısayollara göz atalım. Ben sürekli Terminali kullandığımız için Terminali açmak üzere bir kısayol oluşturmak istiyorum. Bunun için adım adım yapmamız gerekenlere bakalım. Başlamadan önce, kullandığınız sürümüne göre değişiklik göstereceğinden yönerge tam olarak sizi yönlendiremeyebilir. Ancak bunlar biraz kurcalayarak kendi kendinize bulamayacağınız şeyler değil. Ben yine de kullandığım sürüm üzerinden sizlere adımları aktarıyorum:

Öncelikle sistem ayarları(Ayarlar) menüsüne gidelim oradan klavyeyi seçelim karşımıza "klavye kısayolları" şeklinde bir pencere açılacaktır.

Buradan en alta inerek altta yer alan " + " işaretine tıklayalım.

Karşımıza küçük bir "Özel Kısayol" penceresi açılacaktır.

Burada yer alan isim kısmına herhangi bir ad verebilirsiniz ben daha sonrasında değiştirmek istediğimde hatırlamada kolaylık olsun diye Terminal adını verdim.

Komut kısmına terminalin açılması için gerekli olan komutu girmeliyiz ben Terminali açmak istediğimden Terminali açma komutu olan **gnome-terminal** komutunu yazdım.

Daha sonra bir alt kutucuğa istediğimiz kısayol tuş kombinasyonunu giriyoruz.

Ben **Ctrl + Shift + T** şeklinde ayarladım. Elbette siz kısayol tuş kombinasyonunu dilediğiniz gibi ayarlayabilirsiniz.

Son olarak "Ekle" butonuna tıklayarak kısayolumu atmış oluyorum. Artık ne zaman atadığım kısayol tuş kombinasyonunu(**ctrl+shift+T**) uygularsam yeni bir Terminal ekranı açılıyor olacak. Ben burada Terminal üzerinden örnek verdim

ancak herhangi başka şeyler de olabilir o sizlere kalmış. Ayrıca bu kısayollar menüsünde yer alan kısayollara da göz atıp eğer isterseniz dilediğiniz şekilde değiştirebilirsiniz.

Bash Shell Kısayolları

Bash Shell programının ne işe yaradığını daha önceki kısımlarda açıklamıştık. Şimdi ise komut yazarken işimizi kolaylaştıracak olan bazı Bash Shell kısayollarını görecez. Burada verilenleri direk olarak kullanmanız beklenmediğini daha önceki kısımlarda söylemiştim. Kısayolların oturması için çokça pratik gerek. Pratik yaptıkça zaten zamanla istemeden de olsa öğrenmiş ve farkında olmadan kullanıyor olacağız emin olun.

İmleç Hareketleri:

Kısayol	Kısayol Açıklaması
Ctrl + A	imleç satır başına gider.
Ctrl + E	imleç satır sonuna gider.
Ctrl + P	önce çalıştırılmış komut gösterilir.
Ctrl + N	sonra çalıştırılmış komut gösterilir.
Alt + B	sola doğru(geri) bir kelime kadar imleç kayar.
Alt + F	sağa doğru(ileri) bir kelime kadar imleç kayar.
Ctrl + F	imleç bir karakter ileri gider.
Ctrl + B	imleç bir karakter geri gider.
Ctrl + XX	geçerli imleç konumundan, imleç satır başına geçer.

Düzenleme

Kısayol	Kısayol Açıklaması
Ctrl + L	ekran temizlenir ve imleç en üst satıra çıkar yani <code>clear</code> komutu ile aynı işlemi yapar.
Alt + D	imleçten sonraki kelimeyi siler.
Ctrl + U	imlecin solundaki her şeyi siler.
Ctrl + K	imlecin sağındaki her şeyi siler.
Ctrl + C	komutu keser.
Ctrl + Y	kesilmiş olan son metni ekrana yapıştırır.
Ctrl + W	imleçten önceki kelime panoya kopyalanır.

Kısayol	Kısayol Açıklaması
Ctrl + K	imleçten sonraki kelime panoya kopyalanır.
Ctrl + U	imleçten önceki kelime silinir.
Esc + T	imleçten önceki iki kelime yer değiştir.
Ctrl + H	sola doğru tek tek karakterleri siler.(Yani Backspace gibi davranır)
Alt + U	imleç in başladığı yerden sözcüğün sonuna kadar bütün karakterleri büyük harf yapar.
Alt + L	imleç in başladığı yerden sözcüğün sonuna kadar bütün karakterleri küçük harf yapar.
Alt + C	imleç in üstünde bulunduğu karakteri büyük harf yapar.
Ctrl + R	daha önce kullanılmış olan komutlar arasında arama yapma ve o komutu tekrardan kullanma imkanı sağlar.
Alt + R	değişiklikleri iptal eder ve satırı eski haline getirir.
Ctrl + _	değişiklikleri iptal eder.

Özel Tuşlar

Kısayol	Kısayol Açıklaması
Ctrl + I	Tab tuşu görevi görür.(Tab tuşu otomatik tamamlama sağlar, unutmanız veya hatırlamamanız halinde hayat kurtarıcı etk
Ctrl + J	Yeni satır (Newline)
Ctrl + M	Giriş/Onay (Enter)
Ctrl + [Escape(Esc) tuşu işlevi görür.
Ctrl + D	Terminali sonlandırır.

Burada belirttiklerim dışında pek çok kısayol mevcut ancak ben hepsine değinmedim. Kaynak olarak kullandığım [bu adresten](#) veya internette yer alan ücretsiz bir çok kaynaktan çok rahat şekilde yeni kısayol bilgilerine ulaşabilirsiniz.

Hazır konu kısayollardan ve pratiklikten açılmışken aynı anda birden fazla komut kullanımını da anlatmadan geçmek olmaz. Bunun için üç farklı yol izleyebiliriz. Birincisi yan yana olacak şekilde `&&` operatörünü kullanmak. Hemen bu yolu deneyelim. Ancak henüz tam olarak komutları öğrenmediğimiz için önceden kullandığımız komutlardan kullanalım. Bunun için PATH ve TERM ortam değişkenlerini kullanarak aynı anda çıktı almak üzere `echo $PATH && echo $TERM` komutunu konsola yazalım.

```
root@taylan:~# echo $PATH && echo $TERM/root/Desktop/yeni_dizin/yeni/en_yeni:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr,
```

Çıktıda görüldüğü gibi PATH ve TERM ortam değişkenlerinin değerleri yazdığımız birleşik komut sayesinde iki farklı komut aynı anda basıldı. Gelin şimdi de diğer bir yol olan noktalı virgül ";" kullanarak aynı işlemi tekrar edelim.

```
root@taylan:~# echo $PATH;echo $TERM/root/Desktop/yeni_dizin/yeni/en_yeni:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

Ve sonuç bizleri şaşırtmayarak yine birleşik komut içerisinde yazdığımız ortam değişkenlerinin değerlerini aynı anda ekrana bastırdı. Ve son olarak || operatörü var. Bu operatör diğerlerinden farklı olarak kullandığımızda, eğer verilen komutlardan birincisi başarısız olursa diğerini basar. Bu operatörü de kısaca şu örnek ile inceleyelim.

```
root@taylan:~# false || echo $TERMxterm-256color
```

Burada 1. komut başarısız olduğu için 2. komut ekrana basılmış oldu. Yani bir nevi önlem amaçlı bir kullanımdır. Biz konsola bu kullanım ile diyoruz ki; "Eğer verdiğim ilk komut başarısız olursa ikinci komutu çalıştır." yani bu durumda eğer ilk komut başarısız olursa hemen devreye ikinci komut girecektir ancak ilk komut başarılı olursa ikinci komut çalışmayacaktır.

Kısayol Koruması

Buraya gelene kadar bir çok kısayoldan ve bu kısayolların avantajlarından bahsettik ancak kısayollar bazen istemediğimiz sonuçlar doğurabiliyor. Biz de böyle durumlara karşılaşmamak adına şimdiden ufak önlemler alsak iyi olur. Önceki kısımlarda **Ctrl+D** kısayolunun mevcut konsolu kapattığını öğrenmiştik. Bu çok kullanışlı bir kısayol olsa da bazen istemeden de olsa konsolun ansızın kapanmasına neden olabiliyor. Yani bir kontrol mekanizması oluşturmamız sonradan üzülmemek için şart. Bu kontrol sağlama imkanını bize ignoreeof verir. ignoreeof tanımını kullanmanın 2 farklı yolu vardır. Bunlardan birisi sadece mevcut Terminal için geçerli diğeri ise sürekli ve oturumdaki tüm terminallerde geçerli olmak üzere kullanılmasıdır. Sürekli ve tüm Terminal ekranlarında geçerli olsun istiyorsak. Gerekli ayarlamaları yapmak üzere komut satırımıza `nano -w ~/.bashrc` komutunu veriyoruz ve .bashrc dosyasının en alt satırına `export IGNOREEOF=2` yazıp **ctrl+X** kombinasyonunu kullanarak dosyanın ismini değiştirmeden kaydederek çıkıyoruz.(Bütün kullanıcılarda bu korumayı sağlamak için değişikliği `bash.bashrc` dosyasında yapmamız gerektiğini biliyorsunuz.)

Böylece koruma sistemi kuruldu ve artık test etmeye hazır. Komut satırındayken iki kez **Ctrl + D** tuş kombinasyonunu uygularsak konsol bize iki defa uyarıda bulunacaktır ve çıkmak için ancak 3. defa bastığımızda yada komut satırına `exit` yazdığımızda konsol kapanacaktır.

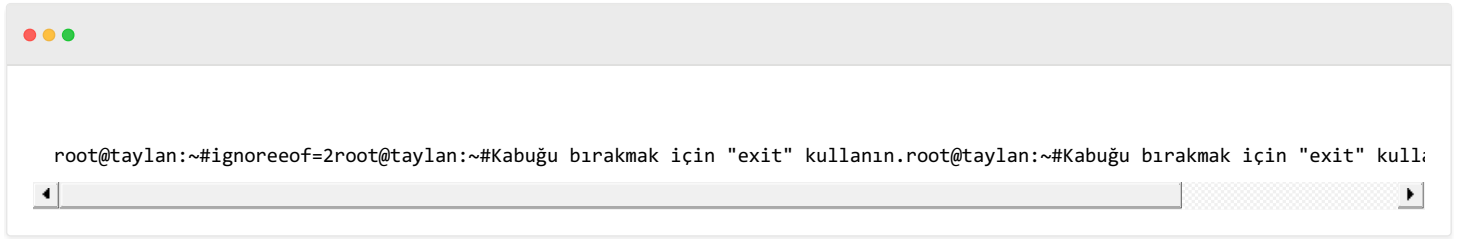



```
root@taylan:~#Kabugu bırakmak için "exit" kullanın.root@taylan:~#Kabugu bırakmak için "exit" kullanın.root@taylan:~#
```

Burada biz `IGNOREEOF=2` şeklinde belirttik ve bu tanımlama bizi 2 kez `Ctrl + D` kapatma kombinasyonundan korudu. Ancak sizler bu bana yetmez yada 2 kez fazla olur diyorsanız tanımlı istediğiniz değerlerde ayarlayabilirsiniz. Örneğin `IGNOREEOF=4` olabilir ya da `IGNOREEOF=1` olabilir, bu değer tamamen size kalmış. Ayrıca ufak bir hatırlatma, yaptığınız değişiklikler ancak Terminali kapatıp açtığınızda geçerli olur.

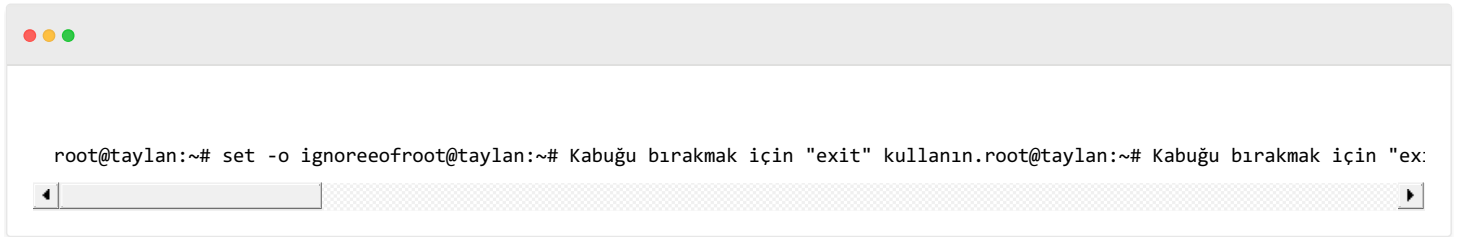
Şimdi de yalnızca mevcut Terminal için geçici koruma nasıl sağlarız ona değinelim. Koruma katmanını ne kadar hayat kurtarıcı olsa da sürekli olması bazen rahatsız edici olabilir. İşte bu noktada sadece önemli ve gerekli gördüğümüz zaman bu özelliği açıp kapatma seçeneğine sahibiz. Bunu da iki farklı yolla yapabiliriz.

İlki, komut satırına `ignoreeof=2` şeklinde komut vermek. Bu komut ile mevcut Terminal ekranı ancak 3.kez `Ctrl + D` tuş kombinasyonu uygulandığında kapanacaktır. Buradaki sayıyı dilediğiniz gibi değiştirebileceğinizi biliyorsunuz. Eğer bu özelliği devre dışı bırakmak istersek komut satırına `ignoreeof=0` yazmamız yeterli.



```
root@taylan:~#ignoreeof=2root@taylan:~#Kabugu bırakmak için "exit" kullanın.root@taylan:~#Kabugu bırakmak için "exit" kull:
```

İkinci yöntem ise komut satırına `set -o ignoreeof` yazmak. Bu komut ile de mevcut komut satırımız ancak 11.kez `ctrl+D` yapmamız sonucunda kapanacaktır. Bunun haricinde birde `exit` komutunu verirse konsol ekranı anında sonlanır. Bu özelliği devre dışı bırakmak isterseniz de komut satırına `set +o ignoreeof` yazmalısınız.



```
root@taylan:~# set -o ignoreeofroot@taylan:~# Kabugu bırakmak için "exit" kullanın.root@taylan:~# Kabugu bırakmak için "ex:
```

Yardım Almak

Bu kısımda bir nevi öğrenmeyi öğrenecez. Bunu da Linux sistemlerinin sahip olduğu çok geniş çaplı yardım sistemi kullanarak başaracağız. Linux sistemlerinde yer alan bu geniş çaplı yardım mekanizmasının bulunmasının birçok nedeni var. Ancak genel olarak, çok fazla komutun çok fazla argüman alması veya her bir programın kendine has kurulum ve kullanım komutlarının olmasından kaynaklanıyor. Linux'un kendi sahip olduğu yardım sayfalarının dışında da birçok yardım alma konuları mevcut. Zaten zamanla göreceksiniz ki Linux ile ilgili hemen her araç ve programla ilgili gerekli yardım ve dokümanlar, programın ve aracın beraberinde

geliyor olacak.

Bu bağlamda Linux sistemlerinin temel döküman-bilgi kaynaklarını 3 türe ayırabiliriz.

Bunlar; bilgi sayfaları(info), kılavuz sayfaları(manuel) ve uygulamalar ile gelen /usr/share/doc konumunda bulunan dökümanlardır. Bizler de zaman zaman unuttuğumuz için veya bilmediğimizden dolayı bu yardım sayfalarına ve dökümanlarına danışacağız. Bu girizgahtan sonra artık yavaş yavaş yardım alma komutlarımıza geçelim.

help Komutu

Hiç ingilizce bilmiyorum diyen birinin bile "help" ifadesinin "yardım" anlamında olduğunu bildiğini düşünüyorum. Yani bu sebepten **help** komutu akılda kalması en kolay komutlardandır. Komutun kullanımına geçecek olursak örneğin daha önce kullandığımız yetki verme işini gören **chmod** komutu ile ilgili yardım almak isteyelim. Bunun için komut satırına **chmod --help** şeklinde komutumuzu yazıyoruz. Ve aşağıda görüldüğü gibi gerekli bilgileri içeren yardım sayfası bizi karşılıyor.

```
root@taylan:~# chmod --helpKullanım: chmod [SEÇENEK]... KİP[,KİP]... DOSYA... veya: chmod [SEÇENEK]... SEKİZLİK-KİP DOSYA ,
```

Bu kullanımın dışında **help** komutunun birde **help komut** şeklinde kullanımı var ancak bu kullanımda her zaman komut hakkında yardım bilgisi bulunmayabiliyor o yüzden ilk öğrendiğimiz yol önceliğiniz olsun. Örneğin **help chmod** yazdığımızda komut satırı yardım bilgisi bulunmadığını belirtti.

```
root@taylan:~# help chmodbash: help: `chmod' ile ilgili bir yardım metni yok. `help help' veya `man -k chmod' ya da `info ,
```

man (Manuel Sayfası)Komutu

man(manuel) sayfaları temel yardım alma dosyalarıdır. Ve kılavuz sayfaları olarak da bilinir.

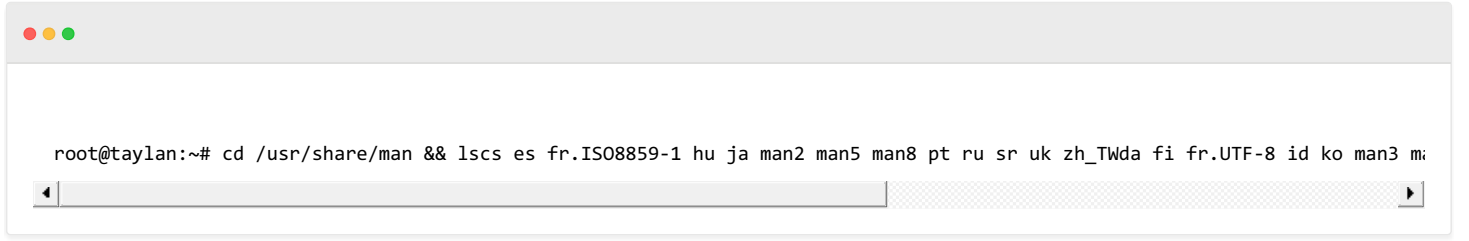
man komutunu kullanmak için komut satırına **man komut** şeklinde hakkında bilgi edinip yardım almak istediğimiz komutu giriyoruz. Örneğin ben **chmod** hakkındaki bilgilere ulaşmak istiyorsam konsola **man chmod** şeklinde yazarak gerekli bilgilere ulaşabilirim. Komutun çıktısında göreceğiniz gibi uzunca bir açıklama sizleri bekliyor. Açılan bu kılavuz sayfasında yön tuşlarını ve space tuşunu kullanarak gezinebilirsiniz. Ayrıca man sayfasının kısayollarını ve kullanımını görmek isterseniz man sayfası açıkken **h** tuşuna basarsanız sizi man klavuzunun yardım sayfası karşılar, orada kullanımı ile ilgili detaylı bilgi yardımı mevcuttur. Kılavuz sayfasını

kapatmak isterseniz ise sadece **q** tuşuna basmanız yeterli.

Şimdi biraz da man sayfasının iç yapısına değinecek olursak:

- **NAME:** Komutun ismi ve açıklama kısmı.
- **SYNOPSIS:** Komutun kullanım açıklaması(nasıl kullanılacağı).
- **DESCRIPTION:** Komutun yaptığı iş(fonksiyonu) hakkında açıklama.
- **EXAMPLES:** Komutun kullanımı ile ilgili örnekler ve açıklamalar.
- **SEE ALSO:** Diğer ilgili başlıklar.

man kılavuzunun komutlarla ilgili tuttuğu bilgi sayfaları **/usr/share/man** konumu altında tutulur. Görmek için konsola **cd /usr/share/man && ls** komutunu girdiğimizde çıktılar aşağıdaki şekildedir.



```
root@taylor:~# cd /usr/share/man && ls es fr.ISO8859-1 hu ja man2 man5 man8 pt ru sr uk zh_Tw da fi fr.UTF-8 id ko man3 m:
```

Komutu açıklayacak olursak **cd** komutu **/usr/share/man** dizinine gitmemizi sağlıyor **&&** ise daha önce gördüğümüz gibi iki komutu aynı anda çalıştırma imkanı tanıyor. **ls** komutu ise ileride de göreceğimiz gibi klasörde bulunan bütün öğeleri listeliyor. Şimdilik bu açıklama yeterli olacaktır. Biraz da bu çıktındaki sonuçlara bakalım örneğin **man** sayfasının yapılanması nasıl oluyor yani yukarıdaki komut çıktısında yer alan dil dosyaları dışındaki dosyalar yani **man1,man2,man3,man4,man5,man6,man7,man8** ne ifade ediyor kısaca ona değinelim.

- **man1:** genel kullanıcı programlarını ifade eder.
- **man2:** sistem programlarını ifade eder.
- **man3:** kütüphane fonksiyonlarını(C programlama ile ilgili) ifade eder.
- **man4:** özel dosyaları ifade eder.
- **man5:** dosya biçimlerini ifade eder.
- **man6:** ekran koruyucuları ve oyunları ifade eder.
- **man7:** diğer kategorilere girmeyen çeşitli komutları ifade eder.
- **man8:** sistem yönetimini ve bakımını ifade eder.

whatis Komutu

Hazır yeri gelmişken yukarıda gördüğümüz **man** sayfası yapılanmasıyla ilgili olarak **whatis** komutundan da söz edelim. Bu komut sayesinde hangi komutun hangi man sayfasında olduğunu öğrenebiliyoruz. Daha iyi anlamak için örnekler yapalım. Komutun kullanımı **whatis komut** şeklindedir.



```
whatis komut
```

```
root@taylan:/usr/share/man/man1# whatis chmodchmod (2) - change permissions of a filechmod (1) - change file mode bits
```

`whatis` komutuna, `chmod` komutunun man sayfasındaki açıklamalarının yukarıdaki açıkladığımız (man1,man2..3..4..5..6..7..8..) dosyalarından hangi dosyada olduğunu sorduk. Konsol yanıt olarak hem 1 hemde 2 de bulunduğunu bizlere bildirdi. Siz bunu istediğiniz komut için sorgulayabilirsiniz hatta `whatis` komutu için bile sorgulayabiliriz.

```
root@taylan:/usr/share/man/man1# whatis whatiswhatis (1) - display one-line manual page descriptions
```

Bu çıktıların doğruluğu man sayfalarının tutulduğu `/usr/share/man` dizinine gidilerek kontrol edilebilir. Örneğin ben `/usr/share/man/man1` konumuna gittiğimde `whatis` komutunun yardım sayfalarının burada olduğunu görebiliyorum. Sizler çıktıları da bu şekilde teyit edebilirsiniz.

`whatis` komutunun kullanımı bu kadar ile sınırlı değil ancak ben geri kalanını burada vermiyorum. Dileyen arkadaşlar `man` komutu yardımı ile gerekli bilgilere ulaşabilirler.

apropos Komutu

Yardım alma komutlarını noktalamdan önce; Sizlere man sayfasındayken `h` tuşuna basarsak man sayfasının kullanımı hakkında detaylı bilgiye ulaşabileceğimizi belirtmiştim. O sayfaya bakarsanız `man -k` şeklinde parametre alan komutun işlevinin, sorguladığımız komutun geçtiği uygulamaları listelemek olduğunu görebilirsiniz. Yani örnek vermek gerekirse komut satırına `man -k chmod` yazdığımızda çıktısı aşağıdaki şekilde olacaktır.

```
root@taylan:~# man -k chmodchmod (1) - change file mode bitschmod (2) - change permissions of a filechmod (2) - change pe
```

Gördüğünüz gibi belirtmiş olduğumuz `chmod` komutunun geçtiği uygulamalar listelenmiş oldu. İşte `apropos` komutu da tam olarak bu işin aynını yapıyor. Örneğin Terminale `apropos chmod` komutunu girerek çıktısı aşağıdaki şekilde olacaktır.

```
root@taylan:~# apropos chmodchmod (1) - change file mode bitschmod (2) - change permissions of a filechmod (2) - change p
```

Çıktılar karşılaştırıldığında görülüyor ki aralarında bir fark yok.

Bu konudaki noktayı da man kılavuz sayfaları güncelleme işlemi ile yapalım. Nedir bu işlem diyecek olursanız. Aradığımız yardımı man sayfasında bulamıyorsa güncelleyerek tekrar sorgulayabiliriz. Ara ara güncellemek yeni bilgilere de ulaşmamıza olanak sağlar.

Güncelleme için konsola `mandb` komutunu girmemiz yeterli olacaktır. Komut satırı, güncelleme işleminden sonra yapılan değişiklikleri de son satırda bizlere bildirir.

Yardım alma komutlarını bilmek bir zorunluluk değil ihtiyaç meselesidir. Zaten zamanla bu komutlara ve kullanımlarına alışacaksınız. Açıklamaların İngilizce olmasını da dert etmeyin, ne yaparsak yapalım eninde sonunda bu işlerin yolu İngilizceden geçiyor artık bu duruma alışmamız gerek. Bu noktada kendimizi biraz zorlamalı ve kesinlikle pes etmemeliyiz. Kendimizi biraz zorlayarak çaba harcayarak öğrenirsek, öğrenilenler kesinlikle daha da kalıcı oluyor. Sakın pes etmeyin çok iyi gidiyoruz...

Bilgi Almak

Bilgi alma komutları sistemimizde bulunan herhangi birşey hakkında(programlar,dosyalar vs..) bilgi almamızı sağlayan komutlara verilen genel isimdir. Bu komutlara gerekli durumlarda çok sık başvururuz. Şimdi bilgi alma komutlarının neler olduğunu ve kullanım şekillerine bakalım.

uname

Tek başına çekirdek adını verse de aldığı parametreler ile farklı bilgiler de sunabilir. Hangi parametrenin ne iş yaptığını `man uname` komutu ile öğrenebileceğinizi biliyorsunuz. Kısaca bilmemiz gereken; bu komut genel olarak sistemde kullanılan çekirdek hakkında bilgiler verir.

```
root@taylan:~# unameLinuxroot@taylan:~# uname -aLinux taylan 4.12.0-kali1-amd64 #1 SMP Debian 4.12.6-1kali6 (2017-08-30) x86_64 GNU/Linux
```

hostname

Bilgisayarımızın adını diğer bir adıyla host adını verir. Eğer isterseniz bu adı değiştirebilirsiniz. Örnek veriyorum komut satırına `hostname burak` yazarsanız hostname yani bilgisayarınızın adı burak olarak değişmiş olacaktır.

```
root@taylan:~# hostname burakroot@taylan:~# hostnameburak
```

Ancak bu ayar kalıcı değildir. Oturumu kapatıp tekrar açtığınızda değiştirmiş olduğunuz adın tekrar eski haline döndüğünü görebilirsiniz. Kalıcı olmasını ayarlardan kolaylıkla sağlayabiliriz. Sürümden sürüme değişiklik göstereceğiyle sırasıyla **ayarlar>sistem>ayrıntılar>genel** aygıt adı: kutusuna istediğiniz ismi yazabilirsiniz. Eğer dediğim yoldan ulaşamadıysanız ayarları biraz kurcalayarak bulabilirsiniz.

Ayrıca tamamen komut satırından da kalıcı bir **hostname** değişikliği yapmak mümkün. Bunun için **/etc/hostname** konumunda yer alan **hostname** dosyasını açmamız gerek. Dosyamızı açmak için konsola **nano -w /etc/hostname** komutunu veriyoruz.

```
root@taylan:~# hostnametaylan root@taylan:~# nano -w /etc/hostname GNU nano 2.9.1 /etc/hostname taylan
```

Komutumuzun ardından açılan dosyamızda ismimizin yerine istediğimiz ismi yazıyoruz. Ve **Ctrl + X** tuş kombinasyonu ile dosyamızdaki değişiklikleri kaydederek çıkıyoruz.

Değişikliğin geçerli olması için oturumun kapatılıp tekrar açılması gerek daha sonra hostname yani bilgisayarınızın ismi değişmiş olacaktır.

lsb_release

Kullanmakta olduğumuz dağıtımın adını verir. Bu komut da farklı bilgiler sunan parametrelere sahiptir. Detaylı bilgi için man kılavuz sayfasını kontrol edebilirsiniz.

```
root@taylan:~# lsb_release -vNo LSB modules are available.root@taylan:~# lsb_release -iDistributor ID:Kaliroot@taylan:~# 1:
```

whoami

Sinemayı takip eden arkadaşların ilk aklına ne geldi biliyorum :) Bu komut kimlik sorgu işlevi görüyor. Komut satırımıza komutumuzu vererek sonucuna bakalım.

```
root@taylan:~# whoamiroot
```

Komut çıktısı bize mevcut kullanıcının **root** kimliği ile çalıştığını göstermiş oldu.

Yine **whoami** komutuna benzer ancak çok ufak farkların olduğu birkaç komutu daha peş peşe komut satırına girerek çıkan sonuçlar üzerinden anlatımımıza devam edelim.

```
root@taylan:~# whoami rootroot@taylan:~# whoroot tty2 2017-11-30 07:34 (:1)root@taylan:~# w 08:11:49 up 39 min, 1 user, lo:
```

Sıra sıra bakalım:

- **whoami:** mevcut kullanıcının hangi kimlikle çalıştığını gösteriyor.
- **who:** sistemde hangi kullanıcının çalıştığını gösteriyor.
- **w:** hangi kullanıcı hangi uygulamayı çalıştırıyor bunun bilgisini gösteriyor.

Bunun dışında bu komutlar da parametre alabiliyorlar. Hangi parametreleri aldıklarını nasıl öğrenebileceğinizi zaten biliyorsunuz. Ben yine aşağıda birkaçının çıktısını bırakıyorum.

```
root@taylan:~# who -a sistem önyükleme 2017-11-30 07:32 açılış-seviyesi 5 2017-11-30 07:32root + tty2 2017-11-30 07:34 00:~
```

uptime

uptime komutu sistemimizin ne kadar zamandır açık olduğu bilgisini verir.

```
root@taylan:~# uptime 07:04:58 up 4:03, 1 user, load average: 1,00, 1,00, 1,00
```

date

İsminden de anlaşılacağı gibi bu komut bize sistemin o anki tarih ve saat bilgisini veriyor.

```
root@taylan:~# datePrş Kas 30 07:00:32 EST 2017
```

which

Herhangi bir komutun tam yol bilgisini öğrenmek için kullanılır.

```
root@taylan:~# which chmod/bin/chmod
```

whereis

Bu komutunda özel parametreleri vardır man sayfasından kontrol edin lütfen. Bunun dışında parametresiz hali **apropos** komutuna benzer şekilde bir çıktı verir. Ancak **apropos** komutundan farklı olarak tam dizin adresini belirtir.

```
root@taylan:~# whereis chmodchmod: /bin/chmod /usr/share/man/man2/chmod.2.gz /usr/share/man/man1/chmod.1.gz
```

Ayrıca diğer parametrelerini de kesinlikle man sayfasından inceleyin.

locate

Bu komut ile aradığımız bir dosyanın nerede olduğunu öğrenebiliriz. Komutun çıktısı bize dosyanın bulunduğu dizin veya dizinleri veriyor. Örneğin daha önce PATH yoluna eklediğim `komut.sh` dosyasını `locate` komutu ile aratıyorum. Sonuç aşağıdaki şekilde:

```
root@taylan:~# locate komut.sh/root/Desktop/yeni_dizin/yeni/en_yeni/komut.sh
```

Bu komutun ne kadar kıymetli olduğunu işiniz düştükçe ve kullandıkça anlayacaksınız. Genelde bir dosyayı arama çubuğu yardımı ile aramak sonuç vermez ve bu arama işlemi oldukça hantal çalışır. Ancak örneğin ben dosya konumunu bilmediğim bir dosyaya ulaşmak istiyorum. İşte burada kahramanız `locate` komutu çıkageliyor ve bize saniyeler içerisinde sonucu veriyor. Komutumu denemek amacıyla kali linux içerisinde yer alan bir araç olan armitage aracını aratıyorum. Bunun için komut satırına `locate armitage` yazmalıyım.

```
root@taylan:~# locate armitage/usr/bin/armitage/usr/share/armitage/usr/share/applications/kali-armitage.desktop/usr/share/:
```

Çıktıda da görüldüğü gibi aradığımız ifadeyi içeriğin tüm dosyalarının nerede olduğunu bir çırpıda buluverdik. Sanırım bu komut sayesinde yavaş yavaş da olsa komut satırının gücünü fark etmeye başlıyoruz.

dmidecode

Eğer daha önce de Linux deneyimi yaşadıysanız ve sorunlar ile karşılaştıysanız forumlarda sorduğunuzda muhtemelen insanlar sizden bu komutun çıktılarını istemiş olabilirler. Bu komutun işlevi sistemin donanım ve Bios bilgilerini göstermektir. Bu komut sayesinde çok fazla bilgiye ulaşabiliriz. Ayrıca bu komutumuz da parametre olarak çalışmaktadır.

Çok fazla bilgiye ulaşabiliyoruz dedik, bu bilgilere DMI(Desktop Management Interface) yani Masaüstü Yönetim Arayüzü tabloları ile ulaşıyoruz. Ve bu tabloların belirli bir düzeni var. Bu sırayı ve hangi bilgileri alabileceğinizi direk olarak görmek isterseniz komut satırına `dmidecode` yazın. Sonuçlar katagorize şekilde karşınıza sunulacaktır.

Bahsi geçen tablo sıralı şekilde ve Numara-Türkçe karşılığı olacak şekilde aşağıdaki gibidir.

Numara	Açıklama
0	bios
1	sistem

2 Numara	baz kurulu Açıklama
3	şasi
4	işlemci
5	bellek denetleyicisi
6	bellek modülü
7	önbellek
8	port bağlantısı
9	sistem yuvaları
10	On Board Cihazları
11	OEM Dizeleri
12	Sistem Yapılandırma Seçenekleri
13	BIOS Dili
14	Grup Dernekler
15	Sistem Event Log
16	Fiziksel Bellek Array
17	Bellek Cihazı
18	32-bit bellek hatası
19	Bellek Dizisi Haritalı Adres
20	Bellek Cihazı Haritalı Adres
21	Dahili İşaret Aygıtı
22	Taşınabilir Pil
23	Sistem Sıfırlama
24	Dananım Güvenlik
25	Sistem Güç Denetimleri
26	Gerilim Probu
27	Soğutma Cihazı
28	Sıcaklık Probu
29	Elektrik Akımı Probu
30	Uzaktan Erişim
31	Boot Bütünlüğü Hizmetleri
32	Sistem Önyükleme
33	64-bit Bellek Hatası
34	Yönetim Cihazı
35	Yönetimi Cihaz Bileseni

Numara	Açıklama
36	Yönetimi Cihaz Eşik Verileri
37	Bellek Kanalı
38	IPMI Cihazı
39	Güç Kaynağı

Başta belirttiğim gibi `dmidecode` kendi içinde parametre alan bir komut olduğu için istediğimiz spesifik bilgiye doğrudan da ulaşmamız mümkün. Detaylı bilgi için man kılavuz sayfasına bakın lütfen. Ancak küçük bir örnek vermem gerekirse sistem(system) hakkında bilgi edinmek istediğimizde `dmidecode -t system` şeklinde komut satırına girmemiz yeterli. Burada ayrıca `dmidecode -t system` yerine sıralamada bulunan numarasını yani `dmidecode -t 1` yazarak da aynı işlemi gerçekleştirebiliriz.

```
root@taylan:~# dmidecode -t system# dmidecode 3.1Getting SMBIOS data from sysfs.SMBIOS 2.7 present.Handle 0x0001, DMI type
```

fdisk-l

Başlıkta da yer aldığı gibi bu bölümde `fdisk` komutunun yalnızca `l` parametresinin işlevini görecez. Bu komutu burada vermemin sebebi sistem hakkında bilgi alırken diskin de sistem dahilinde olmasıdır. Zaten ileride tekrar ele alacağımızdan şimdilik bu kadarı da yeterli olacaktır. Bu komutu diskler üzerinde işlem gerçekleştirirken kullanıyoruz. Eğer komut satırımıza `fdisk -l` şeklinde komut verirse karşımıza sistemimizdeki disk bölümleri gelir.

```
root@taylan:~# fdisk -lDisk /dev/sda: 200 GiB, 214748364800 bytes, 419430400 sectorsUnits: sectors of 1 * 512=512 bytesSec
```

df

Bu komut ile disk kullanımı hakkında ayrıntılı bilgiye ulaşabiliriz.

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1      100G  100G  0% 100% /
/dev/sda2      100G  100G  0% 100% /
/dev/sda3      100G  100G  0% 100% /
```

```
root@taylan:~# df -h /dev/sda1 1K-blocks 1000000 0 1000000 0% /dev/sda1 203188 1372 195816
```

Bu komutumuz da parametre alabiliyor. Dileyen arkadaşlar `man df` şeklinde aratarak gerekli bilgilere ulaşabilirler.

du

Bir dizinin içerdiği tüm dosyalar ile birlikte diskte kapladığı toplam alanı verir. Ayrıca boyutla ilgili düzgün bir çıktı almak istersek `h` parametresini de kullanabiliriz.

```
root@taylan:~# du /root/Desktop12/root/Desktop/yeni_dizin/yeni/en_yeni16/root/Desktop/yeni_dizin/yeni20/root/Desktop/yeni_d
```

free

Bu komut ile kullanılan bellek miktarını KB cinsinden öğrenebiliriz. Ancak çıktımızın MB cinsinden olmasını istersek `-m` parametresini `free` komutumıza ekleyerek kullanmamız yeterli olacaktır.

```
root@taylan:~# free total used free shared buff/cache availableMem: 2031852 1024304 416200 8596 591348 828496Swap: 4882428
```

modinfo

Bu komut sayesinde Linux kernel modüllerinin bilgisi alınabilir. Bu modülleri ekran bastırmak isterseniz komut satırına `lsmod` yazarak modülleri listeleyebilirsiniz.

```
root@taylan:~# lsmodModule Size Used byvmw_vsock_vmci_transport 28672 2vsock 36864 3 vmw_vsock_vmci_transportfuse 98304 7b:
```

Örneğin modüller içinden battery modülü hakkında bilgi almak istersek komut satırına `modinfo battery` şeklinde yazmamız yeterli olacaktır.

```
root@taylan:~# modinfo batteryfilename: /lib/modules/4.12.0-kali1-amd64/kernel/drivers/acpi/battery.kolicense: GPLdescript:
```

stat

Bu komut dosya veya dizin hakkındaki bilgileri almamıza olanak sağlar. Kullanımı `stat izin` şeklindedir. Örneğin `root` dizinine bakalım. Bunun için komut satırımıza `stat /root` komutunu veriyoruz.

```
root@taylan:~# stat /root File: /root Size: 4096 Blocks: 8 IO Block: 4096 izinDevice: 801h/2049dInode: 3276801 Links: 18A
```

vmstat

Bu komut sayesinde sistemimizin o anlık genel durumunu görebiliriz. Ancak komutu verdikten sonra sistem durumu listenecek daha sonra sonlanacaktır. Eğer sistemin durumunu bir süre izlemek istersek `vmstat gecikme_hızı yenilenme_sayısı` şeklinde komut vermeliyiz. Ben her 2 saniyede bir yenilenmesini ve bunu 3 defa yapmasını istiyorum. Bunun için komut satırına `vmstat 2 3` komutunu veriyorum.

```
root@taylan:~# vmstatprocs -----memory----- ---swap-- -----io---- -system-- -----cpu----- r b swpd free buff c:
```

history

Bu kısma gelene kadar konsola bir çok komut yazdık. Peki o yazılan komutların daha sonra kullanılmak üzere saklandığını biliyor muydunuz ?

Evet girilen her komut `.bash_history` dosyasında tutuluyor. Biz bu dosyayı yani daha önceki yazdığımız kodları görmek istersek komut satırına `history` yazmalıyız.Çıktı çok uzun olacağından aşağıda vermedim.

Hazır liste uzun demişken eğer bu listenin limitini öğrenmek istersek komut satırına `echo $HISTSIZE` yazarak ne kadar komutun `history` dosyasında tutulabileceğini görebiliriz.

```
root@taylan:~# echo $HISTSIZE1000
```

Bu çıktı bize komutların tutulduğu dosyada en son 1000 komuta kadar kayıt yapıldığını belirtiyor. Eğer tutulan komutların 1000 den daha fazla olmasını isterseniz `.bashrc` dosyasında `HISTSIZE=1000` yazan değeri istediğiniz doğrultuda düzenlemeniz lazım.

Komutun kullanımına geçmeden önce bu komutu sadece basit ve kısa komutların tekrar kullanılması olarak değerlendirmeyin. Örneğin sürekli kullandığınız çok uzun ve karmaşık bir komut düşünün. Bu komutun her defasında yazılması eziyet, bir yere kopyalanıp oradan kullanılması ise hantalılık olacaktır.

Komutun kullanımına gelecek olursak örnek üzerinden açıklayalım. Benim kayıt dosyamın bir kısmı aşağıdaki şekilde, ben burada yer alan `555. komutu` yani `vmstat` komutunu kullanmak istiyorum. Bunun için komut satırına `!555` yazmam gerek. Komutun kullanımı `!komut_numarası` şeklinde. Örnek için çıktıyı inceleyebilirsiniz.

```
... 555 vmstat 556 vmstat 0 10 557 vmstat 1 10 558 clear 559 vmstat 560 clear 561 vmstat 562 vmstat 2 3 563 history ... ro
```

Daha önceden kullandığımız bir komutu aynı şekilde tekrar kullanmak için ise komut satırına `!komut` şeklinde komut girmeliyiz. Örnek vermek gerekirse daha önceki kısımlarda `stat` komutu ile root dizini hakkında bilgi edinmiştik. Ve bu bilgiye ulaşmak için `stat /root` komutunu kullanmıştık. Ben şimdi tekrar aynı komuta ulaşmak için komut satırıma `!stat` yazıyorum ve çıktısı tıpkı `stat /root` yazdığımda olduğu gibi oluyor. Yani bu sayede komutun geri kalanını uzun uzun yazmak gibi bir dert kalmıyor.

```
root@taylan:~# !statstat /root File: /root Size: 4096 Blocks: 8 IO Block: 4096 dizinDevice: 801h/2049dInode: 3276801 Links
```

Eğer son kullandığımız komutu tekrar kullanmak istersek komut satırına **!!** yazmamız yeterli olacaktır.

```
root@taylan:~# !!stat /root File: /root Size: 4096 Blocks: 8 IO Block: 4096 dizinDevice: 801h/2049dInode: 3276801 Links: 1:
```

Bunlar dışında da çok farklı kullanım şekilleri de mevcut. Bu bilgilere **man** komutu yardımı ile nasıl ulaşacağınızı biliyorsunuz.

Ayrıca son yazılan komutlara ulaşmak için klavyede yer alan yön tuşlarından **ileri** ve **geri** tuşlarını kullanarak önceki ve sonraki komutlarınıza ulaşabilirsiniz. Genellikle yön tuşlarını kullanmak bize anlık işlemlerimizde hız katmaktadır. Sizler de bu pratiklikleri mutlaka yeri geldikçe sıklıkla kullanacaksınız.

Dizinler Hakkında

Linux işletim sisteminde bütün programlar, aygıtlar, dosyalar ve genel olarak sistem, hiyerarşik bir düzen içersindedir. Yani komut satırını kullanacaksak her şeyi oluşturan bu hiyerarşik düzen içerisinde rahat şekilde gezip olabilmemiz gerek. Bu bölümde bu hiyerarşide gezinme ve görüntüleme için gerekli komutlara değineceğiz.

pwd

Bu komut sayesinde o an bulunduğumuz dizinin adını öğrenebiliriz. Genellikle sistem yöneticilerinin sık kullandığı bir komuttur.

Örnek vermek gerekirse Linux'ta hiyerarşik bir düzen var dedik. Bu hiyerarşik düzen kök dizinine(root) bağlıdır her kullanıcı buna root da dahil kendi ev dizinine(home) sahiptir. Böylece neden root(kök) isminin kullanıldığını da anlamış oluyoruz. Konsol çalışmaya varsayılan olarak kendi ev dizininde başlar. Root kullanıcısının ev dizini /root dizinidir. Bunu teyit etmek istersek **pwd** komutunu kullanabiliriz.

```
root@taylan:~# pwd/root
```

Root kullanıcısının dışında da sistemde başka kullanıcılar olabilir. Bu kullanıcıların ev dizini(home) ise **/home/kullanıcı_adi**

şeklinde dir

şeklinde olur.

Bu durumu test etmek için "kullanici" adında yeni bir kullanıcı oturumu oluşturdum ve oluşturduğum bu oturum içerisinde komut satırına `pwd` komutunu verdim. Çıktısı aşağıdaki şekilde oldu.

```
kullanici@taylan:~$ pwd/home/kullanici
```

cd (Change Directory)

Bu komut sayesinde dizinler arası geçiş yapabiliyoruz. Zaten daha önceki kısımlarda da bu komutu kullanmak durumunda kalmıştık hatırlarsanız.

Komutumuzu kullanırken gitmek istediğimiz dizinin adresini vermeliyiz. Ben önceki bölümlerde oluşturduğum dizinin en alt klasörüne gitmek istiyorum. Bunun için komut satırına `cd Desktop/yeni_dizin/yeni/en_yeni` şeklinde bir komut veriyorum.

```
root@taylan:~# cd Desktop/yeni_dizin/yeni/en_yeniroot@taylan:~/Desktop/yeni_dizin/yeni/en_yeni#
```

Artık gitmek istediğim dizinin içerisindeyim. Eğer bir önceki dizine (bir dizin geriye) dönmek istersek komut satırına `cd ..` yazmamız yeterli.

```
root@taylan:~/Desktop/yeni_dizin/yeni/en_yeni# cd ..root@taylan:~/Desktop/yeni_dizin/yeni#
```

Eğer direk olarak ana dizine dönmek istersek `cd` komutunu kullanmalıyız.

```
root@taylan:~/Desktop/yeni_dizin/yeni/en_yeni# cd root@taylan:~#
```


Örneğin bir alt dizine geçip orada yer alan başka bir klasöre girmek istiyoruz diyelim. Bunun için `cd ../klasör_adı` şeklinde bir komut kullanabiliriz.

```
root@taylan:~/Desktop/yeni_dizin/yeni/en_yeni# cd ../yeni_dosya
root@taylan:~/Desktop/yeni_dizin/yeni/yeni_dosya#
```

Eğer sürekli iki dosya arasında gidip geliyorsak bir önceki dosyaya `cd -` komutu ile dönebiliriz.

```
root@taylan:~# pwd/root
root@taylan:~# cd Desktop
root@taylan:~/Desktop# pwd/root/Desktop
root@taylan:~/Desktop# cd -/root
root@taylan:~#
```

Çıktıda da görüldüğü gibi `cd -` komutu ile iki konum arasında pratik şekilde gidip gelebiliyoruz.

ls

Listeleme ve görüntüleme işini yapan bu komutumuzu anlatmadan önce defalarca kullandık. Bu da gösteriyor ki `ls` komutu çok yaygın şekilde kullanılan komutlardan. Bu komuta sürekli işimiz düşecek. Bu komutumuzun da pek çok parametresi mevcut ben hepsini olmasa da birkaçını ele alarak anlatıma devam ediyorum.

En temel kullanımı ile başlayacak olursak, `ls` komutu içinde bulunduğumuz dizinde yer alanları bizlere gösterir. Örneğimize ana dizindeyken `ls` komutunu vererek başlayalım.

```
root@taylan:~# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
```

Görüldüğü üzere `ls` komutumuzu kullanmamız ardından bulunduğumuz ana konumda yer alan tüm dosyalar konsola basılmış oldu.

ls -l

Eğer çıktımız ayrıntılı olsun istiyorsak komut satırımıza komutumuzun parametresi ile beraber `ls -l` şeklinde yazmamız gerekiyor.

```
root@taylan:~# ls -ltoplam 32drwxr-xr-x 3 root root 4096 Kas 30 14:15 Desktopdrwxr-xr-x 2 root root 4096 Eki 7 15:26 Docum
```

Çıktımızda dosya izinleri, dosya boyutu, oluşturulma tarihi gibi bilgiler sıralanmış oldu. Tabii ki `ls` komutu kullanımı bununla sınırlı değil, parametreler ile devam edelim .

ls -a

Bizim yukarıdaki çıktımızda gizli dosyalar bulunmuyor. Eğer istersek gizli dosyaları da aynı şekilde listeleyebiliriz. Bunun için `ls` komutunun `a` parametresini kullanıyoruz. Çıktımızın düzenli ve listeli olması adına `a` parametresinin yanında bir `l` parametresi kullanmak istiyorum. Bunun için komut satırımıza `ls -la` şeklinde veya `ls -al` şeklinde komutunu girmemiz gerekiyor. Ayrıca bu komutu `ls -a -l` şeklinde ayrı arı ayrı olarak da girebiliriz. Sonuçta bu komutun parametresi hem birleşik hemde ayrı ayrı şekilde yazılabilir. Kullanım tercihi size kalmış.

```
root@taylan:~# ls -latoplam 108drwxr-xr-x 19 root root 4096 Ara 2 01:20 .drwxr-xr-x 23 root root 4096 Eki 7 15:17 ..-rw-r--
```

```
root@taylan:~# ls -a -ltoplam 108drwxr-xr-x 19 root root 4096 Ara 2 01:20 .drwxr-xr-x 23 root root 4096 Eki 7 15:17 ..-rw-r--
```

ls -A

`.` ve `..` dizinleri hariç gizli dosyalar da dahil bütün dosyaları görmek istersek komutumuz `ls -A` şeklinde kullanabiliriz. Ayrıca listelemek adına `l` parametresi de dahil edebileceğimizi biliyorsunuz.

```
root@taylan:~# ls -A.baschrc.bak .config .gconf .local .nano .rnd.bash_history Desktop .gnupg .mozilla Pictures Templates.l
```

```
root@taylan:~# ls -Altöplam 100-rw-r--r-- 1 root root 3405 Kas 28 05:52 .baschrc.bak-rw----- 1 root root 7529 Ara 2 00:52
```

ls -h

Listelenen dizinlerin boyutunun okunaklı(human readable) şekilde bize verir. Okunaklıdan kasıt dosya boyutunu KB,MB,GB türünden büyüklük olarak vermesidir. Karşılaştırmanız açısından bir normal şekilde birde daha okunaklı şekilde olanın çıktılarını aşağıda veriyorum.

```
root@taylan:~# ls -hlöplam 32Kdrwxr-xr-x 3 root root 4,0K Kas 30 14:15 Desktopdrwxr-xr-x 2 root root 4,0K Eki 7 15:26 Docu
```

```
root@taylan:~# ls -ltoplam 32drwxr-xr-x 3 root root 4096 Kas 30 14:15 Desktopdrwxr-xr-x 2 root root 4096 Eki 7 15:26 Docu
```

ls -i

Düğüm numarasını verir. Düğüm(inode) numarası nedir diyecek olursanız bu konuya ileride değineceğiz şimdiilik bu parametrenin bu görevi gördüğünü bilsek yeter.

```
root@taylan:~# ls -litöplam 323276863 drwxr-xr-x 3 root root 4096 Kas 30 14:15 Desktop3276867 drwxr-xr-x 2 root root 4096 I
```

ls -r

Konsola basılan çıktıyı ters çevirerek basar. Normal liste sırası aşağıdaki şekildedir.

```
root@taylan:~# ls -ltoplam 32drwxr-xr-x 3 root root 4096 Kas 30 14:15 Desktopdrwxr-xr-x 2 root root 4096 Eki 7 15:26 Docum
```

r parametresini kullandığımızda bu listenin terse döndüğünü yukarıdaki çıktı ile aşağıdaki çıktıyı karşılaştırarak görebilirsiniz.

```
root@taylan:~# ls -rltoplam 32drwxr-xr-x 2 root root 4096 Kas 29 13:11 Videosdrwxr-xr-x 2 root root 4096 Eki 7 15:26 Templi
```

ls -S

Dosya büyüklüğüne göre sıralar.

```
root@taylan:/run# ls -Sltoplam 24dr-xr-xr-x 3 root root 4096 Ara 2 02:33 vmblock-fuse-rw-rw-r-- 1 root utmp 1152 Ara 2 01:1
```

ls -t

Dosyanın değiştirilme tarihine göre sıralar.

```
root@taylan:/run# ls -tltoplam 24dr-xr-xr-x 3 root root 4096 Ara 2 02:41 vmblock-fuse-rw-r--r-- 1 root root 4 Ara 2 02:36
```

Yukarıda bahsi geçen `ls` komutu, açıkladığım parametreleri dışında da oldukça fazla parametreye sahip. Bunları öğrenmek için [man kılavuz sayfasını](#) inceleyin lütfen.

Dizin Oluşturma Ve Silme

Dizin oluşturmak için `mkdir` , silmek için ise ileride tekrar dosya sistemlerinde değineceğimiz `rm` komutu kullanılıyor.

mkdir

Dosya oluşturmamıza olanak sağlayan komuttur. En temel kullanımı `mkdir dosya_adi` şeklindedir. Örneğin bulunduğumuz konuma `dosyalar` adında bir dizin oluşturalım. Bunun için komut satırımıza `mkdir dosyalar` yazmamız yeterli.

```
root@taylan:~/Desktop# mkdir dosyalarroot@taylan:~/Desktop# lsdosya23.txt dosya3.txt dosya.txt komut.sh liste yeni_dizindo:
```

Eğer çoklu şekilde dosya oluşturmak istersek dosya isimlerini peş peşe yazarak bu işlemi gerçekleştirebiliriz. Komutumuzun kullanımı `mkdir dizin1 dizin2 dizin3` şeklindedir.

```
root@taylan:~/Desktop/dosyalar# mkdir dizin1 dizin2 dizin3root@taylan:~/Desktop/dosyalar# lsdizin1 dizin2 dizin3
```

Ayrıca dosyayı oluştururken izin ayarlarını da dilediğimiz şekilde ayarlayabiliriz. Bu imkanı bize `-m` parametresi sağlıyor. Kullanımı `mkdir -m izin_kodu dosya_adi` şeklinde.

```
root@taylan:~/Desktop/dosyalar# mkdir -m 700 secureroot@taylan:~/Desktop/dosyalar# ls -ld securedrwx----- 2 root root 4096
```



rm

Komutumuzu ileride dosya işlemleri bölümünde tekrardan ele alacağız. Ancak şimdi `rm` yani silme komutumuzun dizinler üzerinde kullanımı öğrenmeliyiz.

Dizin dediğimiz şey iç içe sırlanmış dosya ve belgelerden oluşan bir yol. Yani bu yolu silmek istediğimizde içerisindeki her şeyi de silmeliyiz. Bu yüzden komutumuzu `-r` parametresi ile birlikte kullanarak dizinin içerisindeki dosyalar da dahil her şeyi silmesini söylemiş oluruz. Yani kullanım `rm -r silinecek_dizin` şeklindedir.

```
root@taylan:~/Desktop/dosyalar# rm -r dizin1
```

Ve dizinimiz içerisinde bulunanlar ile birlikte silinmiş oldu. Eğer komutumuzu parametresiz olarak yani `-r` olmadan `rm dizin1` şeklinde verseydik komut satırı bu işlemi dizinin içerisinde dosya ve belgelerin bulunması nedeniyle gerçekleştiremezdi.

```
root@taylan:~/Desktop/dosyalar# rm dizin1rm: 'dizin1' silinemedi: Bir dizin
```

Eğer silmeden önce sorulsun yani bizi uyarсын istiyorsak komutumıza `-i` parametresini de eklemeliyiz.

```
root@taylan:~/Desktop/dosyalar# rm -ri aarm: 'aa' dizininin içine inilsin mi?erm: 'aa/bb' dizininin içine inilsin mi?erm:
```

Gördüğünüz gibi konsol işlem yapmadan önce her işi bize sordu ve "e" yanıtını aldığı sürece gerekli işlemlere devam etti. Bu sayede dizinleri kontrollü şekilde silmiş olduk.

Ayrıca dizin silme işlemlerini `rm -ri dizin1 dizin2 dizin3` şeklinde kullanarak toplu şekilde de gerçekleştirebilirsiniz.



Dosya İşlemleri

Konumuz itibari ile dosyalarla epeyce haşır neşir olacağız. Yapacağımız işlemleri kısaca tanımlamak gerekirse; dosya kopyalama, taşıma, değiştirme, görüntüleme, düzenleme.. vs.. gibi işlemlerle dosyalar üzerinde hakimiyet kuracağız.

touch

`touch` komutu bize kolaylıkla metin dosyası oluşturma imkanı tanıyor. İşte bu yüzden dosya işlemleri konumuza `touch` komutu ile başlamak gayet doğru bir karar olacaktır.

Kendimize `touch` komutunu kullanarak boş bir dosya oluşturmak için komut satırına `touch dosya_ismi` şeklinde yazmamız gerekiyor. Burada fark ettiyseniz herhangi bir uzantı girmeme gerek kalmadı. Çünkü Linux sisteminde uzantı belirtme zorunluluğu bulunmuyor.

```
root@taylan:~# touch yeniroot@taylan:~# lsDesktop Documents Downloads Music Pictures Public Templates Videos yeniroot@tayl:
```

Görüldüğü üzere dosyamız bulunduğumuz ana dizinde oluşmuş oldu. Bunu teyit etmek için de çıktıda görüldüğü gibi daha önce öğrenmiş olduğumuz `ls` komutunu kullandık.

Peki bir tane değil de birden fazla dosya oluşturmak istersek. İşte bunun için de komutumuzu `touch yeni1 yeni2 yeni3` şeklinde yazmamız gerekiyor.

```
root@taylan:~# touch yeni1 yeni2 yeni3root@taylan:~# lsDesktop Downloads Pictures Templates yeni yeni2Documents Music Publ:
```

Çıktıda görüldüğü gibi aynı andan 3 dosya oluşturmayı başardık.

Aslında `touch` komutunun çok farklı kullanım yöntemleri de mevcut ancak ben son olarak ileri bir tarihte otomatik olarak dosya oluşturmayı göstererek bu komutun konu anlatımını burda sonlandıracağım.

İleri bir tarihte tıpkı alarm kurarcasına planlanmış bir şekilde dosya oluşturmak mümkün. Bunun için "`t`" parametresini kullanacağız. Kullanımı ise `touch -t YYYYAGGSSDD.SS dosya_adı` şeklindedir. Ben burada örnek olması açısından saat 23.18.30 ve tarih olarak 2017.12.3 zamanlamasını kullanıyorum. Nasıl kullanıldığını daha net anlamak için örneği inceleyebilirsiniz.

```
root@taylan:~# touch -t 201712032318.30 yenedosyaroot@taylan:~# lsDesktop Downloads Pictures Templates yeni yeni2 yenedosy:
```

Neticede benim belirtmiş olduğum ileri bir tarihte dosya kendiliğinden oluşmuş oldu.

cat

`cat` komutu temelde dosya içeriğini terminal ekranından okumamıza olanak sağlayan bir aracı çağıran komuttur. Ancak bu komut bizlere temel işlevinin dışında da dosyalar ile ilgili pek çok işlem yapma imkanı sağlıyor. Sırasıyla bunların bazılarına göz atacağız. Ama ilk olarak temel işlevi olan terminal üzerinden dosya okuma işlevini görelim.

Bunun için konsola açmak istediğimiz dosyanın konumundayken `cat açılacak_dosya` şeklinde komut vermemiz gerek.

```
root@taylan:~/Desktop# cat dosya.txtBu dosyayı okuyorsanız dosya içeriğini açmışsınız demektir. Artık cat komutunun kullan:
```

Şimdi ise tıpkı `touch` komutunda çoklu dosyalar ile ilgili işlem yaptığımız gibi `cat` komutu ile de çoklu dosya okuma işlevini gerçekleştirelim. Çoklu dosya okuma işlemi için komut satırımıza `cat dosya1 dosya2 dosya3` şeklinde okumak istediğimiz dosyaları yazıyoruz.

```
root@taylan:~/Desktop# cat dosya.txt dosya2.txt dosya3.txtBu dosyayı okuyorsanız dosya içeriğini açmışsınız demektir.Artık
```

Dosya içeriğine yazı eklemek istersek `cat > dosya_adı` şeklinde komut verdiğimizde, komut satırı imleci bir alt satıra geçerek bizden yazmak istediğimiz bilgiyi bekler. Yazma işlemimiz bittikten sonra kaydederek kapatmak için `Ctrl + D` tuş kombinasyonunu kullanırız. Ayrıca komutta girdiğimiz isimde bir dosya yoksa o isimde bir dosya oluşur ve girilen veriler içerisine işlenir.

```
root@taylan:~/Desktop# cat dosya2.txtYeni eklediğimiz yazı dizisi bu şekildedir.
```

Ancak fark ettiyseniz eğer dosya içerisinde mevcut yazı bulunuyorsa bu işlemin ardından eski veriler silinerek yalnızca girilen yeni veri dosyaya işlenip kayıt oluyor. Bizler eğer önceki veriler silinmesin ve üzerine yeni veri ekleyebilelim istersek komutumuzu `cat >>`

`dosya2.txt` şeklinde kullanarak ve yeni gireceğimiz verileri girdikten sonra **Ctrl + D** tuş kombinasyonu ile dosyamızı yeni eklenmiş verileriyle birlikte kaydedip kapatabiliriz.

```
root@taylan:~/Desktop# cat dosya2.txtYeni eklediğimiz yazı dizisi bu şekildedir.Bu en son girdiğimiz veri olacak.
```

Gördüğünüz gibi yeni eklediğimiz veri dosyamızın içine işlenmiş oldu üstelik eski veriler de yok olmadan.

Ayrıca yeri gelmişken kullanmış olduğumuz `>` yönlendirme operatörü ile ilgili bir örnek daha yapalım ki kullanım alanının biraz daha farkına varmış olalım. Örneğin `ls -la` çıktısının içeriğini "liste" adında bir dosya oluşturarak içerisine aktaralım. Bunun için `ls -la > liste` komutunu kullanabiliriz. Şayet yeni dosya oluşturmak istemiyorsak ve var olan dosyaya yazılsın istiyorsak o zaman `>` operatörü yerine `>>` operatörünü kullanırsak yeni veriler dosyaya eklenmiş olur.

```
root@taylan:~/Desktop# ls -la > listeroot@taylan:~/Desktop# cat listetoplam 32drwxr-xr-x 3 root root 4096 Ara 10 02:31 .dr
```

Komutta açıklanacak bir durum yok zaten hepsi bildiğimiz komutlar. Zaten burada asıl önemli olan `>` operatörünün kullanımıydı. Eğer anladıysak ne âlâ.. devam edelim.

`cat` komutunun işlevlerine devam ediyoruz şimdi ise bir dosyada yer alan verileri başka bir dosyaya kopyalamak var. Bunun için komutumuzu `cat veri_alınan_dosya > veri_alan_dosya` şeklinde kullanıyoruz.

```
root@taylan:~/Desktop# cat dosya2.txtYeni eklediğimiz yazı dizisi bu şekildedir.Bu en son girdiğimiz veri olacak.Bu dosya:
```

Ve son olarak şaşıracığınızı umduğum bir kullanımı da göstererek `cat` komutunu anlatımını burda noktalayacağım. Örneğin ayrı ayrı yer alan film partları var eğer istersek onları tek part haline dönüştürebiliriz. Komutun kullanımı `cat part1.mkv part2.mkv part3.mkv > tekpartfilm.mkv` şeklindedir.

Belki defalarca söyledim ancak bu komut için de söylemem gerek ki daha fazla detay için lütfen man kılavuz sayfasına göz atın.

İsminden de anlamış olacaksınız ki `tac` komutu önce gördüğümüz `cat` komutunun tersi şekilde çıktı veriyor.

```
root@taylan:~/Desktop# cat dosya2.txtYeni eklediğimiz yazı dizisi bu şekildedir.Bu en son girdiğimiz veri olacak.Bu dosyay:
```

Zaten her iki çıktıyı karşılaştırırsanız aralarındaki farkı rahatlıkla görebilirsiniz.

rev

Hazır terslikten bahsetmişken `rev` komutumuzdan da söz etmemek olmaz. Bu komut dizeleri sondan başa doğru çevirerek ters şekilde ekrana basıyor. Bu durum en iyi örnek ile açıklanır. Hatta `cat` komutu ile de karşılaştırsak sonuç daha net oraya çıkacaktır.

```
root@taylan:~/Desktop# cat merhabamerhabalar..Yeni ifaderoot@taylan:~/Desktop# rev merhaba..ralabahremedafi ineY
```

Çıktılarda görüldüğü gibi dosya içeriğinde yer alan ifadeler `rev` komutu sayesinde ters şekilde ekrana basılmış oldu.

Şimdi ben bu komutu nerede kullanıcam neden öğrendim demeyin. Bu konuya daha önce de değinmiştim, şimdi tekrar söylüyorum; burada öğrendiğimiz her komutu doğrudan her zaman kullanmayabilirsiniz ancak yeri geldiğinde böyle bir komutun varlığından haberdar olarak gerektiğinde dokümantasyona bakmak suretiyle komutun kullanımına en kısa sürede ulaşabilirsiniz. Yani bu komutun varlığından haberdarsınız eğer gerekirse burada olacak..

echo

Bu komutumuzu daha önce defaatle kullanmak durumunda kalmıştık hatırlarsanız. Kullandıkça da işlevinden bahsetmiştik ancak şimdi komutumuzu ele alarak biraz daha yakından bakmaya başlıyoruz.

Temel işlevi terminal ekranına istenilen bilgileri çıktı olarak göndermektir. Komutumuzu kullanırsak daha net anlaşılacaktır. Örneğin komut satırına "Merhabalar" yazdırmak isteyelim bunun için komut satırımıza `echo Merhabalar` şeklinde komutumuzu girmemiz yeterli.

```
root@taylan:~# echo MerhabalarMerhabalar
```

"Merhabalar" çıktımızı almış olduk. `echo` komutumuz sadece ekrana basma işlevine sahip değil. Örneğin bir dosya oluşturup içerisine istediğimiz ifadeyi yazabiliriz. Bunun için komut satırımıza `echo "yazılacak_ifade" > dosya_adı` şeklinde komutumuzu giriyoruz.

```
root@taylan:~# cd Desktoproot@taylan:~/Desktop# echo "merhabalar.." > merhabaroot@taylan:~/Desktop# lsdosya23.txt dosya2.t:
```

Sırasıyla yaptığım işlemleri açıklayayım.

İlk önce `cd Desktop` komutu ile masaüstü konumuna geldim.

Daha sonra `echo "merhabalar.." > merhaba` komutunu vererek masaüstü konumuna `merhaba` isimli bir dosya oluşturarak içerisine "merhabalar.." ifadesini yazmış oldum.

`ls` komutu ile de bulunduğum konumdaki dosyaları listeleyerek içerisinde oluşturduğum "merhaba" adlı dosyanın bulunup bulunmadığını teyit ettim.

`cat merhaba` komutu ile de `cat` komutunun en temel kullanım işlevi olan içeriği ekran basma işlevini kullanarak oluşturduğum dosyanın içeriğine baktım.

Sonuç itibari ile her şey sorunsuz ilerledi ve finalde yeni oluşturmuş olduğum merhaba isimli dosyanın içerisine "merhabalar.." ifadesini yazmış oldum.

Ancak şöyle bir hatırlatmada bulunayım; eğer var olan bir dosyanın içeriğine yeni içerikler eklemek isterseniz daha önce de kullandığımız şekilde `>>` parametresini kullanın. Aksi halde dosya içeriğindeki her şey silinir ve yalnızca sizin son yazdığınız ifade kalır..

Var olan dosya içeriğine yeni bir ifade eklemek için komutu `echo "Yeni ifade" >> dosya_adı` şeklinde girmemiz gerek.

```
root@taylan:~/Desktop# echo "Yeni ifade" >> merhabaroot@taylan:~/Desktop# cat merhabamerhabalar..Yeni ifade
```

Şimdi `echo` komutunun `ls` komutu görevi gördüğü bir kullanıma değineceğiz.

Örneğin bulunduğumuz dizinde yer alan dosyaları görmek istersek komut satırına `echo *` şeklinde yazmamız halinde çıktıda bizlere bulunduğumuz konumda yer alan dosyaları verir. Bu komut kullanımını `ls` komutu ile karşılaştırdığımızda daha net anlaşılacaktır.

```
root@taylan:~/Desktop# echo *dosya23.txt dosya2.txt dosya3.txt dosya.txt komut.sh liste merhaba yeni_dizinroot@taylan:~/De:
```

* joker karakteridir ve herhangi dosya dizin isimlerinin yerini tutar. Yani `echo` komutuna * karakteri eklediğimizde mevcut dizinde yer alan tüm dosya dizinler * parametresi ile kapsanacağı için ne var ne yok listeleniyor. Bu durumu örnek olması açısından bulunduğumuz dizinde yer alan dosyalarda örneğin dosya ile başlayanları çıktı olarak almak istersek komut satırımıza `echo dosya*` şeklinde komut vermemiz gerek.(Joker karakter/wildcard konusunda bu karakterler açıklanmıştır.)

```
root@taylan:~/Desktop# echo dosya*dosya23.txt dosya2.txt dosya3.txt dosya.txt
```

Son olarak `echo` komutunun kullanım şekillerinden olan, bir komutun çıktılarının `echo` komutu ile ekrana basılması var. Ancak doğrudan `echo komut` şeklinde yazılan komutlar istenmeyen bir sonuç çıkaracaktır. Verdiğimiz komutun çıktılarını alma işlemini gerçekleştirebilmek için `echo` komutunun bu iş için kullanım özelliklerinden olan iki farklı seçenek bulunuyor. Bunlardan biri ters tırnak ' ' işaretini diğeri ise `$(komut)` parametre bütünüdür. Bunları sırayla görelim.

İlk olarak ters tırnak işaretli olan kullanımı ele alalım.

Ters tırnak işaretini oluşturmak için Türkçe klavyede `AltGr + ,` tuş kombinasyonunu kullanabiliriz. Komutun kullanımını için ise istediğimiz komutu tırnak içerisinde `echo` komutunun yanına `echo `ls`` şeklinde yazıyoruz.

Yani kullanımı `echo 'komut'` şeklinde oluyor. Hemen bu kullanım ile ilgili bir örnek yapalım ve `ls` komutunun çıktılarını `echo` komutu yardımı ile basalım.

```
root@taylan:~/Desktop# echo `ls`root@taylan:~/Desktop# echo `ls`dosya23.txt dosya2.txt dosya3.txt dosya.txt komut.sh liste
```

Ve ikinci yol olan `$(komut)` parametre bütünü ise yine `ls` komutu için örneklendirilecek olursa, `echo $(ls)` şeklinde komut parantez içine gelecek şekilde kullanılmalıdır.

Sonuç itibarı ile ilk `echo ls` komutunun çıktısı olan `ls` bizim almak istediğimiz çıktıyı bizlere vermedi. Yani bu kullanım bizim yapmak istediğimiz iş olan komutun çıktılarının ekrana bastırılması için doğru bir kullanım değildi. Doğru kullanım şekli ise ikinci ve üçüncü komut olan `echo `ls``, `echo $(ls)` şeklindedir. Bu komutların çıktısı istediğimiz şekilde, yani `ls` komutunun çıktısını(bulduğumuz dizinde yer alan dosyaların bilgisi) ekrana basacak şekilde bizlere verdi.

 [more](#)

Şimdiye kadar öyle yada böyle terminal üzerinden dosya okuma işlemi yapmamıza yardımcı olan komutlar gördük. Bu komutumuzda

aynı şekilde metin dosyalarını terminal üzerinden okumamıza olanak sağlayan bir komuttur.

Örneğin daha önce de içeriğini okuyup değişiklik yapmış olduğumuz dosya olan `profile` dosyasının içeriğini `more` komutumuz yardımı ile okuyalım.

Bunun için dosyanın konumuna gelip `more profile` şeklinde komutumu girerek `profile` dosya içeriğini okumaya teşebbüs ediyorum.

```
root@taylan:/etc# more profile
```

Komutumuzun ardından terminal ekranında karşımıza `profile` dosyasının içeriği aşağıdaki gibi geldi.

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))# and Bourne compatible shells (bash(1), ksh(1), ash(1))
```

Çıktıda görüldüğü gibi dosyanın içeriği ancak terminal ekranına sığacak şekilde görüntüleniyor. Hatta alt kısımda `--More-- (75%)` şeklinde yazan yazı bizlere içeriğin devamı olduğunu belirtiyor. Eğer devamını okumak istersek `enter` tuşuna basarak adım adım alt satırlara bakabiliriz. Ayrıca `space` tuşu ile de sayfa sayfa atlayarak içeriğe göz atabiliriz. Eğer atladığınız sayfadan bir önceki sayfaya geri dönmek istersek `b` tuşunu, şayet dosya içeriğini görüntüleyen bu ekranı kapatmak istersek ise `q` tuşunu kullanmamız yeterli.

less

Bu komutumuz da üst kısımda açıkladığımız `more` komutu ile aynı işlevdedir. Ayrıntısını merak ederseniz man sayfasına göz atmanız yeterli olacaktır.

Ben yinede `less` komutumuz ile `profile` dosyasının içeriğinin okunduğu şekli aşağıya bırakıyorum. Eğer `more` komutunda bir problem yaşamadıysanız bu komut kullanımında da kesinlikle yaşamazsınız.

```
root@taylan:/etc# less profile
```

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))# and Bourne compatible shells (bash(1), ksh(1), ash(1))
```

head-tail

Aslında komutların isimleri yani head(kafa/baş) ve tail(kuyruk) bizlere komut hakkında az çok bilgi veriyor. Şöyleki bir dosyanın sadece baş kısmında ilk 10 satırı görüntülemek istersek `head` şayet son 10 satırı görüntülemek istersek ise `tail` komutunu kullanırız. Ayrıca bu komutlar özel parametre olarak istenildiği sayıda baştan veya sondan olmak üzere istenilen satırların görüntülenmesine olanak sağlar. Daha iyi anlamak adına hemen kullanımlarına geçelim.

Dosya içeriğinin ilk 10 satırını görüntülemek için `head dosya_adı` şeklinde komutumuzu kullanıyoruz.

```
root@taylan:/etc# head profile# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))# and Bourne compatible shells (bash(1), ksh(1), ash(1))
```

Şimdi burada 10 satır içerik yok gibi gelebilir ancak dosya içerisindeki boş satırlar da bu 10 satıra dahil olduğundan çıktıda boş satırlar ile saydığınızda tam olarak 10 satırın görüntülendiğini göreceksiniz.

Dosya içeriğinin son 10 satırını görüntülemek için `tail dosya_adı` şeklinde komutumuzu kullanıyoruz.

```
root@taylan:/etc# tail profileif [ -d /etc/profile.d ]; then for i in /etc/profile.d/*.sh; do if [ -r $i ]; then . $i fi done
```

Şimdi istediğimiz sayıda baştan veya sondan dizinleri görüntüleyelim. Ben örnek olması açısından baştan 6 sondan 2 satırı görüntülemek üzere komutumu veriyorum. Komutun kullanımı `head -n 6` ve `tail -n 2` şeklinde.

Baştan 6 satırın çıktıları.

```
root@taylan:/etc# head -n 6 profile# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))# and Bourne compatible shells (bash(1), ksh(1), ash(1))
```

Sondan 2 satırın çıktıları.

```
root@taylan:/etc# tail -n 2 profilePATH="/root/Desktop/yeni_dizin/yeni/en_yeni:$PATH"export LANG=TR
```

nl

Bu komutun işlevi satırları numaralandırarak çıktı vermektir. Hemen görelim.

```
root@taylan:~/Desktop# lsdosya23.txt dosya3.txt komut.sh merhabadosya2.txt dosya.txt liste yeni_dizinroot@taylan:~/Desktop#
```

Çıktıda da görüldüğü üzere dosya içeriğinde yer alan ifadeler satır satır numaralandırılarak ekrana basılmış oldu.

Ayrıca eğer listelenen içerik uzunsa bir önceki öğrendiğimiz `head-tail` komutları yardımı ile bu alanı düzenli şekilde sınırlandırabiliriz.

```
root@taylan:~/Desktop# nl dosya2.txt | head -n 3 1Yeni eklediğimiz yazı dizisi bu şekildedir. 2Bu en son girdiğimiz veri o:
```

sort

Dosya çıktıların alfabetik olarak düzenlenmesini sağlar. Örnek üzerinden devam edelim.

```
root@taylan:~/Desktop# sort dosya2.txt Bu dosyayı okuyorsanız dosya içeriğini açmışsınız demektir. Artık cat komutunun kul:
```

Eğer ters alfabetik şekilde yani z den a ya doğru sıralamak istersek `-r` parametresi kullanılır.

```
root@taylan:~/Desktop# sort -r dosya2.txtYeni eklediğimiz yazı dizisi bu şekildedir.Netice itibari ile okumakta olduğunu dı
```



paste

Örneğin iki dosyanın içeriğini aynı anda yan yana olacak şekilde komut satırında listelemek istersek bu noktada bu işi yapmamızı `paste` komutu sağlıyor. Kullanımı `paste dosya1 dosya2` şeklindedir.

Ben `yaşlar.txt` ve `isimler.txt` adlı dosyaları aynı anda yan yana görüntülemek adına komut satırına `paste isimler.txt yaşlar.txt` komutumu verdim.

```
root@taylan:~/Desktop# paste isimler.txt yaşlar.txtahmet45mehmet22ali3hasan65hüseyin78burak99
```



tee

Yazmak istediğimiz ifadeleri hem komut satırına yansıtıp hemde bu ifadelerim bir dosya içerisine kayıt olmasını isterseniz kullanacağınız komut `tee` k omutu olacaktır.

Örnek üzerinden devam edelim.

`tee metin` komutu ile metin isminde bir dosya açıyorum ve içerisine birtakım ifadeler ekliyorum daha sonra ekleme işlemim bittiğinde dosyayı kaydederek kapatmak üzere `Ctrl + D` tuş kombinasyonunu uyguluyorum. En son oluşturmuş olduğum dosya içeriğine bakmak üzere `cat` komutunu kullanıyorum.


```
root@taylan:~/Desktop# tee metinLinux Metni Linux Metni Deneme yazısı ilk satırDeneme yazısı ilk satırDeneme yazısı ikinci
```

Çıktıda da görüldüğü üzere terminal ekranında yazdığımız her bir satır tekrar ederek konsola basıldı ve sonuçta oluşturmuş olduğumuz "metin" isimli dosyanın içeriği de yazmış olduğumuz ifadeler ile dolmuş oldu.

cmp

Bu komut sayesinde dosyalar arasından karşılaştırma yapmamız mümkün.

Örneğin ben `isimler.txt` dosyasını kopyladım ve 5. satırındaki ismi değiştirdim. Ve bu iki dosyayı `cmp` komutu ile kıyasladım.

```
root@taylan:~/Desktop# cat isimler.txtahmetmehmetalihasanhüseyinburakroot@taylan:~/Desktop# cat isimler_kopya.txtahmetmehm
```

İki dosyanın da içeriğini `cat` komutu ile yazdırdık görüldüğü üzere iki dosya arasındaki fark, bir dosyada hüseyin ismi yerine can isminin yer almasıdır. `cmp` komutumuzun çıktısında da bu durum farklı olan satır ve bayt bilgisi verilerek ifade ediliyor.

grep

Bu komut en çok kullanılan terminal komutları içerisinde bulunuyor. Bu komutumuzu bizler de oldukça sık kullanıyor olacağız. Dosya ya da komut çıktısında bir ifade aramak için yani bir nevi filtreleme işlemi için `grep` komutunu kullanabiliriz. Komutumuzun kullanım şekillerini örnekler ile açıklamaya devame edelim.

Eğer bir dosya içerisindeki herhangi bir kelimeyi sorgulamak istersek komutumuzu `grep "aranacak_ifade" dosya_adı` şeklinde komut satırına giriyoruz.

```
root@taylan:~/Desktop# grep "Bu" dosya2.txtBu en son girdiğimiz veri olacak.Bu dosyayı okuyorsanız dosya içeriğini açmışsı
```

Arama işleminde linux sistemi gereği büyük küçük harf duyarlılığına sahip olduğundan "Bu" şeklinde arama yaptığımızda diğer

buyuk küçük haldeki anahtar kelimeler çıktıda basılmayacaktır. Eğer bu küçük büyük harf duyarlılığı dışında arama yapmak istersek `-i` parametresini kullanabiliriz.

```
root@taylan:~/Desktop# grep -i "Bu" dosya2.txtYeni eklediğimiz yazı dizisi buşekildedir.Bu en son girdiğimiz veri olacak.Bu
```

Eğer arama işlemini tek dosya yerine bulunduğumuz dizin içerisindeki tüm dosyaları kapsayacak şekilde gerçekleştirmek istersek wildcard yani genel kapsama işlevinde olan joker bir `*` karakteri yardımıyla bu işlemi gerçekleştirebiliriz.

Hemen örneklere geçelim.Ben burak ismini masaüstü konumundayken aratıyorum. Bunun için `grep "burak" *` komutunu konsola verdim.

```
root@taylan:~/Desktop# grep "burak" *isimler_kopya.txt:burakisimler.txt:burakgrep: yeni_dizin: Bir dizin
```

Çıktıda aradığımız ifadenin hangi dosyada yer aldığını bulmuş olduk. Ayrıca yukarıdaki komutu `grep -r "burak" *` şeklinde de yazabilirdik. Genellikle de bu 2. komut kullanılıyor ancak hangi şekilde kullanacağınızı seçmek size kalıyor.

```
root@taylan:~/Desktop# grep -r "burak" *isimler_kopya.txt:burakisimler.txt:burak
```

İkinci kullanım şeklinde ilkinden farklı olarak "grep: yeni_dizin: Bir dizin" gibi bir çıktı almadık.

Wildcardlardan söz etmişken dosya içerisinde arama işleminin başka bir kullanım şekline de değinelim isterim. Eğer aranacak kelime belirli bir isim ile başlayan dosyalar içerisinde aranacaksa bunu wildcard(joker) yardımıyla özellikle belirtebiliriz.

Kullanımı `grep "aranacak_ifade" dosya_adi_başlangıcı*` şeklindedir. Ben ilk önce masaüstünde yer alan dosyaları listeledim daha sonra "burak" ifadesini aramaya koyuldum. İlk önce "dosya" ismi ile başlayan belgelerde `grep "burak" dosya*` komutunu kullanarak aradım. Ancak çıktıda da görüleceği gibi "burak" ifadesi "dosya" ile başlayan belgeler içerisinde yer almıyormuş ki komut çıktısı bize hiç bir sonuç vermedi. Bende daha sonra yine "burak" ismini bu sefer "isimler" ile başlayan belgeler içerisinde aramak için `grep "burak" isimler*` komutunu kullandım. Bunun neticesinde komut satırı bize "burak" isminin geçtiği belgelerin yerini vermiş oldu.

```
root@taylan:~/Desktop# ls dosya23.txt dosya3.txt isimler_kopya.txt komut.sh merhabadosya2.txt dosya.txt isimler.txt liste !
```

Buradaki asıl kullanım amacı komutu belirli isimlerde olan dosyalar içerisinde anahtar kelimeyi aramak üzerine sınırlandırmak.

Bunların dışında eğer arama sonucunun görüntülenmesini sınırlandırmak istersek;

Arama sonucu sonrasında kaç satır gözükeceğini -A parametresi sayesinde `grep -A 3 "aranacak_ifade" dosya_adı` şeklinde komut girerek sağlarız.

Sonucun üzerindeki kaç satırın gözükeceğini -B parametresi sayesinde `grep -B 5 "aranacak_ifade" dosya_adı` şeklinde komut girerek sağlarız.

Baştan ve sondan kaç satır görüntüleneceğini ise -C parametresi sayesinde `grep -c 2 "aranacak_ifade" dosya_adı` şeklinde komut girerek sağlarız.

Şimdi ise sıklıkla kullanılan pipe ile `grep` kullanımına bakalım. Pipe nedir diyecek olursanız kabaca iki işlemi(process) birbirine bağlayan `|` görünümündeki yardımcı argüman diyebiliriz. Zaten daha önce de iki işlemin birbirine bağlanması gereken durumlarla karşılaşmış ve pipe kullanmıştık.

`grep` ile kullanımına geçecek olursak örneğin masaüstünde yer alan dosyalardan yalnızca "dosya" isimli olanları listelemek isteyelim. Bunun için ilk olarak masaüstünde yer alan dosyalarımızı listelemek üzere komut satırımıza `ls` daha sonra ise `ls` komutundan gelecek olan çıktıları pipe yardımı ile `grep` komutuna bağlayarak listelenen bütün bu dosyalar içerisinde sadece "dosya" adı geçenleri filtreleyebilelim..

Bu işlemi gerçekleştirmek için `ls | grep "dosya"` komut bütünü kullanıyorum.

```
root@taylan:~/Desktop# ls | grep "dosya"dosya23.txtdosya2.txtdosya3.txtdosya.txt
```

`grep` komutunun kullanım şekli ve alanı elbette bu kadar ile sınırlı değil ancak sizler daha fazla bilgiye nasıl ulaşacağınızı biliyorsunuz.

find

`find` komutumuz da hemen hemen daha önce görmüş olduğumuz `locate` komutu ile aynı görevi görüyor. Örneğin, isminin birazını bildiğimiz ama dosya dizinini hatırlamadığımız dosyaları bulmamıza yardımcı oluyor.

Komut kullanım kalıbı `find dosya_yolu -name arananak_ifade` şeklindedir. Tam kullanım şekli için aşağıdaki örneği inceleyebilirsiniz.

```
root@taylan:~/Desktop# find /etc -name *local/etc/apparmor.d/local/etc/X11/Xsession.d/35x11-common_xhost-local/etc/dconf/dl
```

Çıktıda **etc** konumunda yer alan içerisinde "local" ifadesi geçen dosyalar komutumuz sayesinde listelenmiş oldu. Ancak aradığımız şeyi bulmak adına değiştirilme tarihi, boyut, dosya-dizin gibi ayrıntıları belirterek çıktıları daha spesifik şekilde sınırlandırabiliriz. Bazı özel arama parametreleri açıklayalım hemen.

- **type f** :Yalnızca dosyalar.
- **type d** :Yalnızca dizinler.
- **size +500k** :500 Kbyt'dan büyük olan dosyalar.
- **size -500k** :500 Kbyt'dan küçük olan dosyalar.
- **ctime 10** :Tam olarak 10 gün önce değişikliğe uğramış dosya/dizinler.
- **ctime -10** :10 günden daha kısa bir süre önce değişikliğe uğramış dosya/dizinler.
- **ctime +10** :10 günden daha uzun bir süre değişikliğe uğramış dosya/dizinler.
- **perm 755** :Yalnızca erişim izni 755 olan dosya/dizinler.

Örnek olması açısından yukarıdaki arama parametrelerinin çıktılarını aşağıya ekliyorum.

```
root@taylan:~/Desktop# find /etc -name *local -type f/etc/X11/Xsession.d/35x11-common_xhost-local/etc/dconf/db/local/etc/gi
```

xargs

xargs komutu kabaca kendisinden önce verilen girdi verilerini kendisinden sonrakine tek tek aktarmaktır. Biliyorum böyle denilince hiçbir şey anlaşılmıyor. O yüzden **xargs** komutunu **find** komutuyla birlikte kullanarak örnek üzerinden komutumuzu ve kullanım mantığını açıklamış olalım.

Örneğin **.jpg** uzantılı bir resim dosyasını araştırıp daha sonra bulduğumuz bu dosyayı silelim. Ben duvar kağıdı için daha önce internetten bir **.jpg** uzantılı bir resim indirmiştım. Bu resim hem indirilenlerde(downloads) hem de resimler konumunda yer alıyor biliyorum ancak komutun kullanımını denemek üzere dosyanın konumunu bilmiyormuşcasına bir test edelim. Testimizde **root** dizinindeki **.jpg** uzantılı dosyaları araştıralım ve silelim.

```
root@taylan:~/Desktop# find /root -name *.jpg/root/Pictures/Wallpapers/HBe3nHJ.jpg/root/Downloads/HBe3nHJ.jpgroot@taylan:~,
```

Çıktıda da görüldüğü gibi `.jpg` uzantılı resim dosyası silmeden önce `/root/Pictures/Wallpapers/HBe3nHJ.jpg` konumunda bulunuyodu. Ancak arama ve silme işlemini gerçekleştirmemiz ile birlikte dosya silindi. Bunu da `/root/Pictures/Wallpapers/` komundayken `ls` komutunu vererek teyit etmiş olduk

Komutumuzun işleyişi tam olarak şöyle oldu;

`find /root -name *.jpg` komutu ile `root` dizini altındaki tüm `.jpg` uzantılı dosyaları listeledik.

Bu listelenen dosya konum ve varlık bilgilerini `xargs` komutuna pipe(`|`) argümanı sayesinde aktardık.

`xargs` komutu ise eline geçen bu bilgiler sayesinde dosyanın konumunu saptayarak silme işlemini gerçekleştirebildi.

`xargs` komutumuz `exec` isimli başka bir komuta benzer şekilde çalışmaktadır. Merak edip öğrenmek isteyen arkadaşlar ayrıca `exec` komutuna bakabilirler.

Dosya Kopyalama-Taşıma-Silme İşlemleri

Bu kısımda dosyaları kopyalama taşıma ve silme gibi işlemleri ele alacağız. Zaten kullanacağımız komutlar kısa oldukları için kullanımı ve aklıda kalması çok kolay. İlk olarak dosya kopyalama işlemi ile anlatıma başlayalım.

cp

`cp` komutu kopyalama işlevindedir. Kullanımı `cp kaynak hedef` şeklindedir. Örnek üzerinden açıklayalım.

```
root@taylan:~/Desktop# lsdosya23.txt dosya3.txt isimler_kopya.txt komut.sh listedosya2.txt dosya.txt isimler.txt linux.txt
```

Komutumuzu açıklayacak olursak;

Desktop dizinindeyken `ls` komutu ile bulunduğumuz dizindeki dosyaları listeliyoruz. Ve çıktıda da görüldüğü üzere masaüstünde merhaba isimli bir belge bulunmuyor.

`cd ..` komutu ile bir alt dizine dönüyoruz.

`ls` komutu ile buradaki dosyaları listeliyoruz. Bu dizinde "merhaba" isimli bir dosya mevcut.

"merhaba" dosyasını `Desktop` konumuna taşımak üzere `cp merhaba Desktop` şeklinde komutumuzu girdik.

Daha sonrasında ise `cd Desktop` komutu ile `Desktop` konumumuza gidip `ls` komutumuzu yazarak "merhaba" dosyamızın kopyalandığını teyit ettik.

Eğer bir dosyayı A dizininden B dizinine taşırsak ve B dizini içerisinde A dizininden taşıdığımız belge ile aynı isimde dosya varsa taşıdığımız dosya mevcut konumda bulunan aynı isimdeki eski dosyanın üzerine yazılacaktır. Bu da B konumunda var olan eski dosyanın önceki içeriğinin yok olması demek. Böyle talihsiz bir durum yaşamamak için `cp` komutunu kullanmadan önce `-i` parametresini de eklememizde fayda var. Bu parametre `cp` komutunun aynı isimli bir dosya ile karşılaşma durumunda bize onay sormasını sağlayacaktır.

Anlatılanları daha iyi anlamak adına hemen bir örnek yapalım. Daha önceden masaüstü konumuna taşımış olduğumuz "merhaba" dosyasını `-i` parametresini kullanarak tekrar `cp` komutu ile taşımaya çalışalım.

```
root@taylan:~# cp -i merhaba Desktopcp: 'Desktop/merhaba''ın üzerine yazılsın mı?
```

Çıktıda da görüldüğü gibi `-i` parametresini kullanarak yazdığımız kopyalama komutu bize `cp: 'Desktop/merhaba''ın üzerine yazılsın mı?` şeklinde bir uyarıda bulundu ve bizden onay bekledi.

Elbette bu işlemleri çoklu dosya taşıma için de kullanabiliriz.

```
root@taylan:~/Desktop# lsdosya23.txt dosya3.txt isimler_kopya.txt komut.sh liste yeni_dizindosya2.txt dosya.txt isimler.tx
```

Hangi komutu neden kullandık zaten bir önceki örneğe benzer olduğu için tekrar açıklamıyorum. Çıktıları inceleyerek çok rahat şekilde anlayabilirsiniz.

Eğer o anda çalıştığımız dizine başka bir konumdan dosya kopyalamak istersek komutumuzu `cp kopyalanacak_dosya_konumu .` şeklinde kullanabiliriz. Nokta(`.`) bizim o anda bulunduğumuz çalışma konumunu ifade ediyor. Örneği incelediğinizde her şey daha net anlaşılacaktır.

```
root@taylan:~# lsDesktop Firefox_wallpaper.png Pictures Videos yeni2Documents merhaba Public yeni yeni3Downloads Music Temp
```

Görüldüğü üzere kopyalanacak olan dosyanın konumunu belirttik ancak bulunduğumuz konuma yani kopyalamak istediğimiz konumun direk adresini belirtmemize gerek kalmadı. Bunun yerine bulunduğumuz konumu temsil eden `.` kullanmamız dosyanın kopyalanmasını sağladı.

Eğer bulunduğumuz konumdan bir üst konuma kopyalamak istersek komutumuzu `cp dosya_adresi ..` şeklinde kullanmamız yeterli olacaktır.

```
root@taylan:~# cd ../root@taylan:~# ls0 dev initrd.img lib64 merhaba proc sbin tmp vmlinuzbin etc initrd.img.old lost+found
```

Çıktıları karşılaştırarak dosyayı bir önceki dizine kopyalamayı iki nokta(`..`) ile kolaylıkla yapabildiğimizi görebilirsiniz.

mv

Bu komutumuz dosya taşıma ve uzantı değiştirme gibi işlemlere sahiptir. Dosya taşıma işlevi tıpkı `cp` komutunda olduğu gibidir.

```
root@taylan:/# mv ../resim.png /root/Pictures/Wallpapers/root@taylan:/# cd .. && ls0 dev initrd.img lib64 merhaba proc sbi
```

Ve dosyamız kullanımı `mv dosya_konumu taşınacak_konum` şeklinde olan kopyalama komutu ile taşınmış oldu.

Şimdi ise `mv` komutunun uzantı değiştirme işlevine göz atalım.

```
root@taylan:~/Pictures/Wallpapers# lsresim.pngroot@taylan:~/Pictures/Wallpapers# mv resim.png resim.jpgroot@taylan:~/Pictu
```

Başta `.png` olan resim dosyamızın uzantısını, komutumuzu `mv dosya_adi dosya_adi_yeni_uzantısı` şeklinde yazarak `.jpg` uzantılı resim dosyasına çevirmiş olduk.

rm

Dosya dizin silme işleminde kullandığımız komuttur. Kullanımı `rm silinecek_dosya_dizin` şeklindedir.

```
root@taylan:~/Pictures/Wallpapers# lsresim.jpgroot@taylan:~/Pictures/Wallpapers# rm resim.jpgroot@taylan:~/Pictures/Wallpa
```

Dosyalar tekil olarak silinebileceği gibi çoklu silme işlemi de gerçekleştirilebilir. Hatta daha önce gördüğümüz joker karakterler ile kullanırsak isim, uzantı gibi filtrelerle sınırlandırarak toplu dosya silme işlemi gerçekleştirebiliriz. Örnek olması açısından masaüstünde yer alan 2 adet `.py` uzantılı dosyayı tek seferde `rm *.py` komutu ile siliyorum.

```
root@taylan:~/Desktop# lscad.py dosya2.txt isimler_kopya.txt linux.txt resim.jpgcom.py dosya3.txt isimler.txt liste yeni_d:
```

Joker karakterlerin kullanım alanları oldukça geniştir. Sizler de ihtiyacınıza göre filtreleme yaparak bu işlemleri istediğiniz doğrultuda gerçekleştirebilirsiniz.

shred

Bu komutumuz sayesinde dosyalarımızı daha güvenli şekilde silebiliriz. **shred** komutu dosyanın içerisine rastgele bitler yazarak dosyanın okunmaz hale gelmesini sağlıyor. Parametresiz kullanımda dosyaya 3 kez rastgele bit eklerken **n** parametresini kullanarak bu eklenecek olan rastgele bit sayısını istediğimiz doğrultuda düzenleyebiliyoruz. Eğer yapılan işlemi komut satırından takip etmek istersek **v** parametresini de kullanabiliriz. Ayrıca **u** parametresini kullanmamız halinde dosya işlem sonrasında silinir.

```
root@taylan:~/Desktop# cat isimler.txtahmetmehmetalihasanhüseyinburakroot@taylan:~/Desktop# shred -n 5 -v isimler.txtshred
```

Dosya Arşiv İşlemleri

Bu bölümde dosya arşivleme, sıkıştırma-açma gibi işlemleri ele alacağız. Zaten bu kavramlar sizlere yabancı gelmemeli zira daha önce hangi işletim sistemini kullanmış olursanız olun öyle ya da böyle karşınıza çıkmıştır. Genellikle yüksek boyutlu dosyaların sıkıştırılmasında veya veri kaybı olmadan güvenli transfer yapabilmek için ve bunlar gibi pek çok sebep dolayısıyla arşiv dosyaları ile sıklıkla karşılaşırız, karşılaşmaya da devam edicez.

Konumuza arşiv dosyası oluşturarak başlayalım.

tar

tar komutunun açılımı (Tape ARchive) şeklindedir. Bu komut bir veya birden fazla dosyayı tek bir forma sokarak arşivlenmiş duruma getirir. Örnekler üzerinden anlatıma devam edelim.

Masaüstünde yer alan ismi "dosya" ile başlayan her şeyi **dosyalar.tar** şeklinde arşivlemek isteyelim. Bunun için komut satırımıza **tar -cf dosyalar.tar dosya*** yazmalıyız.


```
root@taylan:~/Desktop# ls dosya3.txt dosya.txt linux.txt yeni_dizindosya23.txt dosyalar isimler_kopya.txt listedosya2.txt k
```

Kullandığımız komut bütününü tek tek izah edelim;

tar (Tape ARchive) arşivleme işlevini yapan program.

-c (create archive) parametresi joker ***** yardımıyla dosya ismi ile başlayan her şeyi paketleyip **dosyalar.tar** isimli dosyaya yollar.

Kullandığımız **-f** parametresi de hangi dosyaya yazılacağını belirtir.

Dosyaları tekrar açmak için **tar -xf dosyalar.tar** komutunu kullanabiliriz. Veya açılan dosyaları konsol üzerinden takip etmek için **v** parametresi kullanarak aynı işlemi gerçekleştirebiliriz.

```
root@taylan:~/Desktop# tar -xf dosyalar.tar root@taylan:~/Desktop# tar -xvf dosyalar.tar dosya23.txtdosya2.txtdosya3.txtdo:
```

Eğer arşivi başka bir klasöre çıkarmak istersek komutumuzu **tar -xf dosyalar.tar -C hedef_klasör** şeklinde kullanabiliriz.

```
root@taylan:~/Desktop# tar -xf dosyalar.tar -C çıktılarroot@taylan:~/Desktop# ls çıktılar dosya3.txt dosya.txt linux.txt yei
```

Görüldüğü gibi arşivin içerisindekiler öncekilerden farklı olarak direk çalıştığımız dizine değil, belirtmiş olduğumuz hedef klasöre çıkarılmış oldu.

Arşivi çıkarmadan içerisindekileri görmek istersek **tar -tf dosyalar.tar** komutunu kullanabiliriz.

```
root@taylan:~/Desktop# tar -tf dosyalar.tar dosya23.txtdosya2.txtdosya3.txtdosyalar/dosya.txt
```

Oluşturmuş olduğumuz arşive sonradan ekleme yapmak için **-r** parametresini kullanabiliriz.

Hemen `dosyalar.tar` arşivine masaüstünde yer alan `komut.sh` dosyasını eklemeye çalışalım.

```
root@taylan:~/Desktop# tar -tf dosyalar.tardosya23.txt dosya2.txt dosya3.txt dosyalar/dosya.txt
root@taylan:~/Desktop# tar -rv-
```

Başta ve sondaki arşivin durumunu karşılaştırdığımızda `komut.sh` dosyamızın arşive eklenmiş olduğunu gördük.

`tar` komutunun kullanım şekilleri elbetteki yukarıdaki anlatılanlar ile sınırlı değil. Ancak biz genel olarak sıklıkla karşılaşılan kullanım şekillerini ele almış olduk. Daha fazla bilgiye nasıl ulaşacağınızı biliyorsunuz.

gzip-gunzip-bzip2-bunzip2

Biz daha önceki kısımlarda `tar` komutu ile dosyaları sadece arşivlemiş ancak sıkıştırma işlemi yapmamıştık. Şimdi ise gerektiğinde dosyaları sıkıştırmayı ve bu dosyaları açmayı göreceğiz.

Sıkıştırma işleminde iki temel araç kullanılıyor. Bunlar `gzip` ve `bzip2` araçları. Sıkıştırılmış olan dosyaları açma işlemi ise `gzip` ile sıkıştırılan dosyalar için `gunzip`, `bzip2` ile sıkıştırılan dosyalar için ise `bunzip2` kullanılıyor.

Masaüstünde yer alan dosyalardan örnek yapalım hemen. Ben `isim` adıyla başlayan dosyaları `gzip` arşivine almak için daha önceden de defalarca kullanmış olduğumuz `*` joker karakterinin yardımıyla `gzip isim*` komutunu veriyorum. Daha sonra aynı işlemi dosya adı ile başlayan belgeler için bu defa `bzip2` yardımıyla `bzip2 dosya*` şeklinde arşivleyerek gerçekleştiriyorum.

```
root@taylan:~/Desktop# gzip isim*
root@taylan:~/Desktop# bzip2 dosya*
bunzip2: Input file dosyalar is a directory.
```

Çıktıda da görüldüğü gibi `bzip2` arşivi `.bz2` `gzip` arşivi ise `.gz` şeklinde uzantılara sahip oluyor.

Sıkıştırmış olduğumuz bu dosyaları `gzip` için `gunzip` ve `bzip2` için `bunzip2` komutları yardımıyla tekrar açabiliyoruz. Daha önceden sıkıştırmış olduğumuz dosyaları açarak komutlarımızı test edelim.

```
root@taylan:~/Desktop# gunzip isim*
root@taylan:~/Desktop# ls
cıkıtkılar dosya3.txt.bz2 dosya.txt.bz2 linux.txt yeni_dizindosy:
```

Artık hem arşivleme hem de sıkıştırma işlemlerini aördükten sonra her ikisini de birlikte kullanma vakti geldi.

`gzip` ile `bzip2` arasında `tar` komutu yardımı ile arşivleme işlemi yaparken ufak bir fark var. Bu fark `gzip` ile arşivleme yaparken komutun `czvf` şeklinde parametre almasıyla, `bzip2` ile arşivleme işlemi yaparken aldığı parametrenin `cjvf` olmasıdır. Bunun dışında kullanım şekilleri aynıdır.

Örnekler ile açıklayalım.

Her iki şekilde de dosya isimli belgeleri bir arşiv içine almaya çalışalım. Sırasıyla bunu ilk olarak `gzip` ile daha sonra ise `bzip2` ile yapalım. Dosya adı verirken kullanılan `tar.gz` ve `tar.bz2` uzantılarına dikkat edin aksi halde işlem hata verecektir.

```
root@taylan:~/Desktop# tar czvf dosyaarşivi.tar.gz dosya*dosya23.txtdosya2.txtdosya3.txtdosyalar/dosyalar.tardosya.txtroot@
```

Gördüğünüz gibi `gzip` kullanırken parametre olarak `czvf` verdik ve oluşturduğumuz dosyanın uzantısını da `tar.gz` şeklinde yaptık. Aynı şekilde `bzip2` için ise `cjvf` parametrelerini kullandık ve dosya adımızın uzantısını `tar.bz2` şeklinde oldu. Daha önce de söylediğim gibi bu ayrıntılara dikkat etmezseniz işlem kaçınılmaz olarak başarısız olacaktır.

Sıkıştırmış olduğumuz arşivleri tekrar açmak için ise `gzip` için `xzvf` parametresi `bzip2` için ise `xzjf` parametresini kullanacağız.

```
root@taylan:~/Desktop# tar xjvf dosyaarşivi.tar.bz2dosya23.txtdosya2.txtdosya3.txtdosyaarşivi.tar.gzdosyalar/dosyalar.tard
```

Arşivler açılmış oldu. Eğer arşivi belirli bir konuma çıkarmak istersek daha önce de öğrenmiş olduğumuz şekilde `-C` parametresi ile bu işlemi gerçekleştirebiliriz.

```
root@taylan:~/Desktop# tar xjvf dosyaarşivi.tar.bz2 -C çıktılardosya23.txtdosya2.txtdosya3.txtdosyaarşivi.tar.gzdosyalar/d
```

Daha önceden öğrendiğimiz işlemleri de aynı şekilde sıkıştırılmış arşiv dosyalarında da yapabiliyoruz. O yüzden aynı şeyleri burada tekrardan vermemiz anlamsız olur.

İlk başlarda akılda kalması hatırlanması zor gelebilir ancak sizler de zamanla, kullana kullana bu komutlara alışacaksınız. Unuttuğunuz yerde zaten tekrar açıp bakabilirsiniz.


```
root@taylan:~/Desktop# zip resim.zip resim.jpg adding: resim.jpg (deflated 1%)
```

Eğer `zip` komutu ile sıkıştırdığımız dosyayı açmak istersek `.zip` dosyalarını açma işlevindeki `unzip` komutunu `unzip dosya_adı.zip` şeklinde kullanabiliriz.

```
root@taylan:~/Desktop# unzip resim.zipArchive: resim.zip inflating: resim.jpg
```

Eğer `.rar` dosyasını açmak istiyorsak `unrar` komutumuzu `unrar x dosya_adı.rar` şeklinde konsola girerek bu işlemi gerçekleştirebiliriz. Burada yer alan `x` ifadesi dışarı aktarma işlevindedir. Ayrıca başka parametreler de mevcut ancak bu kısım için bu kadarı yeterli.

Erişim Yetkileri

Linux sistemlerinin yapısı gereği güvenlik açısından dosya ve dizinlere ait birçok kısıtlama ve yetkilendirme ayarları vardır. Bu yetkilerin hepsine sahip olan tek kullanıcı ise root kullanıcısıdır. Bu yüzden sistemi kullanırken eğer root kullanıcısı isek önümüzde uyarıcı bizi kısıtlayıcı bir mekanizma olmayacağından kimi durumlarda yapacağımız değişiklikler olumsuz sorunlara yol açabilir. Bu yüzden root kullanıcısıyken yapılan işlemlere dikkat etmek gerekir.

Bu girizgahtan sonra her kullanıcının kendine verilen yetkiler çerçevesince hareket edebildiğini öğrenmiş olduk. Kullanıcıların dosya veya dizinler ile ilgili yapabileceği üç eylem bulunmaktadır. Bunlar;

okuma(r): Klasör listesini ve dosya içeriğini görüntüleme.

yazma(w): Dosya veya klasör üzerinde değişiklik yapma.

çalıştırma(x): Hedef dosyayı çalıştırma veya klasöre içerisine erişme.

Aslında bu eylemler sizlere yabancı gelmemeli. Zira daha önce `chmod` komutu yardımı ile gerekli dosyanın iznini değiştirmiş ve dosyamızı çalıştırmayı başarmıştık.

Şimdi mevcut dosyalarımızın ne tür izinleri barındırdığına göz atalım. Bunun için komut satırımıza ayrıntılı liste yazdırmak üzere `ls -l` komutumuzu veriyoruz.

```
root@taylan:~# ls -ltoplam 2484drwxr-xr-x 5 root root 4096 Ara 12 04:50 Desktopdrwxr-xr-x 2 root root 4096 Eki 7 15:26 Docu
```

Şimdi listemizi inceleyerek bulunan izinleri ele alalım.

`drwxr-xr-x` ve `-rw-r--r--` şeklinde gördüğümüz kısımlar dosya izinlerini ifade ediyor.

Bazı ifadelerin başında olan `d` harfi o ifadenin dizin olduğunu belirtiyor.

Geriye kalan kısımları ayrı ayrı açıklayacak olursak `-` işareti ile ayrılan kısımlar o izine sahip kullanıcı grubunu temsil ediyor. Daha iyi anlamak için `d` harfi hariç `-` işaretini ayırdığımız zaman geri kalan harfleri üç adet üçlü grup haline getirelim;

`rwXr-xr-x` = `rwX` `r-x` `r-x`

`rw-r--r--` = `rw-` `r--` `r--`

Sırayla birinci harf kümesi dosya sahibinin izinlerini, ikinci harf kümesi grup izinleri ve son küme de diğer kullanıcıların izinlerini belirtir. Buna göre yukarıdaki dosyalarda;

r : okuma yetkisi

w : yazma yetkisi

x : çalıştırma yetkisi

rwX : dosyanın sahibi olan kullanıcı okuyabilir, yazabilir, çalıştırabilir.

r-x : dosya sahibi grup ile aynı grup okuyabilir, çalıştırabilir fakat yazamaz.

r-x : diğer kullanıcılar okuyabilir, çalıştırabilir fakat yazamaz.

Yetkilerin Değişimi(chmod)

Erişim yetkisini değiştirme işlemini ancak en yetkili kişi olan root yapabilir. Bu değişim işlemini daha önceden de kullandığımız `chmod` komutu sayesinde gerçekleştirilir.

`chmod` komutunun parametrelerini tanıyarak örnek verme işlemine geçelim.

u : Dosya-dizinin sahibi

g : Dosya-dizinin sahibi ile aynı grupta bulunan kullanıcılar

o : Diğer kullanıcılar

a : Herkese açık.

= : Yetki eşitleme

+ : Yetki ekleme

- : Yetki çıkarma

Genel parametreleri gördüğümüze göre gelin birkaç örnek yapalım.

Örnek göstermek adına anlatımı, içerisindeki dosyaların hiç birinde yetkinin bulunmadığı bir klasör üzerinden gerçekleştireceğim.

İlk olarak klasörde yer alan dosyaların herhangi bir yetkiye sahip olmadıklarını teyit etmek için ayrıntılı çıktı almak üzere `ls -l` komutunu kullandık.

```
root@taylan:~/Desktop/çıkıtlar# ls -ltoplam 32----- 1 root root 0 Ara 4 15:45 dosya23.txt----- 1 root root 378 Aı
```

Daha sonra klasörde yer alan tüm dosyalara `*` joker karakteri ile ulaştık ve `chmod +w *` komutumuzu kullanarak herkese açık olacak şekilde yazma(w) yetkisi verdik.

```
root@taylan:~/Desktop/çıkıtlar# chmod +w *root@taylan:~/Desktop/çıkıtlar# lsdosya23.txt dosya3.txt dosyalar dosya.txtdosya:
```

Aynı grupta bulunan kullanıcılar için yine konumumuzda bulunan tüm dosyaları `*` sayesinde kapsayacak şekilde `g+rx *` komutumuzu verdik.

```
root@taylan:~/Desktop/çıkıtlar# chmod g+rx *root@taylan:~/Desktop/çıkıtlar# ls -ltoplam 32--w-r-x--- 1 root root 0 Ara 4 1!
```

Gruptaki kullanıcılara(g), okuma-yazma-çalıştırma yetkisi (rwx), kullanıcıya(u) yazma yetkisi(r), diğer kullanıcılara ise yalnızca çalıştırma yetkisi(x) verdik.

```
root@taylan:~/Desktop/çıkıtlar# chmod g+rwx,u+w,o+x *root@taylan:~/Desktop/çıkıtlar# ls -ltoplam 32--w-rwx--x 1 root root 0 Ara 4 :
```

Ve en son yine bulunduğumuz konumdaki tüm dosyaların yetkilerini kaldırdık.

```
root@taylan:~/Desktop/çıkıtlar# chmod a-rwx *root@taylan:~/Desktop/çıkıtlar# ls -ltoplam 32----- 1 root root 0 Ara 4 :
```

Bu kullanımların dışında yetkilendirme işlemleri daha önceden de rastladığımız ve fark etmeden de olsa kullanmış olduğumuz sayısal şekilde de ifade edilebiliyor.

Bu durumu yetkilerin sayısal karşılığını vererek anlatmaya devam edelim.

Yetki kalıplarının sayısal karşılıkları.

#	dosyanın sahibi	sahibiyle aynı gruptakiler	diğer kullanıcılar
r	4	4	4
w	2	2	2
x	1	1	1

Yetkilerin sayısal değerlerini kullanarak bir örnek yapalım.

Örneğin biz sadece dosyanın sahibine bütün yetkileri vermek istiyoruz diyelim. Bunun için ilk başta yetki kalıplarının numara karşılıklarını toplamalıyız. Yani bütün yetkileri vericeğimiz için $r=4 + w=2 + x=1$ =toplam sayı 7 etti. Bizler de sadece dosya sahibine bu yetkiyi vermek istediğimizden normalde vereceğimiz `chmod rwx- - - - -` komutumuzu diğer kullanıcılara yetki vermek istemediğimiz için o alanlarının 0 bırakarak komutumuza `chmod 700 dosya` şeklinde veriyoruz. Böylelikle sadece dosyanın sahibi tüm yetkilere sahip olmuş oldu.

```
root@taylan:~/Desktop/çıkıtlar# ls -ltoplam 32----- 1 root root 0 Ara 4 15:45 dosya23.txt----- 1 root root 378 Ar
```

Daha net anlaşılması adına bir örnek daha yapalım. Şimdi de; dosyanın sahibine tüm yetkileri, ortak gruptakilere yalnızca yazma yetkisini, diğer kullanıcılara da sadece okuma yetkisini verelim.

Dosya sahibi kullanıcıya verilecek tüm yetkiler için $r(4)+w(2)+x(1)=7$ sayısını kullanacağız.

Dosya sahibi ile ortak gruptaki kullanıcılar için vereceğimiz yazma yetkisi için yazma(w) karakterinin sayısal karşılığı olan 2 sayısını kullanacağız.

Diğer kullanıcılar için vereceğimiz yalnız okuma yetkisi için ise okuma(r) karakterinin sayısal karşılığı olan 4 sayısını kullanacağız.

```
root@taylan:~/Desktop/çıkıtlar# ls -ltoplam 32----- 1 root root 0 Ara 4 15:45 dosya23.txt----- 1 root root 378 Ar
```


Çıktıdan da anlaşılacağı üzere sayısal karşılıklar istediğimiz yetkilendirme işlemini gerçekleştirdi.

Son bir ayrıntı daha verelim. Eğer verdiğimiz izinlerin o dizinle beraber alt klasörlerinde de etkili olmasını istersek komutumuzu `-R` parametresi ile birlikte kullanmalıyız.

Eğer örneklere ve açıklamalara rağmen yine de anlamadıysanız ister konuyu tekrar okuyup kendiniz de alıştırma yapın isterseniz de bu konuyu şimdilik geçin ihtiyacınız olduğunda burada olduğunu bilerek tekrar göz atın. Seçim sizlere kalmış.

chatr

Hepimizin başına mutlaka gelen ve çok can sıkıcı bir durum var. Bu durum yanlışlıkla silinen dosyalar. Her nasıl ve neden olursa olsun eğer önemli gördüğümüz dosyalar varsa bir şekilde silinmeden onları koruma altına almamız mümkün. Bizlere bu koruma imkanını veren komut `chatr` komutudur. Öyle ki yanlışlıkla dosyanın kaybolmasına engel olmak adına root kullanıcısının bile değişiklik yapmasına imkan tanımıyor.

Komutumuzun kullanımı `chatr +i dosya_adı` şeklindedir. Hemen bir örnek yapalım.

```
root@taylan:~/Desktop/çıkrtılar# chatr +i dosya.txtroot@taylan:~/Desktop/çıkrtılar# rm dosya.txtrm: 'dosya.txt' silinemedi:
```

`dosya.txt` isimli dosyamızı `chatr +i dosya.txt` komutu ile koruma altına aldık.

Daha sonra bunu teyit etmek için `lsattr` komutunu kullandık.

Komutumuzun çıktısında görüldüğü gibi dosyamızın sol tarafında izinler kısmında `-i` şeklinde bir ifade var. İşte bu ifade dosyamızın artık düzenlenemeyecek olduğunun işaretidir.

Eğer bu işlemi geri almak ve dosyamızı üzerinde değişiklikler yapılabilir hale getirmek istersek `chatr -i dosya.txt` komutumuzu kullanmamız yeterli olacaktır. Örneği aşağıda inceleyebilirsiniz.

```
root@taylan:~/Desktop/çıkrtılar# chatr -i dosya.txtroot@taylan:~/Desktop/çıkrtılar# lsattr-----e---- ./dosya2.txt--
```

Çıktıda da görüldüğü gibi hedef dosyamızın solundaki `-i` işareti yok olmuş, dolayısı ile `dosya.txt` isimli dosyamızın artık eski halinde dönerek düzenlenebilir forma girmiş olduğunu gördük.



Kullanıcı İşlemleri

Linux'ta birçok kullanıcı olduğunu ve bunlar içinde en yetkilinin root kullanıcısı olduğunu önceki derslerimizde tecrübe etmiştik. Ancak sistemde root kullanıcısı olmadan da root kullanıcısının yetkilerine sahip olmamız mümkün. Bu imkanı bize linux grup sistemi veriyor.



Grup Yönetimi

Aynı grupta yer alan kullanıcılar bizim tanımlamamızla aynı haklara sahip olabiliyorlar.

Biraz daha ayrıntılı vermek gerekirse Linux ve UNIX sistemlerindeki kullanıcılar bir veya daha fazla gruba aşağıdaki nedenlerle atanır:

- Dosyaları veya diğer kaynakları az sayıda kullanıcıyla paylaşmak için yani sınırlama mekanizması gibi düşünülebilir.
- Kullanıcı yönetim ve denetim kolaylığı sağlar.
- Grup üyeliği büyük Linux sistem kurulumu yerine kullanılan mükemmel bir çözümdür.
- Grup üyeliği, bu gruba izin verilen dosyalara, dizinlere veya cihazlara özel erişim sağlar. Bu madde baştaki madde ile benzer amaca hizmet ederek tamamen kontrolü elde tutmayı sağlamak için kullanılır.

Mevcut grupları görüntülemek istersek `less /etc/group` `more /etc/group` `cat /etc/group` komutlarından herhangi birini kullanabiliriz. Çıktıları aşağıdaki şekilde olacaktır.(Çıktıyı uzun olması nedeni ile kısa kesiyorum)

```
root@taylan:/etc# cat /etc/grouproot:x:0:daemon:x:1:bin:x:2:sys:x:3:adm:x:4:tty:x:5:disk:x:6:lp:x:7:mail:x:8:...rtkit:x:11!
```

Hemen bu çıktıda yer alan kısımlara bir açıklık getirelim. Bunun için çıktıda yer alan bir grubu bölüm bölüm numaralandırarak açıklayalım. Ben bu durum için örnek bir grup belirtiyorum.

```
cdrom:x:24:taylan,dev_usr,neo_____| | | | | 1 2 3 4
```

Belirtilen numaralı kısımların açıklamaları:

1. Grup_ismi : Gruba verilen isimdir.

2. Parola : Genelde parola kullanılmaz ancak kimi durumlarda kullanıldığı oluyor, bizim çıktımızda da x ile belirtilen alan parola kısmının boş olduğunu belirtiyor. Bu parola belirleme işlemi çok sık kullanılmazsa da, ayrıcalıklı gruplarda uygulamak için yararlıdır.

3. Grup Kimliği (GID) : Atanan grup kimliğini(numarası) belirtiyor.

4. Grup Listesi : Grubun üyesi olan kullanıcıların kullanıcı adlarının bir listesidir. Kullanıcı adları, virgülle ayrılmış olmalıdır.

Kullanıcıların ait olduğu grupları görmek istersek komut satırına `id kullanıcı_adı` şeklinde komut vermemiz yeterli olacaktır.

```
root@taylan:~# id rootuid=0(root) gid=0(root) gruplar=0(root)root@taylan:~# id dev_useruid=1002(dev_user) gid=1002(dev_user)
```

Burada yeri gelmişken daha önce değindiğimiz uid(user id) ve gid(group) kavramlarına değinelim. Bu numaralar kullanıcı çeşidine göre şu şekildedir;

- **root kullanıcısı :** UID=0, GID=0
- **sistem kullanıcısı :** UID=1-499, GID=1-499
- **normal kullanıcı :** UID=500<, GID<

Ayrıca `id` komutunun da birçok parametresi vardır bazıları şimdi göreceğiniz grup oluşturma kısmındaki parametrelerden oluşuyor. Detaylı bilgi için man sayfasına bakabilirsiniz. Ben yinede örnek olması açısından `g` parametresini gösteriyorum ve daha sonra grup oluşturma ile konumuza devam ediyoruz.

`id -g grup_adı` belirtilen grubun grup id(GID)'sini vericektir.

Yeni bir grup oluşturmak istersek `groupadd grup_adı` şeklinde komutumuzu kullanırız.

Ve oluşturduğumuz grubu sorgulamak için grub bilgilerinin tutulduğu dosyaya bakmak üzere `cat /etc/group | grep grup_adı` komutunu kullanabiliriz.

```
root@taylan:~# groupadd yeni_gruproot@taylan:~# cat /etc/group | grep yeni_grupyeni_grup:x:1003:
```

Ayrıca grup oluşturulurken kullanılabilecek bazı parametreler var. Bunlar;

g : Grup id belirleme. İstediğiniz numarayı başka bir gruba ait numara ile aynı olamayacak şekilde verebilirsiniz.

```
root@taylan:~# groupadd -g 1234 GNUroot@taylan:~# cat /etc/group | grep GNUGNU:x:1234:
```

Eğer aynı grup id ile başka bir grup eklemek istersek konsol bize bu id ye sahip başka bir grubun halihazırda bulunduğu uyarısını verecektir. Dolayısı ile grup ekleme işlemi başarısız olacaktır.

```
root@taylan:~# groupadd -g 1234 taylangroupadd: GID '1234' already exists
```

o : id olmadan grup oluşturma

r : sistem grubu oluşturma.

f : işlemi hatalar olsa bile zorlayarak tamamlar. Genelde bu kullanım sorunlar çıkardığı için pek edilmez.

Parametreler bunlar ile sınırlı değil ancak sizler `man groupadd` ve `groupadd --help` komutları yardımı ile diğer parametrelere ulaşabilirsiniz.

Kullanıcı Yönetimi

Sistemimize yeni bir kullanıcı eklemek için kullanabileceğimiz iki farklı komut var. Bunlar `adduser` ve `useradd` komutlarıdır.

adduser

Yeni kullanıcı eklemek için komutumuzu `adduser kullanıcı_adi` şeklinde veriyoruz. Böylelikle oluşturduğumuz kullanıcıya ait ev dizini `home/kullanıcı_adi` şeklinde otomatik olarak oluşmuş oluyor.

```
root@taylan:/etc# adduser taylanAdding user `taylan' ...Adding new group `taylan' (1004) ...Adding new user `taylan' (1003)
```

Çıktıları inceleyecek olursak ev dizini otomatik olarak oluşturulmuş ve kullanıcı "taylan" grubuna dahil edilmiş. Bunu da tekrar ev dizini sorgusu için `home` dizinine bakarak ve `id taylan` komutunu kullanarak teyit edelim.

```
root@taylan:/home# lsdev_user lost+found taylan user
```

Çıktıda görüldüğü gibi `home` dizininde oluşturmuş olduğum "taylan" isiminde kullanıcı dosyası bulunuyor.

```
root@taylan:/etc# id taylanuid=1003(taylan) gid=1004(taylan) gruplar=1004(taylan)
```

Yukarıdaki çıktıda oluşturulan **taylan** isimli kullanıcının grubu da "taylan" olarak gözükmekte.

useradd

Kullanıcı oluşturmak için bahsettiğimiz 2. komut olan **useradd** komutu kullanımı **useradd -m kullanıcı_adı** şeklindedir. Burada komutumuz ile birlikte kullanmış olduğumuz **m** parametresi ile oluşturduğumuz kullanıcıya ait ev dizininin otomatik oluşmasını sağladık. Birde son olarak kullanıcıya ait bir parola belirlemek üzere **passwd kullanıcı_adı** şeklinde komutumuzu vererek parolamızı oluşturuyoruz.

Ayrıca bu **passwd kullanıcı_adı** komut kullanımı bütün hesaplar için şifre güncelleme işleminde de kullanılıyor. Eğer herhangi bir hesabınızın parolanızı güncellemek isterseniz **passwd** komutu aklınızda bulunsun.

```
root@taylan:/etc# useradd -m taylan_2root@taylan:/etc# passwd taylan_2Yeni parolayı girin: Yeni parolayı tekrar girin: pas:
```

Görüldüğü gibi "taylan_2" isimli kullanıcı hesabımız başarılı şekilde oluşturulmuş oldu. Şimdi home dizinini ve oluşturduğumuz kullanıcının grubunu kontrol edelim.

```
root@taylan:/home# lsdev_user lost+found taylan taylan_2 userroot@taylan:~# id taylan_2uid=1004(taylan_2) gid=1235(taylan_2)
```

Çıktı sonuçlarında bir problem olmadığına göre konumuza, kullanıcı silme işlemi ile devam edebiliriz.

Kullanıcı Silmek

Eğer kullanıcıyı **adduser** komutu ile oluşturduysak, oluşturulan ev dizini ile birlikte kullanıcıyı silmek için **deluser --remove-home kullanıcı_adı** komutunu kullanmamız gerek.

```
root@taylan:/home# deluser --remove-home taylanLooking for files to backup/remove ...Removing files ...Removing user `tayl:
```

Ancak kullanıcıyı `useradd` komutu ile oluşturduysak, oluşturulan ev dizini ile birlikte kullanıcıyı silmek için `userdel -r kullanıcı_adi` komutunu kullanmamız gerek.

```
root@taylan:/home# userdel -r taylan_2userdel: taylan_2 mail spool (/var/mail/taylan_2) not found
```

Kullanıcı silme işlemlerini kontrol etmek adına `home` dizinini kontrol edelim.

```
root@taylan:/home# lsdev_user lost+found user
```

Çıktıda `taylan` ya da `taylan_2` bulunmadığına göre kullanıcı silme işlemimiz başarılı şekilde tamamlanmış oldu.

Kullanıcı Bilgileri Listelemek(chage)

Kullanıcı bilgileri listeme işleminde `chage` komutu `chage -l kullanıcı_adi` şeklinde kullanılıyor.

Hemen `root` kullanıcısı için bilgileri listeleyelim.

```
root@taylan:/home# chage -l rootSon Parola Değişimi: Kas 28, 2017Parola Kullanım Süresi Dolumu: Hiçbir zamanParola Pasif: Hiçbir zamanHesap Bitimi: Hiçbir zamanŞifre değişiklikleri arasındaki en az gün sayısı: 0Maksimum giriş denemesi sayısı aşıldı : 99999Şifre süresinin dolumundan önceki uyarı gün sayısı: 7
```

Aktif-Pasif Hesap Ayarlama

Varolan bir kullanıcı hesabını kilitlemek için `usermod -L kullanıcı_adi` komutunu kullanırız.

Pasif durumdaki hesabı aktif hale getirmek için ise `usermod -U kullanıcı_adi` komutu kullanılır.

Kimlik Değişimi

Örneğin ben "taylan" kullanıcısıyken, yapmak istediğim işlem ancak root kullanıcısının yetki alanında ise root hesabının parolasını biliyorsam root hesabının kimliğine bürünerek o işlemi gerçekleştirebilirim.

Geçiş işlemleri için `su` komutunu kullanıyoruz. Komutun kullanımı ile ilgili iki farklı durum var bunlar:

- **su kullanıcı_adi** : diğer kullanıcı kimliğine geçiş yapar.
- **su - kullanıcı_adi** : diğer kullanıcı kimliğine geçiş yapar ve direk olarak geçiş yapılan kullanıcının kabuğunda çalışmaya başlar.

Bu durum en iyi örnekler ile açıklanabilir. İlk önce root kullanıcısıyken "taylan" isimli kullanıcı hesabına giriş yapıcım daha sonra "taylan" isimli kullanıcıdan root hesabına giriş işlemini gerçekleştirecem. Adımları sırasıyla takip edin lütfen.

Hemen mevcut kullanıcı oturumunu daha önce öğrenmiş olduğumuz `whoami` komutu ile sorgulayalım.

```
root@taylan:~# whoamiroot
```

root kullanıcı iken "taylan" hesabına geçiş yapmak için `su taylan` komutunu kullanıyoruz. Ve `whoami` komutu ile geçiş durumunu kontrol ediyoruz.

```
root@taylan:~# su taylantaylan@taylan:/root$ whoamitaylan
```

Geçiş yaptığımız "taylan" hesabından root hesabına dönmek için `exit` komutunu kullanmamız yeterli.

```
taylan@taylan:/root$ exitexitroot@taylan:~# whoamiroot
```

Şimdi `su - kullanıcı_adı` komutu kullanarak geçiş yapacağımız kullanıcı kabuğunda çalışalım.

```
root@taylan:~# whoamirootroot@taylan:~# su - taylantaylan@taylan:~$ whoamitaylantaylan@taylan:~$
```

Fark etmiş olacaksınız ki başta geçiş yapabilmemiz için geçeceğimiz hesabın parolasını bilmemiz gerektiğini söylemiştim ancak "taylan" hesabına yaptığımız geçişlerde parola sorulmadı. Bunun sebebi "taylan" hesabının normal kullanıcı olması. Şimdi bu durumu birde "taylan" hesabındayken root kullanıcısı için iki farklı kullanımda da deneyelim.

```
taylan@taylan:~$ su rootPassword: root@taylan:/home/taylan# whoamirootroot@taylan:/home/taylan#
```

```
taylan@taylan:~$ su - rootParola: root@taylan:~# whoamirootroot@taylan:~#
```

Süreçler(process)

Süreç(process) denilen kavram genel olarak bir program çalıştığında programın belleğe yüklenmesine denir. Bizler de bu bölümde süreçler hakkında bilgi almak, işlem yapmak ve gerektiğinde müdahalede bulunmak gibi işlemleri yapabilmek adına ilgili komutları öğrenecez.

ps

Bu komutumuz çalışan süreçleri görüntülememizi sağlıyor.

Komutumuz tek başına **ps** şeklinde kullanıldığında o anda çalışmakta olan süreçleri veriyor.

```
root@taylan:~# ps PID TTY TIME CMD 1699 pts/0 00:00:00 bash 6213 pts/0 00:00:00 ps
```

Gerektiği zaman doğru çıktıları almak adına **ps** komutumuzun parametrelerine bakalım.

Genel Liste

a : Tüm terminallerde çalışan süreçler.

u : Hedef kullanıcıya göre süreçleri listeler.

x : tty'lerde çalışanlar hariç tüm süreçleri verir.

e : Komutlar ve parametrelerini göstermektedir.

f : Tam çıkış şeklinde listeler.

l : Çıkış listesini uzunca verir.

c : Süreçlerin yalnız komut isimlerini verir.

v : Kullanılan hafızalara göre gösterir.

Şimdi bazı parametrelerimizi çıktıları ile birlikte inceleyelim.

a , **u** ve **x** parametrelerini genellikle kullanıldığı gibi yani bir bütün olarak **-aux** şeklinde ele alalım.

Komutumuz **aux** parametreleri ile **ps -aux** şeklinde kullanıldığında sistemde ve terminallerde çalışan tüm süreçleri ve süreç sahiplerini gösterir.

Çıktı çok uzun olacağından üç nokta(...) ile aradaki çıktıları atlادم.

```
root@taylan:~# ps -auxUSER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMANDroot 1 0.0 0.1 219136 7808 ? Ss 02:28 0:01 /sb:
```

e :Sistemde çalışan her süreç.(every)

Çıktı çok uzun olacağından üç nokta(...) ile aradaki işlemleri atlادم.

```
root@taylan:~# ps -e PID TTY TIME CMD 1 ? 00:00:01 systemd 2 ? 00:00:00 kthreadd 3 ? 00:00:00 kworker/0:0 4 ? 00:00:00 kwoi
```

f :Tam çıkış biçimi.(full)

```
root@taylan:~# ps -fUID PID PPID C STIME TTY TIME CMDroot 1699 1693 0 01:51 pts/0 00:00:00 bashroot 6249 1699 0 02:09 pts/i
```

l :Uzun çıkış biçimi.(long)

```
root@taylan:~# ps -lF S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD 0 0 1699 1693 0 80 0 - 5269 - pts/0 00:00:00 bash
```

u : Kullanıcı ile ilgili süreçler.

Parametrenin komut ile kullanımı `ps -u kullanıcı` şeklindedir.

Çıktı çok uzun olacağından üç nokta(...) ile aradaki işlemleri atladım.

```
root@taylan:~# ps -u root PID TTY TIME CMD 1 ? 00:00:01 systemd 2 ? 00:00:00 kthreadd ... 6212 ? 00:00:00 kworker/u256:1 6:
```

c : Süreçlerin komut ismini verir.

```
root@taylan:~# ps -c PID CLS PRI TTY TIME CMD 1699 TS 19 pts/0 00:00:00 bash 6735 TS 19 pts/0 00:00:00 ps
```

v : Süreçleri kullandıkları bellek ile birlikte gösterir.

Çıktı çok uzun olacağından üç nokta(...) ile aradaki işlemleri atladım.

```
root@taylan:~# ps -v PID TTY STAT TIME MAJFL TRS DRS RSS %MEM COMMAND 1219 tty2 Ss1+ 0:00 0 67 203016 5384 0.1 /usr/lib/gdr
```

Eğer aradığımız özel bir süreç varsa daha önce görmüş olduğumuz `grep` komutu yardımı ile o süreç hakkında doğrudan bilgi alabiliriz.

Şöyleki eğer ben bash süreci hakkında bilgi almak istiyorsam konsola `ps -aux | grep "bash"` yazmam yeterli olacaktır.

```
root@taylan:~# ps -aux | grep "bash"root 1699 0.0 0.1 21076 4976 pts/0 Ss 02:29 0:00 bashroot 6778 0.0 0.0 12592 964 pts/0
```

Çıktıda görüldüğü gibi yalnızca ulaşmak istediğim bash süreci hakkında çıktı almış oldum.

pstree

Süreçleri hiyerarşik bir biçimde görüntülemek istersek `pstree` komutumuzu da kullanabiliriz. Bu komut Windows işletim sisteminde de `tree` olarak kullanılabilir. Neyden bahsettiğimizi anlamak için hemen konsolumuza `pstree` komutumuzu verelim.

```
root@taylan:~# pstreesystemd└─ModemManager──2*[{ModemManager}]  └─NetworkManager└─dhclient  |  └─2*[{NetworkManager}]  └─VG/
```

Fark etmiş olacaksınız ki yukarıdaki çıktıyı keserek verdim. Çünkü komutumuzun isminde(`pstree`) de yer aldığı gibi süreçler tıpkı bir ağaç gibi dallanıp budaklanarak hiyerarşik şekilde listelenmiş oldu.

top

Bu komutumuz da tıpkı `ps` komutunda olduğu gibi süreçler hakkında bilgi verir ancak bu işlemi `ps` komutundan farklı olarak 3 saniyede bir yenilenecek şekilde anlık değişimleri verecek şekilde yapar. Temel kullanımı `top` şeklindedir.

```
root@taylan:~# toptop - 04:05:58 up 1:37, 1 user, load average: 0,00, 0,00, 0,00Tasks: 201 total, 1 running, 200 sleeping,
```

Eğer varsayılan olarak kullanılan 3sn de bir güncelleme tekrarında bir değişiklik yapmak istersek komutuzu `d` parametresiyle birlikte yani `top -d saniye` şeklinde yazarak istediğimiz saniye aralığında süreç bilgilerinin çıktılarının güncellenmesini sağlayabiliriz. Eğer bu ekranı kapatmak istesek ise sadece `q` tuşuna basmamız yeterli.

pgrep

Çalışan süreçlerin belirli kriterlere göre sıralanmasını sağlayan komuttur. Süreç işlemlerinde çalışan süreç üzerinde değişiklik yapabilmemiz için ilk önce değişiklik yapacağımız süreci bulmamız gerekiyor. `pgrep` komutumuz da çalışan süreç numaralarını(pid) bize verir. Süreç numarası sistemde süreçlerin haberleşmesini sağlayan numaradır. Buraya takılmadan devam edelim.

Bu komut, süreç numaraları arasında süreçlerin nasıl çalıştığını sağlayan numaraları bulmaya yarar. Buraya bakarak devam edelim.

Örnek vererek komutumuzun kullanımını görmüş olalım.

Örnek göstermek adına armitage aracını çalıştırıyorum. Çalışmakta olan armitage aracının süreci üzerinden örneğimizi yapalım.

```
root@taylan:~# pgrep armitage7683
```

`pgrep armitage` komutumuz ile armitage süreci numaralarını(pid) görmüş olduk.

Şimdi de `pgrep` komutumuzun birkaç işlevsel parametresini görelim.

l : Süreçlerin numaraları ile beraber süreç isimlerini de görmek için kullanılır.

```
root@taylan:~# pgrep -l armitage7417 armitage
```

lu : Belirtilen kullanıcının süreçlerin numaraları ile beraber süreç isimlerini de görmek için kullanılır. Burada **u** parametresi hedef kullanıcı belirtmemizi sağlar.

Çıktı çok uzun olacağından üç nokta(...) ile aradaki işlemleri atladım.

```
root@taylan:~# pgrep -lu root1 systemd2 kthreadd3 kworker/0:0...7409 sh7417 armitage7418 java7439 kworker/0:17460 kworker/1
```

İleride bu bilgiler bize lazım olacak şimdi süreçleri sonlandırma komutları ile devam edelim.

kill-killall

Komutlarımız isimlerinden de anlaşılacağı gibi süreçleri sonlandırma(öldürme) işleminde kullanılıyorlar. Kullanım şekilleri;

kill süreç_numarası(pid) : Süreçleri pid(süreç numarası) ile sonlandırmak.

```
root@taylan:~# pgrep -l armitage7683 armitageroot@taylan:~# kill 7683root@taylan:~# pgrep -l armitage
```

kill -9 süreç_numarası(pid) : Kapanmaya karşı direnen süreçleri pid(süreç numarası) ile sonlandırmak.

```
root@taylan:~# pgrep -l armitage7746 armitageroot@taylan:~# kill -9 7746
```

killall süreç_ismi : Süreçleri isimleri ile sonlandırmak.

```
root@taylan:~# pgrep -l armitage7782 armitageroot@taylan:~# killall armitageroot@taylan:~# pgrep -l armitage
```

killall -9 süreç_ismi : Kapanmaya karşı direnen süreçleri isimleri ile sonlandırmak.

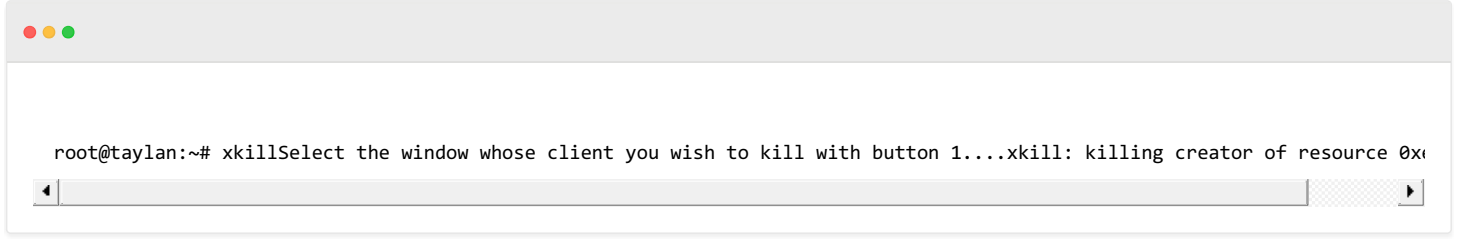
```
root@taylan:~# pgrep -l armitage7845 armitageroot@taylan:~# killall -9 armitageroot@taylan:~# pgrep -l armitage
```

killall -i süreç_ismi : Süreçleri isimlerini kullanarak ve onay alarak sonlandırmak.

```
root@taylan:~# pgrep -l armitage7932 armitageroot@taylan:~# killall -i armitageKill armitage(7932) ? (y/N) yroot@taylan:~#
```

Bu komutumuz yanıt vermeyen bir programı grafiksel arayüzden kapatmamızı sağlayan ilginç bir komuttur.

Diyelim ki leafpad aracımızı açtık ve her ne nedenle olursa olsun yanıt vermeyi kesti bu noktada eğer leafpad aracını anında kapatmak istersek konsola `xkill` yazıyoruz ve fare imleci çarpı işaretine dönüşüyor. Bu çarpı işareti ile tıkladığımız araç otomatik olarak kapanıyor.



```
root@taylan:~# xkillSelect the window whose client you wish to kill with button 1...xkill: killing creator of resource 0x1
```

Anlatım havada kalmış olabilir o yüzden `xkill` komutunun kullanımını kesinlikle sizler de hemen deneyin akılda kalması kolay bir komutumuz zaten.

fg-bg-jobs

Son olarak gayet kullanışlı komutlar olan `fg` ve `bg` komutlarına değinerek süreçler konumuzu bitirelim. Konsoldan komut verildiğinde, verdiğimiz komut doğrultusunda gerçekleşen işlem süreci bitmeden konsoldan tekrar başka bir komut vermemiz mümkün olmuyor. Bu gibi durumlarda eğer yeni bir komut girişi yapmak istersek süreci arkaplana alıp yeni komutumuzu ancak öyle çalıştırabiliriz. İşte bu komutlarımız da konsoldan çalıştırdığımız süreçleri arkaplana taşıma gibi işleve sahiptirler. Ayrıca `jobs` komutumuz da bu süreçlerin durumunu takip etmemizi sağlar. Bu durum örnekler ile daha iyi anlaşılacaktır.

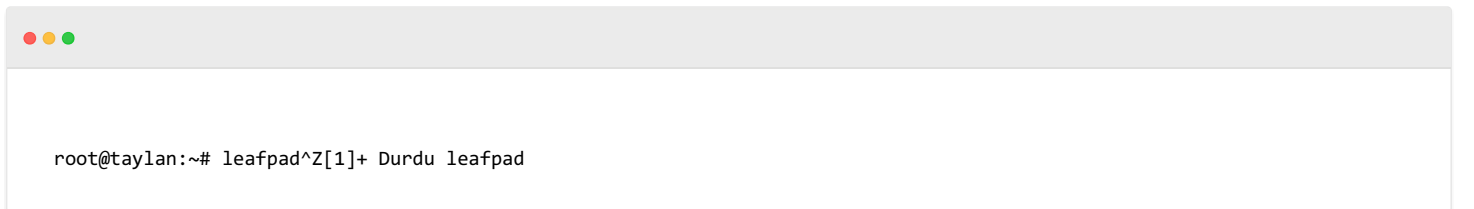
Kullanımın rahat şekilde anlaşılması için adım adım gidelim.

Konsoldan leafpad uygulamasını açalım. Bunun için konsola `leafpad` yazmamız yeterli.

Yazdıktan sonra ekrana leafpad uygulaması geldi.

Programı çalıştırdığımız konsol şu anda leafpad uygulamasını çalıştırmakla meşgul. Yani bu demek oluyor ki eğer biz bu konsol üzerinden yeni bir komut girmek istersek bu mümkün olmayacak. İşte bu yüzden halihazırda çalışan leafpad uygulamasının çalışmasını durdurup arkaplana atmalıyız ki yeni komutlar girdiğimizde konsol leafpad uygulaması ile meşgul olmasın.

Konsolda çalışan leafpad uygulamamızı arkaplana atmak yani duraklatmak için `Ctrl + Z` tuş kombinasyonunu kullanıyoruz. Bu işlemin ardından konsol bize aşağıdaki gibi bir çıktı basarak leafpad uygulamasının durduğunu haber veriyor.



```
root@taylan:~# leafpad^Z[1]+ Durdu leafpad
```

Konsolun bize leafpad programının durduğunu haber verdiğini söylemiştik. Ayrıca bu gibi durumlarda sürecin durumunu sorgulama yapmak için `jobs` komutunu da kullanabiliyoruz.

```
root@taylan:~# jobs[1]+ Durdu leafpad
```

Artık leafpad programımızın durdurulduğundan kesin eminiz. Hatta leafpad uygulamasına grafiksel arayüzden bir şeyler yazmaya çalıştığımızda uygulama durdurulduğu için hiç bir tepki vermiyor. Yani konsolumuz artık leafpad uygulamasının sürecini işlemiyor. Bunu teyit etmek için önceden konsola komut veremediğimizi ele alırsak denemek için konsola `ls` komutumuzu giriyoruz.

```
root@taylan:~# lsDesktop Firefox_wallpaper.png Pictures Templates yeni1 yenisosyaDocuments merhaba Public Videos yeni2Down:
```

`ls` komutumuzun çıktısını alabildik bu da demek oluyor ki konsol ekranımız önceden olduğu gibi leafpad uygulaması ile meşgul değil. Böylelikle yeni komutlar alabiliyor.

Leafpad programı durduruldu ancak biz hem leafpad programını kullanmak hem de aynı konsol ekranından yeni komutlar girebilmek istiyoruz. İşte bu noktada durdurulan leafpad süreçlerinin arkaplana alınarak çalıştırılmasına devam edilmesi gerekiyor. Bu işlevi de `bg` komutu sağlıyor. Komut satırımıza `bg` komutumuzu girdiğimizde artık leafpad programı süreçleri arkaplanda çalışır duruma geçiyor ve biz hem konsol ekranından yeni komutlar girebiliyoruz hem de leafpad uygulamamızı kullanabiliyoruz. Leafpad uygulamasının çalışır olduğunu `jobs` komutu ile de aşağıdaki şekilde teyit edebiliriz.

```
root@taylan:~# bg[1]+ leafpad &root@taylan:~# lsDesktop Firefox_wallpaper.png Pictures Templates yeni1 yenisosyaDocuments :
```

Eğer arkaplana attığımız program sürecini tekrar eskisi gibi çalışır hale getirmek istersek `fg` komutumuzu kullanabiliriz. Bu komutumuzu kullandığımızda konsol artık yeni komut almayacak ve tekrar sadece leafpad uygulaması ile meşgul olacak.

```
root@taylan:~# fgleafpad
```

Disk İşlemleri

Disk ile ilgili yapabileceğimiz bölümlendirme, formatlama, yedekleme ve diğer işlemleri gerçekleştirebileceğimiz çeşitli komutlar var. Bu bölümde bu komutları ve işlevsel yanlarını ele alacağız. Konumuza ilk olarak daha önce de kullanmış olduğumuz `fdisk` komutu ile başlayalım.

Komutlara geçmeden önce disk üzerinde yapacağımız işlemlerde son derece dikkatli olmamız gerektiğini aklınızdan çıkarmayın lütfen.

Yapacağımız yanlışlar dosyalarımızın kalıcı olarak silinmesine ve daha farklı sorunlara yol açabilir.

gparted

Grafiksel basit bir arayüze sahip olmasından dolayı kullanımı en kolay disk aracı denilebilir. Grafiksel arayüze sahip olduğu için açıklamasını burada vermem, resim kullanmayacağım için verimsiz olur. Ancak blog sayfamda ve internetteki kaynaklarda kullanımı hakkında bolca bilgiye ulaşabilirsiniz. Açmak için konsola `gparted` yazmanız yeterli olacaktır.

fdisk

İsminden de anlaşılacağı üzere komutumuz çeşitli disk işlemleri yapılmak üzere kullanılıyor.

Komutumuzun aldığı temel parametreleri ele alarak konumuza devam edelim.

İlk olarak sistemimizde bulunan disk bölümlerini listeleyelim. Bunun için `l` parametresini kullanıyoruz.

`l` : disk bölümlerini listeler.

```
root@taylan:~# fdisk -lDisk /dev/sda: 200 GiB, 214748364800 bytes, 419430400 sectorsUnits: sectors of 1 * 512=512 bytesSector
```

Komutumuzu `fdisk /dev/sda` şeklinde parametresiz olarak vermemiz durumunda konsol bizden komut bekler.

```
root@taylan:~# fdisk /dev/sdaWelcome to fdisk (util-linux 2.29.2).Changes will remain in memory only, until you decide to v
```


Hazır komutumuzu parametresiz kullanmışken bizden komut bekleyen konsola girebileceğimiz parametreleri ele alalım.

m :yardım menüsünü açar. Burada **fdisk** komutu ile kullanabileceğimiz parametreler listelenir.

```
root@taylan:~# fdisk /dev/sdaWelcome to fdisk (util-linux 2.29.2).Changes will remain in memory only, until you decide to i
```

Çıktıda görüldüğü üzere **fdisk** komutumuz disk konusunda çok fazla işleve sahip. Ancak biz sadece birkaç temel işlevini görecez.

p :disk bölümlene tablosunu gösterir

```
root@taylan:~# fdisk /dev/sdaWelcome to fdisk (util-linux 2.29.2).Changes will remain in memory only, until you decide to i
```

Çıktımız tıpkı komutu **fdisk -l** şeklinde kullandığımızdaki gibi oldu.

l : disk bölümlerini listeler.

Çıktının çok uzaması nedeniyle çıktının sonlarına dört nokta (....) koymak sureti ile çıktıları kısaltarak verdim.

```
Command (m for help): 1 0 Empty 24 NEC DOS 81 Minix / old Lin bf Solaris 1 FAT12 27 Hidden NTFS Win 82 Linux swap / So c1 l
```

d : bölüm silmek için kullanılır.

Ben **p** parametresi ile listelediğim disk bölüm tablosundan 4. bölümü yani **/dev/sda4** bölümünü silmek üzere **d** parametresini vererek Partition number kısmını 4 olarak belirttim.

```
Command (m for help): dPartition number (1-4, default 4): 4Partition 4 has been deleted.
```

Çıktıda da görüldüğü gibi seçili alan silinmiş oldu. Artık diskte alan açıldığına göre yeni bir disk bölümü oluşturabiliriz.

n : yeni disk bölümü oluşturur.

n parametremiz ile diskte yeni bir alan oluşturabiliriz. **n** parametremizi girince bize bölümlene tipinin öncelikli(primary) mi yoksa genişletilmiş(extended) mi olması gerektiğini soruyor.Aralarındaki farklar nedir diyecek olursanız.

Primary Partition denilen kısım işletim sistemlerini kurduğumuz sabit disk bölümüdür. Bir sabit diskte maksimum 4 tane primary partition olabiliyor ve bunlardan bir tanesinin mutlaka aktif partition olması gerekiyor ki işletim sisteminiz boot işlemi sırasında çalıştırılabilir.

Extended Partition ise aktif primary partition çıkarıldığında geriye kalan tüm sabit disk alanınızdır, bunun içine pasif primary partitionlar da dahildir.

Bu açıklama yeterli gelmemiş dolayısı ile anlamamış olabilirsiniz ancak kafa karışıklığına sebep olmamak için ayrıntıya girmiyorum.Merak eden arkadaşlar kısa bir araştırma sonucu istediklerinden de fazla bilgiye ulaşabilirler.Şimdi konumuza devam edelim.

```
Command (m for help): n Partition type p primary (3 primary, 0 extended, 1 free) e extended (container for logical partiti
```

First sector alanını **enter** ile geçiyorum, Last sector alanına **+1024M** giriyorum yani 1GB kadar bir alan oluşturmuş oluyorum.Bunu teyit etmek için **p** parametremi kullanıyorum.

```
Command (m for help): pDisk /dev/sda: 200 GiB, 214748364800 bytes, 419430400 sectorsUnits: sectors of 1 * 512=512 bytesSec
```

Çıktıda **/dev/sda4** ayarladığımız gibi yani 1GB olarak gözüküyor.

w :değişiklikleri kaydedip çıkar

Gerekli ayarlamaları ve işlemleri gerçekleştirdikten sonra **fdisk** aracını kaydederek kapatmak istersek **w** parametresini kullanmamız yeterli.

```
Command (m for help): wThe partition table has been altered.Calling ioctl() to re-read partition table.Re-reading the part:
```

q :çıkış yapar

q parametresi direk olarak yapılan deęişiklikleri kaydetmeden **fdisk** aracı konsolundan çıkış yapar.

```
root@taylan:~# fdisk /dev/sdaWelcome to fdisk (util-linux 2.29.2).Changes will remain in memory only, until you decide to v
```

fdisk

fdisk aracının görsel arayüze sahip versiyonudur.Açmak için **cfdisk** komutu yeterli olacaktır.

Ayrıca benden bir tavsiye, **fdisk** 'in kullanımı **cfdisk** 'e oranla biraz zordur, dolayısıyla kolayca hata yapabilir ve istenmeyen durumlarla karşı karşıya kalabilirsiniz. Etkileşimli bir arayüzü olan **cfdisk** 'in kullanımı daha kolay olduğundan hata yapmak daha zordur. Hele birde Linux'unuzu Türkçe kullanıyorsanız, **cfdisk** 'in de Türkçe arayüze sahip olmasından dolayı çok rahat şekilde işlemlerinizi gerçekleştirebilirsiniz.

Kullanımı oldukça kolay olduğundan detaylı anlatımda bulunmuyorum. Çok kolay şekile kendiniz keşfedebilirsiniz.

```
root@taylan:~# cfdisk Disk: /dev/sda Size: 200 GiB, 214748364800 bytes, 419430400 sectors Label: dos, identifier: 0x83e6f0:
```

badblocks

badblocks kısaca, bir aygıtın bozuk bloklarını bulmak için tarama işlemi gerçekleştirme görevindedir.

Sırasıyla kullanım parametrelerine göz atalım:

b : blok uzunluğu bayt cinsinden gösterir.(aşağıdaki kullanımda yer alan **s** ve **v** parametreleri ileride açıklanmıştır)

```
root@taylan:~# badblocks -svb 4096 /dev/sda4Checking blocks 0 to 25402367Checking for bad blocks (read-only test): 0.00% d
```

c : bir seferde sınanacak blok sayısı belirtilir ve bu sayı öntanımlı olarak tek seferde 16 bloktur okuyacak şekildedir. Bu sayının artırılması bu işlemin verimini yani hatalı blokların bulunması olayının daha güvenilir şekilde çalışmasını sağlar ancak arttırıldığı sayıya bağlı olarak bellek kullanımı da artacağı için bu sayının yüksek tutulması durumunda "tamponları ayırmada bellek yetersizliği" hatasını verecek ve işlemi sonlandıracaktır. Tersî durumda yani bu sayının çok düşük tutulması da hatalı blokların gizli kalmasına neden olarak yapılan işin verimini düşürecektir. Yani dengeli kullanım çok önemlidir.

```
root@taylan:~# badblocks -svc 1000 /dev/sda4Checking blocks 0 to 101609471Checking for bad blocks (read-only test): 0.00% done,
```

f : dosya sisteminin zarar görüp sistemin çökebileceği durumlarda **badblocks** tarama yapmayı reddeder. Ancak siz çoğu durumda önerilmese de yine de tarama yapmaya zorlamak isterseniz **f** parametresini kullanabilirsiniz.

```
root@taylan:~# badblocks -svf /dev/sda4Checking blocks 0 to 101609471Checking for bad blocks (read-only test): 0.00% done,
```

p : diskin kaç defa taranacağını belirtir öntanımlı olarak bu değer sıfırdır yani ekstra değer belirtilmezse disk yalnız 1 kez taranır.

```
root@taylan:~# badblocks -svp 2 /dev/sda4Checking blocks 0 to 101609471Checking for bad blocks (read-only test): 0.00% done,
```

s : taranan blokların numaralarını çıktı olarak vererek sürecin gelişimini gösterir.

```
root@taylan:~# badblocks -sc 1000 /dev/sda4Checking for bad blocks (read-only test): 0.00% done, 0:00 elapsed. (0/0/0 errors)
```

v : süreçle ilgili ayrıntılı bilgi verir.

```
root@taylan:~# badblocks -vc 1000 /dev/sda4Checking blocks 0 to 101609471Checking for bad blocks (read-only test): done Pa:
```

Bu parametreler dışında da başka parametreler mevcut konsola `man badblocks` yazarsanız zaten diğer parametreler hakkında da kısaca bilgi sahibi olabilirsiniz.

Neticede `badblocks` komutu ve parametreleri yardımı ile hatalı blok tespiti yapabiliyoruz. Şimdi de bu hatalı blokları düzeltme kısmına gelelim.

fsck

Komutumuzun işlevi diskteki hatalı blokları düzeltmeye çalışmaktır. Aslında bu işlem sistemimizi her açışımızda otomatik olarak gerçekleşir. Biz de şimdi örnek olması açısından USB belleği onarmayı deneyelim.

USB belleğimizin disk adını öğrenmek için sistemimizdeki diskleri `fdisk -l` komutu ile listeleyelim.

```
root@taylan:~# fdisk -lDisk /dev/sda: 100 GiB, 107374182400 bytes, 209715200 sectorsUnits: sectors of 1 * 512=512 bytesSec:
```

Çıktının alt kısmında yer alan `/dev/sdb` USB belleğimizin diskteki adı. Şimdi belleğimizin adını öğrendiğimize göre onarma işlemini deneyebiliriz. Bunun için komut satırına `fsck -p /dev/sdb` komutunu giriyoruz.

```
root@taylan:~# fsck -p /dev/sdbfsck from util-linux 2.30.2fsck.fat 4.1 (2017-01-24)/dev/sdb1: 24 files, 1689384/1964160 c:
```

Ve USB diskimizdeki hatalı bloklar düzeltilerek işlemimiz başarılı bir şekilde gerçekleşmiş oldu.

df Komutu

`df` komutu bizlere disklerin; dosya sistemini, boyutunu, doluluk oranını, bağlandığı yeri(mount) gibi bilgileri kısa bir liste halinde verir. Komutumuz ile birlikte `-h` parametresini kullanarak hakkındaki geniş bilgi içeriği bizlere sunulur. Çıktılar aşağıdaki gibidir.

Komutumuz ile birlikte **-m** parametresini kullanırsak hakkında geniş bilgi içengi bizlere sunulur. Çıktılar aşağıdaki gibidir.

```
root@taylan:~# df -hDosyasistemi Boy Dolu Boş Kull% Bağlanılan yerudev 2,0G 0 2,0G 0% /devtmpfs 394M 13M 382M 4% /run/dev/:
```

Diskler hakkında genel bilgi edinmek için sık kullanılan bir komuttur. Ayrıca komutumuzla birlikte kullanacağımız **m** parametresi çıktıları MB cinsinden verirken, **k** parametresi ile kullanımlarda çıktıları KB cinsinden olacaktır. Bunlar haricinde daha fazla detay almak için man sayfasına bakabilirsiniz.

```
root@taylan:~# df -mDosyasistemi 1M-blok Dolu Boş Kull% Bağlanılan yerudev 1952 0 1952 0% /devtmpfs 394 13 382 4% /run/dev,
```

umount (Diski Ayırmak)

Diski ayırmak gibi bir başlık aklınızda diski bölümlenmek gibi bir anlam ifade etmiş olabilir. Ancak burdaki ayırmadan kasıt diskin bağlantısının sistemden koparılması. Yani şöyleki komutumuz daha önce muhtemelen kullanmış olduğunuz, diskin üzerine sağ tıklayarak "Diski Çıkar" işlemi ile aynı görevdedir. Hatta bu durumu hemen teyit etmek için USB diskimizi **umount** komutu ile sistemden ayıralım.

Bu işlem için öncelikle diskimiz hakkında genel bilgi sahibi olmak adına konsola **df -h** komutunu verelim.

```
root@taylan:~# df -hDosyasistemi Boy Dolu Boş Kull% Bağlanılan yerudev 2,0G 0 2,0G 0% /devtmpfs 394M 13M 382M 4% /run/dev/:
```

USB diskimin adının **/dev/sdb1** ve diskin sisteme bağlandığı konumun da **/media/root/SP** dizini olduğunu öğrenmiş oldum. Artık gerekli bilgileri öğrendiğimize göre diski sistemden çıkarmak için konsola **umount /dev/sdb1** şeklinde komut verebiliriz.

```
root@taylan:~# umount /dev/sdb1root@taylan:~# df -hDosyasistemi Boy Dolu Boş Kull% Bağlanılan yerudev 2,0G 0 2,0G 0% /devt:
```

Diski çıkarma işlemi sonrasında bu işlemi teyit etmek için birde `df -h` komutunu kullandık. Sonuçta diskimiz sistemden bağıntı koparmış oldu.

mount (Diski Bağlamak)

Sistemden ayırdığımızı disk tekrar bağlamak için `mount` komutunu kullanırız. Kullanım şekli `mount -t dosya_sistemi aygıt_adı bağlanacak_konum` şeklindedir.

Diski sisteme bağlamadan önce yapmamız gereken şey diskin bağlanacağı konumu oluşturmaktır. Bunun için ben masaüstüne `mkdir USB` komutu ile USB adında bir klasör oluşturdum.

```
root@taylan:~/Desktop# mkdir USBroot@taylan:~/Desktop# ls -ltotlam 4drwxr-xr-x 2 root root 4096 Oc 2 12:06 USB
```

USB nin açılacağı hedef klasör oluştuğuna göre artık USB diskimizi buraya bağlayabiliriz. Bunun için konsola `mount /dev/sdb1 /Desktop/USB` komutunu veriyorum ve daha sonra diskin durumunu `df -h` komutu ile sorguluyorum.

```
root@taylan:~/Desktop# mount /dev/sdb1 /Desktop/USBroot@taylan:~/Desktop# df -hDosyasistemi Boy Dolu Boş Kull% Bağlanılan !
```

Görüldüğü gibi diskim `Desktop/USB` konumuna bağlanmış bulunuyor.

Çalışma Seviyeleri(Runlevels)

Sistem açıldığında hangi çalışma seviyesindeyse o seviyeye göre belirlenmiş olan hizmetler başlatılır. İşte bu olaya da Runlevels(çalışma seviyeleri)deniyor. Linux sisteminde 7 farklı seviye bulunuyor.

Seviyeler ve ifade ettikleri aşağıdaki tabloda verilmiştir.

Yetki kalıplarının sayısal karşılıkları.

Runlevel	Çalışma Modu	İşlevler
0	Halt	Kapatma işleminin başladığı seviye.
1	Tek Kullanıcı	Ağ servisleri olmadan sistem bakımı için kullanılan seviye.
2	Ağ Desteği Olmadan Çok Kullanıcı	Ağ desteği olmadan normal kullanım seviyesi.
3	Ağ Destekli Çok Kullanıcı	Ağ destekli normal kullanım seviyesi.
4	Tanımsız	Kullanılmıyor ancak kullanıcı tarafından tanımlanabilir durumdaki seviye.
5	Grafiksel Kullanıcı Arayüzü	Grafiksel arayüzün çalıştığı seviye. Hemen her linux dağıtımında bu seviye vardır.
6	Yeniden Başlatma(Reboot)	Sistemin yeniden başlatıldığı seviye.

O anda hangi seviyede çalıştığımızı öğrenmek istersek komut satırımıza `runlevel` komutunu vermemiz yeterli.

```
root@taylan:~# runlevelN 5
```

Çıktıda çalışma seviyem 5 olarak çıktı. Bunun sebebi de daha önce de söylediğimiz gibi linux sistemi varsayılan olarak 5 seviyede başlatıyor. Eğer çalışma seviyesi değiştirmek istersek bunu `init çalışma_seviyesi` şeklinde yapabiliriz.

Bu şekilde sistemi yeniden başlatana kadar seçtiğimiz çalışma seviyesinde devam edebiliriz. Ancak dediğim gibi sistemi yeniden başlattığınızda sistem varsayılan olarak 5. seviyede başlayacaktır.

Ayrıca yeri gelmişken sistemi kapatmak istersek bunu hiç bir hizmetin çalışmadığını 0. runlevel ile `init 0` komutunu vererek yapabiliriz.

Ayrıca kapatma işlemi için;

`shutdown -h now` beklemeden direk sistemi kapatacaktır.

`shutdown -h now+5` 5 dakika sonra sistemi kapatacaktır.

`halt` sisteminizi doğrudan kapatacaktır(halt kavramının 0.seviyede hiç bir servisin çalışmadığı durumu temsil ettiğini söylemiştik).

Eğer sistemi yeniden başlatmak istersek `init 6` veya `reboot` komutunu kullanabiliriz.

service

Şimdi ise sistemdeki servisleri açıp-kapatmayı ve bu servislerin durumunu sorgulamayı görelim. Komutumuzun işlevlerinin kullanım kalıpları aşağıdakiler gibidir.

Örnek kullanımlar için aşağıdaki komutları kullanabiliriz.

Örnek kullanımları ssh servisi üzerinden göstereyim.

Servisimizin durumunu sorgulamak için komut satırımıza `service ssh status` yazıyoruz.

```
root@taylan:~# service ssh status • ssh.service - OpenBSD Secure Shell server Loaded: loaded (/lib/systemd/system/ssh.serv:
```

Komut satırı çıktı olarak bize ssh servisinin kapalı olduğunu bildirdi.

Şimdi servisimizi başlatmak üzere komut satırına `service ssh start` komutumuzu veriyoruz.

```
root@taylan:~# service ssh startroot@taylan:~# service ssh status • ssh.service - OpenBSD Secure Shell server Loaded: load:
```

Servisimizi başlattıktan sonra durumu kontrol etmek için tekrar `service ssh status` komutumuzu verdik ve görüldüğü gibi ssh servisimiz artık çalışır durumda.

Şimdi ise çalışan servisimizi kapatmak için komut satırına `service ssh stop` komutumuzu veriyoruz.

```
root@taylan:~# service ssh stoproot@taylan:~# service ssh status • ssh.service - OpenBSD Secure Shell server Loaded: loader
```

Çıktıda da görüldüğü gibi ssh servisimiz kapatılmış oldu. Bunu da `service ssh status` komutu ile teyit etmiş olduk.

Komutumuzun diğer bir kullanım şeklide de servisi yeniden başlatmak. Bunun için `service ssh restart` komutumuzu kullanıyoruz.

```
root@taylan:~# service ssh restartroot@taylan:~# service ssh status • ssh.service - OpenBSD Secure Shell server Loaded: lo:
```

Servisimizin durumunu `service ssh status` komutu ile kontrol ettik ve servisimizin yeniden başladığını görmüş olduk.

Son olarak ise sistemde bulunan tüm servislerin sıralanmasını sağlayan `service --status-all` komutunu görelim.

```
root@taylan:~# service --status-all [ -] anacron [ +] apache-htcacheclean [ -] apache2 [ -] mysql [ +] network-manager [ +]
```

Çıktıdaki servislerin sol tarafında yer alan `[-]` işareti servisin çalışmadığını, `[+]` işareti servisin çalıştığını ve `[?]` işareti ise servisin durumunun belirsiz olduğunu ifade ediyor.

Sembolik Link Ve Katı Link

Linux sistemlerinde sembolik ve katı olmak üzere iki çeşit bağlantı türü vardır. Bunları sırası ile açıklayacak olursak;

Sembolik link ile oluşturulmuş bağlantılar dosyaların kısayolu görevini görür ve görevi yalnızca ilgili dosyaya yönlendirme yapmaktır.

Katı link ile oluşturulmuş bağlantılarda ise dosyanın kopyasıdır. Orijinal dosya silinse bile katı link içeriği korumaya devam eder.

Bağlantı türlerinin kullanımlarına geçmeden önce ufak bir bilgi daha öğrenmeliyiz o da inode kavramı.

inode(düğüm)

Inode(düğüm), dosyanın sahibi, oluşturulma tarihi, boyutu, tipi, erişim hakları, en son erişim tarihi ve en son değişikliklerin yapıldığı tarih gibi birçok meta verileri içeren yapıdır. Yani biz herhangi bir dosya oluşturduğumuzda disk üzerinde 1 inode yer kaplamaktadır. Bununla ilgili olarak inode tablosunu görmek için konsola `df -i` komutumuzu verelim.

```
root@taylan:~# df -iDosyasistemi Dosyaindeksi Dolu Boş IKull% Bağlanılan yerudev 499901 359 499542 1% /devtmpfs 503783 870
```

Hemen çıktıları ele alarak açıklamamıza devam edelim.

Örneğin `/dev/sda1` disk alanını ele alırsak toplam `6152192` adet inode numarası olabilir yani sınırı bu kadar.

Kullanılan `340890` inode numarasından toplam kullanılabilir olanı çıkarırsak ($6152192 - 340890 = 5811302$) geriye kullanılabilir `5811302` inode numarası kalmış oluyor.

Bunun anlamı her bir dosya 1 inode yer kapladığı için `/dev/sda1` dosya sisteminde `5811302` adet daha dosya oluşturulabilecek alan mevcut.

Her bir inode(düğüm) numarasının benzersiz olduğunu söylemiştik. Bu durumu teyit etmek için bulunduğumuz konumdaki dosyaların

inode numaralarını görebilmek adına komut satırımıza `ls -li` komutumuzu veriyoruz.

```
root@taylan:~# ls -li toplam 24843276863 drwxr-xr-x 5 root root 4096 Ara 12 04:50 Desktop3276867 drwxr-xr-x 2 root root 4096
```

Çıktıda yer alan soldaki numaralar ilgili dosyanın inode numarasını temsil ediyor.

Konumuza sembolik link oluşturma ile devam edelim.

Sembolik Link

Sembolik link oluşturmak için `ln -s` komutu kullanılır. Komutumuzun kullanım örneği ile devam edelim.

Örnek bir dosya oluşturarak sembolik link komutu yardımı ile dosyamıza kısayol oluşturalım.

Music dosya konumundayken `touch` komutu yardımı ile klasik isimli bir dosya oluşturduk.

```
root@taylan:~/Music# touch klasikroot@taylan:~/Music# ls klasik
```

Oluşturduğumuz klasik isimli dosyanın kısayolunu, komutumuzu `ln -s dosya_adı kısayol_adı` şeklinde kullanarak oluşturduk.

```
root@taylan:~/Music# ln -s klasik klasik_kısayolroot@taylan:~/Music# ls -li3277136 klasik 3280213 klasik_kısayol
```

Son olarak `ls -li` komutumuz vererek orijinal dosyanın ve sembolik link yardımı ile oluşturulan kısayol dosyasının inode değerlerini karşılaştırdık. Çıktıda da görüldüğü üzere değerler farklı olarak karşımıza çıkmış oldu.

Ayrıca dosyaları `file` isimli bir komut yardımı ile inceleyerek de bilgi alabiliriz.

```
root@taylan:~/Music# file klasikklassik: emptyroot@taylan:~/Music# file klasik_kısayolklasik_kısayol: symbolic link to klas:
```

Sembolik link komutu ile oluşturduğumuz dosyanın kısayol olduğunu çıktıda görebiliyoruz. Ancak son olarak kısayol dosyamızın çalışırlığını da test edelim.

Bunun için ilk başta dosyamızın içeriğini okuyalım daha sonra orijinal dosyamıza yazı ekleyelim ve kısayoldaki değişimleri gözlemleyelim.

Halihazırda orijinal dosyamın içerisine "Dosya İçeriğinin İlk Sayfası" şeklinde bir yazı bulunuyor biz orijinal dosyada yer alan bu yazıyı silip yeni bir ifade yazdıktan sonra kısayoldaki değişimi gözlemleyeceğiz.

```
root@taylan:~/Music# cat klasikDosya İçeriğinin İlk Sayfasıroot@taylan:~/Music# cat klasik_kısayolDosya İçeriğinin İlk Say-
```

Eğer orijinal dosyayı silersek kısayol dosyası da açılmayacak ve bizlere hata mesajı bildirilecektir.

```
root@taylan:~/Music# ls -ltoplam 4-rw-r--r-- 1 root root 35 Ara 14 07:27 klasiklrwxrwxrwx 1 root root 6 Ara 14 07:14 klasil
```

Görüldüğü gibi kısayol dosyası orijinal kaynağı silindiği için okunamadı.

Ayrıca ilk `ls -l` çıktısında kısayol dosyasının sol tarafında yer alan `lrwxrwxrwx` ifadesindeki `l` harfi, dosyanın kısayol olduğunu ve `->` işareti ise hangi dosyanın kısayolu olduğunu işaret ediyor.

Katı Link

Sıra geldi katı link bağlantısının kullanımına. Katı link bağlantısı için `ln` komutu kullanılıyor. Örnek üzerinden ilerleyelim.

Örnek olması açısından Music klasörü altında rap adında bir dosya oluşturuyorum. Daha sonra oluşturduğum rap isimli klasörün katı linkini `ln rap rap_kısayol` komutu ile oluşturuyorum.

```
root@taylan:~/Music# touch raproot@taylan:~/Music# lsraproot@taylan:~/Music# ln rap rap_katıroot@taylan:~/Music# ls -litop:
```

Çıktıda dikkat edilmesi gereken nokta orijinal rap dosyası ile katı bağlantı rap dosyasının inode değerlerinin aynı olmasıdır. Bu da demek oluyor ki her iki dosya da her yönüyle birbirlerinin aynı durumda.

Ayrıca orijinal ya da katı link ile oluşturulmuş dosyaların birinde yapacağımız değişiklik tıpkı sembolik linkte olduğu gibi diğer dosyada da geçerli olacaktır. Yani bu orijinal dosya ile katı link dosyası halihazırda bağlantılı dosyalar.

Hemen daha önce yaptığımız gibi `find` komutu ile orijinal dosya ve link ile oluşturulmuş dosyalarımızı karşılaştıralım.

```
root@taylan:~/Music# find rapraproot@taylan:~/Music# find rap_katirap_katı
```

Komutumuzun sonucunda her ikisinde ayrı ayrı dosyalar olduğunu görmüş olduk. Bu da demek oluyor ki biz eğer orijinal dosyayı silsek sembolik linkte olduğu gibi link ile oluşturulmuş dosya kullanılmaz hale gelmeyecektir. Bunu da hemen test edelim.

```
root@taylan:~/Music# lsrap rap_katiroot@taylan:~/Music# rm raproot@taylan:~/Music# cat rap_katıDosya İçeriğinin İlk Sayfas:
```

Gördüğümüz gibi orijinal dosyayı silmeme rağmen katı link ile oluşturmuş olduğum dosya hala okunabilir durumda.

Kurma-Kaldırma-Güncelleme İşlemleri

Linux kullanacaksak mutlaka bilmemiz gerekenler arasında sistemi güncelleme, program kurma ve kaldırma gibi işlemleri yerine getirmek var. Bu işlemleri nasıl yerine getirebileceğimizi bu konunun devamında öğrenecez. İlk olarak sistemi güncelleme işlemi yapalım.

Sistemi Güncelleme

Linux sistemleri kullanıcının ihtiyacı olduğunda programa kolayca ulaşabilmesini sağlayacak program paketlerini içinde bulunduran kendi paket depolarına(repository) sahiptirler. Farklı linux dağıtımları için bu paketler de farklılık gösterebiliyor. Bu yüzden farklı linux dağıtımlarının da kendi paketleri üzerinde işlem yapabilmek için farklı komutları vardır.

Dağıtımlar ve kullanılan paketlere göre komutları aşağıdaki tabloda verilmiştir.

Dağıtım	Paketler	Komutlar
Debian	.deb	apt , apt-cache , apt-get , dpkg
Ubuntu	.deb	apt , apt-cache , apt-get , dpkg
CentOs	.rpm	yum
Fedora	.rpm	dnf
FreeBSD	.txz	make , pkg

Debian paketlerinin `.deb` uzantılı olduğunu yukarıdaki tabloda gördük. Kullandığımız dağıtım olan Kali'de debian tabanlı olduğu için anlatıma bu doğrultuda devam edeceğiz.

```
apt-get update
```

Paket listesini(/etc/apt/sources.list) günceller.

```
apt-get upgrade
```

Kurulu olan paketleri /etc/apt/sources.list dosyasındaki paket listesine bakarak en son versiyonlarına günceller.

```
apt-get dist-upgrade
```

Gerekli gördüğü paketleri siler veya günceller.

```
apt-get clean
```

Kurulu olan tüm paketleri siler.

```
-y
```

`-y` parametresinin görevi çıkacak olan onay sorularına evet(yes) cevabını otomatik olarak vermektir. Bu sayede gerekli işlemler de otomatik olarak zaman kaybetmeden yapılmış olur.

Yani bir bütün olarak eğer sistemimizi güncellemek istersek ilk başta `/etc/apt/sources.list` konumunda yer alan `sources.list` dosyasına kullandığımız versiyona uygun depoları eklemeliyiz.

Bunun için [buradaki kaynaktan](#) kullandığınız versiyona uygun olan repository kopyalayarak `sources.list` dosyasına eklememiz gerekiyor. Ben kali 2016.1 sonrası versiyonunu kullandığım için aşağıdaki repoları ekliyorum.

```
deb http://http.kali.org/kali kali-rolling main contrib non-free# For source package access, uncomment the following line#
```

Ancak dediğim gibi sizler kullandığınız versiyona uygun olan repoları seçmelisiniz.

Repoları `nano -w /etc/apt/sources.list` komutu ile açarak `sources.list` dosyasına ekledim.

Sıra gelidi güncelleme işlemine, bunun için `apt-get update && apt-get upgrade -y && apt-get dist-upgrade -y` komut

bütününü kullanabiliriz.

Güncelleme işlemi boyunca kullandığımız `-y` parametresi sayesinde herhangi bir soru sorulmadan bütün işlemler otomatik olarak tamamlanacak ve güncelleme işlemi tamamlanacaktır.

Program Kurmak

Linux'ta program kurmak için birden fazla yöntem bulunuyor. Bunlardan bir tanesi kullandığımız dağıtıma uygun programı, **paket yönetim sistemi** ile kurmaktır. Diğer bir yol, programı **kaynak koddan derleyerek** kurmaktır. Diğer seçenek ise dağıtımın kullandığı **depolardan(repository)** otomatik kurulum yapmaktır.

Depodan Kurulum

Depoda bulunan programların kurulumlarını yaparken `apt-get install program_adi` komut bütünü kullanılıyor.

Örnek olması açısından filezilla isimli bir programın depodan kurulumunu yapmak için konsola `apt-get install filezilla` komutunu girdim.

```
root@taylan:~# apt-get install filezillaPaket listeleri okunuyor... BittiBağımlılık ağacı oluşturuluyor Durum bilgisi okunuyor
```

Evet onayını vermem durumunda program otomatik olarak kurulacaktır.

Paket Yönetim Sistemi İle Kurulum

Bu işlem için kullandığımız dağıtıma uygun derleyiciyi kullanmalıyız. Daha önce Kali'nin `.deb` uzantılı paketleme sistemi olduğunu öğrenmiştik. Bu yüzden biz `.deb` uzantılı kurulum paketlerini açmak için `dpkg` komutunu kullanıyoruz. Sanırım kodun kısaltmasının nereden geldiğini bilirsek daha kolay akılda kalabilir. Kodun kısaltması "debian package(debian paketi)" kısaltmasından gelmektedir. Ayrıca `dpkg` komutunu kullanmada yardımcı bir paket yöneticisi programı(synaptic) kullanarak da kurulum işlemlerini yerine getirebiliriz. Konumuza ilk olarak `dpkg` komutu ve kullanımı ile devam edelim.

Şimdi komutun kullanımına geçelim.

Örnek olması açısından ben master pdf adında bir programın kurulumunu ele aldım. Bunun için öncelikle programın `.deb` uzantılı dosyasını sitesinden indirdim.

Şimdi programı kurmak için `dpkg -i paket_adi.deb` şeklinde komutumu giriyorum.

```
root@taylan:~/Desktop# dpkg -i master-pdf-editor-4.3.61-qt5.amd64.deb(Veritabanı okunuyor ... 330896 dosya veya dizin kuruluyor)
```

Ve program kurulmuş oldu.

Kurduğumuz programı kaldırmak istersek komutumuzu `dpkg -r program_adı` şeklinde yani `-r` parametresini ekleyerek kullanıyoruz.

```
root@taylan:~/Desktop# dpkg -r master-pdf-editor(Veritabanı okunuyor ... 330896 dosya veya dizin kurulu durumda.)Kaldırılıyor
```

Program otomatik olarak kaldırılmış oldu. Ancak `/etc` dizini altındaki konfigürasyon dosyaları silinmedi. Eğer bu dosyaları da kaldırmak istersek `-P` parametresini kullanabiliriz.

```
root@taylan:~/Desktop# dpkg -P master-pdf-editor(Veritabanı okunuyor ... 330733 dosya veya dizin kurulu durumda.)Yapılandırılıyor
```

Bu sayede programa ait konfigürasyon dosyaları da tamamen kaldırılmış oldu.

Konfigürasyondan bahsetmişken aklınızda bulunsun eğer yüklediğimiz paketin konfigürasyon ayarlarını tekrar yapılandırmamız gerekirse `dpkg-reconfigure paket_adı` şeklinde komutumuzu kullanabiliriz.

Kurulu olan tüm paketler hakkında bilgi almak için `-l` parametresi kullanılabilir.(Çıktı uzun olduğundan üç nokta ile kısa kesilmiştir.)

```
...rc tracker-gui 1.10.5-1 amd64 metadata database, indexer and seii tracker-miner- 2.0.3-1 amd64 metadata database, index
```

Çıktıda yer alan paketlerin sol tarafındaki ifadelerin anlamı:

ii :paket normal olarak sisteme yüklendi.

rc :paket yüklendikten sonra silindi ancak konfigürasyon dosyaları halen mevcut.

pn :paket konfigürasyon dosyaları ile birlikte sistemden kaldırıldı.

Kurulu paketin durumunu öğrenmek için `-s` parametresini kullanabiliriz. Ben örnek olması açısından leafpad programı hakkında bilgi almak için komut satırına `dpkg -s leafpad` komutunu giriyorum.


```
root@taylan:~# dpkg -s leafpadPackage: leafpadStatus: install ok installedPriority: optionalSection: editorsInstalled-Size
```

Kurulu paketin içeriğini öğrenmek istersek **-L** parametresini kullanırız.

```
root@taylan:~# dpkg -L leafpad /usr/share/man/man1/leafpad.1.gz/usr/share/
```

Eğer indirmiş olduğumuz **.deb** uzantılı dosyanın içeriğini henüz kurmadan görmek istersek **-c** parametresini kullanabiliriz.

```
root@taylan:~/Desktop# dpkg -c master-pdf-editor-4.3.61_qt5.amd64.deb drwxrwxr-x root/root 0 2016-08-24 14:03 ./drwxrwxr-x
```

Sistemde kurulmuş ve kaldırılmış tüm paketleri görmek istersek **dpkg --get-selections** komutunu kullanabiliriz.

```
root@taylan:~# dpkg --get-selections0traceinstallaaptinstallacccheckinstallaccountsserviceinstallace-voipinstallaclinstall:
```

Bu paket listesini daha sonra kullanmak üzere bir dosya içerisine yedeklemek istersek komutumuzu **dpkg --get-selections >dosya_adı.txt** şeklinde kullanmamız yeterli olacaktır.

```
root@taylan:~/Desktop# dpkg --get-selections >paketler.txtroot@taylan:~/Desktop# less paketler.txt 0trace installaapt inst:
```

Yedeklediğimiz bu program listeleri sayesinde başka bir sisteme aynı paketleri tek seferde yüklememiz mümkün. Bunun için komutlar aşağıdaki şekildedir.

```
root@taylan:~/ dpkg --set-selections > yedek.txtroot@taylan:~/ apt-get deselect-upgrade
```

Şimdi sıra geldi ikinci yol olan yardımcı paket yöneticisi programı aracılığı ile program kurma-kaldırma-güncelleme işlemlerin yapmaya. Bunun için ilk olarak paket yönetim programı olan synaptic programını kurmamız gerek. Komut satırımıza `apt-get install synaptic` yazarak yardımcı programı indiriyoruz.

```
root@taylan:~# apt-get install synapticPaket listeleri okunuyor... BittiBağımlılık ağacı oluşturuluyor Durum bilgisi okunuyor
```

Evet "e" diyerek kurulumu onay veriyoruz ve programımız kurulmuş oluyor. Programı açmak için komut satırına `synaptic` komutunu vermemiz yeterli. Bu program sayesinde program kurup kaldırabilir ve programlar için diğer gerekli işlemleri yerine getirebiliriz. Programın kullanımı oldukça kolay olduğundan keşfini sizlere bırakarak anlatıma devam ediyorum.

Kaynak Koddan Derleyerek Kurulum

Linux'a uyumlu sürümü bulunan açık kaynak kodlu yazılımların kaynak kod paketleri bu yazılımların internet sitelerinde, genellikle `tar.gz` biçiminde arşivlenmiş olarak indirilmeye sunulurlar. Bu arşivlenmiş olan kaynak kod paketi daha önce gördüğümüz `.deb` uzantılı paketlerin aksine ham şekilde yani derlenmemiş kurulumu hazır olmayan şekilde gelirler. Bu derleme işlemini bizim yaparak kurulumu gerçekleştirmemiz gerekir. Ancak her kurulum aynı olmayabilir, `tar.gz` veya `tar.bz2` benzeri uzantılı paketlerde genellikle programın nasıl kurulacağına dair install, readme, configure ve benzeri isimlerde yönergeler bulunur. Bu dosyalar okunarak yükleme işlemi gerçekleştirilmelidir. Ancak biz şimdi burada genel kurulum hakkında bilgi edinelim. Kaynak koddan derleyerek kurulum işleme genel olara aşağıdaki şekildedir:

İndirdiğimiz program arşivini klasöre çıkartırız.

Programı derlemeden önce gerekli kütüphaneler ve bağımlılıkları kontrol ediyoruz.

Eksik çıkarsa bunları kurmamız gerekicek.

Make aşaması için derleme yapacak paketleri `apt-get install build-essential` komutu ile kuruyoruz.

`make` diyerek programımızı derlemiş oluyoruz.

Son olarak `make install` komutu ile programı sistemimize kuruyoruz.

Joker Karakterler(wildcards)

Bu kavram sizlere kesinlikle yabancı değil daha önceki kısımlarda defaatle kullanmış ve joker(wildcard) karakterlerinin az da olsa işlevini görmüştük. Bu kısımda da bu konu hakkında bilgi sahibi olacağız.

joker (wildcard)

Linux kullanımında işimizi konsoldan yürüteceğimiz zaman bir komutun tek seferde birden fazla nesneyi etkilemesini yani kapsamasını isteyebiliriz. Örneğin bir dizindeki dosyaların tamamını silmek istiyoruz bu iş için aşağıdaki gibi tüm dosya adlarını komut satırına yazmak çok zahmetli ve gereksiz olacaktır.

```
root@taylan:~/ rm dosya1 dosya2 dosya3 dosya4 dosya5 dosya6 dosya7
```

Yukarıdaki kullanımın yerine joker karakter(wildcard)desteğini kullanmak bizlere çok fazla avantaj sağlar. Ayrıca gördüğünüzde şaşırmayın, joker karakterler(wildcards) "globbing" olarak da adlandırılmaktadır.

Bahsi geçen joker karakterler ve kullanım alanları aşağıdaki gibidir;

* :Anlamı * olan yere herhangi bir şey gelebileceğidir.

Örneğin çıktılar dosya konumunda yer alan bütün dosya ve belgeleri tek seferde silmek için komutmu `rm -r *` şeklinde kullanabilirim.

```
root@taylan:~/Desktop/çıkıtlar# ls -ltotlam 32----- 1 root root 0 Ara 4 15:45 dosya23.txt-rwx-w-r-- 1 root root 378 Aı
```

Ayrıca kafanıza takılmış olabilir burada `rm` komutu ile birlikte kullandığım `-r` parametresi izinleri silmek için gerekiyordu. Aslında bu bilgiyi önceki kısımlarda öğrenmiştik yinede bu ufak hatırlatma ile tekrar etmiş olduk. Ayrıca bu yıldız (asterix) * işaretinin farklı kullanım şekilleri de var. Farklı kullanımların açıklaması ile devam edelim.

Herhangi bir komutu, örneğin listeleme işlevinde olan `ls` komutunu `ls dosya*` şeklinde kullanırsak komutumuz yıldız (asterix) * işaretinden önce yazmış olduğumuz dosya ismi ile başlayanları kapsar.

```
root@taylan:~/Desktop# ls -ltotlam 3356drwxr-xr-x 2 root root 4096 Ara 15 02:38 çıkıtlar-rw-r--r-- 1 root root 0 Ara 4 15:4
```

Aynı şekilde yıldız (asterix) * işaretiinden sonra bir ifade belirtirsek de komutumuz o ifade ile bitenleri kapsayacak şekilde çalışır.

```
root@taylan:~/Desktop# ls -ltoplam 3356drwxr-xr-x 2 root root 4096 Ara 15 02:38 çıktılar-rw-r--r-- 1 root root 0 Ara 4 15:4
```

? :Herhangi bir tek karakterle eşleşir

Karakterin kullanımına örnek olarak. Diyelim ki dizin içerisinde hem `doysa_adı` hemde `dosya-adı` şeklinde isimlere sahip dosyalarınız var. Yani başlangıç ve bitiş isimleri aynı ancak aradaki işaretler farklı. İşte böyle bir durumda hem `_` işaretini hemde `-` işaretini karşılayacak olan soru işareti `?` joker karakterini kullanabiliriz.

```
root@taylan:~/Desktop/belgeler# ls -l-rw-r--r-- 1 root root 0 Ara 15 06:09 bilgi_dosyaları-rw-r--r-- 1 root root 0 Ara 15 0
```

Çıktılarda da görüldüğü gibi `?` karakteri dosya isimlerinin arasında yer alan `-` ve `_` işaretini de karşılayarak `ls -l dosya?adı` şeklindeki komut ile çıktıya her ikisini de bastı.

[] : `?` karakterine benzer olmakla birlikte daha çok hedefe odaklı çalışır.

[] karakterinin kullanımı, iki köşeli parantez arasına ulaşmak istediğiniz hedefteki ayırıcı karakterli yazmak üzerinedir.

Örnek olaması açısından ben masaüstümde yer alan dosya isimli belgelerden sadece sonunda 2 veya 3 olanları kapsayacak bir komut olması için konsola `ls -l [23]` komutunu verdim.

```
root@taylan:~/Desktop# ls -ltoplam 24-rw-r--r-- 1 root root 0 Ara 4 15:45 dosya23.txt-rw-r--r-- 1 root root 378 Ara 4 15:5
```

Bir örnek daha verelim.

`[Dd]osya[Aa]dı` şeklinde bir belirtme; DosyaAdı, Dosyaadı, dosyaAdı, dosyaadı şeklindeki bütün isimleri kapsayacaktır. Bu sayede ilgili dosya için küçük büyük harf kombinasyonu yakalanmış olur.

Ayrıca kullanım şekillerine çok fazla örnek verilebilir ancak burada birkaç örnek daha vererek keşfi size bırakıyorum.

Burada belirtilen x y z temsili değerleri ifade etmektedir !

[0-9] :0'dan 9'a kadar olan rakamları kapsar.

```
root@taylan:~/Desktop# ls -ltoplam 24-rw-r--r-- 1 root root 0 Ara 15 08:47 1.belge.txt-rw-r--r-- 1 root root 0 Ara 15 08:47
```

[x,y,z] :belirliten değerlerdeb eşleşenleri basar.

```
root@taylan:~/Desktop# ls -l [0-9]*-rw-r--r-- 1 root root 0 Ara 15 08:47 1.belge.txt-rw-r--r-- 1 root root 0 Ara 15 08:47 :
```

[x-z] :x ile z değerleri arasındaki karakterlerle eşleşir.

```
root@taylan:~/Desktop# ls -l [1-4]*-rw-r--r-- 1 root root 0 Ara 15 08:47 1.belge.txt-rw-r--r-- 1 root root 0 Ara 15 08:47 :
```

[xyz] :çıkış yapar.

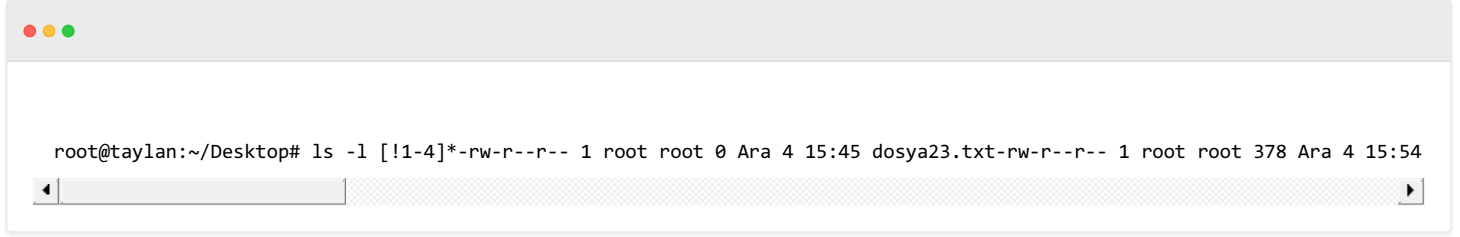
```
root@taylan:~/Desktop# ls -l [123]*-rw-r--r-- 1 root root 0 Ara 15 08:47 1.belge.txt-rw-r--r-- 1 root root 0 Ara 15 08:47 :
```

[!xyz] :Belirtilen karakterlerin dışındakileri ile eşlenir.

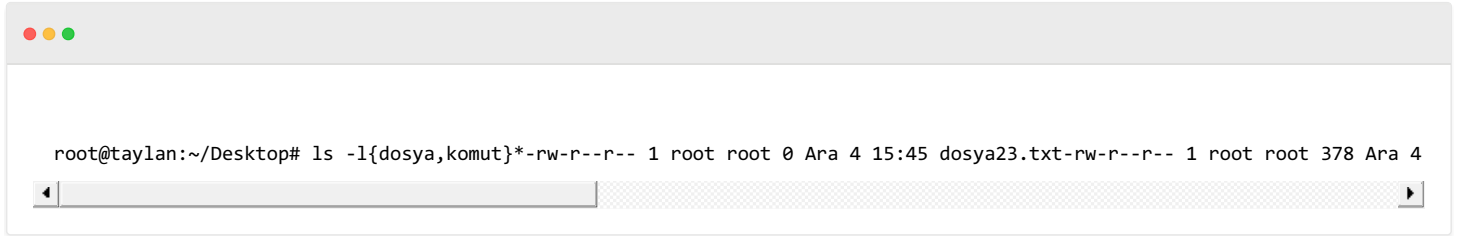
```
root@taylan:~/Desktop# ls -l [!12]*-rw-r--r-- 1 root root 0 Ara 15 08:47 3.belge.txt-rw-r--r-- 1 root root 0 Ara 15 08:47 ,
```



[!x-z] :Verilen x ile z değeri arasındakiler haricindeki karakterler ile eşlenir.



[kelime1-kelime2] :Belirliten kelimeler ile eşleşir.



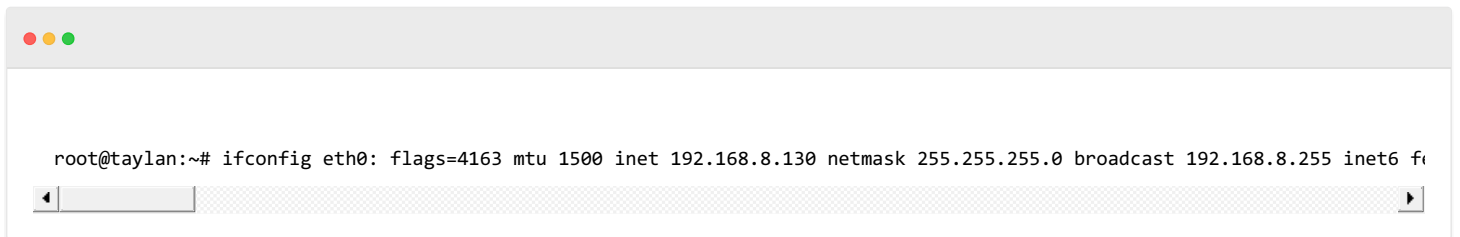
Ağ(Network) Komutları

Ağ ve komutları başlı başına bir kitap konusu o yüzden burada sıkça veya gerekli durumda işimize yarayacak belli başlı komutları ele alıcaz. Eğer detaylı bilgi edinmek istiyorsanız internet aracılığı ile network(ağ) hakkında bir çok güncel kaynağa ulaşabilirsiniz. Lafı daha fazla uzatmadan anlatıma **ifconfig** komutu ile başlayalım.

ifconfig

Sistemde bulunan ağ bağlantı ve IP yapılandırması ayarlarını inceleyip değiştirmemize olanak tanır.

Ağ bağlantı kartlarını listelemek için **ifconfig** komutu kullanılır.(Bu çıktıda ip, mac ve broadcats adresi gibi bilgiler yer alıyor.)



Çıktıda yer alan `eth0` ethernet kartımızı ifade ediyor. Başka ethernet kartları olması durumunda diğer kartlar da `eth1` .. `eth2` şeklinde belirtilir.

`lo` bilgisayarın kendisini yani localhost'u ifade ediyor.

`wlan0` ise kablosuz ağ kartını ifade eder. Yine başka kartlar olması durumunda `wlan1` .. `wlan2` şeklide belirtilir.

Ayrıca bütün kart bilgilerini listelemek yerine teker teker de listelemek de mümkün bunun için komutumuzu `ifconfig kart_adı` şeklinde girmemiz yeterli.

Örneğin yalnızca kablosuz kart bilgilerini listelemek istersem `ifconfig wlan0` komutunu vermem yeterli olacaktır.

```
root@taylan:~# ifconfig wlan0wlan0: flags=4163 mtu 1500 inet 192.168.1.9 netmask 255.255.255.0 broadcast 192.168.1.255 ine'
```

Yukarıdaki çıktıda yer alan ayarlarda değişiklik yapmamız mümkün. Örneğin kablosuz ağ bağlantısının yerel ip adresini değiştirmek istersek komutumuzu `ifconfig wlan0 yeni_ip_adresi` şeklinde kullanabiliriz.

Ben kablosuz bağlantımın 192.168.1.9 olan adresi 192.168.1.10 olarak değiştirmek istiyorum bunun için `ifconfig wlan0 192.168.1.10` komutunu vermem yeterli.

Gelin bu durumu çıktıları karşılaştırarak test edelim.

```
root@taylan:~# ifconfig wlan0wlan0: flags=4163 mtu 1500 inet 192.168.1.9 netmask 255.255.255.0 broadcast 192.168.1.255 ine'
```

Mevcut durumu kontrol ettik.

```
root@taylan:~# ifconfig wlan0 192.168.1.10
```

Yerel ip adresini değiştirdik.

```
root@taylan:~# ifconfig wlan0wlan0: flags=4163 mtu 1500 inet 192.168.1.10 netmask 255.255.255.0 broadcast 192.168.1.255 in
```

Ve son duruma bakarak ip adresimizin istediğimiz şekilde değiştiğini görmüş olduk. Aynı şekilde ağ maskesini(netmask) ve broadcast adreslerini dilediğimiz gibi düzenleyebiliriz. Hatta bu işlemi hepsi birlikte olacak şekilde bile yapabiliriz. Bunun için komutumuzu `ifconfig wlan0 yeni_ip_adresi netmask yeni_ağ_maskesi broadcast yeni_broadcast_adresi` şeklinde yapmamız mümkün.

Bu duruma bir örnek ile açıklayalım.

```
root@taylan:~# ifconfig wlan0wlan0: flags=4163 mtu 1500 inet 192.168.1.10 netmask 255.255.255.0 broadcast 192.168.1.255 in
```

Çıktıları incelediğimizde istediğimiz doğrultuda değişikliklerin yapılmış olduğunu gördük.

`ifconfig` komutu kullanımı ile ilgili son olarak mevcut kartları açma ve kapatma işlemi görelim.

Örnek olarak kablosuz kartı ele alalım:

Kablosuz kartı kapatmak istersek `ifconfig wlan0 down` komutunu kullanırız.

Kapalı kablosuz kartı açmak istersek ise `ifconfig wlan0 up` komutunu kullanırız.

Ayrıca diğer kartlar için de aynı şekilde komutumuzu `ifconfig kart_adı down` ve `ifconfig kart_adı up` şeklinde kullanabiliriz.

ping

Hedef ile bizim sistemimiz arasında iletişimin sağlanıp sağlanmadığını kontrol etmeye yarar. Sonuç olarak hedef sunucunun çalışıp çalışmadığını veya aktarım hızının ne kadar hızlı olduğunu öğrenmemizi sağlar. Bir tür kontrol mekanizması da diyebiliriz. Komutun kullanımı `ping hedef_adresi` şeklindedir.

Örneğin biz www.google.com ile aramızdaki iletişimin hızını sorgulayalım. Bunun için komut satırına `ping www.google.com` yazıyorum. Ancak burda önemli bir nokta var o da bu işlemin biz `Ctrl + C` tuş kombinasyonu ile durdurana kadar devam edecek olamasıdır. Eğer direk olarak sınırlama getirmek isterseniz komutunuzu `ping -c 2 www.google.com` şeklinde kullanabilirsiniz. Bu sayede www.google.com adresine yalnızca 2 sorgu paketi gönderilir. Elbette buradaki sayı sizin belirlemeniz ile değişebilir.

```
root@taylan:~# ping -c 2 www.google.comPING www.google.com (172.217.22.4) 56(84) bytes of data.64 bytes from www.google.co
```


route

Sistemimizde bulunan yönlendirmeleri görmek için kullanabiliriz. En temel kullanımı `route` şeklindedir.

```
root@taylan:~# route Kernel IP routing tableDestination Gateway Genmask Flags Metric Ref Use Ifacedefault _gateway 0.0.0.0
```

Burada yer alan gateway(ağ geçidi adresi) yerel ağdan internet ağına geçerken kullanılan yönlendirme adresidir.

traceroute

Bir önceki kısımda `route` komutu ile gördüğümüz yerel ağda geçerli olan yönlendirme takibini, belirli bir hedef adrese yapabilmemize olanak sağlayan komut `traceroute` komutudur. Yani komutumuz belirli bir hedefe gönderilen paketin hangi host'lerden geçtiğini bizlere gösterir. Bir nevi izlediği yolu yani adımlarını takip etmemizi sağlar. Komutun kullanımı `traceroute hedef_adresi` şeklindedir. Bu sefer hedef adres olarak `www.offensive-security.com` adresini örnek verelim. Bunun için konsola `traceroute www.offensive-security.com` yazıyorum.

```
root@taylan:~# traceroute www.offensive-security.comtraceroute to www.offensive-security.com (192.124.249.5), 30 hops max,
```

Çıktı ile birlikte adresin yönlendirme rotasını görmüş olduk.

whois

Whois kavramını bilmeyenler için whois, genel olarak domain bilgilerini içeren bir mekanizmadır. Yani whois domain ne zaman kurulmuş, ne zamana kadar geçerli, kimin üzerine ve bunun gibi diğer tüm bilgileri tutar. Bizler de bu bilgileri konsol ekranından `whois hedef_adresi` şeklinde kullandığımız komut bütünü ile sorgulayabiliriz.

Whois sorgusuna örnek olarak yine offensive-security adresini hedef alalım.

```
root@taylan:~# whois offensive-security.com Domain Name: OFFENSIVE-SECURITY.COM Registry Domain ID: 606288052_DOMAIN_COM-VI
```

host

Hedef adres hakkında bilgi almaızı sağlar. Bu komutun alabildiği farklı parametrele bulunmaktadır ancak ben bu kısımda bunlara değinmeden yalnızca temel kullanımına örnek veriyorum. Kullanımı `host adres_adı` şeklindedir.

```
root@taylan:~# host offensive-security.comoffensive-security.com has address 192.124.249.5offensive-security.com mail is h:
```

Bu komut hakkında ufak bir araştırma ile çok fazla Türkçe de dahil olmak üzere kaynağa ulaşabilirsiniz.

dig

`dig` (domain information groper/domain bilgi çukuru) DNS kayıtlarına bakmak için kullanımı oldukça kolay olduğundan yaygın olarak kullanılmaktadır.

Bu komutumuz da parametreler alabilmektedir ancak ben burada sizlere yine temel işevinden bahsederek geriye kalan parametrelili kullanımların araştırmasını sizlere bırakıyorum. DNS sorgulaması yapmak istediğimiz adresi konsoldan `dig hedef_adresi` şeklinde belirtiyoruz.

Örnek olması açısından ben tekrar offensive-security adresini hedef alıyorum.

```
root@taylan:~# dig offensive-security.com; <>> DiG 9.11.2-5-Debian <>> offensive-security.com;; global options: +cmd;; G
```

arp

IP-MAC Adresi eşleştirmelerinin tutulduğu tablolardır. Kullanımı `arp` şeklindedir.

```
root@taylan:/# arpAddress HWtype HWaddress Flags Mask IfaceZyxel.Home ether c8:3a:35:7d:46:18 C wlan0192.168.18.254 ether (
```

tcpdump

Sistemizin yaptığı bağlantıları ve sistemimize yapılan bağlantıları anlık olarak görüntülememize olanak sağlar. Kullanımı `tcpdump` şeklindedir.

```
root@taylan:/# tcpdumptcpdump: verbose output suppressed, use -v or -vv for full protocol decodelistening on eth0, link-ty
```

Ayrıca adres çözümlemesi yapmadan direk olarak bağlantıları takip etmek istersek `tcpdump -n` komutunu kullanabiliriz.

```
root@taylan:/# tcpdump -ntcpdump: verbose output suppressed, use -v or -vv for full protocol decodelistening on eth0, link-
```

DNS Ayarları

Komut satırından DNS ayarlarımızı değiştirmek istersek DNS bilgilerinin tutulduğu `/etc/resolv.conf/` dosyasında değişiklik yapmamız gerekiyor. İşlemleri adım adım açıklayarak ilerletiyorum.

İlk olarak DNS ayarlarının bulunduğu dosya içeriğine göz atıyorum. Çünkü daha sonra değişiklik yaptığımızda ilk hali ile kıyaslamamız gerekecek. Bu işlemi `cat` komutu yardımı ile gerçekleştirecez.

```
root@taylan:~# cat /etc/resolv.confnameserver 8.8.8.8nameserver 8.8.4.4
```

Şimdi eski DNS(nameserver) yerine bizim eklemek istediğimiz adresleri `echo` komutu yardımı ile girelim.

```
root@taylan:/etc# echo "nameserver 8.8.8.8" > resolv.confroot@taylan:/etc# echo "nameserver 8.8.4.4" >> resolv.confroot@ta
```

Komutları kısaca açıklayacak olursak ilk olarak `echo "nameserver 8.8.8.8" > resolv.conf` komut bütününde `>` karakteri ile birlikte dosya içerisinde var olan ifadeleri sildik ve dosya içerisine `nameserver 8.8.8.8` ifadesini ekledik.

Daha sonra `echo "nameserver 8.8.4.4" >> resolv.conf` komut bütününü ile de daha önce eklediğimiz ifadeye ek olarak diğer bir DNS adresi olan `8.8.4.4` adresini `>>` karakteri yardımı ile ekledik.

Son olarak da eklediğimiz yeni DNS adreslerinin belgeye eklenme durumunu `cat /etc/resolv.conf` komutu ile teyit ettik.

```
root@taylan:~# cat /etc/resolv.confnameserver 8.8.8.8nameserver 8.8.4.4
```

Ayrıca yapılan ayarların kalıcı olmasını istiyorsak konsola `update-rc.d networking deffaults` komutunu girmemiz gerekiyor.

hosts Dosyası

Yerel bir alan adı sunucusu işlevindedir. Sistemde alan adı çözümlemesi yapılırken bu dosyaya bakılır. Dosyanın konumu `/etc/hosts` şeklindedir. Hemen dosya içeriğine `cat` komutu yardımı ile bir göz atalım.

```
root@taylan:~# cat /etc/hosts127.0.0.1localhost127.0.1.1osboxes# The following lines are desirable for IPv6 capable hosts:
```

Konsol Üzerinden Dosya İndirmek

İndirmek istediğimiz dosyanın direk indirme linkini biliyorsak bu dosyamızı herhangi bir ekstra program kullanmadan veya tarayıcıya ihtiyaç duymadan konsol üzerinden `wget` komutu yardımı ile indirmemiz mümkün. Gelin anlatıma `wget` komutunu ve kullanım şekillerini anlatarak devam edelim.

wget

Eğer daha önce linux ile ilgili yönergeler okumuş veya videolar izlediyseniz `wget` komutuna mutlaka denk gelmişsinizdir. Kullanımı oldukça kolaydır ve parametreler alarak çalışır. Birkaç kullanım şekline değinecek olursak:

Tekil Dosya İndirmek :

`wget` komutunun en temel kullanım şeklidir. Hedef link aracılığı ile tekil dosya indirme işlevinde kullanılır.

Kullanımı: `wget indirilecek_dosya_linki dosya_yolu`

Örnek olarak aircrack-ng aracını indirdim.

```
root@taylan:~# wget http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz--2017-12-26 08:48:38-- http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz
root@taylan:~#
```

Ve.. bulunduğum dizin içerisine istediğim dosya indirilmiş oldu. Bu durumu `ls -l` komutu ile teyit edelim.

```
root@taylan:~# ls -ltoplam 4316-rw-r--r-- 1 root root 4379880 Şub 14 2016 aircrack-ng-1.2-rc4.tar.gzdrwxr-xr-x 3 root root 4096 Şub 14 2016 .
root@taylan:~#
```

Toplu Dosya İndirmek :

Bir metin belgesinin içerisine kayıt ettiğimiz linklerde yer alan dosyaları tek seferde `-i` karakteri sayesinde indirmemiz mümkün. Hemen örnek üzerinden bu durumu görelim.

Örneğin ben T.C Resmi Gazetesi pdflerinden 5 tanesini aynı anda indirmek istiyorum diyelim. Bunun için bu pdflerin indirme linklerini bir metin belgesine kaydetmem gerekiyor. Ben bu işlemi `nano -w resmi_gazeteler` komutu ile `resmi_gazeteler` isimli bir dosya oluşturarak ve içerisine gerekli linkleri ekleyerek gerçekleştiriyorum.

```
GNU nano 2.9.1 resmi_gazeteler Değiştirildi http://www.resmigazete.gov.tr/arsiv/538.pdfhttp://www.resmigazete.gov.tr/arsiv,
```

Linkleri ekledikten sonra artık konsola **e** onayını vererek işlemi tamamlıyorum.

Artık sıra geldi toplu indirmelere. Bu işlem için konsola **wget -i resmi_gazeteler** komutunu giriyorum.

```
root@taylan:~# wget -i resmi_gazeteler--2017-12-26 13:53:10-- http://www.resmigazete.gov.tr/arsiv/538.pdfwww.resmigazete.g
```

PDF belgelerimiz toplu şekilde indirilmiş oldu şimdi de bu dosyaların varlığını teyit etmek için konsola **ls -l** komutunu verelim.

```
oot@taylan:~# ls -ltoplam 776-rw-r--r-- 1 root root 96809 Nis 1 2010 538.pdf-rw-r--r-- 1 root root 156144 Oca 31 2011 539.1
```

Özel Konum Belirterek İndirmek :

Eğer dosyanın konsolun üzerinde çalıştığı konuma değil de bizim istediğimiz özel bir konuma inmesini istersek bunun için **wget -P kaydedilecek_dizin_yolu dosya_linki** şeklinde komut vermemiz gerekiyor.

```
root@taylan:~# wget -P ~/Desktop http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz--2017-12-26 09:04:07-- http://
```

Dosyamızın istediğimiz konuma indirmek için tam yolunu belirtmemiz gerekiyordu bu nedenle buradaki **~/Desktop** kullanımı sizi şaşırtmasın **~** işareti ana dizini temsil ediyor.

```
root@taylan:~# cd Desktop/root@taylan:~/Desktop# ls -ltoplam 4288-rw-r--r-- 1 root root 4379880 Şub 14 2016 aircrack-ng-1.1
```

Ve sonuç olarak indirdiğimiz konuma gelerek dosyanın varlığını teyit etmiş olduk.

Özel Konum Ve İsim Belirterek İndirmek :

İndirilecek dosyanın adını değiştirerek istediğimiz konuma indirmesini istersek `wget` komutumuzu `-O` parametresi ile birlikte `wget -O kaydedilecek_dizin_yolu/dosyanın_yeni_adı indirme_linki` şeklinde kullanıyoruz.

```
root@taylan:~# wget -O ~/Desktop/yeni_aircrack http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz--2017-12-26 09:1'
```

Dosyamızı `yeni_aircrack` ismi ile indirip `/Desktop` dizinine kaydettik bu durumu da hemen aşağıdaki çıktı ile teyit edelim.

```
root@taylan:~# cd Desktop/root@taylan:~/Desktop# ls -ltotlam 8572-rw-r--r-- 1 root root 4379880 Şub 14 2016 aircrack-ng-1.:
```

Ayrıca indirilen dosyanın konumunu değiştirmeden yalnızca ismini değiştirmek isterseniz de herhangi bir konum belirtmeden yalnızca yeni dosya ismini belirtmeniz yeterli olacaktır. Örneğin `wget yeni_isim dosya_link` şeklinde bir kullanım indirilen dosyanın isminde değişiklik yapılarak indirilmesini sağlar.

Kesintiye Uğrayan İndirmenin Devam Ettirilmesi :

Bir şekilde dosyanın indirme süreci kesilirse yani dosya tam olarak indirilemez ise dosyanın geri kalanının daha sonra indirmesi mümkündür. Bunun için kesintiye uğrayan dosyayı tekrar indirmek istersek `wget` komutuna ek olarak `-c` parametresi eklenmelidir. Böylelikle `wget -c indirme_linki` şeklinde bir kullanım sonucunda eğer dosya tam olarak indirilemezse sonradan dosyayı tekrar baştan indirmek yerine kaldığı yerden devam ettirme şansımız olur. Bu durum özellikle de büyük dosyalar için gerçekten çok büyük kolaylık sağlıyor.

Adım adım yaptığım işlemleri inceleyelim.

İlk olarak içi tamamen boş olan `Documents` dizinine `cd /Documents` komutu ile ulaştım. Bu dizinin içerisinde boş olduğunu `ls -l` komutu ile teyit ettim.

```
root@taylan:~# cd /Documentsroot@taylan:~/Documents# ls -ltotlam 0
```

/Documents dizini içerisinde iken `wget` komutu ile dosyayı indirmeye başladım. Daha sonra indirme işlemini `Ctrl + Z` tuş kombinasyonu ile durdurdum. Durduramadaki ama daha sonra devam ettirebilme özelliğini test etmekte.

```
root@taylan:~/Documents# wget http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz--2017-12-26 09:43:52-- http://dow
```

İndirme işlemi durdurduktan sonra `/Documents` dizinini kontrol etmek için `ls -l` komutunu kullandım. Ve dosyanın 1218402 bayt kadarının inmiş olduğunu gördüm.

```
root@taylan:~/Documents# ls -ltotlam 1196-rw-r--r-- 1 root root 1218402 Ara 26 09:43 aircrack-ng-1.2-rc4.tar.gz
```

Durdumuş olduğum indirme işlemine devam etmek üzere `wget` komutu ile beraber `-c` parametresini de kullanarak dosya indirme işlemini kaldığı yerden devam ettirdim.

```
root@taylan:~/Documents# wget -c http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz--2017-12-26 09:44:17-- http://i
```

Ve son olarak dosyanın başarılı ve tam bir şekilde indirildiğini `ls -l` komutu ile teyit ettim.

```
root@taylan:~/Documents# ls -ltotlam 4284-rw-r--r-- 1 root root 4379880 Şub 14 2016 aircrack-ng-1.2-rc4.tar.gz
```

Çıktıları karşılaştıracak olursanız başta 1218402 bayt'ta kalan indirme sonuç itibarı ile 4379880 bayt yani dosyanın tamamı şeklinde indirilmiş oldu. Yani dosya indirme işlemi yarıda dahi kesilse en baştan indirmemize gerek kalmadan `wget` komutunun `-c` parametresi sayesinde kaldığı yerden indirme işlemini devam ettirebiliyoruz.

İndirme Hızını Belirlemek :

Eğer istersek indireceğimiz dosyanın ne kadarlık internet hızında ineceğini belirleyebiliriz. Bu limitleme işlemi için komutumuzu **wget --limit-rate=indirme_limitiK /indirme_linki** şeklinde kullanıyoruz.

Örnek olması açısından ben dosyayı indirirken 300KB/s hızında indirme limiti koyuyorum.

```
root@taylan:~/Documents# wget --limit-rate=300K http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gz--2017-12-26 13::
```

Zaten üst kısımda yer alan çıktının sonucunda da bu hız sınırlamasının sonucu olarak ortalama indirme hızı (303 KB/s) olarak gözüküyor.

Arkaplanda İndirmek :

İndirmek istediğimiz dosyanın arkaplanda inmesini **-b** parametresi ile sağlarız. İndirme işleminin sonuçlarını öğrenmek istersek indirilen dosya ile aynı konumda yer alan **wget-log** dosyasını incelememiz yeterli olacaktır.

```
root@taylan:~/Documents# wget -b http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gzArdalanda sürüyor, pid 6396.Çıkı
```

Dosyamızı arkaplanada indiriyoruz ve çıktıda bize indirme detaylarının **wget-log** dosyasında olduğunu bildiriyor. Bu indirme kayıtlarına göz atmak istersek **cat wget-log** dosyasına bakabiliriz.(Çıktı çok uzun olduğundan üç nokta(...) ile çıktıyı kestim.)

```
root@taylan:~/Documents# cat wget-log --2017-12-26 13:29:55-- http://download.aircrack-ng.org/aircrack-ng-1.2-rc4.tar.gzdoi
```

Vim Editörü

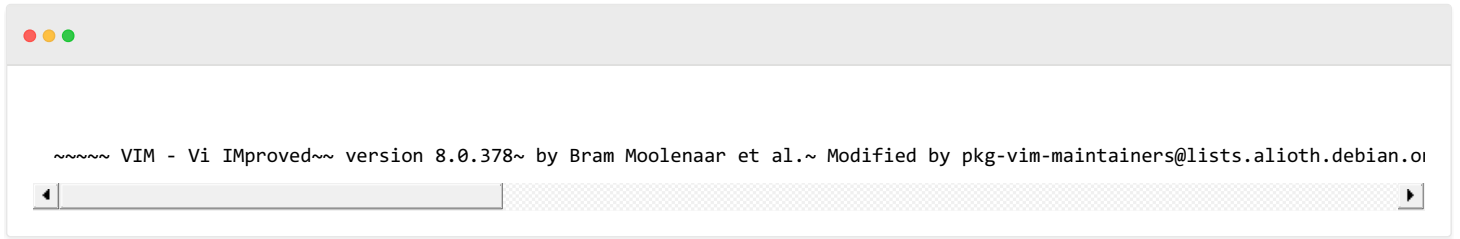
Vim Editörü Terminalden kullanılabilen ve komut alabilen yaygın kullanıma sahip bir editördür. Vim editörünün kullanımının yaygın olmasının nedeni çok hafif bir editör olmasından kaynaklanmaktadır. Uzaktan erişim durumunda bağlantı(ağ/network) üzerinde fazlaca yük bindirmeden kolayca efektif şekilde gerekli işlemleri yerine getirir. Günümüzde daha yetenekli editörler de vardır ancak biraz eski alışkanlıklardan biraz da hafif şekilde çalışmasından dolayı hala tercih edilen bir editördür. Genelde öğrenmek isteyen kişilere ilk öğrenim aşamasında fazlaca karışık delse de sonraları kullandıkça benimsenir. Bu durumu ileride sizler de

görmeyeceksiniz. Ayrıca önceki adı vi olan editörün günümüzde gelişmiş versiyonu, vim ismini almış ve görsel arayüze sahip versiyonu ise gvim adıyla isimlendirilmiştir.

vi-vim-gvim

Editörü konsoldan açmak istersek `vim` komutu yeterli olacaktır. Ayrıca görsel arayüze sahip versiyonu olan gvim editörünü açmak için de `gvim` şeklinde komut kullanımı yeterli olacaktır. Ancak görsel arayüz genel olarak pek tercih edilmiyor. Kullanıcılar terminal üzerinden tek pencerede hızlı işlem yapmak adına ve biraz da geçmişten gelen alışkanlıklarından dolayı editörü terminal üzerinden kullanmayı tercih ediyorlar.

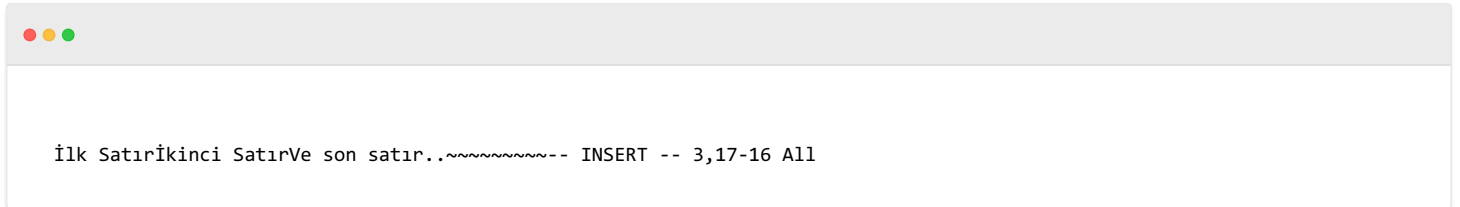
Konsola `vim` komutunu vererek editörümüzü açalım.



```
~~~~~ VIM - Vi IMProved~~ version 8.0.378~ by Bram Moolenaar et al.~ Modified by pkg-vim-maintainers@lists.alioth.debian.o
```

Editörle çalışmak için `i` tuşuna basıyoruz ve editör sol alt göşede `--INSERT--` yazısı ile içerik almaya hazır olduğunu belirtiyor.

Örnek olması açısından editöre aşağıdaki şekilde birkaç dize yazı ekledim.



```
İlk Satırİkinci SatırVe son satır..~~~~~-- INSERT -- 3,17-16 All
```

Gerekli ifadeleri girdikten sonra komut vermek istersek `ESC` tuşuna bastıktan sonra komutları `:` karakteri başta olacak şekilde vermemiz gerekiyor. Kullanabileceğimiz bazı komutlar aşağıdaki şekildedir.

:q= Eğer yapılan değişiklikler kayıt edilmişse Vim editörünü kapatır. Aksi durumda uyarı verecektir.

:q!= Editörü kaydetmeden kapatır.

:w= Yapılan değişiklikleri kaydeder.

:wq= Yapılan değişiklikleri kaydeder ve çıkar. Daha önce isim verilmemiş bir dosya ise `:wq dosya_ismi` şeklinde kullanılmalıdır.

Dosya oluşturmak üzere `:wq vi_dosyası` komutunu kullanalım.

Dosyamız oluştu, hemen bu durumu teyit etmek için `ls -l` komutunu kullanalım.



```
root@taylan:~# ls -ltoplam 36drwxr-xr-x 3 root root 4096 Ara 23 00:54 Desktopdrwxr-xr-x 2 root root 4096 Nis 26 2017 Docum
```

`vi_dosyası` isimli dosyanın oluşturulduğunu teyit ettikten sonra tekrar vim editörü ile açmak için `vim vi_dosyası` şeklinde komutumuzu kullanıyoruz.

Dosya oluşturup tekrar açtığımıza göre artık daha farklı komutlara ve diğer konsol komutlarını vim editörü aracılığı ile nasıl kullanacağımıza bakalım.

:x=Editörü kapatır ve değişiklikleri kaydeder.

:r dosya_adi=Hedef dosyayı okur ve içeriği mevcut dosyaya aktarır. Daha iyi anlaşılması için örneği inceleyin.

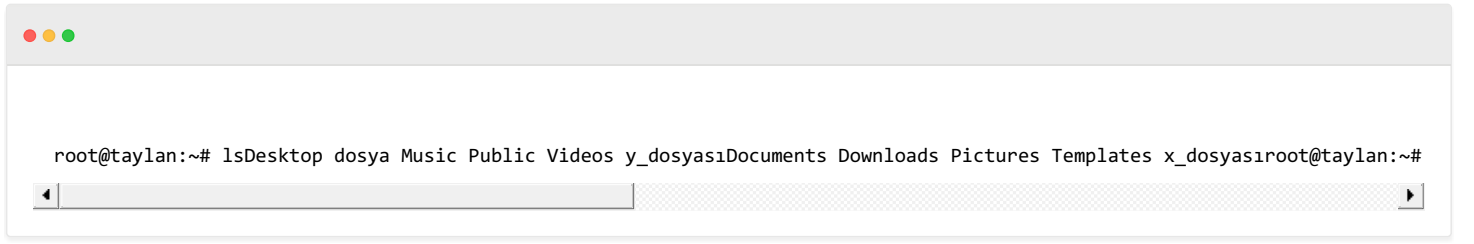
Adım adım açıklayalım:

İlk önce vim editörü ile `x_dosyası` ve `y_dosyası` isminde iki dosya oluşturuyorum ve bu dosyaların içerisine:

`x_dosyası` için ; "Bu dosya x dosyasının içeriğini temsil ediyor".

`y_dosyası` için ; "Bu dosya ise y dosyasının içeriğini temsil etmekte.."

İfadelerini ekliyorum daha sonra dosya içeriklerini `cat` komutu yardımı ile inceleyerek teyit ediyorum.



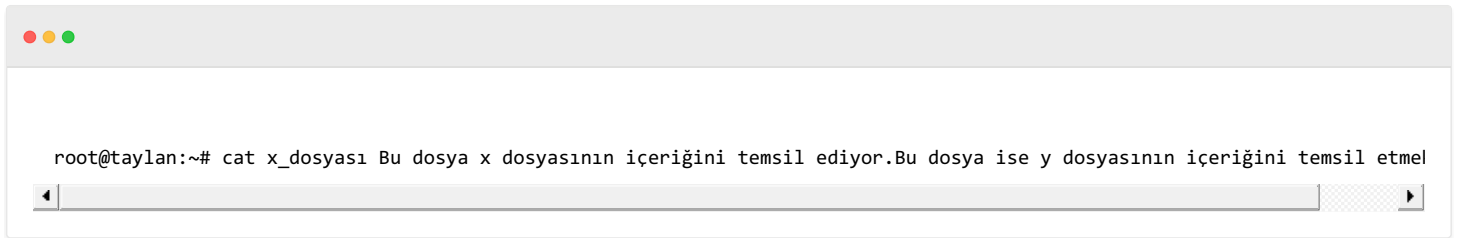
```
root@taylan:~# lsDesktop dosya Music Public Videos y_dosyasıDocuments Downloads Pictures Templates x_dosyasıroot@taylan:~#
```

Sıra geldi vim editörü ile oluşturulan bu dosyalardan birini diğerine eklemeye. Ben `x_dosyası`'na `y_dosyası` içeriğini eklemek üzere, `x_dosyası`'ni `vim x_dosyası` komutu ile açıyorum.

Vim editörü içerisinde açılan `x_dosyası`'ndayken `:r y_dosyası` komutunu veriyorum.

Böylelikle `y_dosyası` içerisinde yer alan ifadeler `x_dosyası` içerisine eklenmiş oldu.

Bu durumu hemen `cat` komutu yardımı ile teyit edelim.



```
root@taylan:~# cat x_dosyası Bu dosya x dosyasının içeriğini temsil ediyor.Bu dosya ise y dosyasının içeriğini temsil etmel
```

Konsol komutlarını vim editörü içerisinde kullanabileceğimizden bahsetmiştik. Bunu yapabilmemiz için komuttan önce iki nokta üst üste ve ünlem karakterlerini kullanmalıyız yani örnek kullanımı `!:komut` şeklindedir. Hemen bu durumu `ls` komutumuz için deneyelim.

```
root@taylan:~# vim x_dosyası Desktop dosya MusicPublic Videos y_dosyasıDocuments Downloads PicturesTemplates x_dosyasıPres:
```

Vim editörü içerisinde `:!ls` komutunu verdiğimizde sistem otomatik olarak vim editöründen çıkarak konsolda bize `ls` komutunun çıktılarını bastı. Eğer bu durumda tekrar vim editörüne dönmek istersek `enter` tuşuna basmamız yeterli olacaktır.

Şimdi biz neden konsol varken bu editörden komut vermekle uğraşalım diye düşünmüş olabilirsiniz sizlere bu durumu örnek ile açıklayalım.

Örnek olması açısından `ls` komutunun çıktılarını bir dosyaya kaydetmek isteyelim.

Bunu için vim editörü ile bir dosya oluştuyorum.

```
~~~~~"dizin_çıkıtları" [New File] 0,0-1 All
```

`vim dizin_çıkıtları` komutu ile vim editörü içerisinde boş bir dosya açıyorum ve bu dosya içerisindeyken `:r !ls` komutunu veriyorum.

Komut sonucunda oluşturmuş olduğum `dizin_çıkıtları` dosyası içeriği aşağıdaki şekilde oluyor.

```
DesktopDocumentsdosyaDownloadsMusicPicturesPublicTemplatesVideosx_dosyasıy_dosyası~~~~~ 12,9 All
```

Kullandığımız komut bütününü açıklayacak olursak (aslında bunları zaten biliyorsunuz yine de açıklama gereksinimi duydum) `:r` komutu içerik aktarma işlevinde `!ls` komutu ise dizin altındaki dosya-klasörleri listelemekle görevli.

Dosyayı `:x` komutu ile kaydediyorum ve ayrıca teyit etmek için `cat dizin_çıkıtları` komutunu kullanıyorum

```
root@taylan:~# cat dizin_çıkıtları DesktopDocumentsdosyaDownloadsMusicPicturesPublicTemplatesVideosx_dosyasıy_dosyası
```

Vim editörü ile birden fazla dosya ile işlem yapmamız mümkün bunun için komutu `vim dosya_1 dosya_2 dosya3` şeklinde kullanabiliriz.

Komutumuzu verdikten sonra `dosya_1` vim editöründe açılacaktır. Gerekli değişiklikleri yaptıktan sonra eğer diğer dosyalara geçiş yapmak istersek çalıştığımız dosyayı kaydettikten sonra `:next` komutu ile `dosya_2`'ye geçiş sağlanacaktır. Aynı şekilde `dosya_3` dosyasına da geçmek mümkün olacaktır.

Eğer geçiş yaptığımız dosyadan bir öncekine dönmek istersek `:previous` komutunu kullanmamız yeterli olacaktır.

Ayrıca kaç dosyanın vim editöründe çalıştığını pencere ekranının üst bilgi kısmından öğrenebiliriz. Örneğin ben 3 dosya ile çalıştığım için üst kısımda bulunduğum dosya bilgisi ile beraber `dosya_1 (~) (1 of 3) - VIM` şeklinde bir üst bilgi yazısı bulunuyor.

Dosya içinde bulunan ifadeleri değiştirmek istersek `:s/satır_sayısı s/eski_ifade/yeni_ifade/g` şeklinde bir komut bütünümlü bulunuyor. Bu gösterim ile anlamamış olabilirsiniz o yüzden hemen birkaç örnek ile konuyu açıklayalım.

`linux` isimli bir dosya oluşturuyorum ve içerisine aşağıdaki ifadeleri ekliyorum.

```
kali linuxlinux mintubuntuFedora linuxopen SUSEManjoroSalix linux
```

İlk önce burada yer alan 1. ve 2. satırdaki `linux` ifadelerini `LINUX` ile değiştirmek üzere vim editörü içerisinde iken `:1,2s/linux/LINUX/g` komutunu veriyorum. Daha sonra `:x` komutu ile dosyayı kaydederek vim editörünü kapatıyorum.

Dosya içeriğini `cat` komutu ile sorgulayalım.

```
root@taylan:~# cat linuxkali LINUXLINUX mintubuntuFedora linuxopen SUSEManjoroSalix linux
```

Görüldüğü gibi 1. ve 2. satırdaki `linux` ifadeleri `LINUX` şeklinde değişmiş oldu.

Eğer tüm satırlardaki `linux` ifadelerini `LINUX` olarak değiştirmek istersek komutumuzun başına yüzde işareti ekleyerek `:%s/linux/LINUX/g` şeklinde kullanmamız yeterli olacaktır.

Komutumuzu verdikten sonraki çıktı ile bu durumu teyit edelim.

```
root@taylan:~# cat linuxkali LINUXLINUX mintubuntuFedora LINUXopen SUSEManjoroSalix LINUX
```

Vim editörü çok güçlü bir araç ve gerçekten çok farklı kullanım şekilleri-özellikleri var. Yani bu editörün kendisini anlatmak zaten başlı başına koca bir kaynak oluşturmakla eşdeğer. Hatta bu durumu vim editörünün kendi [wiki kaynağından](#) da teyit edebilirsiniz.

Biz yine de birkaç kısa kullanım şeklini daha görelim.

Eğer direk olarak konsol üzerinden veri girişi yapmak istersek konsola `vim -` komutunu girmeliyiz.

Böylelikle konsol bize `Vim: Reading from stdin...` çıktısını verecek ve bizden veri girişi bekleyecektir.

Ben örnek olması açısından konsoldan birkaç ifade ekliyorum.

```
root@taylan:~# vim -Vim: Reading from stdin...deneme satır11deneme satır12 ve.. deneme satır13
```

İfade giriş işlemim tamamlandığında `Ctrl + D` tuş kombinasyonu basarak, vim editörü ekranına yazdığım ifadeler ile birlikte dönmüş oluyorum.

```
deneme satır11deneme satır12ve.. deneme satır13~~~~~"d" 3L, 57C 1,1 All
```

Ayrıca oluşturduğumuz dosyaları şifreleyerek koruma imkanımız da var. Bu durumu bir örnek üzerinden anlatalım.

`vim vim_metni` komutu ile bir belge açıyorum ve içeriğini bazı ifadeler ekleyerek belgeyi kaydediyorum.

Dosyamızın özelliklerini `file vim_metni` komutu ile öğrenip, dosyanın içeriğini ise `cat vim_editörü` komutu ile okuyalım.

```
root@taylan:~# file vim_metni vim_metni: UTF-8 Unicode textroot@taylan:~# cat vim_metni Şifrelenmek üzere oluşturulmuş bir
```

İstediğimiz bilgilere kolayca ulaşabildik. Bu durumu birde dosyamızı şifreleyerek test edelim. Bunun için konsola `vim -x vim_metni` şeklinde komut veriyoruz ve konsol bizden şifre belirlememizi istiyor.

```
Enter encryption key: *****Enter same key again: *****
```

Dosyamıza birkaç veri daha ekledikten sonra dosyamızı `:w` komutu ile kaydettik. Ve sonuç olarak vim editörü alt bilgi çubuğunda bize dosyanın şifrelendiğini aşağıdaki çıktıda yer alan `"vim_metni" [cryptd]` şekilde ifade ederek belirtti.

```
"vim_metni" [cryptd] 4L, 151C written 4,14-13 All
```

Şimdi dosyamızı tekrar `file` ve `cat` komutları ile sınavalım.

```
root@taylan:~# file vim_metni vim_metni: Vim encrypted file
root@taylan:~# cat vim_metni VimCrypt~01!F*?s?70?5
```

Çıktı da görüldüğü üzere dosyamız artık şifrelenmiş durumda. Yani dosyayı tekrar okunaklı şekilde açabilmemiz için şifresini girmemiz gerekecek. Bu durumu hemen `vim_metni` isimli dosyamızı açmaya çalışarak test edelim. Bunun için konsola `vim vim_metni` komutunu giriyoruz. Ve konsol bize aşağıdaki çıktıyı veriyor.

```
Need encryption key for "vim_metni"Warning: Using a weak encryption method; see :help 'cm'Enter encryption key:
```

Konsol çıktısında da görüldüğü gibi bizden bir parola isteniyor. Eğer parolayı doğru girersek dosyamız açılacaktır.

Ancak sizlere önemli bir uyarı;

Bu dosya şifreleme olayı iyi hoş ancak çok çok önemli bir nokta var o da belirlediğiniz şifre. Eğer uzun, karmaşık yazarsanız bu şifreyi unutma riskiniz yüksek olacağından daha sonra bulmanız veya bir şekilde kırmanız da bir o kadar zor olacaktır. Ayrıca kısa ve

kolay şifreler de kolayca kırılabilirliğinden pek güvenli bir yol sayılmaz. Yani anlayacağınız ne unutacağınız kadar uzun veya karmaşık ne de kolay kırılacak kadar kısa olsun. Şimdiden sizleri uyarıyorum sonradan bu durum yüzünden aman başınız ağrımasın.

Konumuza yavaş yavaş bitirmek üzere devam edecek olursak.

Örneğin vim editöründe kod yazıyordunuz ve kod 101. satırda hata verdi. Bu durumda 101. satırı aramak yerine editöre `:101` şeklinde hatalı kodun bulunduğu satır girerek ilgili satıra ulaşmış olabiliyoruz. Yani direk olarak atlamak istediğimiz satırı vim editörüne `:satır_numarası` şeklinde girerek bu işlemi gerçekleştirebiliriz.

Eğer yazdığınız ifadeler içerinden herhangi bir ifadeyi bulmak isterseniz `/aranacak_ifade` şeklinde yazarsanız ilgili ifadeye ulaşmış olursunuz. Ve aynı ifadenin başka yerlerde geçmesi durumunda bir sonraki ifadeye gitmek için `:/` komutu kullanılırken bir önceki aynı ifadeye dönmek için ise `:?` komutlarını kullanıyoruz.

Eğer aradığımız ifadede büyük küçük harf duyarlılığı olmadan hedefteki aranan ifadenin aynısı olan bütün ifadeleri bulmak istersek `:set ic` komutunu verdikten sonra ifademizi `/ARNACAK_İFADE` şeklinde ya da `/aranacak_ifade` hatta `ArAnAcAk_İfAdE` şeklinde dahi belirtebiliriz. Sonuçta aynı karakter bütünü içeren ifadeler karşımıza gelecektir.

Vim editörü anlatımının burada sonuna geliyoruz. Ancak dediğim gibi zaten vim editörü başlı başına bir dökümantasyon konusu ki bu bilgileri burda izah etsek bile fazlaca gereksiz ayrıntıya değinmiş, sizlere boş yere yormuş olurduk. Vim editörü ancak baştan beri bahsi geçen kendi kendine öğrenme, deneme, alıştırma yapma ve özümseme ile öğrenilecek bir konudur. Kullanmanız şart değil kimisi çok sever kullanır kimisi ayrıntılarından nefret eder bırakır.

Neticede internette yer alan tüm zengin kaynaklar ile birlikte man kılavuz sayfasından, vim editörü içerisinde iken `:help` komutundan ve daha önce verdiğim vim editörü [wiki kaynağından](#) diğer tüm bilgilere ulaşabilirsiniz.(Benim önerim vim editörü içerisindeki `:help` komutu ile ulaşabileceğiniz dökümantasyondan takip etmenizdir.) Ve unutmayın bu araç ancak ve ancak kendi kendine kurcalayacak öğrenilebilir.

Zamanlanmış Görevler

Sistemde yapılması gereken rutin işlerin zamanı geldiğinde otomatik olarak yapılması işine zamanlanmış görevler deniliyor. Örneğin benim her pazartesi sistemi yedeklemem gerekiyor diyelim, bunu her pazartesi elle yapmak yerine bu işi zamanlanmış görevlere ekleyerek otomatik bağlayabilirim.

cron

Rutin tekrarları sağlayan zamanlanmış görevleri yerine getirmemizi sağlayan servisimizin adı `cron` 'dur.

Anlatıma cron servisinin çalışma durumunu kontrol ederek başlayalım. Bunun için konsola `service cron status` komutunu verelim.


```
root@taylan:~# service cron status● cron.service - Regular background program processing daemon Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled) Active: active (running) since Mon 2020-07-20 15:04:03 CEST; 1min 45s ago Main PID: 1000 (cron) CGroup: /systemd/system/cron.service └─1000 /usr/sbin/cron
```

Cron servisinin çalıştığını teyit etmiş olduk. Cron servisinin yapılandırma dosyası `/etc/crontab` konumunda yer alıyor göz atmak için konsola `cat /etc/crontab` komutunu verebiliriz.

```
root@taylan:~# cat /etc/crontab# /etc/crontab: system-wide crontab# Unlike any other crontab you don't have to run the `crontab` command to install the changes. You may use `systemctl restart cron` to reload the changes. # You can use 'systemd timers' as well. Example entries can be found in the /etc/cron.d/ directory: # * * * * * root run-parts /etc/cron.d
```

Şimdi de düzenleme yapmak adına konsola `crontab -e` komutunu verelim. Burada yer alan `-e` parametresi edit kelimesini yani düzenleme ifadesini temsil ediyor.

```
root@taylan:~# crontab -eNo crontab for root - using an empty oneSelect an editor. To change later, run 'select-editor'. 1. Press <Return> to use the default, 2. press <Space> to toggle between <Return> and <Esc>, or 3. press <Return> or <Esc> to set the default. Press <Return> or <Esc> to enter the editor.
```

Konsol bize düzenleme yaparken hangi aracı kullanmak istediğimizi soruyor. Ben nano aracını seçiyorum yani konsola 2 rakamını giriyorum.

```
GNU nano 2.9.1 /tmp/crontab.9WSzPj/crontab # Edit this file to introduce tasks to be run by cron.# # Each task to run has to be preceded by a '#' followed by a command. # * * * * * root run-parts /etc/cron.d
```

Çıktıda bize bir örnek kullanım gösterilmiş biz bu kullanımı ele alarak servisin kullanım şeklini görelim.

```
#İş tanımı örneği:----- dakika (0 - 59) | .----- saat (0 - 23) | | .----- ayın günü (0 - 31) | # * * * * * root run-parts /etc/cron.d
```

Yukarıdaki şema yeterince açıklayıcı olmadıysa endişelenmenize gerek yok. Örnekler üzerinden yapacağımız anlatım ile cron servisimizin kullanımını rahat şekilde anlamış olacağınıza inanıyorum.

Zamanlanmış görevleri yerine getiren servis `cron` ve bu servisin yerine getirdiği işlere de cron job deniliyor. Bu tanımın çok da önemi yok ancak bunu böyle bilerseniz daha iyi olur. Bizler de kendi zamanlanmış görevlerimizi (cron jobs) atamak istersek `crontab -e` komutu ile açtığımız dosyanın son satırına yapmak istediğimiz işi ve tam tarih düzenini girmemiz gerekiyor. Bu görevleri nasıl oluşturacağımızı konunun devamında örnekler üzerinden öğreniyor olacağız.

Örnekler

Kullanımın anlaşılması amacıyla örnekleri olabildiğince basit tuttum.

O anın tarihi detaylarını `zamanlar.txt` isimli klasöre kaydetmek için `echo `date` zamanlar.txt` komutunu kullanacağım.

Bütün yıl boyunca her dakikanın çıktılarını kaydetmesini istersem aşağıdaki şekilde bir kullanım işimizi görür.

```
DAKİKA SAAT AYIN GÜNLERİ AYLAR HAFTANIN GÜNLERİ KULLANICI ADI KOMUT * * * * * root echo `date` zamanlar.txt
```

Yıldız işareti (*) olan kısımlar hepsini kapsamaktadır yani haftanın bütün günleri, ayın bütün günleri ve saatin bütün dakikaları.. gibi.

Komutun düzenli hali `* * * * * echo `date` >> ~/Desktop/zamanlar.txt` şeklindedir.

Bir süre geçtikten sonra masaüstünde(Desktop dizininde) yer alan `zamanlar.txt` dosyasını `cat zamanlar.txt` komutu ile kontrol ediyorum.

```
root@taylan:~/Desktop# cat zamanlar.txt Çrş Ara 27 16:10:01 +03 2017Çrş Ara 27 16:11:01 +03 2017Çrş Ara 27 16:12:01 +03 20:
```

Görev istediğimiz şekilde yani her dakikada yazılacak şekilde otomatik olarak gerçekleşmiş oldu.

Şimdi spesifik bir zaman belirleyerek görevin yerine getirilme durumunu ele alalım.

Her gün 14:15 da otomatik olarak `test.txt` isimli belgeye "test yazısı" yazdırmak için kalıbı crontab dosyasına aşağıdaki şekilde yazmalıyız.(Ayrıca başta hangi ifadenin neyin karşılığı olduğunu belirttiğim için tekrar tekrar üzerlerine yazmıyorum. Eğer şaşırırsanız yukarıdaki kısımları tekrar gözden geçirebilirsiniz.)

```
15 14 * * * echo "test yazısı" >> ~/Desktop/test.txt
```

```
root@taylan:~/Desktop# dateWed Dec 27 14:15:02 GMT 2017root@taylan:~/Desktop# lstdedst.txt tes.txt test.txtroot@taylan:~/D
```

`ls -l` komutu ile de dosyamızın üretildiği tarihi sorguladık ve çıktıdan da anladığımız üzere verdiğimiz görev zamanında yerine getirilmiş.

Eğer belirli bir saat-tarih arasında belirli kere tekrar eden görev atamak istersek kullanım şekli aşağıdaki örnekteki gibi olacaktır. Örneğin 5 dk bir çalışmasını istediğiniz komutu aşağıdaki şekilde crontab dosyasına ekleyebilirsiniz.

```
*/5 * * * * komut
```

Her saatin ilk 5 dk 2 kere çalışmasını istersek kullanım şekli aşağıdaki örnekteki gibi olacaktır.

```
0-5/2 * * * * komut
```

Örneğin cron job yani verdiğimiz görev komutu günün ilk 5 saatinde 10 defa çalışmasını istersek kullanım şekli aşağıdaki örnekteki gibi olacaktır.

```
* 0-5/10 * * * komut
```

Örneğin cuma günleri 12:30 ve 17 arasında 5 kez çalışsın istersek kullanım şekli aşağıdaki örnekteki gibi olacaktır.

```
30,0 12-17/5 * * 5 komut
```

Elbette örnekleri çoğaltmak mümkün ancak daha fazlası gereksiz olacaktır. Yani bu konu sizlerin de biraz kurcaladıktan sonra anlayacağı kolay bir konudur.

Eğer belirlenen zamanlanmış görevleri listelemek istersek `crontab -l` komutunu kullanabiliriz. Buradaki `-l` parametresi list kelimesini yani listeleme işlevini temsil ediyor.

Çıktı çok uzun olmasın diye yalnızca cron jobs kısmını yani görevlerin listelendiği kısmı alta ekledim.

```
* * * * * echo `date` >> ~/Desktop/zamanlar.txt15 14 * * * echo "test yazısı" >> ~/Desktop/test.txt
```

Şayet oluşturduğumuz tüm zamanlanmış görevleri silmek istersek `crontab -r` komutunu kullanmalıyız. Buradaki `-r` parametresi remove kelimesini yani silme işlevini temsil ediyor.

Log Kayıtları

Sistemde olan sorunlar, işlemler, değişiklikler ve neredeyse her şey kayıt altına alınarak saklanır. Bu kayıt altına alınan bilgilere log deniyor. Neden log tutulmak zorunda diye soracak olursanız kısaca sistemin olumsuz bir durumla karşılaşması halinde sorunun yaşanma nedeninin belirlenmesi, sistem güvenliğini sağlamada, gerektiğinde veri kurtarmada ve adli bilişim gibi alanlarda başvurmamız gereken yegane kaynaklardır. Yani sistem bütünü için çok önemli yer tutmaktadır.

Log Dosyaları

Linux sisteminde log dosyalarının çok büyük kısmı `/var/log` dizini altında tutuluyor. Ayrıca log dizini içerisinde de belli başlı programlara ve servislere ait logları bulunduran başka alt dizinler bulunuyor. Bu durumu gözlemek için komut satırına `cd /var/log` yazarak log dosyalarının tutulduğu dizine gidip dizin içerisinde iken `ls` komutu ile dizin içeriğini listeleyelim.

```
root@taylan:~# cd /var/logroot@taylan:/var/log# lsalternatives.log daemon.log.2.gz kern.log postgresql user.log.2.gzalterni
```

Örneğin `user.log` dosyasını `cat user.log` şeklinde açarak sistemde yapılmış olan oturum açma işlemlerini ve detaylarını inceleyebilirim. Ancak bu noktada bir kısa bilgi linux sistemi log dosyalarının çok fazla yer kaplamasını önlemek için üzerine yazma metodunu kullanıyor. Bu noktada cron servisi ile log kayıtlı her hafta eklenenerek maksimum 1 ay kadar eskiyi yani 4 haftayı kayıt altında tutuyor.

Log dosyalarını incelerken kolaylık olması açısından daha önce de kullanmış olduğumu ve dosyanın alt(tail/kuyruk) kısımlarını listleyen `tail` komutundan yararlanacağız.

Bu sayede uzun uzadıya dosyanın tamamına bakmak yerine son eklenen bilgileri inceleyebileceğiz.

```
root@taylan:/var/log# tail -n 3 user.logDec 27 12:34:43 taylan gvfsd-trash[1965]: dir: /root/.local/share/Trash/files, fil
```

dmesg

Sistem açılışından itibaren çekirdek tarafından üretilen tüm iletiler iletiler ve kernel hakkındaki kayıtlar `/proc/kmsg` dizininde tutuluyor. Ancak biz bütün kernel kayıtları yerine, sistem açılışında yazan açılış notlarını `dmesg` komutu ile görüntüleyebiliriz. Yani `dmesg` komutu sadece tampondaki son iletileri gösterir. Bu komutun kullanımına genelde sistem açılışında bildirilen problemlerin tespiti ve diğer sistem uyarılarını saptamak için başvurulur. Yani genelde sorun yaşadığınızda forum ve benzeri topluluklarda sizden bu komut istenirse komutun hangi amaca hizmet ettiğini bilmeniz için ve aynı zamanda komutumuz log konusu ile bağlantılı olduğu için sizlere açıkladım.

Elbetteki çıktı çok daha uzun ancak ben örnek olması açısından çıktıları kısaca aşağıda verdim.

```
root@taylan:/proc# dmesg[ 0.000000] Linux version 4.14.0-kali1-amd64 (devel@kali.org) (gcc version 7.2.0 (Debian 7.2.0-16))
```

last

Sistemde oturum açan kullanıcıları listelemek için `last` komutunu kullanabiliriz.

```
root@taylan:~# lastroot :1 :1 Wed Dec 27 13:55 gone - no logoutreboot system boot 4.14.0-kali1-amd Wed Dec 27 13:42 still i
```