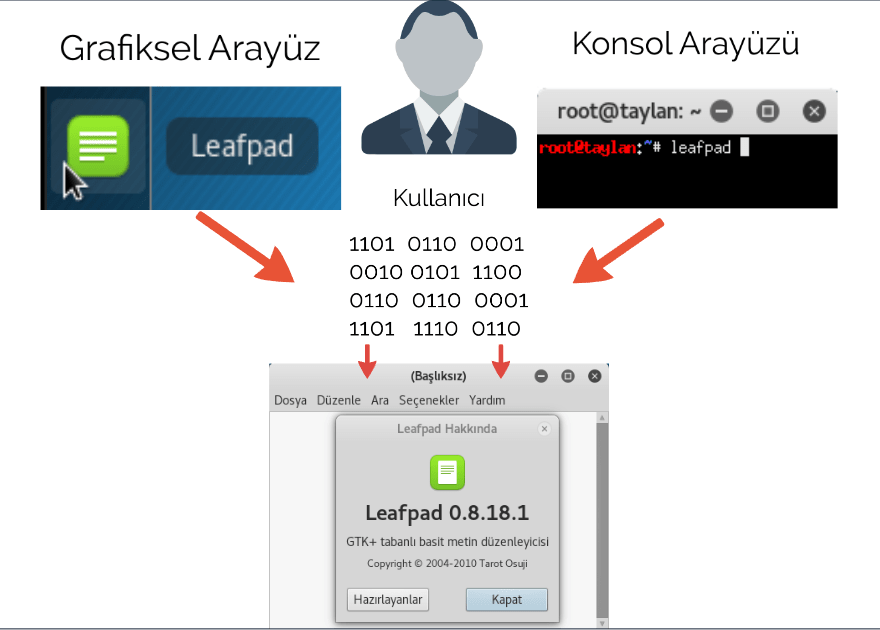
#### Komut Satırı

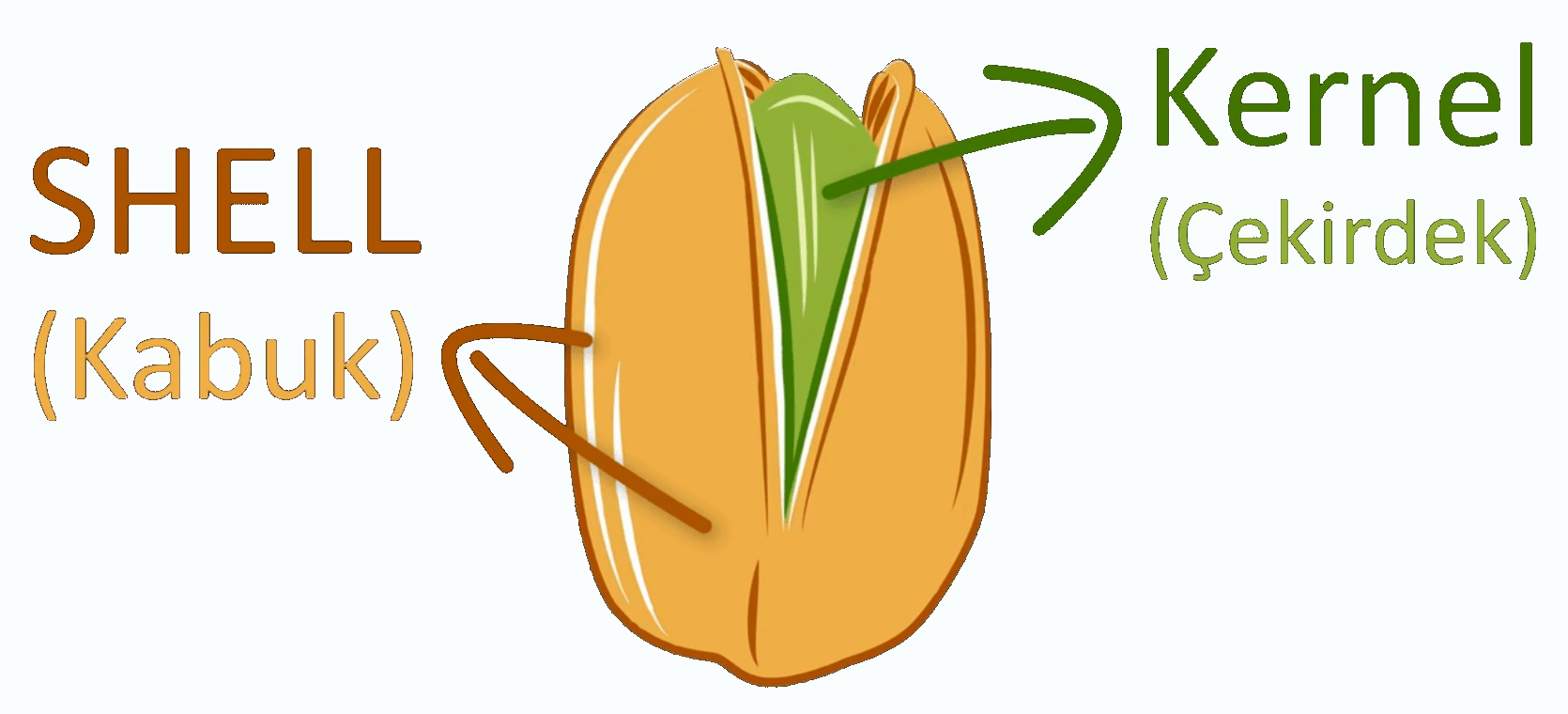
Esaslı bir giriş yapabilmemiz için öğrenmemiz gereken ilk kavram **Linux Komut Satırı** diğer bir adıyla **Linux Terminali** (**konsol**) olacaktır. Ancak bundan önce Linux işletim sisteminin(GNU/**Linux**) **çekirdeğini** ele almalıyız. Linux çekirdeği Linux'un kalbi kabul edilir. En önemli ana görevinin tanımı kısaca; **yazılımla donanımın haberleşmesini sağlamaktır**. Ayrıca Linux'un yani çekirdeğin ingilizce karşılığı "**kernel**"dir. Yabancı terimleri sizlere aktarıyor olma nedenim; eninde sonunda bu terimlere alışmamız gerektiği ve bu durumun ne kadar erken olursa bizim için o kadar iyi olacağıdır. Kernel, yani çekirdeği biraz daha açıklayacak olursak:

#### Kernel

Biz ister grafiksel arayüzü kullanalım, istersek de yalnızca komut satırını kullanacak olalım; örneğin bir dosyayı bir yerden başka bir yere taşırken; ister sürükle bırak yöntemiyle(grafiksel arayüz kullanarak) taşıyalım, istersek de bu işi komut satırından komutlar yardımı ile gerçekleştirelim; neticede yapılan işlem aslında arka tarafta komutların yorumlanarak çalıştırılması ile gerçekleşmektedir. [](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/0.png)İşte tüm bu işlemleri gerçekleştirmekle görevli bazı yapılar vardır. **Çekirdek(kernel) de kullanıcıdan gelen girdilerle birlikte sistemin işleyebilmesi (process) için donanıma iş yaptırmakla görevlidir**. Ancak direk 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 fıstık düşünün. Bu fıstığın dış katmanı Kabuk (Shell) iç kısmı ise Çekirdek (Kernel) olacaktır. [](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/shell-kernel.png)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 kernele 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**.

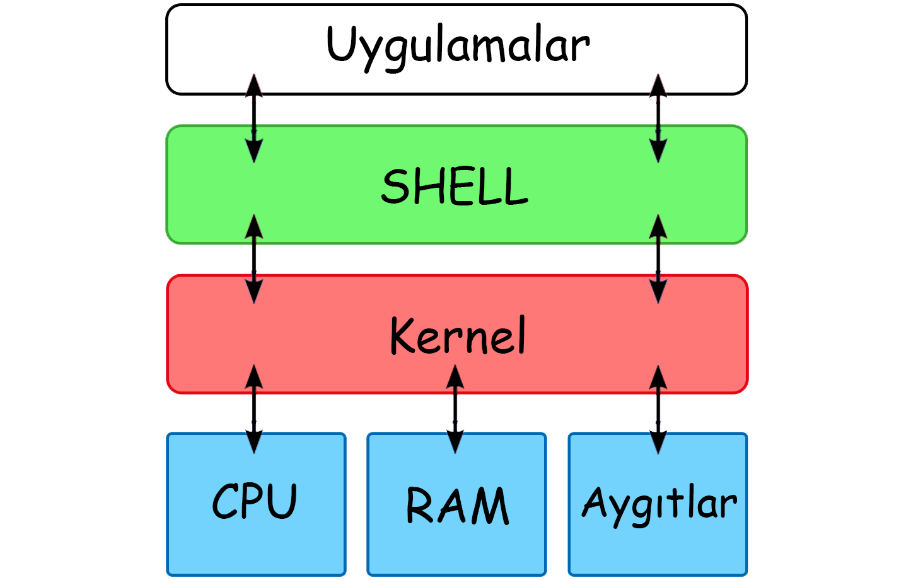
[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/1.png)

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 direk konsol şudur demedik. Ancak zaten bu kısıma kadar **Konsol**'un ne olduğunu dolaylı olarak da olsa açıklamış olduk.

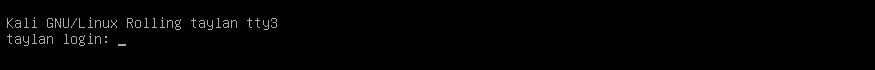
**Konsol, kullanıcı ile Shell arasından yer alarak kullanıcının komut girmesini sağlayan grafiksel ve komut satırı arayüzüne sahip bir araçtır.** Biz komutlarımızı bu araç aracılığı ile Shell'e ulaştırırız Shell ise kullanıcıdan gelen girdileri yorumlayarak Kernel'e aktarır.

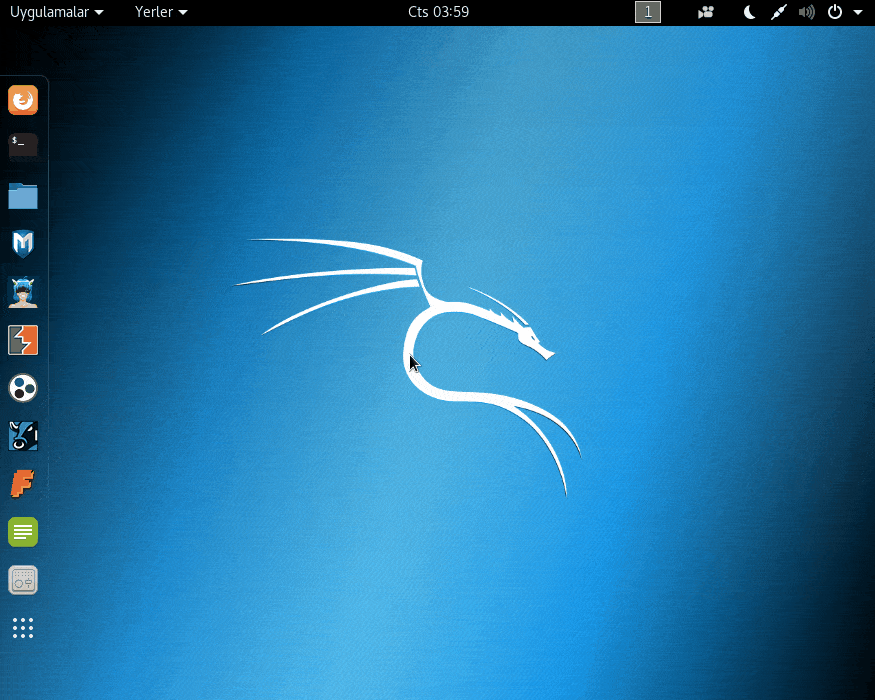
Aşağıdaki basit diyagrama bakarak bu yapıları daha net kavrayabilirsiniz.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/diagram.png)

#### Konsol Ekranı Açmak (Terminal/Komut Satırı)

Konsolu daha yakından incelemek adına öncelikle bir konsol ekranı açalım. Konsol ekranını açmak için kısayol tuşları olduğu gibi grafiksel arayüz aracılığı ile de açmak mümkündür. Ayrıca aynı anda birden fazla konsol ekranı da kullanabiliyoruz. Yani Linux bize çok sayıda konsol açma imkanı tanıyor. Örenğin 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 beklenmektir.) Grafiksel arayüzden komut satırına geçmek için **Crtl + Alt + (f1,f2,f3,f4,f5,f6)**tuş kombinasyonlarını kullanabiliriz. Demiştim ya 6 farklı konsol açılabiliyor 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 uyguladığımda karşıma aşağıdakine benzer bir komut satırı geliyor ve benden login olmamı yani kullanıcı adımı ve sonrasında şifremi yazarak giriş yapmamı bekliyor.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/2.png)Eğer bu komut satırından çıkıp tekrar kullanıcı arayüzüne yani grafiksel arayüze geçmek isterseniz **Ctrl + Alt + (f7,f8,f9,f10,f11,f12)** tuş 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 dönmenizi 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 yardımcılar aracılığı ile lütfen “konsol” veya “terminal” şeklinde aratarak kendiniz bulun. Konsol(terminal) açma işlemi hemen aşağıda gif resmi ile gösterilmiştir.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/3.gif)

Bu işlemeler için kısayolların da kullanılabildiğinden bahsetmiştim, örneğin bazı dağıtımlarda **Ctrl + Alt + T** tuş kombinasyonu direk olarak terminal ekranını açıyor. Ancak sistemdeki kısayollar; ayarlar aracılığı ile ve ayrıca da dağıtımdan dağıtıma farklılık gösterdiği için bu kısayolları sizin kendi kullandığınız dağıtıma göre araştırarak bulmanız gerekiyor. Korkmayın inanılmaz kolay bir işlem, hem böylelikle yavaş yavaş Linux öğrenmenin aslında araştırmak ve kendi kendine öğrenmek olduğunu kavrıyoruz. Zaten ileride Kali Linux için [**kısayollar**](https://www.udemy.com/draft/2180440/learn/v4/t/lecture/13436144) konularına ayrıca değineceğim. Yani ileride kısayollar konusunu teferruatlıca işleyeceğiz merak etmeyin. 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 ise bir parametre ve tanımlanan değişkene normal bir kullanıcı olarak erişebilmeyi sağlıyor. Şimdilik bu komut bütünü hakkında bu kadar bilgi yeterli. Eğer komutun açıklamasını anlamadıysanız hiç bir sorun yok. Okumaya devam ettikçe parçalar yerine oturmaya başlayacak. Zaten ileride bu konuların her birine sırasıyla tek tek değineceğiz ve sizler de anlamış olacaksınız. Şimdi devam edecek olursak:

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/4.png)

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, gereksiz detaylara girdiğimi düşünenler olabilir ancak emin olun öğrendiğiniz her şey birbiri ile bağlı ve bütünü anlamak için bilinmesi gerekli olan kavramlardır. 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 sizler burada yazanları yalnızca kuru kuruya okumayacak, ilerleyiş sırasında konu içerisinde gösterilen uygulamalar ile konuları ve kavramları tam olarak özümsemiş olacaksınız. Tam da bu noktada uygulama yaparak öğrenmenin önemini belirten **Konfüçyüs**'ün bir sözünü hatırlatmak istiyorum:

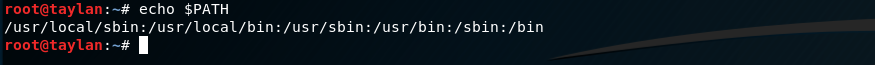
#### “Bana anlat unuturum, bana göster hatırlarım, beni dahil et, anlarım.”

İlerleyişimize gelin ortam değişkenlerini tanıyarak devam edelim.

#### Ortam Değişkenleri (Çevre Değişkenleri)

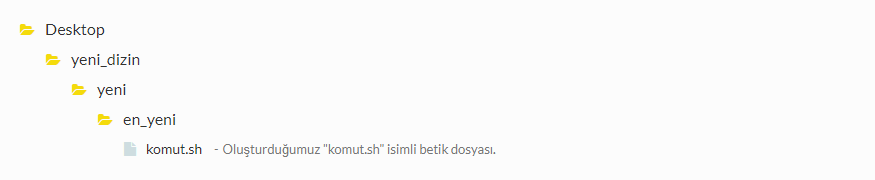
Ortam değişkenlerini anlamak için bir örnek üzerinden gidelim. Örneğin grafiksel arayüzde komut satırına leafpad yazdığımızda karşımıza leafpad(metin editörü) programı gelir. (Özellikle grafiksel arayüz diye belirttim çünkü leafpad uygulaması grafiksel arayüz ile çalışmaktadır yani tek başına komut satırının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

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/5.png)

Çıktının bize gösterdiği yani konsolun demek istediği şu: **Konsoldan girilen herhangi bir 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 yine de emin olamadım diyenler için gelin **PATH** yoluna ekli olmayan bir progamı kendimiz ekleyelim ve konsoldan vereceğimiz bir komutla direk olarak çalışmasını sağlayalım. Adım adım ilerleyelim.

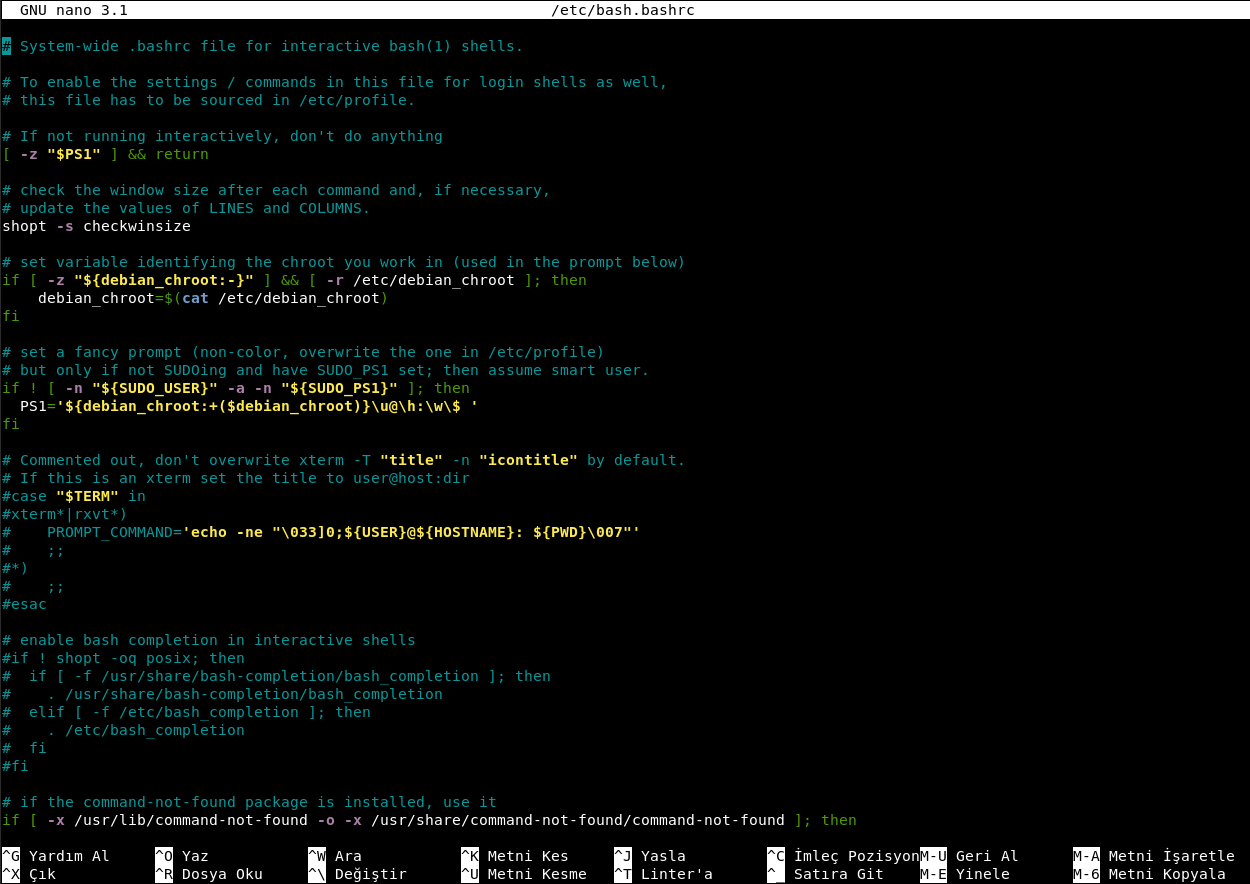
Ben masaüstünde "**yeni\_dizin**" adında bir klasör oluşturdum daha sonra içerisine "**yeni**" diye başka bir klasör daha oluşturdum son olarak bu klasörün de içerisine "**en\_yeni**" şeklinde bir klasör daha oluşturdum. Yani oluşturduğum dizinin tam adresi **yeni\_dizin /yeni /en\_yeni** şeklinde oluşmuş 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 klasör dizininin en altındaki klasöre yani "**en\_yeni**" ismindeki klasöre dosyamı "komut.sh" ismiyle kaydediyorum. Dosya sonuna eklediğimiz **.sh** eki ile dosyamız bir **betik dosyası** halini almış oldu. Burada bahsi geçen betik dosyasının genel tanımı; içerisine yazılmış olan konsol komutlarını tek seferde bir bütün halinde düzenli şekilde çalıştırabilen dosya formatı şeklinde yapılabilir. Yani bizler böylelikle betik dosyası programlayıp konsola yapması gereken bir çok işlemi tek bir dosya ile yaptırabiliriz. Anlayacağınız bu betik dosyası sıradan bir dosya değil, program yerine geçen bir formattır. Konsol programıdır yani.. 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. Oluşturmuş olduğum betik dosyasının dizin içerisindeki tam konumu aşağıdaki şekildedir.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/6.png)

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

#### PATH(yola)Dizin Adresi Ekleme

PATH yolu üzerine ekleyeceğimiz yeni dizini, her oturum başlangıcında sistem tarafından otomatik olarak okunan bir dosya içerisinde belirtmeliyiz ki yaptığımız değişiklik biz silene kadar geçerli olsun. Ekleme işlemi için kullanacağımız dosya, oturum başlangıcında okunan herhangi bir dosya olabilir. Biz örnek olması açısından **/etc** dizini altında yer alan **bash.bashrc** dosyası üzerinden çalışalım. **bash.bashrc**  dosyasında değişiklik yapmak için komut satırına dosyamızın bulunduğu tam konumu belirtmek üzere  nano /etc/bash.bashrc  komutunu giriyoruz.



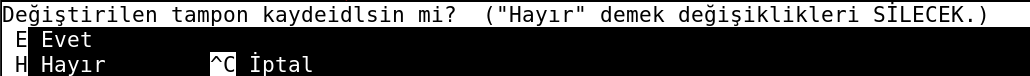
Ve komut satırında karşımıza yukarıdaki gibi  **bash.bashrc**  dosyasının içeriği geliyor. Şimdi yapmamız gereken; programımızın (komut.sh) bulunduğu tam dizin konumunu bu dosyaya uygun şekilde eklemek.

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

PATH="/root/Desktop/yeni\_dizin/yeni/en\_yeni":$PATH



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.

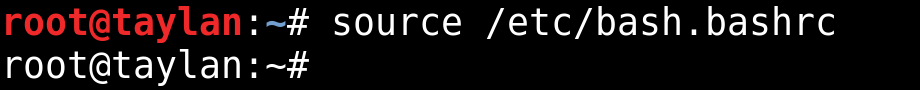


Son olarak bana, "Yazılacak Dosya Adı: /etc/bash.bashrc" diye belirtiyor,  **buradaki ismi değiştirmeden kaydediyoruz**. Eğer ismi değiştirirsek sistem bu dosyayı okuyamayacağı için problem çıkacaktır.

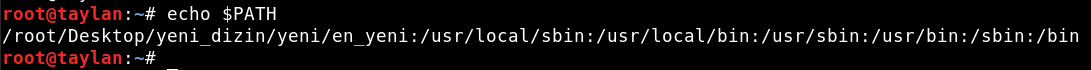


Bu adımları geçtikten sonra istediğimiz dizin  **PATH(yol) ortam değişkenine**  eklenmiş oldu. Ancak yaptığımız değişikliklerin geçerli olabilmesi için, oturumun kapatılıp tekrar açılması ya da source komutu ile değiştirmiş olduğumuz dosyanın tekrar konfigüre edilmesi gerekiyor.

source /etc/bash.bashrc



Neticede yeni dizini PATH yolu üzerine ekleyip, yapılan bu değişikliğin sistem tarafından algınlanmasını sağladık. Şimdi de bu değişikliği teyit etmek üzere konsola echo $PATH komutunu girelim.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/11.png)

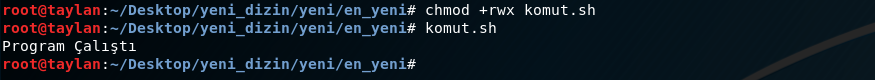
Gördüğünüz gibi dizin, yola(PATH) eklenmiş bulunuyor. Artık dosyayı çalıştırmadan önce son bir ayar daha yapmamız gerek. Daha sonra "**komut.sh**" dosyamızı aynı leafpad programında olduğu gibi istediğimiz zaman komut satırından ismi ile çalıştırabiliyor olacağız.

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.



Gördüğünüz gibi konsol bize "**Erişim engellendi**" diye bildirdi ve dosyayı bulmasına rağmen çalıştıramadı. 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.



Sizler de bu yolla sıklıkla kullandığınız dosyaları kolay erişilebilir kılabilirsiniz. Burada kullanımış olduğumuz komutları ve komutlar ile bağlantılı konuları ileriki bölümlerde zaten tek tek ele alacağız. Sizler şimdilik konunun temelinde ne var ve hangi işlemi neden yaptık ona hakim olsanız yeter. Ancak kafalarda soru işareti ile devam etmemek adına ben yinede tek tek izah edeyim hangi komutu neden kullandık.

nano /etc/bash.bashrc  bu komutta;  nano  komutu, konsol üzerinden dosya içeriğini okumamıza olanak sağlayan bir araçtır. Geri kalan  /etc/bash.bashrc  kısmı ise  **bash.bashrc**  dosyasının bulunduğu konumu belirtiyor. Bu sayede nano aracı ile /etc dizininde bulunan **bash.bashrc**  dosyasını komut satırı üzerinden görebiliyor ve değişiklik yapabiliyoruz.

PATH="/root/Desktop/yeni\_dizin/yeni/en\_yeni:$PATH"  ifadesi ise mevcut  **PATH yoluna yeni dizin ekleme**mize olanak sağlayan bir bütündür. Burada yazdığımız ifade, PATH yolunda yer alan eski dizin adreslerine zarar vermeden yeni dizin ekleme işlemi için gereken ifade bütünüdür.

echo $PATH  komutunu daha önce de görmüştük buradaki  echo  komutu PATH değişken değerini konsola bastırarak  **PATH ortam değişkeni**nde bulunan dizinleri görmemizi sağlıyor.  chmod +rwx komut.sh  komutunda,  chmod  yetkilendirme ayarlamaları yapmamıza olanak sağlıyor.  +rwx  parametrelerinden "**+**" işareti  **ekleme**, "**r**  " işareti  **okuma**, "  **w**  "işareti  **yazma**, "**x**" işareti ise  **çalıştırma**  yetkisi olmak üzere bizlere  **bir bütün olarak dosyayı çalıştırabilmemiz için gereken yetkilendirmeyi**  sağlıyor. Geri kalan  komut.sh  kısım ise zaten yetkilendirme vermek istediğimiz dosyanın adı.

#### PATH Yoluna Dosya Ekleme

PATH yoluna yeni dizin adresi tanımlamadan, komut dosyamızı PATH üzerinde var olan bir konuma taşıyarak da dosyamızın çalıştırılmak üzere bulunabilmesini sağlayabiliriz.

Üstelik bu ikinci yöntem yani var olan PATH yolu üzerine komut dosyamızı taşıma işlemi, ilk işleme oranla çok daha güvenli bir yoldur. Özellikle sunucu yönetiminde erişim yetkilerinin kısıtlanması ve sunucu güvenliğinin sağlanması açısından istisnai durumlar hariç ikinci yöntem tercih edilir. Benim önerim de, yeni bir PATH yolu belirtmek yerine var olan yolları kullanmanızdır. Hemen söylediklerimizi teyit etmek için PATH yolu üzerinde yer alan bir dizine komut dosyamızı taşıyalım ve dosyamızı konsoldan çalıştırmayı deneyelim. Ben örnek olması açısından **/usr/local/sbin** konumuna dosyamı taşıyorum.

Taşımanın ardından komut satırından dosyamı ismi ile çalıştırmayı deniyorum.



Gördüğünüz gibi sistem tarafından tanımlanmış olan PATH yolu üzerinde yer alan herhangi bir dizine taşımış olduğumuz komut dosyamız bulunup çalıştırılabildi. Burada dosyamızın çalıştırma yetkisi bulunduğundan herhangi bir sorun çıkmadı. Ancak ilk kullanımda olduğu gibi çalıştırma yetkisi bulunmayan bir dosyayı PATH yolu üzerindeki bir konuma taşıdığımızda  da çalıştırma yetkisine ihtiyaç duyulur. Yani her iki kullanımda da dosyanın çalıştırılabilirlik yetkisinin bulunması gerektiğini lütfen unutmayın.

Böylelikle iki farklı yöntemle PATH ortam değişkeninin işlevinden nasıl faydalanabileceğimizi net bir biçimde görmüş olduk.

Şimdi bu kısımlar sizlere çok fazla detaya girilmiş belki de gereksiz yere eklenmiş gibi gelebilir ancak bu dokümanı bir bütün olarak kabul etmelisiniz. Dokümantasyon bir sefer okudum bitti değil ihtiyacınız oldukça tekrar tekrar bakasınız diye var. Yani buraları okuyup uygulayıp kavradıktan sonra böyle bir ihtiyacınız olduğunda bu konunun burada olduğunu bilin. Burada yer alan anlatımlar temeli oluşturma üzerine sıralı ve bağlantılı şekilde ilerliyor. Burada bunları ezberlemenize gerek yok zaten öğrenmenin temelinde de ezber yok. O yüzden rahat olun her şey kullandıkça gelişecek ancak dediğim gibi artık böyle bir şeyin varlığından haberdarsınız yeri geldiğinde ihtiyacınız olan bilgiler burada olacak.

#### Ortam Değişkenlerini Görüntülemek

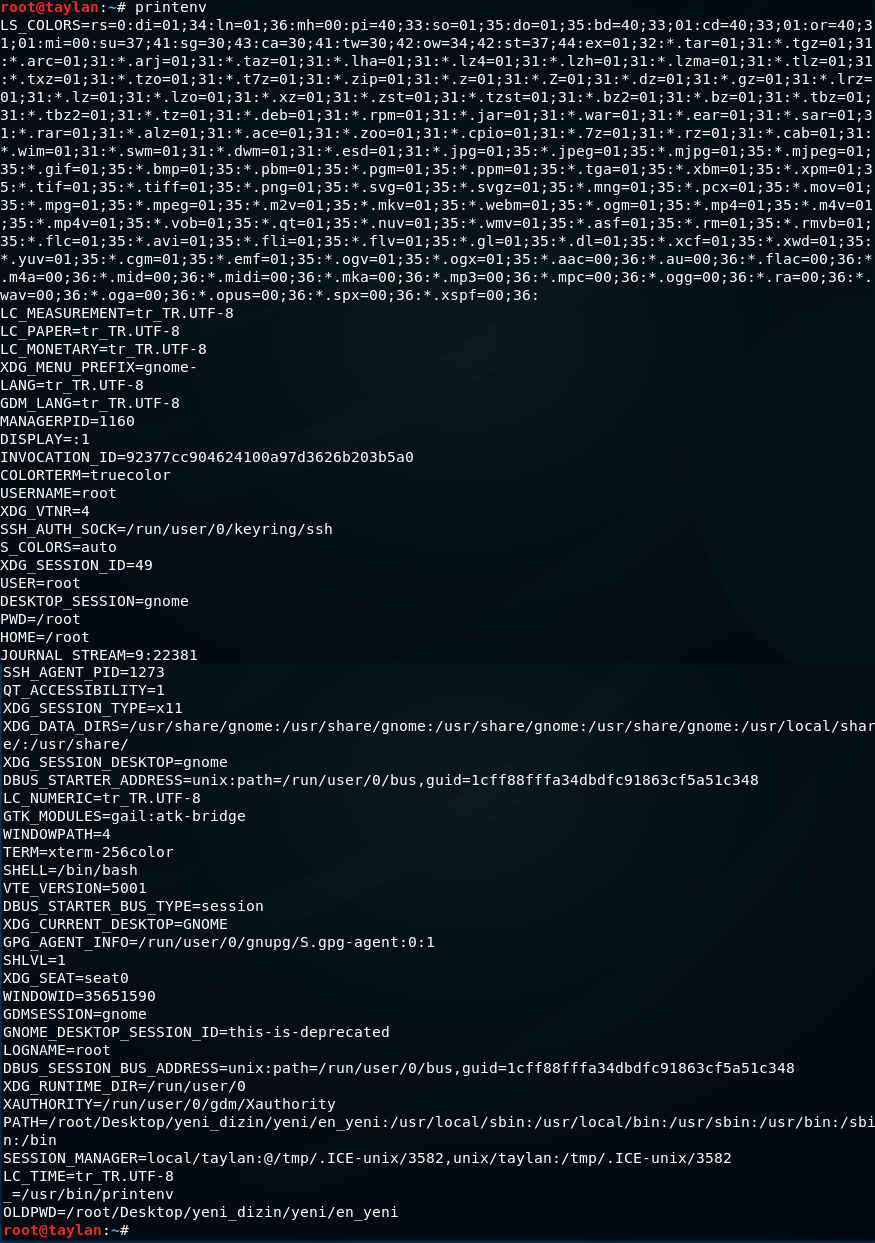
Ortam değişkenleri dedik ancak şu ana kadar sadece **PATH** değişkeninden bahsettik. Tabii ki sistemde sadece PATH değişkeni bulunmuyor bu değişken dışında da pek çok ortam değişkeni mevcut. Bunları görmek istersek komut satırına set , env ya da printenv komutlarından birini ihtiyacımıza göre kullanabiliriz. Bu üç komut arasındaki farklar aşağıdaki şekildedir.

**set :** Shell'e ait olan değişkenlerin adını ve değerlerini verir.

**printenv :** Sistemde bulunan bütün çevresel değişkenleri verir.

**env :** Sistemde bulunan bütün "export" edilmiş yani değiştirilmiş değişkenlerle beraber değişkenleri gösterir.

Ben aşağıda sadece printenv komutunun çıktısını veriyorum. Sizler diğer komutları da mutlaka uygulayarak aralarındaki farkları kendi gözlerinizle görün.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/14.png)

Evet bize burada pek çok ortam değişkeni sıralandı. Ben her birini açıklamayacağım. Neden diye soracak olursanız; mevcut dokümantasyonda temel anlatım içerisinde olduğumuz için, sizlere mümkün olduğunca ilk adım için gerekli olan bilgileri vermeye çalışıyorum. Yani benim burada anlattığım her bilgi sizlere konuyu açıklamak ve konuyu daha iyi şekilde kavratmak için var. İlk aşama için gereksiz, yorucu bilgilere ihtiyacımız yok. Ancak buradan açıkladıklarım haricindekiler gereksiz gibi bir anlam çıkmasın lütfen. Zaten temeli yani asıl işlevlerini kavradıktan sonra yeri geldikçe diğer değişkenleri de biliyor ve kullanıyor olacaksınız. Ancak ben yinede şimdi diğer bir değişken olan **HOME** değişkenini de kısaca açıklayarak örneklendirmiş olacağım. Merak eden arkadaşlar şimdiden diğer ortam değişkenlerini de araştırıp kurcalayabilir.

**HOME** değişkenini örnekleyerek açıklayacak olursak:

Şöyle bir durum düşünün; bir programcı, yazmış olduğu programının kurulduğu zaman masaüstüne bir kısayol oluşturmasını istiyor olsun. Ancak böyle bir şey yapabilmek için, kullanıcıların masaüstüne giden yolunun tespit edilebilmesi lazım. Ama şöyle bir problem var; Bildiğiniz gibi herkesin masaüstüne giden yol aynı değil. Bir bilgisayardaki masaüstü dizinini bulabilmek için, o bilgisayarı kullanan kişinin kullanıcı adını da biliyor olmanız lazım. Çünkü masaüstünün bulunduğu dizin kullanıcı adına bağlı olarak değişiklik gösterecektir.

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 kavrıyoruz. 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 dizininin yol bilgisini tutan değişken** olan **HOME** değişkenini cd $HOME şeklide girmemiz yeterli oluyor. Yani bizler bu değişken sayede, hiç aramamıza gerek kalmadan ev dizini içerisinde yer alan Desktop konumuna da ulaşabiliyor olacağız.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/15.png)

Ayrıca burada ekstra bir bilgi olsun diye 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 kombinaysonu ile oluşturuluyor. )

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/16.png) [](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/17.png)

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. Yani gördüğünüz gibi 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 dizini**ni 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 değişkenlerden birkaçını da kısaca açıklayacak olursak:

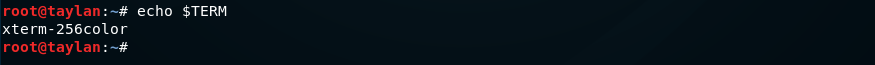
**SHELL:** Çalışmakta olan kabuk programının adını ve yerini verir.

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

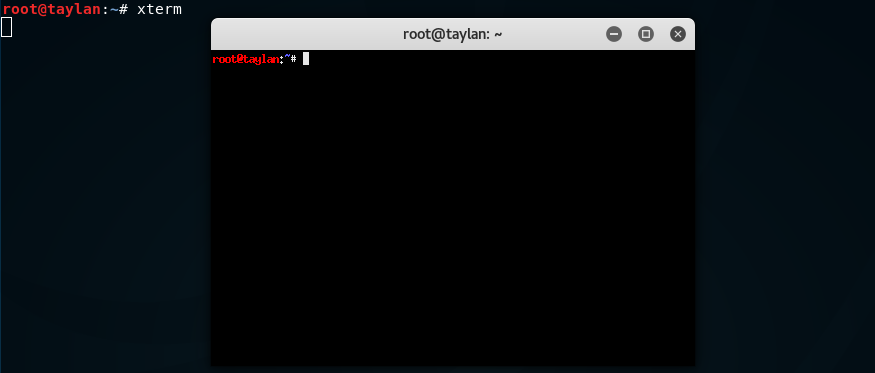
**HOME:** Kullanıncının ev dizinini içeren değişken.

**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.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/18.png)

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 xterm komutunu verdiğimizde xterm açılacak ve çıktı teyit edilmiş olacaktır.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/19.png)

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 açık olan mevcut 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.

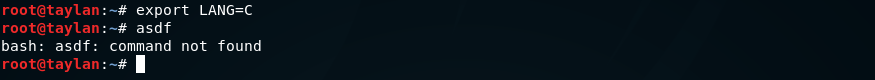
#### Mevcut Konsola Özel

**Sadece o an kullanmakta 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 "(anlamsız karşılığı olmayan bir ifade/komut) yazdım. Konsol yanıt olarak " bash: asdf: komut yok " yazısını ekrana bastı.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/20.png)

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.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/21.png)

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 ingilizceye çevrilmiş oldu.

Bütün bu işlemler sadece bu komut penceresi için sınırlı. Yani açmış olduğumuz terminal ekranını(konsol/komut penceresi) 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.

Bulunduğumuz 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.

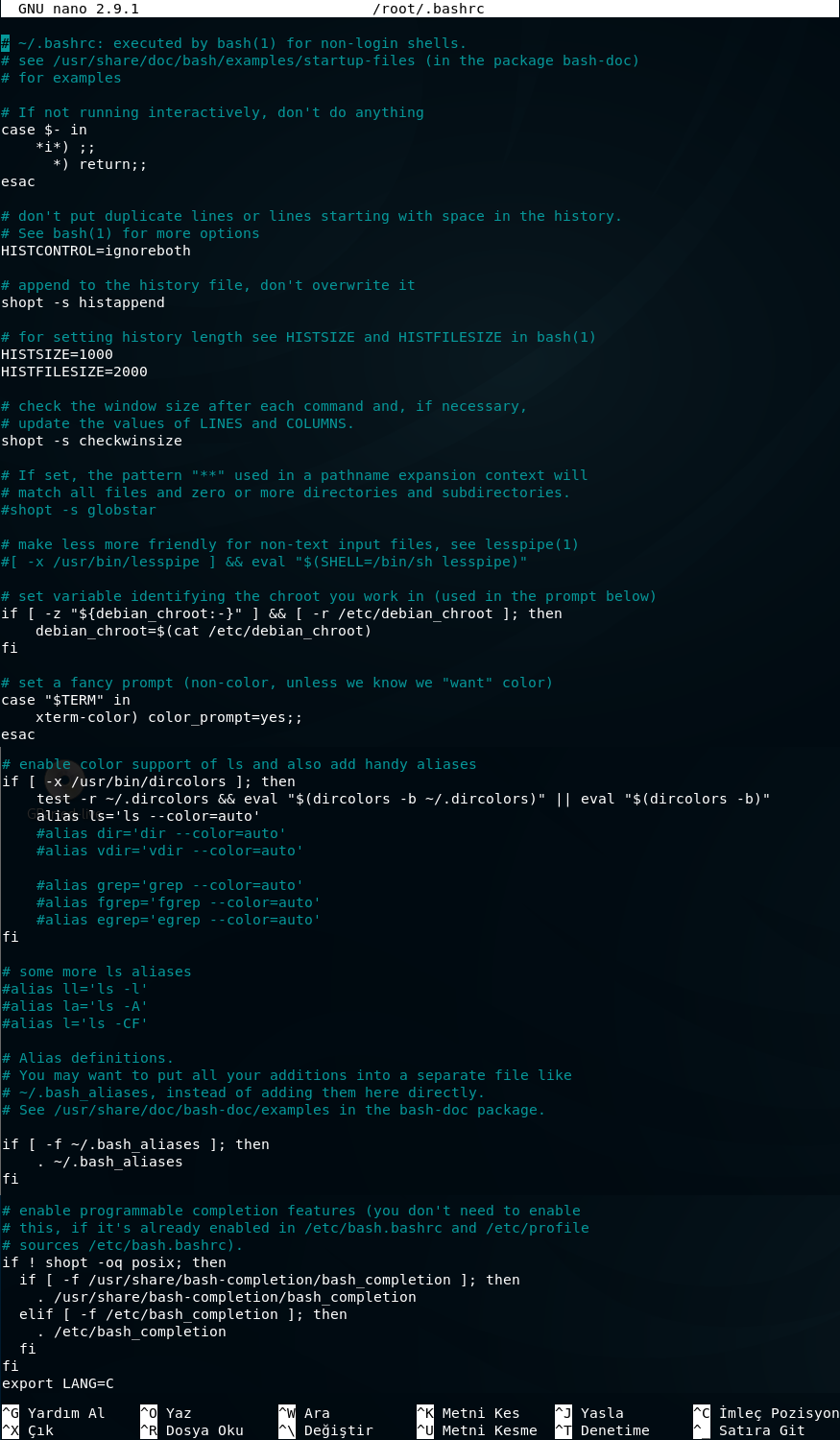
Kodu kısaca açıklayacak olursak leafpad sistemde mevcut bulunan basit metin düzenleyicisinin adıdır, ~(**Alt Gr + ü** kombinasyonu ile oluşturulan "**tilde**" karakteri) karakteri ev dizinini temsil ediyor /.bashrc ise düzenleme yapacağımız dosyanın adıdır. Bu kısa açıklama sonrası komutları tam olarak 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(Türkçe olan dili ingilizceye çevirmek) için gerekli olan yazı dizisini export LANG=C 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. Komutumuzu girdikten sonra karşımıza konsol ekranı içerisinde .bashrc dosyasının içeriği geliyor. Klavyedeki yön tuşlarını kullanarak en alt satıra iniyoruz ve oraya yapmak istediğimiz değişikliği giriyoruz. Ben dilde değişiklik yapacağım için satırın sonuna export LANG=C şeklinde ekleme yaptım .

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/23.png)

Ve **Ctrl + X** tuş kombinasyonunu uyguladım. Daha sonra konsol bana çıkmadan önce dosyada yaptığım değişikliği kaydetmek isteyip istemediğimi sordu bende "**e**" diyerek **dosyanın adını herhangi bir değişime uğratmadan kaydettim** ve işte bu kadar işlem tamamdır.

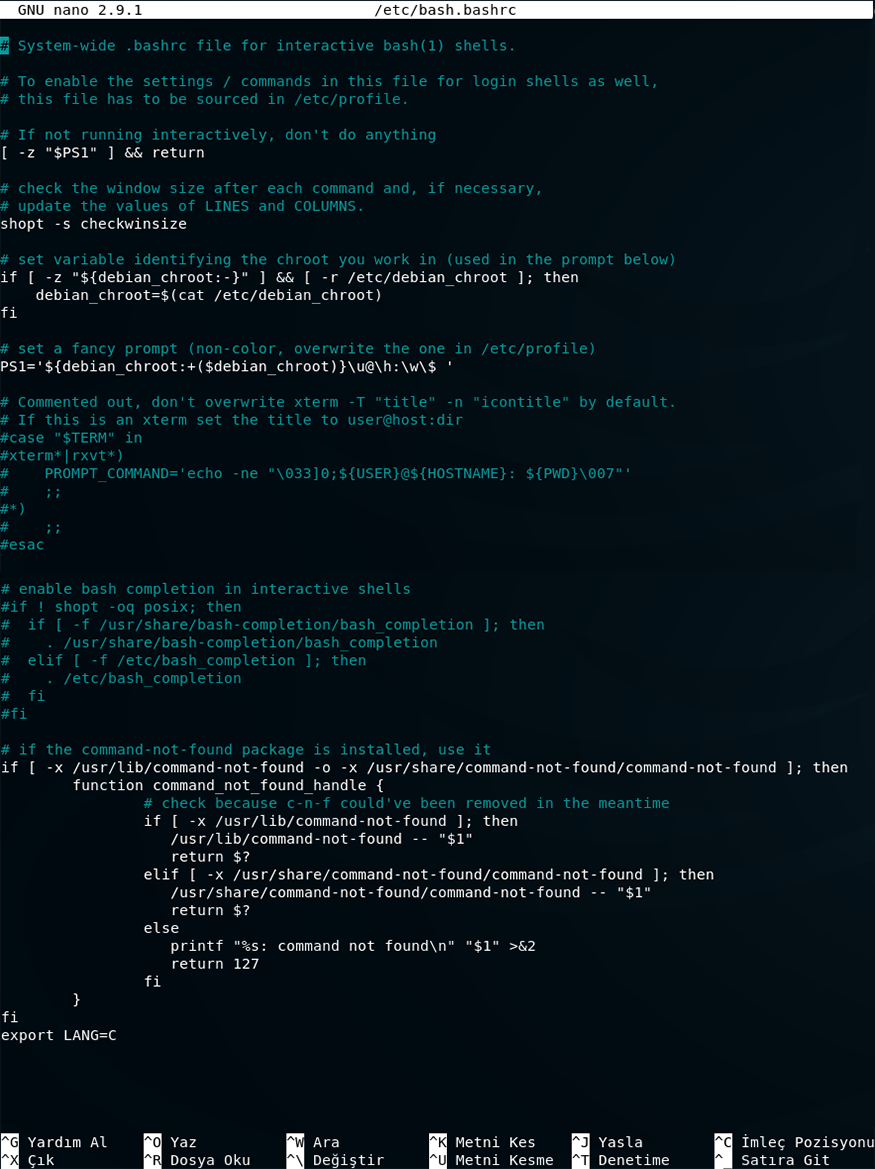
[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/24.png) [](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/25.png)

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 açılıyormuşçasına .bashrc dosyasının tekrar okunup yapılan değişikliklerin geçerli olmasını sağlar. Ancak benim önerim oturumu kapatıp tekrar açmanız yönündedir.

#### Sistem Genelinde

Eğer yaptığımız değişiklik bütün kullanıcı oturumlarında aynı şekilde geçerli olsun istiyorsak değişkenin sistemde her oturum açıldığında okunan bir dosyada bulunması gerekmektedir. İşte bizim de yapacağımız değişiklikler 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.

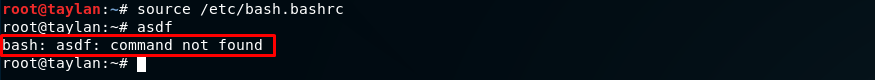
[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/26.png)

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

İlgili ifadeyi bash.bashrc dosyamıza ekleyip dosyamızı kaydediyoruz .

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/27.png) [](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/28.png)

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.

[](https://raw.githubusercontent.com/taylanbildik/Linux_Dersleri/master/img/1-%20Komut%20Satırı/29.png)

Çıktımız ingilizce 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\_adı şeklindeki komut bütünü ile yeniden konfigüre edin, bütün değişimler düzelmiş olacaktır.(source komutunu kullanmak zorunda değilsiniz, hatta bu komut yerine sistemi yeniden başlatmanız daha iyi olacaktır.)

**Hatırlatma**

Bu dokümantasyon ile ilgili her türlü hata, yazım yanlışı ve sorun için bana özel mesaj alanından geri bildirimde bulunabilirsiniz. Hataların mümkün olan en kısa sürede düzeltileceğinden emin olabilirsiniz.