ÖN BİLGİ

Bu belge

- https://www.syslogs.org/docs/kabuk_programlama.doc

resmi adresindeki veya linkin kırık olması ihtimaline karşın alternatif olarak

- https://www.includekarabuk.com/kitaplik/indirmeDeposu/Siber_Guvenlik_Teknik_Makaleler/Teori/BaskalarinaAitMakaleler/Syslogs.org_18_kabuk_programlama.pdf

adresindeki makaleye çalışılarak elde edilen notlarımı kapsamaktadır. Bu çıkarılan notlar belgemde alıntılar ve/veya kişisel ilavelerim mevcuttur.

1)

Her kabuğun kendine özgü programlama dili yapısı vardır. Bash kabuğu ise güçlü programlama özellikleriyle karmaşık programların rahatça yazılmasına izin verir. Mantıksal operatörler, döngüler, değişkenler ve modern programlama dillerinde bulunan pek çok özellik bash kabuğunda da vardır ve işleyiş tarzları da hemen hemen aynıdır.

Bash'in en büyük dezavantajı derlenerek çalıştırılan dillere göre (C, C++ gibi) daha yavaş olması, sistem kaynaklarını biraz daha fazla tüketmesidir.

(Page 2)

2)

Bir kabuk altında çalışırken başka bir kabuk için yazılmış bir programı çalıştırmak mümkündür. Örneğin tcsh altındasınız ve daha evvel bash kullanarak yazdığınız bir programı çalıştırmak istiyorsunuz. Önce bash yazarak kabuk değiştirmeli, ardından programı çalıştırmalı, ve tekrar tcsh'a dönmelisiniz. Tüm bunları otomatik olarak yaptırabilirsiniz. Programın en başına #! karakterini, ardından programın çalışacağı kabuğun patikasını yazın. Örneğin #!/bin/bash komutunu programın en üstüne eklerseniz bu program bash kabuğu altında çalışacaktır.

(Page 3)

3)

Her if, bir fi komutuyla bitmelidir. Aşağıda if-then-else komutunun örnek sözdizimi görülüyor.

```
#!/bin/bash
echo "0 ile 20 arasinda bir sayi secin"
read sec
if [ $sec -lt 10 ]
        then
                echo "Secilen sayi tek basamakli"
        else
                echo "Secilen sayi cift basamakli"
fi
```

Her if komutu bir fi ile son bulur.

(Page 6)

4)

Her case yapısı bir esac ile bitmelidir.

```
#!/bin/bash

clear

echo "1. ekrani temizle"
echo "2. sistemdekileri goruntule"
echo "3. dizindeki dosyalari goster"
echo -n "Secenegi giriniz : "

read secenek


case $secenek in
        1)
                clear
                ;;
        2)
                w
                ;;
        3)
                ls -al
                ;;
        *)
                echo Hatali secenek
esac
```

(page 7-8)

5)

Döngüler

Diğer hemen tüm programlama dillerinin en büyük gücü olan döngü işlemlerine kabuk altında da izin veriliyor. Burada programcı tarafından en çok kullanılan iki döngü tipi anlatılacaktır: while ve for.

while komutu her döngüde bir denetleme mekanizmasını harekete geçirirken for döngüsü bir listenin elemanlarını sırayla seçer.

while-do Döngüsü

```
#!/bin/bash

deger=0

while [ $deger -lt 100 ]
        do
                deger=$((deger+1))
                echo $deger
        done
```

Yukarıda kullanılan ((ve)) karakterleri arasına matematiksel bir işlem getirilebilir. Bu özellik bash kabuğuna özgüdür.

for-do Döngüsü

Bir liste dahilindeki tüm değerlere sırayla erişimi sağlar. for komutundan sonra yeralan liste sırayla kullanılır ve herbirisi için döngü çalıştırılır. Listenin sonuna gelindiğinde ise döngüden çıkılır.

```
for agac in akasya elma visne
        do
                echo $agac
        done
```

Yukarıdaki örnekte for döngüsü kullanılarak ekrana bir dizi kelime yazdırılıyor. Döngü boyunca akasya, elma ve visne kelimeleri "agac" değişkenine kopyalanıyor ve her döngüde bu değişkenin içerdiği bilgi ekrana yazdırılıyor.

for-do döngüsü, dosya isimleri üzerinde yapılan işlemlerde de büyük kolaylıklar sağlar. Bunun için özel karakterlerden yararlanmak da olasıdır. Örnek olarak * karakteri o an ki çalışma dizini içindeki tüm dosyaları seçer. For döngüsü ile de her birinin ismi ekrana basılır.

```
for a in *
        do
                file $a
        done
```