**Fırat Üniversitesi**
**BMÜ-431**
**Bilgisayar Mühendisliği** **Bilgisayar Sistemleri Laboratuvarı**

**DENEY NO: 1**

**LINUX İŞLETİM SİSTEMİ VE BULUT BİLİŞİM**

## 1. GİRİŞ

Linux, açık kaynak kodlu ve ücretsiz bir işletim sistemi çekirdeğidir. Unix'ten esinlenerek geliştirildiği için Unix-like yani Unix-benzeri denilmektedir. Linux kerneli üzerine çeşitli yazılımlar eklenerek oluşturulan işletim sistemi Linux dağıtımı (distrosu) olarak adlandırılır. Örneğin bu deneyde kullanılacak olan işletim sistemi olan Ubuntu bir Linux dağıtımıdır. Kavramların daha net anlaşılması açısından açıklamak gerekirse;

- Linux ve Unix farklı kavramlardır. Linux, Unix temel alınarak geliştirilmiştir.
- Linux bir işletim sistemi değildir, işletim sistemi çekirdeğidir. Bu çekirdek tamamlanarak işletim sistemi yani Linux dağıtımı ortaya çıkar.
- Ubuntu, Pardus, Debian, Slackware, SUSE, Red Hat, Fedora yaygın olarak kullanılan bazı Linux dağıtımlarıdır.

Günümüzde en yaygın olarak kullanılan işletim sistemi Microsoft Windows'tur. Peki, Windows işletim sistemi bu kadar popülerken Sistem Laboratuvarı dersinde Linux'un tercih edilme nedenleri nelerdir?

Linux ağ kavramlarının öğrenilmesi açısından daha elverişlidir:

- ✓ Ip adresi değiştirilmesi, yönlendirme (routing) tablolarının oluşturulması, ağ bağlantısı testleri ve ağ trafiği (gelen/giden paketlerin) incelenmesi gibi işlemler için basit ve güçlü bir ortam sağlar.
- ✓ Server/client uygulamalarının oluşturulması ve çalıştırılması oldukça basittir.
- ✓ Bir Linux bilgisayar kolay bir şekilde router olarak konfigüre edilebilir.

Unix-tabanlı işletim sistemlerinin tecrübe edilmesi önemlidir. Windows en çok kullanılan işletim sistemi olsa da Unix tabanlı işletim sistemleri de ağ sunucuları, ağ cihazları ve gömülü sistemlerde oldukça yaygın olarak kullanılmaktadır. Örneğin bir çok router, switch, özelleştirilmiş bilgisayar Linux kullanmaktadır.

Ubuntu Linux işletim sistemi ve üzerinde kullanacağımız yazılımlar ücretsizdir. Hiç biri korsan yazılım değildir.

Günümüzde gittikçe yaygınlaşan Bulut Bilişim ve Büyük Veri teknolojilerinde Linux sistemler tercih edilmektedir.

## Linux ortamı ve virüsler:

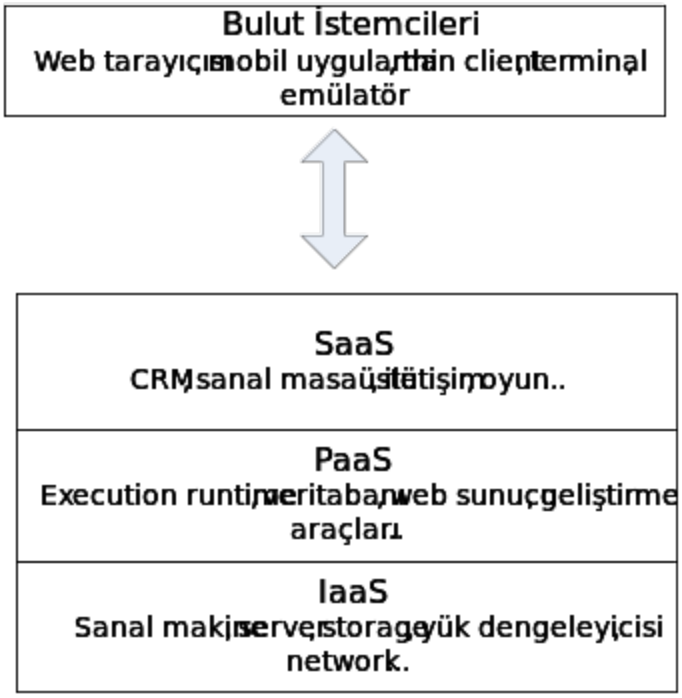
Unix / Linux sistem açıklarından faydalanan virüsler laboratuvar ortamında yazılmıştır. Laboratuvar ortamı diyoruz çünkü bu virüslerin hiçbiri ortalıkta dolaşmıyor, sadece Unix sistemler için virüs olabileceğine dair ispat olsun diye varlar. Son dönemde Linux çekirdeğindeki açıkları kullanan kök takımı (rootkit)'ler de yazıldı ancak bu açıklar da hızla kapatıldı. Linux ortamında çalışacak ve kendisini çoğaltabilecek virüs yazmak teorik olarak çok zor olsa da, olanaksız değil.

- Linux anti-virüs programlarının büyük çoğunluğu e-posta sunucularında çalışır. Bunlar e-posta alıp-göndermek istediğinizde istemcinizin bağlandığı bilgisayarlardır. E-posta, virüs ve truva atlarının yayılmasında kullanılan en temel yol olduğundan, bu sunucular bilgisayar virüsleri ile olan savaştaki ön cephe gibidir. Bu sunucuların bir çoğu Linux kullandığından, Windows virüslerini tespit etmek için bir Linux anti-virüs programının kullanılma ihtiyacı daha iyi anlaşılır.
- Yetkilendirme sistemi sayesinde, sisteme bir virüs girse dahi kullanıcı ev dizini dışına ya da farklı bilgisayarlara yayılamamaktadır.
- Linux'ta kullanıcıların, kurmak istedikleri programları, herhangi bir web sitesi üzerinden değil doğrudan dağıtımın resmi paket depoları üzerinden (Ubuntu Yazılım Merkezi gibi) temin ediyor olmaları, bu güvenli yapıyı desteklemektedir.
- Daha çok göz daha az güvenlik açığı demektir. Linux açık kaynaklı bir yazılımdır, bu da dünyadaki her programcının kodlara bakmasına ve yardımcı olmasına, ya da geliştiricilere "... Bu bir güvenlik açığı değil midir?" diyebilmesine, böylelikle hataların hızlıca kapatılmasına olanak sağlar. Linus Torvalds bu durumu “Hiçbir hata (açık) milyonlarca gözden kaçamaz” sözü ile ifade etmiştir.
- Pek çok Windows kullanıcısında "Windows çok kullanılıyor onun için bu kadar çok virüs yazılmakta. Bir gün Linux da yaygınlaşırsa onun için de virüsler yazılacaktır." şeklinde bir algı vardır. Ancak örneğin açık kaynak Apache Web Sunucusu yazılımı, dünyada en yüksek market payına sahiptir ve buna rağmen Microsoft'un sunucusundan çok daha az saldırıya uğramaktadır.

## Bulut Bilişim

İnternetin bulutlardan oluşan gökyüzüne benzemesinden yola çıkarak bulut ismini alan bulut bilişim için oldukça çeşitli tanımlar bulunmaktadır. Bulut internet demektir. Ağ diyagramlarında interneti göstermek için bulut resmi kullanılır. İnternet üzerinden birbirine bağlanmış, birlikte çalışan, dinamik olarak yönetilebilen, izlenebilen ve bakımı yapılabilen sanal sunucuların oluşturduğu sistemin adıdır.

Bulut bilişim sağlayıcıları farklı birkaç temel model üzerinden servis sunmaktadır. Bu modeller sunulan hizmete İngilizcede “servis olarak” anlamına gelen “as a service” ön eki getirilerek adlandırılmışlardır. Uygulanan 3 temel model Servis Olarak Altyapı (IaaS), Servis Olarak Platform (PaaS) ve Servis Olarak Yazılım (SaaS)’dır.




**Şekil 1-** Bulut Hizmet Modelleri

## 2. UNİX KULLANICI HESAPLARI VE DOSYA HİYERARŞİSİ

Ubuntu ilk başlatıldığında kullanıcı giriş ekranı açılır. Size laboratuvarda verilecek olan kullanıcı adı ve şifreyi kullanarak giriş yapınız.

Ubuntu'da farklı kullanıcıların farklı yetkileri bulunmaktadır. Bu yetkiler sistem dosyalarının, diğer kullanıcıların dosyalarının görüntülenmesi veya değiştirilmesi, işletim sistemindeki önemli parametrelerin değiştirilmesi gibi işlemler için geçerlidir. En yetkili (tüm izinlere sahip) olan kullanıcı root'dur. Root kullanıcısına super-user da denilmektedir. Root kullanıcı olarak giriş yapılması doğru değildir çünki yanlışlıkla sabit diskteki bütün bilgiler silinebilir ve geri dönüşü olmayan hatalar yapılabilir. Bu nedenle root yetkisi gerektiren işlemler "sudo" komutu ile yürütülerek sadece o komut için super-user yetkisiyle işlem yapılması sağlanır.

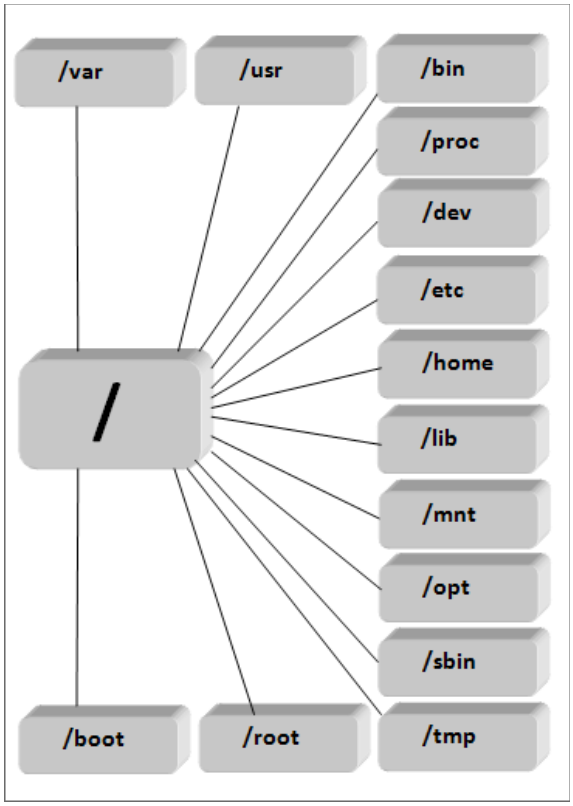
Linux dosya sistemleri ext2, ext3 veya ext4 olabilir. Bu dosya sistemleri sırasıyla birbirinin daha gelişmiş versiyonlarıdır. Ext3 ve ext4'ün günlükleme desteği (journaling) vardır. Amaç sistemin çöktüğü durumlarda daha önce yapılan işlemler log dosyalarından okunarak verilerin kurtarılmasını sağlamaktır. Disk boyutlarının çok fazla büyümesi üzerinde daha büyük büyüklükte dosyaların saklanabilmesi amacıyla ext4 geliştirilmiştir.

İşletim sistemlerinin tümünde dizin veya klasör yapısı bulunmaktadır. Bilgisayar çeşitli bilgileri depolayıp işleyebilmekte ve birden fazla kullanıcı tarafından kullanılabilmektedir. Bu da çok sayıda dosya ile uğraşmak demektir. Bu karışıklığın engellenmesi için dosya/dizin yapısı geliştirilmiştir.

Linux, bir UNIX klonu olması itibariyle "Tekil Hiyerarşik Klasör Yapısı"na sahiptir. Windows'ta olduğu gibi C, D, E bölümleri bulunmaz. Tüm klasörlerin üzerinde kök dizini olan root klasörü bulunur ve kökten dallara doğru uzanan ağaç yapısındadır. Root olarak isimlendiren bu başlangıç dizini / (slash) simgesi ile gösterilir ve

kendisine bağlı diğer tüm dizinleri barındırmaktadır. Dosya sistemi bir kök dizini ve kök dizinin çocukları şeklinde devam eden bir hiyerarşi şeklindedir.

Not: Linux'da yönetici (Windows'ta administrator) kullanıcı adı olan root ile kök dizini olan root birbirinden farklıdır.




**Şekil 2-** Unix Dizin Hiyerarşisi

Şekil -2'de kök dizini altında bulunan klasörler gösterilmiştir. / ( Root) başlangıç dizininin sonrasında bu klasörler yer alır. Windows'ta bir program kurulduğunda genellikle programa ait dosyalar tek bir dosya içerisinde bulunur. Örneğin Adobe programı C:\Program Files\Adobe dizininde yar alır. Linux'da ise bu durum farklıdır. Örneğin programın info dosyaları /usr/share/info dizininde yer alırken programa ait yardım bilgisi ve dökümantasyon dosyaları /usr/share/doc/program_ismi dizinine yerleştirilir. Ubuntu'da bu şekilde gelenekselleşmiş bir dosya organizasyonu bulunmaktadır. Bunun amacı çok kullanıcılı ve dağıtık bir sistemi güvenli ve kolay bir şekilde kurup yönetebilmektir.

cat, mkdir, cp, ls, mv, rm gibi birçok komut /bin içerisindedir.

Sisteme ilk login yapıldığında /home dizininden başlanır. Bu klasörün içerisinde her kullanıcının kendi adında bir alt klasörü bulunur. /home klasörünü, Windows'taki Belgelerim klasörüne benzetebiliriz. Ama ondan daha güvenli bir yapıdır. Çünkü Linux'ta bir başkasının ev klasörüne müdahale edemezken, Windows'ta çok zorlanmadan istediğinizi yapabilirsiniz. Home dizinine ~ ismiyle geçilebilir. Farklı bir kullanıcının \home dizini altındaki alanına geçmek için ~ ile birlikte kullanıcı adı yazılır.

Not: Diğer dizinlerin özelliklerini kaynaklar bölümünden yararlanarak inceleyiniz.

**Gerçek/Göreceli Dizin:**

Bir dosyanın gerçek dizin root kök dizininden başlayarak dosyaya giden yoldur. Örnek gerçek dizinler:

```
/etc/passwd
/home/student/NetBeansProjects
/dev/rdsk/Os3
```

Bir dosyanın yolu göreceli de olabilir. / ile başlamaz. O an içinde bulunan dizinden gidilen yoldur.

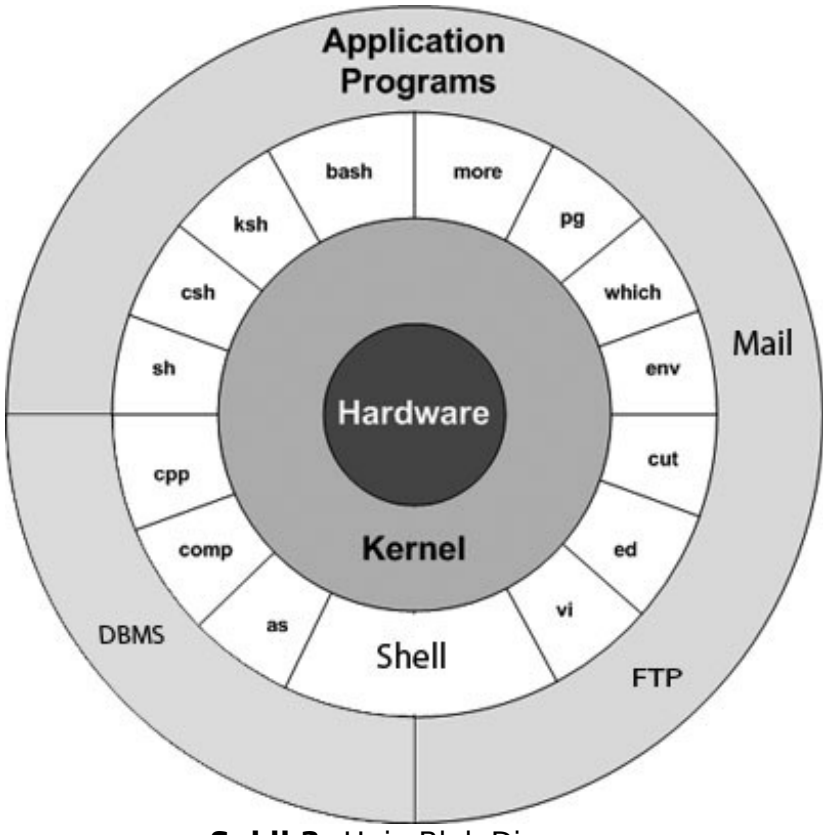
```
Belgelerim/ilk_donem
sislab/notlar/deney1
```

Dosya sistemi hiyerarşisinde nerede olduğunuzu görüntülemek için **pwd** komutu kullanılır.

```
$pwd
/home/student/hadoop/logs
$
```

## 3. TEMEL KOMUTLAR

Şekil -3 'de bir UNIX sisteminin temel blok diyagramı gösterilmiştir. UNIX sistemlerin temel bileşenleri şunlardır:




**Şekil 3-** Unix Blok Diyagramı

**Kernel:** Kernel işletim sisteminin kalbidir. Donanımla ve hafıza yönetimi, task scheduling, dosya yönetimi gibi bir çok işle etkileşim kurar.

**Shell:** Terminalde bir komut yazıldığında Shell komutu yorumlar ve istenen programı çağırır. Shell, bütün komutlar için standart bir sytax kullanır. C Shell, Bourne Shell ve Korn Shell bir çok Unix türevlerinde var olan, bilinen "Shell"lerdir.

**Komutlar ve Hizmetler: cp, mv, cat** ve **grep** gibi birçok komut ve hizmet bulunmaktadır. 250'nin üzerinde standart komut ve bunun yanında üçüncü partilerce sağlanan sayısız yazılım mevcuttur. Komutlar genellikle çok sayıda opsiyona sahiptir.

**Dosyalar ve Dizinler:** UNIX üzerindeki bütün veri dosya şeklinde organize edilir. Bütün dosyalar da dizinlerle organize edilir. Daha önce de belirtildiği üzere bu dizinler dosya sistemi denilen ağaç-benzeri(tree-like) bir yapı oluşturur.

***Sistemin Başlatılması***

Bir UNIX işletim sistemi yüklü bilgisayarın güç düğmesine basıldığında sistem başlatılır ve kullanıcı giriş ekranı açılır.

```
login:
```

Kullanıcı adı girilir ENTER'a basılır ve şifre girilir ENTER'a basılır. Kullanıcı adı ve şifre case-sensitive'dir.

```
login : student
student's password:
Last login: Sun Jun 14 09:32:32 2014 from 62.61.164.73
$
```

Giriş yapılınca $ işaretiyle başlayan bir komut giriş moduna geçilir. Örneğin ajandayı göstermeye yarayan cal komutu yazarsak aşağıdaki ekran çıkar:

```
$ cal

student@student-UnixLab:~$ cal
     Ekim 2014
Pa Pz Sa Çr Pr Cu Ct
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

***Şifre değiştirme:***

Komut satırına **passwd** yazılır.

Mecvut şifre girilir ve enter'a basılır.

Yeni şifre girilir

Yeni şifre tekrar girilir.

```
$ passwd
Changing password for student
(current) Unix password:******
New UNIX password:*******
```

```
Retype new UNIX password:*******
passwd: all authentication tokens updated  successfully
```

***Dizinlerin ve dosyaların listelenmesi :***

Bir dizindeki tüm dosya ve klasörleri **ls** komutuyla listeleyebiliriz. Aşağıda **ls** komutu **-l** opsiyonuyla kullanılmıştır.

```
$ ls -l
total 19621
drwxrwxr-x  2 amrood amrood      4096 Dec 25 09:59 uml
-rw-rw-r--  1 amrood amrood      5341 Dec 25 08:38 uml.jpg
drwxr-xr-x  2 amrood amrood      4096 Feb 15  2006 univ
drwxr-xr-x  2 root   root        4096 Dec  9  2007 urlspedia
-rw-r--r--  1 root   root      276480 Dec  9  2007 urlspedia.tar
drwxr-xr-x  8 root   root        4096 Nov 25  2007 usr
-rwxr-xr-x  1 root   root        3192 Nov 25  2007 webthumb.php
-rw-rw-r--  1 amrood amrood     20480 Nov 25  2007 webthumb.tar
-rw-rw-r--  1 amrood amrood      5654 Aug  9  2007 yourfile.mid
-rw-rw-r--  1 amrood amrood    166255 Aug  9  2007 yourfile.swf

$
```

Burada **d.....** ile başlayan kayıtlar dizin olduğunu gösterir. Burada uml, univ, urlspedia ve usr dizin, diğerleri dosyadır.

***Hangi kullanıcı?***

Kim olduğunuz sorusu yani **whoami** komutu ile kullanıcı adı gösterilir.

```
$ whoami
 student

$
```

***Hangi kullanıcılar bağlanmış?***

Sisteme aynı anda hangi kullanıcıların bağlanmış olduğu görüntülenebilir. Bu bilgiyi göstermek için **users, who** ve **w** gibi komutlar kullanılabilir.

```
$ users
 student admin

$ who
student ttyp0 Oct 8 14:10 (limbo)
admin   ttyp2 Oct 4 09:08 (calliope)

$
```

**w** komutunu kullanarak sonucu inceleyiniz. Bu komut kullanılarak bağlanmış kullanıcılar hakkında daha detaylı bilgi gösterilir.

***Çıkış yapmak:***

Oturumunuz bittiğinde sistemden çıkış yapılarak başkalarının dosylarımıza erişmesini engellememiz gerekir. Bunun için **logout** komutu kullanılır. Sistem herşeyi temizler ve bağlantıyı keser.

Kapatma (Shutdown) komutları: **halt, init 0,  poweroff, , shutdown**

Yeniden başlatma komutları : **init 6, reboot**

Bu komutları kullanabilmek için superuser ya da root olmak gerekir.

***Dosyaların listelenmesi:***

Bir dizindeki tüm dosya ve klasörleri **ls** komutuyla listeleyebiliriz. Aşağıda **ls** komutu **-l** opsiyonuyla kullanılmıştır.

```
$ls
```

Aşağıda örnek bir sonuç görüntülenmiştir

```
$ls

bin         hosts  lib     res.03
ch07        hw1    pub     test_results
ch07.bak    hw2    res.01  users
docs        hw3    res.02  work
```

Aşağıda ls komutu -l opsiyonuyla daha fazla bilgi verir.

```
$ls -l
total 148

drwxr-xr-x  2 student student  4096 Eki 13 10:06 Desktop
drwxr-xr-x  2 student student  4096 Eyl 17 19:47 Documents
drwxr-xr-x  2 student student  4096 Eki 13 10:06 Downloads
drwxrwxr-x  2 student student  4096 Eki 11 16:33 examples
-rw-r--r--  1 student student  8980 Eyl 17 19:35 examples.desktop
-rw-rw-r--  1 student student 77654 Eki 12 16:05 foy2.odt
drwxrwxr-x 12 student student  4096 Eyl 18 17:21 hadoop
-rw-rw-r--  1 student student   457 Eki 12 16:32 HelloWorld.class
-rw-rw-r--  1 student student   174 Eki 12 16:32 HelloWorld.java
drwxrwxr-x  8 student student  4096 Eyl 18 11:06 jdk1.8.0_20
drwxr-xr-x  2 student student  4096 Eyl 17 19:47 Music
drwxrwxr-x 14 student student  4096 Eki 12 16:29 netbeans-8.0.1
...
$
```

Burada **d** ile başlayan kayıtlar dizin olduğunu gösterir. Örneğin burada Desktop, Documents, Downloads … dizindir. foy, HelloWorld ise dosyadır.

İlk kolon dosya (veya dizinin) türünü ve izinlerini gösterir. İkinci kolon kaç memory-block olduğunu gösterir. Üçüncü kolon dosyanın sahibini yani bu dosyayı oluşturan Unix kullanıcısını gösterir ve dördüncü kolon bu kullanıcının ait olduğu kullanıcı grubunu gösterir. Bütün Unix kullanıcılarının bağlı olduğu bir grup vardır. Beşinci kolon byte olarak dosya büyüklüğünü gösterir. Altıncı kolon dosyanın oluşturulduğu tarih ve saati gösterir. Yedinci kolon dosya adını gösterir.

***Meta karakter:***

Unix 'te * ve ? Karakterlerinin özel anlamları vardır. Bu karakterlere meta-karakter denir. Sıfır yada daha fazla karakter için * meta karakteri kullanılır. Tek bir karakter eşleşmesi için ? kullanılır. Örneğin:

```
$ls ch*.doc
```

Komutu ile adı ch ile başlayan ve .doc ile biten bütün dosyalar listelenir.

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc ch02-1.doc c
```

***Gizli dosyalar:***

İsminin ilk karakteri . (nokta) olan dosyalar gizli dosyalardır. Unix programları gizli dosyaları genellikle konfigürasyon dosyası olarak kullanır.

Gizli dosyaları görüntüleyebilmek için ls listeleme komutu -a opsiyonuyla kullanılır.

```
$ ls -a

.          .profile       docs     lib     test_results
..         .rhosts        hosts    pub     users
.emacs     bin            hw1      res.01  work
.exrc      ch07           hw2      res.02
.kshrc     ch07.bak       hw3      res.03
$
```

- tek nokta **.** : İçerisinde bulunulan dizin demektir.
- çift nokta **..** : İçerisinde bulunulan dizinin bir üst dizinini gösterir. (parent)

***Dosya oluşturma:***

Dosya oluşturmak için birden fazla komut bulunmaktadır. Bunlardan bazıları **cat, touch, nano, vi.** nano ve vi aslında metin editörüdür. Örneğin vi edötürü şu şekilde dosya oluşturur:

```
$ vi dosyaadi
```

Bu komutların kullanımı labaratuvarda uygulama ile gösterilecektir.

***Bir dosyadaki kelime sayısını bulma:***

Bir dosyada kaç tane kelime, satır ya da karakter olduğunu **wc** komutu gösterebilir. Aşağıda basit olarak kullanımı gösterilmiştir:

```
$ wc dosya_adi
2  19 103 dosya_adi
$
```

İlk kolon (2) : toplam satır sayısını, ikinci kolon (19) toplam kelime sayısını, üçüncü kolon (103) byte olarak büyüklüğü, dördüncü kolon dosyanın adını gösterir. Aşağıdaki syntax ile birden fazla dosya için bu bilgiler görüntülenebilir.

```
$ wc dosya_adi1 dosya_adi2 dosya_adi3
```

***Dosya kopyalama:***

Bir dosyayı kopyalamak için **cp** komutu kullanılır. Syntax'ı aşağıdaki gibidir.

```
$ cp kaynak_dosya hedef_dosya
```

### *Dosya yeniden adlandırma:*

Dosya ismi mv ile değiştiriliir.

```
$ mv eski_dosya yeni_dosya
```

### *Dosya silme:*

Bir dosya **rm** komutu ile sililebilir. Syntax'ı şöyledir:

```
$ rm dosya_adi
```

### *Dizinin listelenmesi:*

```
$ls dizin_yolu
```

Örnek:

```
$ls /usr/local

X11       bin          gimp       jikes       sbin
ace       doc          include    lib         share
atalk     etc          info       man         ami
```

### *Dizin oluşturma:*

```
$mkdir dizin_yolu
$
```

### *Dizin silme:*

```
$rmdir dizin_yolu
$
```

### *Dizin değiştirme:*

Bir dizinden başka bir dizine geçmek için:

```
$cd dizin_yolu
$
```

### *Dizini yeniden isimlendirme:*

```
$mv eski_dizin yeni_dizin
$
```

### *. (nokta) ve .. (nokta nokta) dizinleri*

. içerisinde bulunulan dizini, .. bu dizinin bir parent yolunu gösterir.

### Unix – Dosya izinleri / Erişim modları

UNIX'de bir dosyanın sahibinin kim olduğu önemli bir konudur.

***İzin belirteçleri:***

Bir dosyanın izinleri **ls -l** komutu ile gösterilebilir.

```
$ls -l /home/student
-rwxr-xr--  1 student student 1024  Nov 2 00:10  myfile
drwxr-xr--- 1 student student 1024  Nov 2 00:10  mydir
```

Burada ilk kolon izinleri gösterir. read (r), write (w), execute (x):

İlk üç karakter (2-4) dosyayı oluşturan kişinin (sahibinin) izinlerini gösterir. Myfile dosyası için gösterilen -rwxr-x**r--** dosya sahibinin read (r), write (w), execute (x) izinlerinin olduğunu gösteriyor.

(5-7) sırasındaki karakterler dosyayı oluşturan kullanıcının ait olduğu grubun izinlerini gösterir. rwxr-x**r--** grubun read (r) ve execute (x) izinlerinin olduğunu fakat write (w) izninin olmadığını gösteriyor.

Karakterlerin son üçü (8-10) diğer tüm kullanıcılar için izinleri gösterir. rwxr-x**r--** kullanıcı grubunda olmayan diğer kullanıcıların sadece read (r) izninin olduğunu gösteriyor.

***Dosya izin modları:***

1. **Read:** Okuma. Dosyanın içeriğine bakma.
2. **Write:** Dosyayı değiştirme ve içeriğini silme.
3. **Execute:** Dosyayı çalıştırma, bir program olarak yürütme.

***Dizin izin modları:***

Dosya için geçerli olan izin modları geçerli olmak üzere bir kaç farklılığın bilinmesi gerekir:

**1. Read:**

Dizindeki dosyaların isimlerini ve içeriğini görebilmektir.

**2. Write:**

Kullanıcının dizine dosya ekleyebilmesi ya da silebilmesi ve dosyalarda değişiklik yapabilmesidir.

**3. Execute:**

Bu izinin dizin açısından pek bir anlamı yoktur.

Bir kullanıcının cd, ls gibi komutları çalıştırabilmesi için **bin** dizinine erişimi olmalıdır.

***İzinlerin değiştirilmesi:***

Bir dosya ya da dizinin izinlerinin değiştirilmesi için **chmod** (change mode) komutu kullanılır. Bu komut  symbolic mode ve absolute mode olarak iki farklı şekilde kullanılabilir.

### *chmod – Sembolik Mod:*

Aşağıdaki tabloda belirtilen operatörler yoluyla değişiklik yapılır.

| Chmod operator | Açıklama |
|---|---|
| + | Dosya ya da dizine belirtilen izni ekler |
| - | Dosya ya da dizinden belirtilen izni çıkarır |
| = | Belirtilen izinleri atar |

```
$ls -l testfile
-rwxrwxr--  1 amrood   users 1024  Nov 2 00:10  testfile

$chmod o+wx testfile

$ls -l testfile
-rwxrwxrwx  1 amrood   users 1024  Nov 2 00:10  testfile

$chmod u-x testfile

$ls -l testfile
-rw-rwxrwx  1 amrood   users 1024  Nov 2 00:10  testfile

$chmod g=rx testfile

$ls -l testfile
-rw-r-xrwx  1 amrood   users 1024  Nov 2 00:10  testfile
```

```
$chmod o+wx,u-x,g=rx testfile

$ls -l testfile
-rw-r-xrwx  1 amrood   users 1024  Nov 2 00:10  testfile
```

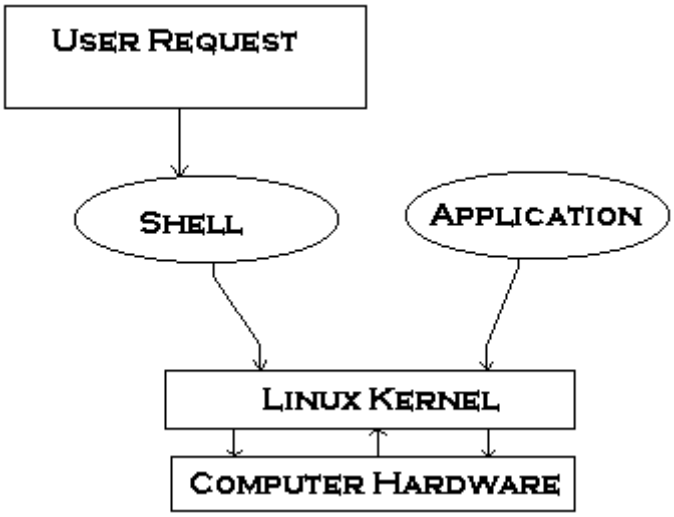
## 4. SHELL PROGRAMLAMA

Shell(terminal) interaktiftir program yürütme birimidir. Yani terminale klavyeden komutlar girilir ve bu komutlar sırayla çalıştırılır. Bunun yerine çalıştırılacak komutlar bir dosyaya kaydedilip sırasıyla çalıştırılabilir. Komutların bu şekilde yürütüldüğü programlara “Shell script” denir.  Bu programlamanın en önemli avantajları şunlardır:

- Shell script kullanıcıdan, dosyadan giriş alabilir ve ekranda görüntüleyebilir.
- Kendi komutlarımızı üretmemize imkan tanır.
- Zaman kazandırır.
- Günlük rutin işlerin otomatikleştirilmesini sağlar.

- Sistem yönetimi işlerinin otomatikleştirilmesini sağlar.

Linux yüklü makineler terminal ekranı ile Shell'e erişim verir. Eğer işletim sistemi "metin modu" (text mode) da ise sistem başladığında Shell otomatik olarak açılır. Eğer Desktop versiyon bir Linux yüklüyse Shell bir GUI tarafından açılır. Genellikle "Uygulamalar->Araçlar->Terminal" şeklinde açılır. Eğer uzak bir makineye (örneğin bir sunucuya) bağlanıp üzerinde Shell komutları çalıştırmak için Secure Shell yani SSH kullanılır. Buraya aynı zamanda komut satırı da denilir.




**Şekil 4-** Kernel ve Shell

Şekil-4'de görüldüğü üzere Kernel Shell ve diğer uygulamar ile donanım arasında köprü görevi görmektedir. Bir Linux makinesinin kaynakları Kernel tarafından yönetilir. I/O yönetimi, Dosya Yönetimi, Process Yönetimi ve Memory Yönetiminden Kernel sorumludur. Kullanıcı Shell ile bir kaynak üzerinde gerçekleştirilmek üzere bir istekte bulunduğu zaman, bu kaynağı kullanma iznine, ne kadar kullanabileceğine Kernel karar verir.

***Sistemde yüklü Shell programlarının adlarının ekrana yazdırılması:***

```
student@student-UnixLab:~/examples$
```

```
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
```

**Aktif olan shell'in gösterilmesi :**
Eğer yukarıdaki gibi birden fazla Shell programı listelendiyse bu sisteminizin birden fazla Shell'i desteklediğini gösterir. Sistemdeki geçerli Shell programının hangisi olduğunu aşağıdaki şekilde öğrenebilirsiniz.

```
$ echo $SHELL
/bin/bash
```

Bir Shell programı, sırayla çalıştırılmak üzere yazılmış komut satırlarından oluşur. İyi bir Shell programı açıklama satırlarına sahip olmalıdır. Bu şekilde adımları daha net anlaşılabilir. Açıklama satırı başına # simgesi getirilir.

Shell script'de şartlı yapılar (A, B'den büyük ise gibi), döngüler, değişkenler, fonksiyonlar mevcuttur.

Shell script derlenmez, yorumlanır. **compile** yerine **interpret** aşaması vardır.

Shell script programları sh uzantılı dosyalara yazılır, çalıştırma izni verilir ve çalıştırılır.

***Örnek Script:***

Bir test.sh isimli script oluşturalım. Bir script dosyasında herşeyden önce ( #! ) yazılır. Bu şekilde sisteme script'in başladığı haber verilir. Oluşturduğunuz test.sh dosyasına aşağıdaki satırları yazıp kaydedin.

Not: #!/bin/bash ile sisteme gelecek komutların Bash Shell tarafından çalıştırılacağını söylenir. (Eğer Python olsaydı *#!/usr/bin/python* yazılırdı vb...).

```
#!/bin/bash

# Author : student
# Unix LAB
# Fırat -CENG
pwd
ls
```

Bu dosya aşağıdaki şekilde çalıştırılabilir hale getirilir.

```
$chmod +x test.sh
```

Çalıştırılmaya hazır olan test.sh dosyası aşağıdaki şekilde çalıştırılır.

```
$./test.sh
```

**Note:** Bir dizinde bulunan programlar **./program_adi** şeklinde çalıştırılır.

Bir Shell programı tabi ki bu kadar basit olmayıp değil değişkenlerin, kontrol yapılarının vb. kullanıldığı daha karmaşık işler için kullanılır. Shell gerçek bir programlama dilidir. Fakat komutlar ne kadar karmaşıklaşırsa da bütün satırların ardışıl olarak çalıştırıldığı bir yapı söz konusudur.

Aşağıdaki script **read** konutunu kullanarak kılaveden değer girişi alır ve bu değeri PERSON değişkenine atar. Son olarak da PERSON değişkeninin değeri ekrana yazılır.

```
#!/bin/sh

echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

Programın örnek bir çıktısı şöyledir:

```
$./test.sh
What is your name?
student
Hello, student
$
```

### *Shell – Değişkenlerin kullanılması*

Değişkenlere karakter, sayı, metin, dosya adı vb. her türlü değer atanabilir. Shell'de değişken oluşturma, değer atama ve silme işlemleri yapılabilir.

Değişken isimleri sadece harf (a'dan z'ye ya da A'dan Z'ye), rakam (0'dan 9'a) ya da altçizgi ( _ ) karakteri içerebilir. Unix Shell değişkenleri bir gelenek olarak büyük HARFLERDEN oluşur.

Aşağıdaki örnek değişken isimleri verilmiştir:

```
_ALI
TOKEN_A
VAR_1
VAR_2
```

Aşağıdaki **hatalı** değişken isimleri verilmiştir:

```
2_VAR
-VARIABLE
VAR1-VAR2
VAR_A!
```

!,*, ya da – gibi karakterlerin kullanılamamasının nedeni bu karakterlerin Shell için özel anlamlara sahip olmasıdır.

Değişkenler aşağıdaki gibi tanımlanır:

```
değişken_adı = değişken_değeri
```

Örnek:

```
NAME="Bilgisayar Mühendisliği"
```

Yukarıdaki örnekte NAME değişkeni tanımlanmış ve bu değişkene "Bilgisayar Mühendisliği" değeri atanmıştır. Bu şekildeki değişkenlere skalar değişken denir. Yani hem string hem de sayı tutabilir:

```
VAR1="Bulut"
VAR2=100
```

### *Değişkenlere erişim:*

Değişken kullanılacağı zaman önüne dolar işareti olan $ getirilerek değerine ulaşılır. Aşağıdaki örnekte NAME değişkenine atanan "Bulut Bilişim" değeri **echo** komutu ile ekrana yazdırılır.

```
#!/bin/sh

NAME="Bulut Bilişim"
```

```
echo $NAME
```

### *Read-only Değişken:*

Read-only yani sadece okunabilen değişkenlerin değerleri değiştirilemez. Örneğin aşağıdaki kod dosyası çalıştırıldığında hatayla karşılaşılır.

```
#!/bin/sh

NAME="student"
readonly NAME
NAME="Ali"
```

```
/bin/sh: NAME: This variable is read only.
```

### *Unsetting Variables:*

Shell'in değişken listesinden bir değişkeni kaldırmak yani silinmesini sağlamak için aşağıdaki syntax kullanılır.

```
unset değişken_adı
```

### *Global ve Local Değişken:*

Aşağıdaki örnek incelenip çalıştırıldığında VAR değişkenine dikkat edilirse Global ve Local değişken farkı gözlenebilir.

```
#!/bin/bash
#Bu değişken globaldir ve script'in her yerinde kullanılabilir
VAR="global değişken"
function bash {
#Bu değişken localdir ve sadece bu fonksiyon içerisinde değerini korur
local VAR="local değişken"
echo $VAR
}
echo $VAR
bash
# Global değişkenin değiştirilmediğine dikkat ediniz
# "local" Shell'de özel bir tanım kelimesidir.
echo $VAR
```

Programın çalıştırılması :

```
$ ./degiskenler.sh
global değişken
local değişken
global değişken
```

### *Shell programına parametre göndermek:*

Shell programına nasıl parametre gönderildiğinin anlaşılabilmesi içim öncelikle aşağıdaki özel Shell değişkenleri incelenmelidir.

| Değişken | Açıklama |
|---|---|

| | |
|---|---|
| **$0** | Yürütülen script'in adı. |
| **$n** | $1 , $2 gibi $n tane parametrenin sırasıyla değerini gösterir. |
| **$#** | Bir script'in kaç tane parametre ile çağrıldığının değerini verir. |

Bu değerlerin kullanımını aşağıdaki programla gözleyebiliriz.

```
#!/bin/sh
echo "Dosya adi: $0"
echo "1. Parametre : $1"
echo "2. Parametre : $2"
echo "3. Parametre : $3"
echo "Toplam parametre sayısı : $#"
```

Programın çalıştırılması:

```
./dene.sh bir iki
Dosya adi: ./dene.sh
1. Parametre : bir
2. Parametre : iki
3. Parametre :
Toplam parametre sayısı : 2
```

Yukarıdaki örnek bir Shell programına nasıl parametre gönderildiğini ve bu parametrelere program içerisinde nasıl erişildiğini göstermektedir. Buna ek olarak aşağıdaki program kodundaki parametre erişim yöntemlerini inceleyiniz.

```
#!/bin/bash
# Gelen parametreler bir dizide tutulabilir
args=("$@")
echo ${args[0]} ${args[1]} ${args[2]}

#$@ kullanılarak bütün parametreler tek seferde yazdırılabilir
echo $@
```

Programın çalıştırılması:

```
$ ./passarg.sh Param_Bir Param_İki Param_Üç
Param_Bir Param_İki Param_Üç
Param_Bir Param_İki Param_Üç
```

### *Shell programı içerisinde komut kullanmak:*

```
#!/bin/bash
# kesme sembolü  " ` ` " içerisinde komut yazılarak Shell'de çağrılır:
echo `date`
# kesme olmadan kod:
echo date
```

Programın çalıştırılması:

```
$ ./shell_komut.sh
Sun Oct 19 18:42:36 UTC 2014
date
```

### *Şartlı İfadeler*

Shell programlamada şartlı ifadeler aşağıdaki gibidir:

| *# Temel if yapısı* | *# if-else yapısı* | *# if-else if yapısı* |
|---|---|---|
| ***if şart ; then***<br>***komutlar***<br>***fi*** | ***if şart ; then***<br>***komutlar***<br>***else***<br>***komutlar***<br>***fi*** | ***if şart ; then***<br>***komutlar***<br>***elif şart ; then***<br>***komutlar***<br>***fi*** |

```
#!/bin/bash

sayı=1

if [ $sayı = "1" ]; then
    echo "sayı eşittir bir"
fi
```

```
#!/bin/bash

sayı=1

if [ $sayı = "1" ]; then
    echo "sayı eşittir bir"
else
    echo "sayı birden farklı"
fi
```

### *Aritmetik karşılaştırmalar:*

Shell programlamada küçüktür, büyüktür, eşittir gibi ifadelere aşağıdaki tablodaki operatörlerle karılaştırılabilir. Aşağıdaki programı diğer operatörleri de kullanarak deneyiniz.

| | |
|---|---|
| -lt | < |
| -gt | > |
| -le | <= |
| -ge | >= |
| -eq | == |
| -ne | != |

```
#!/bin/bash
# NUM1 ve NUM2 sayılarının karşılaştırılması
NUM1=2
NUM2=1
if   [ $NUM1 -eq $NUM2 ]; then
   echo "Sayılar birbirine eşittir"
elif [ $NUM1 -gt $NUM2 ]; then
   echo "NUM1 NUM2'den büyüktür"
else
   echo "NUM2 NUM1'den büyüktür"
fi
```

### *String karşılaştırma:*

Stringlerin karşılaştırılmasında aşağıdaki tabloda verilen operatörler kullanılır.

| | |
|---|---|
| = | eşittir |
| != | eşit değildir |
| < | küçüktür |
| > | büyüktür |
| -n s1 | s1 string'i boş değildir |
| -z s1 | s1 string'i boştur |

```
#!/bin/bash
#String tanımla S1
S1="Bash"
```

```
#String tanımla S2
S2="Scripting"
if [ $S1 = $S2 ]; then
        echo "Stringler aynı"
else
        echo "Stringler birbirinden farklıdır"
fi
```

Not: Birden fazla karşılaştırmayı aynı satırda yapmak için || (or) ya da && (and) operatörü kullanılır. Örneğin aşağıdaki satırda YAŞ değişkeninin 20'den küçük veya 50'den büyük olduğu şart oluşturulmuştur.

```
if [ "$YAŞ" -lt 20 ] || [ "$YAŞ" -ge 50 ]; then
```

***Dosyalar için karşılaştırma operatörleri:***

Dosyalar üzerinde şartlı ifadelerle kullanılabilecek bazı önemli operatörler ve anlamları aşağıdaki tabloda verilmiştir.

| | |
|---|---|
| -s | dosya mevcut ve boş değil |
| -f | dosya mevcut ve dizin değil |
| -d | dizin mevcut |
| -x | dosya çalıştırılabilir |
| -w | dosya yazılabilir |
| -r | dosya okunabilir |

```
#!/bin/bash
#test.sh dosyasının var olup olmadığını test
eden Shell programı
file="./test.sh"
if [ -s $file ]; then
        echo "Dosya mevcut"
else
        echo "Böyle bir dosya yok"
fi
```

***Döngüler:***

Shell programlamada for, while, select ve until döngüleri bulunmaktadır. Burada for ve while döngülerinin kullanımı gösterilecektir. Diğer döngüler bu deneye dahil değildir.

```
#!/bin/bash
#1'den 10'a kadar olan sayıları while döngüsü kullanarak ekrana yaz
sayi=0
while [ $sayi -lt 10 ]
   do
        sayi=$((sayi+1))
        echo $sayi
   done
```

```
#!/bin/bash
#1'den 10'a kadar olan sayıları for döngüsü kullanarak ekrana yaz
#for i in 0 1 2 3 4 5 6 7 8 9
#yukarıdaki şekilde veya aşağıdaki gibi olabilir

for i in {1..10}
do
   echo $i
done
```

```
#bulunduğu dizindeki dosyaların isimlerini ekrana yazan program
#!/bin/bash
        for i in $( ls ); do
            echo item: $i
        done
```

### *Shell programlamada metod kullanımı*

```
#!/bin/sh

# Kendisine gelen 2 parametreyi Merhaba ile ekrana yazan ve 100 değerini döndüren
# Merhaba metodu tanımlanıyor.
Merhaba () {
   echo "Merhaba $1 $2"
   return 100
}

# Metodun çağrılması
Merhaba Unix LAB
# Metodun dönderdiği değeri ekrana yazdır
echo $?
```

Program çıktısı:

```
$ ./fonksiyon.sh
Merhaba Unix LAB
100
```

**Not:** Shell fonksiyonları aslında return ile değer döndürmez. Fakat “ $? “ ile yürütülen son komutun değeri yakalanabilir. Bu da ancak sayısal bir değer dönendirildiğinde mümkündür.

## DENEY SORULARI

1. Ekrana “merhaba dünya” yazan Shell programını yazınız

2. Bir kullanıcının bütün dosyalarını yedekleyen bir komut yazınız. İpucu: tüm dosyalar bir tar dosyası olarak saklanabilir)

3. Dışarıdan bir dizini parametre olarak alıp bu dizinin var olup olmadığını ekrana yazdıran Shell programınını yazınız.

4. Ekrana tarih, saat, kullanıcı adı ve içerisinde bulunan dizini yazan Shell programını yazınız.

.

5. Kullanıcıdan şifre girişi yapmasını isteyip, şifre “unixstudent” ise ekrana “şifre doğru” değilse “şifre hatalı” yazan Shell programını yazınız.

6. Kullanıcıdan dosya ismi (diziniyle birlikte) girmesini isteyip dosya mevcutsa dosya içeriğini ekrana yazan, mevcut değilse dosyayı oluşturup içerisine “merhaba dünya” yazan Shell programını yazınız.

7. "bulut" kullanıcı isimli ve "193.255.124.68" ip adresine sahip bir linux bilgisayara bağlanmak için gerekli komutu yazınız (SSH bağlantısı sağlamak için openssh-server kurulumu gerçekleştiriniz).

8. Yazılım ve donanım kaynaklarının hizmet olarak sunulabilmesini sağlayan açık kaynak bulut bilişim yazılımları nelerdir?

9. Ornek.c

```
   #include <stdio.h>
int main(void)
{
printf("merhaba");
return 0;
}
```

   Ornek.c dosyasındaki kodu derleyen Linux komutunu yazınız.

10. “cat /etc/resolv.conf” ve “ping firat.edu.tr” komutlarını açıklayınız.

## KAYNAKLAR


1. http://en.wikipedia.org/wiki/Unix
2. http://wiki.ubuntu-tr.net/
3. http://www.tutorialspoint.com/unix/index.htm
4. http://www-h.eng.cam.ac.uk/help/tpl/unix/scripts/node2.html
5. aws.amazon.com/ec2/