T.C.
MİLLİ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

AÇIK KAYNAK İŞLETİM SİSTEMİ - 3

ANKARA 2008

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**481BB0050**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Ağ İşletmenliği – Web Programcılığı – Bilgisayar Teknik Servisi**
MODÜLÜN ADI	**Açık Kaynak İşletim Sistemi 3**
MODÜLÜN TANIMI	Bu modül, açık kaynak işletim sisteminde kabuk işlemlerini, dosya ve dizin paylaşımını içeren öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Açık Kaynak İşletim Sistemleri 2 modülünü başarmış olmak
YETERLİK	Açık kaynak kodlu işletim sisteminde kabuk işlemlerini yapabilmek, dosya ve dizin paylaşımını gerçekleştirebilmek
MODÜLÜN AMACI	**Genel Amaç** Bu modül ile gerekli ortam sağlandığında; açık kaynak kodlu işletim sisteminin yönetimini gerçekleştirebileceksiniz. **Amaçlar** 1. Açık kaynak işletim sisteminde kabuk işlemlerini gerçekleştirebileceksiniz. 2. Dosya ve dizin paylaşımı için ağ servislerini kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANİMLÂRI	İnternete ve ağa bağlı bilgisayar laboratuvarı Açık Kaynak İşletim Sistemi yazılımı
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendinizi değerlendireceksiniz. Modül sonunda ise kazandığınız bilgi ve becerileri ölçmek amacıyla hazırlanan ölçme araçları (uygulama, çoktan seçmeli, soru cevap) ile kendinizi değerlendireceksiniz.

GİRİŞ

Sevgili Öğrenci

Bir işletim sistemine grafik ortamda hükmedebilirsiniz. Günlük hayatta gerekli basit bilgisayar işlerinin çoğunu grafik ortamda halletmeniz mümkündür. Fakat bilgisayar alanında çalışan biri için her zaman basit gündelik işler yoktur. Zaman zaman bir ağı veya bir sistemi kurmanız gerekecek ya da bilgisayarda kaybolan önemli bir verinin peşine düşmek zorunda kalacaksınız. Bu işlemler bazen yapılması uzun süren, bir sürü komutun ardarda girilmesini gerektiren işlemler olabilir. İlk öğrenme faaliyetinde, açık kaynak işletim sisteminin temel komutlarından bahsedeceğiz. Bu komutları nasıl kullanacağınızı ve gerektiğinde nasıl kabuk programı yazabileceğinizi öğrenecek, böylece tek bir komutla, birçok komutun işlevini yerine getirebileceksiniz.

Artık bilgi çağında olmanın gereği olarak, bilgi paylaşımı da önem kazanmıştır. Günümüzde bilgi paylaşımı için en önde gelen kaynak, bilgisayarlardır. Bir çalışma ortamında basit bir yerel ağ oluşturmanız birçok sorunu çözecektir. Böylelikle hem zamandan hem de emekten tasarruf edebilirsiniz. İkinci öğrenme faaliyetinde, dosya, dizin ve yazıcıları, Pardus kurulu olan ve olmayan makineler arasında nasıl paylaştıracağınızı öğreneceksiniz. Son olarak İnternet üzerinde dosya paylaşımını sağlayan programlardan bahsedeceğiz.

Bu modül hazırlanırken, sizlere yol gösterecek yazı tipi düzenlemelerine sadık kalınmıştır. Bu düzenlemeler ve anlamları aşağıda belirtilmiştir. Takıldığınız noktalarda, modülün en sonunda verilen *önerilen kaynaklar* kısmından da yararlanabilirsiniz.

Açık kaynak kodlu işletim sisteminde ileri düzeye bir adım olacak bu modülü başarıyla bitirmeniz dileğiyle.

ÖĞRENME FAALİYETİ-1

AMAÇ

Açık kaynak kodlu işletim sisteminde kabuk işlemlerini yapabileceksiniz.

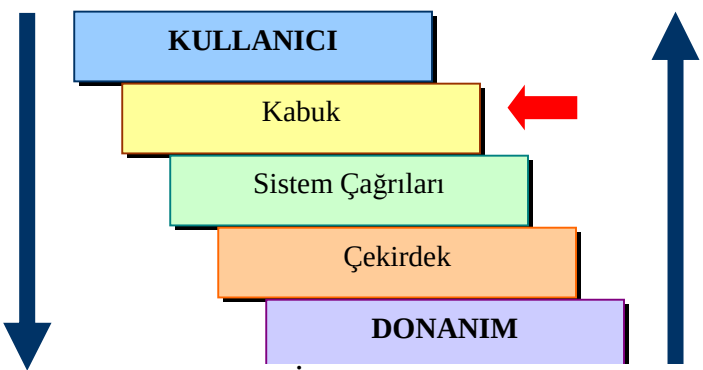
ARAŞTIRMA

- İşletim sistemlerinde "kabuk" hangi işlevleri üstlenir? Araştırınız.
- Komut sistemi ve grafik arayüzde çalışmanın sağladığı yararları ve sınırlılıklarını sınıfta tartışınız.

1. KABUK İŞLEMLERİ

1.1. Kabuk (Shell)

İşletim sistemlerinde kullanıcı ile bilgisayar donanımı arasında genel olarak 3 katmandan söz edilebilir. Şekil 1.1'de görülen bu katmanlar; kabuk, sistem çağrıları ve çekirdek olarak adlandırılır.



Şekil 1.1: İşletim sistemi katmanları

Kabuk (*shell*), kullanıcı ile işletim sistemi çekirdeği arasında bir ara birimdir. Kullanıcı tarafından verilen komutları algılar, yorumlar ve sistem çağrıları yardımıyla çekirdeğe iletir. Çekirdek ise bilgisayar donanımıyla doğrudan etkileşen ve işletim sistemi işlevlerini gerçekleştiren kısımdır. Kabuk, işletim sisteminin kullanıcıya görünen yüzüdür. Birçok kaynakta "**komut yorumlayıcısı**" olarak da adlandırılmaktadır.

Bilgisayar açıldıktan bir süre sonra komut satırı (*prompt*) görüntülenir. Kullanıcı tarafından komut satırına girilen komutlar, bilgisayar tarafından işleme konulur. İşte bu noktada kabuk olarak adlandırılan program devreye girer. Öncelikle komutun geçerliliğini inceler, kullanıcının ne yapmak istediğini çözümler ve bu iş için gerekli programları belleğe yükler.

1.1.1. Kabuk Programı

Kabuk programı, kullanıcı ile işletim sisteminin çekirdeği arasında bağlantı kuran, köprü kuran program olarak adlandırılabilir. MS-DOS işletim sisteminde kabuk olarak *command.com* kullanılır. DOS işletim sisteminin aksine, Unix'te geliştirilmiş birçok kabuk programı vardır. Bu programlar Tablo 1.1'de listelenmiştir.

Tablo 1.1: Kabuk programları

Kabuk Programı	Açıklama
Bash	(Born again shell) GNU tarafından C shell'in özelliklerinin geliştirildiği standart kabuk programı
Csh	C kabuk programı
Ksh	Korn kabuk programı
Sh	Standart UNIX kabuk programı
Tcsh	C kabuk programının geliştirilmiş hali
Zsh	Korn kabuk programının geliştirilmiş hali

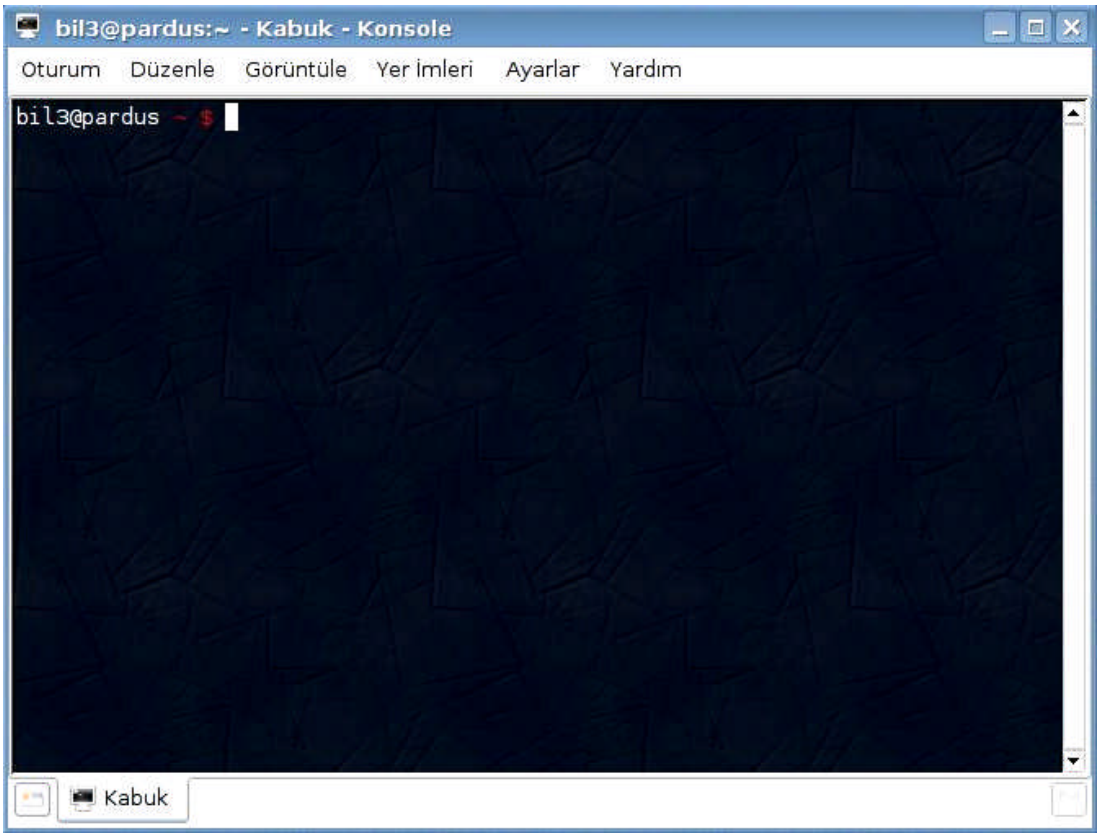
Linux sistemlere erişildiğinde kullanıcı adı ve şifre bilgileri girildikten sonra sistem tarafından ilk olarak kullanıcının kabuk programı çalıştırılır. Kabuk programı, girilen komutları algılayan ve işleyen temel programdır. Kullanıcı hesabı açılırken kullanıcının hangi kabuk programını kullanacağı sistem yöneticisi tarafından belirlenir ve **etc/passwd** dosyasında bu bilgi tutulur. Kullanıcı sisteme bağlandığında kullanıcı bilgileri okunurken çalıştıracağı kabuk program bilgisi de okunur ve çalıştırılır.

1.2. Komut Satırı ve Programlara Parametre Gönderme

Pardus'ta, bir masaüstü işletim sistemiyle yapılabilecek grafik ortamdaki tüm işlemlerin yanı sıra, birçok işlemi komut satırı üzerinden gerçekleştirmek mümkündür. Bu yöntem artık klasikleşmiş olsa da Linux sistemlerin güçlü yanını ortaya koymaktadır.

1.2.1. Komut Satırı (Konsole Uygulaması)

Pardus'ta kabuk üzerinde çalışabilmek için komut satırı uygulamasına geçilmelidir. Pardus'taki komut satırı uygulaması, **Konsole** (Terminal Programı) olarak isimlendirilir. Konsole uygulamasını çalıştırmak için Pardus ana menüsünden ***Programlar → Sistem → Konsole (Terminal Programı)*** seçeneğini seçebilirsiniz.



Şekil 1.2 :Konsole uygulaması

Bu çalıştırma sonrasında sisteme girdiğiniz kullanıcının haklarıyla işlem yapabileceğiniz Şekil 1.2'de görülen konsol penceresi açılacaktır. Bu konsol penceresi klavyeden girdi alır, ayrıca fareyle kopyalama ve yapıştırmaya da izin verir. Klavyeden aldığı komutları işlediğinden dolayı konsol, "komut satırı arayüzü" (*command line interface* – ***CLI***) olarak da isimlendirilir.

- Komut istemcisinin sonunda imleçten önce yer alan karakter; sistem yöneticisi olan *root* kullanıcısı için **#**, diğer kullanıcılar için ise **$** olur.

Konsole programına erişmek için aşağıdaki yöntemleri de kullanabilirsiniz:

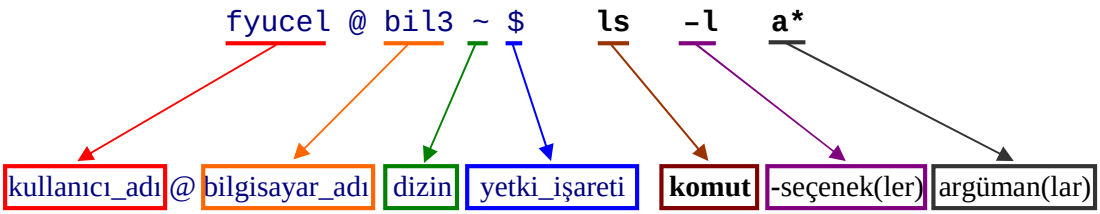
- **Alt+F2** kısayolundan açılan **Komut Çalıştır...** penceresinde "**konsole**" yazarak Konsole programını çalıştırabilirsiniz.

- **Ctrl+Alt+F1..F6** ile de 1'den 6'ya kadar ayrı ayrı kabuk oturumu açmanız mümkündür. Bu seçenekle, komut sisteminde tam ekran modunda çalışabilirsiniz. Komut ekranından grafik ortama geri dönmek için ise **Ctrl+Alt+F7** kısayol tuşunu kullanabilirsiniz.

1.2.2. Komut Yapısı

- Komut, kullanıcı tarafından istenen hizmetin çalıştırılması için, işletim sistemine ya da bir uygulamaya istekte bulunulmasına yarayan sözcük, ifade ya da kısaltmalardır. Örneğin, "Dosya isimlerini listele" isteği için "ls" komutu verilmelidir.

Linux tabanlı sistemlerde komut yapısı genellikle Şekil 1.3'teki gibi olmaktadır.



Şekil 1.3: Komut Yapısı

Bu yapıda kullanıcı adı, bilgisayar adı ve her kabukta gösterilmese de yetki işareti (**$** veya **#**) görüntülenir. Kullanıcı adı, bilgisayar adı ve yetki işareti sistem tarafından otomatik olarak yazıldığı için, kullanıcı tarafından girilmesine gerek yoktur.

Komut ifadesinin ardından varsa komuta ait seçenek ve argümanlar belirtilir.

- **Seçenek ve Argüman**

Bir komutun **hangi şekilde** çalışması isteniyorsa bu, **seçenekler**de belirtilir. **Argümanlar** ise komutun **ne üzerinde** çalışacağını belirler. Argümana örnek olarak dosya, süreç (proses) ya da kullanıcı verilebilir. Seçenek ve argümanlara gerek duymadan çalışabilen komutlar da vardır.

Linux komutlarında seçenekler genellikle tek tire "-" ya da çift tire "--" işaretinden sonra kullanılır. Örneğin, **`ls -l`** komutunda kullanılan "***-l***" seçeneği ile sadece dosya isimleri değil, dosyaların ayrıntıları da listelenir.

Komut seçeneklerinin kısa veya uzun yazılışları olabilir:

- Kısa yazılış : Tek tire (-) tek harf
- Uzun yazılış: Çift tire (--) sözcük

Çoğu komutta bulunan bazı seçenekler aşağıda verilmiştir:

- `-h --help` : Komutla ilgili yardım bilgisini verir.
- `-v --verbose` : İşlemin aşamalarını ayrıntılı gösterir.
- `-q --quite` : Ekrana minimum çıktı verir.
- `-V --version` : Komutun sürüm numarasını gösterir.

Birden fazla seçeneği yan yana veya bitişik olarak kullanmak mümkündür. Örneğin, **`ls -a -l`** komutunda, "***-a***" ve "***-l***" seçenekleri ayrı kullanılmışken, **`ls -al`** komutunda her iki parametre "***-al***" şeklinde birleştirilmiştir. Her iki komutun da yaptığı iş aynıdır.

Argümanlarda ise tek tire "-" ya da çift tire "--" işareti bulunmaz. Örneğin; cd komutu argüman ile beraber kullanıldığında verilen dizini açar. Dizin adı (*belgeler* dizini) argüman olarak verildiğinde, **cd belgeler** şeklinde yazılır.

Aşağıda komut yazımlarına örnekler verilmiştir:

- `$ date` Komut
- `$ cal 12 2000` Komut ve iki argüman
- `$ ls -l` Komut ve bir seçenek
- `$ ls -laR` Komut ve çok seçenek (bitişik yazım)
- `$ ls -l -a -R` Komut ve çok seçenek (ayrı yazım)

➢ **Özel Simgeler**

Komutlara argüman olarak verilebilecek özel simgeler vardır. Bu simgeler yardımıyla, değeri tam olarak bilinmeyen ya da belirli bir aralık içinde tanımlanan karakterler ifade edilebilir. Örneğin, ls m* komutu kullanıldığında, m ile başlayan dosyalar listelenir.

- Yıldız (*****) karakteri, herhangi **bir ya da birden fazla** karakterin yerine kullanılır. Boş kümeyi de ifade eder.
  - Örneğin, **a*** yazımı, 'a' karakteri ile başlayan bütün sözcükleri gösterir: *a, araclar, a75* gibi...
  - ***z** yazımı ise 'z' karakteri ile biten bütün sözcükleri ifade eder: *az, a95z, z* gibi...
  - **re*m** yazımı, "re" ile başlayıp "m" ile biten sözcükleri tanımlar: *resim, rengim, re57m, rem* gibi...
- Soru işareti (**?**) karakteri, herhangi **tek** bir karakterin yerine kullanılır.
  - **a?** yazımı, a harfi ile başlayan 2 karakterli sözcükleri ifade eder. Burada (?) tek bir karakter yerine geçer: ab, a2, a+
  - **kale?** yazımı, "kale" ile başlayıp herhangi bir karakter ile biten sözcükleri gösterir: kalem, kale5 gibi...
  - **a??c** yazımı ise a harfi ile başlayıp c harfi ile biten sözcükleri tanımlar: agac, arac, a2bc gibi..
- Köşeli parantezler **[...]**, belli bir karakter kümesini ya da belirli bir karakter aralığını ifade eder. Bu simge, "köşeli parantez içerisindeki karakterlerden herhangi biri" anlamına gelir.
  - **[abc]z** ifadesi; a, b ve c harfleriyle başlayıp z harfiyle biten, 2 karakterli sözcükleri ifade eder: az, bz ve cz.
  - **a[57b]c** ifadesi; a harfi ile başlayıp c harfiyle sonlanan, ortadaki karakterin 5, 7 veya b olduğu 3 karakterli sözcükleri gösterir: a5c, a7c, abc.

- **k[a-e]** ifadesi, k harfiyle başlayan ve a, b, c, d, e harflerinden biriyle devam eden 2 karakterli sözcükleri tanımlar. Tire (-) işareti, aralık belirtir: ka, kb, kc, kd, ke.
- **a[1-5]*** ifadesi ise a ile başlayıp 1 ile 5 arasındaki herhangi bir sayıyla devam eden tüm sözcükleri gösterir: a1, a2b, a3klm, a456, a5i gibi...
- **k[m,3-7]** ifadesi, k ile başlayıp m harfiyle ya da 3 ve 7 arasındaki rakamlar ile biten sözcükleri tanımlar: km, k3, k4, k5, k6 ve k7.

➢ **Kontrol Karakterleri**

Özel kontrol karakterleri kullanılarak, ekran çıktısını durdurulabilir, tekrar başlatılabilir, girilen bir komut satırı silinebilir ya da çalışan bir komut durdurulabilir.

Kontrol karakterleri kullanılırken **Ctrl** tuşu basılı tutulmalı, ardından yapılması istenen harekete uygun tuşa basılmalıdır. Kontrol karakterleri, Tablo 1.2'de açıklanmıştır.

Tablo 1.2: Kontrol karakterleri

Kontrol Karakteri	Etkisi
Ctrl+C	Çalışmakta olan komutu durdurur.
Ctrl+U	Komut satırında bulunan tüm karakterleri siler.
Ctrl+S	Ekrandan geçmekte olan çıktıyı durdurur.
Ctrl+Q	Ctrl+S ile durdurulmuş çıktının tekrar başlatılmasını sağlar.
Ctrl+D	Dosya sonu karakteri EOF (End-of-File) ya da "çık" anlamındadır.
Ctrl+W	Komut satırında imlecin bulunduğu yerden bir önceki son kelimeyi siler.
Shift+Page Up/Page Down	Sayfa sayfa yukarı/aşağı

1.2.3. Yardım Alma

Linux işletim sistemlerinde **man** komutu ile komutlar hakkında yardım alınabilir. İngilizce "manual" (kullanım kılavuzu) kelimesinin kısaltmasından oluşmuştur.

Kullanım Şekli:

man [<Bölüm>] <komut adı>

Örneğin, `ls` komutunun ne işe yaradığını öğrenmek için Örnek 1.1'deki komut girilebilir:

Örnek 1.1

```
bil3@pardus ~ $ man ls
```

Bu komut girildiğinde, ekranda ls komutunun ne işe yaradığı ve alabileceği parametreler birlikte görülecektir. Klavyedeki aşağı yukarı tuşlarıyla dosya içerisinde gezinilebilir ve "**q**" tuşuna basılarak çıkılabilir.

Yardım almak için bir diğer seçenek **`help`** komutudur. Kullanımı Örnek 1.2'de verilmiştir.

Örnek 1.2

```
bil3@pardus ~ $ help ls
```

`info` ve **`<komut> --help`** yardım almak için kullanılabilecek diğer seçeneklerdir. **`info`** komutunun kullanımı Örnek 1.3'te gösterilmiştir.

Örnek 1.3

```
bil3@pardus ~ $ info ls
```

Komut hakkında kısa yardım almak için Örnek 1.4'teki komut kullanılır.

Örnek 1.4

```
bil3@pardus ~ $ ls --help
```

1.2.4. Dosya ve Dizinler

Komut satırında, dosya ve dizin yolu (adresi) iki şekilde verilebilir:

- **Mutlak Yol:** Kök dizinden (/) başlayarak dosya veya dizin yolu belirtilir. Örneğin; /home/fyucel/a.txt şeklinde verilebilir.
- **Bağıl Yol:** İçerisinde bulunulan dizine göre verilebilir. Örneğin, kullanıcı, **/home/fyucel** dizini içerisinde ise bu dizin içerisindeki "**a.txt**" dosyasını, doğrudan "**a.txt**" yazarak belirtebilir. Ya da **/home** dizini içerisindeyse, "**fyucel/a.txt**" yazarak adreslendirebilir.

Ayrıca Linux'ta bazı dizinlerin özel isimleri vardır. Bunlar aşağıdaki Tablo 1.3'te özetlenmiştir:

Tablo 1.3:. Özel dizin isimleri

Dizin Adı	Anlamı
`.`	Bulunulan dizini gösterir.
`..`	Bir üst dizini gösterir.
`-`	Bir önceki dizini gösterir.
`~`	O anki kullanıcının ev dizinini gösterir.
`~kullanıcı`	Belirtilen kullanıcının ev dizinini gösterir.
`/`	Kök dizin

1.3. Standart Giriş/Çıkış ve Yönlendirme

Standart giriş/çıkış ve yönlendirme işlemleri için birçok komut kullanılmaktadır. Bu komutlardan önemli olan bir kısmı burada ele alınacaktır.

1.3.1. Standart Giriş/Çıkış Komutları

1.3.1.1. pwd Komutu (Present Working Directory)

O anda içinde bulunulan dizinin tam yolunu (*full path*) verir. Kullanımı, Örnek 1.5'te verildiği şekildedir.

Örnek 1.5

```
bil3@pardus ~ $ pwd
/home/bil3
```

1.3.1.2. cd Komutu (Change Directory)

Dizinler arasında geçiş yapmayı sağlayan komuttur.

Örnek 1.6

```
bil3@pardus ~ $ cd /var/log
bil3@pardus log $
```

Örnek 1.6'daki komut çalıştırıldığında, log dizini içerisine geçiş yapılır. Tekrar kullanıcının ev dizinine dönmek için Örnek 1.7'deki komut kullanılabilir.

Örnek 1.7

```
bil3@pardus log $ cd ~
bil3@pardus ~ $
```

1.3.1.3. echo Komutu

Karakterleri ya da değişkenlerin değerlerini görüntüler. Kullanıcının kabuk programı, sistemde **$SHELL** adlı değişkende saklanır. Bu değişkenin değerini görüntülemek için, Örnek 1.8'de gösterildiği gibi **echo** komutu kullanılabilir.

Örnek 1.8

```
bil3@pardus ~ $ echo $SHELL
/bin/bash
```

Linux ortamında değişken tanımlayarak değişkenin değerini ekranda görüntüleyelim.

Örnek 1.9

```
bil3@pardus ~ $ KIM="Fırat YÜCEL"
bil3@pardus ~ $ echo KIM
KIM
bil3@pardus ~ $ echo $KIM
Fırat YÜCEL
bil3@pardus ~ $ echo "Hos geldin KIM"
Hos geldin KIM
bil3@pardus ~ $ echo "Hos geldin $KIM"
Hos geldin Fırat YÜCEL
```

Yukarıdaki örnekte, KIM adında bir değişken tanımlanmış ve değişkenin değeri değişik şekillerde ekrana yazdırılmıştır. Echo komutu ile yazdırırken değişkeni belirtmek için önüne **$** işareti getirilmelidir.

- ✓ Linux'ta büyük-küçük harf ayrımı vardır. Dolayısıyla değişkenleri nasıl tanımladıysanız aynı karakterlerle çağırmalısınız. KIM değişkenini tanımlarken "=" simgesi ile diğer ifadeler arasında boşluk bırakılmadığına dikkat ediniz.

1.3.1.4. ls Komutu (List Directory Contents)

Bulunan dizin içerisindeki dosya ve dizinleri listeler. Kullanımı Örnek 1.10'da gösterildiği şekildedir.

Örnek 1.10

```
bil3@pardus ~ $ ls
Arşiv.tar.gz
belgelerim
belge.txt
deneme.txt
Desktop
erisim.pdf
```

Bu komutun bazı seçenekleri vardır. Örneğin, **`ls -l`** komutu, bulunan dizindeki dosya ve dizinleri ayrıntılı bilgileriyle listeler.

Örnek 1.11

```
bil3@pardus ~ $ ls -l
toplam 37760
-rw-r--r-- 1 bil3 users 8760124 Şub 18 14:01 Arşiv.tar.gz
drwx------ 2 bil3 users    4096 Şub 10 23:59 belgelerim
-rw-r--r-- 1 bil3 users    1062 Oca 30 12:31 belge.txt
-rw-r--r-- 1 root root       43 Şub  9 01:57 deneme.txt
drwx------ 2 bil3 users    4096 Mar 22 22:43 Desktop
-rw-r--r-- 1 bil3 users  321065 Şub  9 23:39 erisim.pdf
-rw-rw-rw- 1 bil3 users  103936 Oca 29 15:26 fyucel_oneriler.doc
```

Örnek 1.11'de ilk kısımda dosya veya dizine ait kullanıcı yetkileri, dizin sayısı, sahibi olan kullanıcının adı, grubu, dosya boyutu (byte), son değişiklik tarihi ve dosyanın ya da dizinin adı yer alır. Dizinler için satır başındaki ilk karakter "**`d`**"dir.

Herhangi bir dizin içerisindeki dosyaları görüntülemek için, Örnek 1.12'de gösterildiği gibi **`ls`** komutundan sonra dizin yolu yazılır.

Örnek 1.12

```
bil3@pardus ~ $ ls /var
cache  db  empty  lib  lock  log  pisi  run  spool  state  tmp
```

Örnek 1.12'de **`/var`** dizini içerisindeki dosyalar görüntülenmektedir.

Eğer **`ls`** komutu **`--color`** seçeneği ile birlikte kullanılırsa, dosya türlerine göre renklendirme uygulanacaktır. Bu seçeneğin üç argümanı vardır:

- **`always`**, her zaman renkli gösterir,
- **`never`**, hiçbir zaman renkli göstermez,
- **`auto`** ise otomatik olarak görünümü ayarlar.

Örnek 1.13

```
bil3@pardus ~ $ ls --color=always belgelerim
deneme1.txt  deneme.txt~  modul_taslak.doc  modul_taslak.doc.tar.gz
```

Örnek 1.13'te, ev dizini (~) altında yer alan belgelerim dizini içerisindeki dosya ve dizinler, renklendirilerek gösterilmiştir.

Gizli dosya ve dizinler dahil tüm dosya ve dizinleri listelemek için ***-a*** seçeneği kullanılır. Bu seçenek **`ls -a`** şeklinde yalnız başına kullanılabileceği gibi, ayrıntılı listelemek için ***-l*** seçeneğiyle beraber de kullanılabilir.

Örnek 1.14

```
bil3@pardus ~ $ ls -al
toplam 38016
drwx--x--x 28 bil3 users    4096 Mar 25 10:41 .
drwxr-xr-x  5 root root     4096 Şub 27 10:35 ..
drwx------  3 bil3 users    4096 Şub 27 13:35 .adobe
-rw-r--r--  1 bil3 users 8760124 Şub 18 14:01 Arşiv.tar.gz
-rw-------  1 bil3 users    2604 Mar 25 00:32 .bash_history
-rw-r--r--  1 bil3 users     127 Oca 20 14:49 .bash_logout
-rw-r--r--  1 bil3 users     194 Oca 20 14:49 .bash_profile
-rw-r--r--  1 bil3 users    1496 Oca 20 14:49 .bashrc
drwx------  2 bil3 users    4096 Şub 10 23:59 belgelerim
-rw-r--r--  1 bil3 users    1062 Oca 30 12:31 belge.txt
-rw-r--r--  1 bil3 users     240 Oca 21 15:10 .chromium
-rw-r--r--  1 bil3 users    3800 Oca 21 15:10 .chromium-score
drwx------  4 bil3 users    4096 Şub  9 02:28 .config
:      :     :     :     :     :     :     :     :
```

Örnek 1.14'te, ***-a*** ve ***-l*** seçenekleri, **`ls -a -l`** biçiminde ayrı kullanılabilirdi. Fakat çok seçenekli komutlarda eğer seçeneklerle ilgili bir argüman kullanılmayacaksa, seçenekleri, örnekte görüldüğü gibi ***-al*** şeklinde birleşik olarak da yazmak mümkündür.

- ✓ Linux'ta gizli dosyaların dosya adı önünde nokta (**.**) bulunur. Bu dosyalar normal koşullarda listelenmezler.

Eğer dosyalarla ilgili tarama yapmak isterseniz, **`ls`** komutuyla birlikte özel simgeleri de kullanabilirsiniz. Örneğin, "m" harfi ile başlayan dosyaları listelemek isterseniz, Örnek 1.15'te gösterildiği gibi **m*** argümanı ile bunu sağlayabilirsiniz.

Örnek 1.15

```
bil3@pardus ~ $ ls m*
maviresim.jpg  menu.lst  modul_taslak.doc  modul_taslak.doc.tar.gz
```

Daha ayrıntılı dosya adı taraması yapabilmek için **Konu 1.2.2.3**'te özel simgeler hakkında ayrıntılı açıklamalar bulabilirsiniz.

Son olarak birkaç seçenek ve argümanın birleşimini gerektiren bir işlem gerçekleştirelim. Örneğin, "**/home/bil3/belgelerim**" dizini altında yer alan "**tar.gz**" uzantılı tüm dosyaları ayrıntılı olarak listeleyelim.

Örnek 1.16

```
bil3@pardus ~ $ ls -al /home/bil3/belgelerim/*.tar.gz
-rw-r--r-- 1 bil3 users 388360 Şub 10 23:59
/home/bil3/belgelerim/modul_taslak.doc.tar.gz
```

Örnek 1.16'da ***.tar.gz** ifadesi, ".tar.gz" ile biten tüm dosyaları göstermektedir.

1.3.1.5. mkdir Komutu (Make Directory)

Yeni dizin oluşturur. Kullanımı basittir. Örneğin, bulunulan dizinde "belgeler" adında yeni bir dizin oluşturmak için aşağıdaki komut kullanılır.

Örnek 1.17

```
bil3@pardus ~ $ mkdir belgeler
```

Yukarıdaki komut, kullanıcının ev dizini (~) içerisinde **belgeler** isimli yeni bir dizin oluşturur.

1.3.1.6. mv Komutu (Move)

Dosya ya da dizinleri taşıma veya isim değiştirme işlemini gerçekleştirir. Örneğin, "**pisi.pdf**" adlı dosyayı **belgeler** dizini içerisine taşımak için Örnek 1.18'deki komut kullanılır.

Örnek 1.18

```
bil3@pardus ~ $ mv pisi.pdf belgeler
bil3@pardus ~ $ ls belgeler
pisi.pdf
```

Artık pisi.pdf dosyası, kullanıcı ev dizininin içerisinden alınarak, kullanıcı ev dizini içerisinde oluşturmuş olduğumuz belgeler dizinine atılmıştır.

“pisi.pdf” dosyasının adını, “pisi1.pdf” olarak değiştirmek istersek, Örnek 1.19’daki komutu kullanabiliriz.

Örnek 1.19

```
bil3@pardus ~ $ cd belgeler
bil3@pardus belgeler $ ls
pisi.pdf
bil3@pardus belgeler $ mv pisi.pdf pisi1.pdf
bil3@pardus belgeler $ ls
pisi1.pdf
```

Yukarıdaki örnekte, **cd** komutu ile belgeler dizini içerisine girilmiş ve **ls** komutuyla içerik listelenmiştir. Böylece, *belgeler* dizini içerisinde “pisi.pdf” dosyasının olduğu görülmüştür. Daha sonra, **mv** komutu kullanılarak “pisi.pdf” dosya adı, “pisi1.pdf” olarak aynı yere taşınmıştır. **ls** ile tekrar listeleme yapıldığında dosya adının “pisi1.pdf” olarak değiştirildiği görülmektedir.

1.3.1.7. cp Komutu (Copy)

Dosya veya dizin kopyalama işlemini gerçekleştirir. **pisi1.pdf** dosyasını bir üst dizine kopyalamak için örnekteki komut kullanılabilir.

Örnek 1.20

```
bil3@pardus belgeler $ cp pisi1.pdf ..
bil3@pardus belgeler $ cd ..
bil3@pardus ~ $ ls pisi1.pdf
pisi1.pdf
```

Yukarıdaki örnekte **/home/bil3/belgeler** dizini içerisinde yer alan “pisi1.pdf” isimli dosya bir üst dizine (**..**) kopyalanmıştır. Çift nokta (**..**), bir üst dizini, yani bu örnek için kullanıcı ev dizinini (/home/bil3) göstermektedir. **cd ..** bir üst dizine çıkar. Burada “pisi1.pdf” dosyası listelendiğinde, kopyalanmış olduğu görülebilir.

Kullanıcı ev dizininde yer alan “**a.txt**” adlı bir dosyayı **/home/bil3/belgeler** dizinine kopyalamak için;

Örnek 1.21

```
bil3@pardus ~ $ cp a.txt /home/bil3/belgeler
```

ya da

Örnek 1.22

```
bil3@pardus ~ $ cp a.txt belgeler
```

komutları kullanılabilir. Burada dikkat edilmesi gereken husus ya kopyalanacak dosyanın bulunduğu dizin içerisinde bulunulmalı ya da dosya yolu doğru bir şekilde belirtilmelidir.

Kopyalama işlemi dizinler üzerinde de yapılabilir. Farklı dosya sistemi üzerinde olmamak kaydıyla bir dizin ve altındaki her şeyi başka bir dizine kopyalamak mümkündür. Kopyalarken, **-R** seçeneğinin kullanılması gerekir.

Örnek 1.23

```
bil3@pardus ~ $ cp -R belgeler belgeler_kopya
```

Örnek 1.23'te **belgeler** dizini, **belgeler_kopya** adıyla kullanıcı ev dizini içerisine kopyalanmıştır.

1.3.1.8. ln Komutu (Link)

Dosya ya da dizinler için bağlantı (link) oluşturur. a.txt adlı dosyaya bir bağlantı oluşturmak için Örnek 1.24'teki komut kullanılır.

Örnek 1.24

```
bil3@pardus ~ $ ln a.txt baglanti.txt
```

Bu komut, "a.txt" dosyasının "baglanti.txt" adında bağlantı dosyasını oluşturur. Her iki dosyanın birinde yapılan değişiklik, diğerini de aynen etkiler.

Örneğin, kullanıcı ev dizininde (/home/bil3) bulunan **belgeler** dizini için masaüstünde (/home/bil3/desktop) bir bağlantı (link) oluşturmak için Örnek 1.25'teki komut kullanılabilir.

Örnek 1.25

```
bil3@pardus ~ $ ln -s /home/bil3/belgeler /home/bil3/Desktop/belgeler
```

Bu komuttaki ***-s*** seçeneğine dikkat ediniz. Dizinler arasında doğrudan bağlantı yerine sembolik bağlantı oluşturulabilir. ***-s*** seçeneği, sembolik bağlantıyı sağlar.

1.3.1.9. rm Komutu (Remove)

Dosya ya da dizin silme işlemini gerçekleştirir. Örneğin, belge.txt isimli dosyayı silmek için Örnek 1.26'daki komut kullanılır.

Örnek 1.26

```
bil3@pardus ~ $ rm belge.txt
rm: normal dosya `belge.txt' silinsin mi?e
```

Dosya silme işlemlerinde, yanlışlıkla veri kaybına neden olmamak için silme işleminde onay alınır. "**e**" (Evet) yanıtını verdiğinizde dosya silinir.

İçerisinde alt dizinler bulunan bir dizini silmek için **-R** seçeneği kullanılır.

Örnek 1.27

```
bil3@pardus ~ $ rm -R belgeler
rm: `belgeler' dizininin içine inilsin mi?e
rm: normal dosya `belgeler/pisi1.pdf' silinsin mi?e
rm: dizin `belgeler' silinsin mi?e
```

Örnek 1.27'de **belgeler** dizini içerisindekilerle birlikte silinmiştir. Eğer silme işleminde yukarıdaki gibi teker teker onay alınması istenmiyorsa ***-f*** seçeneği kullanılabilir.

- ✓ **rm** komutunu kullanırken çok dikkatli olmalısınız. Aksi durumda veri kaybına neden olabilirsiniz. Boşlukların ve küçük büyük harflerin önemini unutmayınız. Örneğin, **`rm *.txt`** komutu tüm *txt* uzantılı dosyaları siler. Yanlışlıkla **`rm * .txt`** şeklinde noktadan önce boşluk bırakarak yazmanız durumunda tüm dosyaların silinmesine neden olabilirsiniz.

1.3.1.10. bc Komutu

Aritmetik işlemlerin gerçekleştirildiği bir programdır. Ondalıklı sayılarla çalışmaya olanak sağlar. Program içerisinde değişken tanımlamak, formül hazırlamak mümkündür. C dilindeki deyimler de kullanılabilir.

Örnek 1.28

```
bil3@pardus ~ $ bc
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
10.45+203.0218
213.4718
a=2^4
b=14*12
b-a
```

1.3.1.11. cat Komutu (Concatenate Files)

Dosyanın içeriğini görüntülemek için kullanılır. Örneğin, "**soz.txt**" dosyasının içeriğini görüntülemek için Örnek 1.29'daki komut kullanılmalıdır.

Örnek 1.29

```
bil3@pardus ~ $ cat soz.txt
Dört Şey Geri Gelmez!!
Atılan Ok!!
Kaçan Fırsat !!
Söylenen Söz!!
Geçen Zaman !!
```

"**belge1**" dosyasının içeriğini "**yedek**" isimli yeni bir dosyaya kopyalamak için Örnek 1.30'daki komut kullanılabilir.

Örnek 1.30

```
bil3@pardus ~ $ cat belge1 > yedek
```

Burada > operatörü, dosyayı aynen kopyalamak için kullanılır. Eğer **belge1**, **belge2** ve **belge3** dosyalarını birleştirip "belge" isimli dosyanın sonuna eklemek istersek aşağıdaki komut kullanılır.

Örnek 1.31

```
bil3@pardus ~ $ cat belge1 belge2 belge3 >> belge
```

Örnek 1.31'de >> operatörü, herhangi bir dosyanın sonuna başka bir dosyayı veya başka dosyaları eklemek için kullanılır.

1.3.1.12. clear Komutu

Terminali temizler ve imleci ilk satıra taşır. Kullanımı, Örnek 1.32'deki gibidir.

Örnek 1.32

```
bil3@pardus ~ $ clear
```

Bu işlemden sonra ekran temizlenir ve imleç ilk satıra taşınır.

1.3.1.13. date Komutu

Sistem saatini ve tarihini (istenirse düzenlenerek) görüntüler ya da sistem yöneticisi olarak ayarlama işlemini gerçekleştirir.

Örnek 1.33

```
bil3@pardus ~ $ date
Sal Mar 15 15:20:39 EET 2008
```

Eğer tarih ve saat düzenlenerek görüntülenmek istenirse Örnek 1.34'teki komut kullanılabilir.

Örnek 1.34

```
bil3@pardus ~ $ date +%H:%M:%S-%d/%m/%Y
15:27:46-25/03/2008
```

1.3.1.14. df Komutu (Display File System)

Bağlanılmış disk bölümlerinin boyut ve doluluk bilgilerini görüntüler. Kullanımı, Örnek 1.35'te verildiği şekildedir.

Örnek 1.35

```
bil3@pardus ~ $ df
Dosyasistemi         1K-blok        Dolu Boş       Kull%Bağlanılan yer
/dev/hda7              8214592   4082540   3965140  51% /
tmpfs                   257856         0    257856   0% /dev/shm
/dev/hda5             20482840     67172  20415668   1% /mnt/hda5
/dev/hda6             10080488   2698148   6870272  29% /mnt/hda6
/dev/hda1             40957684  39101100   1856584  96% /mnt/hda1
```

1.3.1.15. du Komutu (Display Usage Space)

Dosya ve dizinlerin disk kullanım alanlarını görüntüler. Kullanımı, Örnek 1.36'da verildiği şekildedir.

Örnek 1.36

```
bil3@pardus ~ $ df
8       ./.kde3.5/Autostart
8       ./.kde3.5/share/services
24      ./.kde3.5/share/mimelnk/image
8       ./.kde3.5/share/mimelnk/application
36      ./.kde3.5/share/mimelnk
4       ./.kde3.5/share/servicetypes
12      ./.kde3.5/share/config/colors
4       ./.kde3.5/share/config/kdm
8       ./.kde3.5/share/config/kresources/calendar
8       ./.kde3.5/share/config/kresources/contact
20      ./.kde3.5/share/config/kresources
12      ./.kde3.5/share/config/session
636     ./.kde3.5/share/config
920     ./belgelerim
:          :          :          :
```

1.3.1.16. expr Komutu (Evaluate Expressions)

Aritmetiksel ve mantıksal işlemleri gerçekleştirir.

Örnek 1.37

```
bil3@pardus ~ $ expr 3 + 5
8
bil3@pardus ~ $ expr \( 5 + 15 \) / 4
5
bil3@pardus ~ $ expr 3 > 2
bil3@pardus ~ $ expr 3 < 2
3
bil3@pardus ~ $ expr 1 == 1
1
bil3@pardus ~ $ expr 1 == 4
0
```

Örnek 1.37'de basit aritmetiksel ve mantıksal işlemler gerçekleştirilmiştir. Parantez işaretinden önce ters bölü (\) karakteri kullanılmalıdır. Her bir değer veya sembol arasında birer boşluk bulunmalıdır.

1.3.1.17. gzip, gunzip, tar Komutları

Bu komutlar, arşivleme ve sıkıştırma işlemlerini gerçekleştirir. Çok disk alanı kaplayan dosya ve dizinler daha az alan kaplaması için sıkıştırılabilir.

Örnek 1.38

```
bil3@pardus ~ $ gzip deneme.txt
```

Örnek 1.38'deki komut, **deneme.txt** dosyasını sıkıştırarak **deneme.txt.gz** dosyasını oluşturur. Sıkıştırma sonucunda deneme.txt dosyası kaldırılır. Bu nedenle sıkıştırma işleminin, sıkıştırılacak dosyaların kopyalanarak farklı bir birimde gerçekleştirilmesi önerilir.

Sıkıştırılmış bir **gz** dosyasını açmak için ise Örnek 1.39 veya 1.40'daki komutlardan birisi kullanılabilir.

Örnek 1.39

```
bil3@pardus ~ $ gzip -d deneme.txt.gz
```

veya

Örnek 1.40

```
bil3@pardus ~ $ gunzip deneme.txt.gz
```

/home/fyucel/belgeler dizini içerisindeki tüm dosya ve alt dizinleri ayrı ayrı sıkıştırmak için ise Örnek 1.41'deki komut kullanılabilir.

Örnek 1.41

```
bil3@pardus ~ $ gzip -r /home/fyucel/belgeler
```

tar, Linux sistemlerde kullanılan standart arşiv programıdır. Bu programla arşivleme veya sıkıştırarak arşivleme işlemleri gerçekleştirilebilir.

Örnek 1.42

```
bil3@pardus ~ $ tar -cf deneme.tar deneme1.txt deneme2.txt
```

Örnek 1.42'de verilen komut "deneme1.txt" ve "deneme2.txt" dosyalarından "deneme.tar" adlı bir arşiv oluşturur.

Örnek 1.43

```
bil3@pardus ~ $ tar -xf deneme.tar
```

Örnek 1.43'teki komut ise "deneme.tar" adlı arşiv dosyasını açar.

"deneme" adlı dizini sıkıştırarak "deneme.tar.gz" adlı sıkıştırılmış bir arşiv dosyası oluşturmak için Örnek 1.43'teki komut kullanılabilir.

Örnek 1.44

```
bil3@pardus ~ $ tar -czf deneme.tar.gz deneme
```

1.3.1.18. host Komutu

Girilen IP/DNS (alan adı kaydı) bilgisinin karşılığını verir.

Örnek 1.45

```
bil3@pardus ~ $ host eml.serik.gov.tr
eml.serik.gov.tr has address 91.93.128.21
```

Örnek 1.45'teki komut sonucunda, alan adı olarak **eml.serik.gov.tr** girildiğinde bu adrese ait IP numarası görüntülenmektedir.

1.3.1.19. ps, kill Komutları

Bilgisayarda çalışan her programın bir süreç (*process*) kimliği ya da numarası (*PID, Process ID*) bulunur. Bilgisayarda o anda çalışan programların listesini görüntülemek için **`ps`** komutu kullanılır. Eğer ***-ef*** ya da ***-aux*** seçeneği kullanılırsa, o anda çalışan tüm kullanıcılara ait programlar ayrıntılı bir şekilde listelenir.

Örnek 1.46

```
bil3@pardus ~ $ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 10:24 ?        00:00:00 init [3]
root         2     1  0 10:24 ?        00:00:00 [migration/0]
root         3     1  0 10:24 ?        00:00:00 [ksoftirqd/0]
:          :       :        :          :           :
bil3      3598  3596  0 10:37 ?        00:00:06
/usr/lib/MozillaFirefox//firefox
bil3      3792  3446  2 11:00 ?        00:00:00 konsole
[kdeinit]
```

Bilgisayarda çalışan süreçlerden birini durdurmak için **`kill`** komutu kullanılır. *Kill*, "öldürmek" demektir. Burada ise süreçleri yok etmek anlamındadır. Örneğin, Mozilla Firefox programına ait süreci sonlandırmak için ilgili sürece ait süreç numarası (3598) ile birlikte Örnek 1.47'deki komut kullanılır.

Örnek 1.47

```
bil3@pardus ~ $ kill 3598
```

Bu komutla birlikte, örnekteki bilgisayarda 3598 süreç numarasına sahip Mozilla Firefox programı sonlandırılır. Sonlandırmayı zorlamak için ise **`kill -9 <PID>`** komutu kullanılabilir.

1.3.1.20. killall

Çalışan uygulamaları isme göre sonlandırır.

Örnek 1.48

```
bil3@pardus ~ $ ps
  PID TTY          TIME CMD
 3820 pts/2    00:00:00 bash
 3837 pts/2    00:00:00 vim
 3840 pts/2    00:00:00 ps
bil3@pardus ~ $ killall -9 vim
[1]+  Süreç durduruldu      vim
```

Örnek 1.48'de, **ps** komutu ile süreçler listelenmiş, **killall -9** komutu ile **vim** adlı süreç zorlanarak durdurulmuştur.

Eğer bir sürecin, siz başka süreçlerle uğraşırken arka planda çalışmasını isterseniz, Örnek 1.49'daki gibi komutun yanına **&** simgesini parametre olarak verebilirsiniz.

Örnek 1.49

```
bil3@pardus ~ $ vim &
[1] 3837
```

Bu durumda sürece bir **PID** verilir ve bu numara ekranda görüntülenir. Süreç işlemini bitirene kadar arka planda çalışmaya devam eder.

1.3.1.21. su Komutu (Switch User)

Kullanıcı değiştirmek için kullanılır. Herhangi bir seçenek girilmediğinde sistem yöneticisi (*root*) girişi yapar. Komutu kullanan *root* kullanıcısı ise parola sorulmaz, diğer durumlarda değişecek kullanıcının parola bilgisi istenir.

Örnek 1.50

```
bil3@pardus ~ $ su -
Parola:
pardus ~ # _
```

Örnek 1.50'de sistem yöneticisi (*root*) oturumu açılmıştır. Burada, komut satırındaki **$** simgesinin, **#** simgesine dönüştüğüne dikkat ediniz.

Sistem yöneticisi (*root*) yetkileriyle komut satırında işlem yaparken çok dikkatli olmalısınız. Sistem yöneticisine komutların çalışmasında kısıtlama olmadığından, sisteme zarar verebilirsiniz.

logout komutuyla yönetici oturumundan ayrılabilirsiniz.

1.3.1.22. whereis, which Komutları

Komutların ve ilgili dosyaların bulundukları konumu listeler. Kullanımları, Örnek 1.51'de gösterildiği gibidir.

Örnek 1.51

```
bil3@pardus ~ $ whereis ls
ls: /bin/ls /usr/bin/ls /usr/share/man/man1p/ls.1p
/usr/share/man/man1/ls.1
bil3@pardus ~ $ which ls
/usr/bin/ls
```

1.3.1.23. w, who Komutları

w ve **who** komutları sisteme bağlı kullanıcıları listeler. Kullanımları, Örnek 1.52'de verildiği gibidir.

Örnek 1.52

```
bil3@pardus ~ $ w
 12:02:19 up  1:38,  3 users,  load average: 0,10, 0,11, 0,09
USER     TTY      LOGIN@   IDLE   JCPU   PCPU  WHAT
bil3     :0       10:24   ?xdm?   4:08   0.03s /bin/sh
bil3     pts/0    10:25    1:37m  0.00s  0.61s kded [kdeinit] --new-
bil3     pts/1    11:00    0.00s  0.05s  0.00s w

bil3@pardus ~ $ who
bil3     :0           2008-03-27 10:24
bil3     pts/0        2008-03-27 10:25
bil3     pts/1        2008-03-27 11:00
```

1.3.1.24. lpr, enscript Komutları

Belgelerin yazıcıdan çıktı alınmasını sağlar. PostScript yazıcılarda, metin dosyalarının çıktılarının alınabilmesi için, belgeler ***ps*** formatına dönüştürülmeli ya da **enscript** programı ile çıktı alınmalıdır.

➢ **PostScript** daha çok masaüstü yayıncılığında kullanılan bir sayfa tanımlama dili ve bir programlama dilidir. Tasarlanmış bir sayfadaki metin, görsel unsurlar vb. öğelerin kenar çizgilerini belirtmek için Béziger eğrileri olarak bilinen matematik formüller aracılığıyla komutlar biçiminde bir belge oluşturur. Bu belge PostScript komutlarını yorumlayabilen özel işlemcili tüm çıktı birimleriyle kullanılabildiği için PostScript'e "**araç-bağımsız**" da denir. Bu çıktı birimleri lazer yazıcı olduğu kadar mürekkepli bant kullanan nokta-vuruşlu yazıcılar ve/veya görüntü diziciler de olabilir. Sonuçta bu belge yazıcılarda yorumlanarak, görüntü çıktı şeklinde elde edilebilir.

Örnek 1.53'te, ilk komut, "deneme.txt" dosyasını yazıcıya gönderir. İkinci komut olan **`enscript -p deneme.ps deneme.txt`** komutu, "deneme.txt" belgesini **ps** formatıyla "deneme.ps" olarak kaydeder. **`cat`** komutuyla da "deneme.ps" dosyasının içeriği görüntülenmiştir.

Deneme.ps dosyasını **danio** adlı yazıcıya göndermek için ise **`lpr -Pdanio deneme.ps`** komutu kullanılabilir.

Örnek 1.53

```
bil3@pardus ~ $ enscript deneme.txt
[ 1 sayfa * 1 kopya ] yaz?c?'ye g?nderildi

bil3@pardus ~ $ enscript -p deneme.ps deneme.txt
[ 1 sayfa * 1 kopya ] deneme.ps'de b?rak?ld?

bil3@pardus ~ $ cat deneme.ps
%!PS-Adobe-3.0
%%BoundingBox: 24 24 571 818
%%Title: Enscript Output
%%For: Fırat Yücel
%%Creator: GNU enscript 1.6.4
%%CreationDate: Thu Mar 27 12:17:19 2008
%%Orientation: Portrait
%%Pages: (atend)
%%DocumentMedia: A4 595 842 0 () ()
%%DocumentNeededResources: (atend)
%%EndComments
%%BeginProlog
%%BeginResource: procset Enscript-Prolog 1.6 4
%
% Procedures.
%

/_S {   % save current state
```

1.3.2. Girdi ve Çıktıların Yönlendirilmesi

Bazı durumlarda komutların çıktısının ekranda görüntülenmesi yerine bir dosyaya kaydedilmesi veya başka bir birime yönlendirilmesi gerekebilir. Ya da bir komut girdisinin, klavyeden değil de herhangi bir başka birimden alınması istenebilir. Bu durumlarda yönlendirme operatörleri kullanılmaktadır. İki adet yönlendirme operatörü vardır: “>” ve “>>” operatörü.

1.3.2.1. “>” Operatörü

Bir komutun çıktısı normalde standart çıktı birimi olan ekranda görüntülenir. Eğer komutun çıktısını bir dosyaya göndermek isterseniz, “>” operatörünü kullanabilirsiniz.

Örnek 1.54

```
bil3@pardus ~ $ ls -l > deney.txt
bil3@pardus ~ $ cat deney.txt
toplam 916
-rw-r--r-- 1 bil3 users      9 Oca 29 00:12 deneme1.txt
-rw-r--r-- 1 bil3 users      2 Oca 28 21:26 deneme.txt~
-rw-r--r-- 1 bil3 users      0 Mar 27 13:40 deney.txt
-rw-rw---- 1 bil3 users 528896 Şub 10 23:53 modul_taslak.doc
-rw-r--r-- 1 bil3 users 388360 Şub 10 23:59 taslak.doc.tar.gz
```

Örnek 1.54’te, **ls -l** komutunun çıktısı, ekrana değil de “deney.txt” dosyasına yönlendirilmiştir. Eğer “deney.txt” adında bir dosya yoksa otomatik olarak oluşturulur, varsa dosya içindeki önceki veriler silinerek, yeni veriler kaydedilir.

Komut sonucunda oluşacak hatalar “>” operatörü ile yönlendirilemez.

Örnek 1.55

```
bil3@pardus ~ $ ls \l > deney.txt
ls: l'e erişilemedi: Böyle bir dosya ya da dizin yok

bil3@pardus ~ $ cat deney.txt
```

Örnek 1.55’ten görüldüğü üzere, hata çıktısı ekranda görüntülenmiş, “deney.txt” dosyasının içeriği boş bırakılmıştır. Hata çıktılarının dosyaya yazılması için, Örnek 1.56’da görüldüğü gibi “**2**>” operatörü kullanılabilir.

Örnek 1.56

```
bil3@pardus ~ $ ls \l 2> deney.txt

bil3@pardus ~ $ cat deney.txt
ls: l'e erişilemedi: Böyle bir dosya ya da dizin yok
```

Eğer komut çıktısını “cikti.txt” isimli dosyaya, oluşabilecek hata çıktısını da “hata.txt” isimli dosyaya kaydetmek isterseniz, Örnek 1.57’deki komutu kullanabilirsiniz.

Örnek 1.57

```
bil3@pardus ~ $ ls \l 2>hata.txt >cikti.txt
```

1.3.2.2. “>>” Operatörü

“>” operatörü kullanılarak yönlendirme yapıldığında, hedef dosyanın içeriği tamamen silinir. Eğer, oluşan çıktının hedef dosya içeriğinin sonuna eklenmesi istenirse, “>>” operatörü kullanılır.

Komut çıktısının “deney.txt” dosyası sonuna eklenmesi için Örnek 1.58’deki komut kullanılabilir.

Örnek 1.58

```
bil3@pardus ~ $ ls -l >> deney.txt
```

“deneme.txt” dosyasına “deneme1” sözcüğünü eklemek için de bu operatör kullanılabilir.

Örnek 1.59

```
bil3@pardus ~ $ echo deneme1 >> deneme.txt
bil3@pardus ~ $ cat deneme.txt
deneme1
bil3@pardus ~ $ echo deneme2 >> deneme.txt
bil3@pardus ~ $ cat deneme.txt
deneme1
deneme2
```

Örnek 1.59’da **echo** komutuyla “deneme1” sözcüğü ekrana yazdırılır. Fakat bu örnekte yönlendirme yapılarak “deneme.txt” dosyasına yazdırılmıştır. Eğer “deneme.txt” dosyası yoksa otomatik olarak oluşturulur. Sonrasında “deneme2” sözcüğü de “deneme.txt” dosyası sonuna eklenmiştir.

Birden çok dosya içeriğini birleştirerek yeni bir dosyaya yollamak için ise Örnek 1.60’daki komut kullanılabilir.

Örnek 1.60

```
bil3@pardus ~ $ cat dosya1.txt dosya2.txt >> dosya3.txt
```

`cat dosya1.txt dosya2.txt` komutu, “dosya1.txt” ve “dosya2.txt” dosyalarının tümünü ekranda görüntüler. Fakat yukarıdaki örnekte bu komutun çıktısı “dosya3.txt” dosyasına yönlendirme yapıldığından, “dosya1.txt” ve “dosya2.txt” dosyalarının içeriği “dosya3.txt” dosyasına eklenir.

1.4. Temel Filtre Komutları

1.4.1. Düzenli Deyimler (Regular Expressions)

Düzenli deyim, bir metni düzenlemek ya da metin içerisinden belirli kurallara uyan alt metinler elde etmek için kullanılan bir dildir. “abc”, “Merhaba Dünya”, “*ABC”, “.a?b$” deyimleri, düzenli deyime örnektir.

Düzenli deyimlerde kullanılan semboller ve anlamları Tablo 1.4’te özetlenmiştir.

Tablo 1.4: Düzenli Deyimlerde Kullanılan Semboller

Sembol	Anlamı
.	Herhangi tek bir karaktere karşılık gelir.
*****	Kendisinden önceki karakterin sıfır veya daha fazla kez olduğunu gösterir.
[..]	Küme içerisindeki karakterlerden herhangi birine karşılık gelir.
[^..]	Küme içerisindeki karakterlerin dışında herhangi bir karaktere karşılık gelir.
^	Satır başını ifade eder.
$	Satır sonunu ifade eder.
\{n\}	Kendisinden önceki karakterin n kez tekrar edildiğini ifade eder.
\{n,m\}	Kendisinden önceki karakterin en az n kez, en fazla m kez olduğunu gösterir.
\{n,\}	Kendisinden önceki karakterin en az n kez olduğunu gösterir.
\+	Kendisinden önceki karakterin 1 ya da daha fazla olduğunu gösterir.
\?	Kendisinden önceki karakterin 0 ya da 1 kez bulunduğunu gösterir.
|	Kendisinden bir önceki **veya** bir sonraki karaktere karşılık gelir.
\(..\)	Grup olarak düzenli deyimleri tanımlar.
\	Özel karakterlerin normal karakter olarak algılanmasını sağlar.

Aşağıda düzenli deyimlere örnekler verilmiştir:

- ab*c → *ac, abc, abbc, abbbc*,...
- a.b → aab, abb, acb,...
- a\?b → b, ab,...
- a[0-9] → a0, a1,.., a9
- a[0-9]* → a, a0, a00, a19, a19999,...
- a[^a-z] → a0, a1, aA, aZ,...
- Satır başında 'a' karakteriyle başlayıp devam eden sözcükler : ^a
- Satır başında 3 adet 'z' karakteri bulunan sözcükler : ^zzz veya ^z\{3\}
- Satır başında en az 2 adet k karakteri bulunan sözcükler : ^k\{2,\}
- 'y' karakteri ile sonlanan sözcükler : y$

- İçerisinde en az bir kez '5' sayısı geçen sözcükler : 5\+
- İçerisinde 3 veya 5 sayılarından en az 1 kez geçen sözcükler : \(3\|5\)\+
- Satır başında 'K' karakteri ile başlayıp, satır sonunda 'M' ile biten sözcükler : *^K.*M$*

1.4.2. grep Komutu

Düzenli deyimlerin en çok kullanıldığı yerlerden birisi dosya içerisinde herhangi bir deyimin araştırılmasında kullanılan grep komutudur.

Bu komutun kullanımı;

```
grep <Düzenli deyim> <Araştırılacak Dosya Adı>
```

şeklindedir.

Örneğin, içerisinde 'abc' deyiminin geçtiği dosyaları ve 'abc' yazan satırı görüntülemek için Örnek 1.61'deki komut kullanılabilir.

Örnek 1.61

```
bil3@pardus ~ $ grep 'abc' *
deneme.txt:abc
İkilik dosya sayi1.pdf eşleşir
```

Örnekte araştırılacak dosya adı yerine * simgesi getirildiğinden, bulunan dizin altındaki bütün dosyalar üzerinde arama gerçekleştirilir.

"deneme.txt" isimli dosyada 'a' ile başlayan satırları bulan komut Örnek 1.62'deki gibi olabilir.

Örnek 1.62

```
bil3@pardus ~ $ grep '^a' deneme.txt
abc
axyz 15y
aaabxyccc
```

"deneme.txt" adlı dosyada 'a' karakteriyle başlayıp 'c' karakteriyle biten ve toplam 3 karakterden oluşan satırları bulmak için ise Örnek 1.63'teki komut kullanılabilir.

Örnek 1.63

```
bil3@pardus ~ $ grep '^a.c$' deneme.txt
abc
```

"deneme.txt" adlı dosyada içerisinde 2 adet 'y' karakteri bulunan satırları listelemek için kullanılacak komut ise yine düzenli deyimlerle aşağıdaki şekilde oluşturulabilir.

Örnek 1.64

```
bil3@pardus ~ $ grep 'y.*y' deneme.txt
axyz 15y
```

"deneme.txt" dosyasında, içerisinde rakam geçmeyen satırları listelemek için Örnek 1.65'deki deyim kullanılır.

Örnek 1.65

```
bil3@pardus ~ $ grep '^[^0-9]*$' deneme.txt
abc
aaabxyccc
bc
```

İki, üç ya da dört haneli rakamları bulmak için Örnek 1.66'daki komut kullanılabilir.

Örnek 1.66

```
bil3@pardus ~ $ grep '^[0-9]\{2,4\}$' deneme.txt
123
```

1.4.3. cut Komutu

Belirtilen dosya veya dosyalardaki sütunları görüntüler. ***-d*** ile sütun ayıracı, ***-f*** ile sütun numarası belirtilir.

Örneğin, **/etc/passwd** dosyasında sütunlar "**:**" ile ayrılmıştır ve 5. sütunda kullanıcıların isimleri yer almaktadır. Bu dosyanın 5. sütununu görüntülemek için Örnek 1.67'deki komut kullanılabilir.

Örnek 1.67

```
bil3@pardus ~ $ cut -d: -f5 /etc/passwd
root
bin
daemon
:
PnP
Fırat Yücel
Lab Kullanıcı
nobody
```

1.4.4. sort Komutu

Belirtilen dosyayı satır satır sıralar. Örneğin, “deneme.txt” adlı dosyadaki satırları sıralamak için **`sort deneme.txt`** komutu kullanılabilir.

Örnek 1.68

```
bil3@pardus ~ $ cat deneme.txt
abc
123
axyz 15y
aaabxyccc
bc
bil3@pardus ~ $ sort deneme.txt
123
aaabxyccc
abc
axyz 15y
```

“deneme.txt” dosyasını tersten sıralamak için ise **`sort`** komutu Örnek 1.69’daki gibi ***-r*** seçeneği ile kullanılabilir.

Örnek 1.69

```
bil3@pardus ~ $ sort -r deneme.txt
bc
axyz 15y
abc
aaabxyccc
```

1.5. Dosyaların Taranması

1.5.1. cmp Komutu (Compare)

İki dosyayı karşılaştırarak farklılıklarını belirtir. Kullanımı, Örnek 1.70’deki gibidir.

Örnek 1.70

```
bil3@pardus ~ $ cat b1
Bu dosya deneme amaçlı oluşturuldu.
Birinci dosya.
bil3@pardus ~ $ cat b2
Bu dosya deneme amaçlı oluşturuldu.
İkinci dosya.
bil3@pardus ~ $ cmp b1 b2
b1 b2 farklı: bayt 40, satır 2
```

1.5.2. diff Komutu (Different)

İki metin dosyasını karşılaştırır. Farklılıkları ayrıntılı olarak gösterir. Kullanımı, Örnek 1.71'de gösterilmiştir.

Örnek 1.71

```
bil3@pardus ~ $ cat b1
Bu dosya deneme amaçlı oluşturuldu.
Birinci dosya.
bil3@pardus ~ $ cat b2
Bu dosya deneme amaçlı oluşturuldu.
İkinci dosya.
bil3@pardus ~ $ diff b1 b2
2c2
< Birinci dosya.
---
> İkinci dosya.
```

1.5.3. find Komutu

Dosya ya da dizin arama işlemlerini gerçekleştirir. Örneğin, kullanıcının ev dizinindeki bütün dosyaları listelemek için Örnek 1.72'deki komut kullanılabilir.

Örnek 1.72

```
bil3@pardus ~ $ find $HOME -print
```

$HOME bir sistem değişkeni olup kullanıcının ev dizinini saklar. Ev dizinindeki "Resim" ile başlayan dosyaları listelemek için Örnek 1.73'deki komut kullanılabilir.

Örnek 1.73

```
bil3@pardus ~ $ find /home/bil3 -name "Resim*"
/home/bil3/Pictures/Resim 029.jpg
/home/bil3/Pictures/Resim 008.jpg
/home/bil3/.local/share/Trash/files/Resim 027.jpg
/home/bil3/.local/share/Trash/info/Resim 027.jpg.trashinfo
/home/bil3/.wine/drive_c/windows/profiles/bil3/Belgelerim/Resimleri
…
```

Bulunan dizin içerisindeki sıfır boyutlu (boş) dosyaları görüntülemek için ise aşağıdaki komut kullanılabilir.

Örnek 1.74

```
bil3@pardus ~ $ find . -size 0
./.kde3.5/share/apps/kaffeine/wizard_stamp_v0.7.1
./.kde3.5/share/apps/kopete/contactlist.xml.bak
./.kde3.5/share/apps/kabc/std.vcf__0
./.kde3.5/share/apps/kabc/std.vcf
./.kde3.5/share/apps/konqueror/bookmarks.xml.tbcache
./.openoffice.org2/user/uno_packages/cache/uno_packages/9pNbLa
./belgelerim/cikti.txt
./.no-qtrc-to-gtkrc-mapping
./.qt/.qtrc.lock
./.qt/.qt_plugins_3.3rc.lock
./.mozilla/firefox/rbpj2edb.default/.parentlock
```

Örnek 1.74'te **find** komutu yanındaki nokta "**.**", bulunulan dizini ifade etmektedir.

1.5.4. sed Komutu (Stream Editor)

Metin belgeleri üzerinde komutlar ile değişiklik yapmayı sağlayan programdır. Programın kullanımı:

sed 's/*değişecek sözcük*/*yerine yazılacak sözcük*/g'

şeklindedir.

Örneğin, test.txt dosyasındaki "yanlız" sözcüklerini "yalnız" şeklinde düzeltmek için bu komut Örnek 1.75'teki şekilde kullanılır.

Örnek 1.75

```
bil3@pardus ~ $ cat test.txt
Bu parçada yalnız kelimesi yerine yanlışlıkla yanlız yazılmıştır.
Yanlız, bu hata dışında yanlış kelimesi de yalnış yazılmıştır.
Yanlız ve yalnış kelimelerini doğrusu ile düzeltin.

bil3@pardus ~ $ sed 's/yanlız/yalnız/g' test.txt
Bu parçada yalnız kelimesi yerine yanlışlıkla yalnız yazılmıştır.
Yanlız, bu hata dışında yanlış kelimesi de yalnış yazılmıştır.
Yanlız ve yalnış kelimelerini doğrusu ile düzeltin.
```

Yukarıdaki komut sonrasında küçük harfle başlayan "yanlız" sözcüğü "yalnız" olarak düzeltilmiştir. Fakat, büyük harfle başlayan sözcükler düzeltilmemiştir. Tüm "yalnız" ya da "Yalnız" sözcüklerini kapsayan düzeltme yapılabilmesi için, düzenli deyimlerden yararlanılabilir.

Örnek 1.76

```
bil3@pardus ~ $ sed 's/[Yy]anlız/yalnız/g' test.txt
Bu parçada yalnız kelimesi yerine yanlışlıkla yalnız yazılmıştır.
yalnız, bu hata dışında yanlış kelimesi de yalnış yazılmıştır.
yalnız ve yalnış kelimelerini doğrusu ile düzeltin.
```

Varsayılan ayar olarak yapılan değişiklikler ekranda görüntülenir. Eğer değişiklikleri bir dosyaya kaydetmek isterseniz, yönlendirme operatöründen (>) yararlanabilirsiniz.

Örnek 1.77

```
bil3@pardus ~ $ sed 's/[Yy]anlız/yalnız/g' test.txt > deneme.txt
bil3@pardus ~ $ cat deneme.txt
Bu parçada yalnız kelimesi yerine yanlışlıkla yalnız yazılmıştır.
yalnız, bu hata dışında yanlış kelimesi de yalnış yazılmıştır.
yalnız ve yalnış kelimelerini doğrusu ile düzeltin.
```

1.5.5. head, tail Komutları

Dosyaların ilk ya da son bölümlerinin belirtilen sayıdaki satırlarını görüntüler. Herhangi bir seçenek girilmediği takdirde 10 satır görüntülenir.

Örnek 1.78

```
bil3@pardus ~ $ head satir.txt
Bu birinci satır
Bu ikinci
Bu üçüncü
Bu dördüncü
Bu beşinci
Bu altıncı
Bu yedinci
Bu sekizinci
Bu dokuzuncu
Bu onuncu
```

Örnek 1.78'deki komut, ilk baştaki 10 satırı görüntülemektedir. 20 satırdan oluşan bir dosyada son 3 satırı görüntülemek için Örnek 1.79'daki komut kullanılabilir.

Örnek 1.79

```
bil3@pardus ~ $ tail -3 satir.txt
Bu onsekizinci
Bu ondokuzuncu
Bu yirminci satırdır.
```

1.5.6. split Komutu

Dosyayı belirtilen boyutlara böler. Örneğin, satırlar dosyasını "**sf**" ile başlayan dörder satırlık dosyalara bölmek için ***-l*** seçeneği ile birlikte Örnek 1.80'deki komut kullanılabilir.

Örnek 1.80

```
bil3@pardus ~ $ split -l 4 satir.txt sf

bil3@pardus ~ $ ls sf*
sfaa  sfab  sfac  sfad  sfae
```

Örnek 1.80'de 20 satırdan oluşan "satir.txt" dosyası, herbiri dörder satırdan oluşan "sfaa", "sfab", "sfac", "sfad", "sfae" adlı 5 dosyaya bölünmüştür.

1.5.7. wc Komutu (Word Count)

Dosyadaki sözcük ya da satır sayısını görüntüler. Örneğin, bu komut, "deneme.txt" dosyasındaki satır sayısını görüntülemek için ***-l*** seçeneği ile birlikte Örnek 1.81'deki şekilde kullanılabilir.

Örnek 1.81

```
bil3@pardus ~ $ wc -l deneme.txt
20 deneme.txt
```

Sözcük sayısı için ***-w*** seçeneği ile birlikte kullanılır.

Örnek 1.82

```
bil3@pardus ~ $ wc -w deneme.txt
30 deneme.txt
```

Karakter sayısı için ise ***-c*** seçeneği ile birlikte kullanılır.

Örnek 1.83

```
bil3@pardus ~ $ wc -c deneme.txt
202 deneme.txt
```

1.6. Boru (*Pipe*) İşlemleri

Bir komutun çıktısını başka bir komuta yönlendirerek üzerinde işlemler gerçekleştirilebilir. Komut çıktısının bir dosyaya yönlendirilmesinde “>” ve “>>” operatörleri kullanılarak işlem yapılmaktaydı. Bir komut ile bir başka komut arasında yönlendirme işlemlerine ise, boru (pipe) işlemleri adı verilir. Bu işlem için boru (|) operatörü kullanılır. Bu karakter, kendisinden önce gelen komutun çıktısını alarak, işlenmek üzere kendisinden sonraki komuta aktarır.

Örneğin, kullanıcı ev dizinindeki dosyalara ait detaylı bilgileri listeleyip (**`ls -l`**), komut çıktısını yazıcıya **`lpr`** komutu ile gönderelim.

Örnek 1.84

```
bil3@pardus ~ $ ls -l ~ | lpr
```

Örnek 1.84’teki komut, kullanıcı ev dizininin (~) detaylı dosya bilgilerini alarak **`lpr`** komutuna aktarmaktadır. Bu bilgiler, **`lpr`** komutu ile yazıcıya gönderilir.

Örneğin, sistemdeki kullanıcı sayısını bulmak için öncelikle kullanıcıları listeleyip listedeki satır sayısını bulmak gerekir. **`who`** komutu, sistemdeki kullanıcıları listeler. **`wc`** komutu ise satır sayısını tespit eder. Bu iki komutu kullanarak, borulama işlemi yardımıyla kullanıcı sayısını bulmak için Örnek 1.85’teki komut dizgesi kullanılabilir.

Örnek 1.85

```
bil3@pardus ~ $ who | wc -l
3
```

Kullanıcının ev dizinindeki (~) dosya sayısını bulmak için ise, Örnek 1.86’daki gibi **`find`** komutu ile dosyaları listeleyip, komut çıktısını **`wc`** komutuna göndererek sonuca ulaşılabilir.

Örnek 1.86

```
bil3@pardus ~ $ find ~ -print | wc -l
5541
```

1.7. Temel Kabuk Programlama

1.7.1. Kabuk Programlamaya Giriş

Kabuk komutları, bir dosya içerisine yazılarak, bu komutların ardarda çalışması sağlanabilir. Bu komutların girdi ve çıktıları birbirleriyle ilişkilendirilerek belirli işlevleri gerçekleştirirler. Bu yapı ile oluşturulan ve hazırlanan komut dizisi, kabuk programı (*shell script*) olarak adlandırılır.

Temel programlamayı öğrenirken artık geleneksel hale gelen “Merhaba Dünya” çıktısını veren bir programın, kabuk için nasıl yazılacağı Örnek 1.87’deki adımlarda verilmiştir:

Örnek 1.87:

- Ev dizininde “merhaba” adlı bir dosya oluşturarak içerisine aşağıdaki komutu yazınız ve kaydederek çıkınız.

```
echo “Merhaba Dünya”
```

- Daha sonra Konsol uygulamasını açın ve dosyaya **chmod u+x** komutuyla çalıştırma hakkı veriniz.

```
bil3@pardus ~ $ chmod u+x merhaba
```

- Bu adımlardan sonra dosyayı aşağıdaki şekilde çalıştırabilirsiniz.

```
bil3@pardus ~ $ ./merhaba
Merhaba Dünya
```

- Yazılan programa çalıştırma hakkı vermeden de **sh** komutuyla çalıştırabilirsiniz.

```
bil3@pardus ~ $ sh merhaba
Merhaba Dünya
```

Kabuk programlarında ilk satırda;

```
#!/bin/bash
```

ifadesi, programın çalışacağı kabuğu beriltir. Eğer belirtilmezse, program bulunulan kabuk üzerinde çalıştırılır. Kabuk programının yazılım farklarından kaynaklanan hatalarla karşılaşmaması için programın çalıştırılacağı kabuk programı belirtilmelidir.

- # işaretinden sonra gelen ifadeler, açıklama satırı kabul edilir ve kabuk programı tarafından yorumlanmaz.

1.7.2. Değişkenler

Değişkenler (*variables*), bir programın çalışması boyunca, gerekli olan verileri saklayan ve gerektiğinde bunları değiştirmeye olanak sağlayan yapılardır. Değişkenler, bir programın merkezindeki kısımdır.

Değişkenler, programda kullanılan bir sayı değerini (tam veya ondalıklı sayı) ya da bir karakter dizgesini (string) tutabilir.

Değişkenler, Linux kabuğunda "=" ile tanımlanabilir.

Örnek 1.88

```
bil3@pardus ~ $ kullanici_adi="Fırat YÜCEL"
bil3@pardus ~ $ kullanici_mail="fyucel@ieee.org"
bil3@pardus ~ $ echo $kullanici_adi
Fırat YÜCEL
bil3@pardus ~ $ echo $kullanici_mail
fyucel@ieee.org
```

Örnek 1.88'de "kullanici_adi" ve "kullanici_mail" isminde iki adet değişken tanımı yapılmıştır. Daha sonra **echo** komutuyla bu değişkenlerin değerleri yazdırılmıştır.

- Değişkenler tanımlanırken "=" ile ifadeler arasında boşluk bulunmamasına dikkat ediniz.
- Değişken tanımlanırken değişken adının önünde herhangi bir sembol bulunmaz; fakat program içerisinde kullanırken değişken isimlerinin önüne **$** sembolünün getirilmesi gerekir.

1.7.3. Değer Okuma

Değişkenlere programın çalışması sırasında dışarıdan değer atanabilir. Bu işlem, **read** komutu ile gerçekleştirilir.

Örnek 1.89'daki program kodunu bir metin editöründe yazarak, dosya adını "**degeroku**" olarak verip kaydediniz.

Örnek 1.89

```
#!/bin/bash
echo "Kullanici adini giriniz:"
read kullanici
echo "Girdiğiniz kullanici: $kullanici
```

Daha sonra yazdığınız "**degeroku**" adlı programı **Konsole** uygulamasında Örnek 1.90'daki gibi çalıştırınız.

Örnek 1.90

```
bil3@pardus ~ $ chmod u+x degeroku
bil3@pardus ~ $ ./degeroku
Kullanici adini giriniz
fyucel
Girdiginiz kullanici: fyucel
```

1.7.4. Aritmetik İşlemler

Aritmetik işlemler için **let**, değişken tanımlamak için ise **declare** komutu kullanılabilir.

Örnek 1.91

```
01 #!/bin/bash
02
03 #Degisken tanımlamalari:
04
05 a=5
06 b=3
07
08 #Sonuc degiskenini tamsayi degisken olarak belirleme:
09
10 declare -i sonuc
11
12 #Dogrudan hesaplama:
13
14 sonuc=a+b
15 echo $sonuc
```

```
17 #'let' komutu ile hesaplama:
18
19 let "sonuc=$a+$b"
20 echo $sonuc
21
22 #Carpim islemini alma:
23
24 let "sonuc=$a*$b"
25 echo $sonuc
```

5 ve 6. satırlarda a ve b değişkenlerine değerler atanmış, 10. satırda ise “sonuc” değişkeni, tamsayı (integer) değişken olarak tanımlanmıştır.

Aritmetik işlemlerde iki yoldan işlem gerçekleştirilebilir. 14. satırda görüldüğü gibi doğrudan hesaplama yapılarak sonuc değişkenine atılabilir. Bu işlem sırasında değişkenlerin önüne **$** işaretleri kullanılmaz.

İkinci olarak let komutu ile de aritmetik işlem gerçekleştirilebilir. Bu durumda **declare** tanımlamasına gerek kalmaz. 19. satırda a ve b değişkenlerinin değeri toplanarak sonuc değişkenine atanmıştır. İşlem içerisindeki değişkenlerin önünde $ işareti bulunduğuna dikkat ediniz.

24. satırda ise **let** komutu kullanılarak çarpım işlemine örnek verilmiştir. Hesaplamalardan sonra echo komutuyla sonuç ekrana yazdırılmıştır.

sh komutu ile yazılan program doğrudan çalıştırıldığında, Örnek 1.92'deki çıktı elde edilir.

Örnek 1.92

```
bil3@pardus ~ $ sh sayisal
8
8
15
```

Örnek 1.93'deki gibi herhangi bir tanım yapılmadan, **(( ))** ile de aritmetik işlemler gerçekleştirilebilir.

Örnek 1.93

```
(( sonuc=$a-$b ))
echo $sonuc
```

1.7.5. Kontrol Yapıları

Kontrol işlemlerinde **test** komutu, **if-else** ve **case** deyimleri kullanılır.

1.7.5.1. test Komutu

test komutu, mantıksal bir ifadenin sonucunu bulur. Bash kabuğunda en son çalışan komutun sonucu **$?** ile öğrenilebilir. Eğer komut başarılı bir şekilde çalışmışsa 0 (sıfır) sonucunu, diğer durumlarda sıfırdan farklı bir değeri geri döndürür. Bu özellikten yararlanarak **test** komutunun sonucu da ekranda görüntülenebilir.

Örnek 1.94

```
bil3@pardus ~ $ test 5 -gt 3
bil3@pardus ~ $ echo $?
0
bil3@pardus ~ $ test 5 -gt 6
bil3@pardus ~ $ echo $?
1
```

Yukarıdaki örnekte test komutu, ***-gt*** (*greater than*) seçeneğiyle kullanılmıştır. Bu seçenek ile önce 5 sayısının 3'ten büyük olup olmadığı test edilmiş ve komut hatasız çalışarak sonuçta 0 (sıfır) bulunmuştur. Diğer test işleminde ise 5 sayısının 6'dan büyük olup olmadığı test edilmiş, 5 sayısı 6 dan büyük olmadığından sonuç hatalı bir şekilde 1 (bir) olarak geri döndürülmüştür.

Aritmetik, karakter dizileri ve dosya karşılaştırmalarında kullanılabilecek seçenekler Tablo 1.5'te özetlenmiştir.

Tablo 1.5: Test İşlemlerinde Kullanılan Seçenekler

Aritmetik		Karakter Dizileri		Dosya	
-eq	Eşit	**-z**	Boş	**-f**	Dosya var
-gt	Büyük	**-n**	Tanımlı	**-s**	Dosya boş değil
-lt	Küçük	**=**	Eşit	**-r**	Dosya okunabilir
-ge	Büyük eşit	**!=**	Farklı (Eşit değil)	**-w**	Dosya yazılabilir
-le	Küçük eşit			**-x**	Dosya çalıştırılabilir
				-h	Sembolik bağlantı
				-c	Karakter dosyası
				-b	Blok dosyası

1.7.5.2. If-Else Deyimi

If deyimi, mantıksal bir ilişkiyi sınayarak bir işlemin yapılmasına, yapılmamasına ya da başka bir işlemin gerçekleştirilmesine karar vermek için kullanılan deyimdir. *If*, "eğer" anlamındadır.

Genel kullanımı:

```
if [ koşul-1 ]; then
      koşul-1 doğru ise yürütülecek komutlar
elif [ koşul-2 ]; then
      koşul-2 doğru ise yürütülecek komutlar
..............
elif [ koşul-n ]; then
      koşul-n doğru ise yürütülecek komutlar
else
hiçbir koşul doğru değilse yürütülecek komutlar
```

fi

if deyiminde en az bir koşul bulunması zorunludur. Koşullar, birden fazla mantıksal deyimin birleşiminden de oluşabilir.

elif ve **else** durumları ise isteğe bağlı olarak kullanılır. Eğer ilk koşul sağlanmazsa, sınanması istenen diğer koşullar **elif** kısımlarında belirtilir. Eğer hiçbir koşulun sağlanmadığı durumda yapılması gereken bir işlem varsa, **else** kısmında bu işlem gerçekleştirilir.

if deyimi, deyimi oluşturan harflerin tersten yazılmasıyla oluşan **fi** komutu ile sonlandırılır. Bir if deyiminde, **if** komutu, ilk koşul, **then** ifadesi, koşul doğruysa çalışacak komut ve **fi** komutu zorunludur. Diğer kısımlar, gereksinimlere bağlı olarak kullanılabilir.

Örneğin, kullanıcı tarafından girilen bir sayının değeri 10'dan büyükse, "*Girilen sayı 10'dan büyük*"; 10'a eşitse, "*Girilen sayı 10'a eşit*"; diğer durumlarda "*Girilen sayı 10'dan küçük*" şeklinde ekrana mesaj yazdıran kabuk programını oluşturalım.

Örnek 1.95

```
01 #!/bin/bash
02 echo "Bir sayi giriniz:"
03 read sayi
04 if [ $sayi -gt 10 ]; then
05    echo "Girdiginiz sayi 10 dan buyuk";
06 elif [ $sayi -eq 10 ]; then
07    echo "Sayi 10";
08 else
09    echo "Sayi 10 dan kucuk";
10 fi
```

Örnek 1.95'te **if** koşulunda **$sayi** değişkeninin değerinin 10'dan büyük olması durumu sınanmaktadır. Koşul doğru olduğunda 5. satırdaki komut çalışır. **elif** kısmında 10'a eşit olma durumuna bakılmaktadır. Bu koşul doğru olursa, 7. satırdaki komut çalışır. **else** kısmında değişken değeri yukarıdaki koşullara uymadığında (değişken değerinin 10'dan küçük olması durumunda) 9. satırda çalışacak komut belirtilmiştir.

1.7.5.3. Case Deyimi

Bir veya birden fazla koşula bağlı olarak birçok seçenek arasında seçim yapılması gerektiğinde **case** deyimi kullanılır.

Genel kullanımı:

```
case değişken in
      seçenek-1)
            komutlar
      ;;
      seçenek-2)
            komutlar
      ;;
      ...
```

```
        *)
              komutlar
        ;;
    esac
```

Örneğin, kullanıcıya seçenekler sunan ve kullanıcı klavyeden "1" seçeneğini seçtiğinde komut satırı ekranını temizleyen, "2" seçeneğini girdiğinde kullanıcı ev dizinindeki dosyaları ayrıntılı olarak listeleyen, "3" seçeneğini girdiğindeyse sistem yöneticisi olarak oturum açan bir program hazırlayalım.

Örnek 1.96

```
01 #!/bin/bash
02 echo "Islemler Menusu"
03 echo "1-Ekrani temizle"
04 echo "2-Ev dizinini listele"
05 echo "3-Yonetici girisi yap"
06 echo "Secenek seciniz (1-3):"
07 read secenek
08
09 case $secenek in
10 1)
11    clear
12 ;;
13 2)
14    ls -l ~
15 ;;
16 3)
17    su -
18 ;;
19 *)
20    echo "Gecersiz secim"
21 esac
```

***** seçeneği, üstteki koşullardan herhangi biri gerçekleşmediğinde çalışır. Örnek 1.96'da kullanıcı tarafından 1 ile 3 arasında bir değer girilmediğinde, ekrana "Geçersiz seçim" uyarısı yazdırılmaktadır.

1.7.6. Döngüler

Belli sayıda veya belli bir koşula bağlı olarak tekrar etmesi istenen komutlar, bir kez yazılıp döngü içerisine konulduğunda, istenen sayıda veya istenen koşul gerçekleşene kadar tekrar eder. Komutların tekrarını sağlayan yapılara döngü (*loop*) adı verilir.

1.7.6.1. while Döngüsü

Belirli işlemlerin, bir koşulun varlığında tekrarlaması istendiğinde kullanılan yapıdır. while, *iken* anlamındadır. while'ın yanında belirtilen koşul **doğru *iken*** yapı içerisindeki komutların tekrarlamasını sağlar. Koşul yanlış olduğunda döngü sonlanır, **done** ifadesinden sonraki komutlara geçilir.

Genel kullanımı:

```
while koşul
do
      komutlar
done
```

➢ While yapısında dikkat etmeniz gereken husus, koşulu bir yerde yanlış yapmak ve döngüyü sonlandırmak için "komutlar" kısmında kontrol ifadesi kullanmaktır. Örneğin, koşul a değişkeninin değerinin 10'dan küçük olması ise, döngünün komutlar kısmında a değişkeninin değerini kontrollü şekilde değiştirmelisiniz. Aksi durumda, a değişkeninin değeri hep 10'dan küçük kaldığından, ***sonsuz döngü*** oluşur ve döngüdeki komutlar sürekli çalışır.

Örneğin, 1'den 10' a kadar olan sayıları ekrana while döngüsü kullanarak yazdıralım.

Örnek 1.97

```
01 #!/bin/bash
02 sayac=1
03 while [ $sayac -le 10 ]
04 do
05    echo $sayac
06    (( sayac = $sayac + 1 ))
07 done
```

Örnek 1.97'de ilk satır # karakteri ile başladığından yorumlanmaz. İkinci satıra geçilir. Bu satırda "sayac" adında bir değişken tanımlanmış ve ilk değeri 1 olarak verilmiştir. 3. satırda **while** döngüsü koşulu sınanır. Koşula göre "sayac" değişkeninin değeri 10'dan küçük ve 10'a eşitse **do** ile **done** ifadeleri arasındaki komutlar tekrar edilecektir. İlk durumda "sayac" değişkeninin değeri 1 olduğundan döngü komutları çalıştırılır. 5. satırdaki ilk döngü komutu, "sayac" değişkenini ekrana yazar. 6. satırda ise "sayac" değişkeninin değeri 1 artırılır. **done** ifadesine gelindiğinde tekrar **while** komutunun bulunduğu 3. satıra

dönülür ve koşul tekrar sınanır. İkinci sınamada, "sayac" değişkeninin değeri 6. satırdaki komut ile 1 artırılarak 2 olduğundan ve koşulu sağladığından tekrar döngüye girilir ve döngü komutları çalıştırılır. 3, 4, 5,.., 9 ve 10 değerleri için döngü komutları çalışır. 10. döngüden sonra, "sayac" değişkeni 11 değerini aldığından **`while`** komutundaki koşulu geçemez ve döngü sonlanır. Döngüden sonra herhangi bir komut bulunmadığından programdan çıkılır.

Tablo 1.6: Örnek 1.97'deki while döngüsünün çalışma adımları

Döngü Sayısı	*sayac* Önceki Değer	*sayac* Yeni Değer
İlk durum	-	1
1. Döngü	1	2
2. Döngü	2	3
3. Döngü	3	4
4. Döngü	4	5
5. Döngü	5	6
6. Döngü	6	7
7. Döngü	7	8
8. Döngü	8	9
9. Döngü	9	10
10. Döngü	10	**11**
Döngüye girilmez	11	-

1.7.6.2. for Döngüsü

Belirli komutların, belli sayıda ya da belli koşullarda tekrarlamasını sağlar. Döngü sayısı programcı tarafından belirlenebilir. Kontrol mekanizması yapı üzerinde bulunduğundan ayrıca sayaç oluşturulmasına gerek duyulmaz.

Genel kullanımı:

1. Kullanım:

```
for (( ifade1; ifade2; ifade3 ))
do
        ifade2 yanlış olana kadar
        do ve done arasındaki tüm
        komutları tekrarlar.
Done
```

***ifade1*:** Döngü sayaç değişkeninin ilk değeri
***ifade2*:** Döngü koşulu
***ifade3*:** Sayaç değişkeninin değiştirildiği aritmetik ifade

2. Kullanım:

```
for değişken in değerler
do
      komutlar
done
```

Örneğin, 1 ile 10 arasındaki sayıları bu kez de for döngüsü yardımıyla ekrana yazdıralım.

Örnek 1.98

```
01 #!/bin/bash
02 for (( i = 1; i <= 10; i++ ))
03 do
04    echo $i
05 done
```

Örnek 1.98'de 2. satırda for deyimi içerisinde 3 ifade bulunmaktadır. İlk ifade (i = 1), i olarak adlandırılan sayaç değişkenine ilk değer olarak 1 değerini atamaktadır. İkinci ifade (i <= 10), döngünün çalışma koşulunu gösterir. Buradaki ifade doğru iken döngü çalışır. Üçüncü ifade (i++) ise, i değişkeninin her seferinde 1 artacağını göstermektedir.

1 ile 5 arasındaki sayıları ekrana yazdıran programı, for döngüsünün ikinci kullanım şekliyle yapalım.

Örnek 1.99

```
01 #!/bin/bash
02 for i in 1 2 3 4 5
03 do
04    echo $i
05 done
```

Örnek 1.99'da **in** ifadesinden sonra gelen değerler sırasıyla i değişkenine atanarak her seferinde **do** ile **done** arasındaki komut işletilmektedir. Değerler bittiğinde döngü sona erer.

1.7.7. Kabuk Fonksiyonları

Fonksiyonlar, bir program parçasını blok haline getirerek isimlendirmeye ve gerektiği yerde ismi ile çağırarak kullanmaya yarar. Programların kısa ve anlaşılır olmasında çok yararlı ve kullanışlı yapılardır.

Genel kullanımı:

```
fonksiyon_adi()
{
      komutlar
}
.......
```

Tanımlanmış bir fonksiyonu çağırırken ise;

```
fonksiyon_adi
```

kullanılır.

Örneğin, kullanıcının klavyeden iki sayı değerini toplayarak sonucu ekrana yazdıran bir fonksiyon aşağıdaki gibi oluşturulabilir.

Örnek 1.100

```
01 topla()
02 {
03        sayi1=$1
04        sayi2=$2
05        (( sonuc = $sayi1 + $sayi2 ))
06        echo $sonuc
07 }
08
09 topla 5 6
```

Örnek 1.100'de "topla" adında bir fonksiyon tanımlanmıştır. Program ilk olarak fonksiyonun bittiği noktadan (9. satırdan) itibaren çalışmaya başlar. 9. satırda fonksiyon çağrılırken, 5 ve 6 sayıları parametre olarak fonksiyona gönderilmiştir. Bu parametreler (**$1** ilk parametre, **$2** ikinci parametre,...), fonksiyon içerisinde 3. ve 4. satırlarda sayi1 ve sayi2 değişkenlerine atanmıştır. 5. satırda sayi1 ve sayi2 değişkenlerindeki değerlerin toplamı alınmış, 6. satırda ise sonuç ekrana yazdırılmıştır.

UYGULAMA FAALİYETİ

Bu uygulama faaliyetinde, açık kaynak işletim sisteminde kabuk işlemlerini öğrenerek uygulayacaksınız.

İşlem Basamakları	Öneriler
➢ Kök dizinde 'e' karakteri ile başlayıp 'c' karakteri ile biten tüm dosya ve dizinleri ayrıntılı olarak listeleyiniz.	➢ ls komutu ve seçeneklerini kullanınız. ➢ Özel sembollerden yararlanınız.
➢ Kullanıcı ev dizini (~) altında **Belgeler** adında bir dizin oluşturarak, ev dizini altındaki tüm belgeleri bu dizin içerisine kopyalayınız.	➢ mkdir ve copy komutlarını kullanınız. ➢ Özel sembollerden yararlanınız.
➢ Bir önceki adımda oluşturmuş olduğunuz Belgeler dizinini gz formatında arşivleyiniz.	➢ gzip komutunu kullanınız.
➢ Oluşturduğunuz Belgeler dizinindeki bütün dosya ve dizinleri ayrıntılı olarak **liste.txt** dosyasına listeleyiniz.	➢ ls komutunu ve yönlendirme operatörlerini kullanınız. ➢ Özel sembollerden yararlanınız.
➢ Kate editörünü kate komutu ile çalıştırarak, komut satırından ilgili süreci sonlandırmayı deneyiniz.	➢ ps komutu ile kill veya killall komutlarını kullanabilirsiniz.
➢ Bilgisayarınızda, içerisinde "Pardus" kelimesi geçen dosyaları bulan komut dizgesini yazınız.	➢ grep komutunu kullanınız. ➢ Düzenli deyimleri kullanınız.
➢ Ekrana 5 kez alt alta isminizi yazdıran kabuk programını; • Döngü komutu kullanmadan • Döngü komutu kullanarak hazırlayınız.	➢ while ve for döngü deyimlerini kullanabilirsiniz.
➢ İki sayının çarpımını alarak ekrana yazdıran bir fonksiyon hazırlayınız.	➢ (1.7.7) Fonksiyon tanımlama konusuna göz atın. Fonksiyon çağrısı şu şekilde olabilir: `carpim 5 3`

ÖLÇME VE DEĞERLENDİRME

Öğrenme faaliyetinde kazandığınız bilgileri ölçebileceğiniz kısma geldiniz. Bu bölümde yer alan ölçme sorularını dikkatlice okuyarak cevaplandırınız.

Aşağıdaki sorularda verilen boşlukları uygun şekilde doldurunuz.

1. Bulunulan dizindeki tüm dosya ve dizinleri (gizli olanlar dahil) ayrıntılı olarak listelemek için _____ komutu kullanılır.

2. Bir üst dizine çıkmak için _____ komutu kullanılır.

3. Bilgisayarda çalışan süreçleri ayrıntılı olarak _____ komutuyla görebiliriz.

4. Bir dizin ve içerisindekileri tümüyle silmek için _____ komutu kullanılır.

5. A karakteriyle başlayan ve içerisinde rakam olmayan sözcükleri bulmak için _____ düzenli deyimi kullanılır.

Aşağıdaki soruların doğru cevabını verilen seçeneklerden bularak, doğru seçeneği belirleyiniz.

6. Kabuk nedir?
   A) Grafik arayüzdür.
   B) Dosya ve dizinlere verilen addır.
   C) Komut istemcisidir.
   D) İşletim sistemi çekirdeğidir.

7. Aşağıdaki komutlardan hangisi bir süreci PID numarasıyla sonlandırır?
   A) ps B) kill C) killall D) clear

8. Aşağıdaki karakterlerden hangisi, bir dosyanın sonuna ekleme yapmak için kullanılabilecek yönlendirme karakteridir?
   A) > B) -> C) | D) >>

9. Boru (*pipe*) işlemi ne amaçla yapılır?
   A) Bir dosyayı diğer bir dosyaya bağlamak için
   B) Dosyaları birbirine eklemek için
   C) Bir komutun çıktısını başka bir komuta yönlendirmek için
   D) Komutları aynı anda çalıştırmak için

10. Aritmetik karşılaştırma işleminde “eşittir” anlamında kullanılan seçenek aşağıdakilerden hangisidir?
    A) *–eq* B) *–gt* D) != E) *-lt*

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Test içinde cevaplandıramadığınız, yanlış cevaplandırdığınız veya kendinizi bilgi bakımından eksik hissettiğiniz sorular için bilgi sayfalarına tekrar dönüp öğrenme faaliyetini gözden geçirmeniz önerilir.

ÖĞRENME FAALİYETİ-2

AMAÇ

Açık kaynak kodlu işletim sisteminde dosya ve dizin paylaşım işlemlerini gerçekleştirebileceksiniz.

ARAŞTIRMA

- Yerel alan ağı (LAN) hakkında araştırma yapınız.
- Açık kaynak kodlu işletim sistemleri ile Windows işletim sistemi arasında dosya, dizin ve kaynakların paylaşımı gerçekleştirilebilir mi? Araştırınız.
- İnternet üzerinden dosya paylaşım yöntemleri (p2p, bittorrent,…) ve programları hakkında araştırma yapınız.
-

2. DOSYA VE DİZİN PAYLAŞIMI

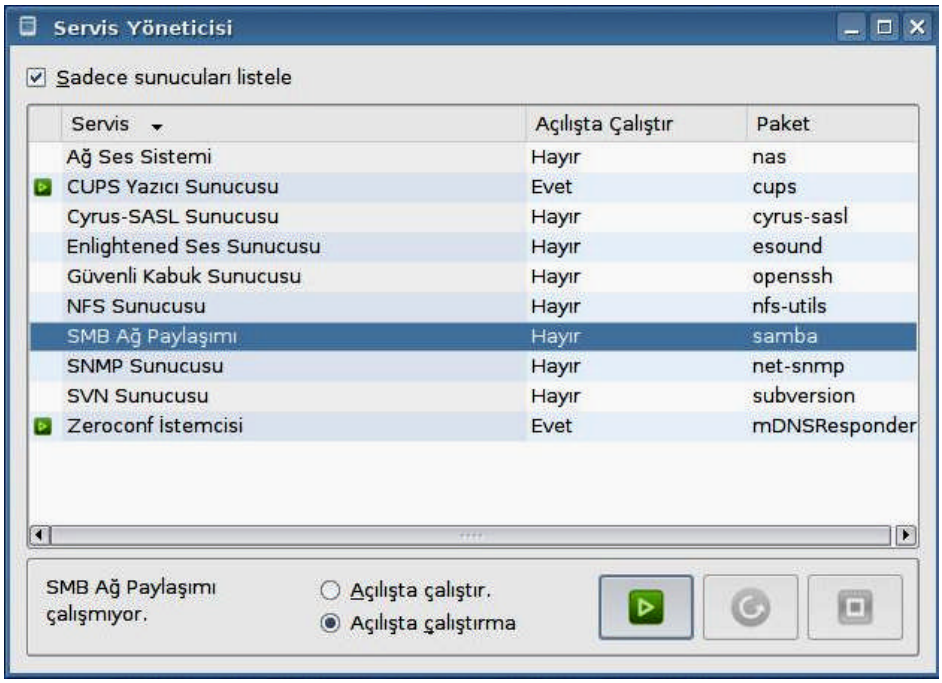
2.1. Dosya ve Dizin Paylaşımı

Dosya ve dizinlerin ağdaki bilgisayarlara paylaştırılmasında iki durumdan bahsedilebilir: Birincisi, bilgisayardaki dosyaları paylaşıma açma; ikincisi ise ağa bağlı bilgisayarlardaki paylaştırılmış dosyalara erişimdir.

2.1.1. Dosya ve Dizinlerin Paylaşıma Açılması

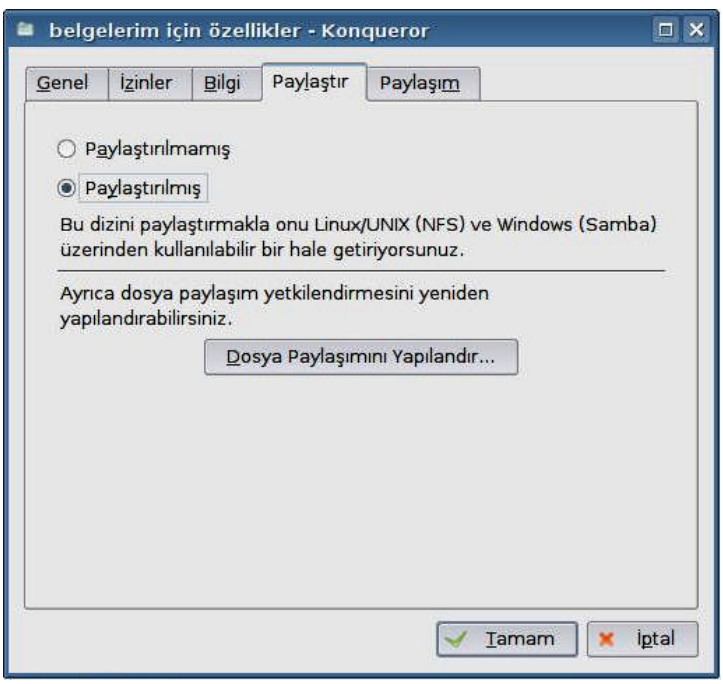
Pardus üzerindeki herhangi bir dosya veya dizin, ağa bağlı diğer makinelerden erişim sağlamak üzere paylaşıma açılabilir. Linux üzerindeki dosyaların Windows kullanıcıları tarafından paylaşılabilmesi için ayrıca Samba adı verilen servisin ayarlanmasına ve gerekli dizinlerin paylaşıma açılmasına ihtiyaç vardır.

Dosya ve dizinlerin ağda paylaşıma açılmasından önce Samba servisinin başlatılması gerekir. Bunun için Pardus ana menüsünden ***TASMA→Sistem Seçenekleri→Servis Yöneticisi*** programını çalıştırmalısınız.



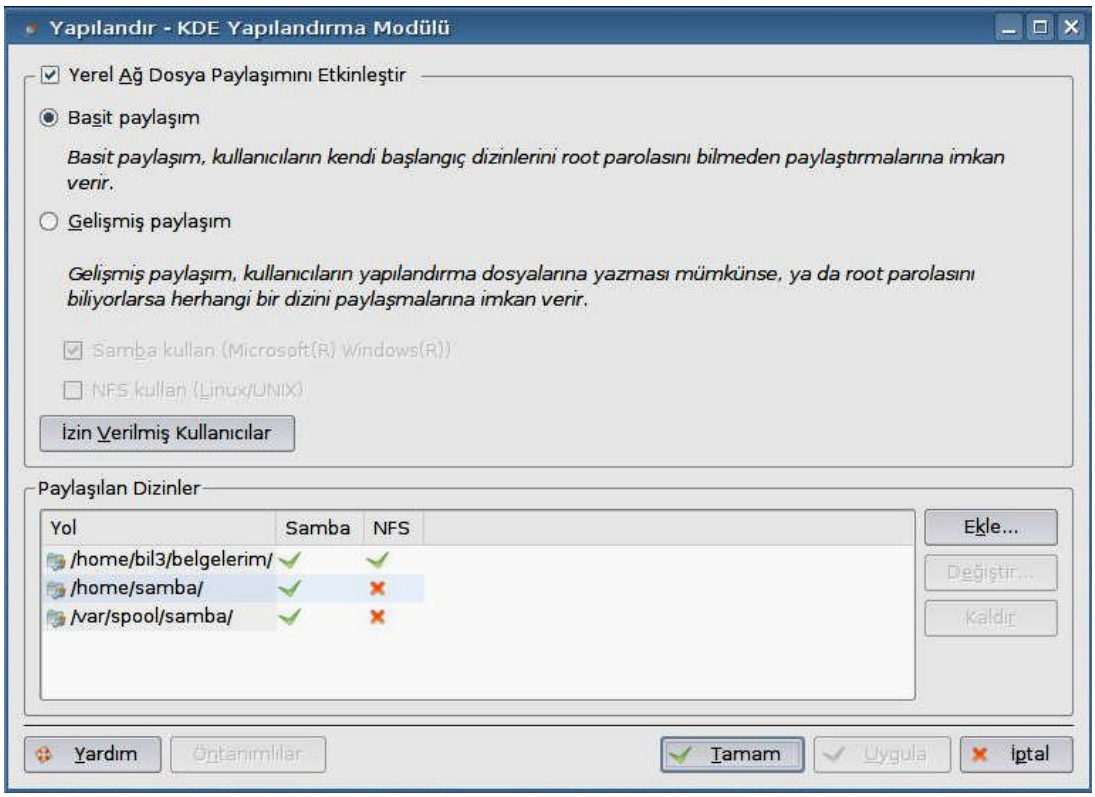
Şekil 2.1: Servis yöneticisi

Şekil 2.1'deki pencerede "**SMB Ağ Paylaşımı**" listeden bulunarak çalıştırılır. Ardından paylaştırılmak istenen dosya veya dizine sağ tıklanarak karşınıza gelen menünün en altındaki "**Paylaştır**" seçeneği seçilir. Eğer bilgisayar her açıldığında bu işlemi yapmak istemiyorsanız, "**Açılışta çalıştır**" seçeneğini seçebilirsiniz.



Şekil 2.2: Dosya/dizin paylaştırma ayarları

Şekil 2.2'deki **Paylaştır** penceresinde, "***Paylaştırılmış***" seçeneği seçilerek işlem tamamlanır. İleri düzeyde ayarlamalar gerçekleştirmek için sistem yöneticisi (*root*) hesabı ile Dosya Paylaşımını Yapılandır... düğmesi tıklanarak, sistem yöneticisi parolasıyla, Şekil 2.3'teki pencerede ayarlar gerçekleştirilebilir.



Şekil 2.3: Paylaşım yapılandırma

Eğer dosya ve dizinlere basit paylaşım verilirse, kullanıcıların kendi başlangıç dizinlerini sistem yöneticisi (*root*) parolasına gereksinim duymadan paylaştırmaları sağlanır. Gelişmiş paylaşım seçeneğinde ise yalnızca izin verilen kullanıcılar tarafından belirli paylaşımlar gerçekleştirilebilir. Hangi kullanıcıların paylaştırma hakkına sahip olduğunu belirlemek için "İzin Verilmiş Kullanıcılar" düğmesine tıklanır. Açılan pencerede kullanıcı ya da grup ismi belirlemek mümkündür.

Dosya ve dizin izinleri de erişim açısından önemlidir. Bu nedenle, "İzinler" sekmesine geçilerek buradaki izinlerin diğerleri için görünür veya değiştirilebilir hale getirildiğinden emin olmalısınız.

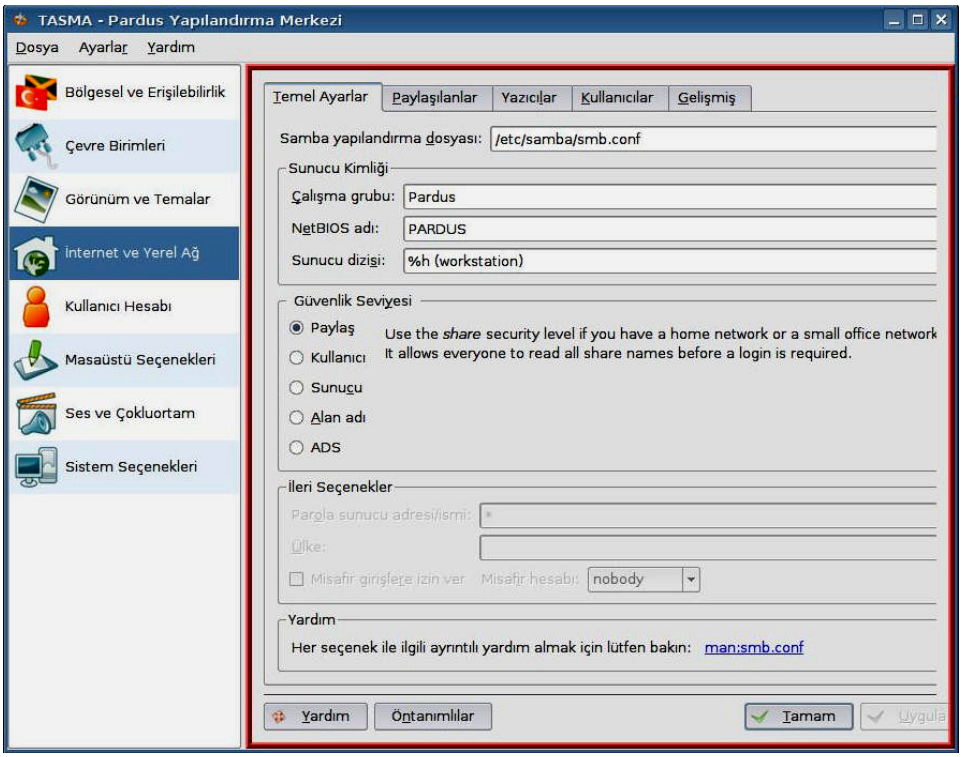
Şekil 2.4: Dosya/dizin erişim izinleri

Burada açıklanan işlemler, basit bir ağ paylaşımı için yeterlidir. Daha ileri düzey seçenekler, ***TASMA→İnternet ve Yerel Ağ→Samba*** seçeneği seçilerek gelen ekranda, **sistem yöneticisi (*root*) parolasıyla** ayarlanabilir.

2.1.2. Samba Ayarları

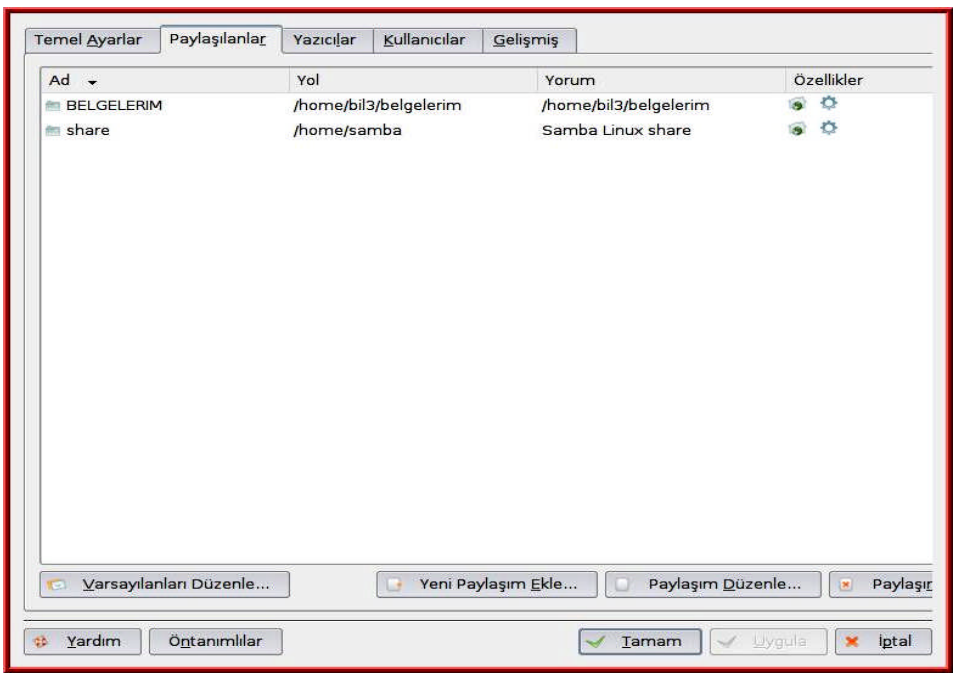
Samba, Linux ve Unix işletim sistemleri ile Windows NT ve Windows 9x işletim sistemleri arasındaki iletişimi sağlayan bir ağ sunucusu uygulamasıdır. Ayrıca, Pardus kurulu bir bilgisayardan diğerine de Samba kullanılarak paylaşım yapılabilir.

Samba programını, ***TASMA→İnternet ve Yerel Ağ→Samba*** yolundan çalıştırabilirsiniz. Ekrana gelen Şekil 2.5'teki **Samba Temel Ayarlar** penceresinde, bilgisayarın çalışma grubu ve bilgisayar adı bilgileri ile paylaşım güvenlik seviyesi sistem yöneticisi (root) tarafından ayarlanabilir.



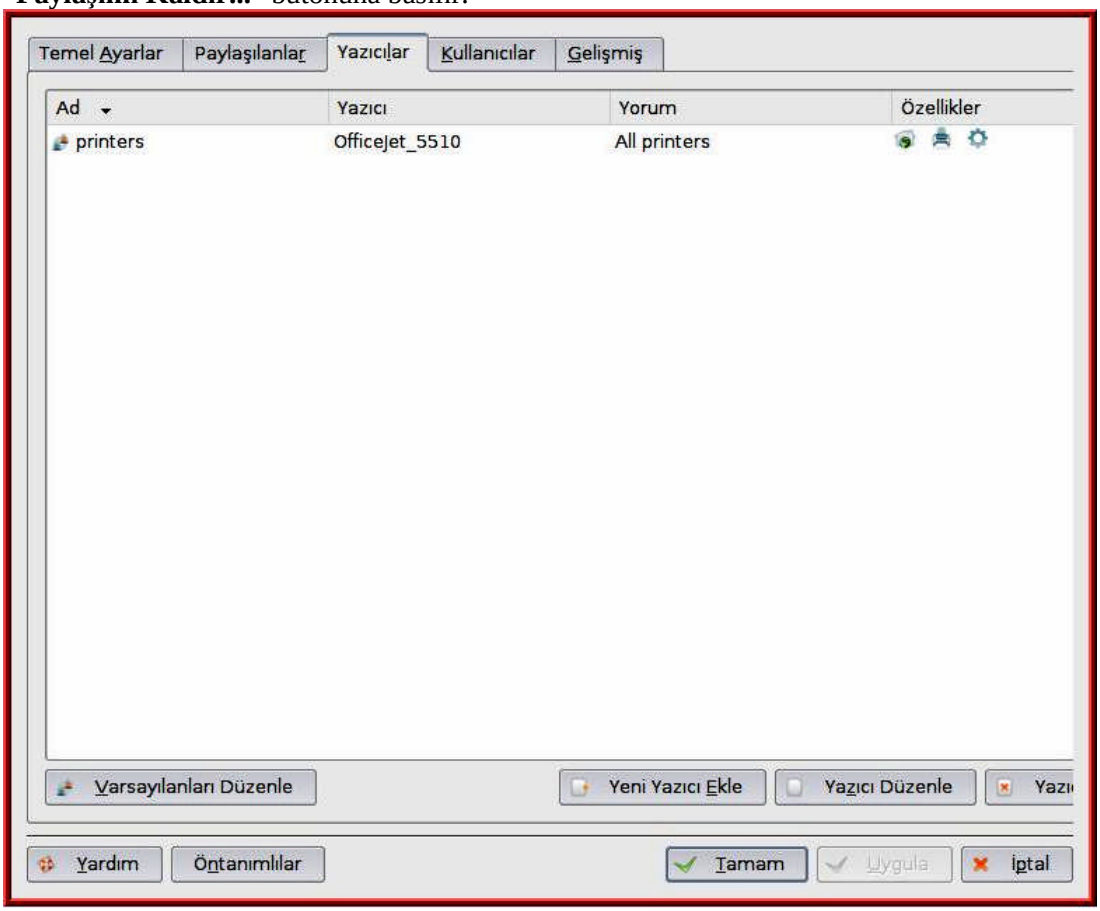
Şekil 2.5: Samba temel ayarları

Samba programının **Paylaşılanlar** sekmesinde paylaşılan dizin ve dosyaları görmek ve bunlar üzerinde işlemler yapmak mümkündür.



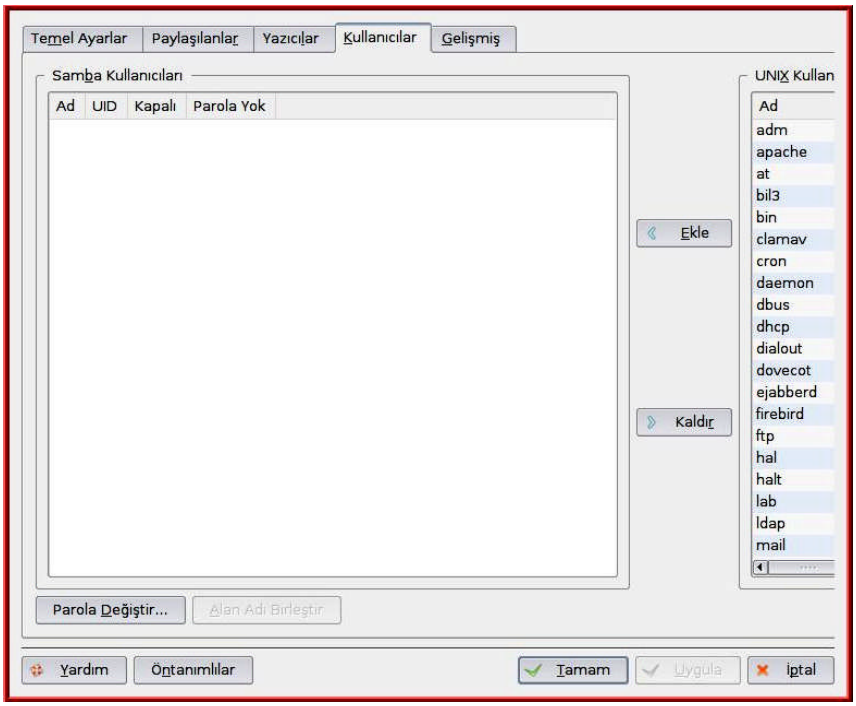
Şekil 2.6: Samba paylaşılanlar sekmesi

Yeni bir paylaşım eklemek için "**Yeni Paylaşım Ekle...**", listedeki paylaşımın ayarlarında değişiklik yapmak için "**Paylaşım Düzenle...**" ve paylaşımı kaldırmak için ise "**Paylaşımı Kaldır...**" butonuna basılır.



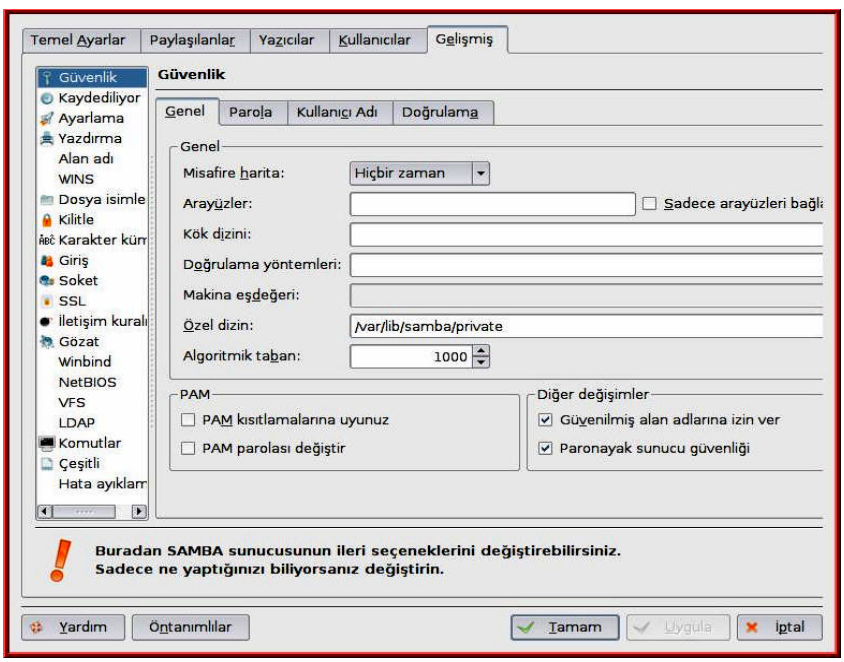
Şekil 2.7: Samba yazıcılar sekmesi

Yazıcılar sekmesinde ise paylaştırılan yazıcılar görüntülenir. Bu yazıcılara Windows makinelerden erişmek mümkündür. Bu pencerede yeni yazıcı eklenebilir, yazıcılarla ilgili ayarlar düzenlenebilir ya da yazıcı paylaşımdan kaldırılabilir.



Şekil 2.8: Samba kullanıcılar sekmesi

Şekil 2.8'deki **Kullanıcılar** sekmesinde, Samba protokolüyle bilgisayara erişecek kullanıcılar, erişemeyecek kullanıcılar ve bu kullanıcıların parolaları tanımlanabilir. Eğer kullanıcı erişim kısıtlaması getirilmeyecekse, bu pencerede değişiklik yapılmasına gerek yoktur.



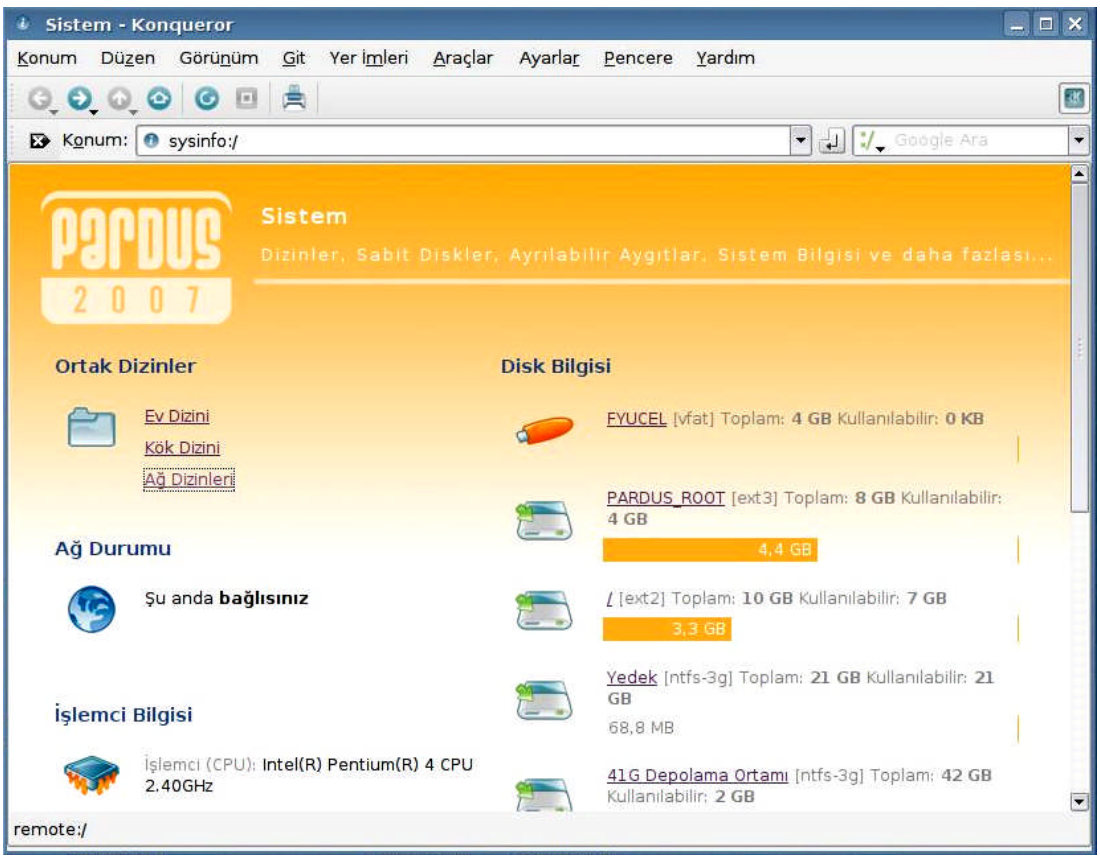
Şekil 2.9: Samba gelişmiş sekmesi

Şekil 2.9'daki **Gelişmiş** sekmesinde ise ileri düzey sistem yöneticilerinin yapabileceği çeşitli ayarlar bulunur. Buradan çeşitli güvenlik seçeneklerine erişebilirsiniz.

2.2. Paylaştırılan Dosyalara Erişim

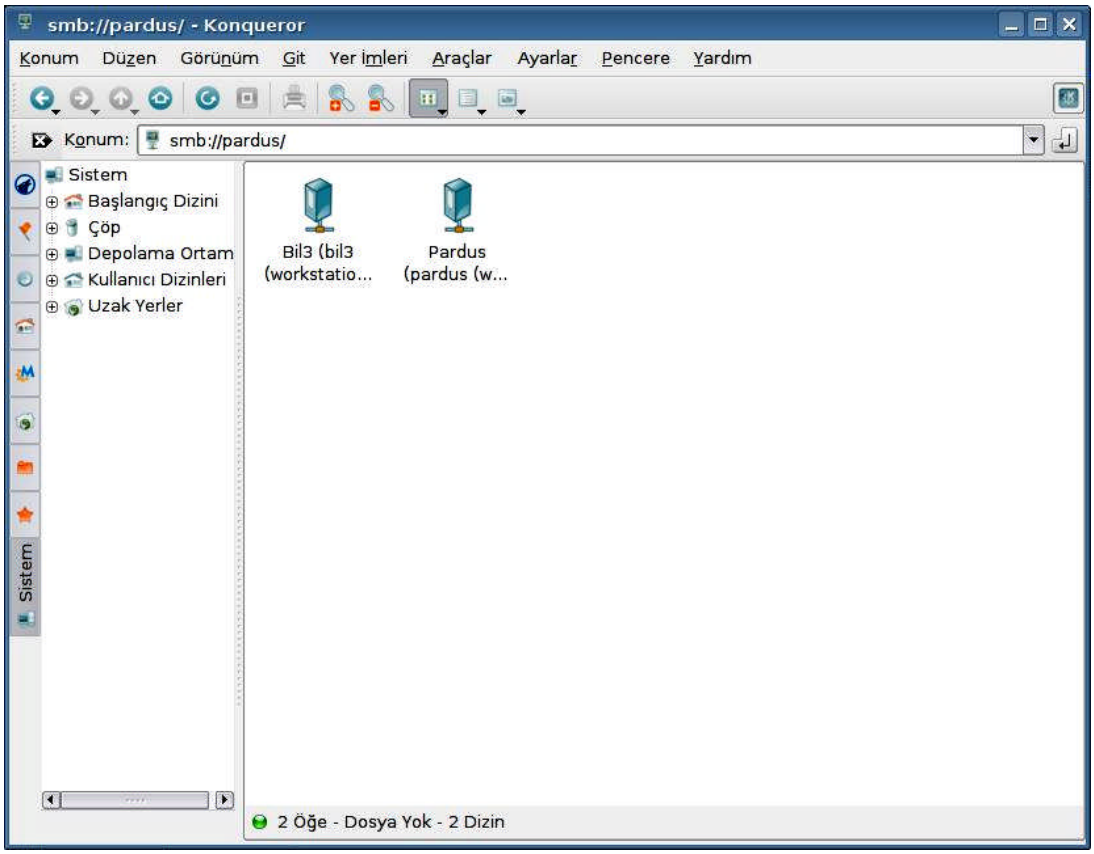
Paylaştırılan dosyalara erişim için aşağıdaki adımlar uygulanır:

- Servis Yöneticisinden **Samba Ağ Paylaşımı** servisi etkinleştirilir.
- **Konqueror** gözatıcısında adres satırına **smb:/** yazılır. Bir diğer yöntem ise masaüstünden **Sistem** programını çalıştırarak Şekil 2.10'daki pencerenin sol tarafından **Ağ Dizinleri**ni seçmektir.



Şekil 2.10: Sistem penceresi

- Bu adımdan sonra gelen pencerede **Samba Payları**'nı seçin. Şekil 2.11'deki pencereden ağdaki diğer makinelere erişmek mümkündür.



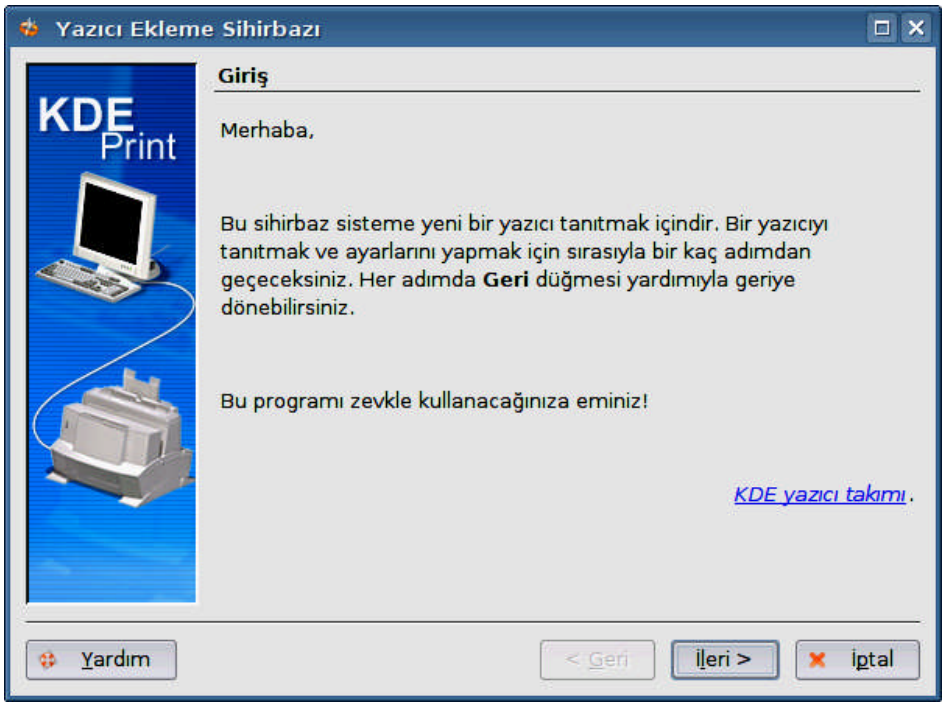
Şekil 2.11: Samba payları

2.3. Paylaştırılan Yazıcılara Erişim

Pardus üzerinden ağdaki bir yazıcıya erişim mümkündür. Windows yüklü bir makineye bağlı olan yazıcıya da erişilebilir. Bu işlem için aşağıdaki adımları uygulamanız yeterlidir.

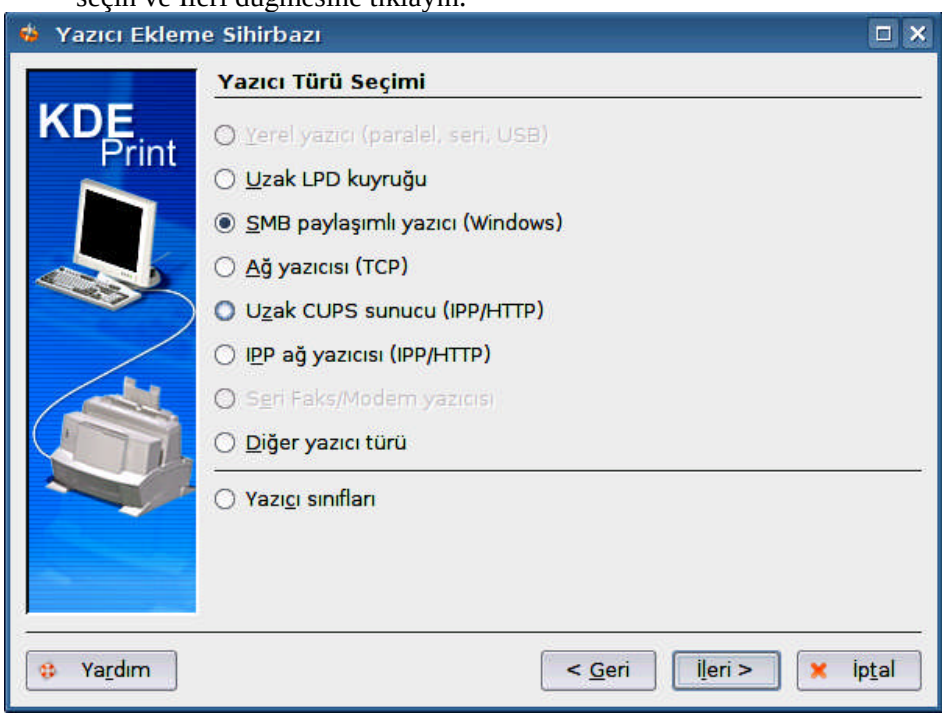
- **TASMA Pardus Yapılandırma Merkezi**'ni açınız. Sol taraftaki menüden **Çevre Birimleri** sekmesini seçiniz.

- Pencerede **Yazıcılar** üzerine tıklayınız. Sol üstteki **Ekle** menüsünden ***Yeni Yazıcı/Sınıf...*** seçeneğini seçiniz. Şekil 2.12'deki yazıcı ekleme sihirbazı ile karşılaşacaksınız.



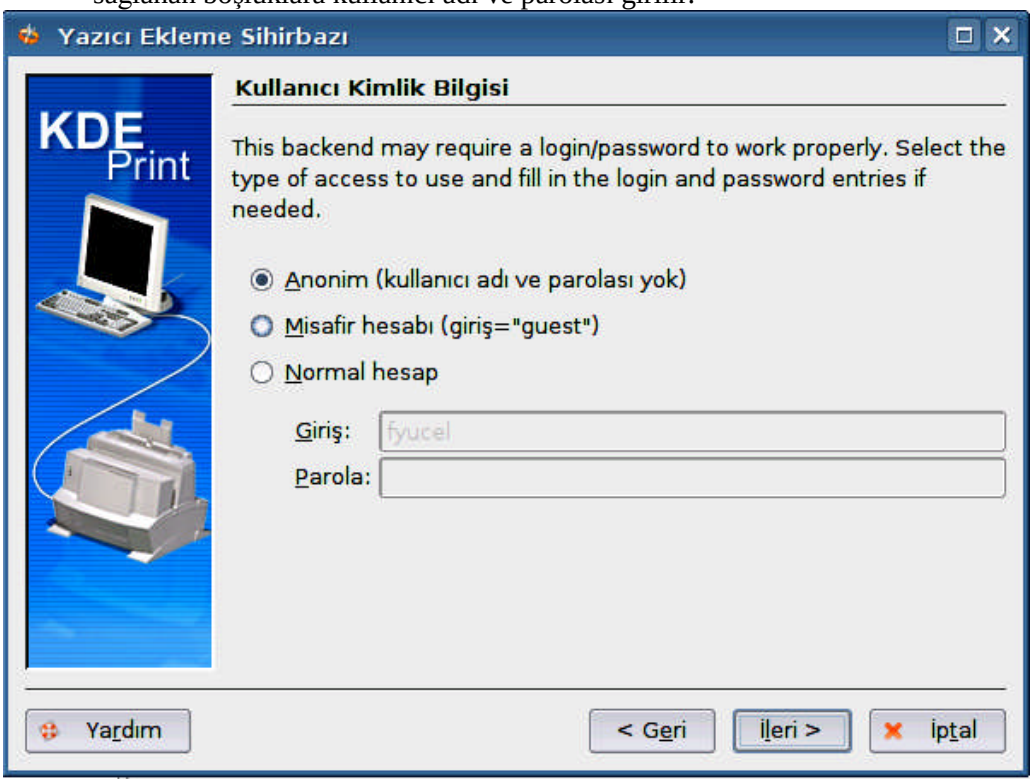
Şekil 2.12 :Yazıcı ekleme sihirbazı (1. adım)

- İkinci adımda (Şekil 2.13) "**SMB Paylaşımlı Yazıcı (Windows)**" seçeneğini seçin ve İleri düğmesine tıklayın.



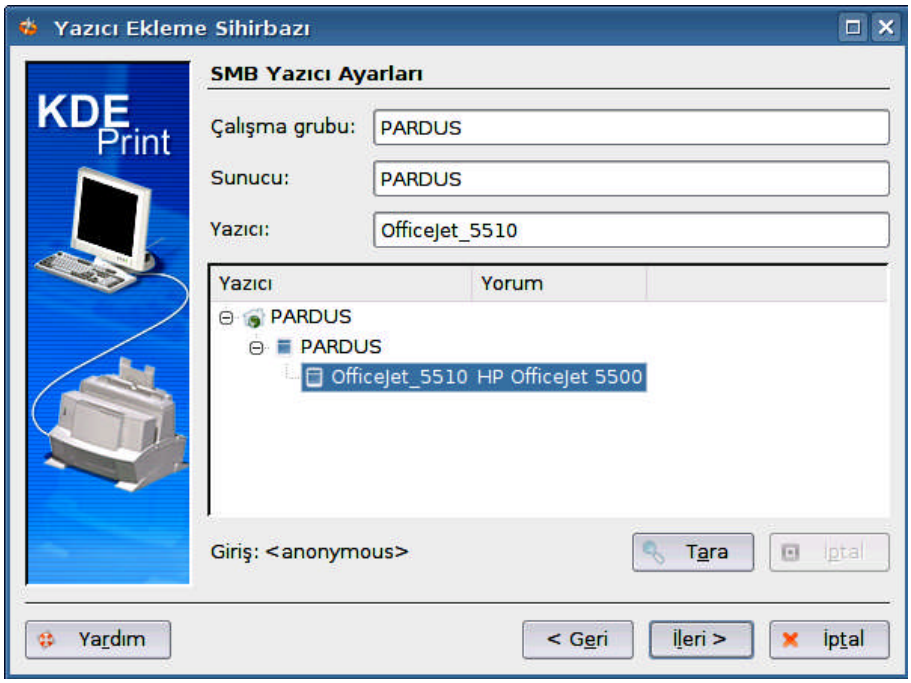
Şekil 2.13: Yazıcı ekleme sihirbazı (2. adım)

➢ Şekil 2.14'teki pencerede kullanıcı kimlik bilgisi istenir. Eğer bu yazıcı herkese açıksa, "**anonim giriş**" tanımlanabilir. Bir misafir hesabı gerekiyorsa, bu durumda "**Misafir hesabı**"nı seçin. Kimi durumlarda bir kullanıcı hesabı ve parolasının verilmesi şarttır, bu durumda "**Normal hesap**" seçeneği seçilerek sağlanan boşluklara kullanıcı adı ve parolası girilir.



Şekil 2.14: Yazıcı ekleme sihirbazı (3. adım)

➢ Bu adımda ise ağ taranır ve yazıcı listesi çıkarılır. Şekil 2.15'teki pencerede **Tara** düğmesine tıklanarak ağa bağlı bütün yazıcıları görmek mümkündür.



Şekil 2.15: Yazıcı ekleme sihirbazı (4. adım)

- Son olarak yazıcı modelinin seçildiği Şekil 2.16'daki pencerede "**Düz yazıcı**" seçeneği tıklanır. Böylece uzaktaki yazıcı için herhangi bir sürücü yüklemeye gerek duyulmaz. Uzaktaki yazıcı bu noktadan itibaren tüm masaüstü uygulamaları tarafından tanınacaktır.

Şekil 2.16: Yazıcı ekleme sihirbazı (5. adım)

2.4. Dosya Paylaşım Programları

Yerel ağ üzerinde dosya paylaşım işlemleri yapılabildiği gibi, en büyük ağ olan İnternet üzerinde de dosya paylaşımını sağlayan bazı yazılımlar, Pardus ile birlikte gelmiştir. Apollon ve Ktorrent programları bunlardan bazılarıdır.

2.4.1. Apollon ile Dosya Paylaşımı

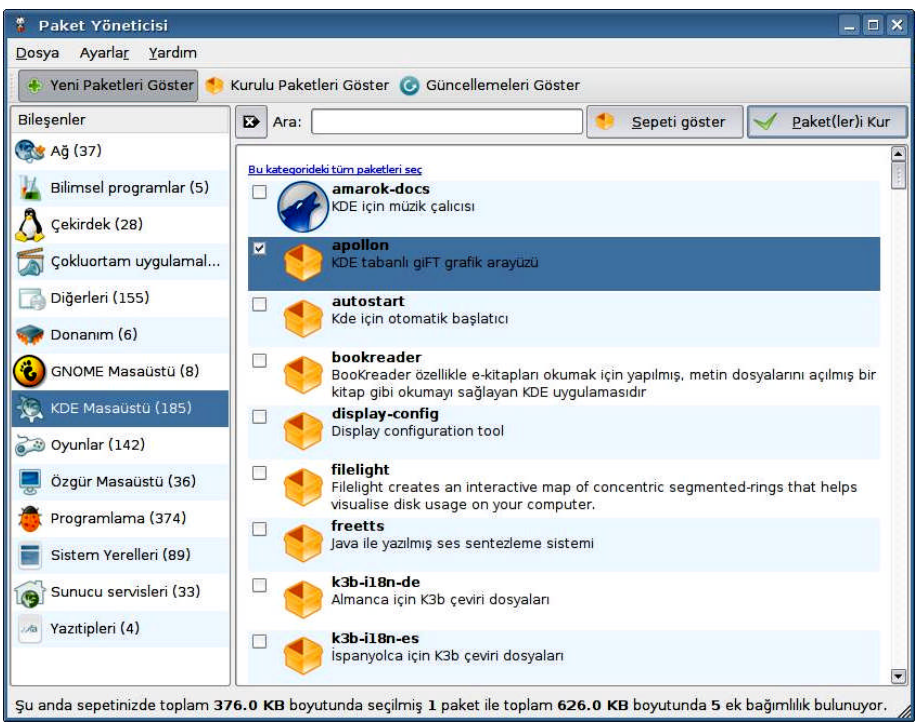
Apollon programı, İnternet üzerindeki pekçok dosya paylaşım ortamında arama yaparak dosyaları paylaşabileceğiniz p2p istemcisi bir uygulamadır. Bu program; Ares, Fasttack, Gnutella ve OpenFT paylaşım ağlarına erişim sağlayabilir.

- Bu programlar yardımıyla yasadışı dosya indirmeniz suçtur. Bu nedenle yalnızca temin edilmesi ve kullanılması suç olmayan dosyaları indirmelisiniz.

2.4.1.1. Apollon Programının Bilgisayara Kurulması

Apollon, Pardus paket deposu içerisinde bulunan bir yazılım olduğundan, buradan kolayca kurulum gerçekleştirilebilir. Kurulumu gerçekleştirmek için aşağıdaki adımları uygulayın:

- Pardus ana menüsünden **Paket Yöneticisi (Yazılım Ekle ve Kaldır)** programını çalıştırın.

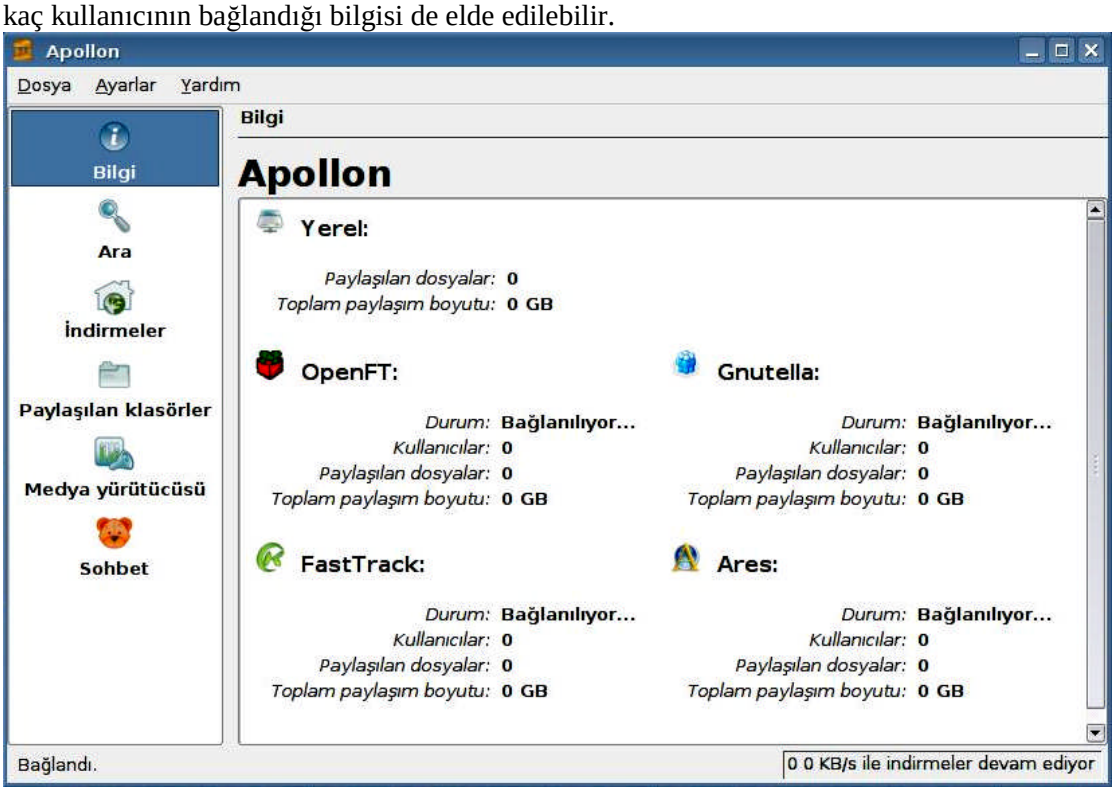


Şekil 2.17: Paket yöneticisi

- Ekrana gelen Şekil 2.17'deki pencerede, **Yeni Paketleri Göster** sekmesindeyken, **Ara** kutucuğuna "*Apollon*" yazın ya da solda yer alan seçeneklerden **KDE Masaüstü**'nü seçerek listeden **Apollon** programını bulunuz.
- Apollon programının yanındaki kutucuğu işaretleyerek, pencerenin sağ üst köşesinde yer alan **Paket(ler)i Kur** düğmesini tıklayın ve kurulumu başlatınız.

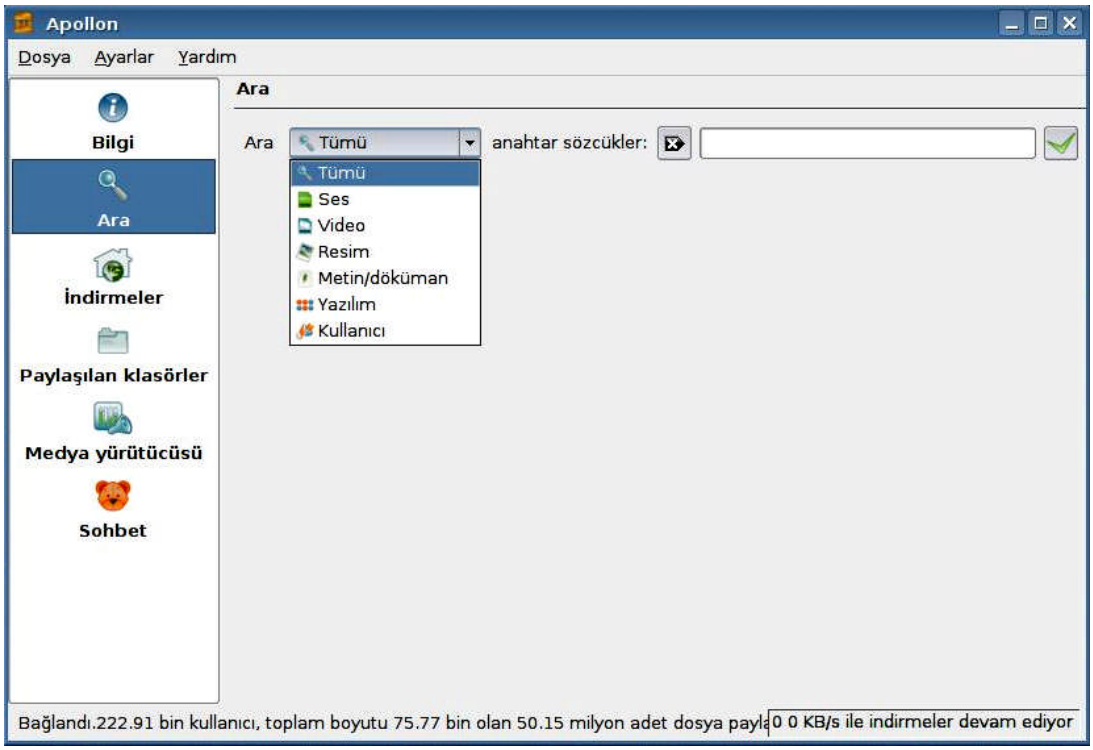
2.4.1.2. Apollon ile Arama Yapma

Apollon programını başlatmak için Pardus ana menüsünden ***Programlar→İnternet→Apollon*** yolunu izleyebilirsiniz. Açılan pencerede Apollon'un hangi paylaşım kanallarına bağlandığı görülebilir. Ayrıca her servisin kaç kullanıcı içerdiği ya da kaç kullanıcının bağlandığı bilgisi de elde edilebilir.



Şekil 2.18: Apollon bilgi penceresi

Arama yapmak için pencerenin sol tarafındaki menüden **Ara** seçeneğini seçiniz. Bu işlemden sonra Şekil 2.19'daki pencere gelecektir.



Şekil 2.19: Apollon arama penceresi

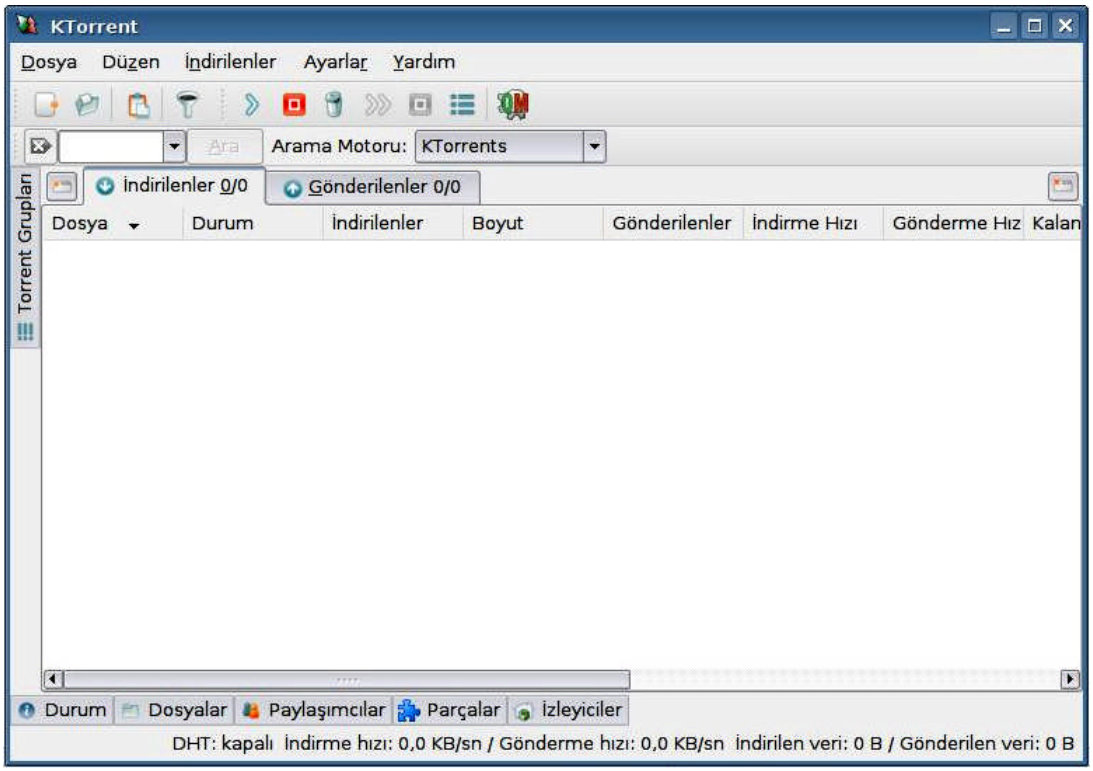
Ara kısmında dosya türünü belirterek (ses, video, metin, resim, yazılım vb.) ve anahtar sözcük girerek aramayı gerçekleştirebilirsiniz. Sonuçlar, pencere üzerinde listelenecektir. İstediğiniz dosyayı erişilebilir kullanıcı sayısını da dikkate alarak çift tıklamanız durumunda indirme başlatılacaktır. Ne kadar çok kullanıcı o dosyaya sahipse, indirme hızı da o oranda artacaktır.

İndirilen dosyalar öntanımlı olarak masaüstünde açılacak "**Shared Folder**" adlı dizine yapılır. İsterseniz bu dizini **Ayarlar** menüsünden değiştirebilirsiniz.

2.4.2. KTorrent

KTorrent ile *bittorrent* alt yapısını kullanarak yazılım indirmek mümkündür. Örneğin, Pardus kurulum dosyasını bittorrent üzerinden indirebilir ve bir Pardus CD'sine kolayca sahip olabilirsiniz.

KTorrent, Pardus ile birlikte gelen bir programdır. *Programlar→İnternet→KTorrent* yolundan çalıştırılabilir.



Şekil 2.20: KTorrent programı

Bu ekranda, sol üst köşedeki arama kutusuna anahtar sözcüğü yazarak, aramayı gerçekleştirebilirsiniz.

KTorrent'i kapatmanız durumunda, sistem çubuğunda sağ alt köşede çalışmasına devam edecektir. Tümden kapatmak için, KTorrent simgesine sağ tıklayarak **Çık** seçeneği seçilmelidir.

KTorrent açık olduğunda başkaları sizinle dosya paylaşımında bulunurken siz de başkalarıyla dosya paylaşımında bulunursunuz. Bu tür programlar, bilgisayarınızdan bir dizini İnternet üzerinde paylaşıma açar. Böylece, başkasından indirmiş olduğunuz yazılımın bir kaynağı da siz olursunuz. Dosya indirirken aynı zamanda sizde bulunan kopyalar başkaları tarafından alınabilir. Bu nedenle hattınız yavaşlarsa KTorrent'i kontrol ediniz.

UYGULAMA FAALİYETİ

Bu uygulama faaliyetinde, açık kaynak işletim sistemi dosya ve dizin paylaşım işlemlerini gerçekleştirmeyi öğreneceksiniz.

Bu uygulama faaliyetini gerçekleştirmek için aynı ağ üzerinde birbirini gören Pardus ve Windows işletim sistemi kurulu iki bilgisayar gereklidir.

İşlem Basamakları	Öneriler
Kullanıcı ev dizini içerisine yeni bir dizin oluşturunuz.	➢ Bu dizine **Paylasim** ismini veriniz.
Bu dizinin erişim haklarını ağda paylaşıma uygun şekilde düzenleyiniz.	➢ Dizini sağ tıklayarak Özellikler seçeneğini seçin ve diğer kullanıcılar için hakları düzenleyiniz.
Samba protokolünü etkinleştiriniz.	➢ Servis yöneticisinden etkinleştirebilirsiniz.
Dizini ağda paylaşıma açınız.	➢ Dizine sağ tıklayarak Paylaşım'ı seçiniz.
Ağa bağlı olan Pardus yüklü başka bir bilgisayarda Samba protokolünü etkinleştiriniz.	➢ Servis yöneticisinden etkinleştirebilirsiniz.
Samba protokolü üzerinden paylaşıma açtığınız dizine erişim sağlayınız.	➢ Konqueror'da **smb:/** yazarak Samba Paylarına erişebilirsiniz.
Ağa bağlı Windows yüklü bir makineden Pardus kurulu makineye erişim sağlayınız.	➢ Windows Ağ Bağlantılarım'dan diğer Windows makinelere erişim sağlandığı şekilde Pardus'a erişebilirsiniz.

ÖLÇME VE DEĞERLENDİRME

Öğrenme faaliyetinde kazandığınız bilgileri ölçebileceğiniz kısma geldiniz. Bu bölümde yer alan ölçme sorularını dikkatlice okuyarak cevaplandırınız.

Aşağıda doğru-yanlış tipinde sorular bulunmaktadır. Cümleleri okuyarak sorudaki ifade doğru ise "Doğru", yanlış ise "Yanlış" seçeneğini işaretleyiniz.

		Doğru	**Yanlış**
1	Pardus kurulu bir makineden Windows kurulu makinedeki dosyalara erişim sağlamak için, Pardus makinede Samba programı kurulu olmalıdır.		
2	Samba programı ayarları, yalnızca sistem yöneticisi (*root*) tarafından yapılabilir.		
3	Pardus sistemlerde yazıcı paylaşımı **yalnızca** ağa bağlı Pardus kurulu makineler arasında gerçekleştirilebilir.		
4	Apollon, bir bittorrent uygulamasıdır.		
5	KTorrent, Pardus ile birlikte gelen bir yazılımdır.		

Aşağıdaki soruların doğru cevabını verilen seçeneklerden bularak doğru seçeneği belirleyiniz.

6. Samba servisinin başlatılması için gerekli program aşağıdakilerden hangisidir?

A) Paket Yöneticisi
B) Sistem Yöneticisi
C) Servis Yöneticisi
D) Samba Yöneticisi

7. Samba ayarları nereden yapılır?

A) TASMA->İnternet ve Yerel Ağ->Samba
B) TASMA->Çevre Birimleri->Samba
C) TASMA->Sistem Yönetimi->Samba
D) TASMA->Bölge ve Dil Seçenekleri->Samba

8. Paylaşılan dosyalara erişim için Konqueror programında adres satırına ne yazılmalıdır?

A) remote:/ B) smb:/

C) konsole D) samba:/

9. Pardus'ta yazıcı ayarlarına nereden erişilir?

A) TASMA->Sistem Aygıtları->Yazıcılar
B) TASMA->Sistem Yöneticisi->Yazıcılar
C) TASMA->Internet ve Yerel Ağ->Yazıcılar
D) TASMA->Çevre Birimleri->Yazıcılar

10. Aşağıdaki programlardan hangisi İnternet üzerinden dosya paylaşım programıdır?

A) Apollon
B) Kate
C) Konsole
D) Firefox

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Test içinde cevaplandıramadığınız, yanlış cevaplandırdığınız veya kendinizi bilgi bakımından eksik hissettiğiniz sorular için bilgi sayfalarına tekrar dönüp öğrenme faaliyetini gözden geçirmeniz önerilir.

MODÜL DEĞERLENDİRME

PERFORMANS DEĞERLENDİRME

Bu kısımda modül sonunda kazandığınız yeterliliğin kontrolü yapılacaktır. Verilen problemlerin çözümünü öğretmeninizin kontrolünde gerçekleştiriniz. Öğretmeniniz sizi işlem basamaklarına göre Çok İyi, İyi, Orta, Vasat ve Başarısız şeklinde değerlendirecek ve yeterlilik performansınızı ölçecektir.

Problem 1:

Aşağıdaki işlemleri yapan ve ekran çıktısı aşağıdaki gibi olan bir kabuk programı yazınız.

```
1- Ev dizinini ayrıntılı olarak listele ve dosyaya yaz.
2- Dosyayı ekrana yazdır.
3- Tarih ve saati görüntüle.
Seçiminiz (1-3): _
```

Kullanıcı;

- 1 seçeneğini seçtiğinde, kullanıcı ev dizinindeki (~) dosya ve dizinleri ayrıntılarıyla listelenerek "liste.txt" dosyasına yazılacaktır.
- 2 seçeneğini seçtiğinde, liste.txt dosyasının içeriği ekranda görüntülenecektir.
- 3 seçeneğini seçtiğinde ise sistem tarih ve saati görüntülenecektir.

Öneri: *Seçenekleri oluşturmak için* ***case*** *yapısını kullanabilirsiniz.*

Problem 2:

- Bilgisayarınızda ortak bir dizin açarak, başka bilgisayarlardan bu dizine paylaşımı gerçekleştiriniz.
- Windows kurulu bir bilgisayardaki yazıcıyı, Pardus kurulu bilgisayarlara ağ üzerinden paylaştırınız.

GÖZLENECEK DAVRANIŞLAR	DEĞER ÖLÇEĞİ						
	Çarpan X	Çok İyi 4	İyi 3	Orta 2	Vasat 1	Başarısız 0	**TOPLAM**
Listeleme ve dosyaya yönlendirme işlemleri	3						
Dosya içeriği görüntüleme	2						
Sistem tarih ve saatini görüntüleme	2						
Kabuk programı hazırlama	10						
Dosya ve dizinlere paylaşım verme	3						
Dosya ve dizinlere ağdan erişim	2						
Yazıcıyı ağ üzerinden tanıtma	3						
TOPLAM	**25**						

DEĞERLENDİRME

Modül değerlendirmesinde verilen işlemi gerçekleştirebildiyseniz bu modülü başarı ile tamamladınız demektir. Eğer, anlayamadığınız bir konu ya da bilgi eksikliğinden sonuca ulaşamadığınız bir nokta var ise bilgi sayfalarını tekrar okuyunuz. Çözemediğiniz ve açıklık getiremediğiniz noktaları arkadaşlarınızla tartışınız. Yine de çözüm bulamazsanız alan öğretmeninize danışınız.

Ayrıca modül faaliyetleri ve araştırma çalışmaları sonunda kazandığınız bilgi ve becerilerin ölçülmesi için öğretmeniniz size değişik ölçme araçları uygulayacaktır. Ölçme sonuçlarına göre sizin modül ile ilgili durumunuz öğretmeniniz tarafından değerlendirilecektir.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1	**ls –al**
2	**cd ..**
3	**ps –aux** veya **ps -ef**
4	**rm –R <dizin>**
5	**^a[^0-9]*$**
6	**C**
7	**B**
8	**D**
9	**C**
10	**A**

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	**Doğru**
2	**Doğru**
3	**Yanlış**
4	**Yanlış**
5	**Doğru**
6	**C**
7	**A**
8	**B**
9	**D**
10	**A**

ÖNERİLEN KAYNAKLAR

- http://www.linfo.org
- http://www.belgeler.org
- http://www.pardus.org.tr
- http://www.uludag.org.tr/belgeler/pardus-kilavuz
- http://tr.pardus-wiki.org
- http://tr.wikipedia.org
- http://www.pardus-linux.org
- http://sanat.pardus.org.tr
- http://gezegen.pardus.org.tr
- http://liste.pardus.org.tr
- http://hata.pardus.org.tr
- http://www.lkd.org.tr
- http://www.kde.org.tr
- http://www.openoffice.org.tr
- ÇETİN, Görkem. Pardus, Seçkin Yayınları, Ankara, 2007.

KAYNAKÇA

- http://www.linfo.org
- http://www.pardus.org.tr
- http://tr.pardus-wiki.org
- http://www.belgeler.org
- http://sct.emu.edu.tr
- ftp://ftp.uybhm.itu.edu.tr
- http://www.uludag.org.tr
- http://tr.wikipedia.org
- http://www.ustuntas.net/belgeler/linuxebaslangic.pdf
- man Konsol Yardım Dosyaları
- AYDIN, Selçuk Han. Linux İşletim Sistemi, ODTÜ Bilgiişlem Daire Başkanlığı, Ankara, Eylül 2002. (http://www.bidb.odtu.edu.tr/index.php?go=usg&sub=cclib_linux)
- ÇETİN, Görkem. Pardus, Seçkin Yay., Ankara, 2007.