# Bölüm 10

# Unix

**Bu Bölümdekiler**

- Unix felsefesi
- Süreçler
- Kullanıcılar
- Araçlar

Unix bir güzelliktir. Unix, en az ile en fazla yapmanın en başarılı örneği, bir yalın tasarım abidesidir. Unix bize bir yazılım mimarinin nasıl olması gerektiğinin en çıplak şekliyle anlatır, tarif eder. Piyasada bu kadar zaman kalabilmiş, ve kritik uygulamalarda servis tarafın pazar payını tamamen eline geçirmiş olmasında bu doğru tasarımın büyük etkisi vardır. Aslında, kitabımızın diğer bölümlerinde önerdiğimiz mimariler ve teknolojiler, ruh olarak Unix felsefesine birebir uyumludurlar. Çünkü biz bu felsefenin takipçileri, ve bu felsefeyi projelerimizde kullanabilmiş sektörün elemanlarıyız.

Peki bu felsefe nedir?

Unix, tasarım bakımından özel hallerden kaçınmış bir işletim sistemidir[14, sf. 77]. Herkes için herşey olmaya çalışmaz, ama herşeyin yapılabilmesine izin verecek en baz altyapıyı sağlar. Bunun yan etkilerinde bir tanesi, altyapının uygulamalardan net bir şekilde ayrılması, ve böylece uygulamanın bir başkası ile rahat bir şekilde değiştirilebilmesidir.

Meselâ komut satırı programı, DOS'ta olduğu gibi işletim sisteminin içine sokuşturulmamıştır. Bu sadece bir görevten ibarettir, ve Unix'te ayrı programı ayrı süreç altında (aynen diğer uygulamalar gibi) çalışır. Unix'te "birşeyler yapan herşey" bir süreç, bir işlemdir[14, sf. 77]. Bu ayrılık sayesinde Unix dünyasında birçok komut satırı alternatifi geliştirilmiştir (`bash`, `sh`, `tcsh`, `ksh`) ama DOS dünyasında hâla sadece tek bir komut satırı mevcuttur.

Unix işletim sisteminin yalın yapısı, insanları en başta en çok şaşırtan, ve sonra merakını uyandıran faktörlerden biridir. Unix'de neredeyse herşey altı temel işlemle yapılır, ki bunlara "sistem çağrıları" adı verilmiştir; Bu altı temel sistem çağrısıyla neredeyse herşeyi geliştirebilirsiniz. Bu altı temel çağrıyı kavrarsanız, Unix'i kavramış olursunuz[14, sf. 88].

Yâni Unix'in güzelliklerinden birisi, karmaşık şeyler oluşturmak için karmaşık ara birimlere gereksinmediğinizi anlamış olmasıdır. Basit şeylerin etkileşimiyle istediğiniz kadar karmaşık yapılar oluşturabilirsiniz. Yapmanız gereken, karmaşık problem çözme yapıları yaratmak için basit işlemler arasında iletişim kanalları (pipe) oluşturmaktır[14, sf. 78].

Temel bilimlerde de bir fiziksel olayı açıklayacak matematiksel kuramlar arasında seçim yapılırken aranan özelliklerden birisi basitliktir. Bilim dünyasına yön veren önemli bir deyiş olan Occam'ın Usturası der ki: "Mevcut alternatifler arasında en basit açıklamayı seçiniz". Temel bilimler, tarih boyunca ve günümüzde bu desturu takip etmektedirler. Kurumsal programcılıkta da aynen temel bilimlerde olduğu gibi, basitlik *esastır* (Kural #1). Bilgisayar dünyasında, karışık yapıyı yaratmak programcının işidir (uygulama yazmak), kıyasla temel bilimler karışıklıktan basit açıklamaya doğru gitmeye çalışır; Fakat felsefe her iki taraf için aynıdır. Sadece gidilen yön değişiktir.

## 10.1 Unix Araçları

Bahsedilen türden basit yapı, Unix için birçok aracın yazılmasını sağlamıştır. Unix felsefesine uyumlu olan Unix araçları, aynen işletim sistemin kendisinin olduğu gibi, sadece *bir* işi, *en iyi* şekilde yapmak için yazılırlar.

Kitabımızın ana amacı kurumsal yazılım sektörde çalışan programcılar ve teknik liderlere yardım etmek olduğu için, her Unix aracının tüm özelliklerini teker teker anlatmayacağız. Bizim faydalı olduğunu düşündüğümüz bilgi, *ihtiyaçlar* ışığında hangi Unix aracı ve araçlar dizisinin (çoğu zaman iletişim kanalları ile birkaçını birbirine bağlayarak) kullanılması gerektiğini anlatmaktır.

### 10.1.1 Komut Birleştirme

Unix araçları arasında iletişim kanalı oluşturmak için "`|`" işareti kullanılır. Meselâ eğer `command1` ve `command2` adında iki Unix programımız olsa ve `command1`'in çıktısını `command2`'ye göndermek istesek, o zaman

```
command1 | command2
```

komut sırasını kullanırdık. Burada en son işletilecek komut olan `command2`'nin çıktısı (büyük ihtimalle) ekrana basılacaktır.

Ekrana basılacak bir çıktıyı bir dosyaya yönlendirmek istersek, "`>`" işareti ya da "`>>`" işaretinden sonra bir dosya ismi kullanılır. "`>`" işareti, yeni bir dosyaya yazmak, "`>>`" ise mevcut bir dosyaya eklemek için kullanılır.

```
command1 | command2 >> /tmp/out.txt
```

Bu işletim sırasına göre, önce `command1` çağırılacak, onun çıktısı `command2`'ye girdi olarak verilecek ve `command2`'nin çıktısı `/tmp/out.txt` adlı dosyaya yazılacaktır.

### 10.1.2 Süreçler

Unix'de işlettiğiniz her program bir süreç (process) hâline gelir.

#### Listelemek

O ana kadar sizin başlatmış olduğunuz süreçleri görmek istiyorsanız, `ps` komutunu kullanabilirsiniz. Şuna benzer bir çıktı gelecektir

```
 PID   PPID   PGID    TTY  UID   STIME COMMAND
 452      1    452    con 1004 11:55:54 /usr/bin/bash
1656    452   1656    con 1004 11:55:58 /usr/bin/ps
```

Bu listedeki `PID` kolonu sürecin kimlik numarasıdır. Eğer sistemde, sizin dahil, tüm kullanıcıların başlatmış olduğu süreçleri görmek istiyorsanız,

```
ps -eaf
```

komutunu kullanabilirsiniz. Bu sefer daha büyük bir liste gelecektir (çünkü herkesin süreçlerini görmek istediniz).

#### Öldürmek

Bir süreci öldürmek için `PID` no'sunu öğrenip o numarayı kullanarak şu komutu kullanabilirsiniz.

```
kill -9 <PID>
```

Eğer bir programı `ps -eaf` listesinde çıkan bir "isme" göre öldürmek istiyorsanız, çoğu Unix'de bunun için bir `pkill` komutu vardır. Ama dikkat edin! Mesela `pkill java` gibi bir komut, sistemde işleyen tüm Java programlarını öldürür; Eğer aynı makinada birden fazla kişi Java uygulamasını test ediyor ve siz de `root` iseniz, insanların çalışmasını etkileyebilirsiniz. Bu yüzden bu komutu dikkatli kullanın.

### 10.1.3 Dosyalar

#### Ekrana Basmak

`cat` ile bir dosyayı tamamen ekrana basabiliriz. Eğer dosyayı kısım kısım görmek istiyorsak, `less <dosya>` komutu işe yarar. `less` işlerken bir sonraki sayfaya geçmek için `SPACE`, geri gitmek için `b`, en tepeye gitmek için `g`, ve en alta gitmek için `G` tuşları kullanılabilir. `q` ile `less` programından çıkmak mümkündür. `less` komutu, ondan önce gelmiş olan `more` komutundan daha kuvvetlidir, tavsiyemiz `less` kullanmanızdır. Buradaki isimlendirme esprisi, raslantısal bir şekilde (ya da bilerek) Unix felsefesinin özünü yine ortaya koymuştur: (daha az anlamına gelen) `less` komutu (daha fazla anlamına gelen) `more`'dan daha kuvvetlidir!

#### Dosya İçinde Bulmak

Herhangi bir dizin altındaki bir veya daha fazla dosyalar *içinde* bir kelimeyi aramak istiyorsanız, `grep` komutunu kullanabilirsiniz.

```
grep 'aranan kelime' *.java
```

gibi. Bu komut sonucundan bulunan dosyalar listenecektir.

#### Dosya Bulmak

Bir isim düzenine uyan tüm dosyaları bulmak için, `find` komutu kullanılır.

```
find /usr/local -name '*.java'
```

Bu komut `/usr/local` seviyesinden başlayarak sonu `*.java` ile biten tüm dosyaları bulup geri getirecektir. `find` komutundan hemen sonra gelen dizin, aramanın nereden başlayacağına işaret eder.

Eğer bulunan dosyalar *üzerinde* bir komut işletmek istiyorsanız, `find` ile `xargs` komutunu birbiri ile iletişim (pipe) kurdurarak kullanabilirsiniz. Meselâ sonu `*.java` ile biten tüm dosyaları ekrana basmak için

```
find . -name '*.java' | xargs cat
```

### Dosya ve Dosya İçinde Bulmak

İşte Unix felsefesinin bir örneği: Öğrendiğimiz iki komutu birleştirerek, bir dizinden başlayarak hem o dizin hem de altındaki tüm dizinler altında belli bir düzene uyan dosyalar *içinde* bir kelimeyi arıyorsak,

```
find . -name '*.java' | xargs grep -l 'aranan kelime'
```

komutunu kullanabiliriz. `-l` seçeneği, bulunan dosya ismini ekrana basmak için kullanılır.

### Dosya ve Dizin Büyüklüklükleri

Bir dizinin ya da tek dosyanın ne kadar yer tuttuğunu anlamak için, `du` kullanılır. Tek başına kullanılınca, komutun kullanıldığı dizin altındaki *her dizinin* ne kadar büyük olduğu ekranda gösterilecektir. Eğer `du -s` kullanılırsa, bu alt dizin büyüklüklerinin toplamı alınacaktır, yâni tek sayı geri gelecektir.

Eğer bir dizinin altındaki hangi alt dizinin en fazla yer tuttuğunu merak ediyorsak (ki bu bazen çok işe yarar), o zaman, `du`'dan gelen sonuçları başka bir Unix komutuna iletmek gerekecektir. Bu diğer komur, sıraya dizme (sort) komutudur.

```
du | sort -n
```

Bu komut dizinleri büyüklük sırasına göre küçükten büyüğe doğru dizer. Peki `sort` komutu, `du`'dan gelen sonuç üzerinde ilk kolona (büyüklüğe) bakacağını nereden bildi? Çünkü `du`'dan gelen sonuçlarda büyüklük, ilk kolonda idi, ve `sort`, satırları ilk karakterlerinden başlayarak dizmek için yazılmıştır.

### Satır Sayısı

Bir dosyanın kaç satır ve kaç kelime içerdiğini anlamak için `wc` kullanılır. Tek başına `wc` bir dosyadaki satır, kelime sayısını beraber gösterir, ayrı ayrı bilgi almak için `wc -l`, sadece satır sayısını, `wc -w` kelime sayısı için kullanılabilir.

### İçerik Değiştirmek

Bunun için Perl kullanacağız. Komut satırında Perl, hem yerinde değişiklik ve hem de düzenli ifade kullanabilmektedir.

```
perl -pi -e 's/filan/falan/sg'
```

komutu, `filan` ile `falan` kelimesini değiştirir. İş bittikten sonra bir yedek dosyasi (`.bak`) bulacaksınız.

## 10.2 Kullanıcılar

Unix'de her kullanıcı `/etc/passwd` dosyasında tutulur. Bu dosyaya erişmeye sadece `root` kullanıcının hakkı vardır. Projenizde genelde programcılara `root` erişimi verilmez, ama teknik lidere bu hak tanınır. Bu, Unix'i daha yeni öğrenmekte olan programcıların yanlışlıkla sistemi hasar vermesine karşı yapılır.

En güçlü kullanıcı olan `root`, birçok admin odaklı işi yapabilir, bu yüzden hakları tüm diğer kullanıcılardan daha fazladır. Mesela `root`, herkesin sürecini `kill` ile öldürebilir. Diğer kullanıcılar böyle bir şey yapmaya kalkışsa, sistem onlara izin vermeyecektir.

Bir kullanıcın giriş yaptığında hangi komut satır progamını (shell) kullanacağı (birçok seçenek mevcuttur), `/etc/passwd` içinde tanımlıdır. Bu shell, kullanıcı yaratılırken `useradd` komutuna parametre olarak verilir;

```
useradd -d /home/user123 -s /bin/bash user123
```

Parametre `-d` kullanıcı (`HOME`) dizini, `-s` ise shell tipi için kullanılır.

Bir kullanıcı sisteme girdiğinde, ya da bir xterm, yeni bir shell başlattığında, kullandığı shell türüne göre işletilen ayar dosyası değiştiktir. Shell `sh`, ve `bash` ise, önce `/etc/profile` script'i, sonra kullanıcı `HOME` dizini altındaki `.profile` işletilir. Eğer admin isek ve tüm kullanıcıların etkileneceği türden bir değişiklik yapmak istersek, bunu `/etc/profile` içinde yaparız. Her kullanıcı kendi isteğine göre `.profile`'ı değiştirebilir.

Her kullanıcı için bir `HOME` değişkeni mevcut olacaktır. Bu değişkeni ekranda göstermek için `echo \$HOME` komutunu kullanabilirsiniz.

Erişim haklarını daha geniş bir kategori üzerinden idare edebilmeye admin'ler için gereklidir. Bu ihtiyaca cevaben Unix, "kullanıcı grupları" kavramını destekler. Bir kullanıcı birden fazla gruba dahil olabilir. Bir kullanıcıyı `useradd` ile yaratırken hangi gruba dahil olmasını istediğinizi biliyorsak, `-g` seçeneği ile bu grubu tanımlayabiliriz. Sisteme bir grup eklemek için `groupadd` komutunu kullanmak gerekiyor.

Eğer `useradd` işlemi bittikten sonra kullanıcıyı bir gruba dahil etmek istersek, `/etc/passwd` dosyasını güncelleyerek bunu başarabiliriz. Bu dosyada kullanıcının tanımlandığı satırı bulup, oradaki grup listesine yeni grup no'sunun eklenmesi gerekiyor. Grup ismine bakadar grup numarasını bulmak için `/etc/group` dosyasına bakabiliriz.

Bir kullanıcı hangi gruba dahil olduğunu görmek isterse, komut satırından `groups` komutunu vermesi yeterlidir. Süper kullanıcı `root` başkalarının gruplarına bakabilir, bunu için `groups <username>` komutunu kullanır.

### 10.2.1 Dosya Hakları ve Kullanıcılar

`ls -al` ile bir dizindeki tüm dosyaların kullanım haklarını görebilirsiniz.

```
drwxr-xr-x  9 burak None 0 Jun 14 15:57 Example
-rwxr-xr-x  9 burak None 0 Jun 14 15:57 Ex.txt
```

Bu listede iki birim görüyoruz; `Example` ve `Ex.txt`. Her dosya ya da dizin için onun olduğu satırın başına bakarsak, `drwxr-xr-x` gibi bir tanım görürüz. `Example` dizini örnek alalım. `drwxr-xr-x` ne demektir?

Bu tanım kelimesini, öncelikle zihnimizde dörde bölmemiz gerekir. Yâni biraz önceki örnek, şuna dönüşür: `d`, `rwx`, `r-x`, ve `r-x`. Bu bölümlerden tek hâneli birinci bölüm birimin dosya mı, dizin mi olduğunu belirtir. `d` dizin, `-` dosya demektir.

Ondan sonra gelen üç büyük grup, sırasıyla kullanıcı (user), grup (group) ve ötekiler (others) içindir. Kullanıcı, kullanıcın kendisi için geçerli olan haklar, grup, kullanıcının içinde olduğu gruptaki herkes için olan haklar, ötekiler ise bunun haricinde kalan herkes için geçerli olan haklardır. Böylece kendinize verdiğini bir hakkı, grubunuzdaki bir kişiye vermemeyi (meselâ) seçebilirsiniz.

Her kelime grubu içindeki haklar, şunlar olabilir. O dosyayı okumak (read), değiştirebilmek (write), ve işletebilmek (execute). Okumak için `r`, değiştirebilmek için `w` ve işletebilmek için `x` karakteri kullanılır. Eğer bu haklar var ise onun karakteri, yok ise `-` karakteri kullanılacaktır.

Hakların karakterleri bir grup içinde hep aynı yerlerde çıkarlar; Grubu içinde `r` hep birinci sırada, `w` ikinci sırada, ve `x` ise üçüncü sırada olacaktır.

#### Dosya Kullanım Hakkı Vermek ve Almak

Hak vermek için `chmod` komutu kullanılır. Kullanım `chmod <haklar> dosya` şeklindedir. Bir dosyanın tüm haklarını aynı anda set edebilen sayı yöntemi yerine (meselâ `chmod 777 file.txt`) biz, `+` ve `-` ile hak ekleme çıkarma yöntemini tercih ediyoruz. Bunun için, meselâ kullanıcıya okuma hakkı vermek için `chmod u+r <dosya>` komutu kullanılır. Hak eksiltmek için `+` yerine `-` kullanmak gerekir. Aynı şekilde gruba okuma hakkı vermek için `chmod g+r <dosya>` kullanılır. "Öteki" kullanıcılar için `o`, ve her kullanıcı *herkes* için `a` kullanılmalıdır. `a` ile tüm kullanıcılara (`u`, `g`, `o`) belli bir hakkı aynı anda verme yeteneğine kavuşuyoruz.

## 10.3 Scripting

Bir arada işlemesini istediğini komutları bir dosyaya koyup, bir script olarak işletebilirsiniz. Script'leriniz herhangi bir shell için yazabilirsiniz, ama bizim tavsiye ettiğimiz shell, her Unix sisteminde olması garanti olan `sh` shell'idir.

Bir script'i işletmek, kurumsal programcılar tarafından Unix'in en az anlaşılan ve en son "tamamiyle" öğrenilen tekniklerden biridir. Bir script içinde set

edilen değişkenler, *çağıran* shell'i nasıl etkileyecektir? Bir script'i birkaç şekilde çağırma şekillerinden hangisi uygundur?

Elimizde `script.sh` adında bir script olduğunu farz edelim. Bu script içinde şunlar olsun;

```
VAR1=/tmp/vs/vs
command1
command2
```

Bu script'i üç şekilde işletebilirsiniz.

1. `sh script.sh`: Script'in işlemesi tamamlandıktan sonra, çağıran shell, içeride tanımlanan ve set edilen `VAR1`'ı göremez.

2. `script.sh`: Avantajı, işletmek için shell üzerinde sadece tek kelime kullanmasıdır. Bunun için `script.sh` dosyasının en üst satırına `#!/bin/sh!` ibaresini eklemeliyiz. Bir de, işletmeden önce (sadece bir kez) `chmod u¬ +x shell.sh` ile bu dosyayı "sadece kendi kullanıcımız için" işletilebilir hâle getirmeliyiz. Bundan sonra komut satırından uygulanacak tek başına `shell.sh` komutu, çalışacaktır. Fakat `VAR1`, aynen biraz önce olduğu gibi, çağıran shell tarafından gözükmeyecektir.

3. `. shell.sh`: Bu kullanım, Unix'de kaynaklama (sourcing) yapar, yâni script *sanki içindeki her satır, komut satırı üzerinde elle yazılıyormuş gibi* işletilir. Ayrıca daha önceki işletim stillerinde, `shell.sh` çağırımı yeni bir süreç altında işliyordu, ve orada yaratılan her değişken yapılan her iş o süreç bitince onunla beraber ölüyordu. Kaynaklama yönteminde durum değişiktir. Bu yöntemde, çağırım geri gelince `VAR1`'in değeri çağıran shell tarafından görülüyor olacaktır.

Ayrıca eğer 1. ve 2. yöntemlerde de `shell.sh` işledikten sonra içeride tanımlanan ve set edilen değişkenlerin dışarıdan (çağıran shell içinden) görülmesini istiyorsak, o zaman bu yöntemlerde script içinde değişkene bir değer set ettikten sonra onu dışarıya "ithal" etmeliyiz; Bourne shell (`sh`) içinde bu `export VAR1` komutu kullanılarak yapılır.

## 10.4 Makina Başlayınca Program İşletmek

Bazı programların bilgisayar açıldığı zaman "otomatik" olarak başlamasını istiyor musunuz? Mesela, bir Linux makinasını tamamen Oracle icin ayırdık, ve Oracle, bilgisayar açılır açılmaz başlamalı. Sistem idarecisi olarak her seferinde oracle kullanıcısına girip, durdurma/başlatma yapmak istemiyoruz..

Unix'te bu işler, diğer pek çok şeyde olduğu gibi, metin bazlı ayar dosyaları üzerinden yapılıyor. Bir Unix sisteminde bilinen, ve önemli dizin bölgeleri vardır. Mesela `/etc/` dizini bunlardan biridir. Çogu Unix versiyonu, `/etc/` altına

onemli ayar dosyalarını koyar. Yâni, bir script ile `/etc/` altındaki dosyaları değiştirirseniz, sistemin işleyişi değişecektir. Dikkatli olmamız isabetli olacaktır.

Durdurup başlatma ayar dosyaları `/etc/rc.d` dizini altındadır. Buraya girip `ls -al` işletirseniz, aşağıdaki tabloyu görebilirsiniz.

```
drwxr-xr-x  10 root    root        4096 Sep 15 01:09 .
drwxr-xr-x  43 root    root        4096 Sep 15 01:23 ..
drwxr-xr-x   2 root    root        4096 Sep 6 15:51 init.d
-rwxr-xr-x   1 root    root        3219 Jul 10 2001 rc
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc0.d
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc1.d
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc2.d
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc3.d
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc4.d
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc5.d
drwxr-xr-x   2 root    root        4096 Apr 4 13:25 rc6.d
-rwxr-xr-x   1 root    root        3200 Sep 15 01:08 rc.local
..
..
```

`rc` kelimesinden sonra gelen sayı, "başlama seviyesini" belirtir. Bir Linux makinasını değişik başlama seviyesinde başlatmamız mümkündür. Meselâ çok önemli bir sistem bakımı gerekiyorsa ve bu bakım yapılırken hiçbir kullanıcının sisteme bağlanamaması lâzım ise, başlama seviyesi 0 yada 1, bu iş için kullanılabilir. Bu seviyelerde kullanıcı giriş programları sağlanmamıştır, ve böylece sadece `root` sisteme terminalden girerek istediği bakımı yapabilir.

Sistemi 4. başlama seviyesinde başlatmak demek, `rc4.d` altına girip, oradaki başlatma script'lerini işletmektir. Şimdi `rc4.d` altında ne var görelim.

```
lrwxrwxrwx   1 root    root          14 Apr 2 13:36 K74ntpd
lrwxrwxrwx   1 root    root          16 Apr 2 13:38 K74ypserv
lrwxrwxrwx   1 root    root          16 Apr 2 13:38 K74ypxfrd
lrwxrwxrwx   1 root    root          15 Apr 2 13:34 S05kudzu
lrwxrwxrwx   1 root    root          18 Apr 2 13:34 S08ipchains
lrwxrwxrwx   1 root    root          18 Apr 2 13:34 S08iptables
lrwxrwxrwx   1 root    root          17 Apr 2 13:34 S10network
lrwxrwxrwx   1 root    root          16 Apr 2 13:33 S12syslog
lrwxrwxrwx   1 root    root          17 Apr 2 13:36 S13portmap
lrwxrwxrwx   1 root    root          17 Apr 2 13:37 S14nfslock
lrwxrwxrwx   1 root    root          18 Apr 2 13:33 S17keytable
lrwxrwxrwx   1 root    root          16 Apr 2 13:34 S20random
lrwxrwxrwx   1 root    root          15 Apr 2 13:34 S25netfs
lrwxrwxrwx   1 root    root          14 Apr 2 13:34 S26apmd
```

`ipchains`, `syslog` gibi programlar tanıdık gelebilir. Bu programların başlatıldığı noktayı böylece görmüş oluyoruz.

Bir kavram daha kaldı: Bazı script'lerin "`K`" ile, ötekilerinin "`S`" ile başladığını görüyoruz. Bunun sebebi nedir? `S` "`Start`" icin `K` "`Kill`" için kullanılır, yâni

başlat ve durdur komutlarıdır. Eğer Unix sistemi, `rc4.d` altındaki servisleri başlatmak istiyorsa, önce, `rc4.d` altinda "`ls S*`" benzeri bir komut işletecektir. Bu komut sadece "`S`" ile başlayan script'leri toplar. Sonra Unix sistemi, bu script'leri teker teker "`start`" kelimesini ekleyerek cağırır. Aynı şekilde sistem kapanırken, Unix `ls K*` benzeri komut işletip, durdurmak için gerekli script'leri toplar, ve onları "`stop`" kelimesini ekleyerek çağırır.

Peki niye çağırım yaparken "`start`" ve "`stop`" eklemek gerekiyor? Bunun sebebini script içeriğine baktığımızda göreceğiz.

```
..
..
start() {
       echo -n >"Kaydediciyi baslatiyoruz.. "
..
..
stop() {
       echo -n >"Kaydediciyi durduruyoruz.. "
..
..
case ">1" in
  start) ;; eger Unix start kelimesi gondermis ise
       start
       ;;
  stop)
       stop eger Unix stop kelimesi gondermis ise
       ;;
  status)
       rhstatus
       ;;
  restart|reload)
       restart
       ;;
..
..
```

## 10.5 Takvime Bağlı Program İşletmek

Bir zamana, takvime bağlı program işletmek için, `cron` programı kullanılır. `cron` programı, `crontab` adında bir ayar dosyası kullanır, ve bu dosya `/etc/¬ crontab` altında bulunur. Bu ayar dosyasını ya direk (`root` olarak) ya da komut satırından `crontab -e` ile edit etmeye başlayabilirsiniz. Eğer herhangi bir dosyanın `crontab` *dosyası olarak* kullanılmasını istiyorsanız, `crontab <file>` komut ile bunu yapabilirsiniz. Bu durum genellikle `crontab` dosyasını CVS gibi bir kaynak kod idare sisteminde tutmak isteyenler için gerekli olmaktadır.

Bazı `crontab` seçenekleri (ve alâkalı bir komut) şunlardır:

- `export EDITOR=vi crontab`'in hangi editör ile açılacağını kontrol eder. `vi` yerine (eğer varsa) `emacs` kullanabilirsiniz.
- `crontab -e`: Ayar dosyasını günceller
- `crontab -l`: Ayarları ekranda gösterir
- `crontab -r`: `crontab` dosyasını siler
- `crontab -v`: Dosyayı en son güncellediğimiz tarihini gösterir

`crontab` dosyasının her satırı, değişik bir programı ayarlamak için kullanılır. Bu her satırda, her kolon, zaman ayarının belli bir bölümü için kullanılır. Kolonlar sırasıyla şunlardır: Dakika, saat, ayın günü, ay, haftanın günü ve yıl. Her kolon için yıldız "`*`" işareti, o ayarın dikkate alınmadığı anlamına gelir, meselâ gün için "`*`" var ise, o program her gün işleyecektir. Tablo 10.1 üzerinde izin verilen kolon değerlerini görüyoruz.

Tablo 10.1: Crontab Kolon Değerleri

| **Alan** | **İzin Verilen Değerler** |
|---|---|
| dakika | 0-59 |
| saat | 0-23 |
| ayın günü | 0-31 |
| ay | 0-12 |
| haftanın günü | 0-7 |

Örnek bir `crontab` dosyası ise, altta görülmektedir.

```
SHELL=/bin/sh
MAILTO=burak
5 0 * * *      \$HOME/bin/daily.job >> \$HOME/tmp/out 2>&1
15 14 1 * *    \$HOME/bin/monthly
0 22 * * 1-5  mail -s "Saat 10pm" burak%Burak,%%Nabersin?%
```

Her bir program tanımını teker teker tarif etmek gerekirse:

1. İlk önce işletilen programların hangi shell'i kullanması gerektiğini tanımlıyoruz. Burada seçim `/bin/sh` olmuştur.
2. Bu satırda, `cron` işledikten sonra onun işlettiği dosyaların çıktısının kime mail edileceğini tanımlıyoruz. Bu kişi burada `burak` adındaki kullanıcıdır.
3. Bu satırda, ilk takvimli program tanımı yapılmıştır. Bu program, her gece yarısından beş dakika sonra her gün işletilecektir. `\$HOME` değişkeni, `crontab` dosyasının sahibidir.

4. Her ayın ilk gününde saat 2:15pm'de `\$HOME/bin/monthly` adlı program işletilecektir.

5. Her iş günü (Pazartesi ve Cuma arasındaki her gün, ve bu günler dahil olmak üzere) saat 10 pm'de bu program işletilir.

## 10.6 Network Durumu

`netstat`, sisteminizdeki network durumu gösterir. Genellikle kullanılma sebebi, bir port'un o anda kullanılıp kullanılmadığını anlamaktır. Sistemde kullanılan tüm port'ların listesini almak için `netstat -a` kullanılır.

## 10.7 Yardım Almak

`man` komutundan sonra bir program ismi belirsek, bu program hakkındaki sistemde olan tüm açıklama ekrandan verilecektir. Unix dünyasında `man` oldukça fazla kullanılır, çünkü yeni öğrendiğimiz (hâtta bazen eski bildiklerimizin bile) komutların seçeneklerini öğrenmek için, `man komut_ismi` kullanırız.

## 10.8 X-Windows Kullanımı

X-windows, Unix dünyasında (Java applet'lerden çok önce) görsel herhangi bir programın ekranını, penceresini, başka bir makinanın monitoründe göstermek için kullanılan tekniktir.

X-windows ile servis tarafında bir programı shell üzerinden işletip, penceresini kendi sisteminize alabilirsiniz. Bunu yapabilmek için öncelikle masaüstü (çağıran) sisteminizde bir X ortamı gerekecektir. Böyle bir ortam, eğer masaüstü Linux ya da diğer bir Unix ise kendiliğinden olacaktır (kurulum sırasında X-windows seçtiyseniz). Windows üzerinde ise, Cygwin üzerinden X-windows kullanabilirsiniz. Cygwin kurulumu için A.6 bölümüne, Cygwin üzerinde X ortamı kurulumu için A.13 bölümüne bakabiliriz.

Görsel programı işletmek için önce servis tarafına `ssh` ile bağlanılır. `ssh` üzerinden X bilgilerinin geriye alınabilmesi için, `-X` seçeneği kullanılmalıdır.

```
(desktop) \$ ssh host1 -l root -X
```

Servis sistemine girdikten sonra, servis makinasına ekranın *nerede* olduğunu söylememiz gerekiyor. Bunun için `DISPLAY` çevre değişkenine ekranın makina ismi (ya da IP numarası) sonuna `:0.0` eklenerek verilir.

```
(host1) \$ export DISPLAY=desktop:0.0
```

Son basamak, masaüstü tarafında servis tarafına görsel bilgileri göndermesi için izin vermektir:

```
(desktop) \$ xhost +
```

Bu komut tüm servis programlarına X bilgisi göndermesi için izin verecektir. Artık X programını başlatabiliriz. Meselâ `xclock`;

```
(host1) \$ xclock
```

Bu programın görüntüsü olan bir saat, masaüstü ekranımızda çıkacaktır.