APT NASIL

Gustavo Noronha Silva <kov@debian.org>
Çeviri: Murat Demirten <murat@debian.org>

1.8.3 - Aralık 2002

Özet

Bu doküman Debian paket yönetim uygulaması APT hakkında genel kullanım bilgileri vermeyi amaçlamaktadır. APT'nin amacı Debian kullanıcılarının hayatını kolaylaştırmak ve sistemin yönetiminin daha iyi, anlaşılır şekilde yapılabilmesini sağlamaktır. APT Debian dağıtımı olarak kullanıcıları daha çok desteklemek amacıyla, Debian projesi için geliştirilmiştir.

Telif Hakkı

Contents

1 Başlarken **1**

2 Temel Konfigürasyon **3**

2.1 /etc/apt/sources.list dosyası 3

2.2 APT'nin yerel olarak kullanımı 4

2.3 sources.list dosyası için en iyi yansıya karar verme: netselect, netselect-apt 5

2.4 sources.list dosyasına CD-ROM ekleme 6

3 Paketlerin yönetimi **9**

3.1 Paket listesini güncelleme 9

3.2 Paket kurma 9

3.3 Paket kaldırma 11

3.4 Paket güncelleme 12

3.5 Yeni bir sürüme güncelleme 13

3.6 Kullanılmayan paket dosyalarını temizleme: apt-get clean ve autoclean 15

3.7 APT ile dselect kullanımı 16

3.8 Karışık bir sistem nasıl kurulur? 18

3.9 Sadece belirli bir Debian versiyonuna sahip paketleri güncelleme 18

3.10 Belirli bir versiyona sahip paketlerin kurulu olarak kalmasını sağlama (ileri düzey) 18

4 Yardımcı araçlar **21**

4.1 Kendi derlediğim paketleri nasıl kuracağım: equivs 21

4.2 Kullanılmayan yerelleştirme dosyalarını kaldırma: localepurge 23

4.3 Güncellenebilir paketleri nasıl öğrenebilirim? 23

5 Paketler hakkında bilgi toplama **25**

5.1 Paket isimlerini keşfetme 25

5.2 Paket adlarını bulmak için dpkg kullanma 27

5.3 Programları anında kurma 28

5.4 Bir dosyanın hangi pakete ait olduğunu bulma 29

5.5 Paketlerdeki değişikliklerden haberdar olma 29

6 Kaynak paketlerle çalışma **31**

6.1 Kaynak paketleri indirme 31

6.2 Kaynak paketleri derlemek için gerekli paketler 32

7 Hatalarla başa çıkma **33**

7.1 Genel hatalar 33

7.2 Nereden yardım bulabilirim? 34

8 Hangi Linux dağıtımları APT destekliyor? **35**

9 Teşekkürler **37**

10 Bu dokümanın yeni versiyonları **39**

Chapter 1

Başlarken

Önce .tar.gz vardı. Kullanıcılar GNU/Linux sistemlerinde kullandıkları her programı derlemek zorundaydılar. Debian geliştirimi sırasında, kurulu paketlerin yönetimini sağlayacak bir sistemin zorunluluğu hemen görüldü. Bu amaçla geliştirilen sisteme `dpkg` adı verildi. Sonuçta GNU/Linux dünyası, RedHat'in kendi paket yönetim sistemini geliştirmesinden biraz önce, dpkg ile tanıştı.

Hemen ardından GNU/Linux sisteminin geliştiricilerinin kafasında yeni bir ikilem oluştu. Paketlerin hızlı, pratik ve verimli bir yöntemle kurulabilmesini, paket bağımlılıkları yönetimi ve paket güncellemeleri sırasında konfigürasyon dosyalarının güncellenmesini sağlayacak bir araca ihtiyaç vardı. Gene Debian projesi kapsamında ilk ürün ortaya çıktı: APT (Advanced Packaging Tool). Apt daha sonra Connectiva tarafından rpm paketler ile kullanılmak üzere port edildi ve bazı dağıtımlar tarafından da kullanılmaya başlandı.

Bu dokümanda Connectiva APT portu olan apt-rpm'den bahsedilmemekle birlikte, dokümana eklenti amacıyla bu konuda da bir şeyler gönderebilirsiniz.

Bu döküman bir sonraki Debian sürümü olacak `Sarge` temel alınarak yazılmıştır.

Chapter 2

Temel Konfigürasyon

2.1 /etc/apt/sources.list dosyası

APT paketlerin bulunduğu kaynaklara nasıl erişebileceğinin bilgisini `/etc/apt/sources.list` dosyasında saklar.

Dosya içindeki girdilerin biçimi aşağıdaki gibidir:

```
deb http://site.http.org/debian sürüm bölüm1 bölüm2 bölüm3
deb-src http://site.http.org/debian distribution section1 section2 section3
```

Elbette yukarıda verdiğimiz örnek kullanılabilir değildir. Her satırın ilk kelimesi mutlaka `deb` veya `deb-src` olmak zorundadır. Bu ifadeler arşivin tipini belirler: derlenmiş ve kullanıma hazır duruma getirilmiş, binary paketler (`deb`) veya programın asıl kaynak kodu + Debian paketi için yapılan eklentilerden oluşan halini içeren paketler (`deb-src`).

Öntanımlı Debian sources.list dosyasının biçimi genellikle aşağıdaki gibi olacaktır:

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-fre
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable non-US
```

En temel Debian kurulumu için ihtiyaç duyulan satırlar bunlardır. Birinci `deb` satırı resmi Debian arşivini gösterirken, ikincisi non-US arşivini ve üçüncüsü ise Debian güvenlik güncellemelerini içeren arşivi göstermektedir.

Sondaki iki satır yorum haline getirilmiş olup (satır başındaki '#' karakteri ile) apt-get tarafından yoksayılmaktadır. Buradaki `deb-src` satırları Debian kaynak paketlerini göstermektedirler. Eğer sıklıkla program kaynak kodlarını test veya yeniden derleme amaçlı olarak indiriyorsanız satırları aktif hale getirmelisiniz.

`/etc/apt/sources.list` dosyası çeşitli tiplerde satırlar içerebilir. APT uygulaması `http`, `ftp`, `file` (yerel dosyalar, örneğin bağlı durumdaki bir ISO9660 dosya sistemi) ve `ssh` arşiv tiplerini tanıyabilmektedir.

`/etc/apt/sources.list` dosyasında değişiklik yaptıktan sonra `apt-get update` komutunu çalıştırmayı unutmayınız. Bu komutla APT'nin dosyada belirtmiş olduğunuz arşivlerdeki güncel paket listesini edinmesini sağlamış olursunuz.

2.2 APT'nin yerel olarak kullanımı

Bazen APT tarafından kurulmasını istediğiniz onlarca paketiniz olabilir. Bunların hepsini birden sisteminizdeki kopyasından kurulumunu isterseniz aşağıdaki adımları izlemelisiniz.

Öncelikle bir dizin yaratıp içerisinde sizdeki .deb paketlerini atınız. Örnek olarak:

```
# mkdir /root/debs
```

Paketlerin control dosyalarında belirtilen tanımlamalarının üzerine kendi tanımlamalarınızı `override` dosyası kullanarak yapabilirsiniz. Bu dosya içerisinde paketle birlikte gelen bazı seçenekleri aşağıdaki gibi tanımlayabilirsiniz:

```
paket öncelik bölüm
```

paket, paketin ismini belirtir; öncelik değeri sırasıyla düşük, orta ve yüksek anlamında low, medium ve high olabilir; bölüm ise paketin ait olduğu bölümü gösterir. Dosya adının mutlaka böyle olması gerekmez, başka bir dosya kullanıp `dpkg-scanpackages`'a parametre olarak verebilirsiniz. Eğer bir `override` dosyası yazmak istemiyorsanız `dpkg-scanpackages` programını çağırırken `/dev/null`'u kullanabilirsiniz.

Halen /root dizini içerisindeyken aşağıdaki komutu çalıştırın:

```
# dpkg-scanpackages debs dosya | gzip > debs/Packages.gz
```

Yukarıdaki satırda *dosya* olarak girilen dosya, `override` dosyasıdır. Bu komut APT tarafından kullanılmak üzere, paketler hakkında çeşitli bilgiler içeren `Packages.gz` dosyasını üretir. Paketleri kullanmak için son olarak aşağıdaki satırı `sources.list` dosyanıza ekleyin:

```
deb file:/root debs/
```

Bu eklentiden sonra artık APT uygulamasını yeni arşivinizle birlikte kullanabilirsiniz. Ayrıca isterseniz kaynak paketler için de yerel bir arşiv oluşturabilirsiniz. Bunun için izlemeniz gereken yol yukarıdaki ile hemen hemen aynıdır. Dikkat etmeniz gereken nokta, `.orig.tar.gz`, `.dsc` ve `.diff.gz` dosyalarını da dizin içerisine kopyalamanız ve `Packages.gz` dosyası yerine `Sources.gz` dosyasını oluşturmanız gerektiğidir. Kullanacağınız program ise `dpkg-scansources` olacaktır. Örnek:

```
# dpkg-scansources debs | gzip > debs/Sources.gz
```

Dikkat ettiyseniz `dpkg-scansources` uygulaması parametre olarak bir `override` dosyasına ihtiyaç duymamaktadır. `sources.list` dosyanıza eklemeniz gereken satır ise aşağıdadır:

```
deb-src file:/root debs/
```

2.3 sources.list dosyası için en iyi yansıya karar verme: netselect, netselect-apt

Yeni Debian kullanıcılarının kararsız oldukları noktalardan biri `sources.list` dosyasına hangi Debian yansısını eklemeleri gerektiğidir. En iyi yansıya karar vermek için pek çok yöntem mevcuttur. Deneyimli kullanıcılar bir betik programı yazarak yansılar ile aradaki iletişim hızını ölçebilirler. Bu işi sizin için büyük ölçüde yapacak bir program mevcuttur: **netselect**.

netselect programını bildiğimiz yöntemle kuralım:

```
# apt-get install netselect
```

Parametre vermeden programı çalıştırdığımızda bir yardım sayfası gözükecektir. Programı birbirinden boşluk karakteri ile ayrılmış yansı adresleriyle çalıştırırsanız geriye bir skor ve yansı adresi döndürecektir. Buradaki değer yaklaşık olarak ping zamanını, ve ilgili yansıya kaç adımda ulaşılabilindiği bilgisiyle hesaplanmıştır. Eğer parametre olarak verdiğiniz tüm yansılara ait skorları görmek isterseniz programı -vv seçeneği ile çalıştırabilirsiniz. Örnek:

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.unes
  365 ftp.debian.org.br
#
```

Bu çıktının anlamı, netselect programına parametre olarak verdiğimiz yansılar arasından en uygununun `ftp.debian.org.br` olduğu ve bu yansı için elde edilen skor değerinin 365 olduğudur. (Bu değer herkes için farklı olacaktır!).

Şimdi netselect tarafından en hızlı olarak belirtilen yansıyı `/etc/apt/sources.list` dosyanıza (bkz. '/etc/apt/sources.list dosyası' on page 3) ekleyebilir ve ardından 'Paketlerin yönetimi' on page 9 kısmındaki yönergeleri takip edebilirsiniz.

Not: Debian tam yansı listesine `http://www.debian.org/mirror/mirrors_full` adresinden erişebilirsiniz.

0.3 versiyonundan itibaren netselect paketi **netselect-apt** betiğini de içermektedir. Bu betik yukarıdaki işlemleri otomatik olarak yapmaktadır. Dağıtım olarak ne kullanmak istediğinizi belirtmeniz durumunda (öntanımlı olarak stable - kararlı) yansılar arasından sizin için en iyileri seçilecek ve bu karar doğrultusunda `sources.list` dosyanız oluşturulacaktır. Aşağıdaki örnek kullanım ile kararlı dağıtım için bir sources.list dosyası üretilmektedir:

```
# ls sources.list
ls: sources.list: Böyle bir dosya ya da dizin yok
# netselect-apt stable
(...) Bir miktar bekliyoruz
# ls -l sources.list
sources.list
#
```

Unutmayın: `sources.list` dosyası bulunduğunuz dizin altında oluşturulacaktır. Bu dosyayı kullanmak istiyorsanız `/etc/apt` dizini altına taşımalısınız.

Ardından 'Paketlerin yönetimi' on page 9 kısmındaki yönergeleri izleyebilirsiniz.

2.4 sources.list dosyasına CD-ROM ekleme

Eğer paketlerin kurulumu veya güncellemesi için halihazırda elinizde bulunan bir CD-ROM'u kullanmak istiyorsanız bunu `sources.list` dosyanıza eklemelisiniz. Bunun için CD-ROM'u takıp `apt-cdrom` programını örnekteki gibi çalıştırın:

```
# apt-cdrom add
```

CD-ROM bağlanacak ve eğer geçerli bir Debian CD'si ise üzerindeki paketlere ait bilgiler okunacaktır. Eğer CD-ROM ayarlamalarınız biraz standart dışı ise aşağıdaki seçenekleri de kullanabilirsiniz:

```
-h          - program yardımı
-d dizin    - CD-ROM bağlama noktası
-r          - Tanınmış bir CD-ROM'u yeniden adlandır
-m          - Bağlama işlemini gerçekleştirme
-f          - Hızlı mod, paket dosyalarını kontrol etme
```

Örneğin:

```
# apt-cdrom -d /home/kov/mycdrom add
```

Ayrıca sources.list dosyanıza eklemeden CD-ROM'u tanıma işlemini yapabilirsiniz:

```
# apt-cdrom ident
```

Not: Bu program CD-ROM aygıtınıza ait ayarların `/etc/fstab` dosyasında düzgün olarak yapılmış olduğu durumda çalışacaktır.

Chapter 3

Paketlerin yönetimi

3.1 Paket listesini güncelleme

Paket yönetim sistemi kurulu paketler, kurulu olmayan paketler, kurulabilir paketler vb. hakkında bilgileri kendi özel veritabanında tutmaktadır. `apt-get` programı bu veritabanını kullanarak istenilen paketin nasıl kurulacağını, başka hangi paketlerin daha kurulması gerektiğini öğrenir.

Bu listeyi güncellemek için `apt-get update` komutunu kullanmalısınız. Bu komut `/etc/apt/sources.list` dosyanıza bakar ve belirtilen arşivlerden güncel paket listesini indirir; ayrıntılı bilgi için bkz. '/etc/apt/sources.list dosyası' on page 3.

Paket güncellemeri ve güvenlikle ilgili güncellemelerden sizin ve sisteminizin haberdar olabilmesi için bu komutu düzenli aralıklarla çalıştırmakta fayda vardır.

3.2 Paket kurma

Sonunda sabırsızlıkla beklediğiniz bölüme geldik! Artık sources.list dosyanız hazır ve paket listeniz güncel durumda. Tek yapmanız gereken kuracağınız paketi belirtmek. Örneğin:

```
# apt-get install xchat
```

APT hemen kendi veritabanını tarayarak bu paketin en son versiyonunu bulacak ve `sources.list` dosyanızda belirtilen arşivden indirmeye başlayacaktır. Bu paketin başka paketlere de bağımlı olması durumunda - ki örneğimizde de öyle - aynı adımları bağımlılık yaratan paketler için de yapacaktır. Aşağıdaki örneğe bakınız:

```
# apt-get install nautilus
Reading Package Lists... Done
Building Dependency Tree... Done
```

```
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

`nautilus` paketi bazı kütüphane paketlerine bağımlı durumdadır. Eğer bağımlılık yaratan paketleri de komutumuza eklemiş olsaydık APT devam etmek istiyor musunuz? sorusunu sormayacak ve hemen paketleri indirmeye başlayacaktı.

Bunun anlamı APT'nin sadece komut satırında belirtilmeyen ancak gereken paketleri de kurmak istediği zaman onay isteyeceğidir.

Aşağıdaki apt-get seçenekleri işinize yarayabilir:

```
-h  Yardım ekranı.
-d  Sadece indir , kurulumu yapma
-f  Bütünlük kontrolü başarısız olsa da devam etmeye çalış
-s  Hiçbir eylem yapma, sadece olayı simüle et
-y  Tüm sorulara Evet cevabı verdiğimi farzet
-u  İşlem sonucunda güncellenecek olan paketleri listele
```

Birden fazla paket kurulmak üzere tek satırda seçim yapılabilir. Gerekli paketler indirilir ve `/var/cache/apt/archives` dizini altına ilerideki kurulumlarda da kullanılmak üzere kaydedilir.

Aynı komut satırında kaldırılmasını istediğiniz paketleri de belirtebilirsiniz. Bunun için kaldırmak istediğiniz paketin ismini sonuna bir '-' karakteri ekleyerek yazmanız yeterli:

```
# apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Paket kaldırmayla ilgili bilgiler için 'Paket kaldırma' on the facing page kısmına bakınız.

Eğer herhangi bir şekilde kurulu bir paketin dosyalarına zarar verdiyseniz veya sadece kurulu bir paketin yeniden indirilip tekrar kurulmasını istiyorsanız `--reinstall` seçeneğini örnekteki gibi kullanmalısınız.

```
# apt-get --reinstall install gdm
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1  not
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

Bu yardım belgesinin yazımında APT versiyonu 0.5.3 idi. Eğer sisteminizde bu veya daha üst versiyon APT varsa ek bazı fonksiyonlara sahipsiniz: paketleri belirli bir dağıtım arşivinden `apt-get install paket/dağıtım` komutu ile kurabilir veya `apt-get install paket=versiyon` komutu ile spesifik bir versiyonunu kurabilirsiniz. Örneğin:

```
# apt-get install nautilus/unstable
```

komutu siz 'stable' dağıtımla çalışıyor olsanız bile nautilus paketini 'unstable' dağıtımından kuracaktır. Dağıtım alanı için kabul edilen değerler `stable`, `testing`, ve `unstable`'dır.

Hedef dağıtım seçerken `-t` anahtarını da kullanabilirsiniz, bu durumda `apt-get` bağımlılıkları bu dağıtıma göre düzenleyecektir.

ÖNEMLİ: 'unstable' dağıtım yeni Debian paketlerinin upload edildiği yerdir. Çok sık güncellen bu arşiv yeni başlayanlar veya sisteminde kararlılık arayanlar tarafından *kullanılmamalıdır*.

'testing' dağıtımı 'unstable' dağıtımdan kararlılık anlamında çok daha iyidir, ancak kritik sistemlerde mutlaka kararlı dağıtım kullanılmalıdır.

3.3 Paket kaldırma

Bir paketi artık kullanmak istemiyorsanız APT ile sisteminizden paketi kaldırabilirsiniz. Bunun için `apt-get remove package` komutunu vermeniz yeterli olacaktır. Örneğin:

```
# apt-get remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Yukarıda da görüldüğü üzere APT kaldırılan pakete bağımlı olan paketler üzerinde de işlem yapmaktadır. Yukarıdaki soruyu onayladığınızda listelenen tüm paketler kaldırılacaktır. APT kullanarak bir paketi, o pakete bağımlı olan diğer paketleri kaldırmadan kaldırmanın :) bir yolu yoktur.

`apt-get`'in yukarıdaki gibi çalıştırılması sonrasında listelenen paketler sistemden kaldırılacak ancak paketlere ait konfigürasyon dosyaları -eğer varsa- sistemde kalacaktır. Konfigürasyon dosyaları ile birlikte paketleri tamamen kaldırmak için aşağıdaki komutu çalıştırın:

```
# apt-get --purge remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Burada isminden sonra '*' karakteri olan paketlere ait konfigürasyon dosyalarının da kaldırılacağını anlıyoruz.

Kurulum `install` yönteminde olduğu gibi kaldırma `remove` işleminde de aynı satırda kurmak istediğiniz paketleri belirtebilirsiniz. Bunun için kurulmasını istediğiniz paketin sonuna `'+'` karakteri eklemeniz gereklidir.

```
# apt-get --purge remove gnome-panel nautilus+
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

`apt-get` burada kaldırma işleminin yanı sıra kurulmasını istediğimiz paketi ve gerektirdiği diğer paketlerle birlikte listeler.

3.4 Paket güncelleme

Paket güncellemeleri APT'nin çok başarılı olduğu bir işlemdir. Tek bir `apt-get upgrade` komutu ile tüm paketlerin güncellenmesi sağlanabilmektedir. Bu komutu, sisteminizdeki paketleri kullandığınız dağıtımdaki programların güncel sürümlerine yükseltmek veya tümüyle

yeni bir Debian sürümüne yükseltme yapmak amacıyla kullanabilirsiniz. Dağıtım yükseltmeleri için önerilen yol `apt-get dist-upgrade` kullanılmasıdır;; ayrıntılar için 'Yeni bir sürüme güncelleme' on the current page kısmına bakınız.

Komutun `-u` seçeneği ile çalıştırılması oldukça yararlıdır. Bu seçeneğin kullanımıyla APT güncellenecek paketleri listeler. APT paketlerin en son versiyonlarını indirecek ve gereken sırada kurulumu gerçekleştirecektir. Böylesi bir güncelleme yapmadan önce `apt-get update` komutu ile paket listenizi de güncellemeniz önemlidir. Aşağıdaki örneğe bakalım:

```
# apt-get -u upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  cpp gcc lilo
The following packages will be upgraded
  adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp indent
  ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
  libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
  libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
Do you want to continue? [Y/n]
```

İşlem oldukça basittir. İlk bir kaç satırda `apt-get` bazı programların sistemde tutulmaya devam edileceğini `kept back` söylemektedir. Bunun anlamı, listelenen programların yeni versiyonları olmasına rağmen herhangi bir nedenden ötürü sisteminize kurulamayacak olmasıdır. Büyük olasılıkla ilgili paketler, paket listenizde yer almayan paketlerin veya paket versiyonlarının kurulmasını gerektirmekte, bu işlem yapılamayacağı için de paketler güncellenmemektedir. İkinci olası neden, paketin yeni versiyonunun sisteminizde daha önce olmayan yeni bir pakete bağımlı olmadır.

Birinci senaryo için kesin bir çözüm yoktur. İkinci durumda problem yaratan paketi `apt-get install` komutu ile kurarak sorunu çözebilirsiniz. Daha temiz bir yöntem ise `dist-upgrade` kullanmaktır, bu sayede sistemde daha önce olmayan yeni paketler de kurulacaktır, bkz. 'Yeni bir sürüme güncelleme' on this page.

3.5 Yeni bir sürüme güncelleme

APT'nin bu özelliği sayesinde bir defada tüm sisteminizi yeni bir Debian sürümüne ister internet üzerinden isterse varolan bir CD üzerinden yükseltebilirsiniz.

Ayrıca bu yöntem kurulu paketler arasındaki paket bağımlılıklarında değişmeler olduğunda da kullanılır. `apt-get upgrade` ile yapılan güncellemelerde bu tür paketlere dokunulmaz.

Örnek olarak, Debian 0 sürümünü kullandığınızı düşünün, ve 3. sürümü içeren bir CD edinmiş olun. APT'yi kullanarak sisteminizi bu CD üzerinden yeni sürüme güncelleyebilirsiniz.

Bunu yapmak için öncelikle `apt-cdrom` ile (bkz. 'sources.list dosyasına CD-ROM ekleme' on page 6) ile CD'yi `/etc/apt/sources.list` dosyanıza ekleyin ve ardından `apt-get dist-upgrade` komutunu çalıştırın.

Unutmayın, APT her zaman paketin güncel versiyonunu arar. Eğer `/etc/apt/sources.list` dosyanızda belirtilen arşivlerde bir paketin CD üzerindekinden daha yeni bir versiyonu var ise güncelleme sırasında bu versiyon kullanılacaktır.

'Paket güncelleme' on page 12 kısmındaki örneğimizde bazı paketlerin güncelleme sırasında korunduğunu `kept back` görmüştük. Bu problemi şimdi `dist-upgrade` ile çözebiliriz:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

Görüldüğü gibi paketler hem güncellenecek hem de yeni bağımlılıklar için yeni paketler de kurulacaktır. Ancak lilo paketi halen sistemde korunmaya devam etmektedir `kept back`. Muhtemelen bağımlılık dışında daha ciddi bir problemi vardır. Problemin ne olduğunu aşağıdaki komutla görebiliriz:

```
# apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
```

```
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

Yukarıdaki belirtildiği üzere, lilo paketi `debconf-tiny` paketi ile çakışmaktadır. Bunun anlamı, debconf-tiny paketi kaldırılmadan lilo paketinin yüklenemeyeceği ve güncellenemeyeceğidir.

Korunan veya kaldırılan paketlerle ilgili daha detaylı bilgi almak için komutu aşağıdaki gibi kullanabilirsiniz:

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
 Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2  not upgraded.
```

Burada açıkça görüldüğü gibi python1.5-dev paketi kurulamayacaktır çünkü python1.5 paketine bağımlıdır. python1.5 kurulamadığı için python1.5-dev paketi de kurulamamaktadır.

3.6 Kullanılmayan paket dosyalarını temizleme: apt-get clean ve autoclean

Bir paket kuracağınız zaman APT gerekli dosyaları /etc/apt/sources.list dosyanızda belirtilen host'lardan alır ve yerel bir dizin (`/var/cache/apt/archives/`) altında saklar. Ardından kurulumu buradaki dosyalar üzerinden gerçekleştirir, bkz. 'Paket kurma' on page 9.

Zamanla dosyaların tutulduğu yerel dizinin boyutları artmaya ve gereksiz yer kaplamaya başlar. APT bu dizini temizlemek için bir takım fonksiyonlara sahiptir: `apt-get`'in `clean` ve `autoclean` seçenekleri.

`apt-get clean` lock dosyaları haricinde `/var/cache/apt/archives/` ve `/var/cache/apt/archives/partial/` dizinlerindeki tüm dosyları siler. Sonuç olarak, eğer bir paketi yeniden kurmak isterseniz (–reinstall) APT gerekli dosyalara artık sahip olmadığından, yeninden indirecektir.

`apt-get autoclean` ise sadece artık indirilmesi mümkün olmayan dosyaları siler.

Aşağıdaki örnek apt-get autoclean özelliğinin nasıl çalıştığını gösteriyor:

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

/var/cache/apt/archives altında `logrotate` paketi için 2 dosya ve `gpm` paketi için de bir 1 dosya bulunmaktadır.

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

`apt-show-versions` çıktısına baktığımızda `logrotate_3.5.9-8_i386.deb` dosyasının `logrotate` paketi için güncel versiyon olduğunu görüyoruz. Bu durumda `logrotate_3.5.9-7_i386.deb` dosyası hiç bir işe yaramamaktadır. Ayrıca `gpm_1.19.6-11_i386.deb` dosyası da, indirilebilecek daha güncel bir versiyonu olduğu için işe yaramamaktadır.

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

Sonuçta, `apt-get autoclean` ile sadece eski ve işe yaramayacak dosyaların kaldırılmasını sağlamış olduk, bkz. 'Sadece belirli bir Debian versiyonuna sahip paketleri güncelleme' on page 18.

3.7 APT ile dselect kullanımı

`dselect`, kurulum için Debian paketleri seçebileceğiniz bir programdır. Kullanımı oldukça karışık ve zor olduğundan yeni kullanıcılar bazen program içerisinde yönlerini kaybedebilirler.

dselect'in özelliklerinden biri, Debian paketlerinin "şiddetle önerilen" ve "tavsiye edilen" paketlere ait bilgilerini işleyebilmesidir ("recommending" ve "suggesting"). Programı çalıştırmak için root iken `dselect` komutunu verin. Erişim metodları menüsünden 'apt' seçimini yapın. Mutlaka gerekli olmamakla birlikte, eğer bir CDROM kullanmıyor ve paket indirmek istiyorsanız erişim metodu olarak 'apt' kullanmalısınız.

dselect kullanımı hakkında ayrıntılı bilgi almak isterseniz dselect dokümantasyonunun bulunduğu `http://www.debian.org/doc/ddp` adresini ziyaret edebilirsiniz.

dselect ile seçimlerinizi yaptıktan sonra, aşağıdaki komutu çalıştırın:

```
# apt-get -u dselect-upgrade
```

Örnek:

```
# apt-get -u dselect-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

Şimdi bu ekran çıktısını aynı sistemdeki apt-get dist-upgrade çıktısı ile karşılaştıralım:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

Görüldüğü gibi dselect ile yapılan güncellemede yukarıdaki listede olmayan paketler de yüklenmek üzere seçilmiştir. Çünkü güncellenecek olan paketlerin "şiddetle önerilen" ve "tavsiye edilen" paketler alanları da incelenmiş, bu paketler listeye dahil edilmiştir. Dselect APT ile birlikte kullanıldığında oldukça güçlü bir araçtır.

3.8 Karışık bir sistem nasıl kurulur?

Kullanıcılar sıklıkla testing dağıtımını da kullanırlar, çünkü testing dağıtımı, kararsız dağıtımdan çok daha kararlı bir yapıya sahiptir ve kullanılabilir durumdadır. Bazı kullanıcılar ise sistemlerindeki bazı paketlerin son versiyonlarını kullanmak isterler. Bu durumda testing/unstable karışımı bir sistem oluşabilir. Gene aynı şekilde başka nedenlerden ötürü stable/testing karışımı bir dağıtım da oluşturulabilir.

Bunu yapabilmek için aşağıdaki satırı `/etc/apt/apt.conf` dosyanıza ekleyin:

```
APT::Default-Release "testing";
```

Ardından, unstable dağıtımdan bir paket yüklemek istediğinizde `-t` seçeneğini kullanın:

```
# apt-get -t unstable install paket_adı
```

Kullanacağınız her bir dağıtım için `/etc/apt/sources.list` dosyasına gerekli kayıtları girmelisiniz. Bu örnekte dosyada `unstable` ve `testing` dağıtımı için gerekli kayıtlar olmalıdır. Bu kayıtlar olmadan APT, ilgili dağıtımdaki bir pakete ait bilgiye sahip olamaz.

3.9 Sadece belirli bir Debian versiyonuna sahip paketleri güncelleme

`apt-show-versions` programı, karışık bir sistem kullanan kullanıcıların sistemlerini güncellemelerinde daha güvenli bir ortam sunar. Örneğin aşağıdaki gibi sadece sisteminizdeki unstable paketleri güncelleyebilirsiniz:

```
# apt-get install ‘apt-show-versions -u -b | grep unstable‘
```

3.10 Belirli bir versiyona sahip paketlerin kurulu olarak kalmasını sağlama (ileri düzey)

Bazen çeşitli nedenlerden ötürü bir paketin mutlaka belirli bir versiyonu ile çalışmanız gerekebilir. Güncelleme yaptığınızda ise paket yeni versiyonu ile değiştirileceğinden sorun çıkacaktır. Örneğin sisteminizi Debian 3.0 sürümüne güncellemiş fakat belirli bir paketin Debian 2.2 sürümündeki halini kullanmaya devam etmek istiyor olabilirsiniz. Bu problemi çözmek için paketlerinizi “pin”leyebilirsiniz.

Bu özelliği kullanmak oldukça kolaydır, tek yapmanız gereken `/etc/apt/preferences` dosyasını düzenlemekten ibaret.

Dosyanın formatı şu şekildedir:

```
Package: <paket_adı>
Pin: <pin tanımlaması>
Priority: <pin önceliği>
```

Örnek olarak, `sylpheed` paketinin 0.4.99 versiyonunu korumak isterseniz aşağıdaki kayıtı buraya ekleyebilirsiniz:

```
Package: sylpheed
Pin: version 0.4.99*
```

Buradaki * kullanımına dikkat! Bu karakter ile girdiğimiz pin değerinin tüm 0.4.99 ile başlayan versiyonlar için geçerli olduğunu belirtebiliyoruz. Bu sayede program versiyonunun yanı sıra paketin Debian paket versiyonunun değişmesi durumunda da koyduğunuz kurallar geçerli olur, örneğin 0.4.99-1 versiyonu ile 0.4.99-10 versiyonu.

`Priority` alanı seçime bağlıdır, girilmediği takdirde öntanımlı değeri 989'dur.

Şimdi de pin önceliklerinin nasıl çalıştığına bakalım. 0 değerinden daha küçük bir önceliğin anlamı paketin hiç bir zaman kurulamayacağıdır. 0 ile 100 arasındaki öncelikler paketin kurulmadığını ve erişilebilir yeni versiyonu olmadığını belirtir. 100 öncelik değeri kurulu olan pakete tanımlanır, paketin kurulu versiyonu yeni bir versiyon ile değiştirilebilir, fakat yeni versiyonun 100'den daha büyük bir önceliğe sahip olması gereklidir.

Öncelik değeri 100'den büyükse paket kurulmalıdır. Tipik olarak paketin kurulu versiyonu sadece daha yeni bir versiyona güncelleme ile değişir. Böyle bir önceliğe sahip paket, daha düşük versiyonu ile değiştirilemez (downgrade). Örneğin, eğer sylpheed 0.5.3 versiyonu kurulu ve sylpheed 0.4.99 versiyonu için önceliği 999 olan bir pin tanımlı ise, 0.4.99 paketi bu pin dolayısıyla *kurulamayacaktır*. Bir paketin eski versiyonları ile değiştirilebilir olmasını pin değerleri de gözönüne alınarak sağlamak için ("downgradable") öncelik değeri 1000'den büyük hale getirilmelidir.

Bir pin değeri paketin `versiyonu`, `sürümü` ve `adına` göre verilebilir.

Yukarıda gördüğümüz gibi `versiyona` pin değeri vermede versiyon numaraları ve özel karakterler kullanılır.

`Sürüme` pin değeri verme işlemi APT'nin kullandığı arşivlerdeki veya CD'deki Release dosyası ile ilişkilidir. Eğer kullandığınız arşivler Release dosyasını sağlamıyorsa bu özellik kullanılamaz. Release dosyanızın içeriğini `/var/lib/apt/lists/` dizini altından görebilirsiniz. Sürüm için kullanılabilecek parametreler `a` (arşiv), `c` (components, bölüm), `v` (versiyon), `o` (origin, kaynak) ve `l` (label, etiket) tir.

Örnek:

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Priority: 1001
```

Bu örnekte Debian 2.2* versiyonlarını (2.2r2, 2.2r3 vb.), `kararlı, stable` arşivi `main` bölümü (`contrib` veya `non-free` de olabilirdi), kaynak ve etiket olarak Debian'ı seçtik. Kaynak (o=) Release dosyasını kimin ürettiğini belirtir, etiket (l=) ise dağıtımın ismidir (Debian dağıtımı için etiket Debian iken Progeny için Progeny'dir). Örnek bir Release dosyasının içeriği aşağıdadır:

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i3
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

Chapter 4

Yardımcı araçlar

4.1 Kendi derlediğim paketleri nasıl kuracağım: equivs

Bazen bir programın belirli bir versiyonunu kullanmak zorunda kalabiliriz. Bu versiyona ait bir Debian paketi mevcut değil ise, programı kaynak koddan derleyerek kendimiz kurarız. Fakat paket yönetim sistemi bu durumda şaşırabilir. Mesela kullandığınız mail sunucu yazılımının yeni bir versiyonunu derlediğinizi düşünün. Burada problem yok, ancak pek çok Debian paketi sistemde bir MTA (Mail Transport Agent) yazılımı olmasını gerektirir. Mail sunucunuzu kendiniz derleyerek kurduğunuz için paket yönetim sistemi bundan haberdar değildir ve bağımlılık hataları oluşur.

Burada `equivs` programı devreye girmektedir. Programı kurmak için aynı isimli paketi sisteminize kurmalısınız. Bu program ile boş bir paket yaratıp, bağımlılık problemlerini çözebilir ve paket yönetim sisteminin içini rahatlatabilirsiniz.

Başlamadan önce hatırlatmak isteriz ki, Debian için zaten paketi hazırlanmış bir programı farklı derleme seçenekleriyle derlemenin daha güvenli ve uygun yolları vardır. Eğer ne yaptığınızı tam olarak bilmiyorsanız equivs kullanmayınız. Ayrıntılı bilgi için bkz. 'Kaynak paketlerle çalışma' on page 31.

MTA örneğimize kaldığımız yerden devam edelim. Yeni derlemiş olduğunuz `postfix` programını sisteminize kurdunuz ve ardından `mutt` paketini kurmak istediniz. Fakat `mutt` paketinin başka bir MTA kurulmasını gerektirdiğini gördünüz ama zaten sisteminizde bir MTA var !

Herhangi bir dizine geçip (örneğin `/tmp`) aşağıdaki komutu çalıştırın:

```
# equivs-control isim
```

isim değerini oluşturmak istediğiniz control dosyasının adı ile değiştiriniz. Ardından dosya aşağıdaki şekilde oluşturulacaktır:

```
Section: misc
```

```
Priority: optional
Standards-Version: 3.0.1

Package: <enter package name; defaults to equivs-dummy>
Version: <enter version here; defaults to 1.0>
Maintainer: <your name and email address; defaults to username>
Pre-Depends: <packages>
Depends: <packages>
Recommends: <packages>
Suggests: <package>
Provides: <(virtual)package>
Architecture: all
Copyright: <copyright file; defaults to GPL2>
Changelog: <changelog file; defaults to a generic changelog>
Readme: <README.Debian file; defaults to a generic one>
Extra-Files: <additional files for the doc directory, commaseperated>
Description: <short description; defaults to some wise words>
 long description and info
 .
 second paragraph
```

Bu dosyayı kendi isteğimize göre değiştirebiliriz. Tüm alanları açıklamaya şimdilik burada gerek yok, hemen amacımızı gerçekleştirelim:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

Evet, hepsi bundan ibaret. `mutt` paketi sistemde `mail-transport-agent` paketi olmasını gerektirir. Bu sanal bir pakettir ve tüm MTA'lar tarafından sağlanır. Aynı davranış için paket ismi olarak doğrudan `mail-transport-agent` seçebilirdik, ancak sanal paket kavramını ve Provides alanını kullanarak bunu gerçekleştirdik.

Şimdi yapmanız gereken paketi oluşturmaktır:

```
# equivs-build isim
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Add here commands to install the package into debian/tmp.
```

```
touch install-stamp
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package ‘isim’ in ‘../isim_1.0_all.deb’.

The package has been created.
Attention, the package has been created in the current directory,
```

Ve sonra üretilen `.deb` paketini sisteminize kurmalısınız.

Gördüğünüz gibi `equivs` programının kullanılabileceği pek çok durum vardır. Örneğin `favorilerim` adlı bir paket oluşturabilir ve bu paketin sıklıkla kullandığınız paketlere bağımlı olmasını sağlayabilirsiniz (Depends alanı ile). Ardından bu paketi sisteminize kurup, bağımlılık verdiğiniz diğer tüm paketlerin de kurulmasını sağlamış olursunuz.

Örnek control dosyaları için `/usr/share/doc/equivs/examples` dizini altına bakabilirsiniz.

4.2 Kullanılmayan yerelleştirme dosyalarını kaldırma: localepurge

Çoğu Debian kullanıcısı sadece bir tip yerel dosyası kullanır. Örneğin bir Türk genellikle `tr_TR` yerelini kullanır ve hemen hiç bir zaman `es` kullanma ihtiyacı hissetmez.

`localepurge` paketi bu kullanıcılar için oldukça faydalıdır. Kullanmadığınız yerellerin sisteminizden kaldırılmasını ve yenilerinin de kurulmamasını sağlayarak disk üzerinde size yer kazandırır. Yapmanız gerekeni biliyorsunuz: `apt-get install localepurge`

Paketin ayarlamaları oldukça kolaydır, debconf tarafından ilgili sorular size yöneltilecektir. Ancak ilk soruya cevap verirken dikkatli olun, yanlış bir seçim sonucu kullandığınız yerel dosyalarının silinmesine neden olabilirsiniz. Bu durumda tek çözüm, ilgili yerel dosyalarını içeren paketleri sisteminize yeniden kurmak olacaktır.

4.3 Güncellenebilir paketleri nasıl öğrenebilirim?

`apt-show-versions` programı sisteminizde güncellenmeye müsait olan paketler hakkında faydalı bilgiler sunar. `-u` seçeneği ile güncellenebilir paketlerin listesini alabilirsiniz:

```
$ apt-show-versions -u
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

Chapter 5

Paketler hakkında bilgi toplama

Paketler hakkında bilgi toplamak için APT sisteminin üzerinde çalışan programlar bulunmaktadır.

Fakat burada amacımız APT ile aynı bilgilere nasıl erişebileceğinizi sizlere anlatmaktır. Mesela, kurmak istediğiniz programın paket adını nasıl öğrenebiliriz?

Bu amaç için birkaç yardımcı araca sahibiz. Önce `apt-cache` ile başlayalım. Bu program APT sistemi tarafından kendi özel veritabanını yönetmede kullanılır. Şimdi pratik kullanımda uygulanmasına geçelim.

5.1 Paket isimlerini keşfetme

Örneğin, eski iyi günleri hatırladınız, Atari 2600 günlerinizi... Ve bir Atari emülator programı kurmak için APT'yi kullanmak istiyorsunuz ancak kurmanız gereken paketin ismini bilmiyorsunuz. Yapmanız gereken:

```
# apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

Gördüğünüz gibi arama sonucumuzda birkaç paket kısa açıklamalarıyla birlikte listelendi. Herhangi biri hakkında daha ayrıntılı bilgi almak istersek kullanacağımız komut:

```
# apt-cache show stella
Package: stella
```

```
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

Bu defaki ekran çıktısında paket hakkında daha ayrıntılı bilgi var ve bu bilgiler doğrultusunda paketi kurmak isteyip istemediğimize karar verebiliriz. Eğer paket sisteminizde kurulu fakat yeni bir versiyonu mevcut ise, her iki versiyon hakkındaki bilgiler listelenecektir. Örnek:

```
# apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
Section: base
```

```
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Burada ilk sırada yeni versiyona ait bilgiler, ikinci sırada ise zaten sisteminizde kurulu olan versiyona ait bilgiler listelenir. Bir paket hakkında daha genel bilgi edinmek için aşağıdaki komutu kullanabilirsiniz:

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_mai

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libs
Provides:
1.4.5-1 -
Reverse Provides:
```

Bir paketin bağımlı olduğu paketlerin listesini almak için:

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

Özetle, aradığımız paket adını bulmak için bir kaç silaha sahibiz.

5.2 Paket adlarını bulmak için dpkg kullanma

Paket adını bulmada diğer bir yöntem de o paket tarafından kullanıldığını bildiğiniz bir dosyadan hareket etmektir. Örneğin belirli bir `".h"` dosyasını içerdiğini bildiğiniz paketin

ismini aşağıdaki gibi bulabilirsiniz:

```
# dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

veya:

```
# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

Eğer sisteminize kurulu bir paketin tam ismini öğrenmek isterseniz aşağıdaki komut işinizi görecektir:

```
# dpkg -l | grep mozilla
ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

Bu komut paket isimlerini belirli bir uzunluktan sonra kesmektedir. Yukarıdaki örnekte paketin tam ismi `mozilla-browser`'dır. Bu problemi gidermek için `COLUMNS` çevresel değişkenini aşağıdaki gibi ayarlamalısınız:

```
[kov]@[couve] $ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser             0.9.6-7                     Mozilla Web Brows
```

veya açıklamasından yola çıkarak paket adını elde edebilirsiniz:

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

5.3 Programları anında kurma

Bir program derliyorsunuz ve aniden bir hata oluştu! Çünkü program sisteminizde olmayan bir `.h` dosyasına ihtiyaç duyuyor. `auto-apt` programı sizi bu senaryolardan kurtaracaktır. Gerektiğinde ilgili paketi kurmak isteyip istemediğinizi soracak, derleme işlemini durduracak ve programı çekip kurduktan sonra derleme işleminin devam etmesini sağlayacaktır.

Peki nasıl yapacaksınız:

```
# auto-apt run komut
```

Burada komut yerine derleme için kullanacağınız komutu yazmalısınız.

```
# auto-apt run ./configure
```

Ardından gerekli paketleri kurmak isteyip istemediğinizi soracak, apt-get'i çağırarak kurulumları otomatik olarak gerçekleştirecektir.

Auto-apt programı efektif olarak çalışabilmek için kendi özel veritabanını tutar. Bu veritabanının güncel tutulması programın beklenilen davranışı gösterebilmesi açısından önemlidir. Veritabanını güncellemek için `auto-apt update`, `auto-apt updatedb` ve `auto-apt update-local` komutlarını çalıştırmalısınız.

5.4 Bir dosyanın hangi pakete ait olduğunu bulma

Bir paketi kurdunuz ancak `apt-cache` ile nasıl çağıracağınızı bilmiyorsunuz. Fakat programın dosya adını biliyorsunuz, veya paketten çıkan başka herhangi bir dosyanın adını bilmektesiniz. Bu durumda `apt-file` programı ile paket adını bulabilirsiniz:

```
$ apt-file search dosya_adı
```

Bu komut aynı `dpkg -S` gibi çalışır ancak ayrıca verdiğiniz dosyayı içermesine rağmen sisteminizden sonradan kaldırılmış olan paketleri de listeler.

Bir paketin içeriğini aşağıdaki komutla listeleyebilirsiniz:

```
$ apt-file list paket_adı
```

`apt-file` tüm paketler hakkında içerdikleri dosyalara ilişkin kayıtları tutar. Aynı auto-apt'de olduğu gibi bu veritabanının da zamanla güncellenmesi gereklidir:

```
# apt-file update
```

Öntanımlı olarak `apt-file` `auto-apt` ile aynı kaynağı kullanır, bkz. 'Programları anında kurma' on the facing page.

5.5 Paketlerdeki değişikliklerden haberdar olma

Sisteminize kurulan her paket kendi dokümantasyon dizini altına (`/usr/share/doc/paket_adı`) `changelog.Debian.gz` adlı bir dosya atar. Bu dosyada ilk versiyondan son versiyona kadar paket üzerinde yapılan değişiklikler belirtilir. Bu dosyayı `zless` ile okuyabilirsiniz ancak tüm paketleri bu şekilde takip etmek epey güç bir uğraştır.

Bu işi otomatik hale getirmek için yardımcı bir araç mevcuttur: `apt-listchanges`. Programı kullanmak için önce `apt-listchanges` paketini sisteminize kurmalısınız. Kurulum esnasında Debconf tarafından sorulan soruları istediğiniz gibi yanıtlayabilirsiniz.

"Should apt-listchanges be automatically run by apt?" seçeneği oldukça yararlıdır, bu seçenek ile paket kurulum ve güncellemelerinde, paketteki değişiklikleri görebilirsiniz. "Should apt-listchanges prompt for confirmation after displaying changes?" seçeneği ile değişiklikleri okuduktan sonra programın kurulumu işlemine onay istenmesini sağlayabilirsiniz. Eğer devam etmek istemediğinizi söylerseniz apt-listchanges bir hata kodu döndürür ve apt-get işlemi sonlandırır.

Chapter 6

Kaynak paketlerle çalışma

6.1 Kaynak paketleri indirme

Özgür yazılım dünyasında kaynak kodlar üzerinde çalışma, hata bulma ve düzeltme oldukça yaygındır. Bunu yapabilmek için programın kaynak kodlarını indirmeniz gereklidir. APT sistemi, dağıtım içerisinde yer alan paketlerin kaynak kodlarını, .deb paketi oluşturulabilmesi için yapılan değişiklikleri de içerecek şekilde indirmenizi sağlayacak fonksiyonlara sahiptir.

Debian kaynak paketlerinin kullanılmasının gerektiği diğer bir durum ise, belirli bir paketin unstable dağıtımdaki versiyonunu stable dağıtım için yeniden hazırlamak istediğiniz durumdur. Bir paketi stable dağıtım için oluşturduğunuzda bağımlılıklar da stable dağıtıma göre düzenlenir.

Bunun için `/etc/apt/sources.list` dosyanızda ilgili `deb-src` satırlarının bulunması gereklidir. Ayrıntı için bkz. '/etc/apt/sources.list dosyası' on page 3.

Bir kaynak paketi indirmek için aşağıdaki komutu kullanınız:

```
# apt-get source paket_adı
```

Bu komut üç dosyanın indirilmesini sağlayacaktır: bir `.orig.tar.gz`, bir `.dsc` ve bir de `.diff.gz`. Eğer paket Debian'a özgü ise, son iki dosya indirilmez ve dosya adı büyük olasılıkla orig ön-ekini içermez.

`.dsc` dosyası dpkg-source tarafından kaynak paketi *paket_adı-versiyon* dizini altına açmak için kullanılır. İndirilen her bir debian kaynak paketi açıldıktan sonra, `debian` adında bir dizin içerir. Bu dizinde .deb paketini oluşturmak için yapılan tüm değişiklikler bulunmaktadır.

Kaynak paket indirildikten hemen sonra .deb paketinin oluşturulmasını istiyorsanız autobuild seçeneği işinize yarayacaktır. Bunun için komutu `-b` anahtarı ile aşağıdaki gibi kullanmalısınız:

```
# apt-get -b source paket_adı
```

Eğer indirdikten hemen sonra yerine daha ileriki zamanlarda .deb paketini oluşturmak istersseniz aşağıdaki komutla bunu yapabilirsiniz:

```
# dpkg-buildpackage -rfakeroot -uc -b
```

Bu komutu kaynak paketin açıldığı dizin içerisindeyken çalıştırmalısınız. İşlem sonunda üretilen paketi aşağıdaki gibi sisteminize kurabilirsiniz:

```
# dpkg -i paket.deb
```

`apt-get`'in `source` metodu ile diğerleri arasında birtakım farklar bulunmaktadır. Burada paketin indirilebilmesi için root kullanıcısı olunmasına gerek yoktur, indirilen paketler o an bulunulan dizin içerisine açılırlar.

6.2 Kaynak paketleri derlemek için gerekli paketler

Doğal olarak derleyeceğiniz programın ihtiyaç duyduğu kütüphanelerin sisteminizde bulunması zorunludur. Tüm kaynak paketler debian/control dosyalarında 'Build-Depends:' adlı bir alana sahiptir, bu alanda paketin kaynak koddan derlenebilmesi için ihtiyaç duyduğu paketler belirtilir.

APT bu paketleri kolayca indirmek ve kurmak için gerekli fonksiyona sahiptir. `apt-get build-dep paket_adı` komutu ile derlemek istediğiniz paketin Build-Depends alanında belirtilen paketlerin sisteminize kurulması sağlanır.

```
# apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev libgnorba-de
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

Burada `gmc` paketinin kaynak halinden derlenebilmesi için gerekli paketler sisteme kurulmaktadır. Bu komutun kaynak paketi indirmediğini unutmayın, kaynak paketi `apt-get source` ile ayrıca indirmelisiniz.

Eğer amacınız belirli bir kaynak paketi derleyebilmek için gereken paketlerin listesini almak ise bunun için `apt-cache show` türevi bir komut kullanılır, bkz. 'Paketler hakkında bilgi toplama' on page 25.

```
# apt-cache showsrc paket_adı
```

Chapter 7

Hatalarla başa çıkma

7.1 Genel hatalar

Hatalı durumlar her zaman olabilir, hataların çoğu basit bir şekilde düzeltilebilecek türdendir. Aşağıda sıklıkla karşılaşılan hatalar ve çözüm yöntemlerinin bir listesi verilmiştir:

Eğer `apt-get install paket_adı` komutunun ardından aşağıdaki gibi bir çıktı aldıysanız...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Pack
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

`/etc/apt/sources.list` dosyasında yaptığınız son değişiklikten sonra `apt-get update` komutunu çalıştırmayı unutmuşsunuz demektir.

Eğer hata aşağıdakine benzer ise:

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root
```

`apt-get`'in `source` dışında bir metodunu normal kullanıcı haklarıyla kullanmaya çalışıyorsunuzdur.

Benzer bir hatayı, aynı anda birden fazla `apt-get` programını çalıştırmanız durumunda ya da `dpkg` programı işlem yaparken APT kullanmaya çalışmanız durumunda alabilirsiniz. Aynı anda kullanılabilecek tek metod `source`'tır.

Eğer kurulum işleminiz herhangi bir nedenle yarıda kesilmiş ve ardından hiç bir paketi kurma ve kaldırma işlemi yapamaz duruma gelmişseniz, aşağıdaki komutlarla APT'nin gerekli düzeltmeleri yapmasını sağlayabilirsiniz:

```
# apt-get -f install
# dpkg --configure -a
```

Ve bazen yukarıdaki komutları birden fazla defa tekrarlamanız gerekebilir. 'unstable' dağıtımı kullanmaya başlayacak olan kullanıcıların öğrenmesi gereken ilk ders budur.

7.2 Nereden yardım bulabilirim?

Debian paket yönetim sistemi hakkındaki ayrıntılı belgeleri inceleyebilirsiniz. Programların `--help` ve kılavuz (man) sayfaları da oldukça yararlı bilgiler içerir. Ayrıca `/usr/share/doc` dizini altındaki her programa ait dokümantasyonu inceleyebilirsiniz, apt için bu dizin `/usr/share/doc/apt`'dir.

Eğer aradığınız sorunun cevabını bu dokümanlarda da bulamazsanız cevabı Debian mail listelerinde aramalısınız. Debian listeleri hakkında ayrıntılı bilgilere `http://www.debian.org` adresinden ulaşabilirsiniz.

Belirttiğimiz listeler Debian kullanıcıları içindir, diğer dağıtımları kullananlar, dağıtımlarının mail listelerinde kendilerine uygun cevapları bulabilirler.

Chapter 8

Hangi Linux dağıtımları APT destekliyor?

Aşağıda APT'yi destekleyen dağıtımların bir listesi yer almaktadır:

Debian GNU/Linux (http://www.debian.org) - APT'nin geliştiriminin yapıldığı ve devam ettiği dağıtımdır.

Conectiva (http://www.conectiva.com.br) - APT sisteminin rpm ile kullanılabilmesi için çalışan ilk dağıtımdır.

Mandrake (http://www.mandrake.com)

PLD (http://www.pld.org.pl)

Vine (http://www.vinelinux.org)

APT4RPM (http://apt4rpm.sf.net)

Alt Linux (http://www.altlinux.ru/)

Red Hat (http://www.redhat.com/)

Sun Solaris (http://www.sun.com/)

SuSE (http://www.suse.de/)

Yellow Dog Linux (http://www.yellowdoglinux.com/)

Chapter 9

Teşekkürler

Debian-BR ve Debian projesindeki arkadaşlarıma, desteklerini esirgemedikleri için teşekkürlerimi sunuyorum.

Ayrıca projemize ve benzer özgür projelere destek veren CIPSGA'ya teşekkür ediyorum.

Ve özel teşekkürler:

Yooseong Yang <yooseong@debian.org>

Michael Bramer <grisu@debian.org>

Bryan Stillwell <bryan@bokeoa.com>

Pawel Tecza <pawel.tecza@poczta.fm>

Hugo Mora <h.mora@melix.com.mx>

Luca Monducci <luca.mo@tiscali.it>

Tomohiro KUBOTA <kubota@debian.org>

Pablo Lorenzzoni <spectra@debian.org>

Steve Langasek <vorlon@netexpress.net>

Arnaldo Carvalho de Melo <acme@conectiva.com.br>

Erik Rossen <rossen@freesurf.ch>

Ross Boylan <RossBoylan@stanfordalumni.org>

Matt Kraai <kraai@debian.org>

Aaron M. Ucko <ucko@debian.org>

Jon Åslund <d98-jas@nada.kth.se>

Chapter 10

Bu dokümanın yeni versiyonları

Bu doküman Debian-BR (http://www.debian-br.org) projesi kapsamında üretilmiş olup, tğm Debian kullanıcılarına yardımcı olması amaçlanmıştır.

Dokümanın yeni versiyonların her zaman Debian Dokümantasyon Projesi sayfalarından erişebilirsiniz: http://www.debian.org/doc/ddp.

Her türlü yorum ve görüşlerinizi <kov@debian.org> adresinden bana gönderebilirsiniz.