# Debian Paket Bakımcılarının Kılavuzu

Yazan:
**Josip Rodin**
<joy–mg (at) debian.org>>

Çeviren:
**Oğuz Yarımtepe**
<oguzy (at) comu.edu.tr>

Eylül 2005

**Özet**

Bu belgede ortalama Debian kullanıcıları ve ayrıntılı tarif isteyen geliştiriciler için bir Debian paketinin nasıl hazırlanacağı açıklanmaya çalışılmıştır.

Bu belge şu iki belge örnek alınarak oluşturulmuştur:

- Bir Debian Paketinin Yapımı (Making a Debian Package) telif hakkı © 1997 Jaldhar Vyas Debmake Kılavuzu (Debmake Manual) olarak da bilinir.
- Yeni Paketleyici için Debian Paketlerinin Oluşturması Nasıl (The New–Maintainer's Debian Packaging Howto), telif hakkı © 1997 Will Lowe.

## Konu Başlıkları

### Bu çevirinin sürüm bilgileri:

| | | |
|---|---|---|
| 1.0 | Eylül 2005 | OY |
| İlk çeviri | | |

### Özgün belgenin sürüm bilgileri:

| | | |
|---|---|---|
| 1.2.3 | 18 Ocak 2005 | JR |



### Yasal Açıklamalar

# 1. Doğru şekilde başlamak

Bu belgede ortalama Debian kullanıcıları ve ayrıntılı tarif isteyen geliştiriciler için bir Debian paketinin nasıl hazırlanacağı açıklanmaya çalışılmıştır. Oldukça genel bir dil kullanılmış ve de çalışma örnekleriyle kapsanmıştır. Romalılara ait eski bir deyiş vardır, *Longum iter est per preaecepta, breve et efficax per exempla!* (Bu, kurallarla uzun, fakat örneklerle kısa ve verimli bir yoldur!).

Debian'ı lider Linux dağıtımlarından birisi yapan paket sistemidir. Hali hazırda çok geniş bir yazılım Debian biçiminde bulunmasına rağmen, olmayan bir yazılımı yükleme ihtiyacı hissedebilirsiniz. Kendi paketlerinizi nasıl yapabileceğinizi düşünüyor olabilirsiniz ve belki de bunun çok güç bir iş olduğunu düşünüyorsunuz. Eğer Linux üzerinde gerçek bir acemiyseniz, o zaman zordur, eğer bir çaylaksanız, şu an bu belgeyi okumuyor olmalıydınız. `:-)` Unix'de yazılım geliştirme hakkında az bir şeyler bilmeye ihtiyacınız vardır, elbette bir sihirbaz olmanıza gerek yok.

Herşeye rağmen kesin olan bir nokta vardır: düzgün bir şekilde bir Debian paketi oluşturmak ve sürdürmek için saatlerce emek harcamanız gerekmektedir. Hata yapmayın, sistemimizin çalışması için paket bakımcılarının teknik olarak yeterli ve gayretli olması gerekmektedir.

Bu belge en küçük (başlangıçta belki de ilgisiz) adımı bile açıklayacak, ilk paketi yapmanıza yardımcı olacak ve daha sonra da bu paketin yeni sürümleri veya başka paketler için tecrübe kazanmanızı sağlayacaktır.

Bu belgenin yeni sürümlerinin her zaman `http://www.debian.org/doc/maint-«guide/` adresinde ve `maint-guide` paketinden erişilebilir olması gerekir.

## 1.1. Geliştirme için gereken araçlar

Herhangi bir şeye başlamadan önce, geliştirmek için kullanacağınız bazı ek paketlerin düzgün biçimde yüklendiğinden emin olmalısınız. Listenin 'temel' veya 'gerekli' gibi herhangi bir paketi içermediğini dikkate alınız; onların zaten kurulu olduğunu varsayıyoruz.

Bu belgenin bu sürümü Debian 2.2 ('potato') ve 3.0 ('woody') dağıtımlarındaki paketlere göre güncellenmiştir.

Aşağıdaki paketler standart Debian kurulumunda gelmektedir, dolayısiyle muhtemelen sisteminizde bulunmaktadır (bağımlı oldukları diğer ek paketlerle beraber). Gene de **`dpkg -s`** *paketadı* ile kontrol etmelisiniz.

`dpkg-dev`
: Bu paket Debian kaynak paketlerini açmak (unpack), derlemek (build), karşıya yüklemek (upload) için gerekli araçları içerir. (**`dpkg-source(1)`** kılavuz sayfasına bakınız.)

`file`
: Bu kullanışlı araçla bir dosyanın türünü belirleyebilir. (**`file(1)`** kılavuz sayfasına bakınız.)

`gcc`
: GNU C derleyicisi, eğer pek çok diğer program gibi sizinki de C dilinde yazıldıysa gerelidir. (**`gcc(1)`** kılavuz sayfasına bakınız.) Bu paket ayrıca C önişlemcisi **`cpp`** (**`cpp(1)`** kılavuz sayfasına bakınız.) ve nesne dosyalarını çevirmek ve ilintilemek için kullanılan araçları içeren `binutils` (binutils–doc paketi içerisindeki **`info binutils`**'e bakınız) gibi çeşitli paketlere de nüfuz eder.

`g++`
: GNU C++ derleyicisi, eğer programınız C++ ile yazılmışsa gereklidir. (**`g++(1)`** kılavuz sayfasına bakınız.)

`libc6-dev`
: gcc'nin nesne dosyalarını oluştururuken ilintilemesi gereken C kütüphaneleri ve başlık dosyaları. (GNU C Kütüphanesi Başvuru Kılavuzu[B9]'na bakınız.)

`make`

Genellikle bir programın oluşturulması birkaç adım alır, aynı komutları tekrar tekrar yazmak yerine, bu programı kullanarak 'Makefile' dosyaları oluşturup işlemi otomatikleştirebilirsiniz. (**info make**'e bakınız.)

`patch`
: Bu pek faydalı araç, farklılıklar listesi içeren bir dosyayı alır (**diff** uygulaması tarafından oluşturulmuş) ve onu orjinal dosyaya uygular, sonucunda da yamalı sürüm üretilir. (**patch(1)** [B10] kılavuz sayfasına bakınız.)

`perl`
: Perl, genellikle "Unix'in İsviçre Çakısı" olarak ifade edilen, günümüz Unix–benzeri sistemlerde en çok kullanılan betik dillerinden biridir. (**perl(1)** kılavuz sayfasına bakınız.)

Muhtemelen aşağıdaki paketkeri de yüklemek isteyeceksiniz:

`autoconf` **ve** `automake`
: Pek çok yeni program bu tür araçların yardımıyla önişleme tabi tutulan `configure` betikleri ve `Makefile` dosyaları kullanmaktadır. ( ekitabına bakınız.)

`dh-make` **ve** `debhelper`
: **dh-make** örnek paketimizin iskeletini oluşturmak için gereklidir ve paketi oluşturuken de **debhelper** araçlarından bazılarını kullanacaktır. Bunlar paket oluşturmak için gerekli değildir, fakat ilk defa paket hazırlayan bakımcılar için şiddetle tavsiye edilmektedir. Başlangıçta ve sonrasında kontrol ederken bütün işlemi çok basit bir hale getirmektedir. (**dh_make(1)**, **debhelper(1)** kılavuz sayfalarına ve `/usr/share/doc/debhelper/README` dosyasına bakınız.)

`devscripts`
: Bu paket, paket bakımcılarına yardımı olabilecek bazı hoş ve faydalı betikler içermektedir, fakat bunlar paket oluşturmak için gerekli değildir. (`/usr/share/doc/devscripts/README.gz` dosyasına bakınız.)

`fakeroot`
: Bu araç yapım derleme aşamasının bazı yerlerinde root gibi davranmanıza izin vermektedir. (**fakeroot(1)** kılavuz sayfasına bakınız.)

`gnupg`
: Paketleri sayısal olarak imzalamayı sağlayan bir araçtır. Bu eğer özellikle paketinizi başkalarına dağıtmak isterseniz önemlidir ve elbette çalışmanız Debian projesine dahil olunca da bunu yapıyor olacaksınız. (**gpg(1)** kılavuz sayfasına bakınız.)

`g77`
: GNU Fortran 77 derleyicisi, eğer programınız Fortran dilinde yazıldıysa gereklidir. (**g77(1)** kılavuz sayfasına bakınız.)

`gpc`
: GNU Pascal derleyicisi, eğer programınız Pascal dilinde yazıldıysa gereklidir. Bu işte iyi olan ve işaret etmeye değer bir tanesi de **fp-compiler**, Özgür Pascal Derleyicisidir (Free Pascal Compiler). (**gpc(1)** ve **ppc386(1)** kılavuz sayfalarına bakınız.)

`xutils`
: Genelde X11 uygulamalarını yapmak için yapılmış bazı programlar, ayrıca bu programlar bazı makro işlev kümelerinden Makefile dosyalarını üretmek için de kullanılır. (**imake(1)** ve **xmkmf(1)** kılavuz sayfalarına bakınız.)

`lintian`

Bu, siz Debian paketi yaptıktan sonra genel hataları bilmenizi sağlayan ve bulduğu hataları açıklayan Debian paket denetimcisidir. (**lintian(1)** kılavuz sayfasına ve /usr/share/doc/lintian/lintian.html/index.html dosyasına bakınız.)

linda
: Bu da başka bir Debian paket denetimcisidir. (**linda(1)** kılavuz sayfasına bakınız.)

pbuilder
: Bu paket chroot ortamını oluşturmak ve bakımını yapmak için gerekli paketleri içerir. Debian paket derleyici, bu chroot ortamında, gerekli derleme bağımlılıklarının doğrulanmasını ve FTBFS hatalarından kaçınabilmeyi sağlar. (**pbuilder(1)** ve **pdebuild(1)** kılavuz sayfalarına bakınız.)

Aşağıdakiler bu belge ile beraber okumanız gereken *çok önemli* belgelerdir:

debian-policy
: Politika, Debian arşivinin yapısı ve içeriğiyle, bazı işletim sistemi meseleleriyle, Dosya Hiyerarşı Standardı (Filesystem Hierarchy Standard) (hangi dosya ve dizin nerede olmalı) ile ilgili açıklamalar içerir. Sizin için en önemlisi, dağıtımda yer alması için bir pakette olması gerekenleri açıklamasıdır. (http://www.debian.org/doc/debian-«policy/ veya /usr/share/doc/debian-policy/policy.html/index.html dosyasına bakınız.)

developers-reference
: Paketlemenin teknik ayrıntıları dışındaki herşey; arşivin yapısı, nasıl yeniden adlandırılacağı, öksüz (orphan), toplama paketler, NMU'ların nasıl yapıldığı, hatalarla nasıl ilgilenilir, en iyi paketleme pratikleri, ne zaman ve nereye karşıya yükleme yapılmalı, v.b. gibi. (http://www.debian.org/doc/manuals/developers-«reference/index.en.html veya /usr/share/doc/developers-reference/index.en-us.iso-8859-1.html dosyasına bakınız.)

Yukarıda verilen kısa açıklamalar sadece her paketin ne amaçla kullanıldığını belirtmek için vermiştir. Devam etmeden önce lütfen her programın belgelendirmesini etraflıca, en azından standart kullanımını, okuyunuz. Şu an çok zor gözükebilir, ama ilerde okuduğunuza memnun olacaksınız.

**Bilgi**

debmake, dh-make benzeri işlevler yapan araçları içeren bir pakettir, fakat ona özel kullanım bu belge içinde anlatılmayacaktır, çünkü artık önerilmemektedir. Lütfen daha ayrıntılı bilgi edinmek için Debmake Kılavuzuna (http://people.debian.org/~jaldhar/make_package1.html) bakınız.)

## 1.2. Diğer bilgiler

Yapabileceğiniz iki tür paket vardır, ikilik ve kaynak. Kaynak paket, içerisinde derlenince programa dönüşen kodları barındırır. İkilik paketse sadece bitmiş programı. Programın kaynağı ile programın kaynak paketi gibi kavramları karıştırmayın! Lütfen terminoloji hakkında daha ayrıntılı bilgi almak için diğer el kitaplarını okuyunuz

Debian'da, 'bakımcı' ('maintainer') terimi paketi yapan için kullanılır, 'üst düzey geliştirici' ('upstream author') programı yazan için, 'üst düzey bakımcı' ('upstream maintainer') hali hazırda Debian dışında paket hazırlayan için kullanılır. Genellikle üst düzey geliştirici ve de üst düzey bakımcı aynı kişidir – ve bazen de paket bakımcısı bile aynı kişidir. Eğer bir program yaptıysanız ve de Debian içerisine girmek istiyorsanız, bir paket bakımcısı olmak için uygulamanızı gönderme konusunda kendinizi özgür hissedin.

Paketinizi kurguladıktan sonra (veya kurgularken), eğer programınızın bir sonraki dağıtımda yer almasını istiyorsanız resmi Debian paket bakımcısı olmak zorundasınız (eğer program faydalıysa neden olmasın?). Bu süreç Geliştirici Başvuru Kılavuzunda (Developer's Reference) anlatılmıştır. Lütfen okuyun.

# 2. İlk adımlar

## 2.1. Oluşturacağınız paketi seçin

Muhtemelen oluşturmak istediğiniz paketi seçmişsinizdir. Yapmanız gereken ilk şey paketin hali hazırdaki dağıtımda olup olmadığına bakmaktır. Eğer 'kararlı' (stable) dağıtımı kullanıyorsanız, belki de en iyisi paket arama sayfasına (http://www.debian.org/distrib/packages) gitmenizdir. Eğer şu anki 'kararsız' (unstable) dağıtımı kullanıyorsanız, şu komutları kullanın:

```
dpkg -s program
dpkg -l '*program*'
```

Eğer paket halihazırda mevcutsa, o zaman onu yükleyin! :-) Eğer öksüz durumuna (orphaned) düştüyse – eğer paket bakımcısı "Debian QA Group"'a geçmişse – onu oradan almalısınız. Öksüz paketler listesine[B30] ve sahiplenilmeyi (evlat edinilme) bekleyen paketler listesine[B31] paketin gerçekten de sahiplenilmek için boşta olup olmadığına bakın.

Eğer bir paketi sahiplenebilecekseniz, kaynak kodlarını edinin (**apt-get source** *paketadı* gibi bir şeyle) ve inceleyin. Bu belge ne yazık ki paketlerin sahiplenilmesiyle ilgili ayrıntılı bilgi içermemektedir. Ne mutlu ki, paketin nasıl çalıştığı ile ilgili zor zamanlar geçirmenize gerek kalmayacaktır, çünkü birileri başlangıç ayarlarını sizin için yapmıştır. Okumaya devam edin, yine de, aşağıdaki pek çok tavsiye sizin durumunuz için uygulanabilir olacaktır.

Eğer paket yeniyse ve siz de onu Debian içerisinde görmeye karar verdiyseniz, aşağıdaki şekilde ilerleyin:

- Üzerinde çalışılan paketler listesini[B32] paket üzerinde herhangi biri çalışıyor mu diye kontrol edin. Eğer halihazırda birileri çalışıyorsa, ihtiyaç duyarsanız onlarla iletişime geçin. Ya da henüz bir paket bakımcısı atanmamış başka bir paket bulun.
- Programların bir lisansı olmalıdır ve mümkünse lisans Debian Özgür Yazılım Anahatları[B33]nda belirtildiği gibi özgür (free) olsun. Eğer bu kurallardan herhangi birini sağlamazsa, fakat herhangi bir şekilde dağıtılabiliyorsa, 'contrib' veya 'non–free' bölümlerine dahil edilebilir. Eğer nereye ait olduğundan emin değilseniz, lisans metnini <debian–legal (at) lists.debian.org> adresine postalayarak yardım isteyiniz.
- Programlar kesinlikle setuid root çalışmamalıdır, hatta daha iyisi setuid veya setgid'e herhangi bir şey için ihtiyaç duymamalıdır.
- Program bir artalan süreci (deamon) veya */sbin dizinleri altına giden bir program ya da root olarak bir port açan bir program olmamalıdır.
- Program ikilik çalışabilir halde olmalıdır, kütüphaneler idare etmek için daha zordur.
- İyi şekilde belgelendirilmiş ve/veya kodu anlaşılabilir olmalıdır. (örn. kafa karıştırıcı olmamalı)
- Programın yazar/yazarlarıyle iletişim kurarak paketlenmesi konusunda bir anlaşma sağlamalısınız. Herhangi bir, programa özgü hata durumunda, yazar/yazarlar ile görüş alışverişinde bulunmak önemlidir, dolayısıyla geliştirilmesi durmuş yazılım parçalarını paketlemeye çalışmayın.
- Ve en sonuncu fakat en önemsiz olmayanı, uygulamanın çalıştığını bilmeli, bir kaç kere denemelisiniz.

Elbette ki, bunlar sadece güvenlik ölçekleridir ve sizleri setuid hayalet programında bir hata yapmanız durumunda öfkeli kullanıcılardan korumayı amaçlamaktadır... Paketleme konusunda biraz daha tecrübe kazandıktan sonra, bu tür paketler yapabileceksiniz, fakat tecrübeli geliştiriciler bile şüpheye düştükleri zaman debian akıl hocaları listesine (debian–mentors mailing list) başvurmaktadır. Oradaki insanlar da memnuniyetle yardım edeceklerdir.

Bunlar hakkına daha ayrıntılı bilgi için, Geliştiricilerin Başvuru Kılavuzuna (Developer's Reference) bakınız.

## 2.2. Programı edinin ve deneyin

Yapılacak ilk iş, paketi bulup indirmektir. Yazarının sayfasından kaynak dosyasını aldığınızı varsayıyorum. Özgür Unix programları için kaynaklar genellikle .tar.gz uzantısı ile tar/gzip formatında gelir ve içerisinde program–sürüm isimli kaynak kodları barındıran bir altdizin içerir. Eğer programınızın kaynağı buna benzer biçimlerden biriyle geldiyse (örn. dosya adı ".Z" veya ".zip" ile bitiyorsa), uygun araçlarla onu açın veya emin değilseniz nasıl açacağınızı debian akıl hocaları (debian–mentors) listesine sorun (ipucu: 'file arşiv.uzantı').

Örnek olarak, bir X GTK+ dosya yöneticisi olan 'gentoo' isimli bir program kullanacağım. Bu paketin zaten paketlenmiş olduğunu ve bu yazı yazılırken oldukça değişikliğe uğradığını dikkate alınız.

Ev dizininiz altında 'debian' veya 'deb' veya uygun bulduğunu herhangi bir isimde (örn. sadece `~/gentoo/` bu durum için yeterli olacaktır) bir dizin oluşturunuz. İndirmiş olduğunuz arşivi oraya yerleştirip açınız ('`tar xzf gentoo-0.9.12.tar.gz`' ile). Hiç hata olmadığından emin olun, hatta "ilgisiz" olan hatalar bile, çünkü başkalarının sistemlerinde bu tür açma araçlarının bu hataları gözardı etmesi ya da etmemesinden dolayı kullanımlarında sorunlar olabilmektedir.

Artık 'gentoo–0.9.12' isimli yeni bir dizininiz var. Bu dizine geçip, ayrıntılı bir şekilde belgelendirmesini okuyunuz. Genellikle `README*, INSTALL*, *.lsm, *.html` isimli dosyalar mevcuttur. Düzgün bir şekilde programı derleyip yüklemek için gerekli talimatları bulmalısınız (büyük ihtimalle `/usr/local/bin` dizinine yükleyeceğinizi varsayacaklardır; bunu yapmayacak, ileride göreceğimiz gibi *Kütüphaneleri değiştirmek* (sayfa: 11) işlemini gerçekleştireceksiniz).

Süreç programdan programa değişir, fakat modern programların çoğu kaynak paketini sisteminize göre yapılandırıp, sisteminizde derlemeye uygun hale getiren bir 'yapılandırma' (`configure`) betiği ile gelir. `./configure` ile yapılandırıldıktan sonra genelde paket içeriği **`make`** ile derlenir. Bazıları dahili kontrol yapılmasını sağlayan **`make check`**'i destekler. Hedef dizine yükleme genelde **`make install`** ile olur.

Şimdi programınızı derleyip çalıştırmaya çalışın. Düzgün çalıştığıntan ve derleme veya çalışma esnasında hatalar oluşmadığından emin olun.

Ayrıca, derleme yaptığınız dizini temizlemek için genellikle **`make clean`** (daha iyisi **`make distclean`**) kullanabilir, hatta bundan önce, bazan **`make uninstall`** ile sistemden kurulu olan dosyaları kaldırabilirsiniz.

## 2.3. Paket adı ve sürümü

Paketlemeye tamamiyle temizlenmiş (ilk hali bozulmamış) ya da arşiv dosyasından yeni çıkarılmış bir kaynak diziniyle başlamalısınız.

Paketin düzgün şekilde derlenebilmesi için orjinal adını küçük harflere çevirmelisiniz (eğer halihazırda değilse) ve de kaynak dizininin ismini *paketadı–sürüm* şeklinde değiştirmelisiniz.

Eğer program ismi birden fazla sözcük içeriyorsa, onları tek kelime haline gelecek şekilde küçültün veya bir kısaltma haline getirin. Örneğin, "John'un X için küçük metin düzenleyicisi" gibi bir isim, johnedx veya jke4x şeklinde ya da, belli bir karakter sayısını, mesela 20, aşmayacak şekilde (takdirinize kalmış) isimlendirilebilir.

Ayrıca programın asıl sürümüne bakın (paket ismine konulacak olan sürüm bilgisi). Eğer bu yazılım parçası X.Y.Z gibi sürüm numaralarıyla numalarandırılmak yerine bir tarih bilgisi yazılmışsa, bu tarih bilgisinin önüne

"0.0." (üst geliştiricilerin bir gün 1.0 ve tarih içeren bir günlük sürüm çıkarabileceklerini göz önüne alarak "0.0.") koyarak sürüm numarası olarak kullanmaktan çekinmeyin. Dolayısiyle, yayınlanma tarihi 19 Aralık 1998 ise, 0.0.19981219 dizgesini sürüm numarası olarak kullanabilirsiniz.

Bazı programlar hiç numaralandırılmazlar, böyle durumlarda üst düzey geliştiricilerle iletişime geçip nasıl bir sürüm izleme yöntemi kullandıklarını öğrenmeniz gerekir.

## 2.4. İlk "debianlaştırma"

Programın kaynak dizininde olduğundan emin olun ve şunu komutu girin:

```
dh_make -e kullanıcı@adres -f ../gentoo-0.9.12.tar.gz
```

Elbette ki, *kullanıcı@adres* kısmını `ChangeLog` ve diğer dosyalarda yer alması için kendi e–posta adresinizle, dosya adını da kendi orjinal kaynak arşivinizle değiştireceksiniz. (Ayrıntılar için dh_make(1) kılavuz sayfasına bakınız.)

Bazı bilgiler su yüzüne çıkacaktır. Size ne tür bir paket oluşturmak istediğinizi soracaktır. Gentoo tek bir ikilik pakettir – sadece bir ikilik, yani bir `.deb` dosyası oluşacaktır – dolayısiyle ilk seçeneği 's' anahtarı ile seçeceğiz, ekrandaki bilginin doğruluğunu kontrol edip <enter>'a basınız.

**`dh_make`**'in bu çalışmasından sonra, aslen Debian olmayan kaynak paketin oluşturulmasına yardımcı olacak `diff.gz`'li bir dosya ile üst düzey geliştiricinin sıkıştırılmış arşiv dosyasının (tarball) bir kopyası `gentoo_0.9.12.orig.tar.gz` adıyla ana dizinde oluşturulur. Lütfen dosya adındaki 2 önemli noktaya dikkat ediniz:

- Paket adı, sürüm numarasından "_" ile ayrılmıştır.
- `tar.gz`'den önce bir `orig.` vardır.

Bir kez daha hatırlatalım, yeni bir paket bakımcısı olarak karmaşık paketlerin yapımından vazgeçmeniz istenir, örn.,

- çoklu ikilik paketler
- kütüphane paketleri
- biçimi ne `tar.gz` ne de `tar.bz2` olan kaynak dosyalar
- dağıtılamayan içeriğe sahip sıkıştırılmış kaynak dosyalar

Bunlar da çok zor değildir, fakat biraz daha bilgiye ihtiyaç gösterirler ve biz de burada herşeyi anlatmak istemiyoruz.

**Uyarı**

**`dh_make`**'i sadece bir kere çalıştırmanız gerektiğine dikkat edin, çünkü halihazırda "debianlaşmış" dizinde tekrar çalışınca düzgün davranmayacaktır. Bu da aynı zamanda programınızın yeni düzeltim veya sürümlerini hazırlarken farklı bir yöntem kullanmanız gerektiğinin de işaretçisidir. Bu konu ileride *Paketin Güncellenmesi* (sayfa: 30) bölümünde anlatılacaktır.

# 3. Kaynak paketinde değişiklik

Normal olarak programlar kendilerini `/usr/local` dizini altındaki dizinlere yükler. Fakat, bu kısım sistem yöneticilerinin (veya kullanıcıların) özel kullanımlarına ayrıldığı için, Debian paketleri bu dizini kullanmamalıdır. Bu da paketin derlenme sistemine, genel olarak da Makefile ile başlayarak bakmak demektir. Makefile, programın

derlenmesini ve kurulmasını otomatikleştirecek bir **make(1)** betiğidir. Makefile ile ilgili daha ayrıntılı bilgi için *rules dosyası* (sayfa: 17) bölümüne bakınız.

Eğer programınız GNU **automake(1)** ve/veya **autoconf(1)** [B40] kullanıyorsa, ki bu da kaynak paketinin `Makefile.am` ve/veya `Makefile.in` dosyalarını içermesi demektir, sırasıyla, bu dosyaları değiştirmeniz gerekir. Bunun sebebi, her bir **automake** çağrımının `Makefile.am` dosyasında üretilen bilgilerin `Makefile.in` dosyasına tekrar yazılmasını, her bir **./configure** çağrımının da `Makefile.in` dosyalarında üretilen bilgi ile benzer işlemi `Makefile` dosyasıda gerçekleştirecek olmasıdır. `Makefile.am` dosyasını düzenlemek biraz yöntemleri vardır; çalıştırılabilirler $PATH'e eklenir, ayrıca belgeleri ve kılavuz sayfalarını da yaygın kullanılan yerlerde bulursunuz. Yine de, bu şekilde yaptığınız zaman, programınız sisteminizde var olan herşeyin arasına yüklenecektir. Bu da paketleme araçlarını zora sokacak ve hangi dosyalar pakete ait hangileri değil tespiti zorlaşacaktır.

Bunun için farklı bir şeyler yapmalısınız: programı, paketleme araçlarının çalışan bir `.deb` paketi yapabileceği bir geçici alt dizine kurun. Bir kullanıcının bu paketi kurmak istemesi durumunda, bu dizinin içerdiği herşey sistemine kurulacaktır, tek farkla ki, **dpkg** dosyaları geçici alt dizine değil kök dizin altına kuruyor olacaktır.

Bu geçici dizin kaynak dizin hiyerarşisi içinde, `debian/` dizini olarak oluşturulur. Genelde `debian/tmp` veya `debian/paketadı` olarak isimlendirilir.

Program kurulumunu `debian/paketadı` içine yüklenmiş şekilde yapmanız gerekse de, kök dizin altına yerleştirilince de düzgün çalışması gerekmektedir, örn. bir `.deb` paketinden kurulum yaparken. Bu bakımdan paket dosyalarına derleme sisteminin `/home/me/deb/gentoo-0.9.12/usr/share/gentoo` gibi dizgeyi uygulamasına izin vermemelisiniz.

GNU autoconf'u kullanan programlarla bu biraz daha kolay olacaktır. Böyle programların çoğu, Örneğin `/usr` gibi belli bir öneki uhdesinde tutarken, kurulumun rasgele bir alt dizine yapılabilmesine imkan veren öntanımlı bir yapılandırma sistemine sahip Makefile dosyaları içeririrler. **dh_make**, programınızın autoconf kullandığını saptarsa, komutları bunu otomatik olarak yapması için ayarlayacaktır. Böylece, bu bölümün devamını okumadan geçebileceksiniz. Fakat diğer program türleri için Makefile dosyalarını incelemeniz ve gereken düzenlemeleri kendiniz yapmanız gerekecektir.

İşte gentoo'nun Makefile dosyasının ilgili kısmı:

```
# Where to put binary on 'make install'?
# 'make install' çalıştırılabilirleri nereye koyacak?
BIN     = /usr/local/bin

# Where to put icons on 'make install'?
# 'make install' simgeleri nereye koyacak?
ICONS   = /usr/local/share/gentoo
```

Dosyaların `/usr/local` altına yükleneceğini görüyoruz. Bu yolları şu şekilde değiştirin:

```
# Where to put binary on 'make install'?
BIN     = $(DESTDIR)/usr/bin

# Where to put icons on 'make install'?
ICONS   = $(DESTDIR)/usr/share/gentoo
```

Fakat neden bu dizin de bir başkası değil? Çünkü Debian paketleri dosyalarını asla `/usr/local` altına yüklemez – bu hiyerarşi sistem yöneticileri için ayrılmıştır. Bunun yerine, bu tür dosyalar Debian sistemlerinde `/usr` altına gider.

İkilikler, simgeler, belgelendirme, v.b. ile ilgili daha kesin yerleştirme bilgisi Dosyasistemi Hiyerarşisi Standardında (Filesystem Hierarchy Standard) belirtilmiştir. Size `/usr/share/doc/debian-policy/fhs/` dizinindekileri okumanızı ve paketinizle ilgili kısımları incelemenizi tavsiye ederim.

O halde, ikilik dosyayı `/usr/local/bin` yerine `/usr/bin`'e, kılavuz sayfalarını `/usr/local/man/man1` yerine `/usr/share/man/man1`'e, v.s. yüklemeliyiz. gentoo'nun makefile'ında bir kılavuz sayfası belirtilmediğini hatırlayın, fakat Debian Politika Kılavuzu[B41] her programın bir kılavuz sayfası olmasını gerektirdiğinden, ilerde bir tane yapıp `/usr/share/man/man1` altına koyacağız.

Bazı programlar bu tür yolları belirtmek için makefile değişkenlerini kullanmaz. Bu da doğru yerleşimleri kullanmaları için asıl C kodunu düzenlemek zorunda kalacağınız anlamına gelir. Fakat nereyi araştıracaksınız, tam olarak ne için? Bunu aşağıdakini uygulayarak öğrenebilirsiniz:

```
grep -rn usr/local/lib *.[ch]
```

Grep kaynak ağacında özyinelemeli olarak çalışacak ve arananla ilgili bir oluşum olduğunda size dosya adını ve satırı söyleyecektir.

Şu dosyaları düzenleyiniz ve `/usr/local/*` satırlarını `usr/*` ile değiştirin. Kodun geri kalan kısmına zarar vermediğinizden emin olun. `:-)`

Bundan sonra hedef yükleme yerini bulmalısınız (`install:` ile başlayan satırı araştırın, genelde işe yarar) ve tüm başvuruları yukarında Makefile dosyasında olduğu gibi yeniden adlandırın. Önceden, gentoo'nun kurulum hedefi şu şekilde idi:

```
install: gentoo
        install ./gentoo $(BIN)
        install icons/* $(ICONS)
        install gentoorc-example $(HOME)/.gentoorc
```

Değiştirdikten sonra şu şekilde olur:

```
install: gentoo-target
        install -d $(BIN) $(ICONS) $(DESTDIR)/etc
        install ./gentoo $(BIN)
        install -m644 icons/* $(ICONS)
        install -m644 gentoorc-example $(DESTDIR)/etc/gentoorc
```

Elbetteki farketmişsinizdir, artık kural setindeki diğerlerinden önce bir **`install -d`** komutu bulunmaktadır. Orjinal makefile bunu içermemektedir, çünkü genelde **`make install`**'ın çalıştırdığı bir sistemde `/usr/local/bin` ve diğer dizinler bulunmaktadır. Yine de, kendi boş (varolmayan) dizinimize kurduğumuz için, bu dizinlerin herbirini oluşturmamız gerekecektir.

Kural kümesinin sonuna bazı başka şeyler de ekleyebiliriz, üst düzey geliştiricinin bazen atlatığı ek belgelendirmenin kurulumu gibi.

```
        install -d $(DESTDIR)/usr/share/doc/gentoo/html
        cp -a docs/* $(DESTDIR)/usr/share/doc/gentoo/html
```

Dikkatli bir okuyucu **`install:`** satırındaki `gentoo`'yu `gentoo-target`'e çevirdiğimi farketmiştir. Buna ilişkisiz hata onarımı (unrelated bug fix) denir `:-)`.

Her ne zaman Debian paketiyle ilgili bir değişiklik yaparsanız, üst düzey geliştiriciye de de bunu yollamayı unutmayın ki böylece programın yeni sürümünde yer alarak herkese faydalı olabilsin. Ayrıca, göndermeden önce yapacağınız düzeltmeleri sadece Debian ya da Linux'a (ve hatta Unix'e!) özgün olmaması gerektiğini unutmayın – onları taşınabilir yapın. Bu da düzeltmelerinizin uygulanması kolaylaştırır.

`debian/*` dosyalarını göndermeniz gerekmemektedir.

## 3.1. Kütüphaneleri değiştirmek

Bir genel sorun daha vardır: kütüphaneler genelde platformdan platforma değişmektedir. Örneğin, bir Makefile Debian sisteminde olmayan bir kütüphaneye başvuruyor olabilir. Bu durumda, onun Debian sisteminde var olan bir kütüphaneye, aynı işi yapacak şekilde başvurmasını sağlamalıyız.

Dolayısiyle, Makefile (veya Makefile.in) dosyasınızda şöyle bir satır varsa (ve programınız derlenmiyorsa):

```
        LIBS = -lcurses -lbirsey -lbaskabirsey
```

Şu şekilde değiştirin ve muhtemelen çalışacaktır:

```
        LIBS = -lncurses -lbirsey -lbaskabirsey
```

(Yazar, şu an `libcurses.so` sembolik bağı ile gelen `libncurses` paketiyle ilgili olarak bunun en iyi örnek olmadığını farkındadır, fakat daha iyisini düşünememektedir. Önerilere her zaman açığız `:-)`.)

# 4. `debian/` altında gerekli olanlar

Programın kaynak dizininin altında yeni bir dizin vardır, `debian` dizini. Paketin davranışını özelleştirmek için düzenleyebileceğimiz bazı dosyalar bu dizinde bulunmaktadır. En önemlileri, tüm paketler için gerekli olan, `control`, `changelog`, `copyright` ve `rules`'dur.

## 4.1. `control` dosyası

Bu dosya, paket yönetimi için kullanılacak olan **`dpkg`**, **`dselect`** ve diğer paket yönetim araçlarına gereken değerleri barındırır.

**`dh_make`**'in bizim için oluşturduğu `control` dosyası:

```
1 Source: gentoo
2 Section: unknown
3 Priority: optional
4 Maintainer: Josip Rodin <joy-mg@debian.org>
5 Build-Depends: debhelper (>> 3.0.0)
6 Standards-Version: 3.6.2
7
8 Package: gentoo
9 Architecture: any
10 Depends: ${shlibs:Depends}
11 Description: <insert up to 60 chars description>
12   <insert long description, indented with spaces>
```

1–6 arası satırlar kaynak paketle ilgili kontrol bilgileridir.

1. satır: kaynak paketin adı,

2. satır: kaynak paketin dağıtımda yer aldığı bölümdür.

Farkettiğiniz gibi, Debian bölümlere ayrılmıştır: main (özgür yazılım), non–free (özgür olmayan yazılım) ve contrib (özgür olmayan yazılıma bağımlı özgür yazılım). Bunların altında, kısaca paket türünü açıklayan mantıksal alt kısımlar vardır. Sadece–sistem yöneticisi için 'admin', temel araçlar için 'base', programlama araçları için 'devel', belgelendirme için 'doc', kütüphaneler için 'libs', elektronik posta istemci ve arka planda çalışan art alan programları için 'mail', ağ uygulamaları ve artalan süreçleri için 'net', herhangi bir yere uymayan X11 programları için 'x11' vs. alt bölümleri bulunmaktadır.

Öyleyse onu x11 olarak değiştirelim. ("main/" öneki ima edilmiştir, dolayısiyle onu atlayabiliriz.)

3. satır: kullanıcının bu paketi yüklemesinin ne kadar önemli olduğunu belirtmektedir. Bu alanı nasıl ayarlayacağınıza karar vermek için Debian Politika Kılavuzu[B42]na bakınız. "optional" önceliği genellikle yeni paketler için kullanılmaktadır.

Section ve priority, paketleri sıralayacak veya öntanımlıları seçecekleri zaman, **dselect** diğer kullanıcı araçları tarafından kullanılır. Bir kere paketi Debian'a yükledikten (upload) sonra, arşiv sürdürücüleri tarafından bu iki alanın değeri değiştirilebilir, ki bu da size e–posta ile bildirilir.

Bu, normal öncelikli bir paket olduğu ve herhangi bir şeyle çakışmadığı için, "optional" olarak bırakacağız.

4. satır: paket bakımcısının adı ve e–posta adresidir. Bu kısmı bir epostadaki geçerli bir "To: " başlığı olarak düzenlemeniz gerekir, çünkü paketi Debian'a gönderdikten sonra, hata izleme sistemi hataları göndermek için bu e–posta adresini kullanacaktır. Virgül, & ve parantez kullanımından kaçının.

5. satır: paketinizin derlenebilmesi (build) için gerekli paketleri listelemektedir. gcc ve make gibi bazı paketler göz ardı edilmiştir; ayrıntılar için `build-essential` paketini inceleyiniz. Eğer paketinizi yapmak için standartdışı bir derleyici veya başka araçlar gerekiyorsa, bunu 'Build–Depends' satırına eklemelisiniz. Birden çok girdi varsa, virgüllerle ayrılır; bu alanın sözdizimi hakkında bilgi edinmek için ikilik paketlerin bağımlıklıklar ile ilgili açıklamalarını okuyunuz.

Bu kısımda ayrıca Build–Depends–Indep (bağımlılıklar), Build–Conflicts (çakışmalar) ve diğer alanlar olabilir. Bu veri Debian otomatik paket yapım sistemi tarafından diğer bilgisayar platformları için ikilik paket yapımında kullanılacaktır. Derleme bağımlılıkları (build–dependency) ile ilgili bilgi edinmek için Debian Politika Kılavuzu[B43]na, diğer platformlar (mimariler) ve bunlara yazılımların uyarlanması ile ilgili bilgi edinmek için Geliştiricilerin Başvuru Kılavuzuna (Developers' Reference) bakınız.

Paketinizin derlenebilmesi için hangi paketlere ihtiyaç duyduğunu bulmakta kullanabileceğiniz küçük bir kod:

```
strace -f -o /tmp/log ./configure
# Paket autoconf kullanmıyorsa, ./configure yerine make kullanılsın
for x in `dpkg -S $(grep open /tmp/log|\
        perl -pe 's!.* open\(\"([^\"]*).*!$1!' |\
        grep "^/"| sort | uniq|\
        grep -v "^\(/tmp\|/dev\|/proc\)" ) 2>/dev/null|\
        cut -f1 -d":"| sort | uniq`; \
      do \
        echo -n "$x (>=" `dpkg -s $x|grep ^Version|cut -f2 -d":"` "), "; \
      done
```

`/usr/bin/foo` için derleme bağımlılıklarını el yordamıyla bulmak isterseniz,

```
objdump -p /usr/bin/foo | grep NEEDED
```

komutunu, listelenmiş her kütüphane için (örn, `libfoo.so.6`),

```
dpkg -S libfoo.so.6
```

komutunu kullanabilirsiniz. Daha sonra da her paketin `-dev` sürümünü `Build-deps` girdisi olarak alırsınız. Eğer bu amaç için **ldd**'yi kullanırsanız, dolaylı kütüphane bağımlılıklarını da rapor edeceğinden, bu gereğinden fazla derleme bağımlılığına sebep olacaktır.

Gentoo ayrıca derleme aşamasında `xlibs-dev, libgtk1.2-dev` ve `libglib1.2-dev`'e ihtiyaç duymaktadır, `debhelper`'ın yanına onları da buraya ekleyeceğiz.

6. satır: paketin uyduğu Debian Politika standartlarının sürümüdür; paketinizi yaparken okuduğunuz Debian Politika Kılavuzu[B44]nun sürüm numarasıdır.

8. satır: ikilik paketin adıdır. Bu genellikle kaynak paketin adıyla aynıdır, ama hep böyle olması gerekmemektedir.

9. Satır: ikilik paketin derlenebileceği işlemci mimarisini açıklamaktadır. Bu kısmı "any" olarak bırakacağız, çünkü **dpkg-gencontrol(1)** bu kısmı paketin delenebileceği mimariye uygun değerlerle dolduracaktır.

Eğer paketiniz mimariden bağımsızsa (mesela, bir kabuk veya bir Perl betiği ya da bir belge), bunu "all" olarak değiştirin ve daha sonra da `binary-arch` yerine `binary-indep` kurallarının kullanımı ile ilgili olarak *rules dosyası* (sayfa: 17) bölümünü okuyunuz.

10. satır, Debian paketleme sisteminin en güçlü özelliklerinden birini göstermektedir. Paketler birbirleriyle değişik şekilde ilişkide olabilir. `Depends:`'den başka, `Recommends:`, `Suggests:`, `Pre-Depends:`, `Conflicts:`, `Provides:` ve `Replaces:` ilişki alanları belirtilebilir.

Paket yönetim araçları bu ilişkilerle muhatap olurken genelde aynı şekilde davranır; bunun aksine durumlar ileride açıklanacaktır. (**dpkg(8)**, **dselect(8)**, **apt(8)**, **aptitude(1)** v.b. kılavuz sayfalarına bakınız.)

Paket içeriğinin çalışabilmesi için (derlenebilmesi için değil) bağımlı olduğu paketlerle ilişki alanları:

**Depends:**
Paket, bağımlılığı olan paketler kurulmadan kurulmayacaktır. Bunu eğer paketiniz belli bir paket olmadan kesinlikle çalışmayacaksa (ya da ciddi bozukluklara sebep olacaksa) kullanın.

**Recommends:**
**dselect** ve **aptitude** gibi araçlar), paketinizle beraber bu alanda tavsiye edilen paketlerin de kurulmasını isteyip istemediğinizi soracaklardır; hatta **dselect** ısrar edecektir. **dpkg** ve **apt-get** bu alanı ihmal edecektir. Bunu kesinlikle zorunlu olmayan fakat genellikle program ile beraber kullanılan paketler için kullanın.

**Suggests:**
Bir kullanıcı programınızı kurmak istediğinde, tüm kurulum araçları bu alanda belirtilen paketlerin de kurulup kurulmayacağını soracaktır. **dpkg** ve **apt-get** bu alan önemsemezler. Bu alanı, paketinizle çalışması hoş olacak fakat çalışması için mutlaka gerekli olmayan paketler için kullanın.

**Pre–Depends:**
Bu, `Depends:`'den daha güçlüdür. Önbağımlılıklar kurulmadan ve düzgün şekilde ayarlanmadan peketinizin kurulumu yapılmayacaktır. Bunu oldukça tutarlı bir şekilde ve debian–devel eposta listesinde tartıştıktan sonra kullanın. (Bunu, hiç kullanmayın şeklinde okuyun. `:-)`)

**Conflicts:**
Paket, çeliştiği tüm paketler kaldırılmadan kurulmayacaktır. Bunu, eğer paketiniz belli bir paketin varlığında çalışmayacak veya ciddi sorunlara yol açacaksa kullanın.

**Provides:**
Birden fazla alternatifi olan paletler için sanal isimler tanımlanmıştır. Tüm listeyi `/usr/share/doc/debian-policy/virtual-package-names-list.txt.gz` dosyasından alabilirsiniz. Bunu, eğer programınız mevcut bir sanal paketin bir işlevini yerine getiriyorsa kullanın.

**Replaces:**
Programınız başka bir paketin dosyaların değiştiriyorsa ya da başka bir paketin tamamen yerini alıyorsa, bunu kullanın (`Conflicts:` ile birlikte kullanılır). Paketinizdeki dosyalar, ismi belirtilen paketlerin dosyalarının üzerine yazılacaktır.

Tüm bu alanların kendilerine has bir sözdizimi vardır. Paket adları virgüllerle ayrılır. Bu paket adları ayrıca '|' (boru sembolü) ile ayrılmış alternatif paket adları şeklinde belirtilebilir.

Alanlar kendi uygulanabilirliklerini belirttikleri paketlerin belirli sürümleriyle sınırlayabilir. Bu sürümler her bir paket adından sonra parentez içinde belirtilir, ayrıca sürüm numarasının önüne şu ilişki gösterimlerinden biri

eklenmelidir: <<, <=, =, >= ve >> (Sırayla: kesinlikle önceki, önceki ya da aynısı, kesinlikle aynısı, sonraki ya da aynısı, kesinlikle sonraki). Örnek:

```
Depends: foo (>= 1.2), libbar1 (= 1.3.4)
Conflicts: baz
Recommends: libbaz4 (>> 4.0.7)
Suggests: quux
Replaces: quux (<< 5), quux-foo (<= 7.6)
```

Bilmeniz gereken en son özellik `${shlibs:Depends}`'dir. Paketiniz derlenip, geçici dizine kurulduktan sonra, **dh_shlibdeps(1)** ikilikler ve kütüphaneler için orayı tarayacak ve `libc6` veya `xlib6g` gibi paylaşımlı kütüphane bağımlılıklarını ve bunları içeren paketleri belirleyecektir. Bunları **dh_gencontrol(1)**'deki listeye aktaracak ve doğru yere yerleştirilecek, bunun hakkında kaygılanmanıza gerek kalmayacaktır.

Tüm bunları anlattıktan sonra `Depends:` satırını şu an olduğu gibi bırakabilir ve ardına `Suggests: file` satırını ekleyebiliriz, Çünkü gentoo o program/paket tarafından sunulan bazı özellikleri kullanabilmektedir.

11. satır: kısa açıklamadır. Pekçok kimsenin ekranı 80 sütun genişliğindedir, dolayısıyle bu kısım yaklaşık 60 karakteri geçmemelidir. Bu kısmı "fully GUI configurable X file manager using GTK+" olarak değiştireceğim.

12. satır: uzun açıklamanın yer aldığı kısımdır. Bu paket hakkında ayrıntılı bilgi veren bir paragraf olmalıdır. Her satırın ilk sütunu boş olmalıdır. Hiç boş satır olmamalıdır, fakat bir sütuna tek bir . (nokta) koyarak boş bir satır bırakılmasını sağlayabilirsiniz. Ek olarak, uzun açıklamadan sonra birden fazla boş satır olmamalıdır.

Sonuçta, güncellenmiş `control` dosyası:

```
1 Source: gentoo
2 Section: x11
3 Priority: optional
4 Maintainer: Josip Rodin <joy-mg@debian.org>
5 Build-Depends: debhelper (>> 3.0.0), xlibs-dev, libgtk1.2-dev, libglib1.2-dev
6 Standards-Version: 3.5.2
7
8 Package: gentoo
9 Architecture: any
10 Depends: ${shlibs:Depends}
11 Suggests: file
12 Description: fully GUI configurable X file manager using GTK+
13  gentoo is a file manager for Linux written from scratch in pure C. It
14  uses the GTK+ toolkit for all of its interface needs. gentoo provides
15  100% GUI configurability; no need to edit config files by hand and re-
16  start the program. gentoo supports identifying the type of various
17  files (using extension, regular expressions, or the 'file' command),
18  and can display files of different types with different colors and icons.
19  .
20  gentoo borrows some of its look and feel from the classic Amiga file
21  manager "Directory OPUS" (written by Jonathan Potter).
```

## 4.2. `copyright` dosyası

Bu dosya paketin üst düzey özkaynakları, telif hakkı ve lisansı hakkında bilgiler içerir. Biçimi Debian Politika Kılavuzu[B53] tarafından bir kurala bağlanmamıştır, fakat içeriği belirtilmiştir (bölüm 13.6 "Copyright information").

**dh_make** şuna benzer öntanımlı bir dosya oluşturur:

```
1 This package was debianized by Josip Rodin <joy-mg@debian.org> on
2 Wed, 11 Nov 1998 21:02:14 +0100.
```

```
3
4 It was downloaded from <fill in ftp site>
5
6 Upstream Author(s): <put author(s) name and email here>
7
8 Copyright:
9
10 <Must follow here>
```

Bu dosyaya eklemeniz gereken önemli şeyler, paketi aldığınız yer, gerçek telif hakkı uyarısı ve lisanstır. Tüm lisansı ilave etmelisiniz, eğer GNU GPL veya LGPL, BSD veya Artistik lisans gibi herhangi bir genel özgür yazılım lisanlarından biri değilse, çünkü Debian sisteminde var olan `/usr/share/common-licenses/` dizinindeki uygun bir dosyaya atıfta bulunmak mümkündür.

Kısacası, gentoo'nun `copyright` dosyasının son hali:

```
1 This package was debianized by Josip Rodin <joy-mg@debian.org> on
2 Wed, 11 Nov 1998 21:02:14 +0100.
3
4 It was downloaded from: ftp://ftp.obsession.se/gentoo/
5
6 Upstream author: Emil Brink <emil@obsession.se>
7
8 This software is copyright (c) 1998-99 by Emil Brink, Obsession
9 Development.
10
11 You are free to distribute this software under the terms of
12 the GNU General Public License.
13 On Debian systems, the complete text of the GNU General Public
14 License can be found in the file `/usr/share/common-licenses/GPL'.
```

## 4.3. `changelog` dosyası

Bu, Debian Politika Kılavuzu[B54]nun 4.4 "debian/changelog" bölümünde tanımlanan özel bir biçime sahip gerekli bir dosyadır. Bu biçim **dpkg** ve diğer araçlar tarafından sürüm, gözden geçirme, dağıtım ve paketinizin aciliyet bilgisini alırken kullanılır.

Yapmış olduğunuz tüm değişiklikleri belgelendirdiğiniz için, sizin için de önemlidir. Bu, paketinizi indiren insanların bilmeleri gereken bir sorun olup olmadığını görmelerini sağlar. Dosya, ikilik paket içerisinde `/usr/share/doc/gentoo/changelog.Debian.gz` olarak kaydedilecektir.

**dh_make** şuna benzer öntanımlı bir dosya oluşturur:

```
1 gentoo (0.9.12-1) unstable; urgency=low
2
3  * Initial Release.
4
5 -- Josip Rodin <joy-mg@debian.org>  Wed, 11 Nov 1998 21:02:14 +0100
6
```

1. satır: paket adı, sürüm, dağıtım ve aciliyettir. İsim kaynak paket adıyla uyuşmalıdır, dağıtım ya `unstable` (veya hatta `experimental`) olmalıdır ve aciliyet de (`urgency`) `low` olmalı, daha yükseğe değiştirilmemelidir. `:-)`

3. satırdan 5. satıra kadar: günlük kayıtlarıdır (log entry), paket içinde yaptığınız değişiklikleri belgelendirdiğiniz yerdir (üst düzey değişiklikleri değil – bu amaç için üst düzey geliştirici tarafından oluşturulmuş bir dosya vardır, ki daha sonra bunu `/usr/share/doc/gentoo/changelog.gz` olarak kuracaksınız). Her yeni günlük

kaydı yıldız (‘*’) ile başlayan en üst satırlardan önce eklenmelidir. Bunu **dch(1)** ile veya bir metin düzenleyici ile kendiniz yapabilirsiniz.

Sonuç şöyle birşey olacak:

```
gentoo (0.9.12-1) unstable; urgency=low

 * Initial Release.
 * This is my first Debian package.
 * Adjusted the Makefile to fix $DESTDIR problems.

-- Josip Rodin <joy-mg@debian.org> Wed, 11 Nov 1998 21:02:14 +0100

```

*Paketin Güncellenmesi* (sayfa: 30) bölümünde `ChangeLog` dosyalarının güncellenmesi ile ilgili ayrıntılı bilgi bulabilirsiniz.

## 4.4. `rules` dosyası

Artık şimdi **dpkg-buildpackage(1)**’in asıl paketi oluşturmak için kullanacağı gerçek kurallara bakmalıyız. Bu dosya aslında bir diğer Makefile dosyasıdır, fakat üst düzey kaynaktaki(ler)den farklıdır. `debian/` içindeki diğer dosyalardan farklı olarak, bu çalıştırılabilir olarak imlenmiştir.

Her `rules` dosyası, diğer Makefile dosyaları gibi, kaynağın nasıl işleneceği ile ilgili çeşitli kurallar içerir. Her kural gerçeklenmesi gereken hedeflerden, dosya adlarından veya eylem adlarından (örn. `build:` veya `install:`) oluşur. Çalıştırmak istediğiniz kurallar komut satırı argümanları olarak çağrılır (**./debian/rules build** veya **make -f rules install** gibi). Hedef isminden sonra, bağımlılığı, kuralın bağlı olduğu program veya dosyanın ismini kullanabilirsiniz. Bundan sonra, sekme ile ayrılmış herhangi bir sayıda komut gelebilir. Yeni bir kural ilk sütunda yeni bir hedef bildirimi ile başlar. Boş satırlar ve ‘#’ (diyez) ile başlayan satırlar yorum satırı olarak algılanır ve göz ardı edilir.

Muhtemelen biraz kafanız karıştı, fakat **dh_make**’in oluşturduğu öntanımlı `rules` dosyasını incelerken hepsi netleşecektir. Ayrıca, ‘**info make**’i daha fazla bilgi için okuyabilirsiniz.

**dh_make**’in oluşturduğu `rules` dosyası için bilinmesi gereken, onun sadece bir öneri oluşudur. Basit paketler için çalışacak fakat karmaşık olanlar için çalışmayacaktır, ihtiyaçlarınıza uygun hale gelmesi için bu dosyadan çıkarma ve bu dosyaya ekleme yapmaktan korkmayın. Değiştirmemeniz gereken tek şey kurallardır, çünkü tüm araçlar Debian Politika Kılavuzu[B58]nda belirtildiği gibi bu isimleri kullanmaktadır.

**dh_make**’in bizim için oluşturduğu (yaklaşık) bir `rules` dosyası:

```
#!/usr/bin/make -f
# Sample debian/rules that uses debhelper.
# GNU copyright 1997 to 1999 by Joey Hess.

# Uncomment this to turn on verbose mode.
#export DH_VERBOSE=1

# This is the debhelper compatibility version to use.
export DH_COMPAT=4

CFLAGS = -g
ifneq (,$(findstring noopt,$(DEB_BUILD_OPTIONS)))
CFLAGS += -O0
else
CFLAGS += -O2
endif
```

```
17
18 build: build-stamp
19 build-stamp:
20         dh_testdir
21
22         # Add here commands to compile the package.
23         $(MAKE)
24         #docbook-to-man debian/gentoo.sgml > gentoo.1
25
26         touch build-stamp
27
28 clean:
29         dh_testdir
30         dh_testroot
31         rm -f build-stamp
32
33         # Add here commands to clean up after the build process.
34         -$(MAKE) clean
35
36         dh_clean
37
38 install: build
39         dh_testdir
40         dh_testroot
41         dh_clean -k
42         dh_installdirs
43
44         # Add here commands to install the package into debian/gentoo.
45         $(MAKE) install DESTDIR=$(CURDIR)/debian/gentoo
46
47 # Build architecture-independent files here.
48 binary-indep: build install
49 # We have nothing to do by default.
50
51 # Build architecture-dependent files here.
52 binary-arch: build install
53         dh_testdir
54         dh_testroot
55 #       dh_installdebconf
56         dh_installdocs
57         dh_installexamples
58         dh_installmenu
59 #       dh_installlogrotate
60 #       dh_installemacsen
61 #       dh_installpam
62 #       dh_installmime
63 #       dh_installinit
64         dh_installcron
65         dh_installman
66         dh_installinfo
67 #       dh_undocumented
68         dh_installchangelogs ChangeLog
69         dh_link
70         dh_strip
71         dh_compress
72         dh_fixperms
73 #       dh_makeshlibs
```

```
74          dh_installdeb
75 #        dh_perl
76          dh_shlibdeps
77          dh_gencontrol
78          dh_md5sums
79          dh_builddeb
80
81 binary: binary-indep binary-arch
82 .PHONY: build clean binary-indep binary-arch binary install
```

Gerçek `debian/rules` dosyasında, satırbaşlarındaki boşluklar sekme karakterleridir.

1. satırdakine benzer satırlara muhtemelen daha önce kabuk veya Perl betiklerinizden aşınasınızdır. Bu, işletim sistemine bu dosyanın `/usr/bin/make` ile yorumlanacağını söyler.

6 ve 9. satırlardaki DH_* değişkenlerinin anlamları kısa açıklamalardan anlaşılıyor olmalı. `DH_COMPAT` ile ilgili ayrıntılı bilgi için, **debhelper(1)** kılavuz sayfasının "Debhelper uyumluluk seviyeleri" ("Debhelper compatibility levels") bölümünü okuyunuz.

11'den 16'ya kadar olan satırlar `DEB_BUILD_OPTIONS` parametreleri için destek iskeleti görevi görür, Debian Politika Kılavuzu[B60]'nun 11.1 "İkilikler" ("Binaries") bölümünde anlatılmıştır. Temelde, bu şeyler ikilik paketin hata ayıklama sembolleriyle beraber mi oluşturulacağını ve kurulum sırasında bunların ayıklanıp ayıklanmayacağını kontrol eder. Yine, bu sadece bir iskelettir, yapmanız gerekenler için bir ipucudur. Üst düzey derleme sisteminin (upstream build system) hata ayıklama sembollerini nasıl dahil ettiğini, kurulumda bunları nasıl ayırdığını ve bunu sizin nasıl gerçekleyeceğinizi incelemelisiniz.

Genellikle `CFLAGS` değişkenini kullanarak **gcc**'ye `-g` seçeneğiyle derlemesini söyleyebilirsiniz – eğer bu sizin paketiniz için uygunsa; `build` kuralı içerisindeki `$(MAKE)` çağrısının sonuna `CFLAGS="$(CFLAGS)"` ekleyerek değişkeni türetiniz (aşağıya bakınız). Alternatif olarak, eğer paketiniz bir autoconf `configure` betiğini kullanıyorsa, yukarıdaki dizgeyi `build` kuralındaki `./configure`'nin öncesine ekleyebilirsiniz.

Hata ayıklama sembollerin ayıklama konusuna gelirsek, programlar genelde ayıklanmamış olarak kurulacak şekilde yapılandırılmışlardır ve çoğunlukla bunu değiştirecek bir seçenek yoktur. Ne büyük şans ki, `DEB_BUILD_OPTIONS=nostrip` bayrağını tespit edip sessizce çıkacak olan **dh_strip(1)** aracına sahipsiniz.

18'den 26'ya kadar olan satırlar `build` kuralını (ve onun çocuğu olan `build-stamp`) açıklar. Bu kural, **make**'i bu programı derlemek için kullanılacak olan kendi Makefile dosyası ile çalıştırır. Eğer programınız ikilikleri derlemek için GNU yapılandırma araçlarını kullanıyorsa, kesinlikle `/usr/share/doc/autotools-dev/README.Debian.gz` dosyasını okuyunuz. Docbook'tan kılavuz sayfası üretme örneğine ileride *manpage.1.ex, manpage.sgml.ex* (sayfa: 22) bölümünde değineceğiz.

28–36 arası satırlarda belirtildiği gibi `clean` kuralı ile ilgili; paketin derlenmesi sonucu üretilen ve artık ihtiyaç duyulmayan öğeleri temizler. Bu kural her zaman çalışır olmalıdır (hatta kaynak ağacı temizlenmiş olsa bile!), dolayısıyle lütfen zorlayıcı parametreleri kullanın (örn. **rm** için bu `-f`'dir) veya make'in geri dönüş değerlerini (hataları) ihmal etmesini komut isminden önce bir '–' kullanarak sağlayın.

Kurulum işlemi, 38. satırdaki `install` kuralı başlar. Temel olarak programın Makefile dosyasındaki `install` kuralını çalıştırır, fakat kurulumu `$(CURDIR)/debian/gentoo` dizinine yapar – gentoo'nun Makefile dosyasında kurulumun kök dizini olarak `$(DESTDIR)` kullanmamızın sebebi budur.

Açıklama satırındada belirtildiği gibi, 48. satırdaki `binary-indep` kuralı, mimariden bağımsız paketleri derlemekte kullanılır. Böyle birşeye sahip olmadığımızdan burada hiçbir şey yapmayacağız.

Bir sonraki kuralda, 52 den 79'a kadar olan satırlardaki `binary-arch` kuralı, paketin Debian Politika Kılavuzu[B63]'na uyumluluğunu sağlamak için paket dosyalarınıza çeşitli işlemler uygulayan bazı **debhelper** yardımcı uygulamalarını çağırır.

Eğer paketiniz 'Architecture: all' (Tüm mimarilerde derlenir) türü bir paketse, `binary-arch` kuralını boş bırakıp, paketinizi derlemek için gerekli tüm komutları `binary-indep` kuralına eklemeniz gerekmektedir.

**`debhelper`** araçlarının isimleri `dh_` ile başlar ve devamı aracın ne yaptığının açıklamasıdır. İsimleri yeterince açıklayıcı olmasına rağmen bir miktar daha açıklama yapabiliriz:

- **`dh_testdir(1)`** doğru dizinde olup olmadığınızı kontrol eder (örn. kaynak ağacının kökü).
- **`dh_testroot(1)`** `binary-arch`, `binary-indep` ve `clean` kuralları için gerekli root izinlerine sahip olup olmadığınızı kontrol eder.
- **`dh_installman(1)`** kılavuz sayfalarını hedef dizindeki doğru yere yükleyecektir, sadece ona kaynak paketinin kök dizinine göre kılavuz sayfalarının yerlerini belirtmelisiniz.
- **`dh_strip(1)`** daha küçük hale getirmek için çalıştırılabilir dosyalar ve kütüphanelerden hata ayıklama başlıklarını temizler.
- **`dh_compress(1)`** 4 kB'tan büyük belgeleri ve kılavuz sayfalarını **`gzip(1)`** [B69] ile sıkıştırır.
- **`dh_installdeb(1)`** paketle ilgili dosyaları (örn. bakımcı betiklerini) `debian/gentoo/DEBIAN` dizinine kopyalar.
- **`dh_shlibdeps(1)`** kütüphanelerin ve çalışan dosyaların paylaşımlı kütüphane bağımlılıklarını hesaplar.
- **`dh_gencontrol(1)`** `control` dosyasının düzgün ayarlanmış bir sürümünü `debian/gentoo/DEBIAN` altına yükler.
- **`dh_md5sums(1)`** paketteki tüm dosyalar için MD5 sağlama toplamları üretir.

Tüm bu `dh_*` betiklerinin ne yaptıkları ve seçeneklerinin neler olduğu ile ilgili bilgi almak için kendi kılavuz sayfalarını okuyunuz. Başka (muhtemelen çok faydalı) ve burada bahsedilememiş `dh_*` betikleri de var. Eğer onlara ihtiyaç durarsanız, **`debhelper(7)`** kılavuz sayfasını okuyunuz.

`binary-arch` kısmı gerçekten de ihtiyacınız olmayan özellikleri içeren satırları çıkarmanız gereken kısımdır. gentoo için, cron, init, man, info ve örneklerle ilgili satırları iptal edeceğim, çünkü gentoo bunlara ihtiyaç duymuyor. Ayrıca 68. satırda, `FIXES`'i üstdüzey changelog dosyasının adı olan `ChangeLog` ile değiştireceğim.

Son iki satır (ve burada bahsedilmeyen diğer satırlarla beraber) bazı az veya çok gerekli şeylerdir, ki bunları **`make(1)`** kılavuz sayfasında ve Debian Politika Kılavuzu[B76]nda okuyabilirsiniz. Şu an için, bilmek o kadar da önemli değil.

# 5. `debian/` altındaki diğer dosyalar

`debian/` dizini altında daha başka alt dizinlerin de olduğunu göreceksiniz. Büyük çoğunluğu `ex` önekine veya sonekine sahiptir ki bu da örnekler (examples) manasına gelmektedir. Hepsine bir göz atın. Eğer onlardan herhangi birinin özelliklerini kullanmak ister veya ihtiyaç hissederseniz:

- ilgili belgelere bakın (ipucu: Debian Politika Kılavuzu[B77]),
- eğer gerekliyse dosyaları ihtiyaçlarınıza göre değiştirin,
- eğer `.ex` sonekine sahipseler bunu kaldıracak şekilde tekrar isimlendirin,
- eğer `ex` önekine sahipseler bunu kaldıracak şekilde tekrar isimlendirin,
- eğer gerekliyse `rules` dosyasında değişiklik yapın.

Bunlardan çok kullanılan bazıları alt bölümlerde açıklanmıştır.

## 5.1. `README.Debian`

Orjinal paketinizle, debianlaştırılmış paketiniz arasındaki ek ayrıntılar ve zıtlıklar burada belgelenmelidir.

**dh_make** şuna benzer öntanımlı bir dosya oluşturur:

```
gentoo for Debian
-----------------

<possible notes regarding this package - if none, delete this file>

  -- Josip Rodin <joy-mg@debian.org>, Wed, 11 Nov 1998 21:02:14 +0100
```

Oraya koyacak birşeyimiz olmadığından dosyayı sileceğiz.

## 5.2. conffiles.ex

Bir yazılımla ilgili en sinir bozucu şey, programı özelleştirmek için büyük bir zaman ve çaba harcadıktan sonra, bir yükseltmenin (upgrade) tüm bunları boşa çıkarmasıdır. Debian bu sorunu yapılandırma dosyalarını imleyerek çözmektedir, böylece bir yükseltme yapmak istediğinizde eski yapılandırma dosyanızı korumak isteyip istemediğiniz sorulacaktır.

Bir pakette bunu yapmanın yolu `conffiles` dosyasının her satırına bir yapılandırma dosyasının tam dosya yolunu (genellikle `/etc` altında olurlar) girmektir. Gentoo'nun bir yapılandırma dosyası vardır, `/etc/gentoorc` ve onu `conffiles` dosyasında gireceğiz.

Eğer programınız yapılandırma dosyaları kullanıyor ve onların da üzerine kendisi yazıyorsa, **dpkg** her seferinde kullanıcıdan değişiklikleri onaylamasını isteyeceğinden, onları `conffile` dosyaları olarak imlemek iyi bir yöntem değildir.

Eğer paketlediğiniz program, çalışması için her kullanıcının yapılandırma dosyasını değiştirmesini gerektiriyorsa, gene onu bir `conffile` dosyası olarak imlemeyin.

Örnek yapılandırma dosyalarını 'bakımcı betikleri"nden ('maintainer scripts') elde edebilirsiniz, daha fazla bilgi *postinst.ex, preinst.ex, postrm.ex, prerm.ex* (sayfa: 24) bölümüne bakınız.

Eğer programınızn bir `conffile` dosyası olarak imlenebilecek bir yapılandırma dosyası yoksa, gönül rahatlığıyla `conffiles` dosyasını `debian/` dizininden silebilirsiniz.

## 5.3. cron.d.ex

Eğer paketiniz düzenli aralıklarla gerçeklenmesi gereken görevler gerektiriyorsa, bu dosyayı bunu ayarlamak için kullanabilirsiniz.

Bu günlük döndürmeyi (log rotation) içermemektedir, bunu için **dh_installlogrotate(1)** ve **logrotate(8)** kılavuz sayfalarına bakınız.

Gerekmiyorsa, dosyayı silin.

## 5.4. dirs

Bu dosyada, ihtiyacımız olan fakat normal kurulum işleminin (**make install**) oluşturmadığı dizinler belirtilir.

Öntanımlı olarak şöyle gözükür:

```
usr/bin
usr/sbin
```

Önlerinde / olmadığna dikkat edin. Muhtemelen şuna benzer şekilde değiştireceğiz:

```
usr/bin
usr/share/man/man1
```

Fakat bu dizinler zaten Makefile tarafından oluşturuluyor, dolayısiyle bu dosyaya ihtiyacımız olmayacaktır, o halde sileceğiz.

## 5.5. docs

Bu dosyada, **dh_installdocs**'un bizim için geçici bir dizine yükleyeceği belgelendirme dosyalarının isimleri belirtilir.

Öntanımlı olarak kaynak paketinin kök dizinindeki "BUGS", "README*", "TODO" gibi dosyaların isimlerini içerecektir.

gentoo için, birkaç tane de ben ekledim:

```
BUGS
CONFIG-CHANGES
CREDITS
ONEWS
README
README.gtkrc
TODO
```

Bu dosyayı kaldırıbiliriz de, bunun yerine bu dosyaları `rules` dosyasında `dh_installdocs` komutuna komut satırı argümanları olarak ekleyebiliriz:

```
        dh_installdocs BUGS CONFIG-CHANGES CREDITS ONEWS README \
                       README.gtkrc TODO
```

Yine de, kaynak dosyanızda böyle herhangi bir dosyaya ihtiyaç duymayabilirsiniz. Bu durumda onu güvenli bir şekilde kaldırabilirsiniz. Fakat `rules` dosyasındaki `dh_installdocs` çağrısını kaldırmayın, çünkü bu `copyright` dosyasının ve diğer bazı şeylerin kurulumu için kullanılır.

## 5.6. emacsen-*.ex

Eğer paketiniz, kurulum sırasında baytderlemeli olması gereken Emacs dosyaları sağlıyorsa, bunun gerçekleşmesi için bu dosyayı kullanabilirsiniz.

Bu dosyalar **dh_installemacsen(1)** tarafından geçici bir dizine kurulur. Eğer bu dosyayı kullanırsanız, `rules` dosyasındaki ilgili satırları etkinleştirmeyi unutmayın.

## 5.7. init.d.ex

Eğer paketiniz sistemin başlatılırken çalışması gereken bir artalan süreciyse, açık bir şekilde benim ilk başlardaki önerimi göz ardı etmişsiniz demektir, değil mi? `:-)`

Bu, bir `/etc/init.d/` betiği için genel iskelet dosyasıdır, dolayısiyle sıklıkla düzenlemek zorunda kalacaksınız. Geçici dizine **dh_installinit(1)** tarafından kurulur.

Eğer ihtiyacınız yoksa, bu dosyayı silin.

## 5.8. manpage.1.ex, manpage.sgml.ex

Programın(ların)ızın bir(er) kılavuz sayfası olmalıdır. Yoksa, bu dosyaları kılavuz sayfası şablonu olarak kullanabilirsiniz. Her biri doldurabileceğiniz taslaklardır.

Kılavuz sayfaları normalde **nroff(1)** ile yazılır. manpage.1.ex örneği de nroff ile yazılmıştır. Bu dosyayı nasıl düzenleyeceğinizi öğrenmek için **man(7)** kılavuz sayfasını inceleyiniz.

Diğer taraftan **nroff** yerine SGML ile yazmayı tercih ederseniz, manpage.sgml.ex şablonunu kullanabilirsiniz. Eğer bunu yapacaksanız:

- docbook-to-man paketini yüklememeli,
- control dosyasındaki Build-Depends satırına Build-Depends'ı eklemeli,
- rules dosyanızdaki build kuralında bulunan docbook-to-man çağrısını içeren satırı açıklama satırı olmaktan çıkarmalısınız. 'build' kuralındaki yorum satırını kaldırmak zorundasınız.

Ve dosyanın adını gentoo.sgml gibi birşey olarak değiştirmeyi unutmayın!

Son kılavuz sayfası ismi belgelendirdiği programın ismini içermelidir, bundan dolayı "manpage" olan ismini "gentoo" olarak değiştireceğiz. Dosya adı ayrıca, dosyanın bir kullanıcı komutunun kılavuz sayfası olduğu anlamına geler ".1" sonekini içererecektir. Bunun doğru bölüm numarası olduğundan emin olmalısınız. Kılazuz sayfa bölümlerinin listesi:

| Bölüm No. | Bölüm ismi | Açıklama |
|---|---|---|
| 1 | Kullanıcı komutları | Çalıştırılabilirler ve betikler. |
| 2 | Sistem çağrıları | Çekirdek tarafından sağlanan işlevler. |
| 3 | Kütüphane çağrıları | Sistem kütüphanelerindeki işlevler. |
| 4 | Özel dosyalar | Genelde /dev dizinindeki aygıt dosyaları. |
| 5 | Dosya biçimleri | /etc/passwd gibi dosyaların biçimleri. |
| 6 | Oyunlar | Veye diğer değersiz programlar. |
| 7 | Makro paketleri | Man makroları gibi makrolar. |
| 8 | Sistem yönetimi | Genellikle root tarafından kullanılan komutlar. |
| 9 | Çekirdek yordamları | Çekirdekle ilgili standartdışı çağrılar. |

O halde, gentoo'nun kılavuz sayfası gentoo.1 olarak isimlendirilmelidir. X programları için bölüm numarasının sonuna bir "x" ilave edebilirsiniz, örn. gentoo.1x. Bir gentoo.1 kılavuz sayfası orjinal kaynakta yoktu, örnekleri ve üstdüzey belgeleri okuyarak bir tane yazdım.

## 5.9. menu.ex

X Pencere Sistemi kullanıcıları genellikle programları çalıştırmak için özelleştirilebilen bir menüye sahiptir. Eğer Debian menu paketini yüklüyse, sistemdeki her programa bir menü grubu oluşturulacaktır.

**dh_make**'in oluşturduğu öntanımlı menu.ex dosyası:

```
?package(gentoo):needs="X11|text|vc|wm" section="Apps/see-menu-manual"\
  title="gentoo" command="/usr/bin/gentoo"
```

İki noktadan sonraki ilk alan needs'dir ve programın nasıl bir arayüze ihtiyaç duyduğunu belirtir. Listelenmiş alternatiflerden biri kalacak şekilde değerini değiştirin, örn. "text" veya "X11" kalsın.

Bir sonraki alan section'dır, programın menü ve altmenü girdisi olarak görüneceği bölümdür. Geçerli bölümler /usr/share/doc/debian-policy/menu-policy.html/ch2.html#s2.1 dosyasındadır.

title alanı programın ismidir. Baş harfini büyük yapabilirsiniz, sadece kısa tutun.

Son olarak, command alanı programı çalıştıran komuttur.

Artık menü girdisini şu şekilde değiştirebiliriz:

```
?package(gentoo): needs="X11" section="Apps/Tools" title="Gentoo" command="gentoo"
```

Ayrıca `longtitle`, `icon`, `hints`, v.b gibi başka bölümler de ekleyebilirsiniz. Ayrıntılı bilgi için **`menufile(5)`**, **`update-menus(1)`** ve Debian Politika Kılavuzu[B87]na bakınız.

## 5.10. `watch.ex`

Bu dosya (`devscripts` paketindeki) **`uscan(1)`** ve **`uupdate(1)`** programlarını yapılandırmak için kullanılır. Bunlar orjinal kaynağı aldığınız siteyi gözlemlemek için kullanılır.

Dosyaya koyduklarım:

```
# watch control file for uscan
# Site            Directory  Pattern               Version  Script
ftp.obsession.se  /gentoo    gentoo-(.*)\.tar\.gz  debian   uupdate
```

**İpucu**

Internete bağlanın, dosyayı oluşturduktan sonra **`uscan`**'i çalıştırın. Kılavuz sayfalarını okuyun! `:-)`

## 5.11. `ex.package.doc-base`

Eğer paketinizin kılavuz sayfaları ve bilgilendirme (info) belgeleri dışında belgelendirmesi varsa, `doc-base`'i kullanarak onu kayıtlı hale getirmelisiniz. Böylece kullanıcı bu belgeleri örn. **`dhelp(1)`**, **`dwww(1)`** veya **`doccentral(1)`** gibi programlarla bulabilir.

Bu daha çok `/usr/share/doc/paketadı/` altına yerleştirilen HTML, PS ve PDF dosyalarını içermektedir.

```
Document: gentoo
Title: Gentoo Manual
Author: Emil Brink
Abstract: This manual describes what Gentoo is, and how it can be used.
Section: Apps/Tools

Format: HTML
Index: /usr/share/doc/gentoo/html/index.html
Files: /usr/share/doc/gentoo/html/*.html
```

Dosya biçimiyle ilgili daha fazla bilgi almak için, `/usr/share/doc/doc-base/doc-base.html/` içindeki `doc-base`'e ve **`install-docs(8)`** kılavuz sayfasına bakınız.

Ek belgelerin kurulması ile ilgili ayrıntılar için *Kütüphaneleri değiştirmek* (sayfa: 11) bölümüne bakınız.

## 5.12. `postinst.ex, preinst.ex, postrm.ex, prerm.ex`

Bu dosyalar paketleyicinin betikleri olarak anılır. Bu betikler paketin `control` alanına konulan ve paket yüklendiği, kaldırıldığı veya yükseltildiği zaman **`dpkg`** tarafından çalıştırılan betiklerdir.

Şu an için, olabildiğince paketleyici betiklerini elle düzenlemekten kaçınmalısınız, çünkü biraz karmaşık yapıya meyillidirler. Ayrıntılı bilgi için Debian Politika Kılavuzu[B95]nun, 6. bölümüne ve **`dh_make`** ile birlikte gelen örneklere bakınız.

# 6. Paketin Derlenmesi

Artık paketi derlemeye hazır olmalıyız.

## 6.1. Baştan yeniden derlemek

Programın ana dizinine girin ve aşağıdaki komutu çalıştırın:

```
$ dpkg-buildpackage -rfakeroot
```

Bu komut herşeyi sizin için yapacak ve şunları gerçekleştirecektir:

- **fakeroot**'u kullanarak kaynak ağacının temizler (`debian/rules clean`)
- Paketi yapılandırır (`dpkg-source -b`)
- Paketi derler (`debian/rules build`)
- **fakeroot**'u kullanarak ikilik paketi oluşturur (`debian/rules binary`)
- **gnupg**'yi kullanarak kaynak `.dsc` dosyasını imzalar
- **dpkg-genchanges** ve **gnupg**'yi kullanarak yükleme (upload) `changes` dosyasını oluşturur ve imzalar

Girdi olarak tek ihtiyacınız olan işlem sırasında iki kere kullanacağınız GPG anahtarınızın parolasıdır.

Bütün bunlar yapıldıktan sonra, yukarıdaki dizin içinde (`~/gentoo/`) şu dosyaları göreceksiniz:

`gentoo_0.9.12.orig.tar.gz`
: Bu, orjinal kaynak kodun sıkıştırılmış arşiv dosyasıdır, sadece paket ismi Debian standartlarına uygun olarak yeniden isimlendirilmiştir. **dh-make**'i ilk olarak `-f` seçeneğiyle çalıştırdığımızda oluşturulduğuna dikkat ediniz.

`gentoo_0.9.12-1.dsc`
: Kaynak kodun içeriğinin bir özetidir. Sizin `control` dosyanızdan oluşturulmuştur ve kaynak paketi **dpkg-source(1)** ile açarken kullanılır. Bu dosya GPG imzalıdır, böylece başkaları dosyanın gerçekten size ait olduğuna emin olabilir.

`gentoo_0.9.12-1.diff.gz`
: Bu sıkıştırılmış dosya orjinal kaynak paketinde yaptığınız her değişikliği tekilleştirilmiş fark dosyası ("unified diff") biçiminde içerir. **dpkg-source(1)** ile oluşturulur ve onun tarafından kullanılır. Uyarı: Eğer orjinal kaynak paketinin ismini `paketismi_sürüm.orig.tar.gz` şeklince oluşturmazsanız, **dpkg-source** `.diff.gz` dosyasını düzgün oluşturmada başarısız olacaktır.

  Eğer birisi paketinizi baştan yeniden oluşturmak isterse, yukarıdaki üç dosyayı kullanarak bunu kolaylıkla başarabilir. Dosyaları açma işlemi oldukça basittir: Sadece dosyaları bir yere kopyalayın ve **dpkg-source -x gentoo_0.9.12-1.dsc** komutunu çalıştırın.

`gentoo_0.9.12-1_i386.deb`
: Bu sizin tamamlanmış ikilik paketinizdir. **dpkg** kullanarak herhangi bir diğer paket gibi bunu da sisteminize kurabilir ya da kaldırabilirsiniz.

`gentoo_0.9.12-1_i386.changes`
: Bu dosya o anki paket gözden geçirimiyle ilgili yapılmış tüm değişiklikleri açıklar ve Debian FTP arşivinin bakım programları tarafından ikilik ve kaynak paketlerini kurarken kullanılır. Kısmen `changelog` ve `.dsc` dosyasından üretilir. GPG ile imzalanır, böylece başkaları bunun gerçekten size ait olduğundan emin olabilir.

  Paket üstünde çalışırken, paketin davranışı değişecek ve yeni özellikler eklenecektir. Paketi indirenler bu dosyaya bakarak yapılmış değişiklikleri kolayca görebilir. Debian arşivinin bakımını yapan programlar bu dosyanın içeriğini ayrıca `debian-devel-changes` ileti listesine de postalayacaktır.

`.dsc` ve `.changes` dosyalarındaki uzun sayısal dizgeler, yukarıda bahsedilen dosyalar için MD5 sağlamalarıdır (MD5 checksum). Dosyalarınızı indiren biri **`md5sum(1)`** [B98] ile sınayıp, sayıların uyuşmaması durumunda dosyaların bozulduğunu veya üzerinde oynandığını anlayabilir.

## 6.2. Hızlı yeniden derleme

`debian/rules` dosyasını ayrıntılı bir şekilde ayarlamaya çalışırken, büyük bir paket sözkonusu olduğunda için herşeyi sıfırdan bir defada oluşturmak istemeyebilirsiniz. Deneme amacıyla, üst düzey kaynak paketinden diğer dosyaları oluşturmaksızın sadece `.deb` paketini oluşturabilirsiniz:

```
$ fakeroot debian/rules binary
```

Ayarlamanız bittikten sonra, yukarıdaki adımları takip ederek herşeyi olması gerektiği gibi yeniden oluşturmayı unutmayın. Aksi takdirde, `.deb` dosyalarını başarılı bir şekilde gönderemeyebilirsiniz (upload).

## 6.3. `debuild` komutu

Paket oluşturma işlemini **`debuild`** komutu ile otomatik hale getirebilirsiniz. (**`debuild(1)`** kılavuz sayfasına bakınız.)

**`debuild`** komutunu `/etc/devscripts.conf` veya `~/.devscripts` üzerinden özelleştirebilirsiniz. En azından şu öğelerin bulunmasını öneririm:

```
DEBSIGN_KEYID="GPG_anahtar_kimliğiniz"
DEBUILD_DPKG_BUILDPACKAGE_OPTS="-i -ICVS -I.svn"
```

Bu şekilde, paketleri her zaman GPG anahtarınız ile imazalamış olarak üretebilir ve de istenmeyen içeriğin eklenmesinden kaçınabilirsiniz. (Sponsorluk için de güzeldir.) Örneğin, kaynağı temizleyip paketi bir kullanıcı hesabını kullanarak tekrar aşağıdaki gibi oluşturmak oldukça basittir:

```
debuild clean
debuild
```

## 6.4. `dpatch` sistemi

**`dh_make`** ve **`dpkg-buildpackage`** komutlarının basit kullanımı, içerisinde `debian/` altındaki paket yapımı için gereken dosyaları ve kaynak koda yapılmış yamaları içeren tek büyük bir `diff.gz` dosyası oluşturacaktır. Bu tür bir paket daha sonra kaynak ağacındaki değişiklikleri anlamak ve incelemek için biraz hantal olacaktır. Bu durum da pek hoş değildir.[1]

Yama kümesinin bakımı ve Debian paketleriyle kullanımı için pekçok yöntem önerilmiştir. `dpatch` sistemi önerilen bu yama sürdürme sistemlerinin en basitlerinden biridir. Diğerleri dbs, cdbs, v.s., dir.

`dpatch` ile paketlenmiş bir pakette kaynak kodda yapılan yapılan değişiklikler `debian/patches/` altında yama dosyaları olarak ve açık bir şekilde belgelendirilir ve `debian/` dizini dışındaki kaynak ağacına dokunulmaz. Eğer paketinizi gönderecek bir sponsor arıyorsanız, sponsorunuzun paketinizi incelemesini hızlandırmak için bu tür açık ayrımlar ve belgelendirmeler önemlidir. `dpatch` kullanım yöntemleri **`dpatch(1)`** kılavuz sayfasında açıklanmıştır.

Kaynakla ilgili birisi (bu kendiniz de olabilir) bir yama gönderdiği zaman, `dpatch` ile bunu pakete uygulamanız çok kolaydır:

- Kaynak ağacına `-p1` yaması olması için yamayı düzenleyin.
- `dpatch patch-template` komutunu kullanarak başlığı ekleyin.
- Bunu `debian/patches` altına koyun.

- `dpatch` dosyalarının isimlerini `debian/patches/00list`'e ekleyin.

`dpatch` CPP makrolarını kullanarak mimariye bağlı yama yapabilme yeteneğine de sahiptir.

### 6.5. Gönderirken `orig.tar.gz`'nin eklenmesi

Paketinizi ilk defa arşive gönderiyorsanız, orjinal `orig.tar.gz` dosyasını da göndermelisiniz. Eğer paket sürümü `-0` veya `-1` Debian gözden geçirmesinde değilse **`dpkg-buildpackage`** komutunu `-sa` seçeneği ile kullanmalısınız. Başka bir deyişle, komutu `-sd` seçeneği ile kullanmak orjinal `orig.tar.gz` dosyasının hariç tutulmasına sebep olacaktır.

## 7. Paketin hatalara karşı denetimi

### 7.1. `lintian` ve `linda` paketleri

`.changes` dosyası üzerinde **`lintian(1)`** ve **`linda(1)`** komutlarını çalıştırdığınızda bu programlar pekçok paket hatasını denetleyecektir:

```
$ lintian -i gentoo_0.9.12-1_i386.changes
$ linda -i gentoo_0.9.12-1_i386.changes
```

Elbetteki, dosya adını paketiniz için üretilen `.changes` dosyasının adıyla değiştirmelisiniz. Eğer bazı hatalar gözüküyorsa (`E:` ile başlayan satırlar), açıklamaları okuyup (`N:` ile başlayan satırları) hataları düzeltin ve *Baştan yeniden derlemek* (sayfa: 24) bölümünde anlatıldığı gibi paketi tekrar oluşturun. Eğer `W:` ile başlayan satırlar varsa, bunlar uyarıdır, bu durumda paketi ayarlayın veya uyarı mesajlarının ilgisiz olduğundan emin olun (ve Lintian değişikliklerini uygulayın, bunun için belgesine bakınız.)

**Bilgi**

**`dpkg-buildpackage`** ile paketi oluşturabilir, **`lintian`** ve **`linda`** komutlarını **`debuild(1)`** komutu ile tek bir komut olarak kullanabilirsiniz.

### 7.2. `mc` komutu

**`dpkg-deb(1)`** ile `*.deb` paketlerini açabilirsiniz. Üretilmiş bir Debian paketinin içeriğini de **`debc(1)`** ile listeleyebilirsiniz.

Bu işlem, sadece `*.deb` dosyalarının içeriğini değil aynı zamanda `*.diff.gz` ve `*.tar.gz` dosyalarının içeriğine de göz atmanızı sağlayan **`mc(1)`** gibi bir dosya yöneticisi kullanılarak sezgisel bir hale getirilebilir.

Gerek ikilik gerekse kaynak paketlerde, ilave istenmeyen dosyalara veya sıfır boyutlu dosyalara dikkat edin. Sıklıkla, yapılan paket başarılı bir şekilde temizlenemez, bunun için `rules` dosyanızı bu durumu telafi edecek şekilde ayarlayınız.

**İpucu**

**`zgrep ^+++ ../gentoo_0.9.12-1.diff.gz`** komutu kaynak dosyasına yapılan ekleme ve değişiklikleri, **`dpkg-deb -c gentoo_0.9.12-1_i386.deb`** veya **`debc gentoo_0.9.12-1_i386.changes`** komutu ise ikilik paketteki dosyaları listeler.

### 7.3. `debdiff` komutu

**debdiff(1)** komutu ile iki Debian ikilik paketinini dosya listelerini karışılaştırabilirsiniz. Bu, paketi güncellerken farkında olmadan dosyaların yerdeğiştirilmemiş veya kaldırılmamış veya herhangi bir gözden kaçmış hatanın yapılmamış olduğunu doğrulamak için faydalıdır. Aynı gruptaki *.deb dosyalarına basitçe **debdiff eski-paket.changes yeni-paket.changes** ile bakabilirsiniz.

## 7.4. interdiff komutu

İki .diff.gz dosyasını **interdiff(1)** komutu ile karşılaştırabilirsiniz. Bu, kaynak kod güncellenirken herhangi bir istenmeyen değişikliğin yapılmadığını doğrulamak için kullanılır: **interdiff -z eski-paket.diff.gz yeni-paket.diff.gz**.

## 7.5. debi komutu

**debi(1)** komutunu root olarak kullanarak paktinizi denemek için kurabilirsiniz. Kurulum ve çalıştırma işlemini kendi makinanız dışındakilerde deneyerek gerek yükleme gerekse çalıştırma sırasında karşılaştığınız uyarı ve hataları dikkatlice inceleyiniz.

## 7.6. pbuilder paketi

Temiz bir çevrede (chroot), paket bağımlılıklarının doğrulanması için pbuilder paketi çok faydalıdır. Bu, farklı mimariler için otoderleyici altında kaynak koddan temiz bir paket oluşturulmasını garanti eder, bu şekilde ciddi bir öneme sahip RC (release critical: dağıtımda kritik önemde) sınıfında yer alan bir seri FTBFS'den (Fails To Build From Source: Kaynak Koddan Derlemede Başarısızlıklar) kaçınılmış olunur. Debian otoderleyicisi için http://buildd.debian.org/ adresine bakınız.

**pbuilder** komutunun en temel kullanım şekli, doğrudan root kullanıcısı tarafından çağrılması durumudur. Örneğin, aşağıdaki komutu .orig.tar.gz, .diff.gz ve .dsc dosyalarının bir arada bulunduğu dizinde çalıştırınız:

```
root # pbuilder create # ikinci seferde: pbuilder update
root # pbuilder build foo.dsc
```

Yeni oluşturulan paketler /var/cache/pbuilder/result/ dizinine root iyeliğinde yerleştirilecektir.

**pdebuild** komutu **pbuilder** komutunun paket işlevlerini normal bir kullanıcı olarak kullanmanıza yardımcı olur. .orig.tar.gz dosyasının bulunduğu kaynak kodun kök dizininde aşağıdaki komutu çalıştırınız:

```
$ sudo pbuilder create # ikinci seferde: sudo pbuilder update
$ pdebuild
```

Yeni oluşturulan paketler yine /var/cache/pbuilder/result/ dizinine yerleştirilecek fakat dosyalar root iyeliğinde olmayacaktır.[2]

**pbuilder** tarafından kullanılması için ilave apt–kaynakları eklemek isterseniz ~/.pbuilderrc veya /etc/pbuilderrc içindeki OTHERMIRROR'ı ayarlayıp aşağıdaki komutu çalıştırınız (Sarge için):

```
$ sudo pbuilder update --distribution sarge --override-config
```

--override-config seçeneğine chroot ortamında apt–kaynaklarının güncellenmesi için ihtiyaç vardır.

http://www.netfort.gr.jp/~dancer/software/pbuilder.html adresine, **pdebuild(1)**, **pbuilderrc(5)** ve **pbuilder(8)** kılavuz sayfalarına bakınız.

# 8. Paketin Debian'a gönderilmesi

Şu ana kadar paketinizi denediniz ve artık http://www.debian.org/devel/join/newmaint adresinde belirtildiği gibi Debian'ın yeni paket bakımcısı haline gelme işlemine başlayabilirsiniz.

## 8.1. Paketlerin Debian arşivine gönderilmesi

Resmen bir geliştirici olduktan sonra paketi Debian arşivine yüklemeniz gerekecektir. Bunu elle yapabilirsiniz, fakat **dupload(1)** ve **dput(1)** gibi bu amaç için geliştirilmiş araçları kullanmak işinizi kolaylaştırır.

Önce, **dupload** komutunun yapılandırma dosyasını düzenlemelisiniz. Bunun için ya /etc/dupload.conf dosyasını veya kendi ~/.dupload.conf dosyanızda bazı yerleri değiştirmelisiniz. Dosyaya aşağıdakine benzer satırlar koyun:

```
package config;

$default_host = "anonymous-ftp-master";

$cfg{'anonymous-ftp-master'} = {
      fqdn => "ftp-master.debian.org",
      method => "ftp",
      incoming => "/pub/UploadQueue/",
      # files pass on to dinstall on ftp-master which sends emails itself
      dinstall_runs => 1,
};

1;
```

Elbetteki benim kişisel ayarlarımı kendinizinkine göre değiştirin ve dosyadaki her bir seçeneğin ne anlama geldiğini anlamak için **dupload.conf(5)** kılavuz sayfasını okuyunuz.

$default_host seçeneği en yanıltıcı olanıdır – hangi yükleme kuyruğunun öntanımlı olarak kullanılacağını belirler. "anonymous-ftp-master", birincil olandır, fakat daha hızlı bir tane eklemek isteyeceksinizdir. Yükleme kuyruklarıyla ilgili daha fazla bilgi için Geliştiricilerin Referansında (Developers' Reference) "Paket Yükleme" ("Uploading a package") bölümünü (/usr/share/doc/developers-reference/ch-pkgs.en-us.iso-8859-1.html#s-upload dosyasındadır) okuyunuz.

İnternet sağlayıcınıza bağlanarak şu komutu çalıştırın:

```
$ dupload gentoo_0.9.12-1_i386.changes
```

**dupload**, .changes içindeki dosyaların MD5 sağlamalarının doğruluğunu kontrol eder, gerektiğinde yeniden paketi, *Baştan yeniden derlemek* (sayfa: 24) bölümündeki gibi yeniden oluşturmanız için uyaracaktır, bu işlemden sonra paketiniz düzgün bir şekilde yüklenebilir.

Eğer ftp://ftp-«master.debian.org/pub/UploadQueue/ adresinde **dupload** ile yüklemeyle ilgili bir sorun yaşarsanız, ftp'yi kullanarak gnupg ile imzalanmış *.commands dosyasını aynı adrese kendiniz yüklemeyi deneyerek sorunu çözebilirsiniz.[3] Örneğin,hello.commands'i kullanın:

```
-----BEGIN PGP SIGNED MESSAGE-----

Uploader: Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>
Commands:
 rm hello_1.0-1_i386.deb
 mv hello_1.0-1.dsx hello_1.0-1.dsc

-----BEGIN PGP SIGNATURE-----
Version: 2.6.3ia
```

```
iQCVAwUBNFiQSXVhJ0HiWnvJAQG58AP+IDJVeSWmDvzMUphScg1EK0mvChgnuD7h
BRiVQubXkB2DphLJW5UUSRnjw1iuFcYwH/lFpNpl7XP95LkLX3iFza9qItw4k2/q
tvylZkmIA9jxCyv/YB6zZCbHmbvUnL473eLRoxlnYZd3JFaCZMJ86B0Ph4GFNPAf
Z4jxNrgh7Bc=
=pH94
-----END PGP SIGNATURE-----
```

## 8.2. Paketin özel bir arşive gönderilmesi

Eğer bir geliştirici olarak `http://people.debian.org/~`*hesap_ismi* adresinde **`dupload -t`** *hedef_ismi* ile bir kişisel paket arşivi oluşturmak isterseniz, aşağıdakileri `/etc/dupload.conf` dosyasına eklemelisiniz:

```
# Geliştirici hesabı
$cfg{'hedef_ismi'} = {
    fqdn => "people.debian.org",
    method => "scpb",
    incoming => "/home/hesap_ismi/public_html/package/",
    # Duyuruya gerek yok.
    dinstall_runs => 1,
};
$cfg{'hedef_ismi'}{preupload}{'changes'} = "
    echo 'mkdir -p public_html/package' | ssh people.debian.org  2>/dev/null ;
    echo 'Paket dizini oluşturuldu!'";

$cfg{'hedef_ismi'}{postupload}{'changes'} = "
    echo 'cd public_html/package ;
    dpkg-scanpackages . /dev/null >Packages || true ;
    dpkg-scansources . /dev/null >Sources || true ;
    gzip -c Packages >Packages.gz ;
    gzip -c Sources >Sources.gz ' | ssh people.debian.org  2>/dev/null ;
    echo 'Paket arşivi oluşturuldu!'";
```

Burada APT arşivi hızlı ve özensizce bir SSH kabuğu çalıştırarak oluşturulmuştur. **`dpkg-scanpackages`** ve **`dpkg-scansources`** ile üzerine yazılan dosyalar `/dev/null`'a yönlendirilmiştir. Bu yöntem Debian Geliştiricisi olmayanların paketleri kendi sunucularında tutmaları için de kullanılabilir. Bundan başka, **`apt-ftparchive`** veya başka betiklerle APT arşivi oluşturulabilir.

# 9. Paketin Güncellenmesi

## 9.1. Yeni Debian gözden geçirmesi

Diyelim ki paketinizle ilgili #54321 numaralı bir bir hata bildirimi var ve çözebileceğiniz bir sorunu içeriyor olsun. Paketinizin yeni bir Debian gözden geçirmesini oluşturmak için, aşağıdakilere ihtiyacınız vardır:

- Elbetteki paketin kaynak kodunda hatayı düzeltiniz.
- **`dch -i`** veya **`dch -v <sürüm>-<yayım>`** kullanarak doğrudan Debian değişiklik kayıtları (`changelog`) dosyasının başına yeni bir gözden geçirme ekleyin ve daha sonra da tercih ettiğiniz bir metin düzenleyici ile açıklamalarınızı yazınız.

**İpucu**

Soru: İstenilen biçimde zaman bilgisini nasıl alırsınız?
Yanıt: **822-date** veya **date -R** kullanınız.

- "Closes: #54321" ifadesini takip edecek şekilde değişiklik kaydındaki hatanın kısa açıklamasını ve çözümünü yazınız. Bu şekilde, paketinizin Debian arşivi tarafından kabul edilmesiyle, hata raporu, arşivin bakımını yapan yazılımlar tarafından otomatik olarak sihirli bir şekilde kapatılacaktır.
- *Baştan yeniden derlemek* (sayfa: 24), *Paketin hatalara karşı denetimi* (sayfa: 27) ve *Paketin Debian'a gönderilmesi* (sayfa: 28) bölümlerinde yaptıklarınızı tekrar yapınız. Farklı olarak, değişikliğe uğramadığından ve de Debian arşivinde mevcut olması nedeniyle paketin orjinal kaynak kodunu göndermeyeceksiniz.

## 9.2. Yeni üstdüzey dağıtım (kolay yol)

Şimdi biraz farklı, biraz daha karışık bir durumu değerlendirelim – yeni bir üst düzey sürüm yayınlanmış olsun ve siz de bunu paketlemek istiyorsunuz. O zaman şunları yapmanız gerekir:

- Yeni kaynak paketini indirin ve sıkıştırılmış tar dosyasını (mesela adı gentoo-0.9.13.tar.gz olsun) eski kaynak ağacının üstündeki dizine koyun (örn. ~/gentoo/).
- Eski kaynak dizinine giriniz ve aşağıdaki komutu çalıştırınız:

```
$ uupdate -u gentoo-0.9.13.tar.gz
```

  Elbetteki bu dosya adını programınızın yeni kaynak adıyla değiştiriniz. **uupdate(1)** tar paketini düzgün şekilde isimlendirecek, .diff.gz dosyasından değişikleri uygulamaya çalışacak ve yeni debian/changelog dosyasını güncelleyecektir.

- ../gentoo-0.9.13 dizinine geçin ve *Baştan yeniden derlemek* (sayfa: 24), *Paketin hatalara karşı denetimi* (sayfa: 27) ve *Paketin Debian'a gönderilmesi* (sayfa: 28) bölümlerinde yaptıklarınızı tekrar yapınız.

Eğer debian/watch dosyasını *watch.ex* (sayfa: 24) bölümünde anlatıldığı gibi düzenlerseniz, **uscan(1)**'i kullanarak otomatik ve sihirli bir şekilde gözden geçirilmiş kaynak kodları bulabilir ve daha sonra da onları indirip **uupdate**'i çalıştırabilirsiniz.

## 9.3. Yeni üstdüzey dağıtım (gerçekçi)

Paketleri Debian arşivi için hazırlarken, oluşan paketleri ayrıntılı bir şekilde kontrol etmelisiniz. Burada bunun daha gerçekçi bir örneği var:

1. Yeni üstdüzey kaynaktaki değişiklikleri doğrulayın:
   - Yeni üstdüzey sürümün changelog, NEWS ve yeni sürümle birlikte dağıtılmış her ne başka belgelesi varsa okuyunuz.
   - diff -urN'yi kullanarak eski ve yeni sürüm arasındaki değişikliklerin esas olarak nerelerde yapılmış olduğunu anlamaya çalışın ve ayrıca, şüpheli gördüğünüz herşeye bir göz atın.
2. Eski Debian paketini yenisine uyarlayın:
   - Kaynak tar paketini açın ve kaynak ağacının kökünü <paketismi>-<üstdüzey_sürümü>/ olarak yeniden isimlendirin, **cd** ile dizinin içine girin.
   - Sıkıştırılmış tar dosyasını üst dizine kopyalayın ve ismini <paketismi>_<üstdüzey_sürümü>.orig.
     olarak değiştirin.

- Eski kaynak ağacına yaptığınız benzer değişiklikleri yeni kaynak ağacına da yapın. Olası yöntemler şunlar olabilir:
  - **zcat** */dosya/yolu/<paketismi>_<eski_sürüm>*`.diff.gz` **| patch -p1** komutu,
  - **uupdate** komutu,
  - Eğer kaynak kod yönetimi için bir Subversion deposunu kullanıyorsanız, **svn merge** komutu kullanın veya
  - eğer **dpatch** ile paketlendiyse basitçe eski kaynak kodundaki `debian/` dizinini kopyalayın
- Eski `changelog` kayıtlarını koruyun (sesler geliyor... ama bazen kazalar olabiliyor...)
- Yeni paket sürümü, üstdüzey sürüm numarasına `-1` Debian yayım numarası eklenmesiyle elde edilen yeni sürüm numarası olacaktır, örn., `0.9.13-1`.
- `debian/changelog` dosyasının başına bu yeni sürümü "New Upstream Release" (Yeni üstdüzey dağıtım) şeklinde bir girdi olarak ekleyin. Örneğin, **dch -v 0.9.13-1**.
- Özlü bir şekilde, yeni sürümde bildirilmiş hataları gidermek üzere yaptığınız değişiklikleri açıklayın ve `changelog` dosyasında bu hataları kapatın.
- Özlü bir şekilde, yeni sürümde bildirilmiş hataları gidermek üzere üstdüzey geliştiricinin yaptığı değişiklikleri açıklayın ve `changelog` dosyasında bu hataları kapatın.
- Eğer yama/birleştir (patch/merge) temiz bir şekilde uygulanmazsa, neyin hatalı gitmiş olabileceğini araştırın (ipuçları `.rej` dosyalarındadır). Genellikle sorun, yaptığınız yamanın üstdüzey kodlara zaten uygulanmış olması nedeniye yamanın ilgisiz kalmasıdır.
- Yeni sürüme geçiş sessiz ve rahatsız etmeden olmalıdır (var olan kullanıcılar yeni sürüme geçişi, eski hataların düzeltildiğini ve yeni özelliklerinin eklendiğini farketmeleri dışında farketmemelidir).[4]
- Eğer silinmiş şablon dosyaları herhangi bir nedenle eklemek ihtiyacı duyarsanız, **dh_make** komutunu `-o` seçeneği ile zaten debianlaşmış dizinde çalıştırıp daha sonra da bunları düzenleyiniz.
- Mevcut Debian değişiklikleri yeniden değerlendirilmelidir; aksini zorlayan nedenler olmadıkça üstdüzey sürümün (öyla ya da böyle) zaten sağladığı elemanları atın, yoksa onları saklamayı unutmayın.
- Eğer derleme sistemine özgü herhangi bir değişiklik yapıldıysa (bunu 1. adımdan bileceksiniz), `debian/rules` ve `debian/control` dosyalarını derleme bağımlılıkları açısından gerekliyse güncelleyin.

3. Yeni paketi *debuild komutu* (sayfa: 26) veya *pbuilder paketi* (sayfa: 28) bölümünde açıklandığı gibi derleyiniz. Bunun için **pbuilder** kullanımı da tercih edilebilir.
4. Yeni paketlerin olması gerektiği gibi derlendiğini doğrulayın:
   - *Paketin hatalara karşı denetimi* (sayfa: 27) bölümündekileri uygulayın.
   - *Paket yükseltiminin doğrulanması* (sayfa: 33) bölümündekileri uygulayın.
   - Debian Hata İzleme Sistemi[B136]nde hala açık olan fakat giderilmiş hatalar var mı diye tekrar kontrol edin.
   - Paketin doğru dağıtıma gönderildiğinden, uygun hata kapatmalarının `Closes:` alanında listelendiğinden, `Maintainer:` ve `Changed-By:` alanlarının eşleştiğinden ve dosyanın GPG imzalı olduğundan, v.s. emin olmak için `.changes` dosyasının içeriğini kontrol edin.

5. Paketleme sırasında birşeyi düzeltmek için bir değişiklik yaparsanız, 2. adıma dönerek herşey aklınıza yatana kadar işlemleri tekrarlayın.

6. Eğer paketinizin sponsorlarca desteklenmesi gerekiyorsa, derleme aşamasında özel seçenekler kullanılması gerekliyse (**`dpkg-buildpackage -sa -v ...`** gibi) bunları not alın ve sponsorlarınızın paketi düzgün şekilde oluşturmaları için onları bilgilendirin.

7. Paketi arşive kendiniz göndereceksiniz, *Paketin Debian'a gönderilmesi* (sayfa: 28) bölümünde açıklananları uygulayın.

## 9.4. `orig.tar.gz` dosyası

Eğer paketleri sadece `debian/` dizini olan yeni kaynak ağacından üst dizinde `.orig.tar.gz` dosyası olmaksızın oluşturmaya çalışırsanız, farkında olmadan `diff.gz` dosyası bulunmayan doğal bir kaynak paketi oluşturmuş olursunuz. Bu tür bir paketleme başka hiçbir dağıtımda kullanılamayacak, sadece Debian'a özgü paketler için geçerlidir.[5]

Doğal olmayan ve hem `orig.tar.gz` hem de `debian/` dosyasını içeren bir kaynak paket elde etmek için, *İlk "debianlaştırma"* (sayfa: 9) bölümünde açıklandığı gibi **`dh_make`** komutu tarafından yapılana benzer şekilde, sürümün sıkıştırılmış tar dosyasını ana dizine kopyalamalı ve ismini `<paketismi>_<üstdüzey_sürüm>.orig.tar.gz` olarak değiştirmelisiniz.

## 9.5. `cvs-buildpackage` komutu ve benzetmeler

Paketleme etkinliklerini yönetmek için bir kaynak kod yönetim sistemi kullanmayı düşünmelisiniz. Bunlardan çok kullanılanları için özelleştirilmiş çeşitli sarmalayıcı betikler vardır.

**CVS**
: `cvs-buildpackage`

**Subversion**
: `svn-buildpackage`

**Arch (tla)**
: - `tla-buildpackage`
  - `arch-buildpackage`

Bu komutlar ayrıca, yeni üstdüzey paket sürümlerini de kendileri yapabilmektedir.

## 9.6. Paket yükseltiminin doğrulanması

Yeni bir paket sürümü oluşturduğunuzda, paketin olması gerektiği gibi yükseltildiğinden emin olmak için şu doğrulamaları yapmalısınız:

- Eski sürümden yükseltme yapın.
- Yeni sürümü kaldırmak için eski sürümün kurulmasını sağlayın.
- Yeni paketin kurulumunu yapın.
- Paketi kaldırıp yeniden kurun.
- Yapılandırma dosyaları dahil herşeyiyle sistemden tamamen kaldırın.

Eğer paket `pre/post/inst/rm` betiklerinin kullanımına önem veriyorsa, bunların güncelleme yollarını da denemeyi unutmayın.

Eğer paketiniz Debian tarafından evvelce dağıtıma sokulmuş ise kullanıcıların çoğu son Debian dağıtımındaki sürümden bu paketinize yükseltme yapacaklardır. Bu bakımdan, bu sürümden de yükseltme yapmayı denemeyi unutmayın.

# 10. Nereden yardım alınabilir

Herkese açık bir yerde sorularınızı sormadan önce, lütfen olası kaynakları araştırınız.

- `/usr/share/doc/dpkg`,
- `/usr/share/doc/debian`,
- `/usr/share/doc/autotools-dev/README.Debian.gz`,
- `/usr/share/doc/package/*`

dosyaları ile bu belgede bahsedilen kılavuz ve bilgi sayfayarı bu kaynaklar arasında sayılabilir. Ayrıca, `http://nm.debian.org/` ve `http://people.debian.org/~mpalmer/debian-«mentors_FAQ.html` adreslerindeki tüm bilgilere de bakınız.

Eğer belgelerde paketlemeyle ilgili aradığınız bilgiyi bulamazsanız bunu Debian akıl hocaları <debian–mentors (at) lists.debian.org> ileti listesinde sorabilirsiniz. Daha tecrübeli Debian geliştiricileri sizlere memnuniyetle yardım edeceklerdir, fakat soru sormadan önce en azından bir kaç belgeyi okumuş olmalısınız.

Bu ileti listesi hakkında daha ayrıntılı bilgi edinmek için `http://lists.debian.org/debian-«mentors/` adresine bakınız.

Bir hata raporu (evet, gerçekten hata raporları) aldığınızda, Debian Hata İzleme Sistemi'ne (`http://www.debian.org/Bugs/`) girmenizin ve oradaki belgeleri okuyarak hatalarla verimli bir şekilde baş edebilmenizin vakti gelmiş demektir. `/usr/share/doc/developers-reference/ch-pkgs.en-us.iso-8859-1.html` dosyasındaki Geliştiricilerin Referansı (Developers' Reference) altındaki Hatalarla Baş Etme (Handling Bugs) kısmını okumanızı özellikle tavsiye ederim.

Hala sorularınız varsa onları Debian Geliştiricileri <debian–devel (at) lists.debian.org> ileti listesinde sorabilirsiniz. `http://lists.debian.org/debian-«devel/` adresini bu liste ile ilgili ayrıntılı bilgi almak için inceleyebilirsiniz.

Herşey yolunda gitse bile dua etmeniz gerekir, neden mi? Çünkü bir kaç saat (veya gün) içinde Dünya'nın çeşitli yerlerinden kullanıcılar paketinizi kullanmaya başlayacaklar ve eğer ciddi bir hata yaptıysanız bunların pekçoğundan kızgın içerikli pekçok ileti alacaksınız... Sadece şaka yapıyorum `:)`

Rahat olun ve hata raporları için hazır olun, çünkü paketinizin Debian politikalarına tam uyması için daha yapılacak çok iş var. (Bir kere daha, belgeleri lütfen okuyunuz). İyi şanslar!

# A. Örnekler

Burada üstdüzey geliştiricinin `gentoo-1.0.2.tar.gz` tar paketini paketleyip, tüm paketleri `nm_target`'e göndereceğiz.

## A.1. Basit paketleme örneği

```
$ mkdir -p /dosya/yolu # yeni bir dizin
$ cd /dosya/yolu
$ tar -xvzf /dosya/yolu/gentoo–1.0.2.tar.gz # kaynak paket
$ cd gentoo–1.0.2
$ dh_make -e isim@alan.dom -f /dosya/yolu/gentoo–1.0.2.tar.gz
...  Soruları yanıtla.
...  Kaynak ağacını düzelt.
...  Bu bir betik paketiyse debian/control dosyasında "Architecture: all" yap.
...  ../gentoo_1.0.2.orig.tar.gz dosyasını silme.
$ debuild
... Hiç uyarı gelmemesini sağla.
$ cd ..
$ dupload -t nm_target gentoo_1.0.2–1_i386.changes
```

## A.2. `dpatch` ve `pbuilder` ile paketleme örneği

```
$ mkdir -p /dosya/yolu # yeni bir dizin
$ cd /dosya/yolu
$ tar -xvzf /dosya/yolu/gentoo–1.0.2.tar.gz
$ cp -a gentoo–1.0.2 gentoo–1.0.2–orig
$ cd gentoo–1.0.2
$ dh_make -e isim@alan.dom -f /dosya/yolu/gentoo–1.0.2.tar.gz
...  Soruları yanıtla.
...  Kaynak ağacını düzelt.
...  Paketleri "dpkg-buildpackage -rfakeroot -us -uc" ile derlemeyi dene.
...  Kaynak paketindekileri düzenleyip derlenebeler hale getir.
...  ../gentoo_1.0.2.orig.tar.gz dosyasını silme
$ cd ..
$ cp -a gentoo–1.0.2 gentoo–1.0.2–keep # yedekle
$ mv gentoo–1.0.2/debian debian
$ diff -Nru gentoo–1.0.2–orig gentoo–1.0.2 > yama–dosyası
... Bunu yaparken gentoo-1.0.2 dizininin üstüne yazabilirsiniz.
... Her ihtimale karşı gentoo-1.0.2-keep dizinini silme.
$ mkdir -p debian/patches
$ dpatch patch-template yama/dosyası -p "01_patchname" "patch-file description"\
   < yama–dosyası > debian/patches/01_patchname.dpatch
$ cd debian/patches
$ echo 01_patchname.dpatch >00list
$ cd ../..  # /dosya/yolu'na geç.
$ rm -rf gentoo–1.0.2
$ editor debian/rules
```

`debian/rules` dosyasının özgün hali:

```
config.status: configure
     ./configure --prefix=/usr --mandir=/usr/share
build: config.status
```

```
	${MAKE}
clean:
	$(testdir)
	$(testroot)
	${MAKE} distclean
	rm -rf debian/imaginary-package debian/files debian/substvars
```

`debian/rules` dosyasını **dpatch**'ın kullanabilmesi için bir metin düzenleyici ile şu hale getirin:

```
config.status: patch configure
	./configure --prefix=/usr --mandir=/usr/share
build: config.status
	${MAKE}
clean: clean-patched unpatch
clean-patched:
	$(testdir)
	$(testroot)
	${MAKE} distclean
	rm -rf debian/imaginary-package debian/files debian/substvars
patch: patch-stamp
patch-stamp:
	dpatch apply-all
	dpatch call-all -a=pkg-info >patch-stamp

unpatch:
	dpatch deapply-all
	rm -rf patch-stamp debian/patched
```

**dpatch** sistemi ile kaynak ağacını yeniden paketlemek için artık hazırsınız.

```
$ tar -xvzf gentoo_1.0.2.orig.tar.gz
$ cp -a debian/ gentoo-1.0.2/debian
$ cd gentoo-1.0.2
$ sudo pbuilder update
$ pdebuild
$ cd /var/cache/pbuilder/result/
$ dupload -tnm_target gentoo_1.0.2-1_i386.changes
```

# Notlar

a) Belge içinde dipnotlar ve dış bağlantılar varsa, bunlarla ilgili bilgiler bulundukları sayfanın sonunda dipnot olarak verilmeyip, hepsi toplu olarak burada listelenmiş olacaktır.

b) Konsol görüntüsünü temsil eden sarı zeminli alanlarda metin genişliğine sığmayan satırların sığmayan kısmı ¬ karakteri kullanılarak bir alt satıra indirilmiştir. Sarı zeminli alanlarda ¬ karakteri ile başlayan satırlar bir önceki satırın devamı olarak ele alınmalıdır.

(B1) ../howto/gpl.pdf

(B9) ../glibc/glibc.pdf

(B10) ../man/man1/man1-«patch.pdf

(B30) http://www.debian.org/devel/wnpp/orphaned

(B31) http://www.debian.org/devel/wnpp/rfa_bypackage

(B32) http://www.de.debian.org/devel/wnpp/being_packaged

(B33) http://www.debian.org/social_contract#guidelines

(B40) ../man/man1/man1-«autoconf.pdf

(B41) http://www.debian.org/doc/debian-«policy/

(B42) http://www.debian.org/doc/debian-«policy/

(B43) http://www.debian.org/doc/debian-«policy/

(B44) http://www.debian.org/doc/debian-«policy/

(B53) http://www.debian.org/doc/debian-«policy/

(B54) http://www.debian.org/doc/debian-«policy/

(B58) http://www.debian.org/doc/debian-«policy/

(B60) http://www.debian.org/doc/debian-«policy/

(B63) http://www.debian.org/doc/debian-«policy/

(B69) ../man/man1/man1-«gzip.pdf

(B76) http://www.debian.org/doc/debian-«policy/

(B77) http://www.debian.org/doc/debian-«policy/

(B87) http://www.debian.org/doc/debian-«policy/

(B95) http://www.debian.org/doc/debian-«policy/

(B98) ../man/man1/man1-«md5sum.pdf

(1) Henüz Debian geliştiricisi değilseniz ve paketinizi sponsorunuzun inceledikten sonra göndermesi gerekiyorsa, paketi sponsorunuzun görüp incelemesini mümkün olduğunca kolaylaştırmanız gerekir.

(2) Bu noktada, /var/cache/pbuilder/result/ dizinine kullanıcının yazabilmesi için gerekli izinleri vermenizi ve ~/.pbuilderrc veya /etc/pbuilderrc dosyasının

```
AUTO_DEBSIGN=yes
```

satırını içermesini sağlamanızı öneririm.

Bu, üretilen paketleri ~/.gnupg/ dizinindeki kendi GPG anahtarınızla imzalamanızı mümkün kılacaktır. pbuilder paketinin geliştirilmesi sürmekte olduğundan en güncel resmi belgeleri inceleyerek yapılandırmayı bir de kendiniz gözden geçirmelisiniz.

(3) ftp://ftp-«master.debian.org/pub/UploadQueue/README dosyasına bakın. Bundan başka, dput paketindeki **dcut** komutunu da kullanabilirsiniz.

(4) Please make your package properly updates the config file upon upgrades using well designed `postinst` etc., so that it **doesn't** do things not wanted by the user! These are the enhancements that explain **why** people choose Debian. When the upgrade is necessarily intrusive (eg., config files scattered through various home directories with totally different structure), you may consider to set package to the safe default (e.g., disabled service) and provide proper documentations required by the policy (`README.Debian` and `NEWS.Debian` ) as the last resort. But don't bother with the debconf note.

(B136) `http://www.debian.org/Bugs/`

(5) Paket Debian'a özel bile olsa bazıları hala, `orig.tar.gz` dosyasından başka, `debian/` dizininin içeriğini `diff.gz` içinde paketlemenin daha iyi bir uygulama oldugunu ileri sürerler.

Bu dosya (maint–guide–tr.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

6 Şubat 2007