Panduan Pembuatan Paket Ubuntu

Ubuntu Documentation Project <*ubuntu-doc@lists.ubuntu.com*>
Diterjemahkan oleh Tim Penerjemah Ubuntu Indonesia <*l10n@ubuntu-id.org*>

Panduan Pembuatan Paket Ubuntu

oleh Ubuntu Documentation Project <*ubuntu-doc@lists.ubuntu.com*>

Abstrak

Panduan Pembuatan Paket Ubuntu adalan pengantar pada program pembuatan paket untuk Ubuntu dan distribusi berbasis Debian lainnya

Penghargaan dan Lisensi

Berikut adalah penulis Paguyuban Dokumentasi Ubuntu yang menangani dokumen ini:

- Jordan Mantha

Panduan Pembuatan Paket Ubuntu juga berdasarkan kontribusi dari:

- Alexandre Vassalotti
- Jonathan Patrick Davies
- Ankur Kotwal
- Raphaël Pinson
- Daniel Chen
- Martin Pitt

Beberapa bagian dari Panduan Pembuatan Paket Ubuntu menggunakan Debian New Maintainer's Guide dan Debian Policy Manual.

Disclaimer

Daftar Isi

Tentang Panduan Ini iv
1. Konvensi v
2. Partisipasi dan Umpan balik vi
1. Pengenalan 1
1. Harus Mulai dari Mana 2
2. Prasyarat 3
2. Mulai 5
1. Paket Biner dan Paket Source 6
2. Peralatan Pembuatan Paket 7
3. Pembangun Pribadi: pbuilder 8
3. Dasar Pembuatan Paket 10
1. Pembuatan Paket Dari Awal 11
2. Pembuatan paket menggunakan Debhelper 21
3. Pembuatan Paket Dengan CDBS 26
4. Kesalahan Umum 28
4. Sistem Patch 31
1. Patching Tanpa Sistem Patch 32
2. CDBS dengan Patchsys Sederhana 35
3. dpatch 36
4. Patching paket orang lain 37
5. Meng-update Paket 38
6. Pembuatan Paket Ubuntu 40
1. Meng-upload dan Tinjauan 41
2. Merges dan Syncs 43
3. Pembuatan Paket untuk Kubuntu 47
7. Bug 49
1. Sistem Pelacakan Bug 50
2. Tips Untuk Bug 51
A. Lampiran 55
1. Source Tambahan 56
2. Lingkungan Chroot 57
3. Berkas contoh dh_make 60
4. Daftar dari skrip debhelper 62
B. GNU General Public License 64
1. Preamble 65
2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 66
3. How to Apply These Terms to Your New Programs 71

Tentang Panduan Ini

1. Konvensi

Catatan-catatan berikut akan digunakan diseluruh buku:

Ikon catatan menandakan cuplikan informasi yang menarik, terkadang teknikal, berkenaan dengan diskusi yang ada

Ikon info memberikan petuah atau cara mudah untuk melakukan sesuatu

Ikon hati-hati memberitahukan pembaca akan kemungkinan terjadi masalah dan bantuan untuk menghindarinya.

Ikon peringatan memperingatkan pembaca akan resiko yang dapat terjadi dalam skenario yang diberikan.

Ketentuan referensi silang untuk cetakan akan ditampilkan sebagai berikut:

- Pranala ke dokumen lain atau situs web akan terlihat seperti

Versi PDF, HTML dan XHTML dari dokumen ini akan menggunakan hyperlink untuk menangani referensi silang.

Ketentuan tampilan huruf akan ditampilkan sebagai berikut:

- Nama berkas atau tujuan ke direktori akan ditampilkan dengan huruf `monospace`.
- Perintah yang anda ketik dalam aplikasi perintah siap ketik Terminal akan ditampilkan seperti ini:

  ```
  perintah untuk diketik
  ```

- Opsi yang Anda klik, pilih, atau pakai di antarmuka pengguna akan terlhiat seperti ini.

Pilihan menu, aksi mouse, dan shortcut keyboard:

- Urutan pilihan menu akan ditampilkan sebagai berikut: File ->Open
- Aksi untuk mouse ini diasumsikan Anda memakai konfigurasi mouse untuk tangan kanan. Istilah “klik” dan “klik dua kali” menunjuk pada tombol kiri dari mouse. Istilah “klik kanan” menunjuk pada tombol kanan dari tetikus. Istilah “klik tengah” menunjuk pada tombol tengah dari mouse, menekan roda penggulung, atau menekan tombol kiri dan kanan secara bersamaan, tergantung dari desain mouse Anda.
- Kombinasi shortcut keyboard akan ditampilkan sebagai berikut:**Ctrl-N**. Dimana konvensi untuk tombol “Control”, “Shift,” and “Alternate” adalah **Ctrl**, **Shift**, dan **Alt**, berurutan, dan bearti tombol pertama harus ditekan terlebih dulu sebelum menekan tombol kedua.

2. Partisipasi dan Umpan balik

Buku ini dikembangkan oleh *Tim Dokumentasi Ubuntu* [https://wiki.ubuntu.com/DocumentationTeam]. *Anda* dapat turut serta berpartisipasi pada dokumentasi ini dengan mengirimkan ide atau komentar ke milis Tim Dokumentasi Ubuntu. Informasi mengenai tim, milis, proyek, dll. dapat ditemukan pada *Situs Tim Dokumentasi Ubuntu* [https://wiki.ubuntu.com/DocumentationTeam].

Jika Anda melihat permasalahan dengan dokumentasi ini, atau ingin memberikan saran, Anda bisa mengirimkan laporan bug pada *Bugtracker Ubuntu* [https://launchpad.net/products/ubuntu-doc/+bugs]. Bantuan Anda sangat penting demi kesuksesan dokumentasi kami.

Banyak terima kasih,

-Paguyuban Dokumentasi Ubuntu

Bab 1. Pengenalan

Selamat datang di Panduan Pembuatan Paket Ubuntu! Panduan ini ditujukan bagi Anda yang ingin membuat dan mengelola paket Ubuntu. Walaupun beberapa konsep dalam panduan ini dapat digunakan untuk membuat paket binary untuk penggunaan pribadi, panduan ini didesain untuk mereka yang ingin mendistribusikan paket ke, dan, untuk orang lain. Walau panduan ini ditulis untuk digunakan pada distribusi Linux Ubuntu, panduan ini dapat berguna untuk distribusi lain yang berbasis dari Debian.

Mungkin ada beberapa alasan yang mendasari Anda ketika belajar cara memaketkan program untuk Ubuntu. Pertama, membuat dan memperbaiki paket Ubuntu adalah salah satu cara yang bagus untuk berkontribusi bagi komunitas Ubuntu. Alasan lainnya adalah cara ini merupakan cara yang baik untuk mengetahui bagaimana Ubuntu dan aplikasi yang telah Anda instal bekerja. Atau mungkin Anda ingin menginstal sebuah paket yang tidak ada di repository Ubuntu. Semoga setelah selesai membaca panduan ini Anda memiliki perkakas dan pengetahuan yang cukup untuk melakukan semua itu.

Versi HTML dan PDF dari manual ini tersedia online di *situs web Dokumentasi Ubuntu* [http://help.ubuntu.com].

Anda dapat membeli panduan ini dalam bentuk buku dari *toko Lulu* [http://www.lulu.com/ubuntu-doc]. Anda hanya perlu membayar harga untuk cetakan dan ongkos kirim.

1. Harus Mulai dari Mana

Jika Anda baru mengenal sistem pembuatan paket Debian maka Anda dapat membaca panduan ini menyeluruh, perhatikan dengan baik pada *Bagian 2, “Prasyarat” [3]* , *Bab 2, Mulai [5]* , dan *Bab 3, Dasar Pembuatan Paket [10]* . Bagi Anda yang telah mempunyai pengalaman dengan sistem pembuatan paket Debian dapat melihat *Bab 6, Pembuatan Paket Ubuntu [40]* dan *Bab 7, Bug [49]* .

2. Prasyarat

Panduan ini mengasumsikan bahwa pembaca sudah memiliki pengetahuan yang cukup untuk membangun dan menginstal perangkat lunak dari source pada suatu distribusi Linux. Panduan ini juga menggunakan Command Line Interface (CLI) di seluruh bagian, jadi Anda harus nyaman untuk menggunakan terminal. Setidaknya Anda juga dapat menggunakan hal berikut:

- **make:** GNU Make adalah perkakas yang sangat penting untuk membangun perangkat lunak. Digunakan untuk mengubah tugas kompilasi yang kompleks menjadi mudah. Sangat penting untuk tahu bagaimana menggunakannya, karena kami akan menyimpan informasi mengenai proses pembuatan paket di dalam Makefile. Dokumentasi tersedia pada situs *GNU* [http://www.gnu.org/software/make/manual/make.html].
- **./configure:** Skrip ini disertakan di hampir semua source Linux, khususnya di perangkat lunak yang ditulis menggunakan bahasa kompilasi seperti C dan C++. Skrip ini digunakan untuk membuat Makefile (berkas yang digunakan oleh make) yang terkonfigurasi sesuai dengan sistem komputer yang Anda gunakan. Alat pembuatan paket standar Debian menggunakan skrip ini, jadi Anda diharapkan mengetahui apa yang skrip `configure` lakukan. Informasi tentang `configure` dapat ditemukan dalam dokumentasi `make`.
- **Apt/Dpkg:** Selain digunakan untuk menginstal program, apt dan dpkg memiliki banyak fitur yang berguna untuk pembuatan paket.
  - **apt-cache dump** - menampilkan daftar setiap paket dalam cache. Paket ini sangat berguna apabila digunakan bersama grep pipe seperti apt-cache dump | grep foo untuk mencari paket yang nama atau dependencies-nya menyertakan “foo”.
  - **apt-cache policy** - menampilkan daftar repository (main/restricted/universe/multiverse) dimana suatu paket tersebut ada.
  - **apt-cache show** - menampilkan informasi mengenai paket binary.
  - **apt-cache showsrc** - menampilkan informasi mengenai paket source.
  - **apt-cache rdepends** - menampilkan reverse dependencies unutk sebuat paket (yang dibutuhkan paket untuk melakukan query).
  - **dpkg -S** - menampilkan daftar paket binary yang dimiliki oleh suatu berkas tertentu.
  - **dpkg -l** - menampilkan daftar paket yang baru saja diinstal. Perintah ini mirip dengan apt-cache dump tetapi hanya untuk paket yang diinstal.
  - **dpkg -c** - menampilkan daftar isi dari paket binary. Berguna untuk memastikan bahwa berkas sudah terinstal di tempat yang benar.
  - **dpkg -f** - menampilkan berkas control untuk paket binary. Berguna untuk memastikan bahwa dependencies sudah benar.
  - **grep-dctrl** - mencari informasi khusus di dalam paket. Merupakan penggunaan khusus dari paketgrep (tapi tidak terinstal secara baku).
- **diff dan patch:** Program diff dapat digunakan untuk membandingkan dua berkas dan membuat patch. Contoh yang sering dipakai yaitu diff -ruN file.old file.new > file.diff. Perintah ini akan

membuat diff (rekrusif jika ada direktori) yang menampilkan perubahan, atau “delta”, diantara dua buah berkas.

Program patch digunakan untuk menerapkan suatu patch (biasanya dibuat dengandiff atau program lain) ke suatu berkas atau direktori. Untuk menerapkan patch yang dibuat di atas, Anda dapat mengetik patch -p0 < file.diff. Opsi *-p* memberitahu patch berapa banyak harus men-strip path dari nama berkas dalam patch. -p0 artinya tidak men-strip apapun, atau tetap menggunakan path.

Bab 2. Mulai

1. Paket Biner dan Paket Source

Sebagian besar pengguna distribusi berbasis Debian seperti halnya Ubuntu tidak akan pernah berurusan dengan kode source sebenarnya yang digunakan untuk membuat aplikasi di komputer mereka. Kode source malahdikompilasi menjadi paket *binary* dari paket*source*yang berisi kode source itu sendiri dan aturan untuk membuat paket binary. Pemaket meng-upload paket source dengan perubahan yang telah mereka lakukan untuk sistem build lalu mengompilasi paket binary untuk setiap arsitektur komputer. Sebuah sistem yang terpisah lalu mendistribusikan berkas .deb binary dan source yang telah berubah ke repository mirror.

2. Peralatan Pembuatan Paket

Ada banyak perkakas yang dibuat khusus untuk pembuatan paket di sistem berbasis Debian. Kebanyakan dari perkakas tersebut tidak *penting* untuk membuat paket namun sangat berguna dan sering mengotomatisasi pekerjaan yang diulang. Halaman man dan info yang menyertai mereka adalah sumber informasi yang berharga. Daftar paket berikut dirasa berguna untuk memulai pemaketan:

build-essential
: adalah metapackage yang bergantung pada libc6-dev, gcc, g++, make, and dpkg-dev. Satu paket yang mungkin kurang Anda kenali adalah dpkg-dev. Paket ini mengandung perkakas seperti dpkg-buildpackage dan dpkg-source yang digunakan untuk membuat, membongkar, dan membangun paket source dan binary.

devscripts
: memuat banyak skrip yang membuat pekerjaan pengelolaan paket menjadi mudah. Beberapa yang sering digunakan adalah debdiff, dch, debuild, dan debsign.

debhelper dan dh-make
: adalah skrip untuk mengotomatisasi tugas pembuatan paket. dh-make dapat digunakan untuk membuat "debianization" dan menyediakan banyak berkas contoh.

diff dan patch
: digunakan untuk membuat dan menerapkan patch, berturut-turut. Kedua aplikasi tersebut sering digunakan dalam pembuatan paket karena lebih mudah, bersih dan efisien untuk menampilkan perubahan kecil sebagai patch daripada harus menggunakan banyak salinan berkas.

gnupg
: adalah pengganti dari PGP yang lengkap dan bebas untuk digunakan menandatangani berkas secara digital (termasuk juga paket).

fakeroot
: mensimulasi cara menjalankan perintah dengan hak akses root. Hal ini sangat berguna bila Anda ingin membuat paket binary dari pengguna biasa.

lintian dan linda
: membedah paket Debian, pelaporan bug, dan pelanggaran Kebijakan. Paket ini berisi pemeriksaan otomatis untuk banyak aspek dari Kebijakan Debian seperti halnya error umum.

pbuilder
: membangun sistem chroot dan membuat paket di dalam chroot. Merupakan sistem yang ideal untuk digunakan jika sebelumnya sebuah paket diperiksa apakah telah memiliki dependency yang tepat serta membangun paket yang bersih dan siap untuk diuji serta didistribusikan.

3. Pembangun Pribadi: pbuilder

Penggunaan pbuilder sebagai pembangun paket memungkinkan Anda untuk membangun paket dari dalam lingkungan chroot. Anda bisa saja membangun paket binary tanpa perlu menggunakan pbuilder, tapi Anda harus memiliki semua depedency terinstal di sistem Anda terlebih dahulu. Tapi pbuilder memungkinkan pemaket untuk memeriksa depedency pembangun karena paket dibuat dalam instalasi Ubuntu minimal, dan ketergantungan pembangun di-download menurut berkas `debian/control`.

Berikut adalah panduan singkat cara menginstal, menggunakan, dan meng-update lingkungan pbuilder, tapi banyak detail dari penggunaanpbuilder yang tidak disertakan dalam panduan ini. Halaman manual pbuilder memiliki banyak informasi yang dapat digunakan sebagai rujukan bila Anda mengalami masalah atau membutuhkan informasi yang lebih mendetail.

3.1. Menginstal dan mengonfigurasi lingkungan pbuilder

Pertama kali, dan paling penting, adalah menginstal pbuilder. Jika Anda ingin membuat pbuilder untuk rilis yang lebih baru daripada yang Anda instal saat ini, Anda perlu menginstal secara manual .deb debootstrap (di *http://packages.ubuntu.com*) dari rilis terbaru. Untuk membuat pbuilder jalankan:

```
sudo pbuilder create --distribution <distro> \
 --othermirror "deb http://archive.ubuntu.com/ubuntu <distro> universe multiverse"
```

dimana <distro> adalah rilis yang Anda inginkan (*edgy* sebagai contoh) untuk membuat pbuilder. Jika Anda ingin membuat lebih dari satu lingkungan pbuilder maka Anda perlu menambahkan bendera *--basetgz* di lokasi yang diinginkan untuk lingkungan pbuilder yang terkompres. Lokasi yang baku berada di `/var/cache/pbuilder/base.tgz`. Jika Anda memilih untuk menggunakan *--basetgz* maka Anda perlu menggunakan perintah pbuilder yang lain sehingga pbuilder tahu lingkungan pembangunan mana yang sedang digunakan.

Untuk membuat lingkungan pbuilder memerlukan waktu karena debootstrap akan men-download instalasi Ubuntu minimal.

Cara mudah untuk membuat pbuilder (dan mungkin beberapa pbuilder) adalah dengan menggunakan skrip shell.

3.2. Menggunakan pbuilder

Sekarang Anda telah menjalankan pbuilder untuk itu Anda dapat membangun paket binary dari paket source dengan menggunakan:

```
sudo pbuilder build *.dsc
```

Perintah ini akan membangun seluruh paket source dalam direktori saat ini. Hasil dari paket .deb dan source dapat ditemukan di `/var/cache/pbuilder/result/` (dapat diubah melalui opsi *--buildresult*).

3.3. Meng-update pbuilder

Anda harus memiliki pbuilder terkini kapanpun Anda menguji paket source Anda, khususnya ketika Anda sedang membangun untuk rilis pengembangan yang sangat cepat berubah, untuk menjamin semua depedency terpenuhi. Untuk meng-update pbuilder Anda, gunakan:

```
sudo pbuilder update
```

Jika Anda ingin meng-upgrade pbuilder ke rilis baru Anda dapat menggunakan pbuilder update dalam kombinasi dengan opsi --*distribution*:

```
sudo pbuilder update --distribution <newdistro> --override-config
```

3.4. Beragam pbuilder

Sejauh ini semua informasi dalam bab ini tentang pbuilder hanya berlaku untuk satu pbuilder.
Jika Anda ingin membuat lebih dari satu pbuilder maka Anda harus membuat sebuah skrip shell untuk menangani masing-masing pbuilder yang ingin Anda buat. Salah satu contoh dari skrip shell seperti itu dapat ditemukan di `/usr/share/doc/pbuilder/examples/pbuilder-distribution.sh`. Anda dapat menyalin berkas contoh ini ke suatu tempat di path Anda (menaruhnya di `~/bin/` dan menambah direktori ini ke dalam path eksekusi juga langkah baik) lalu mengubahnya menurut kebutuhan Anda. Normalnya Anda hanya perlu mengubah DISTRIBUTION lalu menambah --othermirror seperti diatas. Anda lalu dapat memanggil skrip ini daripada harus memanggil pbuilder secara langsung.

Bab 3. Dasar Pembuatan Paket

Dua masalah yang seringkali pemaket baru hadapi adalah adanya beragam cara untuk pembuatan paket, dan ada lebih dari satu perkakas untuk melakukan hal itu. Kita akan melihat tiga contoh sistem build umum. Pertama, kita tidak akan menggunakan build helper. Pendekatan ini biasanya yang paling sulit dan jarang digunakan dalam praktiknya tapi memberikan gambaran langsung tentang proses pemaketan. Kedua, kita akan menggunakan debhelper, sistem build yang paling umum digunakan di Debian. Sistem ini membantu pemaket dengan mengotomatisasi tugas yang berulang-ulang. Ketiga, kita akan mengulas tentang **C**ommon **D**ebian **B**uild **S**ystem (CDBS), sistem build yang lebih efektif karena menggunakan debhelper.

Pengembangan paket seringkali membutuhkan instalasi banyak paket (terutama paket -dev yang mengandung header dan berkas pengembangan umum lainnya) yang bukan bagian dari instalasi desktop normal Ubuntu. Jika Anda menghindari instalasi paket ekstra atau ingin mengembangkan untuk rilis Ubuntu yang berbeda (rilis yang sedang dalam pengembangan misalnya) dengan apa Anda sekarang miliki, penggunaan lingkungan chroot sangat direkomendasikan. Panduan untuk mengatur *chroot [57]* dapat ditemukan di Appendix.

1. Pembuatan Paket Dari Awal

Yang dibutuhkan: build-essential, automake, gnupg, lintian, fakeroot and *pbuilder [8]*.

Dalam contoh ini Anda akan menggunakan program GNU *hello* [http://www.gnu.org/software/hello/hello.html] sebagai contoh. Anda dapat men-download tarball source dari *ftp.gnu.org* [http://ftp.gnu.org/gnu/hello/hello-2.1.1.tar.gz]. Untuk keperluan contoh ini, Anda akan menggunakan direktori `~/hello/`.

```
mkdir ~/hello
cd ~/hello
wget http://ftp.gnu.org/gnu/hello/hello-2.1.1.tar.gz
```

Kita juga akan membandingkan paket kita dengan paket yang telah diletakkan di repository Ubuntu. Untuk saat ini, kita akan menempatkannya di direktori `ubuntu` supaya kita dapat melihatnya nanti. Untuk mendapatkan paket source, pastikan Anda menambahkan baris "deb-src" di repository Main pada berkas `/etc/apt/sources.list`. Lalu eksekusi dengan:

```
mkdir ubuntu
cd ubuntu
apt-get source hello
cd ..
```

Tidak seperti kebanyakan perintah apt-get, Anda tidak membutuhkan hak akses root untuk memperoleh paket source, karena paket tersebut di-download pada direktori tempat Anda berada sekarang. Malah, Anda disarankan untuk *hanya* menggunakan apt-get source sebagai pengguna biasa, karena dengan demikian Anda dapat menyunting berkas di paket source tanpa perlu hak akses root.

Yang dilakukan oleh perintah apt-get source adalah:

1. Download paket source. Paket source umumnya mengandung berkas .dsc yang menjelaskan tentang paket dan memberikan md5sum untuk paket source, berkas .orig.tar.gz yang berisi source code dari pengembang, dan berkas .diff.gz yang berisi patch yang telah diterapkan di source code beserta informasi pembuatan paket.
2. Untar berkas .orig.tar.gz ke dalam direktori saat ini.
3. Terapkan berkas .diff.gz ke direktori source yang belum dibongkar.

Jika Anda men-download paket source (.dsc, .orig.tar.gz, and .diff.gz files) secara manual, Anda dapat membongkarnya dengan cara yang sama apt-get source lakukan dengan menggunakan dpkg-source seperti ini:

```
dpkg-source -x *.dsc
```

Hal pertama yang perlu Anda lakukan adalah membuat salinan dari tarball original (terkadang disebut "upstream") dengan format berikut: `<packagename>_<version>.orig.tar.gz`. Langkah

ini melakukan dua hal. Pertama, membuat dua salinan dari source code. Jika Anda tidak sengaja mengubah atau menghapus salinan yang sedang Anda kerjakan maka Anda dapat menggunakan salinan yang telah Anda download. Kedua, akan dianggap praktik pembuatan paket yang tidak baik bila mengubah source tarball original kecuali benar-benar diperlukan. Lihat *Bagian 4, “Kesalahan Umum” [2* untuk alasannya.

```
cp hello-2.1.1.tar.gz hello_2.1.1.orig.tar.gz
tar -xzvf hello_2.1.1.orig.tar.gz
```

Sangat penting untuk menggunakan garis, "_" daripada tanda penghubung "-" diantara nama paket (hello) dan versi (2.1.1). Apabila tidak menggunakannya paket Anda tidak akan dapat dibangun sempurna sebagan paket Debian.

Sekarang kita telah memiliki direktori `hello-2.1.1` yang mengandung berkas source. Sekarang kita perlu membuat direktori kustom debian tempat dimana semua informasi tentang pembuatan paket disimpan, mengizinkan kita untuk memisahkan berkas paket dari berkas source aplikasi.

```
mkdir hello-2.1.1/debian
cd hello-2.1.1/debian/
```

Sekarang kita perlu membuat berkas penting untuk paket source Ubuntu apapun: `changelog`, `control`, `copyright`, and `rules`. Ini adalah berkas yang diperlukan untuk membuat paket binary (berkas .deb) dari source code original (upstream). Mari kita lihat isinya satu per satu.

1.1. changelog

Berkas `changelog` berisi, seperti namanya, daftar perubahan yang telah dilakukan di tiap versi. Berkas ini memiliki format khusus yang berisi nama paket, versi, distribusi, perubahan, dan siapa yang membuat perubahan beserta waktunya. Jika Anda memiliki GPG key, pastikan menggunakan nama dan alamat email yang sama antara `changelog` dengan key Anda. Berikut adalah templat `changelog`:

```
package (version) distribution; urgency=urgency

  * change details
    more change details
  * even more change details

-- maintainer name <email address>[two spaces]  date
```

Format (khususnya untuk tanggal) sangatlah penting. Tanggal harus dalam format RFC822, yang dapat diperoleh dari program 822-date.

Berikut adalah contoh berkas `changelog` untuk aplikasi hello:

```
hello (2.1.1-1) edgy; urgency=low
```

```
   * New upstream release with lots of bug fixes.

-- Captain Packager <packager@coolness.com>  Wed,  5 Apr 2006 22:38:49 -0700
```

Perhatikan bahwa versi tersebut memiliki -1 dibelakangnya, atau sering disebut revisi Debian, yang digunakan supaya pembuatan paket dapat di-update (untuk membenahi bug misalnya) dengan upload baru dalam versi rilis source yang sama.

Ubuntu dan Debian memiliki skema pemberian nomor versi yang sedikit berbeda untuk menghindari konflik antar paket dalam versi source yang sama. Jika paket Debian telah diubah di Ubuntu, maka pake memiliki *ubuntuX* (dimana *X* adalah nomor revisi Ubuntu) yang ditambahkan di akhir nama versi Debian. Jadi jika paket hello Debian telah diubah oleh Ubuntu, maka versi string menjadi `2.1.1-1ubuntu1`. Jika sebuah paket untuk aplikasi tidak ada di Debian, maka nama revisi Debian menjadi *0* (*misalnya*, `2.1.1-0ubuntu1`).

Sekarang lihat `changelog` untuk Ubuntu paket source yang telah kita download sebelumnya:

```
less ../../ubuntu/hello-2.1.1/debian/changelog
```

Perhatikan bahwa dalam kasus ini *distribusinya* adalah *unstable* (cabang dari Debian), karena paket Debian belum diubah oleh Ubuntu. Perlu diingat untuk mengubah *distribusi* menjadi rilis distribusi target Anda.

Pada bagian ini silakan membuat berkas `changelog` dalam direktori `debian` yang sedang Anda masuki sekarang.

1.2. control

Berkas control mengandung informasi yang manajer paket (seperti apt-get, synaptic, dan aptitude) gunakan, build-time dependencies, informasi pengelola, dan banyak lagi.

Untuk paket hello Ubuntu berkas control terlihat seperti ini:

```
Source: hello
Section: devel
Priority: optional
Maintainer: Captain Packager <packager@coolness.com>
Standards-Version: 3.6.1

Package: hello
Architecture: any
Depends: ${shlibs:Depends}
Description: The classic greeting, and a good example
 The GNU hello program produces a familiar, friendly greeting.  It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian
 package.
```

```
It is the Debian version of the GNU Project's `hello world' program
(which is itself an example for the GNU Project).
```

Buat berkas `control` menggunakan informasi di atas (pastikan untuk menyediakan informasi Anda pada ruas *Maintainer*).

Paragraf pertama memberikan informasi mengenai paket source. Mari kita lihat satu persatu:

- **Source:** Ini adalah nama dari paket source, pada kasus disini adalah *hello*.
- **Section:** Repositori apt dibagi ke dalam beberapa bagian untuk mempermudah browsing dan penggolongan perangkat lunak. Dalam kasus ini, hello termasuk ke dalam bagian *devel*.
- **Priority:** Seberapa penting paket ini terhadap pengguna. Harus salah satu dari di bawah ini:
  - **Required** - paket-paket yang sangat penting untuk sistem agar dapat bekerja dengan baik. Jika paket tersebut dihapus biasanya sistem Anda akan rusak dan sulit untuk dipulihkan kembali.
  - **Important** - kumpulan paket minimal untuk sistem yang dapat digunakan. Menghapus kumpulan paket ini tidak akan menyebabkan kerusakan yang tidak bisa diperbaiki pada sistem Anda, tapi kumpulan ini dianggap perkakas utama sehingga tanpanya instalasi Linux apapun tidak akan komplet. Catatan: Ini tidak termasuk program seperti Emacs atau bahkan X Window System.
  - **Standard** - Judul menjelaskan maksudnya.
  - **Optional** - intinya kategori ini adalah untuk paket yang tidak dibutuhkan, atau bulk paket. Tapi, paket-paket ini tidak boleh menimbulkan konflik satu sama lain.
  - **Extra** - paket yang berpotensi konflik dengan paket di salah satu kategori diatas. Juga digunakan untuk paket khusus yang mungkin hanya berguna bagi orang yang telah mengerti tujuan dari paket tersebut.
- **Maintainer:** Nama pengeloa paket dengan alamat email.
- **Standards-Version:** Versi dari *Debian Policy* [http://www.debian.org/doc/debian-policy/] yang melekat pada paket (dalam hal ini, versi 3.6.1). Cara mudah untuk mengetahui versi terkini adalah menggunakan *apt-cache show debian-policy | grep Version*.
- **Build-Depends:** Salah satu bidang terpenting dan seringkali merupakan source dari bug, baris ini mendaftar paket binary (dengan nomor versi jika perlu) yang perlu diinstal untuk membuat paket binary dari paket source. Paket yang penting dibutuhkan oleh *build-essential* dan tidak perlu disertakan dalam baris Build-Depends. Dalam kasus ini, hello, semua paket yang dibutuhkan adalah bagian dari build-essential, jadi baris Build-Depends tidak dibutuhkan. Daftar paket dari build-essential dapat ditemukan di `/usr/share/doc/build-essential/list`.

Paragraf kedua adalah untuk paket binary yang akan dibangun dari source. Jika multiple paket binary dibangun dari paket source, akan ada satu bagian untuk *setiap* paket binary. Sekali lagi, mari kita lihat setiap baris:

- **Package:** Nama untuk paket binary.Banyak kali untuk program sederhana (seperti hello), nama paket source dan binary adalah serupa.
- **Architecture:** Arsitektur dimana paket binary tersebut dibangun. Contohnya adalah:

  - **all** - Source *tidak* bergantung pada suatu arsitektur. Program yang menggunakan Python atau bahasa pemrograman tafsiran lainnya akan mengunakan ini. Paket binary yang dihasilkan akan berakhiran dengan `_all.deb`.
  - **any** - Source *adalah* bergantung pada suatu arsitektur dan harus di-compile untuk seluruh arsitektur yang didukung. Akan ada berkas .deb untuk setiap arsitektur (`_i386.deb` sebagai contohnya)
  - Subset dari arsitektur (i386, amd64, ppc, dll.) dapat dilihat untuk mengindikasikan bahwa source tersebut tergantung pada suatu arsitektur dan tidak bekerja diseluruh arsitektur yang didukung oleh Ubuntu.
- **Depends:** Daftar paket yang paket binary tergantung padanya agar dapat berfungsi. Untuk aplikasi hello, kita lihat `${shlibs:Depends}`, yang adalah variabel untuk mengganti share library yang dibutuhkan. Lihat halaman man `dpkg-source` untuk informasi lebih lanjut.
- **Recommends:** Digunakan bagi paket yang sangat disarankan dan biasanya diinstal bersama paket. Beberapa manajer paket, khususnya aptitude, akan menginstal paket yang disarankan dengan otomatis.
- **Suggests:** Digunakan bagi paket yang serupa atau berguna jika paket ini diinstal.
- **Conflicts:** Digunakan untuk paket yang akan menimbulkan konflik dengan paket ini. Keduanya tidak diinstal secara bersamaan. Jika salah satu diinstal, maka yang lain akan dihapus.
- **Description:** Keterangan singkat dan panjang yang digunakan oleh manajer paket. Formatnya adalah seperti berikut:

  ```
  Description: <single line synopsis>
   <extended description over several lines>
  ```

  Perlu dicatat bahwa ada satu spasi di tiap original baris dalam penjelasan panjang.
  Untuk informasi tentang cara membuat penjelasan yang baik dapat ditemukan di *http://people.debian.org/~walters/descriptions.html*.

1.3. copyright

Berkas ini memberikan informasi tentang hak cipta. Umumnya, informasi hak cipta ditemukan di berkas `COPYING` dalam direktori source program. Berkas ini harus menyertakan informasi seperti nama pencipta dan pemaket, URL dari mana source berasal, baris Hak Cipta dengan tahun dan pemegang hak cipta, dan teks tentang hak cipta itu sendiri. Contoh templatnya adalah:

```
This package was debianized by {Your Name} <your email address>
{Date}

It was downloaded from: {URL of webpage}

Upstream Author(s): {Name(s) and email address(es) of author(s)}

Copyright:
 Copyright (C) {Year(s)} by {Author(s)} {Email address(es)}
```

```
License:
```

Seperti dibanyangkan, aplikasi hello dirilis dalam lisensi GPL. Pada kasus ini cara termudah adalah cukup menyalin berkas `copyright` dari paket Ubuntu:

```
cp ../../ubuntu/hello-2.1.1/debian/copyright .
```

Anda harus menyertakan hak cipta yang lengkap kecuali hak cipta tersebut adalah GPL, LGPL, BSD, atau Artistic License, dalam hal ini Anda dapat merujuk ke berkas yang dimaksud di direktori `/usr/share/common-licenses/`.

Perhatian bahwa berkas `copyright` paket Ubuntu menyertakan pernyataan lisensi untuk manual. Ini sangat penting agar *seluruh* berkas dalam source dilindungi oleh lisensi.

1.4. rules

Berkas `rules` adalah Makefile yang dapat dieksekusi yang memiliki aturan untuk membuat paket binary dari paket source. Untuk hello, akan lebih mudah untuk menggunakan `rules` dari paket Ubuntu:

```
#!/usr/bin/make -f
# Contoh berkas debian/rules - untuk GNU Hello.
# Copyright 1994,1995 by Ian Jackson.
# I hereby give you perpetual unlimited permission to copy,
# modify and relicense this file, provided that you do not remove
# my name from the file itself.  (I assert my moral right of
# paternity under the Copyright, Designs and Patents Act 1988.)
# This file may have to be extensively modified

package = hello
docdir = debian/tmp/usr/share/doc/$(package)

CC = gcc
CFLAGS = -g -Wall
INSTALL_PROGRAM = instal

ifeq (,$(findstring noopt,$(DEB_BUILD_OPTIONS)))
  CFLAGS += -O2
endif
ifeq (,$(findstring nostrip,$(DEB_BUILD_OPTIONS)))
  INSTALL_PROGRAM += -s
endif

build:
        $(checkdir)
        ./configure --prefix=/usr
        $(MAKE) CC="$(CC)" CFLAGS="$(CFLAGS)"
        touch build
```

```
clean:
        $(checkdir)
        rm -f build
        -$(MAKE) -i distclean
        rm -rf *~ debian/tmp debian/*~ debian/files* debian/substvars

binary-indep:   checkroot build
        $(checkdir)
# There are no architecture-independent files to be uploaded
# generated by this package.  If there were any they would be
# made here.

binary-arch:    checkroot build
        $(checkdir)
        rm -rf debian/tmp
        instal -d debian/tmp/DEBIAN $(docdir)
        instal -m 755 debian/postinst debian/prerm debian/tmp/DEBIAN
        $(MAKE) INSTALL_PROGRAM="$(INSTALL_PROGRAM)" \
                prefix=$$(pwd)/debian/tmp/usr instal
        cd debian/tmp && mv usr/info usr/man usr/share
        cp -a NEWS debian/copyright $(docdir)
        cp -a debian/changelog $(docdir)/changelog.Debian
        cp -a ChangeLog $(docdir)/changelog
        cd $(docdir) && gzip -9 changelog changelog.Debian
        gzip -r9 debian/tmp/usr/share/man
        gzip -9 debian/tmp/usr/share/info/*
        dpkg-shlibdeps debian/tmp/usr/bin/hello
        dpkg-gencontrol -isp
        chown -R root:root debian/tmp
        chmod -R u+w,go=rX debian/tmp
        dpkg --build debian/tmp ..

define checkdir
        test -f src/$(package).c -a -f debian/rules
endef

binary: binary-indep binary-arch

checkroot:
        $(checkdir)
        test $$(id -u) = 0

.PHONY: binary binary-arch binary-indep clean checkroot
```

Mari kita lihat berkas ini untuk lebih rinci. Dibagian awal Anda akan melihat deklarasi dari beberapa variabel:

```
package = hello
docdir = debian/tmp/usr/share/doc/$(package)

CC = gcc
CFLAGS = -g -Wall
INSTALL_PROGRAM = instal
```

```
ifeq (,$(findstring noopt,$(DEB_BUILD_OPTIONS)))
  CFLAGS += -O2
endif
ifeq (,$(findstring nostrip,$(DEB_BUILD_OPTIONS)))
  INSTALL_PROGRAM += -s
endif
```

Bagian ini menetapkan CFLAGS untuk compiler dan juga menangani `noopt` dan `nostrip` DEB_BUILD_OPTIONS untuk debugging.

Selanjutnya adalah `build` rule:

```
build:
 $(checkdir)
 ./configure --prefix=/usr
 $(MAKE) CC="$(CC)" CFLAGS="$(CFLAGS)"
 touch build
```

Aturan ini menjalankan *./configure* dengan prefix yang tepat, menjalankan *make*, dan membuat berkas `build` yang berisi kesimpulan dari build yang baru saja dilakukan untuk menghindari berbagai masalah kompilasi.

Aturan berikutnya adalah `clean`, yang menjalankan *make -i distclean* dan menghapus berkas yang dibuat ketika membangun paket.

```
clean:
 $(checkdir)
 rm -f build
 -$(MAKE) -i distclean
 rm -rf *~ debian/tmp debian/*~ debian/files* debian/substvars
```

Selanjutnya kita lihaat rule kosong `binary-indep`, karena tidak ada file architecture-independent yang dibuat dalam paket ini.

Ada beberapa, banyak berkas yang tergantung arsitektur tertentu, jadi `binary-arch` akan digunakan:

```
binary-arch:    checkroot build
  $(checkdir)
  rm -rf debian/tmp
  instal -d debian/tmp/DEBIAN $(docdir)
  instal -m 755 debian/postinst debian/prerm debian/tmp/DEBIAN
  $(MAKE) INSTALL_PROGRAM="$(INSTALL_PROGRAM)" \
  prefix=$$(pwd)/debian/tmp/usr instal
  cd debian/tmp && mv usr/info usr/man usr/share
  cp -a NEWS debian/copyright $(docdir)
  cp -a debian/changelog $(docdir)/changelog.Debian
  cp -a ChangeLog $(docdir)/changelog
  cd $(docdir) && gzip -9 changelog changelog.Debian
  gzip -r9 debian/tmp/usr/share/man
  gzip -9 debian/tmp/usr/share/info/*
  dpkg-shlibdeps debian/tmp/usr/bin/hello
```

```
dpkg-gencontrol -isp
chown -R root:root debian/tmp
chmod -R u+w,go=rX debian/tmp
dpkg --build debian/tmp ..
```

Pertama, perhatikan bahwa aturan ini memanggil aturan `checkroot` untuk memastikan bahwa paket dibangun sebagai root dan memanggil aturan `build` untuk mengompilasi source. Lalu berkas `debian/tmp/DEBIAN` and `debian/tmp/usr/share/doc/hello` dibuat, dan skrip `postinst` dan `prerm>` diinstal di `debian/tmp/DEBIAN`. Lalu *make instal* dijalankan dengan prefix yang memerintahkan untuk menginstal di direktori `debian/tmp/usr`. Setelah itu berkas dokumentasi (NEWS, ChangeLog, dan changelog Debian) dikompres gzip dan diinstal. *dpkg-shlibdeps* dipanggil untuk mencari depedency shared library dari berkas yang dapat dieksekusi hello, lalu menyimpan daftar di berkas `debian/substvars` untuk variabel ${shlibs:Depends} di `control`. Kemudian *dpkg-gencontrol* dijalankan untuk membuat berkas pengendali untuk paket binary, dan membuat pengganti yang dibuat oleh *dpkg-shlibdeps*. Akhirnya, setelah hak akses dari `debian/tmp` telah diatur, *dpkg --build* dijalankan untuk membangun paket binary .deb dan menempatkannya di direktori utama.

1.5. postinst dan prerm

Berkas `postinst` dan `prerm` adalah contoh dari skrip pengelola. Berkas-berkas ini adalah skrip yang dapat dieksekusi setelah instalasi dan sebelum penghapusan paket. Dalam kasus paket hello Ubuntu, berkas-berkas ini digunakan untuk menginstal (dan menghapus) berkas info. Silakan lihat dan salin berkas-berkas ini ke direktori `debian` yang sekarang.

```
cp ../../ubuntu/hello-2.1.1/debian/postinst .
cp ../../ubuntu/hello-2.1.1/debian/prerm .
```

1.6. Membangun Paket Source

Sekarang kita telah melihat berkas-berkas yang ada di direktori `debian` untuk hello secara detail, kita dapat membangun paket source (dan binary). Pertama mari kita pindah ke root dari source yang telah diekstrak:

```
cd ..
```

Sekarang kita membangun paket source menggunakan dpkg-buildpackage:

```
dpkg-buildpackage -S -rfakeroot
```

Tanda -S memerintahkan dpkg-buildpackage untuk membangun paket source, dan tanda -r memerintahkan untuk menggunakan fakeroot supaya kita dapat memiliki hak akses root palsu ketika membuat paket. dpkg-buildpackage akan menggunakan berkas `.orig.tar.gz` dan membuat berkas `.diff.gz` (perbedaan antara tarball original dari pencipta dan direktori yang telah kita buat, `debian/` beserta isinya) dan berkas `.dsc` yang berisi penjelasan serta md5sum dari paket source. Berkas `.dsc` dan `*_source.changes` (digunakan untuk meng-upload paket source) ditandatangani menggunakan GPG key Anda.

Jika Anda tidak memiliki gpg key yang telah dibuat maka Anda akan mendapatkan pesan kesalahan dari debuild. Anda dapat membuat gpg key atau menggunakan *-us -uc* key pada debuild untuk mematikan tandatangan. Tapi Anda tidak bisa meng-upload paket Anda ke Ubuntu tanpa menandatangani paket tersebut.

Untuk memastikan agar debuild mencari gpg key yang tepat maka Anda harus mengatur variabel lingkungan DEBFULLNAME dan DEBEMAIL (misalnya di berkas `~/.bashrc` Anda) dengan nama dan alamat email yang Anda gunakan untuk gpg key di `debian/changelog`

Beberapa orang telah melaporkan bahwa mereka tidak dapat memerintahkan debuild untuk mencari gpg key dengan benar, bahkan setelah mereka mengatur lingkungan variabel seperti contoh diatas. Untuk mengatasi hal ini Anda perlu memberi tanda *-k<keyid>* pada debuild dimana <keyid> adalah ID gpg key Anda.

Sebagai tambahan dari paket source, Anda dapat juga membangun paket binary dengan pbuilder:

```
sudo pbuilder build ../*.dsc
```

Penggunaan pbuilder untuk membangun paket binary adalah sangat penting. Penggunaan ini menjamin bahwa depedency build benar, karena builder menyediakan hanya lingkungan minimal, jadi depedency selama build-time ditentukan oleh berkas `control`.

Anda dapat memeriksa paket sourace dari kesalahan umum dengan menggunakan lintian:

```
cd ..
lintian -i *.dsc
```

2. Pembuatan paket menggunakan Debhelper

Yang diperlukan: Kebutuhan dari *Bagian 1, “Pembuatan Paket Dari Awal” [11]* ditambah debhelper dan dh-make

Sebagai pemaket, Anda akan jarang sekali membuat paket dari original seperti yang telah kita singgung di bab sebelumnya. Dapat Anda bayangkan, banyak sekali pekerjaan dan informasi dalam berkas `rules`, misalnya, jamak ditemukan dalam paket. Agar pembuatan paket semakin mudah dan lebih efisien, Anda dapat menggunakan debhelper untuk membantu pekerjaan ini. Debhelper adalah kumpulan skrip Perl (diawali dengan *dh_* yang mengotomatisasi proses pembangunan paket. Dengan skrip ini, membangun paket di Debian menjadi cukup sederhana.

Dalam contoh ini, Anda akan membangun kembali paket GNU Hello, tetapi sekarang Anda akan membandingkan pekerjaan Anda dengan paket hello-debhelper Ubuntu. Sekali lagi, buat direktori dimana Anda akan bekerja:

```
mkdir ~/hello-debhelper
cd ~/hello-debhelper
wget http://ftp.gnu.org/gnu/hello/hello-2.1.1.tar.gz
mkdir ubuntu
cd ubuntu
```

Kemudian, dapatkan paket source Ubuntu:

```
apt-get source hello-debhelper
cd ..
```

Seperti contoh sebelumnya, yang pertama Anda harus lakukan adalah membongkar paket tarball original (upstream).

```
tar -xzvf hello-2.1.1.tar.gz
```

Daripada menyalin tarball upstream ke `hello_2.1.1.orig.tar.gz` seperti yang telah kita lakukan di contoh sebelumnya, kita akan membiarkan dh_make mengerjakannya untuk kita. Hal yang perlu Anda lakukan adalah mengubah nama folder source sehingga menjadi bentuk *<packagename>-<version>* dimana packagename ditulis menggunakan huruf kecil. Dalam hal ini, untar tarbal akan menghasilkan direktori source yang benar sehingga kita dapat berpindah ke dalamnya:

```
cd hello-2.1.1
```

Untuk membuat "debianization" original dari source kita akan menggunakan dh_make.

```
dh_make -e your.maintainer@address -f ../hello-2.1.1.tar.gz
```

dh_make kemudian akan mengajukan beberapa pertanyaan:

```
Type of package: single binary, multiple binary, library, kernel module or cdbs?
[s/m/l/k/b] s
```

```
Maintainer name : Captain Packager
Email-Address   : packager@coolness.com
Date            : Thu,  6 Apr 2006 10:07:19 -0700
Package Name    : hello
Version         : 2.1.1
License         : blank
Type of Package : Single
Hit <enter> to confirm: Enter
```

Jalankan dh_make -e hanya sekali saja. Jika Anda menjalankannya lagi setelah Anda melakukannya untuk pertama kali, maka aplikasi tidak akan bekerja dengan baik. Jika Anda ingin mengubah atau membuat suatu kesalahan, hapus direktori source dan untar tarball upstream kembali. Kemudian baru Anda masuk ke dalam direktori source dan coba lagi.

Menjalankan dh_make -e akan menyebabkan dua hal:

1. Membuat berkas `hello_2.1.1.orig.tar.gz` di direktori induk,
2. Membuat berkas dasar yang dibutuhkan dalam `debian/` dan beberapa berkas templat (.ex) yang mungkin dibutuhkan.

Program Hello tidak begitu sulit, dan seperti yang sudah Anda lihat di *Bagian 1, “Pembuatan Paket Dari Awal” [11]*, pembuatan paket program tersebut tidak membutuhkan banyak berkas selain berkas dasar. Oleh karena itu, silakan Anda hapus berkas `.ex`:

```
cd debian
rm *.ex *.EX
```

Untuk program hello, Anda tidak memerlukan berkas-berkas `README.Debian` (berkas README untuk hal mengenai Debian, bukan README program), `dirs` (digunakan olehdh_instaldirs untuk membuat direktori yang dibutuhkan), `docs` (digunakan oleh dh_instaldocs untuk menginstal dokumentasi program), atau `info` (digunakan oleh dh_instalinfo untuk menginstal berkas info) di dalam direktori `debian`. Untuk informasi lebih lanjut mengenai berkas tersebut, lihat *Bagian 3, “Berkas contoh dh_make” [6]*.

Pada bagian ini, Anda diwajibkan hanya memiliki berkas `changelog`, `compat`, `control`, `copyright`, dan `rules` dalam direktori `debian`. Dari *Bagian 1, “Pembuatan Paket Dari Awal” [11]*, hanya ada satu tambahan berkas baru yaitu `compat`, yaitu berkas yang memuat versi debhelper (dalam kasus ini adalah 4) yang akan digunakan.

Anda perlu menyesuaikan entri berkas `changelog` di panduan ini, silakan ubah nama paket menjadi hello-debhelper daripada hanya hello:

```
hello-debhelper (2.1.1-1) edgy; urgency=low

  * Initial release

  -- Captain Packager <packager@coolness.com>  Thu,  6 Apr 2006 10:07:19 -0700
```

Dengan menggunakan debhelper, satu hal yang perlu Anda ubah dalam berkas `control` adalah nama (mengganti hello menjadi hello-debhelper) dan menambah debhelper (>= 4.0.0) ke ruas*Build-Depends* untuk paket source. Paket Ubuntu untuk hello-debhelper adalah seperti ini:

```
Source: hello-debhelper
Section: devel
Priority: extra
Maintainer: Capitan Packager <packager@coolness.com>
Standards-Version: 3.6.1
Build-Depends: debhelper (>= 4)

Package: hello-debhelper
Architecture: any
Depends: ${shlibs:Depends}
Conflicts: hello
Provides: hello
Replaces: hello
Description: The classic greeting, and a good example
 The GNU hello program produces a familiar, friendly greeting.  It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's `hello world' program
 (which is itself an example for the GNU Project).
 .
 This is the same as the hello package, except it uses debhelper to
 make the deb.  Please see debhelper as to what it is.
```

Anda dapat menyalin berkas skrip `copyright` dan `postinst` dan `prerm` dari paket Ubuntu hello-debhelper, karena berkas tersebut tidak diubah sejak *Bagian 1, “Pembuatan Paket Dari Awal” [11]*. Anda juga perlu menyalin berkas `rules` sehingga Anda dapat memeriksanya.

```
cp ../../ubuntu/hello-debhelper-2.1.1/debian/copyright .
cp ../../ubuntu/hello-debhelper-2.1.1/debian/postinst .
cp ../../ubuntu/hello-debhelper-2.1.1/debian/prerm .
cp ../../ubuntu/hello-debhelper-2.1.1/debian/rules .
```

Berkas terakhir yang harus Anda lihat adalah `rules`, disinilah kehandalan skrip debhelper terlihat. Versi debhelper dari berkas `rules` adalah terlihat lebih kecil (54 baris dibanding 72 baris yang ada di versi dari *Bagian 1.4, “rules” [16]*).

Versi debhelper terlihat seperti ini:

```
 #!/usr/bin/make -f

package = hello-debhelper

CC = gcc
```

```
CFLAGS = -g -Wall

ifeq (,$(findstring noopt,$(DEB_BUILD_OPTIONS)))
  CFLAGS += -O2
endif

#export DH_VERBOSE=1

clean:
        dh_testdir
        dh_clean
        rm -f build
        -$(MAKE) -i distclean

instal: build
        dh_clean
        dh_instaldirs
        $(MAKE) prefix=$(CURDIR)/debian/$(package)/usr \
                mandir=$(CURDIR)/debian/$(package)/usr/share/man \
                infodir=$(CURDIR)/debian/$(package)/usr/share/info \
                instal

build:
        ./configure --prefix=/usr
        $(MAKE) CC="$(CC)" CFLAGS="$(CFLAGS)"
        touch build

binary-indep: instal
# There are no architecture-independent files to be uploaded
# generated by this package.  If there were any they would be
# made here.

binary-arch: instal
        dh_testdir -a
        dh_testroot -a
        dh_instaldocs -a NEWS
        dh_instalchangelogs -a ChangeLog
        dh_strip -a
        dh_compress -a
        dh_fixperms -a
        dh_instaldeb -a
        dh_shlibdeps -a
        dh_gencontrol -a
        dh_md5sums -a
        dh_builddeb -a

binary: binary-indep binary-arch

.PHONY: binary binary-arch binary-indep clean checkroot
```

Perlu dicatat bahwa tugas seperti testing jika Anda berada di dalam direktori yang benar (dh_testdir), pastikan Anda membangun paket dengan hak akses root (dh_testroot), menginstal dokumentasi (dh_instaldocs dan dh_instalchangelogs), dan pembersihan setelah membangun (dh_clean) telah ditangani secara otomatis. Banyak paket yang lebih rumit dibanding hello memiliki berkas `rules` yang tidak lebih besar karena skrip debhelper telah menangani beberapa tugas tersebut. Untuk daftar lengkap dari skrip debhelper, silakan lihat *Bagian 4, “Daftar dari skrip debhelper” [62]* . Skrip tersebut didokumentasikan dengan baik didalam halaman man. Sangat berguna untuk melihat halaman man (man telah ditulis dengan baik dan tidak terlalu panjang) untuk setiap skrip bantuan yang digunakan oleh berkas `rules` di atas.

2.1. Membangun Paket Source

Sekarang setelah Anda melihat seluruh berkas dalam direktori `debian` untuk hello-debhelper, Anda dapat membangun paket source (dan binary) packages. Pertama, silakan Anda kembali dulu ke direktori source:

```
cd ..
```

Sekarang Anda dapat membangun paket source dengan menggunakan debuild, sebuah skrip wrapper untuk dpkg-buildpackage:

```
debuild -S
```

paket binary, menggunakan pbuilder:

```
sudo pbuilder build ../*.dsc
```

dan terakhir periksa paket source untuk mencari kesalahan yang biasa terjadi dengan menggunakan lintian:

```
cd ..
lintian -i *.dsc
```

3. Pembuatan Paket Dengan CDBS

CDBS adalah perkakas yang menggunakan debhelper sehingga membangun dan mengelola paket Debian menjadi mudah. CDBS memiliki beberapa keuntungan:

- CDBS menghasilkan berkas `debian/rules` yang singkat, mudah dibaca dan efisien
- CDBS membuat debhelper dan autotools dapat digunakan secara otomatis, Jadi Anda tidak perlu kuatir akan pengulangan pekerjaan
- CDBS membantu Anda untuk fokus pada permasalahan pembuatan paket yang lebih penting, karena CDBS membantu tanpa membatasi kustomisasi
- Kelas pada CDBS sudah diuji dengan baik, jadi Anda dapat menghindari hack yang kotor untuk memecahkan permasalahan umum
- Sangat mudah untuk pindah ke CDBS
- CDBS mudah diperluas

3.1. Menggunakan CDBS dalam paket

Menggunakan CDBS untuk paket Ubuntu sangat mudah. Setelah menambah cdbs ke Build-Depends dalam `debian/control`, berkas dasar `debian/rules` yang menggunakan CDBS dapat dimuat dalam 2 baris. Untuk aplikasi C/C++ sederhana tanpa aturan tambahan, seperti hello, berkas `debian/rules` dapat terlihat seperti ini:

```
#!/usr/bin/make -f

include /usr/share/cdbs/1/rules/debhelper.mk
include /usr/share/cdbs/1/class/autotools.mk
```

Hanya inilah yang perlu Anda lakukan untuk membangun program! CDBS menangani instalasi dan pembersihan. Anda lalu dapat menggunakan `.install` dan berkas `.info` untuk mengubah paket Anda dengan fungsi debhelper biasa di beragam bagian untuk `debian/rules`.

Jangan gunakan DEB_AUTO_UPDATE_DEBIAN_CONTROL:=yes untuk mengubah secara otomatis berkas `debian/control`. Hal ini dapat menyebabkan sesuatu yang buruk, dan Debian mempertimbangkan hal itu sebagai alasan untuk menolak paket sebelum masuk ke dalam arsip. Lihat *http://ftp-master.debian.org/REJECT-FAQ.html* [http://ftp-master.debian.org/REJECT-FAQ.html] untuk informasi lebih lanjut.

Seperti yang Anda lihat, CDBS bekerja dengan menyertakan Makefile `.mk` di dalam `debian/rules`. Paket cdbs menyediakan berkas tersebut di `/usr/share/cdbs/1/` yang memungkinkan Anda untuk melakukan berbagai tugas pembuatan paket. Paket lain, seperti quilt menambah modul ke CDBS dan dapat digunakan sebagai Build-Depends. Perlu dicatat bahwa Anda juga dapat menggunakan aturan CDBS sendiri dan menyertakannya di dalam paket. Modul paling berguna yang ikut disertakan dalam paket cdbs adalah:

- `rules/debhelper.mk`: Memanggil debhelper di seluruh bagian yang dibutuhkan

- `rules/dpatch.mk`: Mengizinkan Anda untuk menggunakan dpatch agar mempermudah mem-patch source
- `rules/simple-patchsys.mk`: Cara mudah untuk mem-patch source
- `rules/tarball.mk`: Mengizinkan Anda untuk membangun paket menggunakan tarball terkompresi dalam suatu paket
- `class/autotools.mk`: Memanggil autotools di seluruh bagian yang dibutuhkan
- `class/gnome.mk`: Membangun program GNOME (membutuhkan Build-Depends yang tepat dalam `debian/control`)
- `class/kde.mk`: Membangun program KDE (membutuhkan Build-Depends yang tepat dalam `debian/control`)
- `class/python-distutils.mk`: Mempermudah pembuatan paket program Python

3.2. Informasi lebih lanjut mengenai CDBS

Untuk informasi lebih lanjut mengenai CDBS, lihat panduan Marc Dequènes's pada *https://perso.duckcorp.org/duck/cdbs-doc/cdbs-doc.xhtml*.

4. Kesalahan Umum

4.1. dh_make Berkas Contoh

Ketika Anda menggunakan dh_make untuk membuat dasar "debianization", contoh berkas untuk beragam tugas telah dibuat di direktori `debian/`. Templat mempunyai akhiran .ex. Jika Anda ingin menggunakannya, ubah namanya untuk menghapus ekstensi tersebut. Tapi jika Anda tidak membutuhkannya, hapus berkas tersebut untuk menjaga agar direktori `debian/` tetap bersih.

4.2. Mengubah Tarbal Asli

Ada dua tipe dari paket source, asli dan tidak asli. Paket yang asli adalah paket yang khusus dibuat untuk Ubuntu/Debian. Paket tersebut memiliki direktori debian/ yang berisi informasi pembuatan paket serta perubahan yang telah dilakukan terhadap source yang disertakan dalam tarball (biasanya <packagename>_<version>.tar.gz). Paket tidak asli lebih umum lagi. Paket tidak asli membagi paket source menjadi tarball <packagename>_<version>.orig.tar.gz yang mirip (semoga termasuk md5sum) dengan tarball source yang di-download dari laman proyek dan berkas .diff.gz yang berisi semua perbedaan (direktori `debian/` dan patch) dari tarball source original.

Berikit adalah daftar masalah potensial yang dapat terjadi bila Anda mengubah tarbal original:

1. Reproducibility

   Jika Anda hanya mengambil .diff.gz dan .dsc, berarti Anda atau orang lain tidak bermaksud mereproduksi perubahan di tarball original.

2. Mampu untuk di-upgrade

   Akan lebih mudah untuk meng-upgrade ke versi upstream (dari pencipta) terbaru jika .orig.tar.gz dijaga dan ada batas yang jelas antara source upstream dengan perubahan yang dilakukan untuk menghasilkan paket source Ubuntu.

3. Sinkronisasi Debian ke Ubuntu

   Mengubah tarball original menyulitkan ketika ingin melakukan singkronisasi otomatis dari Debian ke Ubuntu. Normalnya, hanya berkas .diff.gz dan .dsc yang berubah di versi upstream yang sama, karena berkas .orig.tar.gz dibagi di semua revisi Debian atau Ubuntu. Akan lebih sulit untuk menyingkronisasikan jika berkas md5sum dan .orig.tar.gz tidak sama.

4. Penggunaan Kendali Revisi untuk paket Debian

   Jika Anda menggunakan svn (svn-buildpackage) untuk mengatur paket Debian Anda, biasanya Anda tidak menyimpan tarball original di dalam. Jika ada orang lain yang melakukan checkout, dia akan mendapatkan tarball original secara terpisah. Sistem kendali revisi lainnya dapat digunakan hanya untuk melacak berkas pembuatan paket (`debian/`, etc.) dan bukannya keseluruhan source. Tapi, jika .orig.tar.gz tidak sama, maka jelas akan timbul masalah.

5. Melacak Keamanan

Bayangkan situasi dimana seseorang *ingin* memasukkan backdoor/rootkit atau hal jahat lainnya. Jika tarball original diubah, akan mudah untuk melacak melalui .diff.gz apakah orang yang memodifikasi paket tersebut mencoba melakukan perbuatan jahat. Jika tarball diubah, Anda juga perlu memeriksa perbedaan antara tarball dengan source original.

Anda tetap harus mempercayai bahwa pencipta perangkat lunak tidak melakukan suatu perbuatan jahat, tapi hal itu terlepas dari apakah paket original yang diubah.

6. .diff.gz

   Opsi untuk menggunakan .diff.gz untuk merefleksikan perubahan yang dilakukan terhadap tarball original telah ada, jadi akan memudahkan untuk melakukan perubahan tanpa harus menyentuh tarball original.

Sangat masuk akal untuk mengubah tarbal original jika satu atau lebih hal berikut adalah benar:

- Tarbal tersebut mengandung bagian tidak gratis yang tidak bisa didistribusikan. Hapus bagian tersebut, dan catat hal itu ketika membuat paket. Seringkali paket seperti itu menggunakan "dfsg" (merupakan kependekan dari Debian Free Software Guidelines) di nama paket dan/atau versi untuk mengindikasikan bahwa bagian non-free telah dihapus.
- Pemrogram hanya menyediakan source bzip2
  - Cukup bunzip2 untuk .tar.bz2 dan gzip -9 tar yang dihasilkan.
  - md5sums dari .tar yang Anda miliki dengan .tar original haruslah sesuai!
  - Menyediakan aturan get-orig-source di debian/rules yang mengatur konversi ini secara otomatis.
- Langsung mengimpor dari SVN
  - Menyediakan get-orig-source dalam debian/rules.

Berikut ini adalah not alasan untuk mengubah tarball original:

- Tata Letak Direktori Yang Salah

  

  dpkg-source cukup fleksibel dan mengatur untuk pembuatan tata letak direktori yang benar bahkan jika:
  - Direktori di dalam tarball tidak memakai nama <upstream>-<version>.
  - Tidak ada subdirektori di dalam tarball.
- Berkas harus dihapus untuk menjaga agar .diff.gz tetap berukuran kecil (misalnya berkas yang dibuat oleh autotools). Semua hal yang perlu dihapus atau dipindahkan ada di aturan clean. Karena .diff.gz dibuat oleh diff -u, Anda tidak akan melihat berkas yang dihapus di .diff.gz.
- Berkas-berkas yang harus diubah. Berkas yang harus diubah harus masuk ke dalam .diff.gz. Karena itulah kegunaannya!
- Hak akses berkas yang salah. Anda dapat menggunakan `debian/rules` untuk keperlukan ini.

Apa yang harus saya lakukan dengan berkas .orig.tar.gz yang sudah disertakan dalam dir `debian/`?

Jangan memaketkan ulang. Anda dapat meminta kepada pencipta untuk menghapus direktori debian dan menyediakan .diff.gz. Ini membuat mudah ketika mereka melihat ulang hasil kerjanya, dan memisahkan pembuatan paket dari source paket.

Ide yang bagus untuk mengontak pencipta program dan menanyakan apakah Anda boleh membenahi masalah autoconf, tata letak direktori, alamat Free Software Foundation yang lama di berkas COPYRIGHT, atau hal lain yang tidak berhubungan dengan pembuatan paket tapi akan lebih baik bagi Anda sehingga Anda tidak perlu "patch" source di .diff.gz.

4.3. Informasi Hak Cipta:

Berkas `debian/copyright` harus memuat:

- Informasi lisensi untuk *semua* berkas di source. Terkadang pencipta menaruh lisensi di `COPYING` tapi memiliki informasi lisensi yang berbeda untuk beberapa berkas di source.
- The copyright holder(s) and year(s).
- *Semua* lisensi kecuali itu merupakan salah satu dari lisensi yang ditemukan di `/usr/share/common-licenses`, dimana Anda cukup menyertakan mukadimahnya saja.

Bab 4. Sistem Patch

Sering kali terjadi source yang berada di upstream perlu di-patch, atau program perlu disesuaikan agar dapat bekerja di Ubuntu atau bug perlu dibenahi di dalam source sebelum diperbaiki di upstream. Tapi bagaimana cara kita menunjukkan perubahan yang telah dilakukan? Cara yang paling sederhana adalah dengan membuat perubahan di dalam paket source yang telah dibongkar, dalam hal ini perubahan akan ditunjukkan di dalam berkas `.diff.gz`.Tapi, ini tidak ideal. Jika terdapat lebih dari satu patch maka Anda akan kehilangan kemampuan untuk memisahkan beberapa patch, hal ini dikarenakan Anda hanya akan melihat satu buah berkas diff besar yang juga berisi berkas paket (dalam `debian/`. Ini juga menyulitkan Anda ketika akan mengirim patch ke upstream. Yang baik adalah memisahkan source pengembang dari perubahan yang dilakukan untuk Ubuntu. Tempat yang paling baik untuk menaruh informasi ini adalah di dalam `debian/` yang telah digunakan untuk berkas paket. Dalam bab ini kita akan mengulas berbagai cara untuk membuat patch dengan cara yang elegan.

1. Patching Tanpa Sistem Patch

Seperti telah disebutkan diatas, seseorang dapat mem-patch source original dengan hanya membuat perubahaan dalam direktori source yang telah dibongkar. Contoh nyata dari hal ini adalah cron. Jika Anda mengambil paket source cron lalu melihat isi `.diff.gz` maka Anda akan melihat bahwa beberapa berkas paket source original telah diubah.

```
apt-get source cron
zgrep +++ cron*.diff.gz
```

Tapi seperti telah kita singgung sebelumnya bahwa cara ini bukan cara terbaik untuk menunjukkan patch. Salah satu cara yang lebih baik adalah dengan membuat berkas patch secara individu, tapi menaruhnya di `debian/patches/` dan menerapkan patch (menggunakan patch) di `debian/rules`. Cara inilah yang digunakan untuk udev:

```
apt-get source udev
zgrep +++ udev*.diff.gz
ls udev*/debian/patches/
less udev*/debian/rules
```

Berkas `rules` memiliki peraturan berikut untuk menerima dan tidak menerima patch:

```
# Apply patches to the package
patch: patch-stamp
patch-stamp:
        dh_testdir
        @patches=debian/patches/*.patch; for patch in $$patches; do \
                test -f $$patch || continue; \
                echo "Applying $$patch"; \
                patch -stuN -p1 < $$patch || exit 1; \
        done
        touch $@

# Remove patches from the package
unpatch:
        dh_testdir
        @if test -f patch-stamp; then \
                patches=debian/patches/*.patch; \
                for patch in $$patches; do \
                        reversepatches="$$patch $$reversepatches"; \
                done; \
                for patch in $$reversepatches; do \
                        test -f $$patch || continue; \
                        echo "Reversing $$patch"; \
                        patch -suRf -p1 < $$patch || exit 1; \
                done; \
                rm -f patch-stamp; \
        fi
```

Semua ini terlihat sangat baik, tapi bagaimana cara kita membuat patch baru untuk udev dengan menggunakan skema ini? Pendekatan yang umum adalah dengan:

1. salin isi source yang ada ke direktori sementara
2. terapkan seluruh patch ke berkas yang ingin Anda sunting; Jika Anda ingin membuat patch baru, terapkan seluruh patch yang telah ada. (hal ini penting karena patch yang ada tergantung dari patch sebelumnya)

   Jika Anda ingin, dapat pula menggunakan debian/rules untuk ini: caranya hapus patch yang datang *setelah* patch yang ingin Anda sunting, lalu jalankan 'debian/rules patch'. Nama sebenarnya untuk target patch bervariasi, Saya pribadi sejauh ini telah melihat beberapa nama yang digunakan: patch setup apply-patches unpack patch-stamp. Anda perlu melihat di dalam `debian/rules` untuk mengetahui nama yang digunakan untuk target patch.
3. salin lagi seluruh isi source:

   ```
   cp -a /tmp/old /tmp/new
   ```
4. masuk ke /tmp/new, dan lakukan modifikasi
5. kembali ke direktori source original Anda, lalu generate patch dengan:

   ```
   diff -Nurp /tmp/old /tmp/new > mypatchname.patch
   ```

1.1. Contoh 1.

Mari kita membuat patch baru untuk udev dengan nama `90_penguins.patch` yang isinya mengganti kata *Linux* dengan *Penguin* dalam berkas `README` udev:

```
cd udev*/
cp -a . /tmp/old
pushd /tmp/old
debian/rules patch
cp -a . /tmp/new; cd ../new
sed -i 's/Linux/Penguin/g' README
cd ..
diff -Nurp old new > 90_penguins.patch
popd
mv /tmp/90_penguins.patch debian/patches
rm -rf /tmp/old /tmp/new
```

1.2. Contoh 2.

Apa yang terjadi jika kita ingin menyunting patch yang telah ada? Kita dapat menggunakan prosedur yang sama dengan Contoh 1 tetapi kita harus menerapkan patch untuk disunting terlebih dahulu:

```
cp -a . /tmp/old
pushd /tmp/old
cp -a . /tmp/new; cd ../new
patch -p1 < debian/patches/10-selinux-include-udev-h.patch
```

```
sed -i '1 s/$/***** HELLO WORLD ****/' udev_selinux.c
cd ..
diff -Nurp old new > 10-selinux-include-udev-h.patch
popd
mv /tmp/10-selinux-include-udev-h.patch debian/patches
rm -rf /tmp/old /tmp/new
```

Sementara cara ini secara teknis baik, namun dapat berubah menjadi rumit dan tidak terkendali. Untuk membuat proses patch lebih mudah dan tidak berbelit maka sistem patch dikembangkan. Kita akan melihat beberapa contoh populernya.

2. CDBS dengan Patchsys Sederhana

Sistem CDBS build helper (lihat *Bagian 3, “Pembuatan Paket Dengan CDBS” [26]*) di dalamnya sudah memiliki sistem patch sederhana. Anda cukup menambahkan berkas *simple-patchsys.mk* dalam `debian/rules`. Contohnya adalah pmount. Berkas `rules` yang ada akan terlihat seperti ini:

```
#!/usr/bin/make -f
include /usr/share/cdbs/1/rules/debhelper.mk
include /usr/share/cdbs/1/class/autotools.mk
include /usr/share/cdbs/1/rules/simple-patchsys.mk

common-post-build-arch::
        # Generate a POT file
        cd po; intltool-update -p --verbose
```

Patchsys sederhana juga memiliki editor patch terintegrasi yang dinamakan cdbs-edit-patch. Anda dapat menggunakan cdbs-edit-patch untuk mengubah patch yang telah ada atau membuat patch baru. Aplikasi ini akan berlaku untuk patch yang ada, dan jika patch tersebut ditemukan, maka shell baru akan ditampilkan kehadapan Anda. Anda lalu diperkenankan membuat perubahan yang ingin Anda tambahkan ke patch lalu ketika selesai ketik *Ctrl-D* untuk keluar dari shell dan membuat patch baru. Patch ini selanjutnya akan disimpan di `debian/patches/`

3. dpatch

Salah satu sistem patch yang populer adalah dpatch. Sistem ini memiliki skrip dpatch-edit-patch yang mirip dengan cdbs namun menyimpan patch dengan cara yang sedikit berbeda. Aplikasi ini menggunakan berkas yang bernama `debian/patches/00list` untuk mencari dan mengurutkan nama patch yang ingin diubah. Ini berarti Anda dapat mengurutkan patch sesuai keinginan Anda dan Anda dapat menonaktifkan sebuah patch tanpa perlu menghapusnya. Tapi, ini berarti juga Anda harus memutakhirkan `00list` setiap kali Anda akan menambah suatu patch. Jika dpatch-edit-patch dipanggil dengan dua argumen maka aplikasi tersebut akan mengubah/membuat patch yang disebut oleh argumen pertama relatif terhadap patch yang disebut oleh argumen kedua. Atau dengan kata lain:

```
dpatch-edit-patch new.dpatch old.dpatch
```

akan menerapkan perubahan di `old.dpatch` lalu membuat `new.dpatch`. Perlu dicatat bahwa patch yang dibuat dengan dpatch biasanya mempunyai akhiran .dpatch. Hal ini karena dpatch menyimpan patch di tempat yang sedikit berbeda dibanding dengan patch normai yang hanya menambah header khusus.

Contoh sesungguhnya dari penggunaan dpatch adalah paket xterm.

4. Patching paket orang lain

Hal paling penting yang harus dipertimbangkan saat patching paket yang dikelola oleh orang lain adalah untuk tetap menyimpan sistem patch (atau kurang lebih) yang pengelola telah terapkan. Hal ini untuk memastikan konsistensi dan membuat pengelola paket senang menerima patch dari Anda.

Ide yang bagus juga untuk memisahkan patch secara logika daripada membuat satu patch yang besar. Jika pencipta upstream menerapkan salah satu perubahan Anda tetapi tidak yang lainnya maka akan lebih mudah untuk men-drop patch lalu sunting patch monolithic untuk meng-update-nya.

Bab 5. Meng-update Paket

Jika Anda telah menggunakan distribusi Linux sebelumnya, Anda sadar kadang-kadang ada bug di dalam program. Dalam distribusi Debian dan Ubuntu, bug sering diperbaiki melalui pembuatan paket dengan cara mem-patch kode source. Terkadang ada juga bug pada pembuatan paket itu sendiri yang dapat menimbulkan kesulitan.

Untuk mem-patch kode source program, Anda cukup men-download paket source Ubuntu (dengan apt-get source) dan lakukan perubahan. Anda kemudian dapat menambah entri baru ke `debian/changelog` menggunakan dch -i atau dch -v <version>-<revision> untuk menetapkan revisi baru. Ketika Anda menjalankan debuild -S dari direktori source Anda akan mendapatkan paket source baru dengan berkas `.diff.gz` baru di dalam direktori utama yang berisikan perubahan. Permasalahan yang terjadi karena pendekatan ini adalah pembedaan antara source dan patch yang tidak jelas.

Solusi untuk masalah ini adalah dengan memisahkan perubahan dalam source code menjadi patch individu yang disimpan dalam direktori `debian`. Salah satu sistem patch yang ada adalah dpatch. Patch disimpan dalam direktori `debian/patches/` dan menggunakan format khusus.

Untuk membuat dpatch, lakukan langkah-langkah berikut secara berurutan.

Buat ruang kerja sementara dan dua salinan dari direktori source saat ini:

```
mkdir tmp
cd tmp
cp -a ../<package>-<version> .
cp -a <package>-<version> <package>-<version>.orig
```

Buat perubahan di dalam direktori `<package>-<version>`

Buat berkas patch menggunakan diff dan tempatkan di dalam direktori `debian/patches`.

```
diff -Nru  <package>-<version>.orig <package>-<version> > patch-file
```

Buat dpatch menggunakan dpatch patch-template dan berkas dengan nama `00list` yang memuat daftar dpatches:

```
dpatch patch-template -p "01_patchname" "patch-file description" \
 < patch-file > 01_patchname.dpatch
echo 01_patchname.dpatch >00list
```

Sekarang Anda dapat menempatkan `01_patchname.dpatch` dan `00list` di direktori `debian/patches` dalam paket source Anda:

```
mkdir ../<package>-<version>/debian/patches
cp 01_patchname.dpatch 00list ../<package>-<version>/debian/patches
```

```
cd ..
rm -rf tmp
```

Anda juga dapat menyunting patch yang sudah ada dengan menggunakan dpatch-edit-patch.

Setelah seluruh perubahan Anda lakukan, entri changelog akan ditambah, dan dpatch telah ditambahkan ke berkas `debian/control` (jika diperlukan), Anda kemudian dapat membangun ulang paket source dengan debuild -S.

Agar paket source yang telah Anda perbaiki dapat di-upload ke repository Ubuntu, Anda harus mendapatkan sponsor dari seseorang yang memiliki hak untuk meng-upload. Lihat *Bagian 1, “Meng-upload dan Tinjauan” [4]* untuk lebih jelasnya. Terkadang, daripada mengirimkan seluruh paket source (.diff.gz, .dsc, dan .orig.tar.gz), akan lebih mudah dan efisien jika hanya mengirimkan perbedaan antara paket source yang ada di repository saat ini dengan paket source yang telah Anda perbaiki. Perkakas untuk melakukan hal ini disebut debdiff. Penggunaan debdiff sangat mirip dengan diff tetapi perkakas tersebut dibuat khusus untuk pembuatan paket. Anda dapat melakukan debdiff pada paket source dengan:

```
debdiff <oldpackage>.dsc <newpackage>.dsc > package.debdiff
```

atau paket binary dengan:

```
debdiff <oldpackage>.deb <newpackage>.deb > package.debdiff
```

Debdiffs sangat bagus untuk melampirkan laporan bug dan apabila Anda telah mempunyai sponsor untuk meng-upload.

Bab 6. Pembuatan Paket Ubuntu

1. Meng-upload dan Tinjauan

Setelah Anda membuat paket source (baik itu paket yang baru atau hanya update/perbaikan bug), Anda ingin mendistribusikan paket itu sehingga orang lain dapat menikmati pekerjaan Anda. Cara yang paling efektif untuk melakukan ini untuk Ubuntu adalah dengan memasukkannya ke dalam repository Universe. Komunitas para pengembang yang bertanggung jawab untuk repository Universe dikenal sebagai Masters of the Universe (*MOTU* [https://wiki.ubuntu.com/MOTU]). *REVU* [http://revu.tauware.de] adalah perkakas berbasis web yang memberikan tempat bagi orang-orang untuk meng-upload paket source mereka untuk dilihat oleh orang lain dan untuk ditinjau oleh MOTU dengan cara yang terstruktur.

1.1. Memberikan kontribusi sebagai Peng-upload

Pertama, Anda diharuskan untuk menambahkan *GPG Key* [https://wiki.ubuntu.com/GPGKey] Anda ke keyring REVU. Hal ini untuk menjamin agar paket tersebut benar-benar datang dari Anda dan membantu untuk melacak upload.

REVU menggunakan akun *Launchpad* [https://launchpad.net] untuk melihat gpg key Anda jadi pastikan Anda memiliki akun Launchpad dan taruh gpg key Anda dalam profil pengguna yang ada disana. Setelah itu Anda dapat bergabung dengan tim *Kontributor Universe* [https://launchpad.net/people/ubuntu-universe-contributors] dan kirim email ke *admin@tiber.tauware.de* [mailto:admin@tiber.tauware.de] untuk meminta agar gpg key Anda di-update pada REVU. Setelah selesai Anda dapat segera meng-upload paket. Anda tidak memerlukan kata sandi untuk meng-upload paket, kata sandi hanya diperlukan untuk login ke situs web dan membalas komentar.

Untuk dapat meng-upload ke REVU, key Anda tidak perlu ditanda-tangani oleh yang lain, tetapi hal ini ide yang baik untuk dilakukan.

1.1.1. Meng-upload paket

dput digunakan untuk meng-upload ke REVU. Versi dput pada Ubuntu sudah mengetahui tentang REVU jadi Anda tidak perlu mengubah berkas konfigurasi apapun. Hanya upload paket yang sudah ditanda-tangani, dan tidak seperti repository lain, Anda harus selalu menyertakan berkas tarbal original, jika tidak peninjau tidak dapat melihat paket source yang telah diekstrak. Untuk melakukan hal ini, gunakan opsi "-S -sa" pada debuild atau dpkg-buildpackage agar hanya membuat paket source dan untuk menyertakan source original dalam upload.

Setelah paket source dibuat, Anda dapat menggunakan dput dengan berkas konfigurasi perubahan diatas untuk meng-upload paket dengan hanya cukup menetapkan berkas `_source.changes` yang telah dibuat:

```
dput revu *_source.changes
```

Jika Anda meng-upload ulang paket yang diubah (setelah mendapatkan tinjauan), Anda akan mendapatkan kesalahan seperti ini:

```
Upload package to host revu
Already uploaded to tauware.de
Doing nothing for myapp_source.changes
```

Untuk memperbaikinya, tambahkan opsi -f pada dput untuk memaksakan upload atau untuk menghapus berkas `.upload` yang telah dibuat oleh dput.

Proses untuk upload dilakukan setiap lima menit, jadi jika upload Anda tidak juga muncul, silakan kontak pengelola REVU lewat email, atau kunjungi kanal #ubuntu-motu pada IRC Freenode.

1.1.2. Bagaimana cara login ke REVU

Setelah upload pertama, Anda otomatis terdaftar dalam database dan akan diberikan kata sandi acak. Pada situs web *REVU* [http://revu.tauware.de], gunakan alamat e-mail yang Anda gunakan dalam berkas changelog sebagai login dan klik link 'recover password'. Setelah ini Anda akan menuju halaman yang memiliki kata sandi terenkripsi Anda dan juga instruksi bagaimana menggunakannya.

1.1.3. Melihat dan mengomentari upload

Paket yang di-upload ke REVU adalah milik publik. Anda dapat menjelajah tanpa login ke dalam sistem. Akan tetapi, untuk komentar terhadap paket yang telah di-upload hanya tersedia bagi pengguna yang telah terdaftar. Sebagai peng-upload, Anda hanya dapat mengomentari paket yang Anda upload. Hal ini berguna untuk memberi informasi kepada peninjau mengenai perubahan yang Anda buat diantara dua paket yang Anda upload.

1.1.4. Aturan tambahan

- Anda harus meninjau ulang paket dari kelemahan keamanan dan harus menyiapkan patch untuk kelemahan tersebut.
- Paket dapat ditolak karena dasar masalah keamanan yang telah diketahui.
- Anda harus memasukkan berkas hak cipta dan lisensi, dan harus memberi izin bagi paket untuk dimasukkan ke dalam komponen Universe dan dapat didistribusikan ulang melalui mirror Ubuntu.
- Paket harus dibangun diatas komponen utama dalam rilis stabil Ubuntu saat ini. Suatu paket diizinkan untuk bergantung pada paket lain yang telah berada dalam repository Universe.

1.1.5. Mendapatkan Bantuan

Jika Anda membutuhkan bantuan pada langkah ini, atau jika Anda mempunyai pertanyaan mengenai REVU, Anda dapat bertanya di #ubuntu-motu yang ada di jaringan IRC Freenode.

2. Merges dan Syncs

Requirements: build-essential, automake, gnupg, lintian, fakeroot, patchutils, debhelper dan *pbuilder [8]*.

Ubuntu berbasis dari distribusi Linux Debian dan menggunakan sistem manajemen paket (APT) yang sama. Pada original tiap siklus pengembangan Ubuntu, paket-paket di Ubuntu di-update menggunakan arsip dari Debian unstable. Akan tetapi, karena Ubuntu tidak sama dengan Debian, maka beberapa paket itu harus dimodifikasi agar dapat bekerja di Ubuntu. Mungkin juga ada perbaikan bug yang dilakukan oleh pengembang Ubuntu dalam suatu paket. Anda dapat menentukan apakah suatu perbaikan telah dilakukan dengan melihat versi paket. Jika versi paket menyertakan ubuntu di dalamnya (contohnya seperti gimp-2.2.9-3ubuntu2), maka pengembang Ubuntu telah melakukan perubahan, dan paket itu tidak sama lagi dengan paket Debian. Ada sekitar 1000 lebih paket yang dimodifikasi dalam repository Universe.

Pada original tiap siklus pengembangan Ubuntu, suatu keputusan dibuat berkenaan dengan versi suatu paket. Tentu saja jika versi Debian tidak berubah sejak rilis Ubuntu terakhir, maka tidak ada yang perlu diubah. Akan tetapi, jika terdapat versi paket terbaru dalam Debian, maka satu dari dua hal dapat terjadi. Jika semua hasil modifikasi Ubuntu (perbaikan bug, dependency, dll) juga diperbaiki dalam paket baru Debian, maka kami cukup langsung mengambil paket Debian tersebut. Keputusan ini yang disebut sebagai *sync*. Akan tetapi, jika versi baru Debian masih memiliki beberapa persoalan yang mengharuskan dibuatnya versi Ubuntu, maka perubahan tersebut harus diterapkan juga ke dalam versi baru Debian. Keputusan ini yang disebut sebagai *merging*.

2.1. Tutorial Untuk Merging

Proses merging mengharuskan kita untuk melihat perubahan yang terjadi pada paket source Debian dan Ubuntu serta menentukan apa yang telah diubah dan perubahan mana yang terkait dengan Ubuntu. Mari kita lihat sebuah contoh, program untuk membuat CD yang disebut xcdroast.

Untuk memulainya, buat folder untuk menyimpan proyek kita, lalu pindah ke sana:

```
mkdir ~/xcdroast
cd ~/xcdroast
```

Sekarang download seluruh paket source yang dibutuhkan ke dalam direktori ini:

- Paket tarball source xcdroast yang digunakan untuk seluruh versi:
  - *xcdroast_0.98+0alpha15.orig.tar.gz* [http://snapshot.debian.net/archive/2006/01/16/debian/pool/main/x/xcdroast/xcdroast_0.98+0alpha15.orig.tar.gz]
- Berkas paket source Ubuntu Breezy:
  - *xcdroast_0.98+0alpha15-1.1ubuntu1.dsc* [http://doc.ubuntu.com/files/packagingguide/xcdroast_0.98+0alpha15-1.1ubuntu1.dsc]

    - *xcdroast_0.98+0alpha15-1.1ubuntu1.diff.gz* [http://doc.ubuntu.com/files/packagingguide/xcdroast_0.98+0alpha15-1.1ubuntu1.diff.gz]
- Berkas paket source Debian yang digunakan untuk paket Breezy:
    - *xcdroast_0.98+0alpha15-1.1.diff.gz* [http://doc.ubuntu.com/files/packagingguide/xcdroast_0.98+0alpha15-1.1.diff.gz]
    - *xcdroast_0.98+0alpha15-1.1.dsc* [http://doc.ubuntu.com/files/packagingguide/xcdroast_0.98+0alpha15-1.1.dsc]
- Berkas paket source baru Debian yang akan digunakan untuk paket Dapper:
    - *xcdroast_0.98+0alpha15-3.dsc* [http://doc.ubuntu.com/files/packagingguide/xcdroast_0.98+0alpha15-3.dsc]
    - *xcdroast_0.98+0alpha15-3.diff.gz* [http://doc.ubuntu.com/files/packagingguide/xcdroast_0.98+0alpha15-3.diff.gz]

Langkah ini dapat diselesaikan dengan mencari paket Debian pada *packages.debian.org* dan paket Ubuntu pada *packages.ubuntu.com*.

Paket penting yang harus diinstal saat melakukan penggabungan (atau untuk setiap pembuatan paket Ubuntu) adalah devscripts. Jika Anda belum menginstalnya, silakan instal terlebih dahulu sebelum melanjutkan ini.

Dengan melihat changelog Ubuntu Anda sudah dapat melihat perbedaan mana yang terjadi antara paket Ubuntu dan paket Debian. Untuk xcdroast, changelog Ubuntu dapat ditemukan pada *changelogs.ubuntu.com* [http://changelogs.ubuntu.com/changelogs/pool/universe/x/xcdroast/xcdroast_0.98+0alpha15-1.1ubuntu1/changelog]. Dari sini diketahui bahwa berkas .desktop telah diperbaiki dan dapat diinstal dengan baik untuk menutup laporan bug dalam *Malone* [https://launchpad.net/malone/bug/2698].

Sekarang teliti perubahan yang terjadi dalam paket source:

```
debdiff xcdroast_0.98+0alpha15-1.1.dsc xcdroast_0.98+0alpha15-1.1ubuntu1.dsc | \
 ubuntu.debdiff | less ubuntu.debdiff
```

Baris yang dimulai dengan - menunjukan baris yang telah dihapus dari paket Debian, dan baris yang dimulai dengan + menunjukkan baris yang ditambahkan ke paket Ubuntu.

Hal berikut inilah yang kita lihat:

- Dalam debian/rules perintah instal, dibandingkan cp, telah digunakan untuk menginstal ikon xcdroast. Sekarang juga terdapat baris baru untuk menginstal berkas .desktop.
- Dalam debian/changelog perubahan yang terjadi ditambahkan ke entri changelog.
- Dalam debian/dirs usr/share/applications telah ditambahkan agar baris instal diatas dapat bekerja dengan baik.
- xcdroast.desktop telah ditambah

Sekarang kita mengetahui bagaimana source Ubuntu telah berubah. Sekarang kita harus melihat perubahan yang terjadi dalam source Debian.

```
debdiff xcdroast_0.98+0alpha15-1.1.dsc xcdroast_0.98+0alpha15-3.dsc > debian.debdiff
less debian.debdiff
```

Ada lebih banyak dalam debdiff ini dibanding yang sebelumnya. Salah satu cara agar kita lebih mengerti tentang apa yang telah diubah adalah dengan melihat berkas yang diubah dalam debdiff:

```
grep diff debian.debdiff
```

Ini menandakan bahwa debian/postinst, debian/rules, debian/changelog, debian/doc-base.manual, debian/control, dan debian/menu telah diubah dalam versi baru Debian.

Dengan demikian kita harus memeriksa debian/rules untuk melihat apakah perubahan pada Ubuntu juga dipakai. Kita juga dapat melihat bahwa debian/dirs tidak berubah dari versi lama Debian. Sekarang coba lihat pada berkas. Kita dapat meng-unpack berkas source dengan menggunakan dpkg-source:

```
dpkg-source -x xcdroast_0.98+0alpha15-3.dsc
```

Perintah ini akan melakukan men-decompress berkas xcdroast_0.98+0alpha15.orig.tar.gz, lalu membuat direktori xcdroast-0.98+0alpha15 dan kemudian menerapkan perubahan yang terdapat dalam xcdroast_0.98+0alpha15-3.diff.gz.

Sekarang silakan pindah ke direktori debian:

```
cd xcdroast-0.98+0alpha15/debian
```

Kita dapat melihat pada berkas `rules` bahwa perubahan yang dilakukan oleh Ubuntu tidak diterapkan ke versi baru Debian. Ini berarti bahwa:

```
cp debian/xcdroast.xpm `pwd`/debian/$(PACKAGE)/usr/share/pixmaps
```

...harus diubah menjadi:

```
#cp debian/xcdroast.xpm `pwd`/debian/$(PACKAGE)/usr/share/pixmaps

#instal desktop and icon
instal -D -m 644 $(CURDIR)/debian/xcdroast.desktop \
 $(CURDIR)/debian/xcdroast/usr/share/applications/xcdroast.desktop
instal -D -m 644 $(CURDIR)/debian/xcdroast.xpm \
 $(CURDIR)/debian/xcdroast/usr/share/pixmaps/xcdroast.xpm
```

Sekarang dalam berkas `dirs`, baris berikut harus ditambahkan agar berkas .desktop dapat diinstal:

```
usr/share/applications
```

Sekarang kita membutuhkan berkas .desktop (simpan sebagai *debian/xcdroast.desktop*). Dari ubuntu.debdiff (atau dari paket source Ubuntu), dapat kita lihat bahwa:

```
[Desktop Entry]
Encoding=UTF-8
Name=X-CD-Roast
Comment=Create a CD
Exec=xcdroast
Icon=xcdroast.xpm
Type=Application
Categories=Application;AudioVideo;
```

Perubahan terakhir yang harus dilakukan adalah didalam berkas `changelog`. Tidak hanya kita perlu menambahkan apa yang telah kita lakukan (penggabungan dengan Debian), tetapi kita juga harus menambahkan entri changelog Ubuntu sebelumnya. Untuk melakukan ini, jalankan dch -i -D dapper dan isi sesuatu untuk entri changelog ini:

```
xcdroast (0.98+0alpha15-3ubuntu1) dapper; urgency=low

  * Resynchronise with Debian.
```

Pastikan untuk mengubah nomor versi ke versi Ubuntu yang benar. Juga tambah:

```
xcdroast (0.98+0alpha15-1.1ubuntu1) breezy; urgency=low

  * Fix and instal existing .desktop file. (Closes Malone #2698)

  -- Captain Packager <packager@coolness.com>  Sat,  1 Oct 2005 19:39:04 -0400
```

diantara entri log 0.98+0alpha15-1.1 dan 0.98+0alpha15-2.

Sekarang Anda dapat membangun dan menguji paket source baru. Ada beberapa cara untuk melakukan hal ini, salah satu contohnya:

```
cd ..
debuild -S
cd ..
sudo pbuilder build xcdroast_0.98+0alpha15-3ubuntu1.dsc
```

Perintah ini akan membuat ulang paket source, menanda-tangani dengan GPG key Anda, dan membuat paket dalam lingkungan pbuilder untuk memastikan agar source dibuat dengan benar. Pastikan untuk selalu menguji paket Anda sebelum mengeluarkan patch. Langkah terakhir adalah membuat debdiff yang dapat dilampirkan pada laporan bug atau memberikannya kepada MOTU yang ada di kanal IRC #ubuntu-motu. Untuk melakukan hal ini, kita akan mendapatkan perbedaan antara paket source Debian unstable dan versi baru Ubuntu:

```
debdiff xcdroast_0.98+0alpha15-3.dsc xcdroast_0.98+0alpha15-3ubuntu1.dsc > \
 xcdroast_0.98+0alpha15-3ubuntu1.debdiff
```

3. Pembuatan Paket untuk Kubuntu

Seperti yang Anda bayangkan, untuk Kubuntu masalah pembuatan paket berhubungan dengan KDE dan Qt.

3.1. Build Dependencies

Program untuk Kubuntu kebanyakan adalah aplikasi KDE. Untuk itu, program tersebut harus Build-Depend pada `kdelibs4-dev`. Karena fokus KDE adalah untuk membuat program saling berhubungan, beberapa program mungkin harus Build-Depend pada aplikasi KDE lainnya, seperti `kdepim-dev`. Pastikan untuk mendapatkan daftar dependency yang diperlukan untuk program Anda.

3.2. Berkas Desktop

KDE mempunyai path khusus. Kebanyakan pengaturan untuk KDE diinstal dalam `/etc/kde3/` atau `/usr/share/apps/`. Penting untuk dicatat bahwa berkas desktop umum KDE harus diarahkan ke `/usr/share/applications/kde/`. Path instalasi untuk berkas desktop harus diperbaiki dulu jika path tersebut tidak sesuai dengan path diatas (kecuali untuk berkas desktop seperti menu service).

KDE desktop files also need specific entries to fit in the KMenu. A minimal desktop file for a KDE program could be something like this:

```
[Desktop Entry]
Encoding=UTF-8
Name=Kfoo
Name[xx]=Kfoo
GenericName=Bar description
Exec=kfoo
Icon=kfoo
Terminal=false
Categories=Qt;KDE;Utility;
```

Perlu dicatat bahwa ruas Categories harus dimulai dengan Qt;KDE;. Ini adalah entri khusus berkas desktop untuk program dan modul KDE yang mengizinkan su memberitahukan program seperti KCModules atau untuk autostart pada saat login.

3.3. Membuat Berkas .pot

Situs web terjemahan Ubuntu, *Rosetta* [https://launchpad.net/rosetta/], saat ini telah mendukung KDE, artinya paket KDE harus mendukung Rosetta dengan membuat berkas templat .pot untuk para penerjemah. Jika Anda menggunakan cdbs di Dapper, paket Anda akan membuat berkas .pot secara otomatis dan silakan lihat dalam direktori po/.

Anda membutuhkan *kdepot patch* [../files/kubuntu_01_kdepot.diff] (atau yang mirip; Hal ini mungkin tidak diterapkan dengan baik tergantung dari umur dari direktori admin).

Jika paket Anda menggunakan debhelper atau cdbs dan menyertakan berkas kde.mk tersendiri, maka Anda harus menambahkan rules sendiri.

Untuk cdbs, tambah baris ini ke debian/rules:

```
common-post-build-arch::
 mkdir -p po
        XGETTEXT=/usr/bin/kde-xgettext sh admin/cvs.sh extract-messages

clean::
        rm -rf po
```

Untuk debhelper, tambahkan baris berikut ke akhir *instal* rule:

```
mkdir -p po
XGETTEXT=/usr/bin/kde-xgettext sh admin/cvs.sh extract-messages
```

Juga untuk debhelper, tambahkan baris berikut ke *clean* rule:

```
rm -f po/*.pot
```

Bab 7. Bug

Satu hal yang selalu Anda hadapi sebagai pembuat paket adalah bug dalam perangkat lunak itu sendiri atau dalam paket Anda. Bug dalam pembuatan paket seringkali mudah dan langsung dapat diperbaiki. Akan tetapi, pembuat paket sering bertindak sebagai kontak original untuk bug perangkat lunak yang dilaporkan oleh pengguna, pembuat paket menerapkan perbaikan sementara dan bertanggung jawab untuk menyampaikan laporan bug dan perbaikan ke pencipta original (upstream) perangkat lunak tersebut.

Ubuntu Bug Squad [https://wiki.ubuntu.com/BugSquad] adalah tim Penjamin Mutu (QA) untuk Ubuntu. Orang-orang yang tergabung dalam tim ini bekerja tanpa lelah untuk membuat Ubuntu menjadi sistem yang lebih baik. Mereka melacak semua bug yang ada di Distribusi Ubuntu dan memastikan bahwa bug mayor terdeteksi oleh para pengembang. Setiap orang dapat bergabung dalam Bug Squad dan ini adalah salah satu pintu masuk bagi orang yang ingin berkontribusi untuk Ubuntu. Bug Squad dapat ditemui di kanal IRC *#ubuntu-bug* di *irc.ubuntu.com*

1. Sistem Pelacakan Bug

Untuk melacak bug (baik perangkat lunak maupun pembuatan paket), kebanyakan distribusi yang ada telah mengembangkan sistem pelacakan bug untuk mengatur pelaporan bug dan memberitahu pengembang paket serta reporter tentang perubahan yang terjadi. Tabel di bawah ini menunjukkan beberapa contoh perkakas di Debian dan Ubuntu untuk pelacakan bug.

Bug Tracking Systems (BTS)

Debian : *http://bug.debian.org*

Ubuntu : *http://launchpad.net/malone/distros/ubuntu*

Bug untuk Paket Tertentu

Debian : http://bug.debian.org/<*packagename*>

Ubuntu : gunakan fitur search pada BTS Ubuntu

Bug untuk Paket Source

Debian : http://bug.debian.org/src:<*packagename*>

Ubuntu : https://launchpad.net/distros/ubuntu/+source/<*packagename*>/+bug

Informasi Paket

Debian : *http://packages.debian.org* or http://packages.qa.debian.org/<*packagename*>

Ubuntu : *http://packages.ubuntu.com* atau
https://launchpad.net/distros/ubuntu/+source/<*packagename*> untuk paket source

2. Tips Untuk Bug

2.1. Paket source yang tepat

Penunjukkan bug di paket membantu pelaporan bug secara langsung kepada pengembang sepertinya akan dapat dibantu. Dengan menjamin bahwa informasi ini akurat, maka Anda meningkatkan peluang bahwa bug akan dibenahi. Sering, tidak jelas paket mana yang mengandung bug, dan dalam kasus ini penting untuk melaporkan bug di Ubuntu. Jika sebuah bug di tunjuk ke sebuah paket yang jelas-jelas salah, dan Anda tidak tahu paket yang benar, ubah itu ke Ubuntu.

Paket yang benar untuk bug dalam kernel Linux adalah linux, dengan mengabaikan paket tertentu yang digunakan (ada banyak paket yang mengandung kernel Linux).

2.2. Menetapkan permasalahan

Jika sebuah bug ditandai sebagai Unconfirmed, akan sangat membantu jika Anda bersedia mereproduksi masalah dan mencatat hasilnya di Malone. Jika Anda dapat mengonfirmasi masalah, Anda dapat mengubah status bug menjadi Confirmed. Namun jika Anda tidak dapat mengonfirmasi masalahnya, informasi yang Anda sampaikan juga berguna sehingga harus disimpan dalam sebuah komentar.

Menyampaikan bug ke upstream

Anda dapat menyampaikan bug ke pencinta perangkat lunak tersebut (upstream), jika

- Anda yakin bug tersebut tidak terjadi karena perubahan yang berhubungan dengan Ubuntu
- perubahan terlalu sulit diperbaiki oleh Anda sendiri atau orang lain pada tim

JIka Anda melakukan hal ini, pastikan untuk menyertakan informasi yang diperlukan, seperti

- bagaimana mereproduksi bug
- versi yang digunakan (versi dari libraries yang dibutuhkan, jika bug menunjuk permasalahan disana)
- siapa yang melaporkannya
- dimana seluruh percakapan dapat ditemukan

Pastikan juga untuk membuat bug watch di Malone untuk bug ini.

2.3. Bagaimana Menangani Permintaan Fitur

Jika Anda merasa bahwa bug yang dilaporkan adalah fitur pesanan yang disalahartikan sebagai laporan bug, sila kenalkan pelapor kepada *specification process* yang kami miliki. Pastikan untuk menyebutkan source daya spesifikasi berikut: FeatureSpecifications, SpecSpec, SpecTemplate, and *http://launchpad.net/specs*

2.4. Bagaimana Menangani Permintaan Dukungan

Jika Anda merasa bahwa bug yang dilaporkan adalah permintaan dukungan yang disalahartikan sebagai pelaporan bug, sila kenalkan pelapor kepada Support Tracker yang kami miliki. Pastikan untuk merujuk ke *http://launchpad.net/support*.

2.5. Bagaimana menangani saran untuk mengubah default

Jika Anda merasa bahwa bug yang dilaporkan sebenarnya adalah saran untuk mengubah sistem baku yang disalahartikan sebagai laporan bug, mohon untuk memindahkan diskusi ke milis atau forum diskusi yang tepat. Jika perubahan ini telah didiskusikan dan ditolak, berikan alasan kepada pengguna dan arahkan ia ke diskusi yang relevan untuk saran/komentar lebih lanjut.

2.6. Menemukan Duplikat

Mencari laporan bug yang terduplikasi adalah kontribusi yang berharga bagi komunitas Bug. Pengguna kadangkala tidak tahu cara memeriksa apakah bug yang sama telah dilaporkan, dan kadangkala mereka tidak peduli. Mengirimkan pesan sederhana SAYA JUGA dan mengumpulkan informasi penting dalam proses membenahi bug.

Ada beberapa pendekatan yang bisa Anda gunakan untuk membantu melalui cara ini. Yang pertama adalah mencari bug yang dilaporkan pada komponen yang sama. Coba juga untuk mengatakan dengan kata lain pada pencarian Anda, dan perhatikan langkah-langkah serta kata-kata yang menjelaskan cara yang diperlukan untuk menghasilkan ulang bug.

Contoh:

- Cara mudah: *DAAP support* [https://launchpad.net/malone/bug/24932] adalah duplikat dari *tolong aktifkan daap* [https://launchpad.net/malone/bug/24860].
- Yang lebih sulit:*plug:spdif on emu10k1 gone after breezy upgrade* [https://launchpad.net/malone/bug/24011] adalah duplikat dari *Muted sound after dist-upgrade from Hoary to Breezy* [https://launchpad.net/malone/bug/21804].

Jika Anda tidak dapat menemukan duplikas di daftar bug terbuka, Anda dapat pula mencarinya di daftar tertutup. Jangan merasa kecewa jika Anda tidak dapat menemukan duplikasi pelaporan bug secara cepat pada awalnya. Seiring berjalannya waktu, Anda akan mengenali laporan yang mencurigakan dan dapat mengindentifikasikan mereka dengan lebih mudah.

Jika Anda menemukan bug yang memiliki judul buruk/tidak dapat dimengerti, katakan judul dengan frase lain sehingga orang-orang dapat mencari bug tersebut lebih cepat.

2.7. Mengingatkan akan adanya Code of Conduct

Perlu dicatat bahwa Code of Conduct juga berlaku di percakapan dalam pelaporan bug. Jika Anda melihat orang yang tidak saling hormat, sila arahkan mereka ke *Ubuntu Code of Conduct* [http://www.ubuntu.com/community/conduct].

2.8. Mengelola Status

Sebagai penyortir bug atau pengembang, status bug adalah perkakas utama untuk mengategorikan bug dan memberikan gambaran umum tentang tingkat perkembangan paket dan perangkat lunak.

Berikut adalah daftar ringkas dan penjelasan dari beragam status yang ada:

- **Unconfirmed:** Bugs mulai dengan status ini. Bugs yang ditandai Unconfirmed adalah bug yang kurang informasi, belum siap, atau belum dikonfirmasi. Banyak dari bug tersebut yang belum di triaged.
- **Needs Info:** Jika Anda harus bertanya kepada pelapor, mohon labeli bug ini dengan "Needs Info". Tugas rutin untuk bug dengan label Needs Info adalah dengan bertanya balik. Jika tidak ada jawaban setelah periode waktu tertentu, tutup laporan tersebut dengan menyertakan pesan "If you have more information on this bug, please reopen."
- **Rejected:** Bugs yang ditandai sebagai Rejected berarti sudah ditutup. Pastikan untuk memeriksa ulang bug sebelum Anda menolaknya.
- **Confirmed:** Bugs yang dikonfirmasi membutuhkan seseorang untuk mengkonfirmasikannya. Tolong jangan mengkonfirmasi bug Anda sendiri.
- **In Progress:** Jika Anda mulai bekerja pada suatu bug, tetapkan status In Progress jadi orang lain tahu seseorang sedang menangani bug itu.
- **Fix Committed:** Bagi proyek upstream ini berarti perbaikan ada di CVS/SVN/bzr atau dilakukan di suatu tempat. Bagi pengelola paket ini berarti perubahan sedang dalam daftar tunggu dan siap untuk segera di-upload (padanan di Bugzilla adalah PENDINGUPLOAD)
- **Fix Released:** Bagi paket upstream ini berarti rilis tarball telah diumumkan dan tersedia untuk umum. Bagi pengelola paket ini berarti perbaikan telah di-upload. Mohon jangan ragu untuk menambahkan changelog sebagai komentar, supaya orang tahu perubahan apa yang mempengaruhi bug mereka.

2.9. Mengelola tingkat kepentingan

Launchpad menggunakan panduan untuk menetapkan tingkat kepentingan:

- **Untriaged:** laporan bug belum diperiksa. Ini adalah tingkat kepentingan baku untuk bug yang baru dilaporkan.
- **Wishlist:** permintaan untuk menambah fitur baru dalam sebuah program di Ubuntu. Gunakan ini untuk bug yang sebenarnya bukan bug tapi ide untuk fitur baru yang belum ada.
- **Low:** bug yang mempengaharui kemampuan aplikasi, tetapi lebih sedikit luas dibanding kebanyakan bug
- **Medium:**bug pada kemampuan aplikasi dari variasi standar. Kebanyakan bug memiliki tingkat kerawanan "Medium".
- **High:** bug yang memiliki tingkat kerawanan tinggi di sebagian kecil pengguna Ubuntu (kira-kira) atau memiliki tingkat kerawanan sedang di sebagian besar pengguna Ubuntu (kira-kira)

- **Critical:** sebuah bug yang memiliki tingkat kerawanan paling tinggi di pengguna Ubuntu

Lampiran A. Lampiran

1. Source Tambahan

Source Debian

- *Panduan Pengelola Baru Debian* [http://www.debian.org/doc/manuals/maint-guide/] - Source yang baik untuk belajar tentang paket.
- *Kebijakan Debian* [http://www.debian.org/doc/debian-policy/] - Manual Kebijakan pokok untuk Debian dan distro berbasis Debian.
- *Debian Developer's Reference* [http://www.debian.org/doc/manuals/developers-reference/] - Informasi khusus untuk Pengembang Debian tetapi ada bagian yang berguna untuk pembuat paket.
- *Library Packaging Guide* [http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html] - Pantuan pembuatan paket untuk libraries.
- *Debian Women Packaging Tutorial* [http://women.alioth.debian.org/wiki/index.php/English/PackagingTutorial] - Pengenalan lain akan pembuatan paket Debian.

Source Lain

- *Tutorial Pembuatan Paket IBM* [http://www-106.ibm.com/developerworks/linux/library/l-debpkg.html]
- *Dokumentasi Duckcorp CDBS* [https://perso.duckcorp.org/duck/cdbs-doc/cdbs-doc.xhtml]
- *Dokumentasi MOTU Ubuntu* [https://wiki.ubuntu.com/MOTU/Documentation]
- *Panduan Pembuatan Paket Kubuntu* [https://wiki.ubuntu.com/KubuntuPackagingGuide]

2. Lingkungan Chroot

Lingkungan chroot umumnya digunakan untuk pekerjaan yang berhubungan dengan pengembangan dan pada dasarnya adalah perangkat lunak yang berhubungan dengan pembangunan. Ide yang bagus untuk selalu melakukan pekerjaan pengembangan di lingkungan chroot, karena sering membutuhkan instalasi paket pengembangan (dimana tujuan utamanya adalah untuk pembangunan paket). Contohnya adalah ketika sebuah aplikasi membutuhkan header dan versi pengembangan dari sebuah pustaka untuk membangun (misalnya libabc-dev). Pengguna biasa tidak membutuhkan versi pengembangan dari libabc. Maka akan lebih baik bila menginstal paket pengembangan seperti itu dalam chroot, membiarkan lingkungan operasi normal bersih dan tidak berantakan. Pertama, kita perlu mengsintal paket yang dibutuhkan:

```
sudo apt-get instal dchroot debootstrap
```

Pastikan untuk menginstal versi terakhir dari debootstrap yang berasal dari rilis Ubuntu dimana Anda sedang mencoba membuat chroot sekarang. Anda perlu men-download-nya dari *packages.ubuntu.com* [http://packages.ubuntu.com] dan menginstal secara manual dengan perintah dpkg -i.

Langkah berikut adalah untuk membuat, mengkonfigurasi dan memasuki lingkungan chroot.

```
sudo mkdir /var/chroot
echo "mychroot /var/chroot" | sudo tee -a /etc/dchroot.conf
sudo debootstrap --variant=buildd edgy /var/chroot/ http://archive.ubuntu.com/ubuntu/
```

Membuat lingkungan chroot memerlukan waktu beberapa saat karena debootstrap akan men-download dan mengkonfigurasi instalasi Ubuntu minimal.

```
sudo cp /etc/resolv.conf /var/chroot/etc/resolv.conf
sudo cp /etc/apt/sources.list /var/chroot/etc/apt/
sudo chroot /var/chroot/
```

Agar dapat menggunakan apt dalam chroot, tambah source Ubuntu ke source apt chroot. Untuk sementara, abaikan setiap peringatan mengenai autentifikasi paket:

```
echo "deb     http://archive.ubuntu.com/ubuntu edgy main restricted \
 universe multiverse" > /etc/apt/sources.list
echo "deb-src http://archive.ubuntu.com/ubuntu edgy main restricted \
 universe multiverse" >> /etc/apt/sources.list
apt-get update
apt-get instal build-essential dh-make automake pbuilder gnupg lintian \
 wget debconf devscripts gnupg sudo
apt-get update
exit
```

Jalankan perintah berikut untuk mengkonfigurasi locales:

```
sudo chroot /var/chroot/
```

```
apt-get instal dialog language-pack-en
exit
```

Jika Anda ingin dukungan bahasa selain dari Inggris ganti *en* dalam language-pack-en dengan kode bahasa yang diinginkan.

Lalu, perbaiki kata sandi pengguna dan root untuk lingkungan chroot. Baris akhir di bawah adalah untuk menghindari peringatan sudo saat resolving dalam lingkungan chroot:

```
sudo cp /etc/passwd /var/chroot/etc/
sudo sed 's/\([^:]*\):[^:]*:/\1:*:/' /etc/shadow | sudo tee /var/chroot/etc/shadow
sudo cp /etc/group /var/chroot/etc/
sudo cp /etc/hosts /var/chroot/etc/
```

Untuk mengaktifkan sudo, set up kata sandi root Anda dan pengguna pertama sudo dalam grup admin (untuk lingkungan chroot). Dalam perintah berikut, ganti "<user>" dengan nama pengguna yang akan digunakan dalam lingkungan chroot:

```
sudo cp /etc/sudoers /var/chroot/etc/
sudo chroot /var/chroot/
dpkg-reconfigure passwd
passwd <user>
exit
```

Berkas fstab harus dimodifikasi agar lingkungan chroot dapat mengakses direktori home sistem, direktori temp, dll. Catatan bahwa direktori home sistem yang sesungguhnya digunakan dalam lingkungan chroot.

```
sudo editor /etc/fstab
```

Tambahkan baris ini:

```
/home           /var/chroot/home        none    bind            0       0
/tmp            /var/chroot/tmp         none    bind            0       0
proc-chroot     /var/chroot/proc        proc    defaults        0       0
devpts-chroot   /var/chroot/dev/pts     devpts  defaults        0       0
```

Mount entri baru fstab

```
sudo mount -a
```

Profil standar bash menyertakan informasi chroot dalam prompt. Untuk mengaktifkannya:

```
sudo chroot /var/chroot/
echo mychroot > /etc/debian_chroot
exit
```

Sekarang gunakan chroot Anda (Anda dapat menghilangkan opsi -c mychroot jika hanya ada satu entri atau Anda hanya ingin mengunakan baris pertama dalam `/etc/dchroot.conf`). Parameter -d

artinya lingkungan chroot sudah di-preserved. Parameter ini biasanya berguna jika Anda ingin chroot aplikasi yang menggunakan server X, manajer sesi Anda, dll.

```
dchroot -c mychroot -d
```

3. Berkas contoh dh_make

Readme.Debian

Berkas ini digunakan untuk mendokumentasikan perubahan yang Anda lakukan pada source original upstream agar orang lain mengetahui informasi perubahan yang dilakukan pada Debian atau Ubuntu.

conffiles.ex

Jika paket menginstal berkas konfigurasi, ketika paket di-upgrade dpkg akan menanyakan pengguna apakah ingin menyimpan versi Anda jika telah dimodifikasi atau menginstal versi baru. Berkas konfigurasi seperti ini harus didaftarkan dalam `conffiles` (satu entri per baris). Jangan daftarkan berkas konfigurasi yang hanya dimodifikasi oleh paket atau memerlukan user untuk men-setup agar dapat bekerja.

cron.d.ex

Jika paket Anda membutuhkan tugas berjadwal agar dapat bekerja dengan baik, Anda dapat menggunakan berkas ini mengkonfigurasikannya. Jika Anda menggunakan berkas ini, silakan ganti nama menjadi `cron.d`.

dirs

Berkas ini menetapkan direktori yang dibutuhkan tetapi prosedur instalasi normal (make instalapplication) tidak membuatnya.

docs

Berkas ini menetapkan nama dari berkas dokumentasi yang dh_instaldocs akan instal ke direktori sementara.

emacsen-*.ex

Berkas ini menetapkan berkas Emacs yang akan di-bytecompile saat waktu instal. Berkas ini diinstal ke direktori sementara oleh dh_instalemacsen.

init.d.ex

Jika paket Anda adalah sebuat daemon yang harus dijalankan saat startup sistem silakan ganti berkas ini ke `init.d` dan sesuaikan dengan kebutuhan Anda.

manpage.1.ex dan manpage.sgml.ex

Berkas-berkas ini adalah templat untuk halaman man jika paket tersebut tidak memilikinya.

menu.ex

Berkas ini digunakan untuk menambah paket Anda ke menu Debian. Ubuntu tidak menggunakan berkas menu Debian tetapi menggunakan berkas *.desktop* [http://standards.freedesktop.org/desktop-entry-spec/latest/] menurut standar *freedesktop.org* [http://www.freedesktop.org].

watch.ex

Pengelola paket dapat menggunakan program uscan dan berkas `watch` untuk memeriksa tarball source baru dari upstream.

ex.package.doc-base

Berkas ini digunakan untuk meregistrasi dokumentasi paket Anda (selain dari halaman man dan info) dengan doc-base.

postinst.ex, preinst.ex, postrm.ex, dan prerm.ex

Skrip pengelola ini dijalankan oleh dpkg saat paket diinstal, upgrade, atau dihapus.

Untuk penjelasan lebih lanjut silakan lihat *Debian New Maintainer's Guide* [http://www.debian.org/doc/maint-guide/ch-dother.en.html].

4. Daftar dari skrip debhelper

- dh_builddeb
- dh_clean
- dh_compress
- dh_desktop
- dh_fixperms
- dh_gconf
- dh_gencontrol
- dh_iconcache
- dh_instal
- dh_instalcatalogs
- dh_instalchangelogs
- dh_instalcron
- dh_instaldeb
- dh_instaldebconf
- dh_instaldefoma
- dh_instaldirs
- dh_instaldocs
- dh_instalemacsen
- dh_instalexamples
- dh_instalinfo
- dh_instalinit
- dh_installogcheck
- dh_installogrotate
- dh_instalman
- dh_instalmenu
- dh_instalmime
- dh_instalmodules
- dh_instalpam
- dh_instalppp
- dh_instaltexfonts
- dh_instalwm
- dh_instalxfonts
- dh_instalxmlcatalogs
- dh_link

- dh_listpackages
- dh_makeshlibs
- dh_md5sums
- dh_perl
- dh_python
- dh_scrollkeeper
- dh_shlibdeps
- dh_strip
- dh_testdir
- dh_testroot
- dh_usrlocal

Lampiran B. GNU General Public License

Version 2, June 1991

Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor,
Boston, MA 02110-1301
USA

Version 2, June 1991

1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

2.1. Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2.2. Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2.3. Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of *Section 1* above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under

these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

2.4. Section 3

You may copy and distribute the Program (or a work based on it, under *Section 2* in object code or executable form under the terms of *Sections 1* and *2* above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

2.5. Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

2.6. Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

2.7. Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

2.8. Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

2.9. Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

2.10. Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

2.11. Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

2.12. NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND,

EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

2.13. Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

3. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.