Custom Debian Distributions

Andreas Tille <tille@debian.org>

5 November 2008

Abstract

This paper is intended for people who are interested in the philosophy of Custom Debian Distributions, and the technique that is used to manage those projects. It is explained in detail why these are not forks from Debian, but reside completely inside the Debian GNU/Linux distribution, and which advantages can be enjoyed by taking this approach. The concept of metapackages and user role based menus is explained. In short: This document describes why Custom Debian Distributions are important to the vitality and quality of Debian.

Copyright Notice

You can find the source of this article in the Subversion repository at svn.debian.org (`http://svn.debian.org/wsvn/cdd/cdd/trunk/cdd/doc/`). It is also available as Debian package `cdd-doc`.

A printable version in PDF format (`../debian-cdd.en.pdf`) will be built from time to time.

Contents

Chapter 1

Introduction

A general purpose operating system like Debian can be the perfect solution for many different problems. Whether you want Debian to work for you in the classroom, as a games machine, or in the office, each problem area has its own unique needs and requires a different subset of packages tailored in a different way.

Custom Debian Distributions (formerly merged together with Debian Internal Projects) provide support for special user interests. They implement a new approach to cover interests of specialised users, who might be children, lawyers, medical staff, visually impaired people, etc. Of late, several Custom Debian Distributions have evolved. The common goal of those is to make installation and administration of computers for their target users as easy as possible, and to serve in the role as the missing link between software developers and users well.

Using the object oriented approach as an analogy, if Debian as a whole is an object, a Custom Debian Distribution is an instance of this object that inherits all features while providing certain properties.

So the Debian project releases the Debian Distribution and other Custom Debian Distributions. In contrast to this, there might be some other Debian related Projects, either external or non-official, which may create "derivative distributions". But these are not the responsibility of the Debian project.

The effort might fall into the same category as the Componentized Linux (http://en.wikipedia.org/wiki/Progeny_Componentized_Linux) of Progeny, but there are certain differences that will be outlined in this paper.

Chapter 2

What are Custom Debian Distributions?

2.1 What is Debian?

The core of an operating system is a piece of software that interacts with the hardware of the computer, and provides basic functionality for several applications. On Linux based systems, the so-called kernel provides this functionality, and the term Linux just means this core without those applications that provide the functionality for users. Other examples are the Hurd, or the flavours of the BSD kernel.

Many applications around UNIX-like kernels are provided by the GNU (http://www.gnu.org/) system. That is why Linux based operating systems are described as GNU/Linux systems. The GNU tools around the Linux kernel build a complete operating system.

Users do not need only an operating system. They also need certain applications like web servers, or office suites. A *distribution* is a collection of software packages around the GNU/Linux operating system that satisfies the needs of the target user group. There are general distributions, which try to support all users, and there are several specialised distributions, which each target a special group of users.

Distributors are those companies that are building these collections of software around the GNU/Linux operating system. Because it is Free Software, the user who buys a distribution pays for the service that the distributor is providing. These services might be:

- Preparing a useful collection of software around GNU/Linux.
- Caring for smooth installation that the target user is able to manage.
- Providing software updates and security fixes.
- Writing documentation and translations to enable the user to use the distribution with maximum effect.

- Selling Boxes with ready to install CDs and printed documentation.
- Offering training and qualification.

Most distributors ship their distribution in binary packages. Two package formats are widely used:

RPM (RedHat Package Manager) which is supported by RedHat, SuSE, Mandrake and others.

DEB (Debian Package) used by Debian and derived distributions.

All GNU/Linux distributions have a certain amount of common ground, and the Linux Standard Base (http://www.linuxbase.org/) (LSB) is attempting to develop and promote a set of standards that will increase compatibility among Linux distributions, and enable software applications to run on any compliant system.

The very essence of any distribution, (whether delivered as RPMs, DEBs, Source tarballs or ports) is the choice of *policy statements* made (or not made, as the case may be) by the creators of the distribution.

Policy statements in this sense are things like "configuration files live in `/etc/$package/$package.conf`, logfiles go to `/var/log/$package/$package.log` and the documentation files can be found in `/usr/share/doc/$package`."

The policy statements are followed by the tool-chains and libraries used to build the software, and the lists of dependencies, which dictate the prerequisites and order in which the software has to be built and installed. (It's easier to ride a bicycle if you put the wheels on first. ;-))

It is this *adherence to policy* that causes a distribution to remain consistent within its own bounds. At the same time, this is the reason why packages can not always be safely installed across distribution boundaries. A SuSE `package.rpm` might not play well with a RedHat `package.rpm`, although the packages work perfectly well within their own distributions. A similar compatability problem could also apply to packages from the same distributor, but from a different version or generation of the distribution.

As you will see later in more detail, Custom Debian Distributions are just a modified ruleset for producing a modified (specialised) version of Debian GNU/Linux.

A package management system is a very strong tool to manage software packages on your computer. A large amount of the work of a distributor is building these software packages.

Distributors you might know are Mandrake (http://www.mandrakelinux.com/), RedHat (http://www.redhat.com/), SuSE (http://www.suse.com/) (now owned by Novell (http://www.novell.com/linux/)) and others.

Debian (http://www.debian.org) is just one of them.

Well, at least this is what people who do not know Debian well might think about it. But, in fact, Debian is a different kind of distribution ...

2.2 What is Debian? (next try)

The Debian Project is an association of individuals who have made common cause to create a free operating system. This operating system that we have created is called **Debian GNU/Linux**, or simply Debian for short.

Moreover, work is in progress to provide Debian of kernels other than Linux, primarily for the Hurd. Other possible kernels are the flavours of BSD, and there are even people who think about ports to MS Windows.

All members of the Debian project are connected in a web of trust (http://people.debian.org/~edward/globe/earthkeyring/), which is woven by signing GPG keys. One requirement to become a member of the Debian project is to have a GPG key signed by a Debian developer. Every time one Debian developer meets another developer for the first time, they sign each other's keys. In this way, the web of trust is woven.

2.3 Differences from other distributions

- Debian is not a company, but an organisation.
- It does not sell anything.
- Debian members are volunteers.
- Maintainers are working on the common goal: to build the best operating system they can achieve.
- Debian maintains the largest collection of ready-to-install Free Software on the Internet.
- There are two ways to obtain Debian GNU/Linux:

  1 Buy it from some *other* distributor on CD. Perhaps the correct term would be *re*distributor. Because Debian is free, anybody can build his own distribution based on it, sell CDs, and even add new features, such as printed documentation, more software, support for different installers and more.

  2 Download Debian from the web for free.

  The latter is the common way, and there are really great tools to do it this way. Certainly it is always possible to copy Debian from a friend.

2.4 Custom Debian Distributions

Debian contains nearly 10000 binary packages, and this number is constantly increasing. There is no single user who needs all these packages (even if conflicting packages are not considered).

The normal user is interested in a subset of these packages. But how does the user find out which packages are really interesting?

One solution is provided by the `tasksel` package. It provides a reasonable selection of quite general tasks that can be accomplished using a set of packages installed on a Debian GNU/Linux system. But this is not really fine grained, and does not address all of the needs of user groups with special interests.

Custom Debian Distributions (formerly known as Debian Internal Projects) try to provide a solution for *special groups of target users with different skills and interests*. Not only do they provide handy collections of specific program packages, but they also ease installation and configuration for the intended purpose.

To clarify a common misunderstanding: Custom Debian Distributions are **not forks** from Debian. They are completely included, and if you obtain the complete Debian GNU/Linux distribution, you have all available Custom Debian Distributions included.

Chapter 3

General ideas

3.1 Looking beyond

Commercial Linux distributors sell certain products that try to address special user needs.

Enterprise solutions
- Corporate Server - Mandrake
- Advanced Server - RedHat
- Enterprise Server - SuSE

Small Office and Home Office (SOHO) There are a couple of workstation or home editions, as well as office desktops built by several GNU/Linux distributors.

Special task products **Mail server** SuSE Linux Openexchange Server

Firewall Multi Network Firewall - Mandrake, SuSE Firewall on CD, ...

Cluster Mandrake Clustering

Content Management System RedHat

Portal Server RedHat

This is only a small set of examples of commercial GNU/Linux distributors addressing specific user interests with certain products.

Debian solves this problem with **Custom Debian Distributions**.

3.2 Motivation

3.2.1 Profile of target users

The target user of a Custom Debian Distribution may be a specialist of a certain profession, (e.g. a doctor or lawyer,) a person who has not (yet) gathered a certain amount of computer

knowledge, (e.g. a child,) or a person with disabilities (e.g. a visually or hearing impaired person.) Moreover, the customisation might deal with peculiarities of certain regions where users have needs that differ from Debian as a whole.

It is not unusual for these target users to be less technically competent than the stereotypical Linux user. These people are often not interested in the computer for its own sake, but just want it to work for them. Imagine the frustration of a doctor who has to move the focus of interest from the patient to his stupid computer that does not work as expected.

Because of limited knowledge or time, the target user is usually unable to install upstream programs. This means that in the first place, they must find out which software packages in their distribution might serve for a certain problem. The next step would be to download and install the packages they choose, perhaps requiring a certain amount of configuration effort. This problem is nearly impossible for a user with limited technical competence and perhaps poor English language comprehension, which prevents the user from understanding the installation manual.

The language barrier in this field is an important issue, because we are targeting everyday users who are not compelled to learn English, like Free Software developers are, for everyday communication. So the installation process has to involve the least possible user interaction, and any such interaction has to be internationalised.

Furthermore, most target users have no or little interest in administration of their computer. In short, the optimal situation would be that he would not even notice the existence of the computer, but just focus on using the application to accomplish the task at hand.

Common to all groups of target users is their interest in a defined subset of available Free Software. None of them would like to spend much time searching for the package that fits his interest. Instead, the target user would prefer to immediately and effortlessly locate and access all material relevant to solving his own problems.

There is an absolute need for easy usage of the programs. This is not to say users expect to not have to learn to use the software. Adults generally accept that they must spend a reasonable amount of time in learning how to use a piece of software before they can do something useful and productive with it. But a simple-to-learn environment greatly enhances the value of the software, and if you consider children as target users, they just want to start using it right away without reading any documentation.

The more important part of the request for easy usage is a professional design that is functional and effective. To accomplish this, the programmers need expert knowledge, or at least a quick communication channel to experts to learn more about their requirements. One task for Custom Debian Distributions is to bring programmers and experts who will use those special programs together.

Last, but not least, we find certain requirements beyond just which packages are provided in each target user group. These may differ between different Custom Debian Distributions. For instance, while a doctor has to protect his database against snooping by outside attackers, the privacy risk for a child's system are of lesser importance. Thus, the Debian Junior project cares more for ensuring that the user himself does not damage the desktop environment while playing around with it than about remote attacks. So we find a "defined security profile" for

each single Custom Debian Distribution.

3.2.2 Profile of target administrators

In the field that should be covered by Custom Debian Distributions, we have to face also some common problems for system administrators. Often they have limited time in which they must serve quite a number of computers, and thus they are happy about each simplification of the administration process. The time required to make special adaptations for the intended purpose has to be reduced to a minimum.

So, administrators are looking for timesaving in repetitive tasks. While this is a common issue for each general GNU/Linux distribution, this could have certain consequences in the special fields Custom Debian Distributions want to address.

Another problem administrators face is that they are often not experts in their clients' special field of work. Thus, they may need some specialist knowledge to explain the use of special programs to their users, or at least need to be able to communicate well with the experts about their special needs, and how the software can be used to address them.

3.3 Status of specialised Free Software

Programs like a web server, or a mail user agent are used by many different users. That is why many gifted programmers feel obliged for this kind of Free Software - they just need it for their own. So you normally find a fast, growing community around Free Software packages that have a wide use. This is different for specialised software.

In this context, the term "specialised software" refers to the kind of software that is needed by some experts for their job. This might be a practice management system that is used by doctors, a graphical information system (GIS) that is used by geographers, a screen reader that helps blind people to work with the computer, etc. The difference between such software and widely used software like office suites is that the user base is relatively small. This is also true for certain software that supports special localisation issues.

- Specialist software is used only by a limited set of users (i.e. the specialists). There exists a set of software tools that work perfectly in the environment where they were developed. If the developers catch the idea of Free Software, and just release this software as-is, people in the new, broader user community often run into trouble getting it to work in their environment. This happens because the developers did not really care about a robust installation process that works outside their special environment. As well, installation instructions are often badly written, if they exist at all. But these problem can be easily solved by shipping the software as policy-compliant binary packages, which not only ease installation, but also require documentation to be included. Thus, mere inclusion in Debian benefits the whole user base of any specialised software.
- The trouble often continues in the maintenance of the installed software.

- When it comes to the usage of the specialist software, it often happens that it perfectly fits the needs of the developer who wrote it for his own purposes, and who is familiar with its quirks, but in many cases such software does not comply with ergonomic standards of user interfaces.
- Several existing programs that might be useful for specialists are not really free in the sense of the Debian Free Software Guidelines (DFSG) (http://www.debian.org/social_contract#guidelines). Programs that are incompatible with the DFSG cannot be included in Debian. This is possibly a drawback for those programs, because they could profit by spreading widely on the back of Debian over the whole world.
- A certain number of programs are developed at universities by students or graduates. Once these people leave the university, the programs they developed might be orphaned; *i.e.*, not actively maintained anymore. If their licenses are too restrictive, it may be impossible for anyone else to take over; sticking to DFSG (http://www.debian.org/social_contract#guidelines)-free licenses avoids that problem.
- In special fields, often "typical" (not necessarily Intel-based) hardware architectures are used. Debian currently runs on 11 different architectures, and automatic build servers normally compile software packages as necessary. If auto-builders for other architectures show problems, Debian maintainers will normally fix them, and send the original authors a patch. Moreover, users can report run-time problems via the Debian Bug Tracking System (http://www.debian.org/Bugs/).
- Many programs that are written from scratch use their own non-standard file formats. However, it is often important for programs to be able to share data with each other.
- Often there are several programs that try to solve identical or similar problems. The paper about Debian-Med (http://people.debian.org/~tille/debian-med/talks/paper/debian-med.html) illustrates this in detail for the problem of medical practice management. Normally, all these programs take very interesting approaches but all of them have certain drawbacks. So, joining programmers' forces might make sense here.
- Sometimes the tools or back-ends used in Free Software are not appropriate for such applications. For instance, sometimes database servers that do not use transactions are used to store medical records, which is completely unacceptable. Other programs use web clients as their front-end, which is not really good for quick (mouse-less) usage, a great shortcoming for repetitive tasks.

3.4 General problem

Free Software development is a kind of evolutionary process. It needs a critical mass of supporters, who are:

- programmers *and*

- users

Because specialised software has a limited set of users, (specialists,) this results in a limited set of programmers.

Debian wants to attract both groups to get it working.

Debian is the missing link between upstream developers and users.

3.5 Custom Debian Distributions from philosophical point of view

Debian currently grows in several directions:

- Number of involved people
- Number of packages
- Number of architectures
- Number of bugs
- Number of users
- Number of derivatives
- Time span between releases

So several features are changing at different rates their quantity. According to Hegel a change of quantity leads into a change in quality. That means that Debian will change at a certain point in time (or over a certain time span) its quality.

"To determine at the right moment the critical point where quantity changes into quality is one of the most important and difficult tasks in all the spheres of knowledge." (Trotzki) This might mean that we just passed the point in time when Debian changed its quality. At one point we even observed a change once the package pool system was implemented to cope with the increased number of packages while trying to reduce the time span between releases. Even if the plan to increase the frequencies of releases failed Debian became a new quality. People started using the `testing` distribution even in production which was not really intended and in a consequence even security in `testing` was implemented for Sarge.

According to Darwin evolution happens through quantitative transformations passing into qualitative. So Debian has to evolve and to cope with the inner changes and outer requirements to survive in the Linux distribution environment.

Chapter 4

Existing Custom Debian Distributions

4.1 Debian Junior: Debian for children from 1 to 99

Start beginning of 2000

URL Debian Jr. (http://www.debian.org/devel/debian-jr)

Mailing list debian-jr@lists.debian.org (http://lists.debian.org/debian-jr/)

Initiator Ben Armstrong <synrg@debian.org>

Release Debian 3.0 (Woody)

Goals
- To make Debian an OS that children of all ages will *want* to use, preferring it over the alternatives.
- To care for those applications in Debian suitable for children, and ensure their quality, to the best of our abilities.
- To make Debian a playground for children's enjoyment and exploration.

The main target is young children. By the time children are teenaged, they should be comfortable with using Debian without any special modifications.

Debian Jr. was the first Custom Debian Distribution. In fact, at the time this project was created, the idea of Custom Debian Distributions was born, although then, we used the term "internal project". Over time, this name was changed because it was too broad, as it was equally descriptive of a number of quite different projects, such as IPv6 and QA.

Debian Jr. not only provides games, but is also concerned about their quality from a child's perspective. Thus, games that are regarded as not well suited to young children are omitted. Moreover, choices are made about which packages are best suited for children to use for various other activities and tasks that interest them. This includes, for example, simple text processing, web browsing and drawing.

4.2 Debian-Med: Debian in Health Care

Start beginning of 2002 (http://lists.debian.org/debian-devel/2002/01/msg01730.html)

URL Debian-Med (http://www.debian.org/devel/debian-med)

Mailing list debian-med@lists.debian.org (http://lists.debian.org/debian-med/)

Initiator Andreas Tille <tille@debian.org>

Release Sarge

Goals
- To build an integrated software environment for all medical tasks.
- To care especially for the quality of program packages in the field of medicine that are already integrated within Debian.
- To build and include in Debian packages of medical software that are missing in Debian.
- To care for a general infrastructure for medical users.
- To make efforts to increase the quality of third party Free Software in the field of medicine.

4.3 Debian Edu: Debian for Education

Start Summer of 2002, since 2003 merged with SkoleLinux, which is now synonymous with Debian Edu

URL Debian Edu Wiki (http://wiki.debian.org/DebianEdu)

Mailing list debian-edu@lists.debian.org (http://lists.debian.org/debian-edu/)

Former Initiator Raphaël Hertzog <hertzog@debian.org>

Now responsible Petter Reinholdtsen <pere@hungry.com>

Release Sarge

Goals
- To make Debian the best distribution available for educational use.
- Provide a ready to run classroom installation with free educational software. An automatically installed server provides net-boot services for disk-less thin clients and all necessary applications for educational use.
- To federate many initiatives around education, which are partly based on forks of Debian.
- To continue the internationalisation efforts of SkoleLinux.
- To focus on easy installation in schools.

- To cooperate with other education-related projects (like Schoolforge (http://schoolforge.net/), Ofset (http://www.ofset.org/), KdeEdu (http://edu.kde.org/)).

This project started with the intention to bring back into Debian a fork from Debian that was started by some people in France. Because they had some time constraints, the people who initially started this effort handed over responsibility to the Norwegian Skolelinux (http://www.skolelinux.org), which is currently more or less identical to Debian Edu.

The Debian Edu project gathered special interest in Spain because there are derived Debian distributions from this country that are intended to be used in schools. For instance there are:

LinEX (http://www.linex.org/) A Debian derivative distribution used in all schools in Extremadura.

Currently a fruitful cooperation between Debian Edu and LinEX is established.

LliureX (http://www.lliurex.net/) A Debian derivative distribution in development to be used in schools in Valencia. The goal is to integrate as much as possible as a Custom Debian Distribution.

Guadalinex (http://www.guadalinex.org/) This distribution is not only related to education, but might try also to integrate what they have produced back into Debian.

4.4 DeMuDi: Debian Multimedia Distribution

Start Currently not announced as an official Custom Debian Distribution, but intends to integrate back. DeMuDi is part of the Agnula (http://www.agnula.org/) project (founded by European Community) (since 2000).

URL Demudi (http://www.agnula.org/)

Initiator Marco Trevisani <marco@centrotemporeale.it>

Goals

- To bring back this fork into Debian.
- Oriented toward music and multimedia.
- To make GNU/Linux a platform of choice for the musician and the multimedia artist.

The initiator is not yet a Debian developer, but it is possible to work on Debian without being an official developer.

4.5 Debian-GIS: Geographical Information Systems

Start October 2004 (http://lists.debian.org/debian-devel-announce/2004/10/msg00007.html)

URL DebianGIS Wiki (http://wiki.debian.org/DebianGis)

Mailing list user and developer list (http://wiki.debian.org/DebianGis/MailingLists)

Initiator Francesco P. Lovergine <frankie@debian.org>

4.6 DebiChem: Debian for Chemistry

Start October 2004

URL Debichem Alioth page (http://alioth.debian.org/projects/debichem/)

Mailing list debichem-users@lists.alioth.debian.org (http://lists.alioth.debian.org/mailman/listinfo/debichem-users)

Initiator Michael Banck <mbanck@debian.org>

4.7 Debian-Science: Debian for science

Start July 2005 (http://lists.debian.org/debian-devel/2005/07/msg01555.html)

URL Debian-Science Wiki (http://wiki.debian.org/DebianScience)

Mailing list debian-science@lists.debian.org (http://lists.debian.org/debian-science/)

Initiator Helen Faulkner <helen@debian.org>

While there are Custom Debian Distributions that care for certain sciences (Debian-Med deals in a main part with Biology, DebiChem for Chemistry and Debian-GIS for geography) not all sciences are covered by a specific CDD. The main reason is that at the moment not enough people support such an effort for every science. The temporary solution was to build a general Debian-Science CDD that makes use of the work of other CDDs in case it exists.

4.8 CDDs that were announced but development is stalled

4.8.1 Debian-Desktop: Debian GNU/Linux for everybody

Motto: "Software that Just Works".

Start October 2002

URL Debian-Desktop (http://www.debian.org/devel/debian-desktop)

Mailing list debian-desktop@lists.debian.org (http://lists.debian.org/debian-desktop/)

Initiator Colin Walters <walters@debian.org>

Goals
- To try to build the best possible operating system for home and corporate workstation use.
- To ensure desktops like GNOME and KDE coexist well in Debian and work optimally.
- To balance ease of use for beginners with flexibility for experts.
- To make the system easy to install and configure (e.g. via hardware-detection).

This Custom Debian Distribution has many common issues with other Custom Distributions. The latest move of Debian-Desktop was to care about more up to date software that can be used as common base for all Custom Debian Distributions. The common interest is described in detail in 'New way to distribute Debian' on page 56.

4.8.2 Debian-Lex: Debian GNU/Linux for Lawyers

Start April 2003

URL Debian-Lex (http://www.debian.org/devel/debian-lex)

Mailing list debian-lex@lists.debian.org (http://lists.debian.org/debian-lex/)

Initiator Jeremy Malcolm <Jeremy@Malcolm.id.au>

Goals
- To build a complete system for all tasks in legal practice.
- To add value by providing customised templates for lawyers to existing packages like OpenOffice.org and SQL-Ledger, and sample database schemas for PostgreSQL.

The word *lex* is the Latin word for law.

4.8.3 Debian Accessibility Project

Debian for blind and visually impaired people

Start February 2003

Mailing list debian-accessibility@lists.debian.org (http://lists.debian.org/debian-accessibility/)

URL Debian-Accessibility (http://www.debian.org/devel/debian-accessibility/)

Initiator Mario Lang <mlang@debian.org>

Goals
- To make Debian accessible to people with disabilities.
- To take special care for: Screen readers; Screen magnification programs; Software speech synthesisers; Speech recognition software; Scanner drivers and OCR software; Specialised software like edbrowse (web-browse in the spirit of line-editors)
- To make text-mode interfaces available.
- To provide screen reader functionality during installation.

4.8.4 Debian Enterprise

Debian GNU/Linux for Enterprise Computing

Start End of 2003

URL Debian-Enterprise

Initiator Zenaan Harkness <zen@iptaustralia.net>

Goals
- To apply the UserLinux Manifesto.
- To establish the benchmark in world class Enterprise operating systems engineered within an industry driven shared-cost development model.
- To vigorously defend its distinctive trademarks and branding.
- To develop extensive and professional quality documentation.
- To provide engineer certification through partner organisations.
- To certify the Debian Enterprise GNU/Linux operating system to specific industry standards.
- To pre-configure server tasks

4.8.5 Other possible Custom Debian Distributions

There are fields that could be served nicely by not yet existing Custom Debian Distributions:

Debian-eGov Could address government issues, administration, offices of authorities, accounting.

Office Could cover all office issues.

Accounting Could integrate accounting systems into Debian.

Biology Could perhaps take over some stuff from Debian-Med.

Physics Might look after simulation software.

Mathematics There is even already a live CD - see Quantian in 'Building Live CDs of each Custom Debian Distribution' on page 55

??? There are a lot more potential Custom Debian Distributions.

Chapter 5

Distributions inside Debian

5.1 To fork or not to fork

There are many distributions that decided to fork from a certain state of Debian. This is perfectly all right because Debian is completely free and everybody is allowed to do this. People who built those derived distributions had certain reasons to proceed this way.

5.1.1 Commercial forks

If Debian should be used as the base for a commercial distribution like Linspire (http://www.linspire.com/) (formerly Lindows), Libranet (http://www.libranet.com/) or Xandros (http://www.xandros.com/), there is no other choice than forking because these companies normally add some stuff that is non-free. While Custom Debian Distributions might be interesting in technical terms for those commercial distributions by making it easier to build a separate distribution, these non-free additions are not allowed to be integrated into Debian, and thus integration into Debian is impossible.

5.1.2 Non-commercial forks

Custom Debian Distributions are a solution for derivatives from Debian that are as free as Debian but had certain reasons to fork. Most of these reasons existed in the past but have now vanished because Debian is becoming easier to adapt for special purposes. To increase this flexibility, the Custom Debian Distributions approach was invented. Some examples of forks from Debian that are probably now able to integrate back into Debian are:

SkoleLinux (http://www.skolelinux.org) Mentioning SkoleLinux in the category of forks is more or less history. The merge back into Debian started with the SkoleLinux people really doing a great job to enhance Debian for their own purposes in the form of their work on debian-installer, and culminated with the formal merging of the Custom

Debian Distribution Debian-Edu and SkoleLinux, so that they are now virtually equivalent. This is the recommended way for forked distributions, and the reasons for this recommendation are given below.

DeMuDi The Agnula (http://www.agnula.org/) project, which is founded by the European Community, (and in fact is the first Free Software project that was founded by the EU at all,) forked for the following reasons:

Technical They had some special requirements for the kernel and configuration. This is more or less solved in the upcoming Debian release.

License When DeMuDi started, not enough free programs in this field existed. This situation is better now.

Organisational Because of the founded status of the project, an extra distribution had to be developed. To accomplish this requirement, Custom Debian Distributions plan to build common tools to facilitate building separate CDs with the contents of only a single distribution.

This shows that there is no longer a real need for a fork, and in fact, the organiser of the DeMuDi project is in contact to start bringing DeMuDi back into Debian. (That is why DeMuDi is mentioned in the list of Custom Debian Distributions above.)

LinEx LinEx is the very successful Distribution for schools in the Region Extremadura in Spain. The work of the LinEx people perhaps made Debian more popular than any other distribution. The project was founded by the local government of Extremadura, and each school in this region is running this distribution. While this is a great success, the further development of LinEx has to face the problems that will be explained below. So it might be worth considering following the path of SkoleLinux to integrate the needed stuff back into Debian. The LinEx people just did the first step, for instance, to try to get a free license for the very nice program Squeak (http://www.squeak.org/).

If developers of a non-commercial fork consider integrating back into Debian in the form of a Custom Debian Distribution, it might happen that their field is covered already by a Custom Debian Distribution. For instance, this would be the case for LinEx, which has exactly the same group of target users as Debian-Edu. On the other hand, some special adaptations might be necessary to fit the requirements of the local educational system. The specific changes that might be necessary would be called **flavours** of a Custom Debian Distribution.

5.1.3 Disadvantages of separate distribution

In general, a separate distribution costs extra effort. Because it is hardly possible to hire enough developers who can double the great work of many volunteer Debian developers, this would be a bad idea for economical reasons. These people would have to deal with continuous changes to keep the base system, installer, etc. up to date with the current Debian development. It would be more sane to send patches that address their special requirements to Debian instead of maintaining a complete Debian tree containing these patches.

Debian is well known for its strong focus on security. Security is mainly based on manpower and knowledge. So the best way to deal with security issues would be to base it on the Debian infrastructure, instead of inventing something new.

New projects with special intentions often have trouble to become popular to the user group they want to address. This is a matter of attaining the critical mass that was explained in 'General problem' on page 10.

Larger Free Software projects need certain infrastructure like web servers, ftp servers, (both with mirrors,) a bug tracking system, etc. It takes a fair amount of extra effort to build an entire infrastructure that is already available for free in Debian.

Forking would be a bad idea.

5.1.4 Advantages of integration into Debian

Debian has a huge user base all over the world. Any project that is integrated within Debian has a good chance to become popular on the back of Debian if the target users of the project just notice that it enables them to solve their problems. So there is no need for extra research on the side of the users, and no need for advertising for a special distribution. This fact has been observed in the Debian-Med project, which is well known for many people in medical care. It would not have gained this popularity if it had been separated from Debian.

You get a secure and stable system without extra effort for free.

Debian offers a sophisticated Bug Tracking System for free, which is a really important resource for development.

There is a solid infrastructure of web servers, ftp servers with mirrors, mail servers, and an LDAP directory of developers with a strongly woven web of trust (through gpg key signing) for free.

5.1.5 Enhancing Debian

By making changes to some packages to make them fit the needs of a target user group, the overall quality of Debian can be enhanced. In this way, enhancing Debian by making it more user friendly is a good way for the community to give back something to Debian. It would be a shame if somebody would refuse all the advantages to keeping a project inside Debian, and instead would decide to try to cope with the disadvantages because he just does not know how to do it the right way, and that it is normally easy to propogate changes into Debian. For instance, see 'How to ask for packages which are not yet included' on page 69. This section explains how you can ask for a certain piece of software to be included in Debian. The next section describes the reason why Debian is flexible enough to be adapted to any purpose.

5.2 Adaptation to any purpose

Debian is developed by about 1000 volunteers. Within this large group, the developers are encouraged to care for their own interests in packages they have chosen to look after. Thus, Debian is not bound to commercial interests.

Those who might fear this amount of freedom given to every single developer should realize that there are very strict rules, as laid out in Debian's policy (http://www.debian.org/doc/debian-policy/), which glue everything together. To keep their packages in each new release, every developer must ensure that their packages abide by that policy.

One common interest each individual developer shares is to make the best operating system for himself. This way, people with similar interests and tasks profit from the work of single developers. If users, in turn, work together with the developers by sending patches or bug reports for further enhancement, Debian can be improved also for special tasks.

For instance, developers may have children, or may work in some special fields of work, and so they try to make the best system for their own needs. For children, they contribute to Debian Jr. or Debian-Edu. For their field of work, they contribute to the appropriate CDD: Debian-Med, Debian-Lex, and so forth.

In contrast to employees of companies, every single Debian developer has the freedom and ability to realize his vision. He is not bound to decisions of the management of his company. Commercial distributors have to aim their distributions at gaining a big market share. The commercial possibilities in targeting children's PCs at home are slight, so distributions comparable to Debian-Junior are not attractive for commercial distributors to make.

Thus, single developers have influence on development - they just have to **do** it, which is a very different position compared with employees of a commercial distributor. This is the reason for the flexibility of Debian that makes it adaptable for any purpose. In the Debian world, this kind of community is called "***Do****cracy*" - the one who does, rules.

Chapter 6

Technology

6.1 Metapackages

6.1.1 Metapackage definition

A metapackage, as used by CDDs, is a Debian package that contains:

- Dependencies on other Debian packages (essential)

  Depends Use "Depends" for packages that are definitely needed for all basic stuff of the CDD in question.

  Recommends The packages that are listed as "Recommends" in the tasks file should be installed on the machine where the metapackage is installed and which are needed to work on a specific task.

  Suggests Use "Suggests" for others of lesser importance that might be possibly useful, or non-free packages. When a package is not available for the target distribution at metapackage build time the "Recommends" is turned into a "Suggests" to enable a flawless installation of the metapackage.

- Menu entries (recommended)
  - Place these in `/etc/cdd/<cdd>/menu/<pkg-name>`
  - Maintain these via role based tools
- Configuration stuff (optional)
  - `debconf` questions or pre-seeding
  - `cfengine` scripts (or similar see 'Enhancing basic technologies regarding Custom Debian Distributions' on page 54)
- Special metapackages:

- <*cdd*>-`tasks`: Contains information for `tasksel`
- <*cdd*>-`config`: Special configurations, basic stuff for user menus

Metapackages are small packages with nearly no contents. The main feature of this type of package is its dependencies on other packages. The naming of metapackages follows the pattern <*cdd*>-<*task*> where <*cdd*> stands for the short name of a Custom Debian Distribution, e.g. `junior` for Debian Jr. or `med` for Debian-Med, and <*task*> means the certain task inside the Custom Debian Distribution, e.g. puzzle or bio.

Examples:

`junior-puzzle` Debian Jr. Puzzles

`education-tasks` Tasksel files for SkoleLinux systems

`med-bio` Debian-Med micro-biology packages

6.1.2 Collection of specific software

When using metapackages, no research for available software inside Debian is necessary. It would not be acceptable for normal users to have to browse the descriptions of the whole list of the 10000 packages in Debian to find everything they need. So, metapackages are an easy method to help users to find the packages that are interesting for their work quickly.

If the author of a metapackage includes several packages with similar functionality, an easy comparison between software covering the same task is possible.

Moreover, the installation of a metapackage ensures that no package that is necessary for the intended task can be removed without explicit notice that also the metapackage has to be removed. This helps non specialist administrators to keep the installation fit for the specialized users.

By defining conflicts with some other packages inside the metapackage, it is possible to ensure that a package that might conflict for some reasons for the intended task can not be installed at the same time as the metapackage is installed.

All in all, metapackages enable an easy installation from scratch, and keep the effort required for administration low.

6.1.3 Adapted configuration inside metapackages

Besides the simplification of installing relevant packages by dependencies inside metapackages, these packages might contain special configuration for the intended task. This might either be accomplished by pre-seeding `debconf` questions, or by modifying configuration files in a `postinst` script. It has to be ensured that no changes that have been done manually by the administrator will be changed by this procedure. So to speak, the `postinst` script takes over the role of a local administrator.

6.1.4 Documentation packages

A "traditional" weakness of Free Software projects is missing documentation. To fix this, Custom Debian Distributions try to provide relevant documentation to help users to solve their problems. This can be done by building `*-doc` packages of existing documentation, and by writing extra documentation, like manpages, etc. By supplying documentation, Custom Debian Distributions fulfil their role in addressing the needs of specialised users, who have a great need for good documentation in their native language.

Thus, translation is a very important thing to make programs more useful for the target user group. Debian has established a Debian Description Translation Project (http://ddtp.debian.net/), which has the goal to translate package descriptions. There is a good chance this system could also be used for other types of documentation, which might be a great help for Custom Debian Distributions.

6.2 Handling of metapackages

In short, there are no special tools available to handle metapackages nicely. But there are some tricks that might help, for the moment.

6.2.1 Command line tools

`apt-cache` The program `apt-cache` is useful to search for relevant keywords in package descriptions. With it, you could search for a certain keyword connected to your topic (for instance "`med`") and combine it reasonably with `grep`:

```
~> apt-cache search med | grep '^med-'
med-bio - Debian-Med micro-biology packages
med-common-dev - Debian-Med Project common files for developing...
med-dent - Debian-Med package for dental practice client
med-doc - Debian-Med documentation packages
med-imaging - Debian-Med imaging packages
med-imaging-dev - Debian-Med packages for medical image develop...
med-tools - Debian-Med several tools
med-bio-contrib - Debian-Med micro-biology packages (contrib an...
med-common - Debian-Med Project common package
med-cms - Debian-Med content management systems
```

This is **not really straightforward**, and absolutely unacceptable for end users.

`grep-dctrl` The program `grep-dctrl` is a grep for Debian package information, which is helpful for extracting specific package details matching certain patterns:

```
~> grep-dctrl ': med-' /var/lib/dpkg/available | \
   grep -v '^[SIMAVF]' | \
   grep -v '^Pri'
Package: med-imaging
Depends: paul, ctsim, ctn, minc-tools, medcon, xmedcon, med-common
Description: Debian-Med imaging packages

Package: med-dent
Depends: debianutils (>= 2.6.2), mozilla-browser | www-browser, ...
Description: Debian-Med package for dental practice client

Package: med-bio
Depends: bioperl, blast2, bugsx, fastdnaml, fastlink, garlic...
Description: Debian-Med micro-biology packages

Package: med-common
Depends: adduser, debconf (>= 0.5), menu
Description: Debian-Med Project common package

Package: med-common-dev
Depends: debconf (>= 0.5)
Description: Debian-Med Project common files for developing ...

Package: med-tools
Depends: mencal, med-common
Description: Debian-Med several tools

Package: med-doc
Depends: doc-linux-html | doc-linux-text, resmed-doc, med-co...
Description: Debian-Med documentation packages

Package: med-cms
Depends: zope-zms
Description: Debian-Med content management systems

Package: med-imaging-dev
Depends: libgtkimreg-dev, ctn-dev, libminc0-dev, libmdc2-dev...
Description: Debian-Med packages for medical image development

Package: med-bio-contrib
Depends: clustalw | clustalw-mpi, clustalx, molphy, phylip, ...
Description: Debian-Med micro-biology packages (contrib and ...
```

This is, like the `apt-cache` example, **also a bit cryptic**, and again is not acceptable for end users.

auto-apt The program `auto-apt` is really cool if you are running a computer that was installed from scratch in a hurry, and are sitting at a tradeshow booth preparing to do a demo. If you had no time to figure out which packages you needed for the demo were missing so you could install all of them in advance, you could use `auto-apt` in the following manner to guarantee that you have all of the files or programs you need:

```
~> sudo auto-apt update
put: 880730 files,  1074158 entries
put: 903018 files,  1101981 entries
~> auto-apt -x -y run
Entering auto-apt mode: /bin/bash
Exit the command to leave auto-apt mode.
bash-2.05b$ less /usr/share/doc/med-bio/copyright
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bugsx fastlink readseq
The following NEW packages will be installed:
  bugsx fastlink med-bio readseq
0 packages upgraded, 4 newly installed, 0 to remove and 183 ...
Need to get 0B/1263kB of archives. After unpacking 2008kB wi...
Reading changelogs... Done
Selecting previously deselected package bugsx.
(Reading database ... 133094 files and directories currently...
Unpacking bugsx (from .../b/bugsx/bugsx_1.08-6_i386.deb) ...
Selecting previously deselected package fastlink.
Unpacking fastlink (from .../fastlink_4.1P-fix81-2_i386.deb) ...
Selecting previously deselected package med-bio.
Unpacking med-bio (from .../med-bio_0.4-1_all.deb) ...
Setting up bugsx (1.08-6) ...

Setting up fastlink (4.1P-fix81-2) ...

Setting up med-bio (0.4-1) ...

localepurge: checking for new locale files ...
localepurge: processing locale files ...
localepurge: processing man pages ...
This package is Copyright 2002 by Andreas Tille <tille@debian.org>

This software is licensed under the GPL.

On Debian systems, the GPL can be found at /usr/share/common-...
/usr/share/doc/med-bio/copyright
```

Just do your normal business - in the above example, `less`

`/usr/share/doc/med-bio/copyright` - and if the necessary package is not yet installed, `auto-apt` will care for the installation and proceed with your command. While this is really cool, this is **not really intended for a production machine**.

The short conclusion here is: **There are no sophisticated tools that might be helpful to handle metapackages as they are used in Custom Debian Distributions - just some hacks using the powerful tools inside Debian.**

6.2.2 Text user interfaces

`dselect` This good old package handling tool provides no special help to handle metapackages in an elegant manner.

`tasksel` The Debian task installer `Tasksel` is the first interface for package selection that is presented to the user when installing a new computer. The `End-user` section should contain an entry for each Custom Debian Distribution. This is currently the case for Debian-Jr.

```
Debian Task Installer v1.43 - (c) 1999-2003 SPI and others

---------- Select tasks to install ----------
   -- End-user ----
   [X] Debian Jr.
   [ ] Desktop environment
   [ ] Games
   [ ] Linux Standard Base
   [ ] X window system
   [ ] Office environment
   -- Hardware Support  ----
   [ ] Dialup internet
   [ ] Laptop
   [ ] Broadband internet connection
   -- Servers  ----
   [ ] DNS server
   [ ] File server
   [ ] Mail server
   [ ] Usenet news server
   [ ] SQL database
   [ ] Print server
   [ ] Conventional Unix server

     <Finish>        <Task Info>        <Help>
```

Unfortunately, there are some issues that prevent further Custom Debian Distributions from being included in the `tasksel` list, because the dependencies of this task can affect

what appears on the first installation CD. This problem would be even greater if all Custom Debian Distributions were added, and so a different solution has to be found here. (See #186085 (http://bugs.debian.org/186085).) In principle, `tasksel` is a good tool for easy installation of Custom Debian Distributions.

aptitude This is a better replacement for `dselect`, and has some useful support for searching for and grouping of packages. While this is not bad, it was not intended for the purpose of handling Custom Debian Distributions, and thus there could be some better support to handle metapackages more cleverly.

Short conclusion: **There is a good chance metapackages could be handled nicely by the text based Debian package administration tools, but this is not yet implemented.**

6.2.3 Graphical user interfaces

Debian *Woody* does not contain a really nice graphical user interface for the Debian package management system. But the efforts to support users with an easy to use tool have increased, and so there there will be some usable options in Sarge.

gnome-apt This is the native GNOME flavour of graphical user interfaces to apt. It has a nice `Search` feature that can be found in the `Package` menu section. If for instance the packages of the Debian Jr. project come into the focus of interest a search for "`junior-*`" will show up all related packages including their descriptions. This will give a reasonable overview about metapackages of the project.

synaptic Even more sophisticated and perhaps the best choice for users of Custom Debian Distributions. `Synaptic` has a nice filter feature, which makes it a great tool here. Moreover `synaptic` is currently the only user interface that supports Debian Package Tags (see 'Debian Package Tags' on page 53).

kpackage This is the user interface of choice for KDE lovers. Regarding its features (with exception of Debian Package Tags) it is similar to both above.

Short conclusion: **As well as the text based user interfaces these tools are quite usable but need enhancements to be regarded as powerful tools for Custom Debian Distributions.**

6.2.4 Web interfaces

Web search (http://packages.debian.org/) Debian has a web interface that can be used to search for certain substrings in package names. For instance if you are searching the meta packages of Debian-Med you could point your favourite Browser to

http://packages.debian.org/cgi-bin/search_packages.pl?keywords=med-subword=1

As a result you will get a list of all Debian-Med packages.

Package Tracking System (http://qa.debian.org/developer.php) The Package Tracking System is a really great tool that provides essential information about packages. Regarding Custom Debian Distributions it can help if you know the Debian user name of the developer who is responsible for the metapackages:

Debian-Jr: http://qa.debian.org/developer.php?login=synrg

Debian-Med: http://qa.debian.org/developer.php?login=tille

Debian-Edu: http://qa.debian.org/developer.php?login=pere

The other way to use the Package Tracking System is to search for packages starting with a certain letter:

Debian-Jr: http://packages.qa.debian.org/j

Debian-Med: http://packages.qa.debian.org/m

But the list that is obtained by this method is much larger than it would be useful for a good overview.

So the conclusion is - we just need better support here for Custom Debian Distributions.

list-junior.sh The package junior-doc contains a script /usr/share/doc/junior-doc/examples/scripts/list-junior.sh that checks for the installed packages of a Custom Debian Distribution and builds a simple web page describing these packages. (The BTS contains a patch to let this script work also for other Custom Debian Distributions.)

Short conclusion: **Some very basic things can be done with the web interfaces described above but techniques have to be developed to provide useful information about each Custom Debian Distribution.**

6.2.5 Future handling of metapackages

Obviously there are no nifty tools as you might know them from Debian available yet. The user interfaces for apt-get have to be enhanced drastically to make them easy enough to make them useful in the hands of an end user. This might implicitly mean that we need some additional control fields in dpkg to implement reasonable functionality. The following items are target of future development:

- Searching for existing metapackages
- Overview about dependencies of these metapackages
- Enhancing tools like aptitude, synaptic, etc.
- Special tasksel section
- Web tools that keep metapackage information up to date

Furthermore it is necessary to find a set of keywords for each Custom Debian Distribution and write a tool to search these keywords comfortable. The best way to accomplish this might be to make use of Debian Package Tags, which is a quite promising technique.

Tools that grep the apt cache directly for metapackages have to be written or rather the available tools for this should be patched for this actual functionality.

6.3 User roles

As stated above specialists have only interest in a subset of the available software on the system they are using. In an ideal world, this would be the only software that is presented in the menu. This would allow the user to concentrate on his real world tasks instead of browsing large menu trees with entries he does not understand.

To accomplish this, a technique has to be implemented that allows to define a set of users who get a task-specific menu while getting rid of the part of software they are not interested in. Moreover this has to be implemented for certain groups of users of one Custom Debian Distribution, which are called "roles". There are several techniques available to manage user roles. Currently in the field of Custom Debian Distributions a UNIX group based role system is implemented. This means, that a user who belongs to a certain group of a Custom Debian Distribution is mentioned in the `/etc/group` file in the appropriate group and gets a special user menu that is provided for exactly this group.

Strictly speaking it is not the best solution to conflate a configuration mechanism, which users see with menus, with access control, i.e. unix groups. It might be confusing, and wastes the limited number of groups to which a user can belong. On the other hand this is a solution that works for the moment, and has no real negative impact on the general use of the system. The benefit of using unix groups is that there is a defined set of tools provided to handle user groups. This makes life much easier; there is no *practical* limit to the number of groups to which a user may belong for the existing Custom Debian Distributions at this time.

In the long run, this role system might even be enhanced to certain "*levels*" a user can have and here the UNIX groups approach will definitely fail and has to be replaced by other mechanisms. This will include the possibility to enable the user adjust his own level ("novice", "intermediate", "expert") while only the administrator is able to access the UNIX groups. On the other hand such kind of user level maintenance is not only a topic for Custom Debian Distributions but might be interesting for Debian in general.

Another point that speaks against using UNIX groups for role administration is the fact that local administrators are not in all cases competent enough to understand the UNIX role concept as a security feature and thus a real role concept including tools to maintain roles are needed in the future.

The handling of the user menus according to the groups is implemented in a flexible plugin system and other ways of handling groups (i.e. LDAP) should be easy to implement.

6.3.1 User menu tools

Using the Debian menu system

The Debian menu system cares for menu updates after each package installation. To enable compliance with the *role* based menu approach it is necessary to rebuild the user menu after each package installation or after adding new users to the intended role. This can be done by using the `cdd-update-menus(8)` (see 'cdd-update-menus(8)' on page 62) script from `cdd-common`. It has to be said that using `cdd-update-menus` is not enough to change the menu of a user. To accomplish this a call of the general `update-menu` script for every single user of a Custom Debian Distribution is necessary if this is not done by the `postinst` script of a metapackage. This can easily been done if the configuration file of a Custom Debian Distribution `/etc/cdd/<cdd>/<cdd>.conf` contains the line

```
UPDATEUSERMENU=yes
```

It is strongly suggested to use the package `cdd-dev` to build metapackages of a Custom Debian Distribution that will move all necessary files right into place if there exists a `menu` directory with the menu entries. Note, that the users `${HOME}/.menu` directory remains untouched.

Managing Custom Debian Distribution users with `debconf`

Using `cdd-dev` it is very easy to build a *cdd*`-config` package that contains `debconf` scripts to configure system users who should belong to the group of users of the Custom Debian Distribution *cdd*. For example see the `med-common` package.

```
~> dpkg-reconfigure med-common

Configuring med-common
----------------------

Here is a list of all normal users of the system.  Now you can select those u
should get a Debian-Med user menu.

  1. auser (normal user A)          6. fmeduser (med user F)
  2. bmeduser (med user B)          7. glexuser (lex user G)
  3. cjruser (jr user C)            8. hmeduser (med user H)
  4. djruser (jr user D)            9. iadmin (administrator I)
  5. eadmin (administrator E)       10. juser (normal user J)

(Enter the items you want to select, separated by spaces.)

:-! Please specify the Debian-Med users! 2 8
```

This example shows the situation when you `dpkg-reconfigure med-common` if *med user B* and *med user H* were defined as users of Debian-Med previously and *med user F* should be added to the group of medical staff. (For sure it is more convenient to use the more comfortable interfaces to `debconf` but the used SGML DTD does not yet support screen shots (http://bugs.debian.org/140684).)

6.4 Development tools

Building a metapackage is more or less equal for each meta package. This was the reason to build a common source package `cdd` that builds into two binary packages

`cdd-dev` Helpful tools to build metapackages from a set of template files. These tools are interesting for people who want to build metapackages in the style Debian-Edu and Debian-Med are currently doing this. The purpose of this package is to make maintenance of metapackages as easy as possible.

This package is described in detail in appendix 'Package `cdd-dev`' on page 59.

`cdd-common` This package provides some files that are common to meta packages of Common Debian Distributions especially those that were built using the tools of the package `cdd-dev`. It introduces a method to handle system users in a group named according to the name of the Custom Debian Distribution. The user menu approach is explained in detail in 'User roles' on page 33.

This package is described in detail in appendix 'Package `cdd-common`' on page 61.

The usage of the tools that are contained in these packages are described now in detail.

6.5 Other interesting tools

6.5.1 Simple-CDD

The tool `simple-cdd` is a limited though relatively easy tool to create a customized debian-installer CD.

It includes simple mechanisms to create "profiles" that define common system configurations, which can be selected during system installation. `Simple-cdd` also makes it easy to build CDs with language and country settings pre-configured, or to use a 2.6 kernel by default.

This can be used to create a crude "Custom Debian Distribution" using packages from debian, with pre-configuration of packages that use `debconf`. Custom configuration scripts can be specified to handle packages that don't support `debconf` pre-configuration.

Testing CD images with `qemu` is also made simple with a provided script.

It has only been tested with `debian-cd` from sarge (other than `debpartial-mirror`), so if using a new `debian-cd` or `debian-installer`, it may require some tweaks.

Chapter 7

How to start a Custom Debian Distribution

This chapter more or less covers the text of the Debian Subproject HOWTO (http://packages.debian.org/unstable/doc/subproject-howto.html), which was written by Ben Armstrong <synrg@debian.org>. Ben has agreed that his text should be included here, and the subproject-howto will be orphaned once the current document is ready for packaging.

7.1 Planning to form a Custom Debian Distribution

In this section, issues to think about before starting a Custom Debian Distribution will be discussed. It is important to have a clear idea where to head and how to get there before launching into this adventure.

7.1.1 Leadership

The existing Custom Debian Distributions have clearly shown that they depend on a person who keeps things running. If anybody wants to start a project at first, he has to answer the question: *"Am I the right person for the job?"* Surely this is a question that may be faced with some amount of uncertainty. The way Custom Debian Distributions started in the past was for the person with the idea for the project to just start doing the work. After some time using this approach, it became clear that if the project lacked a person to take leadership, the project would become stale. So the initiator has to answer the question clearly, whether or not he is able to continue in the *job* of leader, considering the amount of time he will have to spend, and the technical and social skills which are needed.

7.1.2 Defining the subproject scope

It is as important to decide what your group is not going to do as it is what it is going to do. A clear borderline is essential for the development of the project. Without it, outsiders might either expect more from the project than it can accomplish, or may ignore the project, finding it not helpful because they are not able to find out the purpose.

By maintaining a good relationship with other Free Software projects, some common tasks can be done very effectively. When efforts can be shared, the amount of work for each project can be reduced.

Checking for cooperation with other Custom Debian Distributions is always a good idea. In technical terms, this is obvious, but sometimes there are possibilities to share efforts when the goals of two projects have parts in common.

The one who decides to start a Custom Debian Distribution takes on a responsibility for this project. It has to be for the good of Debian as a whole, and should bring an extra reputation to our common goal to build the best operating system.

7.1.3 Initial discussion

By the time you have begun to think about forming the subproject, have made the commitment to lead it, and have sketched out a bit of where you want to go and how you'll get there, you have likely already done some informal discussion with your peers. It is time, if you haven't already, to take these ideas to the broader Debian developer community, opening discussion on the creation of your group.

Calling all developers

At this stage, you will want to reach as broad an audience as possible. You have carefully thought out what you're going to do, and are able to articulate it to Debian as a whole. Let everyone know through the `<debian-devel-announce@lists.debian.org>` mailing list, setting the `Reply-to:<debian-devel@lists.debian.org>` and listen to what everyone has to say about your idea. You may learn some valuable things about pitfalls that may lie ahead for your group. Maybe even show-stoppers at that. You may also find a number of like-minded individuals who are willing to join your group and help get it established.

Steering the discussion

It's all too easy to get lost in ever-branching-out sub-threads at this point. Many people will be firing off ideas left, right and centre about what your group should do. Don't worry too much about containing the discussion and keeping it on track with your main idea. You would rather not squelch enthusiasm at this point. But do try to steer the discussion a bit, focusing on the ideas that are central to your subproject and not getting lost in the details.

At some point, you'll decide you've heard enough, and you're ready to get down to the business of starting your group.

7.2 Setting up

7.2.1 Mailing list

It is fairly important to enable some means for communication for the project. The most natural way to do this is with a mailing list.

Creating a new mailing list starts with a wishlist bug against *lists.debian.org*. The format of this bug has to follow certain rules (http://www.debian.org/MailingLists/HOWTO_start_list).

Before your list can be created, the listmasters will want assurance that creation of the list is, in fact, necessary. So for this reason, don't wait for your list to be created. Start discussing your new project on <debian-devel@lists.debian.org> immediately. To help distinguish your project's posts from the large amount of traffic on this list, tag them in the Subject field with an agreed-upon tag. For instance for general discussion about Custom Debian Distributions the tag [custom] should be used. An example bug report to create the relevant list is bug #237017 (http://bugs.debian.org/237017).

When sufficient discussion on the developer's list has taken place and it is time to move it to a subproject list, add to your wishlist bug report some URLs pointing to these discussions in the archives as justification for creation of your list.

7.2.2 Web space

A fairly important way to let people know what your Custom Debian Distribution is about is certainly a web page. While there are a number of ways to go about this, the simplest is to put them at the developer home page at http://people.debian.org if an official Debian developer is starting the project.

Another possibility, and one which is fairly attractive because it facilitates collaborative web site creation and maintenance, is to put a page on the Wiki (http://wiki.debian.org). There is a special Wiki page for Custom Debian Distributions (http://wiki.debian.org/index.cgi?CustomDebian).

A third, and more recent possibility is to start a project at alioth.debian.org (http://alioth.debian.org), since hosting Debian development projects is the whole reason for this site.

Finally, the best way is to have a page under http://www.debian.org/devel. While not as straightforward as any of the other options, this approach has its advantages. First, the site is mirrored everywhere. Second, the Debian web site translators translate pages into many different languages, reaching new potential audiences for your Custom Debian Distribution, and improving communication with other members of your project and interested parties for

whom English is not their most comfortable language. Third, a number of templates are available to make your site more integrated with the main web site, and to assist with incorporating some dynamic content into your site. Before you join the Debian-Web team you should learn more about building Debian web pages (http://www.debian.org/devel/website).

Once this is done, the Debian web pages team should be contacted via the mailing list <debian-www@lists.debian.org> to add the project to the organisation page (http://www.debian.org/intro/organization).

7.2.3 Repository

On alioth.debian.org (http://alioth.debian.org/) a Gforge (http://gforge.org/)-site is running to host all Debian related project work. Creating a project on Alioth is a good idea to start teamwork on the code a certain Custom Debian Distribution is releasing.

7.2.4 Formal announcement

Once there is a list, or at least enough preliminary discussion on debian-devel to get started, and there is some information about the newly planned Custom Debian Distribution available on the web, it is time to send a formal announcement to <debian-devel-announce@lists.debian.org>. The announcement should include references to past discussions, any web pages and code which might already exist, and summarise in a well thought out manner what the project is setting out to achieve. Enlisting the help of fellow developers on irc or in private email to look over the draft and work out the final wording before it is sent out is always a good idea.

Emails to <debian-devel-announce@lists.debian.org> have to be signed by the GPG key of an official Debian developer. However, it should not be a very hard task if somebody wants to support Debian while not yet being a developer to find a developer who volunteers to sign an announcement of a reasonable project. It might be reasonable to send this announcement also to other relevant non-Debian lists. If your announcement is well done, it will draw a number of responses from many outsiders, and will attract people to Debian.

7.2.5 Explaining the project

Now the real work starts. People who are involved in the project should be aware that they have to answer questions about the project whenever they show up at conferences or at an exhibition booth. So being prepared with some flyers or posters is always a good idea.

7.3 Project structure

7.3.1 Sub-setting Debian

While there are a variety of different kinds of work to be done in Debian, and not all of them follow this pattern, this document describes one particular kind of project. Our discussion about Custom Debian Distributions concerns sub-setting Debian. A sub-setting project aims to identify, expand, integrate, enhance, and maintain a collection of packages suitable for a particular purpose by a particular kind of user.

Now, strictly speaking, a subset of packages could be more general than described above. A subset could be a broad category like "audio applications" or "network applications". Or it could be more specific, such as "web browsers" or "text editors". But what a sub-setting project such as Debian Jr. aims to do is not focus on the kind of package, but rather the kind of user. In the case of Debian Jr. it is a young child.

The sort of user the project looks after, and which of the needs the project hopes to address are defined by the project's goals. Thus, Debian Jr. first had to decide which children the project would reach: "What age?" "English speaking children only, or other languages as well?" Then the project had to determine how and where they would be using Debian: "At home?" "In school?" "Playing games?" "On their own systems?" "On their parents' systems?"

The answers to all of these questions are not straightforward. It is very much up to the project to choose some arbitrary limits for the scope of their work. Choose too broad a focus, or one which duplicates work already done elsewhere, and the energy of the project dissipates, making the project ineffective. Choose too narrow a focus and the project ends up being marginal, lacking the critical mass necessary to sustain itself.

A good example was the request to split the microbiology related packages out of Debian-Med into a Debian-Bio project. This is reasonable in principle, and should really be done. In fact, the initiator of Debian-Med would support this idea. So he gave the answer: "Just start the Debian-Bio project to take over all related material. Until this happens, Debian-Med will cover medical material that deals with sequence analysis and so forth." Unfortunately, there was silence from the Debian-Bio proponents after this answer.

Of course, it sometimes turns out that you start working on a project thinking you know what it is about, only to find out later that you really had no idea what it would become until the user base has grown beyond the small community of developers that started it. So, none of the decisions you make about your project's scope at the beginning should be taken as set in stone. On the other hand, it is your project, and if you see it veering off in directions that are contrary to your vision for it, by all means steer it back on course.

7.3.2 Using tasksel and metapackages

According to the plan of the project, the first metapackages ('Metapackages' on page 25) should be developed. It is not always easy to decide what should be included, and which metapackages should be built. The best way to decide on this point is to discuss on the mailing list some well thought out proposals.

Section 'Text user interfaces' on page 30 mentions `tasksel` as a tool to select a Custom Debian Distribution, and explains why it is currently not possible to get a Custom Debian Distribution included into the task selection list.

7.4 First release

7.4.1 Release announcement

Beyond the release announcement for Debian itself, it is necessary to put some thought and work into a release announcement for the first release of a Custom Debian Distribution. This will not only be directed at the Debian developer community, but also at the users. This will include potential new Debian users abroad, who may not be on a Debian mailing list. Here, the same principle applies as for the first announcement of the project: it is important to consider sending the information to other relevant forums.

7.4.2 Users of a Custom Debian Distribution

By this time, people have newly installed Debian along with the material in the Custom Debian Distribution, or have installed the meta packages on their existing Debian systems. Now comes the fun part, building relationships with the user community.

Devoting resources to the users

Users are a mixed blessing. In the first development phase there are some developers who are users, and some intrepid "early adopters." But once it is released, the first version is "out there," and the project will certainly attract all kinds of users who are not necessarily as technically savvy as your small development user community. Be prepared to spend some time with them. Be patient with them. And be listening carefully for the underlying questions beneath the surface questions. As draining as it can be to deal with users, they are a very key component to keeping your development effort vital.

Developer vs. user mailing list

Should a user list be created? It's not as cut-and-dried as it might at first appear. When user help requests start coming in, you might at first see them as a distraction from the development effort. However, you don't necessarily want to "ghettoize" the user community into a separate list early. That's a recipe for developers to get out of touch very quickly with the users. Tolerate the new user questions on the developer list for a while. Once a user list is finally set up, courteously redirect user questions to the user list. Treat your users as the valuable resource about how your project is working "in the field" that they are.

User support beyond Debian

Fortunately, we're not in the business of supporting users alone. Look beyond Debian for your allies in user support: Linux user groups (LUGs) and the users themselves. Develop an awareness of who has stakes in seeing your project succeed, and enlist their help in getting a strong network of support established for your work.

Chapter 8

The web sentinel

8.1 Existing and prospective packages

The result of the stuff described in this paragraph could for example be viewed at the tasks page of the Debian-Med project (http://debian-med.alioth.debian.org/tasks/). If you want stuff like that just follow the instructions below.

If a Custom Debian Distribution should be presented one of the first questions is, what packages are available. The next question might be which packages are on the todo list for inclusion in Debian to make Debian even more attractive for people the CDD is targeting at. Both questions can be answered if you point people to the dynamically created tasks page. The page is rebuild daily to stay up to date according to recent developments of the CDD. The build process works as follows:

- Read dependency information of the `tasks` files.
- Verify whether there is really a package with this name and print the description of this package.
- If there is no such package in Debian try to parse the `tasks` file whether there is some information given and mark the result as prospective package for further inclusion.

The rationale behind this is to provide as much as possible information about packages that might be interesting for the target user of the CDD. Moreover the page can provide useful information for developers about things that might be a useful help for the project to work down the todo list and build Debian packages for software that is not yet included in Debian. To get the todo list builded it is necessary to add some additional information to the task files which are the main database of information for the CDD. The information is following the RFC822 syntax as all Debian control files do and is kept quite simple:

Depends / Recommends / Suggests Even if there is no Debian package available for the moment the names of prospective packages are given as if they would exists. The rationale

behind is that once such a package might exist the source of the metapackage does not have to be changed and will work out of the box after rebuilding.

Ignore The Ignore key should be the favourite way to use for specifying prospective packages in case the packages should be evaluated once it appears in the Debian package pool. If "Depends", "Recommends" or "Suggests" are used for not yet existing packages they will be turned into the list of Suggests of the metapackage and thus might be possible to become installed on a users machine if the user decides to install all suggested packages. If some evaluation should be done first the "Ignore" tag is your friend.

Homepage This is the URL to the software that should be packaged.

WNPP In case there might be a WNPP bug filed for this software the bug number is given here. This helps to keep track of the ongoing effort to package the software.

Responsible In case some developer claimed to care for the software (perhaps by issuing the WNPP bug report) the e-mail address of this developer is given here to enable an easy way to contact this person.

License Debian cares always about the license. This information about prospective packages should be easily available.

Pkg-URL In some cases there are unofficial packages for some software which are prepared by a third party. It helps our users if they could install such a package and thus the URL to it might be a helpful hint. This is also true for developers because they might have a look at this packaging before they start from scratch. Often packages are available at mentors.debian.net (`http://mentors.debian.net/`) and prepared by people who do not yet have an official Debian maintainer status and thus are not able to upload packages to the Debian mirror. The packages at mentors are waiting for sponsoring of an official Debian maintainer and if such a package shows up in the CDD tasks list it might speed up the inclusion into official Debian distribution.

Pkg-Description This tag should give reasonable information about the software as it normally is done in `debian/control` files. It can be either a copy of the description of the WNPP bug or could be used to file a WNPP bug and thus helps to simplify the packaging work.

8.2 Debian Description Translation Project

The Debian Description Translation Project (`http://ddtp.debian.net/`) (see 'Documentation packages' on page 27) provides users of non English languages with information about Debian packages. The sense of supporting especially the translations of descriptions which are in the focus of a CDD is to make the CDD even more usable for our target users. Moreover people interested in the special field of the CDD are most probably able to provide good translations if it comes to texts that are specific to their field of knowledge. Thus there is a web page automatically created that parses the tasks packages for package names and verifies the translation status of the package descriptions.

Finally the DDTP descriptions can be used to create internationalised pages of existing packages which might help users with insufficient skills in English to easily find their software of interest.

8.3 Bugs overview

The goal of a CDD is to support their user as best as possible. So a feature to have a quick overview about all packages in our focus might be helpful. This is solved by the bugs overview page. To create this page the `tasks` files are parsed for the listed dependencies. Then the Debian Bug Tracking System is consulted about known bugs of these packages. All bugs are listed and marked with different colours according to their severity. So the developers can easily check this page in case they plan to fix some bugs that are relevant for the CDD.

8.4 SVN overview

This page gives a recent overview about the current development activities of the CDD developers.

8.5 Quality assurance report

The Debian Quality Assurance group does a decent job in watching the status o f Debian packages. If a package features a valid `debian/watch` the tool `uscan` is able to verify the upstream source location for newer versions. The QA report page reports issues about the packages that are relevant for a CDD.

Chapter 9

To do

9.1 Establishing and using communication platforms

Each Custom Debian Distribution has an own mailing list for discussion of specific development issues. Because there are several common issues between all Custom Debian Distributions also a common mailing list was created. People who are interested in working on common issues like building metapackages, technical issues of menu systems or how to create CDs for Custom Debian Distributions could subscribe to this list or read the list archive (http://lists.debian.org/debian-custom/).

Moreover the project cdd (http://alioth.debian.org/projects/cdd/) on Alioth was started. The subversion repository (http://svn.debian.org/viewcvs/cdd/) can be browsed or checked out by by

```
svn checkout svn://svn.debian.org/svn/cdd/cdd/trunk
```

for anonymous users. Developers should check out via

```
svn checkout svn+ssh://username@svn.debian.org/svn/cdd/cdd/trunk cdd
```

The current layout for the repository is as follows:

```
cdd -+- cdd -+- branches -+- cdd -+- 0.3.x
     |       |
     |       +- tags -----+- cdd -+- 0.3   [older versions of cdd tools]
     |       |            |       +- 0.3.1
     |       |            |          ...
     |       |            |       +- <latest>
     |       |            |
     |       |            +- doc -+- 0.1   [older versions of this doc]
```

```
|       |                    +- 0.2
|       |                    +- 0.3
|       |                        [Since 0.4.1 the doc is in cdd direct
|       |
|       +- trunk ----cdd [code in development for cdd source package]
|                     |
|                     +-- doc [this document = cdd-doc package]
|
+- projects -+--- med ---+- branches
             |           |
             |           +- tags
             |           |
             |           +- trunk [development version of Debian-Med]
             |
             +- science -+- branches
             |           |
             |           +- tags
             |           |
             |           +- trunk [development version of Debian-Scien
             |
             +- ...
             |
             ...
```

There is a mailing list (http://lists.alioth.debian.org/mailman/listinfo/cdd-commits) with subversion changes and a CIA system (http://cia.navi.cx/stats/project/debian-custom) for tracking changes in the Custom Debian Distributions projects in real-time.

9.2 Enhancing visibility

If a user installs Debian via official install CDs the first chance to do a package selection to customise the box is `tasksel`. The first Custom Debian Distribution Debian-Junior is mentioned in the task selection list and thus it is clearly visible to the user who installs Debian.

In bug #186085 (http://bugs.debian.org/186085) a request was filed to include Debian-Med in the same manner. The problem with the `tasksel`-approach is that all included packages should be on the first install CD. This would immediately have the consequence that the first install CD would run out of space if all Custom Debian Distributions would be included in the task selection list.

How to enhance visibility of Custom Debian Distributions for the user who installs Debian from scratch?

Change `tasksel` policy. If the *packages must be on the first CD* feature of `tasksel` would be

dropped all Custom Debian Distributions could be listed under this topic in the task selection list.

Custom Debian Distributions information screen. Alternatively a new feature could be added to `tasksel` or in addition to `tasksel` in the installation procedure which presents a screen which gives some very short information about Custom Debian Distributions (perhaps pointing to this document for further reference) and enables the user to select from a list of the available Custom Debian Distributions.

Provide separate install CDs By completely ignoring the installation of the official installation CD each Custom Debian Distribution can offer a separate installation CD. This will be done anyway for certain practical reasons (see for instance the Debian-Edu - SkoleLinux approach). But this is really no solution we could prefer because this does not work if the user wants to install more than one Custom Debian Distribution on one computer.

Change overall distribution philosophy of Debian. This way is concerned to some ideas from Debian developers who took part in Open Source World Conference in Malaga and is explained in Detail in 'New way to distribute Debian' on page 56. This would save the problem of making Custom Debian Distribution visible to users in a completely different way because in this case Debian would be released as its various flavours of Custom Debian Distributions.

Whichever way Debian developers will decide to go it is our vital interest to support users and *show* our users the tools we invented to support them.

9.2.1 Custom Debian Distributions web pages

Most Custom Debian Distributions maintain their own web space under `http://www.debian.org/devel/cdd` to provide general information which will be translated by the Debian web team. This is a good way to inform users about the progress of a project. A special way to announce what is done and what is planed is the list of yet packaged software and software which is intended to be included. To do this in a nice manner Tobias Toedter `<t.toedter@gmx.net>` defined a new tag for Debian-Med in order to ease translation by making use of the `gettext` functionality. In the meantime, this new tag was extended to be useful for other Custom Debian Distributions as well.

As a result, a new `.pot` file was created, called `debian-cdd.pot`. Translators of the web pages should update their `.po` files and translate this new file into their language. For the translation teams who have already begun to translate the web pages of the Debian-Med Custom Debian Distribution, here is a short explanation of the newly introduced tag and its use. The tag is called `project`, and it takes a few attributes. The complete (empty) tag looks like this:

```
<project name=""
  url=""
  license=""
```

```
    deb=""
    anchorid="">
    Here goes the description of the project.
  </project>
```

The meaning of the attributes is as follows:

name This is the name of the project which is yet packaged or should be packaged for the Custom Debian Distribution in question. In most cases, you won't have to translate this attribute.

url This is the URL of the homepage of the project. You will almost never have to do any translational work here. At least I can't think of any.

license The license under which the project is released. In most cases (e.g. GPL, BSD) you don't have to modify anything here. Some projects use a custom license or there's anything other which requires some more explanation in plain text. Of course, this plain text should be translated.

deb This is the URL of a Debian package. If the project is available as an official Debian package (i.e. it is included in the Debian distribution or available in the contrib or non-free section) or if the project is being packaged by someone, but not stored on the Debian servers, this attribute is used. If there's no package available at all, this attribute is either left blank or omitted completely. There's no need to change this value in your translations.

anchorid Every project gets an automatically created HTML anchor. This auto-anchor is created by using the project's name (Convert all ASCII characters into lowercase and replace any other character with the underscore. Silly example: "BioSig - For Everyone" -> "biosig___for_everyone"). If for some reason this is not wanted, the id and name of the anchor can be specified with this attribute. An example of the use of this attribute can be found in other.wml. The project's name is "HARP - HArmonisation for the secuRity of web technologies and aPplications", which is awfully long. The anchorid attribute in this case is set to "harp". Note that you should never have to change anything here, if it is already defined in the English page. If you have to translate the name of the project, the automatic creation of the anchorid will naturally give a different result then the English anchorid. In this case, please use this attribute to manually specify the anchor which should be used, according to the rules above: Convert all (ASCII) characters into lowercase, and replace any other character with an underscore. So in conclusion, you should always use the anchorid attribute if you've translated the name attribute.

The interesting part of the `project`-tag is the body, between the opening and the closing tag. This is the description of the project, and this is the part where you're going to have to translate the text. The best way to learn the usage of this tag is to have a look at the Debian-Med examples.

Moreover it might make sense to sort the list of projects according to the following keys:

available Debian package Top most you should list all packages which are just inside Debian and thus (hopefully) included in the metapackage dependencies. This should be followed by the projects which has unofficial packages outside Debian. A last group should list all not yet packaged projects which are interesting for the Custom Debian Distribution. These groups are using a certain green-yellow-red color code.

alphabetically Inside each of these three groups an alphabetical order is proposed to gain some consistency between all Custom Debian Distributions.

The users who are visiting the web pages will like this comfortable overview ...

9.2.2 Automatically graphing the metapackage dependencies for web pages

The new `cddtk` contains a tool which allows browsing a `Packages` file for all packages that are listed in the dependency list of a package.

```
pkgtool -p UNCOMPRESSED_PACKAGES_FILE pdeps LIST_OF_PACKAGE_NAMES
```

This could be used as a start to build the web-pages as suggested above automatically - at least for the packages which are just at the Debian-Mirror. For the unofficial packages and not yet existing packages a fake-`Packages` following the same syntax could be created and processed with the same tool.

9.3 Debian Package Tags

Debian Package Tags (http://deb-usability.alioth.debian.org/debtags/) is a work to add more metadata to Debian packages. At the beginning it could be seen as a way to allow to specify multiple sections (called "tags") per package where now only one can be used.

However, the system has evolved so that tags are organised in "facets", which are separate ontologies used to categorise the packages under different points of view.

This means that the new categorisation system supports tagging different facets of packages. There can be a set of tags for the "purpose" of a package (like "chatting", "searching", "editing"), a set of tags for the technologies used by a package (like "html", "http", "vorbis") and so on.

Besides being able to perform package selection more efficiently by being able to use a better categorisation, one of the first outcomes of Debian Package Tags for CDDs is that every CDD could maintain its own set of tags organised under a "facet", providing categorisation data which could be used by its users and which automatically interrelates with the rest of the tags.

For example, Debian-Edu could look for "edu::administration" packages and then select "use::configuring". The "edu::administration" classification would be managed by the Debian-Edu people, while "use::configuring" would be managed by Debian. At the same time, non

Debian-Edu users looking for "use::configuring" could have a look at what packages in that category are suggested by the Debian-Edu community.

It is not excluded that this could evolve in being able to create a Custom Debian Distribution just by selecting all packages tagged by "edu::*" tags, plus dependencies; however, this option is still being investigated.

Please write to the list <deb-usability-list@lists.alioth.debian.org> for more information about Debian Package Tags or if you want to get involved in Debian Package Tags development.

9.4 Enhancing basic technologies regarding Custom Debian Distributions

In section 'Future handling of metapackages' on page 32 several issues where raised how handling of metapackages should be enhanced.

Regarding to building metapackages for all Custom Debian Distributions consistently it might make sense to use the following approach:

The method how Debian-Edu currently builds its metapackages from a kind of *database* (in the `tasks` directory of the source) was generalised in the packages `cdd-dev` ('Package `cdd-dev`' on page 59) and `cdd-common` ('Package `cdd-common`' on page 61). This approach definitely needs some enhancements to fit the needs of all Custom Debian Distributions. It might be a good idea to maintain a more general kind of database than this `tasks` directory approach currently represents for each Custom Debian Distribution. From this database the control files for all metapackages could be built on demand to build the necessary files of the `debian` directory in the package build process dynamically. The extra plus would be that it would be easy to build tools which parse this database to generate docs and websites dynamically. It would drastically reduce the amount of work for keeping the project related web sites up to date if this could be done automatically. Some tools like the following might be easily done to support maintenance and documentation of the metapackages:

```
build_cdd-package med bio
build_cdd-package junior toys
build_cdd-package education [all]

build_cdd-wml-template nonprofit <foo>
build_cdd-wml-template demudi    <bar>

cdd-package-info.php?cdd=<foo>&pkg=<bar>
```

If the database structure is well thought (perhaps using XML or by stealing the format of other databases which are usually used in Debian) not really hard to implement.

Last but not least the special configuration issue has to be addressed. In general developers of metapackages should provide patches for dependent packages if they need a certain configuration option and the package in question does feature a `debconf` configuration for this case. Then the metapackage could provide the needed options by pre-seeding the `debconf` database while using very low priority questions which do not came to users notice.

If the maintainer of a package which is listed in a metapackage dependency and needs some specific configuration does not accept such kind of patch it would be possible to go with a `cfengine` script which just does the configuration work. According to the following arguing this is no policy violation: A local maintainer can change the configuration of any package and the installation scripts have to care for these changes and are not allowed to disturb these adaptations. In the case described above the `cfengine` script takes over the role of the local administrator: It just handles as an "automated-`cfengine`-driven-administrator-robot".

If there is some agreement to use `cfengine` scripts to change configuration - either according to `debconf` questions or even to adapt local configuration for Custom Debian Distribution use in general - a common location for this kind of stuff should be found. Because these scripts are not configuration itself but substantial part of a metapackage the suggestion would be to store this stuff under

```
/usr/share/cdd/#CDD#/#METAPACKAGE#/cf.#SOMETHING#
```

There was another suggestion at the Valencia workshop: Make use of `ucf` for the purpose mentioned above. This is a topic for discussion. At least currently Debian-Edu seems to have good experiences with `cfengine` but perhaps it is worth comparing both.

A further option might be Config4GNU (http://freedesktop.org/Software/CFG) from freedesktop.org but it is not even yet packaged for Debian.

9.5 Building Live CDs of each Custom Debian Distribution

The first step to convince a user to switch to Debian is to show him how it works while leaving his running system untouched. Knoppix (http://www.knoppix.org/) - *the "mother" of all Debian-based live CDs* - is a really great success and it is a fact that can not be ignored that Debian gains a certain amount of popularity because people want to know what distribution is working behind the scenes of Knoppix.

But Knoppix is a very common demonstration and its purpose is to work in everyday live. There is no room left for special applications and thus people started to adopt it for there special needs. In fact there exist so many Debian based Live CDs that it makes hardly sense to list them all here. The main problem is that most of them containing special applications and thus are interesting in the CDD scope are out of date because they way the usually were builded was a pain. One exception is perhaps Quantian (http://dirk.eddelbuettel.com/quantian.html) which is quite regularly updated and is intended for scientists.

The good news is that the problem of orphaned or outdated Live CDs can easily solved by debian-live and the `live-helper`. This package turns all work to get an up to date ISO

image for a Live CD into calling a single script. For the CDD tools this would simply mean that the tasks files have to be turned into a live-helper input file and the basic work is done. This will be done in a future cdd-dev version.

9.6 New way to distribute Debian

This section is kind of "Request For Comments" in the sense that solid input and arguing is needed to find out whether it is worth implementing it or drop this idea in favour of a better solution.

At Open Source World Conference in Malaga 2004 there was a workshop of Debian Developers. Among other things the topic was raised how the distribution cycle or rather the method of distribution could be changed to increase release frequency and to better fit user interests.

There was a suggestion by Bdale Garbee <bdale@gag.com> to think about kind of sub-setting Debian in the following way: Debian developers upload their packages to `unstable`. The normal process which propagates packages to `testing` and releasing a complete `stable` distribution also remains untouched. The new thing is that the package pool could be enhanced to store more package versions which belong to certain subsets alias Custom Debian Distributions which all have a set of `tested inside the subset` distribution which leads to a `stable` subset release. The following graph might clarify this:

```
DD -> unstable   -->  testing  -->  stable
         |
         +--->  CDD_A testing  -->  stable CDD_A
         |
         +--->  CDD_B testing  -->  stable CDD_B
         |
         +--->  ...
```

where `CDD_A` / `CDD_B` might be something like `debian-edu` / `debian-med`. To implement this sub-setting the following things are needed:

Promotion rules There was a general agreement that technical implementation of this idea in the package pool scripts / database is not too hard. In fact at LinuxTag Chemnitz 2004 Martin Loschwitz <madkiss@debian.org> announced exactly this as "nearly implemented for testing purpose" which should solve the problem of outdated software for desktop users as a goal of the `debian-desktop` project.

Reasonable subsets Once the promotion tools are able to work with sub-setting, reasonable subsets have to be defined and maintained. A decision has to be made (if this will be implemented at all) whether this sub-setting should be done according to the Custom Debian Distribution layout or if there are better ways to find subsets.

BTS support The Bug Tracking System has to deal with different package versions or even version ranges to work nicely together with the sub-setting approach.

Security As a consequence of having more than only a single `stable` each CDD-team has to form a security team to care for those package versions that are not identically with the "old" `stable`.

A not so drastically change would be to find a *common* set of packages which are interesting for all Custom Debian Distributions which will obtained from the "releasable set" of testing (i.e. no RC-bugs). This would make the structure above a little bit more flat:

```
DD -> unstable --> testing --> releasable --> stable
                                    |
                                    +--->      stable CDD_A
                                    |
                                    +--->      stable CDD_B
                                    |
                                    +--->  ...
```

A third suggestion was given at Congreso Software Libre Comunidad Valenciana:

```
             testing_proposed_updated
                        |
                        |
                        v
DD -> unstable --> testing --> stable
                        |
                        +--->    stable CDD_A
                        |
                        +--->    stable CDD_B
                        |
                        +--->  ...
```

The rationale behind these testing backports is that sometimes a Custom Debian Distribution is able to reduce the set of releasable architectures. Thus some essential packages could be moved much faster to testing and these might be "backported" to testing for this special Custom Debian Distribution. For instance this might make sense for Debian-Edu where usually i386 architecture is used.

All these different suggestions would lead to a modification of the package pool scripts which could end up in a new way to distribute Debian. This might result from the fact that some Custom Debian Distributions need a defined release cycle. For instance the education related distributions might trigger their release by the start-end-cycle of the school year. Another reason to change the package pool system is the fact that some interested groups, who provide special service for a certain Custom Debian Distribution, would take over support only for the subset of packages which is included in the metapackage dependencies or suggestions but they refuse to provide full support for the whole range of Debian packages. This would lead to a new layout of the file structures of the Debian mirrors:

```
debian/dists/stable/binary-i386
                   /binary-sparc
                   /binary-...
            /testing/...
            /unstable/...
debian-CDD_A/dists/stable/binary-[supported_architecture1]
                         /binary-[supported_architecture2]
                         /...
                  /testing/...
debian-CDD_B/dists/testing/...
                  /stable/...
...
pool/main
    /contrib
    /non-free
```

To avoid flooding the archive with unnecessarily many versions of packages for each single Custom Debian Distribution a common base of all these Custom Debian Distributions has to be defined. Here some LSB conformance statement comes into mind: The base system of all currently released (stable) Custom Debian Distributions is compliant to LSB version x.y.

Regarding to security issues there are two ways: Either one Custom Debian Distribution goes with the current stable Debian and thus the `Packages.gz` is just pointing to the very same versions which are also in debian/stable. Then no extra effort regarding to security issues is need. But if there would be a special support team which takes over maintenance and security service for the packages in one certain Custom Debian Distribution they should be made reliable for this certain subset.

This reduced subset of Debian packages of a Custom Debian Distribution would also make it easier to provide special install CDs at is it currently done by Debian-Edu.

Appendix A

Description of development tools

A.1 Package cdd-dev

If metapackages are builded using the tools inside the cdd-dev package it can be ensured that the resulting metapackages will work nicely with the same version of cdd-common package. The goal is to keep necessary changes for the source of the metapackages of a Custom Debian Distribution as low as possible when the version of the cdd source package changes. Thus it is strongly recommended to use the tools described below.

The usage of the tools in the cdd-dev package might introduce a versioned dependency in the <*cdd*>-common package from which all other metapackages of the *CDD* in question will depend. This <*cdd*>-common package instantiates the *CDD* in the common registry for all CDDs in /etc/cdd.

The best idea to use the tools provided by the cdd-dev is to put a Makefile into the build directory containig one single line

```
include /usr/share/cdd-dev/Makefile
```

(see /usr/share/doc/cdd-dev/examples/Makefile). Using this Makefile all tools that were contained in cdd-dev package versions before 0.4. These tools are moved to /usr /share/cdd-dev/ because there is no need to call them directly. Here is a list of the make targets.

A.1.1 CDD-tasks.desk

This target is the description file that is used in tasksel to enable selecting the tasks belonging to the CDD. The file will be moved to the *cdd*-tasks. All information is obtained from the single task files in the tasks directory of the CDD source.

A.1.2 debian/control

The debian/control file of a CDD metapackage source archive is auto generated out of dependencies that are specified in so called tasks files. The rationale behind this is to enhance flexibility about changes inside the Debian package pool where new packages might appear and others might be renamed. The tasks just define which dependencies the CDD maintainer group wants to be fulfilled and the script cdd-gen-control verifies whether these dependencies exist in the specified package pool and create the debian/control file according to the available packages. This does not only work for the Debian package pool containing the distributions stable, testing and unstable. You can even build your metapackages against a different package pool of a Debian based distribution. This is for instance used to create the SkoleLinux metapackages.

The syntax of the tasks files which serve as the central database for the information in the debian/control file is defined by RFC822. Some of the tags were mentioned formerly in 'Existing and prospective packages' on page 45 to explain how the tasks files can be used to create the web sentinel pages. Now the tags that influence the creation of the debian /control file are given.

Depends Will be turned to "Recommends" in the resulting debian/control file. The rationale behind this is to enable installing the metapackage even if a package belonging to this task is missing for whatever reason. It also allows finally to remove the metapackage. This makes even more sense since apt-get considers "Recommends" as default installation targets.

Recommends The packages that are listed as "Recommends" in the tasks file should be installed on the machine where the metapackage is installed and which are needed to work on a specific task.

Suggests Use "Suggests" for packages of lesser importance that might be possibly useful, or non-free packages.

If a package is not available in the package pool of the target distribution when creating the debian/control file inside the meta package source archive any "Depends" or "Recommends" are turned into "Suggests" to enable a flawless installation of the metapackage. Packages that are specified with "Suggests" will not be taken over to the tasksel control file (CDD-tasks.desk, see 'CDD-tasks.desk' on the preceding page) but only to the list of suggested packages of the according metapackage.

Ignore The "Ignore" key can be used as kind of "Soft-Suggests" to put a package on the radar of the CDD. Packages that are tagged with Ignore will not be taken over into the list of dependencies inside the debian/control file of the resulting metapackage neither to the CDD-tasks.desk control file for tasksel but will be taken over onto the installation medium of a CDD in case there is some space left. This key becomes especially important for specifying not yet packaged software that might be packaged in the future (prospective packages). This is explained in detail in the paragraph about the web sentinel (see 'Existing and prospective packages' on page 45).

Avoids The "Avoids" key specifies existing packages that will be listed in the the `debian/control` file as "Recommends" or "Suggests" but, should not go to a installation medium (CD, DVD, etc.) that might be produced by the CDD. A reason to avoid a package might be that it belongs to the non-free section.

A.1.3 `cdd-clean-helper(1)`

NAME `cdd-clean-helper` - cleans up debian directory in a metapackage building tree

SYNOPSIS `cdd-clean-helper`

DESCRIPTION This script can be used in `debian/rules` file to revert the changes which were done by `cdd-install-helper(1)` to get a clean packaging tree. Using this helper ensures that the `debian/rules` file does not needed to be changed if there are changes in the `cdd-dev` package.

EXAMPLES For the usage of this tool just have a look at the `debian-med` source package.

AUTHOR Andreas Tille `<tille@debian.org>`.

A.1.4 Apt `sources.list` files in `/etc/cdd/`

These files are used by `cdd-gen-control(1)` to build valid `debian/control` files that contain only available packages in their dependencies. This enables building meta packages for `stable`, `testing`, `unstable` or even a completely different distribution that has valid `sources.list` entries. The file `/etc/cdd/control.list` is used as default for `cdd-gen-control(1)` and usually is a symbolic link (see `ln(1)`) to `sources.list.`*distribution*. It might be changed using the `-s`*dist* option of `cdd-gen-control(1)`.

TODO: *Either parse the available `/etc/apt/sources.list` or use a sane `debconf` question to use the "nearest" mirror.*

A.1.5 Templates in `/usr/share/cdd/templates`

The directory `/usr/share/cdd/templates` contains templates that can be used to build a <*cdd*>`-common`, which uses the tools that are contained in the `cdd-common` package, and are useful to manage <*cdd*> user groups (see 'User roles' on page 33).

A.2 Package `cdd-common`

This package creates a common registry for all CDDs in `/etc/cdd`. Each CDD should put the files that are used into a subdirectory named like the CDD of `/etc/cdd`. The `cdd-common` package installs a common configuration file `/etc/cdd/cdd.conf`, which can be used to influence the behaviour of the tools described below.

A.2.1 `cdd-role(8)`

NAME `cdd-role` - add/remove roles in registered Custom Debian Distribution

SYNOPSIS `cdd-role` *add*|*del CDD* [*Role*]

DESCRIPTION Add/remove (register/unregister) *Role* for the specified *CDD*. If *Role* is not specified, it's assumed to be named like *CDD*.

OPTIONS ***CDD*** A registered custom distribution in /etc/cdd, for example one of `med`, `junior`, `desktop`, `edu` or `demudi`

AUTHOR Andreas Tille <`tille@debian.org`>, Cosimo Alfarano <`kalfa@debian.org`>.

A.2.2 `cdd-update-menus(8)`

NAME `cdd-update-menus` - add menu of metapackage to all Custom Debian Distribution users

SYNOPSIS `cdd-update-menus` [*–cdd CDD* | *–user user*]

DESCRIPTION cdd-update-menus behaves differently depending on who run the command:

If it is called by a user, it adds, and keeps updated, menu entries for the user who runs it.

If it is called by root, it adds and keeps updated user's menu entries (see menu package for users' menus) for all users who belong to the group of the specified Custom Debian Distribution, or only for a specified user, depending on which parameter is passed to the script.

OPTIONS ***CDD*** one of the installed CDDs, listed in /etc/cdd/, for example (if installed: `med`, `junior`, `desktop`, `edu` or `demudi`

user system user

AUTHOR Andreas Tille <`tille@debian.org`>, Cosimo Alfarano <`kalfa@debian.org`>.

A.2.3 `cdd-user(8)`

NAME `cdd-user` - add/remove user to Role of a registered Custom Debian Distribution

SYNOPSIS `cdd-user` *add*|*del CDD user* [*Role*]

DESCRIPTION Add/remove user to a *Role* of the specified *CDD*. If *Role* is not specified, it's assumed to be named like *CDD*

OPTIONS ***CDD*** A registered custom distribution in /etc/cdd, for example one of `med`, `junior`, `desktop`, `edu` or `demudi`

user user to add

Role the role in the *CDD* that *user* will assume

AUTHOR Andreas Tille <tille@debian.org>, Cosimo Alfarano <kalfa@debian.org>.

A.2.4 cdd.conf(5)

NAME cdd.conf - configuration for Custom Debian Distribution registry

DESCRIPTION This file is sourced from shell scripts inside the Custom Debian Distribution package cdd-common and thus it has to follow shell syntax. The variables that are set inside this configuration file can be overriden by special CDD configration files /etc/cdd/<>*cdd*>/<>*cdd*>.conf for each single CDD.

SYNTAX The following variables can be set:

DBBACKEND Set the backend for the user role management system. Currently the only implemented role management system is unixgroups but others might be implemented later. Unsetting this variable leads to use no roles at all.

UPDATEUSERMENU If this is set to yes, the user menus of meta packages can be created automatically at install time of the package if the postinst script of the package allows this. It is suggested to use this option in the specific configuration files of a special Custom Debian Distribution that override the settings of the general configuration file.

SHAREDIR Set the base directory for the user role management system. While this is more or less a feature for debugging this might be also used otherwise.

DRYRUN This variable can be set for debugging. Normally it should be left unset (*NOT* set to false or anything else!). If set to true a dry run of the tools is performed or echo DRYRUN: would print debugging information.

DEBUG If set to 1 debugging mode is switched on.

SEE ALSO cdd-role(8), cdd-update-menus(8), cdd-user(8)

AUTHOR Andreas Tille <tille@debian.org>, Cosimo Alfarano <kalfa@debian.org>.

A.3 Working with svn

Sometimes it might be interesting for developers to check out the latest code of the CDD tools or a special CDD code for the meta packages. In 'Establishing and using communication platforms' on page 49 the directory layout of the svn-directory was described. How to derive the Debian packages from this layout?

Checkout For the CDD tools

```
svn checkout svn+ssh://username@svn.debian.org/svn/cdd/cdd/trunk/cdd
```

or for the Custom Debian Distribution *cdd-name*

```
svn checkout svn+ssh://username@svn.debian.org/svn/cdd/projects/cdd-n
```

Build source package Change into the created directory and type

```
make -f debian/rules get-orig-source
```

This creates a `tar.gz` source archive of the packages you want to build. For your personal comfort you can create a file `Makefile` in your package source directory containing

```
#!/usr/bin/make -f
include /usr/share/cdd-dev/Makefile
```

Which enables you to simply say

```
make dist
```

to create the source tarball.

Build Debian packages Unpack the created source tarball and proceed like you build Debian packages normally.

The current Debian-Med packages provide a working example how to use the tools described below.

Appendix B

Quick intro into building metapackages

There are several descriptions available how to build Debian packages in general. The main resource might be the repository of Debian packaging manuals (http://www.debian.org/doc/packaging-manuals/) (especially developers reference chapter 6, best packaging practices (http://www.debian.org/doc/packaging-manuals/developers-reference/ch-best-pkging-practices.en.html)). There are several external packaging HOWTOs for example the one from Joe 'Zonker' Brockmeier (http://www-106.ibm.com/developerworks/linux/library/l-debpkg.html).

B.1 Defining dependencies for metapackages

This howto describes the building of metapackages by using the `cdd-dev` package. It is perfectly possible to build a metapackage as any other normal Debian package but this HOWTO has the only purpose to describe the profit you might gain by using these tools.

```
~> cp -a /usr/share/doc/cdd-dev/examples/tasks .
~> cat tasks/README
~> edit tasks/task1
Description: short description
 long description as in any debian/control file

Depends: dependency1, dependency2, ...
```

For each metapackage this skeleton of a `debian/control` entry is needed. All necessary information is available in the directory `/usr/share/doc/cdd-dev/examples/tasks`.

B.2 The packaging directory

To build any Debian package you always need a directory named `debian`, which contains a certain set of files. The package `cdd-dev` provides a complete set of example files that only

have to be copied and after editing some place holders are ready to use.

```
~> cp -a /usr/share/doc/cdd-dev/examples/debian .
~> cat debian/README
~> edit debian/control.stub
```

Now the variables in the file `control.stub` change the variables named *_CDD_*, *_MAINTAINER_* etc. to match the names of the Custom Debian Distribution to be built. Please note that the file `debian/control` is and has to be a symbolic link to `control.stub` to let the `cdd-dev` tools work.

```
~> edit debian/rules
```

Also in the `debian/rules` the name of the Custom Debian Distribution has to be inserted where the template contains *_CDD_*. Depending from the way the `sources.list` should be scanned the options for the `gen-control` call can be adjusted.

You have to build the tarball using the command

```
~> make -f debian/rules get-orig-source
```

For your comfort you might like to create a file `Makefile` containing

```
#!/usr/bin/make -f
include /usr/share/cdd-dev/Makefile
```

which enables you to simply use

```
~> make dist
```

to build the source tarball. This tarball has to be moved to a location where the metapackages will be built. Unpack the tarball there and start the build process using for instance

```
~> debuild
```

That's all for the very simple case when the metapackages should not contain user menus. Even if user menus are suggested they are not necessary. The following paragraphs describe how to use the `cdd-dev` tools to support these menus.

B.3 The common metapackage

The creation of a common package is optional, but suggested, because it adds some special features like menus, user groups, and probably more in the future. It is automatically built by `cdd-install-helper`, which is called in `debian/rules`, if the `common` directory exists. The easiest way to create this is as follows:

```
~> cp -a /usr/share/doc/cdd-dev/examples/common .
~> cat common/README
~> edit common/conf common/control common/common.1
```

The variables (*_CDD_*) in these three files have to be adjusted to the name of the Custom Debian Distribution in question. This *cdd*-`common` cares for the initialisation of the role based menu system and might contain adjustments of the general configuration inside the `cdd-common`.

If the metapackage *cdd*-`common` will be created according to these rules all other metapackages will depend automatically from this common package. For the friends of `auto-apt`, a helper `/usr/bin/`*<metapackage-name>* will be installed as well, which just prints some information about the meta package. All in all, the usage of the common package is strongly suggested to have a common registry for stuff like user roles and possibly other things that will be implementd in the future.

B.4 The metapackage menus

As explained in 'User menu tools' on page 34 the metapackages can contain user menus. This optional feature can be implemented easily by using the template from the `cdd-dev` in the following way:

```
~> cp -a /usr/share/doc/cdd-dev/examples/menu .
~> cat menu/README
~> edit menu/task1
 Edit the example to legal menu entries of the
 dependencies of this metapackage
~> cp menu/task1 menu/<metapackage name>
```

A menu file for each task should be created containing valid menu entries for each dependant package. The easiest way to obtain those menu entries is to simply copy the original menu entry files that are contained in the packages on which the metapackage will depend. The only thing that has to be changed in these menu entries is the `package` field, which has to be changed from `<dependent package>` to *cdd-task*. All other entries might remain unchanged. This is a good point to check whether the menu entries of the packages you depend from are formated nicely and print the necessary information (for instance make use of

"hints"). Here the metapackage maintainer has a good chance for quality assurance work, which is also part of the Custom Debian Distributions issue.

In principle these menu items could be created automatically either at metapackage build time or even better in the `postinst` script of the metapackage because it is granted that the needed menu files are installed on the system, which is not really necessary on the metapackage build machine. This might be implemented in later versions of `cdd-dev`. Currently the policy is that we like to have a little bit of control about the menu entries for the quality assurance issue mentioned above. Last, but not least, there are packages that do not provide a menu entry. If this is the case because the package maintainer just forgot it a bug report should be filed. On the other hand, there are packages with programs that provide a command line interface that does not allow a reasonable menu entry. A solution for this case is provided in the next paragraph.

B.5 Menu for any dependency

The idea of the metapackage menu is to provide the user with easily viewable traces of any installed package that helps solving everyday tasks. So if there are packages that do not contain a menu, a screen with relevant documentation should be provided in a viewer by the creator of the metapackage. Such documentation can be created using the following templates:

```
~> cp -a /usr/share/doc/cdd-dev/examples/docs .
~> cat docs/README
~> edit docs/task1/dep1
 Provide information about a package <dep1> that is
 a dependency of the metapackage <task1>, but does not
 contain a useful menu entry.
~> cp docs/task1/dep1 docs/task1/<dependent pkg>
~> cp -a docs/task1 docs/<metapackage name>
```

This ensures that our users become aware of all interesting packages on their system. The documentation files should contain hints to man pages to read, URLs that should be visited to learn more about the package or some short introduction how to get started.

Appendix C

Using the Bug Tracking System

C.1 How to ask for packages which are not yet included

A very frequently asked question in mailing list is, whether `program_xy` can be integrated into a Custom Debian Distribution. As long as there is an official package of this program it is an easy task. But mostly users ask for software which is not yet integrated into Debian.

There is a detailed description (http://www.debian.org/devel/wnpp/#l1) how anybody can ask for including a certain piece of software into Debian. It explains how to use the program `reportbug` for this purpose.

If you use wnpp sanely you can even tag this bug for the intended purpose to include it in a certain Custom Debian Distribution. This was described in a mail of Anthony Towns (http://lists.debian.org/debian-devel-announce/2005/09/msg00002.html) and the Debian wiki (http://wiki.debian.org/DebianScience/Sponsoring). It was also described in a mail of Ben Armstrong (http://lists.debian.org/debian-custom/2005/04/msg00017.html) and in short works like this:

- Add an usertag to an existing bug by sending to request@bugs.debian.org a mail containing

  ```
  user <email>
  usertag <bug number> + wnpp <metapackage name>
  ```

  , where email is the electronic address of the person or mailing list which coordinates the relevant Custom Debian Distribution.

  For instance if you want to tag an ITP with bug number #123456 for Debian-Med section biology you would send the following mail to `<request@bugs.debian.org>`:

  ```
  user debian-med@lists.debian.org
  usertag 123456 + wnpp med-bio
  thanks
  ```

Note: Because the search is case sensitive please always use lower case tags!

- To search for WNPP bugs of a certain Custom Debian Distribution just visit the URL

  `http://bugs.debian.org/cgi-bin/pkgreport.cgi?tag=wnpp;users=`*`email`*

C.2 How to report problems

Debian has a very useful Bug Tracking System (BTS) but unfortunately users seldom know about this fact and how to use it right. This is the reason why users sometimes become angry about errors because they do not know what to do next and just install a different distribution instead of trying to solve the problem.

A detailed explanation how to report errors (`http://www.debian.org/Bugs/Reporting`) is helpful in these cases. While the program `reportbug` fetches other reports from BTS before creating the bug report it is always a good idea to search http://bugs.debian.org/_package_ (`http://bugs.debian.org`) for known problems and probably suggested solutions before calling `reportbug`.