Custom Debian Distributions

Andreas Tille <tille@debian.org>

28 July 2004

Abstract

This paper is intended to people who are interested in the philosophy of Custom Debian Distributions and the technique which is used to manage those projects. It is explained in detail why these are no forks from Debian but reside completely inside the Debian GNU/Linux distribution and which advantages can be gathered by this approach. The concept of meta-packages 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 CVS repository at alioth.debian.org (`http://cvs.alioth.debian.org/cgi-bin/cvsweb.cgi/cdd/doc/common/?cvsroot=cdd`). 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) provides support for special user interests. They implement a new approach to cover interests of specialied users. Special users might be children, lawyers, medical stuff, visually impaired people etc. Of late several Custom Debian Distributions evolved. The common goal of those is to make installation and administration of computers for these target users as easy as possible and to fit our role as missing link between software developers and users well.

An illustrative explanation what Custom Debian Distributions are can be given by some kind of object oriented approach: If Debian as a whole is an object a Custom Debian Distribution is an instance of this object which 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 external or non-official which may create “derivative distributions” but do not belong to the responsibility of the Debian project.

The effort might fall into the same category as the Componentized Linux (http://platform.progeny.com/componentized-linux/) of progeny but there are certain differences which 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 which 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 which 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 pronounced GNU/Linux system. 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 which satisfy the needs of the target user group. There are general distributions which try to support all users and there are several specialised distributions that are targeting on a special group of users.

Distributors are those companies that are building these collections of software around GNU/Linux. GNU/Linux is Free Software and the user who buys a distribution pays for the service which the distributor is providing. These services are:

- Preparing a useful collection of software around GNU/Linux.
- Caring for smooth installation which the target user is able to manage.
- Provide software updated and security fixes.
- Writing documentation and translation 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, (RPM, DEB, 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 live in `/usr/share/doc/$package`."

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

It is this *adherance to policy* which causes a distribution to remain consistant 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/`) (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 right 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 - who are called *Maintainers* - are connected in a Web of trust (`http://people.debian.org/~edward/globe/earthkeyring/`) which is weaved by signing of 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 an other developer they sign their keys and in this way the web of trust is weaved.

2.3 Differences from other distributions

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

  1 Buying 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 and 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.

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 ignored).

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

One solution is provided by the `tasksel` package. It provides a reasonable selection of quite general tasks which should be solved with the computer which is intended to run Debian GNU/Linux. But this is not really fine grained and does not fit 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 handy collections of specific program packages but care for easy installation and configuration for the intended purpose.

To clarify a common misunderstanding: Custom Debian Distributions are **no fork** 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 which 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 how commercial GNU/Linux distributors try to address specific user interests with certain products.

The solution which Debian found for this problem is just called "**Debian Custom Distributions**".

3.2 Motivation

3.2.1 Profile of target users

The target user of a Custom Debian Distribution is either a specialist of a certain real live profession (i.e. doctor, lawyer, etc.), a person which has not (yet) gathered a certain amount

of computer knowledge (for instance children) or people with disabilities (visually or aurally impaired people).

The common attribute to these target users is less technical competence. These people do not even want to notice the computer - they just want to work on their real live job. Imagine a doctor who has to move the focus of interest from the patient to his stupid computer who does not really work as expected.

Because of the limited knowledge or time property the target user is not able to install upstream programs which means at the first place to find out, which software package might serve for a certain problem. The next step would be to download and install the program in question perhaps requiring a certain amount of configuration effort. This problem is nearly impossible for those users with limited technical competence and perhaps no knowledge of English language which prevents the user from understanding the installation manual.

The language barrier in this field is an important issue because we are aiming to everyday users who are not forced to learn English like Free Software developers for every day communication. So the installation process has to cope with as less as possible user interaction and this interaction has to be internationalised.

Furthermore the target user has no interest in administration of his computer. In short: The optimal situation would be that he would not even notice the existence of the computer and just focus to the concrete application which provides the functionality for the real live job.

Common to all groups of target users is their interest in defined subset of available Free Software. None of them wants to do researches nor has the time to search for the package which fits his interest. Instead the target user wants to see all stuff which is relevant for his own task immediately.

There is an absolute need for easy usage of the programs. This is not so much a concern of learning to use the software because if people want to make money with a certain piece of software they accept to spend a reasonable amount of time in learning. But a simple to learn environment is an extra plus and if you think of children as target users they just want to start without reading any technical stuff.

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

Last but not least we find certain requirements for each target user group. This can be different between different Custom Debian Distributions. For instance while a doctor has to protect his database against attacks of an external spy the chance of this event is quite low for a box of a child. So the Debian Junior project has more or less to care that the user itself does not disturb his own desktop while playing around with all these buttons and external attacks are not in the main focus. So we find a “defined security profile” for each single Custom Debian Distribution.

3.2.2 Profile of target administrators

In the field which should be covered by Custom Debian Distributions we have to face also some common things on the administrator side. Often they have a limited time frame to serve a huge lot of computers and thus they are happy about each simplification of administration process. The need for special adaptations for the intended purpose has to be reduced to a minimum.

So administrators are looking for time saving 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.

The problem of administrators is often that they are no experts on the special field their clients work on. So they need some specialist knowledge to explain their users the programs they will work on or at least need to be able to communicate nicely with the experts.

3.3 Status of specialised Free Software

Programs like a web server or an office suite 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 which have a wide use. This is different for specialised software.

- Specialist software is used only by a limited set of users (i.e. the specialist). There are a set of software tools which work perfectly in the environment where they were developed. If the developers catch the idea of Free Software and just release the software which was grown up in its initial environment people who like to work with the same tool run into trouble to get it working in their environment because normally the developers did not really care about a robust installation process and relay on their special environment. Installation instructions are often badly written if existing at all. But this problem can be easily solved by shipping the software as binary package - exactly what inclusion in Debian means.
- The trouble often continues in the maintenance of the installed software.
- When it comes to the usage of the specialist software it shows up that it perfectly fits to the needs of the developer for its special needs but in most cases does not comply with ergonomic standards of user interfaces.
- Several existing programs which might be useful for specialist are not really free in the sense of the Debian Free Software Guidelines (DFSG) (`http://www.debian.org/social_contract#guidelines`). Programs which 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 amount 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 which 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 which 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 show 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 which do not use transactions are used to store patent data which is completely unacceptable. Other programs use web clients as front-end which is not really good for quick (mouse-less) usage.

3.4 General problem

Free Software development is kind of evolutionary process. It needs a critical mass of supporters which 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 . . . so

Debian is the missing link between upstream developers and users.

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
- Make Debian an OS that children of all ages will *want* to use.
- Making it so nice that they like it.
- Making it a playground for children experiments.

Main target are young children, teens should become comfortable with using Debian without any special modifications.

Debian-Jr was the first Custom Debian Distribution - in fact the idea of Custom Distributions was born (these days under the name Debian internal projects but this name was changed because it was also taken for some more technical related projects like IPv6 etc.).

Debian-Jr not only provides games but also cares for their quality. So games which are regarded as "not really for children of this age" are omitted. Moreover choices are made which packages are best suitable to enable children to "work" (simple text processing, web browsing and drawing).

4.2 Debian-Med: Debian in Health Care

Start beginning of 2002

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
- Build an integrated software environment for all medical tasks.
- Care especially for the quality of program packages in the field of medicine which are integrated in Debian yet.
- Building packages of medical software which are missing inside Debian and inclusion of those packages.
- Caring for a general infrastructure for medical users.
- 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

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
- Make Debian the best distribution available for educational use.
- Federate many initiatives around education (which are partly based on forks of Debian).
- Continue the internationalisation efforts of SkoleLinux.
- Focus on easy installation in schools
- Cooperate with other education-related projects (like SEUL/edu (http://www.seul.org/edu/), Ofset (http://www.ofset.org/), KdeEdu (http://edu.kde.org/)).

This project started with the intention to bring back a fork from Debian which was done by some people in France. Because they had some time constraints the people who did 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 which are intended to be used in schools. For instance there are:

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

LliureX (http://www.lliurex.net/) 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/) Not only related to education but might try also to integrate back their stuff back to Debian.

4.4 DeMuDi: Debian Multimedia Distribution

Start Currently not announced as official Custom Debian Distribution but part of the Agnula (http://www.agnula.org/) project (founded by European Community) (since 2000)

Initiator Marco Trevisani <marco@centrotemporeale.it>

Goals
- Bringing back a fork from Debian.
- Oriented toward music and multimedia.
- Make GNU/Linux a platform of choice for the musician and the multimedia artist

The initiator is not yet Debian developer but you are able to work on Debian without being an official developer.

4.5 Debian-Desktop: Debian GNU/Linux for everybody

Motto: "Software, which 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
- Try to build the best possible operating system for home and corporate workstation use.
- Desktops like Gnome and KDE should coexist well in Debian and work optimal.
- Easy usage for beginners without restrictions of flexibility for experts.
- Easy installation and configuration (hardware-detection).
- Internationalisation

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 which 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 50.

4.6 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
- Build a complete system for all tasks in legal practice.
- Add value by providing customised templates for lawyers to existing packages like OpenOffice.org, SQL-Ledger or sample database schema for PostgreSQL.

The word *lex* originates from Latin word for law.

4.7 Debian-NP: Debian GNU/Linux for Non-profit Organisations

Start July 2003

URL Debian-NP (http://www.debian.org/devel/debian-nonprofit)

Initiator Benjamin 'Mako' Hill <mako@debian.org>

Goals
- Address requirements of small non-profit organisations.
- Prepare Debian for desktop use in non-profit organisations.
- Provide software which handles fund raising issues.
- Handling of membership lists.
- Interfaces for conference organisation

Non-profits are often familiar with Free Software.

4.8 Debian Accessibility Project

Debian for blind and visually impaired people

Start February 2003

Initiator Mario Lang <mlang@debian.org>

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

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

- Make Debian accessible to people with disabilities
  - 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)
  - Making text-mode interfaces available.
  - Providing screen reader functionality during installation.

4.9 Debian Enterprise

Debian GNU/Linux for Enterprise Computing

Start End of 2003

URL Debian-Enterprise (http://debian-enterprise.org/)

Initiator Zenaan Harkness <zen@iptaustralia.net>

Goals
- Apply the UserLinux Manifesto
- Establishes the benchmark in world class Enterprise operating systems engineered within an industry driven shared-cost development model
- Vigorously defends its distinctive trademarks and branding
- Develops extensive and professional quality documentation
- Provides engineer certification through partner organisations
- Certifies the Debian Enterprise GNU/Linux operating system to specific industry
- Pre-configuration of server tasks
- Install time options regarding to the intended purpose

4.10 Other possible Custom Debian Distributions

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

Debian-eGov Government issues, administration, authorities office, accounting

Office Cover all office issues

Accounting Integrate accounting systems into Debian

Geography Geographical information systems (GIS)

Biology perhaps taking over some stuff from Debian-Med

Physics Simulation

Mathematics even have a live CD - see Quantian in 'Building Live CDs of each Custom Debian Distribution' on page 46

??? There are a lot off more Custom Debian Distributions thinkable

Chapter 5

Distributions inside Debian

5.1 To fork or not to fork

There are many distributions which 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 Lindows (http://www.lindows.com/), Libranet (http://www.libranet.com/) or Xandros (http://www.xandros.com/). etc. there is no other choice than forking because these companies normally add some stuff which is non-free. While Custom Debian Distributions might be interesting in technical terms for those commercial distributions to make 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 which are as free as Debian but had certain reasons to do a fork. Most of these reasons existed in the past but are vanished now because Debian was becoming more flexible regarding to adaptations for special purposes. To increase this flexibility the Custom Debian Distributions approach was invented. Some examples of forks from Debian which probably are able to integrate back into Debian:

SkoleLinux (http://www.skolelinux.org) Mentioning SkoleLinux in the category of forks is more or less history because the SkoleLinux people did a really great job to enhance Debian for their own purpose (debian-installer) and now the Custom Debian Distribution Debian-Edu (see above) is equal to SkoleLinux. 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 European Community (in fact the first Free Software project which was founded by the EU at all) did a fork for the following reasons:

Technical Some special requirements for the kernel and configuration stuff. This is more or less solved in the upcoming Debian release

License When DeMuDi started not enough free programs of this field did exist. This situation is better now.

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

This shows that there is no real need for a fork any more and in fact the organiser of the DeMuDi project is in contact to bring back all the necessary start. (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 which will be explained below. So it might be worth considering to follow 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 .

If developers of a non-commercial fork consider to integrate back into Debian in the form of a Custom Debian Distribution it might occure that there topic is covered by a Custom Debian Distribution yet. For instance this would be the case for LinEx which has absolutely the same target like Debian-Edu. On the other hand some special adaptations might be necessary to fit the requirements of the local educational system. These specific changes which 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 base system, installer etc. up to date with the current Debian development. It would be more sane to send patches which might be solve some special requirements to Debian instead of maintaining a complete Debian tree containing these patches.

Debian is well known for its strong focus to security. Security is mainly based on manpower and knowledge. So the best way to deal with security issues would be to base onto 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 the critical mass which was explained in 'General problem' on page 10.

Larger Free Software projects need certain infrastructure like HTTP-servers, FTP-servers (both with mirrors), a bug tracking system etc. Normally it is hard to build up an infrastructure which is 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 which is integrated in Debian has a good chance to become public on the back of Debian if potentially interested people just notice that there is something which 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 necessary. This fact was 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 would have 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 infra structure of HTTP-servers, FTP-servers with mirrors, mail-servers, LDAP directory of developers with a strongly weaved web of trust (GPG-key signing) for free.

5.1.5 Enhancing Debian

By making changes to some packages to make them fit for the target user group the overall quality of Debian can be enhanced. In this way enhancing Debian by making them more user friendly is a good way for the community to bring back something to Debian. It would be a shame if somebody would refuse all the advantages from keeping a project inside Debian and tries to cope with the disadvantages because he just does not know how to do it right and that it is normally easy to propagate changes to Debian. For instance see 'How to ask for packages which are not yet included' on page 53 how you can ask for including a certain piece of software into 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. This means that most of the Developers are flexible enough to care for their own interests. So Debian is not bound on commercial interest.

Who is afraid about this amount of freedom of every single developer should know that there are very strict rules (the policy) which glue all things together.

There is one common interest of each individual developer to get the best operating system for himself. This way people with similar interests and tasks profit from the work of single developers and if those users work together with the developers by sending patches or bug reports for further enhancement Debian can be prepared also for special tasks.

For instance developers have children in real life or work in some special fields of work and so they try to make the best system for their children (Debian-Jr / Debian-Edu) or their field of work (Debian-Med, Debian-Lex, ...).

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 build their distribution to gain a big market share. It is hardly possible to earn much money by targeting at children PC at home and distributions comparable to Debian-Junior will not be built by commercial distributors.

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 and is the reason for the flexibility of Debian which makes it adaptable for any purpose. In the Debian world this kind of community is called "***Do****ocracy*" - the one who does rules.

Chapter 6

Technology

6.1 Meta packages

6.1.1 Meta package definition

A meta package is a Debian package which contains:

- Dependencies from other Debian packages
  - Depend from packages to do a certain task
  - Recommend further interesting packages
  - Suggest others or non-free packages
- User menu entries (recommended)
  - in `/etc/cdd/<cdd>/menu/<pkg-name>`
  - maintained via role based tools
- `debconf` questions or pre-seeding (optional)
- `cfengine` scripts (optional)
- Special meta package: `<cdd>-common`

Meta packages are small packages with nearly no contents. The main feature of this type of packages is their dependencies from other packages. The naming scheme of meta packages <*cdd*>-<*task*> where <*cdd*> stands for the shortcut of a Custom Debian Distribution, i.e. `junior` for Debian-Jr or `med` for Debian-Med, and <*task*> means the certain task inside the Custom Debian Distribution.

Examples:

junior-puzzle Debian-Jr. Puzzles

debian-edu-config Configuration files for SkoleLinux systems

med-bio Debian-Med micro-biology packages

6.1.2 Collection of specific software

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

If the author of the meta package decided to include packages with similar functionality an easy comparison between software covering the same task is possible.

Moreover the installation of a meta package ensures that no package which is necessary for the intended task can be removed without explicit notice that also the meta package has to be removed.

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

All in all meta packages enable an easy installation from scratch and keep the effort for administration low.

6.1.3 Adapted configuration inside meta packages

Besides the simplification of installing relevant packages by dependencies inside meta packages these might contain special configuration for the intended task. This might either be accomplished by pre-feeding `debconf` questions or by modifying configuration files in a `postinst` script. It has to be ensured that no changes which have been done manually by the administrator will be changed by this procedure. So to say 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` of existing documentation, writing extra documentation like manpages etc. This complies with the statement that Custom Debian Distributions are focussed to interests of specialised users who have a big 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.org/) which has the goal to translate package descriptions. There are good chances to use this system also for other types of documentation which might be a great help for Custom Debian Distributions.

6.2 Handling of meta packages

In short: There are no special tools available to handle meta packages nicely. But there eare some tricks which 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. So 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 meta pa
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 development
med-tools - Debian-Med several tools
med-bio-contrib - Debian-Med micro-biology packages (contrib and non-fr
med-common - Debian-Med Project common package
med-cms - Debian-Med content management systems
```

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

grep-dctrl The program grep-dctrl is a grep for Debian package information which is helpful to find out something about packages matching a certain pattern:

```
~> grep-dctrl ': med-' /var/lib/dpkg/available | grep -v '^[SIMAVF]' |
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, debconf
Description: Debian-Med package for dental practice client

Package: med-bio
Depends: bioperl, blast2, bugsx, fastdnaml, fastlink, garlic, hmmer, nc
```

```
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 meta packag

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-common, galeo
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, med-commo
Description: Debian-Med packages for medical image development

Package: med-bio-contrib
Depends: clustalw | clustalw-mpi, clustalx, molphy, phylip, seaview, tr
Description: Debian-Med micro-biology packages (contrib and non-free)
```

This is as well as the `apt-cache` example **also a bit cryptic** and either not acceptable for end users.

`auto-apt` The program `auto-apt` is really cool if you are running a computer which was installed from scratch in a hurry and are sitting on a booth for some demonstration purpose. If you had no time to install all stuff you wanted to demonstrate just start `auto-apt` in the following manner and you will never face some missing files or programs …

```
~> 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  not upgrad
Need to get 0B/1263kB of archives. After unpacking 2008kB will be used.
Reading changelogs... Done
Selecting previously deselected package bugsx.
(Reading database ... 133094 files and directories currently installed.
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-licenses/G
/usr/share/doc/med-bio/copyright
```

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

The short conclusion here is: **There are no sophisticated tools which might be helpful to handle meta packages 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 meta packages in an elegant manner.

tasksel `Tasksel` is the Debian task installer and the first interface for package selection which 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 which prevent further Custom Debian Distributions from being included in the `tasksel` list because all dependencies of this task have to be solved on the first installation CD. This can not be accomplished for all Custom Debian Distributions 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 to handle Custom Debian Distributions and there could be some better support to handle meta packages more clever.

Short conclusion: **There are good chances to get meta packages handled nicely by the text based Debian package administration tools but currently this is not yet implemented.**

6.2.3 Graphical user interfaces

Debian *Woody* does not yet 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 were increased and so there are some usable options now.

`gnome-apt` This is the native Gnome flavour of graphical user interfaces to apt. It has a nice `Search` feature which can be found in the `Package` menu section. If you 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 meta packages 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 which supports Debian Package Tags (`http://deb-usability.alioth.debian.org/debtags/`).

`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 which 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 which 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 meta packages:

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 you get by this method is much larger than you would wish 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` which 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 meta packages

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 meta packages
- Overview about dependencies of these meta packages
- Enhancing tools like `aptitude`, `synaptic`, etc.
- Special `tasksel` section
- Web tools which keep meta package 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 which grep the apt cache directly for meta packages 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 software would be the only one which is presented in the

menu. This would allow the user to concentrate on his real world tasks instead of browsing funny menus.

To accomplish this a technique has to be implemented which 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 which is provided for exactly this group.

Strictly speaking it is not the best solution to conflate a configuration mechanism (which users see which menus) with access control (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 which 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 and there is no *practical* limit of the number of groups which a user can belong for the existing Custom Debian Distributions at the time beeing.

For 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 which 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 34) 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 meta package. 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 meta packages of a Custom Debian Distribution which will move all necessary files right into place if there exists a `menu` directory with the menu entries as described in '`cdd-install-helper(1)`' on page 32. Note, that the users `${HOME}/.menu` directory remains untouched.

Managing Custom Debian Distribution users with `debconf`

Using `cdd-install-helper(8)` (see '`cdd-install-helper(1)`' on page 32) it is very easy to build a *cdd*`-common` package which 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 meta package is more or less equal for each meta package. This was the reason to build a common source package `cdd` which builds into two binary packages

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

cdd-common This package provides some files which are common to meta packages of Common Debian Distributions especially those which were builded 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 28.

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

6.4.1 Package `cdd-dev`

If meta packages are builded using the tools inside the `cdd-dev` package it can be ensured that the resulting meta packages will work nicely with the same version of `cdd-common` package. The goal is to keep necessary changes for the source of the meta packages 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 meta packages 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 current Debian-Med packages provide a working example how to use the tools described below.

`cdd-gen-control(1)`

NAME `cdd-gen-control` - install menu and link to helper bin and according manpage

SYNOPSIS `cdd-gen-control`

DESCRIPTION The script `cdd-gen-control` parses the `tasks` directory for text files which have a similar syntax to `debian/control` files. Each text file is used as template for a *cdd-textfile_name* meta package and has to define the dependencies.

OPTIONS **`-c`** Create debian/control file using template `debian/control.stub`.

`-d` Turn on debugging mode.

`-a` Print all available packages.

`-e` Print excluded packages.

`-m` Print missing packages.

-s *dist* Without the -s option /etc/cdd/sources.list is used to verify which packages are available. You can specify one of stable, testing or unstable as argument or a complete path to a valid sources.list file.

AUTHORS Petter Reinholdtsen <pere@hungry.com>, Andreas Tille <tille@debian.org>

The interesting thing in this script is that it can be ensured that the resulting meta package can be installed in the target distribution. For instance it might be possible that for certain reasons a meta package should work together with the current Debian stable distribution. If the package is builded against a sources.list package which contains entries for stable and some newer packages are not yet available, these ones are not listed as dependencies but only as suggested packages. That way it is possible to provide meta packages for using a stable distribution using the same package source as for testing or unstable where new packages normally go.

cdd-install-helper(1)

NAME cdd-install-helper - install menu and link to helper bin and according manpage

SYNOPSIS cdd-install-helper

DESCRIPTION This script can be used in debian/rules file to install the user menu files to /etc/cdd/<*cdd*>/menu/<*package*>, a link for the helper script of every <*cdd*>-* package and the link to the manpage for this script.

menu If a directory menu exists in the building directory it checks for files named like meta packages (without the <*cdd*>- name prefix). These files should be valid menu files as they are provided for Debian packages in debian/menu. They will be copied to debian/<*pkg*>/etc/cdd/<*cdd*>/menu/<*pkg*> where tools like cdd-update-menus(8) will expect them.

A check will be performed whether there are text files named docs/<*pkg_without_cdd-prefix*>/<*dependency*>.txt where <*dependency*> is a package which is listed in the dependencies of the meta-package. These text files should provide reasonable information how to use this program in text form which can be viewed by a pager which is better than having no menu entry at all. A menu entry will be created which call the pager to this text file after checking whether this package is really listed in the dependencies.

common If there exists a file common/common and has a size greater than 0 a <*cdd*>-common package is builded. The file common/control was just used to build the appropriate debian/control file using the cdd-gen-control(1) tool.

docs The files which reside in the optional directory docs/<*pkg_without_cdd-prefix*>/ will be copied to the appropriate doc directory of the meta-package.

conf If there exists a file `common/conf` and has a size greater than 0 this is used as special configuration file `/etc/cdd/<cdd>/<cdd>.conf` which can override variables from the general configuration file `/etc/cdd/cdd.conf` or add further variables. Because it is sourced from shell it has to follow shell syntax.

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

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

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

These files are used by `cdd-gen-control(1)` to build valid `debian/control` files which contain only available packages in their dependencies. This enables building meta packages for `stable`, `testing`, `unstable` or even a completely different distribution which 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.*

Templates in `/usr/share/cdd/templates`

The directory `/usr/share/cdd/templates` contains templates which can be used to build a <*cdd*>`-common` which uses the tools which are contained in the `cdd-common` package and are useful to manage <*cdd*> user groups (see 'User roles' on page 28).

6.4.2 Package `cdd-common`

This package creates a common registry for all CDDs in `/etc/cdd`. Each CDD should put the files which 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.

`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>`.

`cdd-update-menus(8)`

NAME `cdd-update-menus` - add menu of meta package 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's called by a user, it addds and keeps updated menu entries for the user who runs it.

If it's 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 parameted 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>`.

`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>`.

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 which 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 which overrides 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>.

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 currently existing Custom Debian Distributions showed clearly that they depend from 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 which leaves a certain amount of uncertainty. The way the currently existing Custom Debian Distributions started in the past was that somebody had the idea that something has to be done and started doing it. After some time it became clearly visible that without a person who is taking over the leadership the project stays stale. So the initiator has to answer the question clearly, whether he is able to take over this *job* regarding the time he has 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 border line is essential for the development of the project and also for outsiders who might either expect more from the project as it can do or sometimes ignore the project because they do not regard it as helpful because they are not able to find out the purpose.

By maintaining a good relationship to other Free Software projects some common tasks can be done very effectively. Thus efforts can be shared and looking for allies could reduce the amount of work to do.

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 by goals that have parts in common.

Who decided to start a Custom Debian Distribution takes over 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 give people an idea as to 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 `people.debian.org` in case 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.net`). There is a special Wiki page for Custom Debian Distributions (`http://wiki.debian.net/index.cgi?CustomDebian`).

Finally, the best way is to have a page under `www.debian.org/devel`. While not as straightforward as maintaining a personal page or a Wiki site, 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.

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 some preliminary discussion on debian-devel to determine, and there is some information about the newly to create Custom Debian Distribution available on the web which can be linked 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. Enlist 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 send out is always a good idea.

Mails to <debian-devel-announce@lists.debian.org> have to be signed by the GPG key of an official Debian developer but 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 non-Debian lists as well if they cover the same topic as the Custom Debian Distribution. If your announcement is well done it will generate a huge amount of answers from many outsiders and is able to attract people to Debian.

7.2.5 Explaining the project

Now the real work starts. People who are involved into the project should be aware that the 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 projects goals. Thus, Debian Jr. first had to decide which children: What age? English speaking children only, or other languages as well? And 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 part of microbiology related packages out of Debian-Med into a Debian-Bio project. This is reasonable in principle and should be really done in fact the initiator of Debian-Med would support this idea. So he gave the answer: “Just start the Debian-Bio project and take over all related stuff. Until this happened we will serve medical stuff which has to deal with sequence analysis etc. with Debian-Med services.” Unfortunately there was silence around Debian-Bio 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 meta packages

According to the plan of the project the first meta packages (‘Meta packages’ on page 21) should be developed. It is not always easy to decide what should be included and which meta packages should be built. The best way to decide here is discussion on the mailing list some well though proposals.

Section ‘Text user interfaces’ on page 25 mentions `tasksel` as a tool to select a Custom Debian Distributions and explains the problem 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 to the users. This will include potential new Debian users abroad, who may not be on a Debian mailing list. Here applies the same as for the first announcement of the project: Propagating the information to relevant places is an important issue.

7.4.2 Users of a Custom Debian Distribution

By this time people have newly installed Debian along with the stuff of 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 mixed blessing. In the first development phase, there are some developers as users and some intrepid "early adopters", but once it is released, the first version is "out there" and the project is necessarily going to 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. I think 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), 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

To do

8.1 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 meta packages, 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. This project contains a repository for all Custom Debian Distribution related work. The current layout for the repository is as follows:

```
cdd -+-- doc +-- common [this document]
     |       |
     |       +-- med    [Debian-Med documentation]
     |       |
     |       +-- junior [Debian-Jr documentation]
     |       +
     |       ...
     |
     +-- common [common tools for all CDDs]
     |
     +-- junior [junior-* meta packages]
     |
     +-- med    [med-* meta packages]
     |
     ...
```

8.2 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 50. 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.

It might make sense to maintain also a common Custom Debian Distributions web page under `http://www.debian.org/devel/cdd` to provide general information which will be translated by the Debian web team.

8.3 Debian Package Tags

Debian Package Tags (http://deb-usability.alioth.debian.org/debtags/) are a really nice feature which should definitely be used in future Custom Debian Distribution related tools.

8.4 Enhancing basic technologies regarding Custom Debian Distributions

In section 'Future handling of meta packages' on page 28 several issues where raised how handling of meta packages should be enhanced.

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

The method how Debian-Edu currently builds its meta packages from a kind of *database* (in the `tasks` directory of the source) was generalised in the packages `cdd-dev` ('Package `cdd-dev`' on page 31) and `cdd-common` ('Package `cdd-common`' on page 33). 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 meta packages 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 meta packages:

```
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 meta packages should provide patches for dependend packages if they need a certain configuration option and the package in question does feature a `debconf` configuration for this case. Then the meta package 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 meta package 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 meta package 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.

8.5 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. This adaptation can have different focuses:

Distribution The original Knoppix CD is based on a mixture of Debian `testing`, `unstable` and even packages from other sources than the official Debian mirror. There are Knoppix derivatives like Gnoppix (`http://www.gnoppix.org/`) which try to stick to `stable` or at least to one defined set of Debian packages.

User interface Knoppix has a highly customised KDE environment which just works as it is. There are efforts to release live CDs with Gnome interface (Gnoppix), XFCE or other desktops which are able to cope with less system resources.

Kernel There are certain reasons to exchange the kernel of the Knoppix CD like in the Cluster-Knoppix (`http://bofh.be/clusterknoppix/`)-Project which uses an OpenMosix kernel.

Special applications Most of the Knoppix derivatives aim at providing special applications for the purpose of demonstration, training of a classroom using the Knoppix net-boot feature or just having the favourite application always available by just carrying a CD in the wallet. Examples are:

Knoppix4Kids (http://www.osef.org/) Knoppix for Children - connected to Debian-Jr.

Quantian (http://dirk.eddelbuettel.com/quantian.html) Remastered "ClusterKnoppix" for the needs of Mathematicians

LiveOIO (http://sourceforge.net/project/showfiles.php?group_id=9295) Knoppix with PostgreSQL and Zope to run OIO - interesting for Debian-Med.

ISO image of GnuMed Knoppix (http://marvin.ba-loerrach.de/gnumed.iso) Knoppix with PostgreSQL and GnuMed - interesting for Debian-Med.

Vigyaan (http://www.vigyaancd.org/) Knoppix for computational biology and computational chemistry containing ClustalX, BLAST (NCBI-tools), Open Babel, EMBOSS tools, GROMACS, Ghemical, PyMOL and others.

Similar projects In the past there was some confusion about of the goals of Live-CD building projects. Even at the Debian development plattform alioth.debian.org do some similar projects exist.

Debix (http://alioth.debian.org/projects/debix/) Debix is a collection of scripts to create live filesystems ranging from cloning any existing linux system, providing a comfortable environment for the boot-floppies and debian installer up to a full blown live filesystem comparable with Knoppix.

When the author Goswin von Brederlow <brederlo@informatik.uni-tuebingen.de> was asked about his goals he answered: "Debix is a level below knoppix I would say. If you handle the knoppix debs and scripts you could use debix to make seemingly writeable cd images out of a tar.gz or a directory containing the knoppix tree."

Debix is more than one thing:

1. A tool to make a live-cd out of any linux system.
2. Premade sets of package lists and configuration and patches to automatically create nice live-cds like knoppix.

In cvs (on alioth) is a version of Debix that can be called "proof of concept" of part 1. Work is in progress of changing the build scripts to be modular and flexible so part 2 can be started.

Debix can provide the infrastructure. Knoppix has to supply the debs that should be in a Knoppix set for debix. The 2 parts mean that Debix is suposed as tool to create a live-cd from an existing or hand build system but also a tool that can build such system automatically according to preset rules (list of debs and some cleanup scripts if needed).

Metadistros (http://alioth.debian.org/projects/metadistros/) Debian Metadistros goal is to allow you easily remastering Live-CD distributions like Knoppix to fit your or you users needs, within Debian.

It is a little bit hard to get information about this project, because most of the information is in Spanish language.

One piece of the docs which Sergio Talens-Oliag <sto@debian.org> has kindly translated says: "...the main problem is that Debian wants a Debian tool to make its own LiveCDs but Metadistros wants to give tools to let anyone create a distribution that can be used as LiveCD and/or installed and be based in whatever linux distribution the user wants. Anyway, he said that if they can cooperate in any way they will be happy to do it."

Morphix (http://www.morphix.org/) Morphix is a modular GNU/Linux distribution on livecd's (you burn the CD, you put it in your CD-Rom drive, you boot and it works...no harddisk-installation necessary, doesn't touch your data). Also, installing Morphix on a harddisk is a breeze, if you want to. Just click on the icon on the desktop, or choose the installer from the morphix/babytux submenu. Morphix should still be considered experimental in nature. No guarentees are given, use Morphix at your own risk!

ISO's with XFCE4, Gnome2.4, KDE3.1, a game iso and a large number of derivatives are available. Morphix is an Open Source/Free software project, based on Debian GNU/Linux and Knoppix.

So building Live CDs is a common issue for each Custom Debian Distribution and the goal is to develop a mastering system which drastically decreases the effort to build such live CDs. To accomplish this goal the debian-knoppix (http://alioth.debian.org/projects/debian-knoppix) project on Alioth was created.

Currently *re-mastering* is a top-down strategy: People who want to build there own Knoppix-based live CD proceed this way

1 Download a complete ISO image. Even with `bittorrent` or similar techniques it makes no sense to download 700MBytes for each new Knoppix version if you might probably need only half of this size for your intended use. Moreover regarding to the fact that Knoppix consists mostly of installed Debian packages you might have nearly all stuff you need on a local (or at least nearby) Debian mirror with a fast connection.

2 Copy the stuff from the CD to a temporary space.

3 Remove packages which are not needed. This requires some research for packages which are worth removing (regarding the space which is gained) and which are not needed later on.

4 After these steps (all of these are quite time consuming and need a certain amount of knowledge) some further packages can be installed. In case you want to include some database applications or some other applications that need to write a certain amount of data your are more or less on your own to invent techniques to find out how to do that.

Except from some postings in the Knoppix-Mailing list there is no reasonable documentation, how to do this right.

5 Create ISO image from chroot environment and burn it.

It would make much more sense to use a bottom-up strategy and *master* the CD instead of re-mastering. It might even make sense to build a Custom Debian Distribution for itself to build the necessary tools for this *mastering a Knoppix-Live-CD* approach. The general way would be as follows:

1 Use `debootstrap` to build a basic system you could `chroot` into.

6 Install Knoppix stuff into chroot environment. This is the hardware detection stuff, the special configuration, etc. After this step the system should be in a state like after step 3. of the re-mastering process above. The tricky part to accomplish this is to build reasonable Debian packages like this:

`knoppix-hardware` Contains all the hardware detection stuff

`knoppix-x` Contains stuff from Knoppix which cares for X. This is not necessarily needed for simple rescue CDs.

`knoppix-config` Special configuration stuff. Please note these packages will be installed into a chroot environment which is *not* a Debian host system. It might be necessary to change the configuration of some packages installed in this chroot environment which conflicts to Debian policy in a *real* Debian system. But here we face a special part of our hard-disk (say `/var/cache/knoppix/etc`) which is not covered by policy. The only point is to make sure that this `knoppix-config` package will not be installed on a Debian host system (if and only if anything is really needed which would not comply with the policy).

`knoppix-misc` Whatever might be needed and is not covered by the things above. Here user support for integrating database applications might be integrated.

7 Customise chroot environment for intended purpose. This is the same as in the re-mastering step 4. but it could be supported by some tools from `knoppix-misc`.

8 Create ISO image from chroot environment and burn it. While this is the same as step 5. but it might also be supported by some nifty tools which would simplify things for anybody wanting to build their own CD.

This approach would have the additional advantage of being portable also to non-i386 architectures and in fact Fabian Franz `<FabianFranz@gmx.de>` managed to prove this true for Power-PC architecture.

8.6 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 meta package 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

Using the Bug Tracking System

A.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.

A.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`.