Debian Pure Blends

Andreas Tille

Contents

1 Introduction **1**

2 What are Debian Pure Blends? **2**

2.1 What is Debian? 2

2.2 What is Debian? (next try) 3

2.3 Differences from other distributions 3

2.4 Debian Pure Blends 4

2.5 Difference between a Blend and a remastered system 4

2.5.1 Technical 4

2.5.2 Philosophical 4

2.5.2.1 Technical advantage 4

2.5.2.2 Community advantage 5

2.6 Further resources about Blends 5

3 General ideas **6**

3.1 Looking beyond 6

3.2 Motivation 6

3.2.1 Profile of target users 6

3.2.2 Profile of target administrators 7

3.3 Status of specialised Free Software 7

3.4 General problem 8

3.5 Debian Pure Blends from philosophical point of view 8

4 Existing Debian Pure Blends **10**

4.1 Debian Junior: Debian for children from 1 to 99 10

4.2 Debian Med: Debian in Health Care 11

4.3 Debian Edu: Debian for Education 11

4.4 Debian GIS: Geographical Information Systems 12

4.5 Debian Astro: professional and hobby astronomers 12

4.6 DebiChem: Debian for Chemistry 12

4.7 Debian Science: Debian for science 13

4.8 Debian Accessibility Project 13

5 Distributions inside Debian 14
5.1 To fork or not to fork . . . 14
5.1.1 Commercial forks . . . 14
5.1.2 Non-commercial forks . . . 14
5.1.3 Disadvantages of separate distribution . . . 15
5.1.4 Advantages of integration into Debian . . . 15
5.1.5 Enhancing Debian . . . 15
5.2 Adaptation to any purpose . . . 16

6 Technology 17
6.1 Metapackages . . . 17
6.1.1 Metapackage definition . . . 17
6.1.2 Collection of specific software . . . 18
6.1.3 Packages showing up in more than one metapackage . . . 18
6.1.4 Adapted configuration inside metapackages . . . 18
6.1.5 Documentation packages . . . 18
6.2 Handling of metapackages . . . 18
6.2.1 Command line tools . . . 19
6.2.2 Text user interfaces . . . 20
6.2.3 Graphical user interfaces . . . 21
6.2.4 Web interfaces . . . 21
6.2.5 Future handling of metapackages . . . 22
6.3 User roles . . . 23
6.3.1 User menu tools . . . 23
6.3.1.1 Using the Debian menu system . . . 23
6.3.1.2 Managing Debian Pure Blend users with debconf . . . 24
6.4 Development tools . . . 24
6.5 Dealing with name space pollution . . . 24

7 How to start a Debian Pure Blend 26
7.1 Planning to form a Debian Pure Blend . . . 26
7.1.1 Leadership . . . 26
7.1.2 Defining the scope of the Blend . . . 26
7.1.3 Initial discussion . . . 26
7.1.3.1 Calling all developers . . . 27
7.1.3.2 Steering the discussion . . . 27
7.2 Setting up . . . 27
7.2.1 Mailing list . . . 27
7.2.2 Web space . . . 27

7.2.3 Repository . . . 28
7.2.4 Formal announcement . . . 28
7.2.5 Explaining the project . . . 28
7.3 Project structure . . . 28
7.3.1 Sub-setting Debian . . . 28
7.3.2 Using tasksel and metapackages . . . 29
7.3.3 Adding new "normal" packages . . . 29
7.4 First release . . . 29
7.4.1 Release announcement . . . 29
7.4.2 Users of a Debian Pure Blend . . . 29
7.4.2.1 Devoting resources to the users . . . 29
7.4.2.2 Developer vs. user mailing list . . . 29
7.4.2.3 User support beyond Debian . . . 30

8 The web sentinel **31**
8.1 Existing and prospective packages . . . 31
8.2 Tasks files controlling web sentinel content . . . 31
8.2.1 Configuring Web Sentinel pages per Blend . . . 32
8.2.2 Debian Description Translation Project . . . 33
8.2.3 Features of the web sentinel tasks pages . . . 33
8.3 Bugs overview . . . 34
8.4 Versions in stable / testing / unstable / new / VCS upstream . . . 34
8.5 Quality assurance report . . . 34

9 To do **35**
9.1 Establishing and using communication platforms . . . 35
9.2 Enhancing visibility . . . 35
9.2.1 Debian Pure Blends web pages . . . 36
9.3 Debian Package Tags . . . 36
9.4 Enhancing basic technologies regarding Debian Pure Blends . . . 37
9.5 Building Live CDs of each Debian Pure Blend . . . 37
9.6 New way to distribute Debian . . . 37

A Description of development tools **40**
A.1 Package `blends-dev` . . . 40
A.1.1 Blend`-tasks.desc` . . . 40
A.1.2 `debian/control` . . . 41
A.1.3 `statusdump` . . . 42
A.1.4 changelogentry . . . 42
A.1.5 Apt `sources.list` files in `/etc/blends/` . . . 42

A.1.6 Templates in `/usr/share/blends/templates` . . . 43
A.2 Package `blends-common` . . . 43
A.2.1 blend-role(8) . . . 43
A.2.2 blend-update-menus(8) . . . 43
A.2.3 blend-user(8) . . . 44
A.2.4 blends.conf(5) . . . 44
A.3 How to develop new normal packages in Pure Blends . . . 44
A.4 Working with the source repository in `Git` . . . 45
A.5 How to create tasks and bugs pages of web sentinel . . . 46
A.6 Editing static web pages of Blends on blends.debian.org . . . 46
A.7 Description how Blends relevant data are gathered and stored . . . 46
A.7.1 Packages in Debian ftp new queue . . . 46
A.7.2 Machine readable data in Git repositories of Blends and some packaging teams . . . 46

B Quick intro into building metapackages 47
B.1 Defining dependencies for metapackages . . . 47
B.2 The packaging directory . . . 47
B.3 The common metapackage . . . 48
B.4 The metapackage menus . . . 48
B.5 Menu for any dependency . . . 49

C Using the Bug Tracking System 50
C.1 How to ask for packages which are not yet included . . . 50
C.2 How to report problems . . . 50

D FAQ 51
D.1 How can I add a dependency? . . . 51
D.2 What additional information should be provided? . . . 51
D.3 Should I add binary or source packages? . . . 51
D.4 Should I add a library package to a user task? . . . 51
D.5 Can I create a new task if the existing ones do not fit? . . . 52
D.6 Why not simply use a Wiki? . . . 52
D.7 Why not simply using DebTags? . . . 52

Abstract

This paper is intended for people who are interested in the philosophy of Debian Pure Blends (in short "Blends" if it is used clearly in internal Debian context), and the technique that is used to manage those projects. For those who are familiar with the concept of Custom Debian Distributions: We just found a new name for this concept because the old name just not expressed what actually is done. It is explained in detail why Blends are not forks from Debian, but reside completely inside the Debian GNU/Linux distribution, and which advantages can be enjoyed by taking this approach. The concept of metapackages and user role based menus is explained. In short: This document describes why Debian Pure Blends are important to the vitality and quality of Debian.

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.

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

To clarify the relation between a Blend and a derivative which is frequently mixed up Ben Armstrong said in a discussion on the Blends mailing list: "While a Blend strives to mainstream with Debian, a derivative strives to differentiate from Debian."

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

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

A word of warning: The fact that a Blend covering a certain field of work does exist does not mean that it might be a complete drop in replacement of Free Software solutions for all tasks in this specific field. Some Blends just started to work on this and adopted the technical framework to formalise the work on the project but it might perfectly happen that there is just a lack of available Free Software solutions for certain tasks. Debian can do less about this because we just assemble a set of software which was developed outside the Debian GNU/Linux distribution. So it has to be checked whether a specific Blend is fit for the intended purpose, whether it might cover just some parts of a fields of work or whether it is just a concept to develop some solutions for the future.

Chapter 2

What are Debian Pure Blends?

2.1 What is Debian?

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

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

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

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

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

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

RPM (RedHat Package Manager) which is supported by RedHat, SuSE 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 (LSB) is attempting to develop and promote a set of standards that will increase compatibility among Linux distributions, and enable software applications to run on any compliant system.

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

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

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

It is this *adherence to policy* that causes a distribution to remain consistent within its own bounds. At the same time, this is the reason why packages can not always be safely installed across distribution boundaries. A SuSE `package.rpm` might not play well with a RedHat `package.rpm`, although the packages work perfectly well within their own distributions. A similar compatibility 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, Debian Pure Blends 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 RedHat, SuSE, Ubuntu and others.

Debian is just one of them.

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

2.2 What is Debian? (next try)

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

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

All members of the Debian project are connected in a web of trust, which is woven by signing GPG keys. One requirement to become a member of the Debian project is to have a GPG key signed by a Debian developer. Every time one Debian developer meets another developer for the first time, they sign each other's keys. In this way, the web of trust is woven.

2.3 Differences from other distributions

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

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

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

2.4 Debian Pure Blends

Debian contains nearly 22.000 binary packages, and this number is constantly increasing. There is no single user who needs all these packages (even if conflicting packages are not considered).

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

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

Debian Pure Blends - in short Blends if used clearly in the Debian internal context which makes "Pure" and "Debian" obvious - which were formerly known as Custom Debian Distributions (this name was confusing because it left to much room for speculation that this might be something else than Debian) try to provide a solution for *special groups of target users with different skills and interests*. Not only do they provide handy collections of specific program packages, but they also ease installation and configuration for the intended purpose.

Debian Pure Blends are *not forks* from Debian. As the new name says clearly they are pure Debian and just provide a specific flavour. So if you obtain the complete Debian GNU/Linux distribution, you have all available Debian Pure Blends included.

The concept of what is called *Blend* in Debian is also known in other distributions. For instance in Fedora there are Special Interest Groups (SIGs) even if some SIGs in Fedora are what in Debian is known as internal project because it is focused on technical implementations and not on user-oriented applications.

2.5 Difference between a Blend and a remastered system

Not necessarily all currently existing Blends are actually providing installation media (live media or installer). The reason for this is that such installation media are not always necessary / wanted. You can just install plain Debian and install some metapackages on top of it. However, the metapackage approach makes the creation of installation media quite simple by using Debian Live. Here are some reasons for this approach compared to a remastering strategy.

2.5.1 Technical

The process for creation of a blend involves starting with a Debian or derivative repository and creating an image directly from that (live, install or otherwise) that contains a selection of material from that repository delivered in such a way that it is usable by a particular target user for a particular purpose with a minimum of effort.

By contrast, the process of remastering generally involves first downloading an image produced by the parent distro (live, install or otherwise,) then tearing it apart and reassembling it with your customizations applied.

2.5.2 Philosophical

The blends philosophy is to work as closely with the parent distro as possible. If possible, the project should be done entirely within the distro as a subproject, containing only material supplied by the parent distro. We call this a "Pure Blend".

The remastering philosophy (if it can be called that) seems to be "whatever works" and involves little or no interaction with the parent distro. It's a lazy approach used by people who have newly discovered that they can hack images to make them into custom images to make something uniquely theirs. Probably fine for quick-and-dirty results, but hard to support in the long run.

The users of a blend are served better than the users of a remaster because of the following advantages:

2.5.2.1 Technical advantage

A new version of a well-crafted blend ought to be able to be produced at any time directly from the repository simply by building it; the user has some assurance that the resulting system remains 'untainted' by hacking it up with scripts that 'damage' the original system by removing files from packages, changing files in packages, etc. something that hurts maintainability / support for such a system.

2.5.2.2 Community advantage

A blend project aims to leverage support resources from the existing community to serve some sub-community within it. They accomplish this by not violating Debian packaging policy, producing something that is either pure Debian (a "pure blend") or Debian + additional packages, rather than some frankendistro artlessly stitched together from someone else's distro with scripts that change things everywhere with no regard to policy. Thus, normal support channels can be used with a pure blend since what you end up with is not a derivative at all, but just Debian, set up and ready to go for whatever you wanted to use it for.

2.6 Further resources about Blends

Wiki `https://wiki.debian.org/DebianPureBlends`

Mailing list `https://lists.debian.org/debian-blends/`

IRC #debian-blends `irc://irc.debian.org/debian-blends`

Chapter 3

General ideas

3.1 Looking beyond

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

Enterprise solutions

- 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 SuSE Firewall on CD, ...

Content Management System RedHat

Portal Server RedHat

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

Debian solves this problem with *Debian Pure Blends*.

3.2 Motivation

3.2.1 Profile of target users

The target user of a Blend may be a specialist of a certain profession, (e.g. a doctor or lawyer,) a person who has not (yet) gathered a certain amount of computer knowledge, (e.g. a child,) or a person with disabilities (e.g. a visually or hearing impaired person.) Moreover, the customisation might deal with peculiarities of certain regions where users have needs that differ from Debian as a whole.

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

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

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

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

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

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

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

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

3.2.2 Profile of target administrators

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

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

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

3.3 Status of specialised Free Software

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

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

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

- The trouble often continues in the maintenance of the installed software.
- When it comes to the usage of the specialist software, it often happens that it perfectly fits the needs of the developer who wrote it for his own purposes, and who is familiar with its quirks, but in many cases such software does not comply with ergonomic standards of user interfaces.
- Several existing programs that might be useful for specialists are not really free in the sense of the Debian Free Software Guidelines (DFSG). Programs that are incompatible with the DFSG cannot be included in Debian. This is possibly a drawback for those programs, because they could profit by spreading widely on the back of Debian over the whole world.
- A certain number of programs are developed at universities by students or graduates. Once these people leave the university, the programs they developed might be orphaned; *i.e.*, not a[illegible]ly maintained anymore. If their licenses are too restrictive, it may be impossible for anyone else to take over; sticking to DFSG-free licenses avoid 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.
- Many programs that are written from scratch use their own non-standard file formats. However, it is often important for programs to be able to share data with each other.
- Often there are several programs that try to solve identical or similar problems. For instance the Debian Med team faces this in the case of programs claiming to serve as a medical practice management solution. Normally, all these programs take very interesting approaches but all of them have certain drawbacks. So, joining programmers' forces might make sense here.
- Sometimes the tools or back-ends used in Free Software are not appropriate for such applications. For instance, sometimes database servers that do not use transactions are used to store medical records, which is completely unacceptable. Other programs use web clients as their front-end, which is not really good for quick (mouse-less) usage, a great shortcoming for repetitive tasks.

3.4 General problem

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

- programmers *and*
- users

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

Debian wants to attract both groups to get it working.

Debian is the missing link between upstream developers and users.

3.5 Debian Pure Blends from philosophical point of view

Debian currently grows in several directions:

- Number of involved people
- Number of packages

- Number of architectures
- Number of bugs
- Number of users
- Number of derivatives
- Time span between releases

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

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

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

Chapter 4

Existing Debian Pure Blends

4.1 Debian Junior: Debian for children from 1 to 99

Start beginning of 2000

URL `http://www.debian.org/devel/debian-jr`

Tasks Tasks of Debian Jr.

Mailing list debian-jr@lists.debian.org

Initiator Ben Armstrong synrg@debian.org

Activity Activists on Debian Jr. mailing list

Release Debian 3.0 (Woody)

Goals

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

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

Debian Jr. was the first Blend. In fact, at the time this project was created, the idea behind of Debian Pure Blends was born, although then, we used the term "Debian Internal Project". Over time, this name was changed to "Custom Debian Distributions" first because it was too broad, as it was equally descriptive of a number of quite different projects, such as IPv6 and QA. The next change of names became necessary when it was realised that the term "Custom Debian Distribution" was considered as "something else than Debian" by any newcomer. This was so misleading that it effectively blocked a wide propagation of the principle.

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

4.2 Debian Med: Debian in Health Care

Start beginning of 2002

URL Debian Med

Tasks Tasks of Debian Med

Mailing list debian-med@lists.debian.org

Initiator Andreas Tille tille@debian.org

Activity Activists on Debian Med mailing list
Activists on Debian Med developer list
Committers to Debian Med VCS
Uploaders of Debian Med team
Team members closing the most bugs in Debian Med packages

Release Sarge

Goals

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

4.3 Debian Edu: Debian for Education

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

URL Debian Edu Wiki

Tasks Tasks of Debian Edu

Mailing list debian-edu@lists.debian.org

Activity Activists on Debian Edu mailing list

Responsible Petter Reinholdtsen pere@hungry.com

Release Sarge

Goals

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

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

4.4 Debian GIS: Geographical Information Systems

Start October 2004

URL DebianGIS Wiki

Tasks Tasks of Debian GIS

Mailing list user and developer list

Activity Activists on Debian GIS mailing list

Initiator Francesco P. Lovergine frankie@debian.org

Goals

- Geographical Information Systems
- OpenStreetMap and GPS devices

4.5 Debian Astro: professional and hobby astronomers

Start March 2014

URL Debian Astro Blends page

Tasks Tasks of Debian Astro

Mailing list user and developer list

Activity Activists on Debian GIS mailing list

Initiator Ole Streicher olebole@debian.org

Goals Debian Astro is a "Debian Pure Blend" with the aim to develop a Debian based operating system that fits the requirements of both professional and hobby astronomers. It integrates a large number of software packages covering telescope control, data reduction, presentation and other fields.

4.6 DebiChem: Debian for Chemistry

Start October 2004

URL Debichem Alioth page

Tasks Tasks of DebiChem

Mailing list debichem-users@lists.alioth.debian.org

Activity Activists on DebiChem mailing list
Committers to DebiChem VCS
Uploaders of DebiChem team
Team members closing the most bugs in DebiChem packages

Initiator Michael Banck mbanck@debian.org

Goals

- Chemical applications in Debian

4.7 Debian Science: Debian for science

Start July 2005

URL Debian Science Wiki

Tasks Tasks of Debian Science

Mailing list debian-science@lists.debian.org

Activity Activists on Debian Science mailing list
Activists on Debian Science maintainers list
Committers to Debian Science VCS
Uploaders of Debian Science team
Team members closing the most bugs in Debian Science packages

Responsible Sylvestre Ledru sylvestre@debian.org

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

4.8 Debian Accessibility Project

Debian for blind and visually impaired people

Start February 2003

Mailing list debian-accessibility@lists.debian.org

URL Debian Accessibility

Tasks Tasks of Debian Accessibility

Activity Activists on Debian Accessibility mailing list

Initiator Mario Lang mlang@debian.org

Goals

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

Chapter 5

Distributions inside Debian

5.1 To fork or not to fork

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

5.1.1 Commercial forks

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

5.1.2 Non-commercial forks

As a completely free distribution Debian GNU/Linux is quite often a welcome starting point for derived distributions with a certain purpose that are as free as Debian but had certain reasons to fork. One main reason for a fork was that Debian was not flexible enough for certain purposes and some needed features had to be added. One reason for the Debian Pure Blends effort is to increase flexibility and to make the reason mentioned above void (if it is not yet void because of the general development of Debian). Some examples of forks from Debian that are probably now able to integrate back into Debian as a Debian Pure Blend are:

SkoleLinux Mentioning SkoleLinux in the category of forks is more or less history. The merge back into Debian started with the SkoleLinux people really doing a great job to enhance Debian for their own purposes in the form of their work on debian-installer, and culminated with the formal merging of the Blend Debian Edu and SkoleLinux, so that they are now virtually equivalent. This is the recommended way for derived distributions, and the reasons for this recommendation are given below.

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

- **Technical** They had some special requirements for the kernel and configuration. This is more or less solved in the upcoming Debian release.
- **License** When DeMuDi started, not enough free programs in this field existed. This situation is better now.
- **Organisational** Because of the founded status of the project, an extra distribution had to be developed. To accomplish this requirement, Debian Pure Blends plan to build common tools to facilitate building separate CDs with the contents of only a single distribution.

This shows that there is no longer a real need for a fork, and in fact, the organiser of the DeMuDi project was in contact to start bringing DeMuDi back into Debian. That is why DeMuDi is mentioned in the list of Debian Pure Blends above. Unfortunately the effort to merge back has stalled but it might be an interesting project to apply Blends techniques to support multimedia experts who want to use Debian.

LinEx LinEx is the very successful distribution for schools in the Region Extremadura in Spain. The work of the LinEx people perhaps made Debian more popular than any other distribution. The project was founded by the local government of Extremadura, and each school in this region is running this distribution. While this is a great success, the further development of LinEx has to face the problems that will be explained below. Because the creators of LinEx are aware of this fact they started joining the educational part of LinEx with Debian Edu which in turn led to an even stronger position of this Blend.

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

5.1.3 Disadvantages of separate distribution

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

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

New projects with special intentions often have trouble to become popular to the user group they want to address. This is a matter of attaining the critical mass that was explained in Section 3.4.

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

Forking would be a bad idea.

5.1.4 Advantages of integration into Debian

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

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

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

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

5.1.5 Enhancing Debian

By making changes to some packages to make them fit the needs of a target user group, the overall quality of Debian can be enhanced. In this way, enhancing Debian by making it more user friendly is a good way for the community to give back something to Debian. It would be a shame if somebody would refuse all the advantages to keeping a project inside Debian, and instead would decide to try to cope with the disadvantages because he just does not know how to do it the right way, and that it is normally easy to propagate changes into Debian. For instance, see Section C.1. This section explains how you can ask for a certain piece of software to be included in Debian. The next section describes the reason why Debian is flexible enough to be adapted to any purpose.

5.2 Adaptation to any purpose

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

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

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

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

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

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

Chapter 6

Technology

6.1 Metapackages

6.1.1 Metapackage definition

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

- Dependencies on other Debian packages (essential)

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

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

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

- Menu entries (recommended)
  - Place these in `/etc/blends/<blend> /menu/<pkg-name>`
  - Maintain these via role based tools
- Configuration stuff (optional)
  - debconf questions or pre-seeding
  - cfengine scripts (or similar see Section 9.4)
- Special metapackages:
  - `<blend>-tasks`: Contains information for tasksel
  - `<blend>-config`: Special configurations, basic stuff for user menus

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

Examples:

`junior-puzzle` Debian Jr. Puzzles

`education-tasks` Tasksel files for SkoleLinux systems

`med-bio` Debian Med micro-biology packages

6.1.2 Collection of specific software

When using metapackages, no research for available software inside Debian is necessary. It would not be acceptable for normal users to have to browse the descriptions of the whole list of the 20000 packages in Debian to find everything they need. So, metapackages are an easy method to help users to find the packages that are interesting for their work quickly.

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

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

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

6.1.3 Packages showing up in more than one metapackage

This seems to be an FAQ: If a package `A` is in the list of dependencies of metapackage `m` is it allowed or reasonable to add it to the list of dependencies of metapackage `n`?

The answer is: Why not?

The "overlap" is no problem because we do not want to build an exclusive categorisation which might be hard to understand for our users. Metapackages are like "normal" packages: Nobody would assume that because package `x` depends from package `libc` no other package is allowed to add `libc` to its depends. So why not adding a dependency to more than one metapackage if it is just useful for a certain task?

The important thing is to support our users. A specific user wants to solve a certain task (and thus installs a certain metapackage). The question whether some Dependencies are also mentioned in a different metapackage is completely useless for this task. So in fact we do *not* build a categorisation tree but build pools of useful software for certain tasks which can definitely have overlaps.

To give a certain example which was asked by a member of Debian Multimedia team: A user who is seeking for his optimal sound player is not served best if we "hide" an application from his view by including it into sound recorders exclusively. While chances might be good that a sound recorder is not as lightweight as a pure player the user will find out this quickly if he is looking for only a lightweight player - but perhaps he becomes happy later about the "added value" of his favourite player if it also is able to record sound.

6.1.4 Adapted configuration inside metapackages

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

6.1.5 Documentation packages

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

Thus, translation is a very important thing to make programs more useful for the target user group. Debian has established a Debian Description Translation Project, which has the goal to translate package descriptions. There is a good chance this system could also be used for other types of documentation, which might be a great help for Debian Pure Blends.

6.2 Handling of metapackages

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

6.2.1 Command line tools

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

```
~> apt-cache search med | grep '^med-'
med-bio - Debian Med micro-biology packages
med-bio-dev - Debian Med micro-biology development packages
med-doc - Debian Med documentation packages
med-imaging - Debian Med imaging packages
med-imaging-dev - Debian Med packages for medical image develop...
med-tools - Debian Med several tools
med-common - Debian Med Project common package
med-cms - Debian Med content management systems
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This software is licensed under the GPL.

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

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

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

6.2.2 Text user interfaces

tasksel The Debian task installer `Tasksel` is the first interface for package selection that is presented to the user when installing a new computer. The `End-user` section should contain an entry for each Debian Pure Blend. Unfortunately, there are some issues that prevent Blends from being included in the `tasksel` list, because the dependencies of this task can affect what appears on the first installation CD. This problem would be even greater if all Blends were added, and so a different solution has to be found here. (See #186085.) In principle, `tasksel` is a good tool for easy installation of Blends.

As a workaround for this problem the `blends-dev` framework creates a package `BLEND-tasks` which contains a `tasksel` control file. If you install this package all tasks of the Blend will be added to the default list of tasks inside `tasksel`. So a solution for Blend specific installation media might be to just remove the default tasksel list and provide the Blends own tasks exclusively.

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

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

6.2.3 Graphical user interfaces

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

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

synaptic Even more sophisticated and perhaps the best choice for users of Debian Pure Blends. `Synaptic` has a nice filter feature, which makes it a great tool here. Moreover `synaptic` is currently the only user interface that supports Debian Package Tags (see Section 9.3).

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 Debian Pure Blends.*

6.2.4 Web interfaces

Tasks pages The tasks pages probably provide the best overview about the actual work which is done in a Debian Pure Blend. These pages are automatically generated by reading the tasks files (see Section A.1.2) and verifying the existence of the packages that are mentioned as dependencies. On the resulting web page the packages are listed with some meta information and the description of the package. As user oriented pages they are translated into more than 10 languages while translated means, the navigation text of the page generating code is using `gettext` which enables translation (the work is not yet completely done for all languages) but even more importantly the descriptions of the packages are translated as well by using the information from Debian Description Translation Project.

These tasks pages are available via

```
            http://blends.debian.org/BLEND/tasks
```

where `BLEND` has to be replaced by the name of the Blend. Currently these pages are available for the Blends:

```
      accessibility, edu, gis, junior, lex, med, science, debichem
```

In short: If you want to know more about a specific Blend go to its task page and have a look what is listed there.

Bugs pages The more developer oriented bugs pages try to match the scope of the tasks pages mentioned above but there is no description of the packages given but rather the bugs that are reported in the Debian Bug Tracking System (BTS) are listed there. This is a quite valuable source of information if somebody is interested in increasing the quality of a Blend: Fixing bugs is always welcome and listing all relevant bugs at a single place is a nice way to detect problems quickly.

These bugs pages are available via

```
            http://blends.debian.org/BLEND/bugs
```

where `BLEND` has to be replaced by the name of the Blend. Currently these pages are available for the Blends:

```
      accessibility, edu, gis, junior, lex, med, science, debichem
```

In short: If you want to help enhancing the quality of a specific Blend go to its bug page and start working on the bugs listed there.

search Debian has a web interface that can be used to search for certain substrings in package names. For instance if you are searching the meta packages of Debian Med you could point your favourite Browser to

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

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

Package Tracking System The Package Tracking System is a really great tool that provides essential information about packages. Most Debian Pure Blends are using a mailing list address as Maintainer of their key packages which includes the metapackages. This so called team maintenance of packages is on one hand very handy from a developers point of view on the other hand it enables using the Package Tracking System to get a quick overview:

Debian Jr: `http://qa.debian.org/developer.php?login=debian-jr@lists.debian.org`

Debian Med: `http://qa.debian.org/developer.php?login=debian-med-packaging@lists.alioth.debian.org`

Debian Edu: `http://qa.debian.org/developer.php?login=debian-edu@lists.debian.org`

Debian Science: `http://qa.debian.org/developer.php?login=debian-science-maintainers@lists.alioth.debian.org`

Hint: If you append the option `&ordering=3` you might get some sectioning of this page according to the metapackage categories. This result is approached by a tool which subscribes all dependent packages to the group maintenance address and adds a section according to a metapackage name.

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

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

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

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

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

Short conclusion: *The Debian Pure Blends provide some nice web tools for a whole set of packages for a certain working field that provide a better overview than the usual Debian tools that are basically dealing with single packages..*

6.2.5 Future handling of metapackages

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

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

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

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

6.3 User roles

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

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

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

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

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

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

6.3.1 User menu tools

6.3.1.1 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 blend-update-menus(8) (see Section A.2.2) script from `blends-common`. It has to be said that using blend-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 Blend is necessary if this is not done by the `postinst` script of a metapackage. This can easily been done if the configuration file of a Debian Pure Blend `/etc/blends/<blend>/<blend>.conf` contains the line

```
UPDATEUSERMENU=yes
```

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

6.3.1.2 Managing Debian Pure Blend users with debconf

Using `blends-dev` it is very easy to build a `blend-config` package that contains debconf scripts to configure system users who should belong to the group of users of the Debian Pure Blend `blend`. 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 users who
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.)

6.4 Development tools

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

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

This package is described in detail in appendix Section A.1.

`blends-common` This package provides some files that are common to meta packages of Debian Pure Blends especially those that were built using the tools of the package `blends-dev`. It introduces a method to handle system users in a group named according to the name of the Blend. The user menu approach is explained in detail in Section 6.3.

This package is described in detail in appendix Section A.2.

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

6.5 Dealing with name space pollution

Due to the fact that Blends might deal with quite specialised software the user base is often quite small. In such use cases it happens that some programs are using names that are used by other more frequently used tools. For instance the Debian Med program `plink` has the same name as the program `plink` that belongs to the ssh clone `putty`. According to the Debian policy both packages, `putty` and `plink` need to be co-installable. Thus one tool has to be renamed.

Name space conflicts in Debian are usually solved by the principle that whoever comes first has taken the name if there is no better agreement. However, it makes sense to keep the original name chosen by upstream for the more frequently used program - and here it might be that the Blends packager steps back. There might be a chance to discuss with upstream about a better name but there is no guarantee that this will be successfully and thus there is another option for the Blends developer to provide the original upstream name to the Blends users by circumventing the file name conflict - at least for users of the `bash` shell.

Since `blends-dev >= 0.6.92.3` in the `blend-config` a `bash` init script is provided in

```
/etc/profile.d/<BLEND-name>.sh
```

This script is executed in login shells and checks for two things:

1. Does the path `/usr/lib/<BLEND-name>/bin` exist (since a package installs files there)?
2. Has the user starting the `bash` instance created a file `$HOME/.blends` and is there a line featuring the name of the Blend (in the example above "debian-med")

If both conditions are fulfilled the PATH gets prepended by `/usr/lib/<BLEND-name>/bin` and thus the tools residing in this directory were found first.

Moreover also MANPATH is prepended by `/usr/lib/<BLEND-name>/share/man` to enable providing proper manpages to the binaries in the Blends private PATH. Otherwise it might happen that a user might see a manpage of the executable in `/usr/bin` which is not the first in the search PATH any more. The location is a topic of further discussion since manpages under `/usr/lib` are in conflict with FHS.

Chapter 7

How to start a Debian Pure Blend

This chapter is based on the Debian Subproject HOWTO, which was written by Ben Armstrong synrg@debian.org.

7.1 Planning to form a Debian Pure Blend

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

7.1.1 Leadership

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

7.1.2 Defining the scope of the Blend

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

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

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

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

7.1.3 Initial discussion

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

7.1.3.1 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.

7.1.3.2 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.

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. An example bug report to create the relevant list is bug #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 simple possibility, and one which is fairly attractive because it facilitates collaborative web site creation and maintenance, is to put a page on the Wiki. There is a special Wiki page for Debian Pure Blends.

A good place to put static web pages is the common place for all Blends: `http://blends.debian.org`. There is a subdirectory for each Blend and it is very easy to create a simple index page there which points to the automatically generated web pages which are mentioned in Section 6.2.4. Following this strategy is quite cheap and has a big effect when using the tools provided by the Debian Pure Blends effort.

Sooner or later a Debian Pure Blend will establish an own project at salsa.debian.org to host own Git repositories.

Finally, the best way is to have a page under `http://www.debian.org/devel`. While not as straightforward as any of the other options, this approach has its advantages. First, the site is mirrored everywhere. Second, the Debian web site translators translate pages into many different languages, reaching new potential audiences for your Debian Pure Blend, and improving communication with other members of your project and interested parties for whom English is not their most comfortable language. Third, a number of templates are available to make your site more integrated with the main web site, and to assist with incorporating some dynamic content into your site. Before you join the Debian Web team you should learn more about building Debian web pages.

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.

7.2.3 Repository

On salsa.debian.org a GitLab instance is running to host all Debian related project work. Creating a project on Salsa is a good idea to start teamwork on the code a Debian Pure Blend is releasing.

7.2.4 Formal announcement

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

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

7.2.5 Explaining the project

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

7.3 Project structure

7.3.1 Sub-setting Debian

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

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

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

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

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

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

7.3.2 Using tasksel and metapackages

According to the plan of the project, the first metapackages (Section 6.1) should be developed. It is not always easy to decide what should be included, and which metapackages should be built. The best way to decide on this point is to discuss on the mailing list some well thought out proposals.

Section Section 6.2.2 mentions `tasksel` as a tool to select a Debian Pure Blend, and explains why it is currently not possible to get a Blend included into the task selection list.

7.3.3 Adding new "normal" packages

Besides metapackages, you may want to develop some new packages for your blend in order to customize some system behavior, like changing XDG menu style. Those normal packages are not defined in your tasks, hence we need to define it properly so that blends-dev will know how to package these new, normal packages. Adding new normal packages is almost identical to debian packaging, just that you need to define your package parameters in debian/control.stub instead of debian/control. Section Section A.3 describes what you need to know for adding new normal packages in Debian Pure Blends in more detail.

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 Debian Pure Blend. This will not only be directed at the Debian developer community, but also at the users. This will include potential new Debian users abroad, who may not be on a Debian mailing list. Here, the same principle applies as for the first announcement of the project: it is important to consider sending the information to other relevant forums.

7.4.2 Users of a Debian Pure Blend

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

7.4.2.1 Devoting resources to the users

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

7.4.2.2 Developer vs. user mailing list

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

7.4.2.3 User support beyond Debian

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

Chapter 8

The web sentinel

8.1 Existing and prospective packages

The so called tasks pages probably give the most interesting overview about what a Blend is actually doing for newcomers as well as a nice quality assurance tool for developers. If you want to see examples for tasks pages just have a look at the list of all Blends using this technique and follow the links to the tasks pages once you selected one specific Blend.

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

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

The rationale behind this is to provide as much as possible information about packages that might be interesting for the target user of the Blend. Moreover the page can provide useful information for developers about things that might be a useful help for the project to work down the todo list and build Debian packages for software that is not yet included in Debian.

8.2 Tasks files controlling web sentinel content

The content of the tasks files that are used to build the metapackage content of a Blend is also used to determine the content of the web sentinel pages. Thus you can influence the tasks pages by simply editing the tasks files inside the VCS of the sources of the Blends metapackages. The canonical location of these tasks files inside the sources is (with the exception of Debian Edu that is locatet in Debian Edu Git repositories) the Blends Git at

```
    https://salsa.debian.org/blends-team/BLEND.git
```

Here you can add or remove additional Dependencies to existing packages or you can add additional information about so called prospective packages. The syntax how to do this is explained below.

To get the todo list built it is necessary to add some additional information to the task files which are the main database of information for the Blend. The information is following the RFC822 syntax as all Debian control files do and is kept quite simple:

Depends / Recommends / Suggests Even if there is no Debian package available for the moment the names of prospective packages are given as if they would exists. The rationale behind is that once such a package might exist the source of the metapackage does not have to be changed and will work out of the box after rebuilding.

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

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

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

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

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

Vcs-Git If there is some Debian packaging stuff available this can be addressed in this field. Unofficial packages which have this field set are rendered in a separate section with links to the packaging Git.

The usage for this field is the same as it is described in paragraph 6.2.5 of Debian developers reference

Vcs-Browser If there is some Debian packaging stuff available this can be addressed in this field. Unofficial packages which have this field set are rendered in a separate section above unofficial packages outside the official Debian mirrors. If you have set `Vcs-Git` there is no need to set `Vcs-Browser` explicitly because it is obtained automatically from the other fields. But you might override this automatically generated URL if needed.

The usage for this field is the same as it is described in paragraph 6.2.5 of Debian developers reference

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

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

Remark In some cases it makes perfectly sense to add a remark on behalf of the Blends team to a package which is visible on the tasks page. This can be done using the Remark field. It can be used in the same syntax as a Description field in Debian control files.

Registration Sometimes, specifically in the case of scientific software, the project authors ask for registration of their software to get numbers of users of their software. These numbers enable them to ask for further funding of their project. To support this intend of authors the tasks pages can feature a link to this registration page if the link is given in the tasks file in the Registration field.

8.2.1 Configuring Web Sentinel pages per Blend

To set some typical values for the web sentinel per Blend each Blend has to provide a configuration file. These files are formatted in RFC822 files and maintained in Git. The following values can be set:

Blend Id of the Blend (for instance debian-edu, debian-med, etc.)

ProjectName Name the Blend in correct spelling. Please note that English spelling is without a dash and it is "Debian Edu" (not Debian-Edu) and Debian Med (not Debian-Med)

ProjectUrl URL to technical information about the Blend

Homepage URL to the user oriented homepage of the Blend

LogoUrl URL to a logo image of the Blend

ProjectList E-Mail address of a mailing list where developers and users of the Blend are communicating

PkgList E-Mail address of a mailing list where developers of the Blend are discussing technical details of packaging

OutputDir Directory where to put the rendered HTML pages. The Blends pages are hosted on blends.debian.org and usually stored at `/srv/blends.debian.org/www/<ProjectName>`

DataDir Directory where the rendering code stores temporary data like Git clones of tasks files etc. This is usually set to `/srv/blends.debian.org/data/<ProjectName>`

VcsDir URL of tasks files in Blends VCS

CSS In case a CSS file different from the default Blend CSS is wanted for a specific Blend a (relative) path to this file can be specified.

8.2.2 Debian Description Translation Project

The Debian Description Translation Project (see Section 6.1.5) provides users of non-english languages with information about Debian packages. The sense of supporting especially the translations of descriptions which are in the focus of a Blend is to make the Blend even more usable for our target users. Moreover people interested in the special field of the Blend are most probably able to provide good translations if it comes to texts that are specific to their field of knowledge. Thus there is a web page automatically created that parses the tasks packages for package names and verifies the translation status of the package descriptions.

Finally the DDTP descriptions are used to create internationalised pages of existing packages which might help users with insufficient skills in English to easily find their software of interest. If the browser locale is properly set the user will get translated descriptions of existing packages - provided that the DDTP translations for all these packages are existing.

8.2.3 Features of the web sentinel tasks pages

For so called prospective packages - packages which are not yet in Debian as discussed above - only the information given explicitly in the tasks file can be provided. For official packages the Debian infrastructure provides a lot of metadata, that is collected in the Ultimate Debian Database (UDD). The script which is creating the web sentinel pages collects all this data from UDD and tries to provide the most comprehensive overview over a set of packages for a given task of a Blend. The following list gives an overview over the metadata of packages belonging to the tasks of a Blend.

Short and long Description If there is a DDTP translation the descriptions are translated.

Homepage The Homepage field is taken from the `debian/control` file. If this information is missing for some package on the tasks page it might be a sign that the package is not properly maintained and deserves a wishlist bug report to add this information (at least for non-native Debian packages.

Maintainer / last Uploader Both are provides as mailto-links. If the latest Uploader is different from the Maintainer this information is given as well. This is specifically interesting for group maintained packages - actually a tendency in Blends to maintain packages as Blends team - where the maintainer is the Blends team and the Uploader is the name of the actual developer who uploaded the package.

Popularity contest The popularity contest database contains different values. The tasks page is presenting the most relevant of them: the number of people who really use this package regularly and the number of people who upgraded this package recently

Versions and architectures A button can be used to uncover the versions of this package in the Debian package pool as well as the architectures for which this version exists. The button is coloures in yellow if there is a new upstream version available which enables the Blends packaging team to easily detect work items to do.

8.3 Bugs overview

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

If you want to see examples for those bugs pages just have a look at the list of all Blends using this technique and follow the links to the bugs pages once you selected one specific Blend.

8.4 Versions in stable / testing / unstable / new / VCS upstream

The Debian GIS team has named this overview "Thermometer" - just find the Debian GIS thermometer as example view here.

8.5 Quality assurance report

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

Chapter 9

To do

9.1 Establishing and using communication platforms

Each Debian Pure Blend has an own mailing list for discussion of specific development issues. Because there are several common issues between all Blends also a common mailing list was created. People who are interested in working on common issues like building metapackages, technical issues of menu systems or how to create CDs for Blends could subscribe to this list or read the list archive.

Moreover the project Blends on Salsa exists to organise the cooperation of developers. The Git repository can be browsed or checked out by by

```
  git clone https://salsa.debian.org/blends-team/blends.git
```

```
for anonymous users. Developers should check out via
```

```
  gbp clone git@salsa.debian.org:blends-team/blends.git
```

```
The current layout for the repository is as follows:
```

```
  blends -+- blends (code of blends-dev and this documentation)
          |
          +- website (code to create the web sentinel + other tools)
          |
          +- med     (Debian Med)
          |
          +- science (Debian Science)
          |
          +- ... (most other Blends)
          |
          ...
```

There is a mailing list with subversion changes and a CIA system for tracking changes in the Debian Pure Blends projects in real-time.

9.2 Enhancing visibility

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

In bug #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 Blends would be included in the task selection list.

How to enhance visibility of Debian Pure Blends 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 Blends could be listed under this topic in the task selection list.

Debian Pure Blends 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 Debian Pure Blends (perhaps pointing to this document for further reference) and enables the user to select from a list of the available Blends.

Provide separate install CDs By completely ignoring the installation of the official installation CD each Blend 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 Blend 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 Section 9.6. This would save the problem of making Debian Pure Blends visible to users in a completely different way because in this case Debian would be released as its various flavours of Blends.

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

9.2.1 Debian Pure Blends web pages

Some Blends maintain their own web space under `http://www.debian.org/devel/BLEND-name` to provide general information which will be translated by the Debian web team. This is a good way to inform users about the progress of a project. This page should link to the appropriate autogenerated pages as described in Section 6.2.4 to make sure that the content of the page remains up to date at any time.

9.3 Debian Package Tags

Debian Package Tags is a work to add more metadata to Debian packages. At the beginning it could be seen as a way to allow to specify multiple sections (called "tags") per package where now only one can be used.

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

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

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

For example, Debian Edu could look for "edu::administration" packages and then select "use::configuring". The "edu::administration" classification would be managed by the Debian Edu people, while "use::configuring" would be managed by Debian. At the same time, non Debian Edu users looking for "use::configuring" could have a look at what packages in that category are suggested by the Debian Edu community.

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

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

9.4 Enhancing basic technologies regarding Debian Pure Blends

In section Section 6.2.5 several issues where raised how handling of metapackages should be enhanced.

Currently there is no solution to address the special configuration issue has to be addressed. In general developers of metapackages should provide patches for dependent packages if they need a certain configuration option and the package in question does feature a debconf configuration for this case. Then the metapackage could provide the needed options by pre-seeding the debconf database while using very low priority questions which do not came to users notice.

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

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

```
/usr/share/blends/#BLEND#/#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 from freedesktop.org but it is not even yet packaged for Debian.

9.5 Building Live CDs of each Debian Pure Blend

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 - *the "mother" of all Debian-based live CDs* - is a really great success and it is a fact that can not be ignored that Debian gains a certain amount of popularity because people want to know what distribution is working behind the scenes of Knoppix.

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

The good news is that the problem of orphaned or outdated Live CDs can easily solved by debian-live and the `live-helper`. This package turns all work to get an up to date ISO image for a Live CD into calling a single script. For the Blends tools this would simply mean that the tasks files have to be turned into a live-helper input file and the basic work is done. This will be done in a future `blends-dev` version.

9.6 New way to distribute Debian

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

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

There was a suggestion by Bdale Garbee bdale@gag.com to think about kind of sub-setting Debian in the following way: Debian developers upload their packages to `unstable`. The normal process which propagates packages to `testing` and releasing a complete `stable` distribution also remains untouched. The new thing is that the package pool could be enhanced to store more package versions which belong to certain subsets alias Debian Pure Blends 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
         |
         +--->  BLEND_A testing  -->  stable BLEND_A
         |
         +--->  BLEND_B testing  -->  stable BLEND_B
         |
         +--->  ...
```

where BLEND_A / BLEND_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. Unfortunately this goal was not realised finally.

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 Blend 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 Blend 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 Debian Pure Blends 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 BLEND_A
                                   |
                                   +--->      stable BLEND_B
                                   |
                                   +--->  ...
```

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

```
           testing_proposed_updated
                      |
                      |
                      v
DD -> unstable --> testing --> stable
                      |
                      +--->    stable BLEND_A
                      |
                      +--->    stable BLEND_B
                      |
                      +--->  ...
```

The rationale behind these testing backports is that sometimes a Debian Pure Blend 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 Blend. For instance this might make sense for Debian Edu where usually neither mainframes nor embedded devices are 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 Debian Pure Blends 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 Blend, would take over support only for the subset of packages which is included in the metapackage dependencies or suggestions but they refuse to provide full support for the whole range of Debian packages. This would lead to a new layout of the file structures of the Debian mirrors:

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

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

Regarding to security issues there are two ways: Either one Debian Pure Blend 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 a certain Blend they should be made reliable for this certain subset.

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

Appendix A

Description of development tools

A.1 Package `blends-dev`

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

The usage of the tools in the `blends-dev` package might introduce a versioned dependency in the `<blend>-config` package from which all other metapackages of the `Blend` in question will depend. This `<blend>-config` package instantiates the `Blend` in the common registry for all Blends in `/etc/blends`.

The version `0.7.0` of `blends-dev` uses UDD to generate Blends' metapackages. Currently all Blends' info is stored into UDD. Information such as VCs, description, homepage etc for a Blend can be found into the blends_metadata UDD table. All the info about Blends' tasks and their package dependencies are also stored into the blends_tasks and blends_dependencies_alternatives tables. Having the latter in combination with other UDD tables (such as a table with info about all Debian available packages) provides the ability to check whether a package exists for an architecture or not thus `blends-dev` can generate architecture dependent metapackages.

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

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

(see `/usr/share/doc/blends-dev/examples/Makefile`). Users using `blends-dev 0.7.0` on existing Blends which have more than one releases might encounter some `Makefile` errors for more info see Section A.1.3 and Section A.1.4. Using this `Makefile` all tools that were contained in `blends-dev` package versions before 0.4. These tools are moved to `/usr/share/blends-dev/` because there is no need to call them directly. Here is a list of the `make` targets.

A.1.1 Blend-`tasks.desc`

This target generates a `task-description.template` file. The template can be converted to a proper description file that is used in `tasksel` to enable selecting the tasks belonging to the Blend. The initial template contains all the needed package dependencies for a Blend. But because some packages might not be available for a(or multiple) architectures the template uses the following syntax when specifying packages:

```
package1 [!arch1 arch2]
```

That says do not include the package1 in the taskdescription file when arch1 or arch2 is used. When a Blends' `orig.tar.gz` is generated, the initial template gets converted from the `blends-dev rules` file to a proper taskdescription file. The conversion is filtering out the packages which are not available for the host's (where the `orig.tar.gz` is generated) architecture. This make sure that the taskdescription file will not include package which are not available for the target architecture. Finally the file will be moved to the `blend-tasks`. All information about Blends package dependencies is obtained from the UDD.

A.1.2 debian/control

The debian/control file of a Blend metapackage source archive is auto generated out of dependencies that are specified in so called tasks files. The rationale behind this is to enhance flexibility about changes inside the Debian package pool where new packages might appear and others might be renamed. The tasks just define which dependencies the Blend maintainer group wants to be fulfilled and the script blend-gen-control using UDD verifies whether these dependencies exist in a specified package pool. A package pool can be considered as the packages available for a combination of distribution, component and release values. By default when creating metapackages debian,main,testing values are used to "create" a package pool from UDD. Once a Blends' dependencies are verified the debian/control file is generated according to the available packages. This works not only for the Debian package pool containing the distributions stable, testing and unstable, but also when building your metapackages against a different package pool of a Debian based distribution. This is for instance used to create the SkoleLinux metapackages.

As mentioned in the previous section, using UDD in Blends' tools provides the ability to generate architecture dependent metapackages. Thus the generated debian/control specifies for every task source target as architecture value:

```
Architecture: any
```

Specifying any indicates that the source package isn't dependent on any particular architecture and should compile fine on any one. To fulfil this in case of missing packages control file uses the following syntax:

```
Depends: package1 [!arch1 !arch2]
```

If a package is not available for a specific arch, exclude it from it. So the above example says: depend on package1 but not when architecture arch1 or arch2 is used. More info about debian/control syntax can be found in Debian Policy Manual

The syntax of the tasks files which serve as the central database for the information in the debian/control file is defined by RFC822. Some of the tags were mentioned formerly in Section 8.1 to explain how the tasks files can be used to create the web sentinel pages. In order to write valid task files it is mandatory to separate a task file paragraph by an empty line. Otherwise the task pages of the web sentinel will not be built correctly. Now the tags that influence the creation of the debian/control file are given.

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

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

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

If a package is not available in the package pool of the target distribution when creating the debian/control file inside the meta package source archive any "Depends" or "Recommends" are turned into "Suggests" to enable a flawless installation of the metapackage. Generally packages are concerned as missing if they do not exist into Debian main component(default is testing release). Packages that are specified with "Suggests" will not be taken over to the tasksel control file (Blend-tasks.desc, see Section A.1.1) but only to the list of suggested packages of the according metapackage.

Ignore The "Ignore" key can be used as kind of "Soft-Suggests" to put a package on the radar of the Blend. Packages that are tagged with Ignore will not be taken over into the list of dependencies inside the debian/control file of the resulting metapackage neither to the Blend-tasks.desc control file for tasksel but will be taken over onto the installation medium of a Blend in case there is some space left. This key becomes especially important for specifying not yet packaged software that might be packaged in the future (prospective packages). This is explained in detail in the paragraph about the web sentinel (see Section 8.1).

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

Example:

```
Task: finest
Description: the finest software

Depends: foo

Depends: bar

Suggests: foobar
```

A.1.3 statusdump

This target generates a json file containing the latest package dependencies for the selected Blend. It parses the files from the `tasks` directory and generates a `blend_version.json` into a `dependency_data` directory. As `version` it gets the latest version specified in the Blend's `debian/changelog` file. In case the `dependency_data` directory does not exist into a Blend's root directory it automatically creates it.

A user can also generate a json dependencies file manually using the `tasks_diff` script. The script can be called from a Blend's root directory:

```
 /usr/share/blends-dev/task_diff --status-dump --tasks .  --output blend_version.json
```

If the user does not specify the output argument the script by default will generate the json file under the `tasks.json` name in the current directory.

Note: in case a user needs to generate a json file for a previous release (rather than the latest) to get the `changelogentry` (see Section A.1.4) target to work, must keep the following thing in mind: The user must provide to `task_diff` script the *root* directory of a previous Blend release (through the --task(-t) argument). He should also save the output into the `dependency_data` directory into the latest Blend release providing manually the name `blend_version.json` (through the --output(-o) argument:

```
/usr/share/blends-dev/task_diff --status-dump -t blend/tags/previous/ -o latest_blend/ ↩
    dependency_data/blend_version.json
```

For example if the name of the Blend is `myblend` and the release is `0.2.0` then the json file must have the name `myblend_0.2.0.js`

A.1.4 changelogentry

This target compares the latest and the previous Blend release and dumps the tasks' package differences. It reports the added/removed packages per task (or added/removed task files) between releases. This "report" is automatically added into the `debian/changelog` in the latest release section under the file's manual changes. In case a previous difference report exists, it overrides it. In case a Blend does not have more than release (initial release) then this target is ignored.

In order the comparison to be properly performed the `blend_version.json` files for the two latest releases must exist under the `dependency_data` directory. In case any of the previous files is missing then the target will fail with an error (specifying the missing version_file). The json file for the latest release is automatically generated from the `statusdump` target so it this will not cause the problem.

This changelog entry is a new feature so the problem of this target failing (because of a missing json file) will appear for existing Blends which have more than one releases and do not have a `blend_version.json` for the previous release under their `dependency_data` directory. Usually Blend's releases are tagged into the VCs, so the previous problem can be solved by generating the dependency json file for the previous release (using a previous VCs tag). This can be done by calling manually the `task_diff` script (see Section A.1.3)

A.1.5 Apt `sources.list` files in `/etc/blends/`

These files are used by blend-gen-control(1) to build valid `debian/control` files that contain only available packages in their dependencies. This enables building meta packages for `stable`, `testing`, `unstable` or even a completely different distribution that has valid `sources.list` entries. The file `/etc/blends/control.list` is used as default for blend-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 blend-gen-control(1).

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

A.1.6 Templates in `/usr/share/blends/templates`

The directory `/usr/share/blends/templates` contains templates that can be used to build a `<blend>-config`, which uses the tools that are contained in the `blends-common` package, and are useful to manage `<blend>` user groups (see Section 6.3).

A.2 Package `blends-common`

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

A.2.1 blend-role(8)

NAME `blend-role` - add/remove roles in registered Debian Pure Blend

SYNOPSIS `blend-role add|del Blend [Role]`

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

OPTIONS

`Blend` A registered Blend in /etc/blends, for example one of `med`, `junior`, `edu` or `science`

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

A.2.2 blend-update-menus(8)

NAME `blend-update-menus` - add menu of metapackage to all Blend users

SYNOPSIS `blend-update-menus [--blend Blend|--user user]`

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

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

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

OPTIONS

`Blend` one of the installed Blends, listed in /etc/blends/, for example (if installed: `med`, `junior`, `edu` or `science`.

`user` system user

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

A.2.3 blend-user(8)

NAME `blend-user` - add/remove user to Role of a registered Blend

SYNOPSIS `blend-user add|del Blend user [Role]`

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

OPTIONS

- **`Blend`** A registered Blend in /etc/blends, for example one of `med`, `junior`, `edu` or `science`.
- **`user`** user to add
- **`Role`** the role in the `Blend` that `user` will assume

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

A.2.4 blends.conf(5)

NAME `blends.conf` - configuration for Debian Pure Blends registry

DESCRIPTION This file is sourced from shell scripts inside the Debian Pure Blends package `blends-common` and thus it has to follow shell syntax. The variables that are set inside this configuration file can be overridden by special Blend configuration files `/etc/blends/Blend>/Blend>.conf` for each single Blend.

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 Debian Pure Blend that override the settings of the general configuration file.
- **`SHAREDIR`** Set the base directory for the user role management system. While this is more or less a feature for debugging this might be also used otherwise.
- **`DRYRUN`** This variable can be set for debugging. Normally it should be left unset (*NOT* set to `false` or anything else!). If set to `true` a dry run of the tools is performed or `echo DRYRUN:` would print debugging information.
- **`DEBUG`** If set to `1` debugging mode is switched on.

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

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

A.3 How to develop new normal packages in Pure Blends

Sometimes you may want to develop new packages for your Pure Blend. It is almost identical to creating a new debian package. But in Blends framework you can do it more easily.

Assume that you have already had a well-defined blend. That is, those necessary files like changelog, compat, copyright and control.stub are in your debian/ folder, maybe with postinst.stub or prerm.stub for task metapackages. Notice that, all these control and maintainer scripts are just for metapackages defined in your task, not for your new, normal packages since they aren't listed in task.

When you want to develop and add new packages, just like debian packaging, you need to add package information and parameters in the control file. However in blends, you should add them in debian/control.stub instead of defining their own control file. For the maintainer script files, you need to add them in debian/ folder. However, notice that the maintainer script files should not add ".stub" or blends-dev won't process them.

For example, in blend "foo", we want to add a new normal package "bar". In debian/control.stub, we may have:

```
Source: foo
Section: misc
Priority: extra
Maintainer: Foo Team <debian-foo@lists.org>
Uploaders: Foo bar <foo-bar@nowhere.com>
Build-Depends: debhelper (>= 7), blends-dev (>= 0.6.91)
Standards-Version: 3.9.3
Dm-Upload-Allowed: yes
Homepage: http://foo.bar/

Package: foo-bar
Architecture: all
Description: sample blends and packages
```

In the control.stub it defines foo blends, and a new package foo-bar. Then, in the debian folder, we can add our own preinst, postinst, prerm, and/or postrm maintainer scripts. However the filename should be foo-bar.preinst, foo-bar.postinst, ... etc., instead of foo-bar.postinst.stub, since foo-bar doesn't exist in tasks.

After those control files are done, you can use

```
make dist
```

to create blends control files, and then use

```
debuild
```

to start packaging.

A.4 Working with the source repository in `Git`

Sometimes it might be interesting for developers to check out the latest code of the Blend tools or a special Blend code for the meta packages. In Section 9.1 the directory layout of the `Git`-repository was described. How to derive the Debian packages from this layout?

Checkout For the Blend tools and this documentation

```
gbp clone git@salsa.debian.org:blends-team/blends.git
```

or for the Debian Pure Blend `BLEND-name`

```
gbp clone git@salsa.debian.org:blends-team//BLEND-name.git
```

Build source package Change into the created directory and type

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

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

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

```
  Which enables you to simply say
```

```
  make dist
```

to create the source tarball.

Build Debian packages gbp buildpackage

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

A.5 How to create tasks and bugs pages of web sentinel

In Chapter 8 the creation of so called tasks pages Section 8.1 and bugs pages Section 8.3 was described. These pages are automatically build by a cron job on blends.debian.org from the current state of the tasks files in the Git repositories of all registered Blends.

To know what a valid `<blend-name>` might be have a look into `/srv/blends.debian.org/webtools/webconf`. Each Blend has an according config file there. Leave out the `.conf` extension and you have a valid `<blend-name>`.

In case you are planing some more experimental changes there is another host (sponsored by rackspace.com) called `blends.debian.n` which is running a copy of UDD and also hosts the latest development snapshot of the Blends web tools. Just ask Andreas Tille tille@debian.org in case you like a login on this host.

The code which builds web and tasks pages is available in Git at `https://salsa.debian.org/blends-team/website`. It is using the Genshi templating system which enables influencing the layout of the pages by editing the templates in the `templates` directory. You can also influence some parameters of the web pages in the configuration files in the `webconf` directory. Last but not least you can provide translations for the web pages in the `po` directory.

Please note that the `css` and `js` files which are influencing the layout of the automatically created pages are in the same area as the static web pages (see below Section A.6).

A.6 Editing static web pages of Blends on blends.debian.org

A very simple entry page is created for each Blend which is linked from the list of all Blends. Most probably the maintainers of a Blend want to enhance this page a bit. Maintainers of a Blend should care for this index page which currently is just featuring links to the automatically updated pages as well as an image which shows the activity of the relevant mailing list. Maintaining these static pages is not a "service" which is done for you. The maintainers of a Blend should care for this!

The static pages are maintained in Git at `https://salsa.debian.org/blends-team/website` in the `websites` directory. A cron job is watching changes on Salsa and will update the static web pages.

A.7 Description how Blends relevant data are gathered and stored

All data relevant for Blends are available in Ultimate Debian Database (UDD). Here is a description of the data gatherers that were explicitly invented to provide information for usage in Blends websentinel.

A.7.1 Packages in Debian ftp new queue

New packages might reside for some time in the so called new queue until a member of the ftpmaster team has evaluated the package as fit for the Debian distribution. To enable informing our users about that status the new queue is parsed and the information displayed in the web sentinel pages. The actual gatherer code can be found in UDD Git and is split into a Script that fetches the data and a Parser which injects the data into UDD.

A.7.2 Machine readable data in Git repositories of Blends and some packaging teams

Several machine readable data of packages are parsed from Git repositories. There are data that might be more up to date in Git (for instance scientific publication data or some registry entries). There are also data for not yet available packages a Blends team is working on. To enable referring to this data some selected Git repositories are parsed. The parser for Salsa is available in the website Git of the Blends project. It is running in a daily cron job on the host blends.debian.net and creates an archive with all the machine readable files found in the specified Git repositories.

This archive is read into UDD by an UDD importer script in a cron job which is totally independent from the job that collects the data from Salsa. Due to the disconnected jobs running on different hosts there might be some undetermined delay between changing some metadata in a Git repository and the time when the data are available in UDD.

Appendix B

Quick intro into building metapackages

There are several descriptions available how to build Debian packages in general. The main resource might be the repository of Debian packaging manuals (especially developers reference chapter 6, best packaging practices). There are several external packaging HOWTOs for example the one from Joe 'Zonker' Brockmeier.

B.1 Defining dependencies for metapackages

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

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

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

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

B.2 The packaging directory

To build any Debian package you always need a directory named debian, which contains a certain set of files. The package blends-dev provides a complete set of example files that only have to be copied and after editing some place holders are ready to use.

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

Now the variables in the file control.stub change the variables named _BLEND_, _MAINTAINER_ etc. to match the names of the Debian Pure Blend to be built. Please note that the file debian/control is and has to be a symbolic link to control.stub to let the blends-dev tools work.

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

Also in the debian/rules the name of the Blend has to be inserted where the template contains _BLEND_. Depending from the way the sources.list should be scanned the options for the gen-control call can be adjusted.

You have to build the tarball using the command

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

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

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

which enables you to simply use

```
~> make dist
```

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

```
~> debuild
```

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

B.3 The common metapackage

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

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

The variables (`_BLEND_`) in these three files have to be adjusted to the name of the Debian Pure Blend in question. This `blend-config` cares for the initialisation of the role based menu system and might contain adjustments of the general configuration inside the `blends-common`.

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

B.4 The metapackage menus

As explained in Section 6.3.1 the metapackages can contain user menus. This optional feature can be implemented easily by using the template from the `blends-dev` in the following way:

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

A menu file for each task should be created containing valid menu entries for each dependent package. The easiest way to obtain those menu entries is to simply copy the original menu entry files that are contained in the packages on which the metapackage will depend. The only thing that has to be changed in these menu entries is the `package` field, which has to be changed from `<dependent package>` to `blend-task`. All other entries might remain unchanged. This is a good point to check whether the menu entries of the packages you depend from are formatted nicely and print the necessary information (for instance make use of "hints"). Here the metapackage maintainer has a good chance for quality assurance work, which is also part of the Debian Pure Blends issue.

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

B.5 Menu for any dependency

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

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

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

Appendix C

Using the Bug Tracking System

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

A very frequently asked question in mailing list is, whether `program_xy` can be integrated into a Debian Pure Blend. 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 how anybody can ask for including a certain piece of software into Debian. It explains how to use the program `reportbug` for this purpose.

In Debian Pure Blends some house keeping of interesting packages is done and once an ITP is issued it should be mentioned at the relevant tasks page to keep other team members informed. To make this happen it is a minimum requirement to at least forward the ITP mail to the relevant mailing list crossing fingers that somebody feels responsible to enter this information into the according tasks file. It is even better if you just include the information yourself (see Section 8.1 for more details).

C.2 How to report problems

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

A detailed explanation how to report errors 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_ for known problems and probably suggested solutions before calling `reportbug`.

Appendix D

FAQ

D.1 How can I add a dependency?

If the source for the Blends package is maintained in Blends VCS every Debian developer and each member of the Alioth team "Debian Pure Blends" have commit permission to the package source. Once you checked it out you find the single tasks files inside the `tasks/` directory. Fine one or more tasks you want to put the package into and add the line.

```
Depends: package
```

or

```
Suggests: package
```

D.2 What additional information should be provided?

As explained in Section 8.2 you can specify several metadata in addition to the pure dependency which is used for the metapackage creation to enrich the web sentinel with extra information about the package. Usually the information is obtained from the Debian package information. However, as it was explained it makes perfectly sense to even specify packages which are not yet included into official Debian. The information is taken from the Ultimate Debian Database (UDD)

Since the Blend team also has injected the packages in the NEW queue as well as the information about packages in VCS of Blends it is sufficient even for packages which are not yet ready to only specify the name. In this case it is detected that the package is work in progress and the tasks page of the web sentinel will put the package into the appropriate section.

The big advantage of injecting a package into the relevant task is that visitory of the tasks page will be informed about the work in progress and can follow for instance enhancements of the desciprion or the migration of the package into the Debian pool without any further action. Since this is very convenient you want to add your package to the tasks immediately after you injected the first rough packaging into VCS.

D.3 Should I add binary or source packages?

A tasks file is in principle a `debian/control` file snippet. In these files you are specifying *binary* packages. It is a common error to inject source package names which is wrong if the resulting package is different.

D.4 Should I add a library package to a user task?

User oriented metapackages should depend from user oriented applications. You should avoid specifying a package containing a dynamic library. If the package you want to add is a library you should rather add the development package (containing the static library and header files) to the according development task if this exists.

D.5 Can I create a new task if the existing ones do not fit?

Yes. Please discuss this with the Blends team on their mailing list but in principle it is totally OK to add a new task to a Blend.

D.6 Why not simply use a Wiki?

People frequently claim that they prefer a simple list in a Wiki to list the interesting packages for their work. This is a waste of time since:

1. An entry in a tasks file makes the same effort (measured in time and numbers of keys pressed).
2. You need to manually add the metadata (like Homepage, description etc.)
3. Metadata are updated automatically in the web sentinel tasks - Wikis are usually aging (despite being a Wiki)
4. You get a lot of metadata for free:
   - *Translated* descriptions
   - Version + release
   - Popularity contest results
   - Debtags
   - Screenshot

   You will have a hard time to provide all this on a Wiki and keep it up to date
5. Automatic information whether Debian is lagging begind upstream
6. If a package was introduced into Debian after the work in VCS was successfully finished the web sentinel is updated automatically without any extra work.
7. If you want to create metapackages for user installation or to easily feed some live DVD you need to create tasks files anyway and thus you are not duplicating any work.

D.7 Why not simply using DebTags?

In general I'd like to quote Enrico Zini about the relation of DebTags and Blends tasks: "Both efforts are orthogonal to each other and a well designed task should cover a specific workfield which is not necessarily given due to the fact that a package belongs to a certain (DebTag) category."

Moreover if you exclusively rely on the DebTags you are ignoring additional options the Blends framework is providing. For instance the Biology task of Debian Med contains several sections about packages which are not yet available in the Debian package pool. This includes:

1. Packages inside the new queue
2. Packages the team is working on in VCS
3. Unofficial packages at some other places
4. Information about not yet packaged software that might be of certain interest

This turns out as a very useful todo list to grab work items from as well as information to prevent duplication of work. (Yes ITP bugs do not always work out that reliably.)

The Blends framework works quite nicely automatically if packages "upgrade" from one section to the other which means if you specify a package which has only some packaging in VCS it is sufficient to simply add a dependency to the tasks file and it shows up here (to stick to our example above). So all relevant information is just taken from VCS and if you for instance change the description in your packaging the tasks page will be updated automatically (after some cron job delay).

Once the package might be uploaded the first time it migrates to the section named "Debian packages in New queue (hopefully available soon)" (which is only available if there are such packages) and finally once ftpmaster might have accepted the package it shows up in the tasks ... even not yet DebTagged. You have another delay until the DebTag information is propagated properly and thus relying on DebTags in this whole process does not work.