

Книга за Debian GNU/Linux

4 юли 2003 г.

Версия 0.1

Copyright © 2002–2003 Александър Велин, Георги Данчев, Дамян Иванов, Никола Антонов, Огнян Кулев, Пламен Тонев, Стоян Жеков

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Съдържание

I	Въведение	1
1	За този документ и потребителите	3
1.1	Достъп до документа	3
1.2	Накратко за съдържанието на книгата	4
1.3	Други книги за Debian на английски език	4
2	Защо Debian GNU/Linux	5
3	Основни понятия и команди	7
3.1	Общи понятия	7
3.1.1	Компютър, хардуер и софтуер	7
3.1.2	Операционни системи	7
3.2	Дистрибуции на софтуер с ядрото Linux	7
3.2.1	Пакети	7
3.2.2	Дистрибуции и пакетни системи	8
3.3	Debian	8
3.3.1	Пакети и техните състояния	8
3.3.2	<i>dpkg</i>	9
	Списък на всички инсталирани пакети	9
3.3.3	<i>apt</i> : Боравене с мрежа от пакети	9
	Практическа употреба	9
	Конфигуриране	9
	Поддържане на мрежата от пакети	10
	Инсталиране и премахване на пакет	10
	Търсене на пакет	10
	Информация за пакет	10
3.3.4	<i>aptitude</i> : Удобният начин	10
II	Проблеми	11
4	Проблеми и задачи при управлението на софтуера	13

5	Потребителски проблеми	15
III	Решения и предложения за подобрения	17
6	Бърз преглед, без инсталация	19
6.1	Knoppix LiveCD	19
6.1.1	От Knoppix в паметта...	19
6.1.2	... към Debian на диска	21
6.1.3	Custom Debian Knoppix	22
6.2	Gibraltar LiveCD	22
6.3	Public accessible Debian Machines	22
7	По-user-friendly debian-installer and custom installer(s)	23
7.1	Официалният debian-installer	23
7.2	<i>pgi</i> : The Progeny Graphical Installer	23
7.3	<i>fai</i> : Fully Automatic Installation for Debian GNU/Linux	24
7.4	SystemInstaller, SystemImager, SystemConfigurator	24
7.5	Replicator	25
8	По-user-friendly десктоп	27
8.1	Debian Desktop Project	27
8.2	Debian Menu System	27
8.3	Package Browser	28
9	Интернационализация, Локализация, Българизация	29
9.1	Въведение	29
9.2	По време на инсталацията на Дебиан	30
9.3	<i>locales</i> : Добавяне на български език	30
9.4	<i>console-cyrillic</i> : Конзола	30
9.4.1	<i>cyr</i> : Команда за конфигуриране на конзолата	31
9.5	<i>XFree86</i> : Графична среда	31
9.5.1	<i>XKB</i> : Клавиатура	31
	<i>xserver-xfree86</i> : Конфигуриране при инсталиране	31
	<i>XFree86Config</i> : Редактиране на конфигурацията на X	32
	<i>setxkbmap</i> : Различна подредба за отделен потребител	32
	Важна корекция	33
9.5.2	Шрифтове	33
	Пакети с шрифтове, съдържащи кирилица	33
	Препоръчвани пакети с шрифтове	33
	TrueType шрифтове	34
	Инсталиране на TrueType шрифтове	34
	Използване като X шрифтове	35
	Използване като Xft шрифтове	35

9.5.3	<i>xfс</i> : Шрифтов сървър	35
9.6	Справяне с някои „упорити“ програми	36
9.6.1	GNOME	36
	GTK	36
	AbiWord	36
	GDM	37
9.6.2	KDE	37
9.6.3	Mozilla	38
9.6.4	Midnight Commander	38
9.6.5	teTex	38
9.7	<i>taskel</i> : Бързо българизиране на Debian	38
9.7.1	<i>language-env</i> : Автоматично конфигуриране на програми	38
9.8	Какво още може да се направи...	39
9.8.1	Превод на сайта http://www.debian.org	39
9.8.2	Превод на официалните документи на http://www.debian.org/doc	39
9.8.3	Превод на официалния <i>debian-installer</i>	39
9.8.4	Поддържане на неофициално <i>apt</i> хранилище	39
10	Други	41
IV	Управление на софтуера	43
11	Общи положения	45
11.1	Официалният архив	47
11.1.1	Поддръжка	48
11.2	Неофициалните архиви	48
11.3	Контролиране на избора на пакети за инсталиране	50
11.3.1	Избор на release, от който да се вземат пакети	54
11.3.2	Възстановяване на стари версии на пакети	54
11.3.3	Приоритети на пакетите	55
11.3.4	Downgrade	55
11.3.5	Виртуални и мета-пакети	56
11.3.6	Алтернативи на <i>dpkg</i> и <i>apt</i>	57
12	Бързи инструкции за работа с <i>apt</i> и <i>dpkg</i>	59
12.1	<i>apt</i>	59
12.2	<i>dpkg</i>	60
12.3	Компилиране на Debian <i>binary packages</i> от <i>source packages</i>	61
12.4	Решаване на dependencies проблеми - <i>автоматично</i>	62
12.5	Справяне с проблеми - <i>ръчно</i>	63
12.5.1	Справяне с някои конфигурационни проблеми чрез <i>maintainer's scripts</i>	63

13 Конфигуриране на пакети - проблеми и решения	65
13.1 Официалните документи	65
13.2 Многото лица на <i>debconf</i>	66
13.2.1 Обработване на конфигурационните файлове на пакетите	66
13.2.2 Манипулация на файлове от други пакети	66
13.3 <i>dh-make</i> : Начално дебианизиране на програма	68
13.4 <i>debhelper</i> : Инструменти за инсталиращите и конфигуриращите скриптове	68
13.5 Разработка и поддръжка на Debian source packages	71
13.5.1 <i>devscripts</i> : Съвременният начин	71
13.5.2 <i>debmake</i> : Старият начин	71
14 Изграждане на дистрибуцията и средства за контрол	73
14.1 Packaging - <i>Debian official maintainer's way</i>	73
14.1.1 Средства за контрол	73
14.2 Packaging - <i>at home</i>	74
15 Компилиране и инсталиране на софтуера - проблеми и решения	81
15.1 Local APT Repositories	81
15.1.1 Създаване и управление на локално apt-хранилище с готови <i>deb</i> -файлове	81
<i>debuild</i> : binary пакети от source пакети	82
15.1.2 <i>apt-build</i> : Инсталиране на пакети от сорс	82
Накратко	82
Какво е необходимо	82
Създаване на собствени пакети с <i>apt-build</i>	83
Създаване на <i>.deb</i> пакети и добавяне в хранилището	83
Полезни процедури с <i>apt-build</i>	83
15.1.3 <i>apt-src, pbuilder</i>	84
15.1.4 <i>kernel-package</i> : Компилиране на ядро по дебиански	84
Накратко	84
Основни процедури	84
Инсталиране на драйвери за <i>ALSA</i> и <i>NVIDIA</i> с <i>kernel-package</i>	85
15.2 <i>stow</i> : Управление на upstream sources	86
16 Сигурност и надеждност	87
16.1 Документи и пакети	87
16.2 Прилагане на security updates за множество машини	88

17 Some Nice Hints and Tricks - Special experience	91
17.1 Как да си направим custom installer за Debian Base и LiveCD image	91
17.2 Използване на вашата домашна директория със CVS	91
17.3 chroot-ed Debian	91
17.4 Take Over Installations	92
17.5 Използване на Debian GNU/Linux при по-необикновени положения	92
17.6 Други платформи	92
17.6.1 HP PA-RISC	93
Хардуер	93
Документация	93
Начало на инсталацията	93
Основни конфигурации	94
Довършителни процедури	95
17.6.2 SGI (netboot)	96
Хардуер	96
Документация	96
Начало на инсталацията	96
Основни конфигурации	98
Довършителни процедури	100
17.6.3 Mosix, OpenMosix и други	101
18 Анализи, оценки, предложения	103
19 More	105
20 Погрешно схващане	107
21 Обобщение	109
V Често задавани въпроси за Debian	111
VI Участие в писането на книгата	115
22 За този документ и авторите	117
22.1 Замисъл и стил на писане	117
22.2 Препоръки към авторите	117
22.3 Какво остава да се направи	118
23 Регистрация в SourceForge	119

24 Работа с L^AT_EX	121
24.1 Инсталиране на L ^A T _E X	121
24.2 Писане на L ^A T _E X	122
24.2.1 Форматиране на текста	122
24.2.2 Списъци	122
24.2.3 Таблици	123
24.2.4 Специални макроси	123
24.2.5 Специални знаци	123
24.2.6 Кавички	124
24.2.7 Тирета	124
24.3 Структура и съдържание - организация на файловете	124
25 Работа със CVS	125
25.1 Достъп до изходните кодове чрез CVS	125
25.2 Бързи инструкции за CVS	125
25.2.1 CVS session with project-x	126
25.2.2 Добавяне на файлове	127
25.2.3 Добавяне на директории	127
25.2.4 Премахване на файлове	127
25.2.5 Премахване на директории	127
25.2.6 Преименуване на файлове и директории	128
25.2.7 CVS и бинарните файлове	129
25.2.8 Заключение	129
25.3 Използване на общодостъпен ключ за достъп до CVS	129
26 Как да генерираме PDF, DVI, Postscript, HTML	131
VII Мотивация и благодарности	133
VIII Лиценз	137
27 GNU Free Documentation License	139
27.1 APPLICABILITY AND DEFINITIONS	139
27.2 VERBATIM COPYING	140
27.3 COPYING IN QUANTITY	140
27.4 MODIFICATIONS	141
27.5 COMBINING DOCUMENTS	142
27.6 COLLECTIONS OF DOCUMENTS	143
27.7 AGGREGATION WITH INDEPENDENT WORKS	143
27.8 TRANSLATION	143
27.9 TERMINATION	143
27.10 FUTURE REVISIONS OF THIS LICENSE	143

Част I

Въведение

Глава 1

За този документ и потребителите

1.1. Достъп до документа

- <http://lug-bg.sourceforge.net/debian-book/> е заглавната страница, където ще се публикуват официалните издания.
- **Детайли относно Anonymous и Developer достъп до CVS¹**, където `modulename` е **debian-book**.
- **Web интерфейс към кода в CVS²**

Най-добре би било, ако всички потребители използват направо кода от това CVS-хранилище, синхронизирайки своето локално копие от него и компилирайки по свое собствено желание различните изходи — HTML всичко в един файл, HTML splitted, PDF и т.н. Така всички бързо и лесно ще се сдобиват с последните промени в хранилището и ще могат да добавят свои добавки, когато намерят за добре. За тези, които не могат или не искат да си настройат съответното обкръжение за компилиране на книгата, и така да компилират всичко при себе си, могат да ползват следните сървъри, които ежедневно синхронизират своето локално копие със CVS хранилището, компилират при себе си и публикуват на своето web пространство. Ако не компилирате сами за себе си, това е препоръчителният вариант, с който ще имате достъп до последните промени правени по съдържанието на документа. Ето и местата от които можете да разглеждате или изтеглите ежедневни издания по HTTP или FTP:

- **Nightly CVS Mirror & Build Locations:**

N	Сайтове	Задачи	Час	Достъп
1	http://zadnik.org/debian-book/	cvsup&build; rsync server	01:00 bg	бърз
2	http://www.logos.goto.bg/debian-book/	cvsup&build	22:40 bg	бърз
3	http://debian.fmi.uni-sofia.bg/~ogi/debian-book/	cvsup&build	05:00 bg	бавен
4	http://mirrors.ethereal-space.net/debian-book/	rsync от N 1	04:40 bg	бърз
5	http://www.mathematik.uni-karlsruhe.de/~kuhlmann/debian-book/	rsync от N 1	04:40 bg	бърз

В директория `utils/` се намират скриптовете `debuild.sh`, които са все още тестови, но използвани. Ако не разбирате за какво са и как го правят, по-добре не ги ползвайте.

Цялата документация и информация, необходима на потребителите, за да генерират документа на своите машини, както и такава за участие в проекта и лиценза, под който се разпространява, идва със сорс-кода.

¹http://sourceforge.net/cvs/?group_id=50701

²<http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/lug-bg/debian-book/>

1.2. Накратко за съдържанието на книгата

Тази книга се разработва в свободното време с помощта на \LaTeX CVS. Предназначен е за бързо нахвърляне, съхранение и споделяне на информация между множество потребители, и то на български език, за неща, уникални за Debian, които може би не се срещат като функционалност при други системи или са залегнали лошо като замисъл и/или реализация. Това не значи, разбира се, че Debian е безгрешен. Интересно е да се споделят опит и знания за възможностите, които може и да не се срещат в официалната документация на Debian. Документът към момента се опитва да стане по-добър от стилистична, дизайнерска и техническа гледна точка, така че подобна помощ, а както и всякакви конструктивни критики и коментари по съдържанието, се приемат с удоволствие. Все още не е добре филологически и педагогически издържан и довършен и така ще е поне докато махнем и последното FIXME, включително и това. От методична гледна точка на излагане на материала, с цел по-лесното му възприемане, ако някой филолог може да внесе корекции, да заповеда с конкретни предложения, в случай, че това писание все още изглежда неразбираемо по един или друг начин. Идеята не е точно описание на конкретен Release на Debian, това е преходно и е описано на доста места, затова ще гледаме малко по-глобално. Също така, няма за цел да се обяснява как се работи с различните команди, за това си има официална документация, идваща със системата (която ще бъде посочвана, където е необходимо). Важното е да се посочи идеята и дизайна, които са разширявани, допълвани, тествани и доказали своята гъвкавост и работоспособност през годините, а самата имплементация може да се разгледа в съответните изходни кодове.

Предполага се, че имате някаква идея какво е [Linux](#)³ и [GNU/Linux](#)⁴, но все пак силно препоръчително е да се прочете внимателно и задълбочено [Linux kernel mailing list FAQ](#)⁵, където се изясняват неща по принцип, като не е нужно да се записвате в този списък, разбира се. Ако неразбирате или не ви е ясно нещо от гореспоменатото, то се препоръчва да започнете първо с <http://linux-book.hit.bg>, която дава добро общо въведение в средата на GNU/Linux.

Ще е добре да имате Debian наоколо, инсталиран на хард диска на някоя машина, или ако нямате такава с инсталиран Debian подръка, можете да заредите на произволна x86 машина (PC) [Knoppix](#)⁶ от CDROM (без да предприемате каквито и да било интервенции по харддискете, виж по-долу) за да е по-лесно схващането на нещата, защото само с четене от този документ е доста по-трудно за разбиране. Може би ще ви е интересно да прочетете [Историята на проекта Debian](#)⁷ (от пакета `debian-history`) и отговорите на [Често Задаваните Въпроси за Debian](#)⁸ (от пакета `doc-debian`).

1.3. Други книги за Debian на английски език

Следните книги могат свободно да се четат през Интернет:

- [Debian Reference](#)⁹
- [The Debian Universe](#)¹⁰
- [Guide to Debian GNU/Linux Desktop Survival](#)¹¹

³<http://www.kernel.org>

⁴<http://www.gnu.org/gnu/linux-and-gnu.html>

⁵<http://www.kernel.org/pub/linux/docs/lkml/>

⁶<http://www.knoppix.org>

⁷<http://www.debian.org/doc/manuals/project-history/>

⁸<http://www.debian.org/doc/FAQ/>

⁹<http://qref.sourceforge.net/>

¹⁰<http://www.debianuniverse.com/>

¹¹<http://www.togaware.com/linux/survivor/>

Глава 2

Защо Debian GNU/Linux

Debian GNU/Linux¹ не е току-що появила се дистрибуция на операционна система, но като че ли не е достатъчно популярна и позната сред новите потребители. Затова този документ е предназначен предимно за тях, но също и за по-напредналите, които въобще не подозират какво се крие зад Debian GNU/Linux.

Нека направим и уговорката, че това изложение е направено от следната позиция:

- Дистрибуцията трябва да бъде чиста и да пази свободата — [Debian Social Contract](#)², [Debian Free Software Guidelines](#)³, [Free Software](#)⁴ — това гарантира, че ще се разработва, оценява и излага безпристрастно и само и единствено от технически аспект, без влагане на какъвто и да било друг нюанс. Така няма да можем да бъдем обвинени в недобросъвестна или неясна користна реклама. Ако трябва да бъдем честни и безмилостни към съществуващите проблеми, първо трябва да започнем с тях, като постепенно ще се преминава към излагане на силните страни на дистрибуцията. Това гарантира, че читателят няма да бъде заблуден или оплетен още в самото начало с гръмки и сложни велики изречения, сякаш проблеми едва ли не няма, като ги споменаваме бегло в края. Дори се приема, че читателят, осведомен за проблемите, може ще се откаже от понататъшно четене ако е на мнение, че тези проблеми са прекалено големи за него.
- Дистрибуцията трябва да предоставя начин или схема за добре обмислено доставяне на софтуера в произволно големи количества, в предварително компилиран вид и като изходни кодове, които тя предлага в различните свои версии до потребителя, т.е. да е чисто `installable` и `upgradeable`, като освен това трябва да обучава потребителя кое как се прави и защо се прави по този начин, чрез съответната документация, стил, политика и изходни кодове.

¹<http://www.debian.org>

²http://www.debian.org/social_contract

³http://www.debian.org/social_contract#guidelines

⁴<http://www.debian.org/intro/free>

Глава 3

Основни понятия и команди

3.1. Общи понятия

3.1.1. Компютър, хардуер и софтуер

Да започнем с най-простото. Компютрите са създадени, за да си вършим по-добре работата. Това се постига с използването на най-различни *програми*, които носят общото име *софтуер*. Програмите доставят функционалността, която се очаква от компютъра. Това обаче не е физическият компютър, който виждаме. Този физически компютър, заедно с всички прикачени към него устройства, наричаме *хардуер*.

Софтуерът и хардуерът са двете части, които заедно образуват това, което наричаме *компютър*. Софтуерът е нещо като „духа на компютъра“.

3.1.2. Операционни системи

В миналото програмите са се грижили за цялото общуване с хардуера, което е било тежка задача. Части от програмите са се повтаряли в толкова много програми, че те се отделяли в *библиотеки*.

Това обаче не се оказало достатъчно. Нужно било постоянно в паметта да стои програма, която да се грижи за връзката между програмата и другите програми, както и между програмата и периферията (клавиатура, екран). Тази програма се нарича *ядро*, защото стои в центъра на дейността на компютъра и всичко останало зависи от нея.

Комбинацията от ядро, *стандартни библиотеки* и много често използвани програми се нарича *операционна система*. Операционната система дава основния тон на цялото общуване между човек, програми и хардуер.

3.2. Дистрибуции на софтуер с ядрото Linux

3.2.1. Пакети

Това, което обикновено се нарича „програма“, в света на операционните системи с ядро Linux, както и във вариантите на BSD, се нарича *пакет*. Пакетът е основната единица за функционалност в операционната система. Затова най-важните задачи на една операционна система са инсталирането и премахването на пакети.

3.2.2. Дистрибуции и пакетни системи

Въпреки общото име „пакет“ има много различни начини да се реализират действията с пакетите. Така се образуват *пакетните системи*. Допълнително диференциране се получава от *начините на пакетиране*, които могат да бъдат различни за една и съща пакетна система. Именно тук стигаме до понятието *дистрибуция*, което значи организирано пакетиране на програми, при което полученият резултат се разпространява.

Например дистрибуцията **Red Hat**¹ използва пакетната система **RPM**², също както дистрибуцията **Mandrake**³. Въпреки общата си пакетна система тези дистрибуции са различни, защото различни организации пакетираат програмите, може би даже по различен начин.

3.3. Debian

Debian е една от многото дистрибуции на софтуер, които използват ядрото Linux. Нейната пакетна система е призната за една от най-развитите. Разбирането на пакетната система е важно, защото чрез нея се осъществява цялото управление на софтуера.

3.3.1. Пакети и техните състояния

Пакетите имат различни състояния. В началото на всичко стои *първоначалният сорс* (*upstream source*). Това е сорсът на програмата, който не зависи не само от дистрибуцията, но и от операционната система. Задачата на дистрибуцията е да събере много такива първоначални сорсове в едно подредено цяло. За целта често се налагат леки промени в този първоначален сорс, които се наричат *кръпки* (*patches*, в `.diff` файлове). За да се образува пакет, освен кръпки е нужно и да се напишат правилата, по които да се образува дебиански пакет. И така, първоначалният сорс, кръпките и дебианските правила образуват дебиански *сорс пакет*, който се състои от следните файлове:

- `пакет_версия.dsc` е текстово описание на сорс пакета
- `пакет_версия.diff.gz` е кръпка, която включва дебианските правила (този файл може понякога да липсва)
- `пакет_версия.orig.tar.gz` е първоначалният сорс

От един сорс пакет могат да се образуват един или повече пакета, готови за инсталиране (*binary packages*), които имат вида `пакет_версия_архитектура.deb` и затова се наричат също *deb-файлове*.

След инсталирането си пакетът минава в ново състояние и става *инсталиран пакет*. От един файл-архив, какъвто е всъщност `deb`-файла, той се преобразува в множество файлове, записани на точно определени места във файловата система. Процесът на инсталиране не е само разархивиране на `deb`-файла. Той включва конфигуриране, както и интегриране в цялата система. Пример за интегриране е стартирането на сървъра `apache` веднага след инсталирането му. При премахване на пакет файловете му се изтриват, но без конфигурационните файлове, и пакетът се премахва като част от системата — например `apache` се спира. Ако пакетът е само разархивиран, но не е конфигуриран или не е интегриран, той се нарича *полуинсталиран пакет*, отразявайки половинчатото му състояние. Счита се, че ако има полуинсталиран пакет в системата, то цялата пакетна система не е в нормално състояние и трябва да се поправи.

Точно тези процеси около инсталирането и премахването на пакети са част от всепризнатата гъвкавост и удобство на пакетната система на Debian. Това обаче съвсем не е всичко, защото `apt` добавя още удобства.

И така, пътят на пакетите протича така:

- първоначален сорс

¹<http://www.redhat.com/>

²<http://www.rpm.org/>

³<http://www.mandrakelinux.com/>

- сорс пакет
- пакет, deb-файл
- полуинсталиран пакет
- инсталиран пакет

3.3.2. *dpkg*

Пакетът в Debian, чрез който се управляват другите пакети, се нарича *dpkg*. С командата *dpkg(8)* могат да се инсталират конкретни deb-файлове, както и да се премахват инсталирани пакети. Конфигурирането и интегрирането също се включват в тези процеси. Освен тези процеси *dpkg* се грижи и за точното състояние на инсталираните и полуинсталираните пакети.

Обикновено няма да се налага да използвате *dpkg(8)*, освен ако не поправяте състоянието на пакетната система или не сте изтеглили deb-файл от Интернет. Инсталирането на пакет се извършва така:

```
# dpkg -i xpdf_2.02p11-1_all.deb
```

Списък на всички инсталирани пакети

Такъв списък може да се получи с командата

```
$ dpkg -l
```

3.3.3. *apt*: Боравене с мрежа от пакети

От чисто потребителска гледна точка *dpkg* не е много удобен за употреба. Причината за това е, че почти винаги желаният пакет *зависи* от присъствието (да бъдат инсталирани) на други пакети. От своя страна тези пакети могат да зависят от други пакети. Всичко това може да превърне едно просто инсталиране на пакет в дълга и досадна рутина. Но това не е всичко. Възможно е всички тези пакети да бъдат разхвърляни на различни компактдискове, Интернет сайтове или места в локалната мрежа.

Решението на този проблем е пакетът *apt*, който изцяло стъпва върху *dpkg* за основните задачи по инсталиране и премахване на deb-файлове. Самото *apt* се грижи по доставката на тези файлове.

Основният конфигурационен файл на *apt* е *sources.list(5)*. Той съдържа списък на всички места, наричани *източници*, откъдето *apt* да взема пакети. Всеки източник съдържа списък на пакети, които могат да се вземат от него. Така на разположение на *apt* стои едно голямо множество от пакети, които могат да бъдат инсталирани. Както беше казано и по-горе, пакетите могат да зависят от други пакети. Например пакет с програма на Perl ще зависи от пакета *perl*. Взимайки впредвид тези отношения между пакети, можем да си представим това множество от пакети като мислена *мрежа от пакети*. Цялата функционалност на *apt* се върти около поддържането на тази мрежа от пакети, така че инсталирането на пакет да доведе до инсталирането и на всички останали нужни пакети.

Практическа употреба

Конфигуриране Преди всяка употреба на *apt* трябва се зададат източниците на пакети в *sources.list(5)*. Във втората фаза на инсталацията на Debian, след рестартирането, се задават въпроси в тази насока, които редактират файла */etc/apt/sources.list*. Примерно съдържание на този файл е следното:

```
deb http://security.debian.org/ stable/updates main contrib non-free
deb http://mirrors.ludost.net/debian stable main contrib non-free
deb http://ftp.bg.debian.org/debian stable main contrib non-free
```

3 Основни понятия и команди

Освен източници в Интернет могат да се добавят и компактдискове с дистрибуцията. Дисковете трябва да се слагат един по един в компактдисковото устройство и да се изпълнява следната команда за всеки един от тях

```
# apt-cdrom add
```

С това се добавя по един ред в `/etc/apt/sources.list` за всеки компант-диск.

Поддържане на мрежата от пакети Мрежата от пакети не е статична и непроменяща се с времето. Списъкът от пакети на всеки от източниците може да се мени. Затова тези списъци, а съответно и мрежата от пакети, трябва да се обновяват. Това се осъществява с командата

```
# apt-get update
```

Дори да използвате компактдисковете на стабилната дистрибуция на Debian, вие най-вероятно ще включите като източник `security.debian.org`, както е показано по-горе. Това е източник с пакети, т.нар. *security updates*, които са подновени след издаването на стабилната дистрибуция, защото правят системата уязвима на атаки. Този източник съдържа същите пакети с (в повечето случаи) същите версии, но подновени така, че да не застрашават сигурността на системата. Затова присъствието на източника е много важно, ако сте свързан към Интернет или локална мрежа. От само себе си се разбира, че от време на време трябва да обновявате мрежата от пакети, за да можете да „виждате“ тези коригирани пакети.

Повече подробности могат да се намерят в [страницата в сайта на Debian относно сигурността](#)⁴. В случай на нов коригиран пакет е достатъчно да изпълните

```
# apt-get update
# apt-get upgrade
```

Инсталиране и премахване на пакет След като източниците са конфигурирани и мрежата от пакети е обновена, инсталирането на пакети се заключава в изпълнението на командата

```
# apt-get install xpdf kernel-package
```

В случая тази команда инсталира пакетите `xpdf` и `kernel-package`, инсталирайки допълнително всички нужни пакети.

Премахването на пакет също е лесно:

```
# apt-get remove xpdf
```

За съжаление това няма премахне пакетите, които са били инсталирани само за да може `xpdf` да се инсталира.

Търсене на пакет Пакети могат да се търсят лесно, ако се използват подобрени ключови думи, които да се търсят чрез `apt-cache(8)` в описанието на всички пакети. Пример за търсене е следната команда:

```
$ apt-cache search pdf viewer
```

Информация за пакет Командата `apt-cache(8)` може да се използва и за тази цел:

```
$ apt-cache show xpdf
```

3.3.4. *aptitude*: Удобният начин

Освен конфигурирането почти всички действия на `apt-get(8)` могат да се извършат и с интерактивната програма `aptitude`. Употребата ѝ е препоръчителна за новаци.

⁴<http://www.debian.org/security/>

Част II

Проблеми

Глава 4

Проблеми и задачи при управлението на софтуера

При дистрибутирането на какъвто и да е софтуер, проблеми и задачи за решаване винаги ще има. Нещата могат да се подхванат наистина издълбоко така, че и най-малките подборности и крайни случаи да бъдат отчетени и обработвани по определен начин. Важното е потребителят да не бъде принуден да използва точно определена версия на даден софтуер, ако това не се налага, а да му се даде възможността да маневрира в рамките на безопасното и надеждното. Или с други думи казано, взаимовръзките между различните части на вашата система могат да бъдат контролирани в полза на потребителя. По-нататък ще разберете как става това.

Глава 5

Потребителски проблеми

Недоволни или недоразбрали потребители винаги ще има. Ето няколко примера, които може вече и да не са актуални:

- Доколко използваема от потребителите, в това число и българските разбира се, може да бъде дистрибуция като DebianGNU/Linux.
- По принцип Unix и Unix-like операционните системи не са много снизходителни към новите потребители:
- [How to fix the Unix configuration nightmare](#)¹
- забележете доколко friendly оценяват и Mac OS X, който също е смятан за Unix.
- Доколко снизходителен към потребителите си към момента е Debian: [An Unbiased Review of Debian 3.0](#)²

Оплаквания от объркани потребители, които в повечето случаи са основателни:

- [why kde and gnome's menu situation sucks](#)³
- [Make Debian better](#)⁴

¹<http://www.cat.org.au/maffew/cat/unix-config.html>

²<http://debianplanet.net/node.php?id=831>

³<http://lists.debian.org/debian-devel/2002/debian-devel-200210/thrd3.html#01391>

⁴<http://lists.debian.org/debian-devel/2002/debian-devel-200210/msg01400.html>

Часть III

Решения и предложения за подобрения

Глава 6

Бърз преглед, без инсталация

6.1. Knoppix LiveCD

6.1.1. От Knoppix в паметта...

Knoppix¹ е самостоятелен проект, отделно от проекта Debian, но много удобен и бърз начин човек да се запознае с GNU/Linux, и в частност с доста вкултурен Debian GNU/Linux, инсталиран на CDROM.

Поради това, че проектът Debian засега не предоставя официално т.н. **LiveCD**², се препоръчва като такова да се ползва именно Knoppix. По-добро решение трудно ще бъде измислено, като освен това може да послужи и като `debian-installer`. Трябва ви само PC, което да може да зарежда операционна система от диск в CDROM-устройството, или пък ако не може да boot-ва от CDROM, да има флопи дисково устройство, за да заредите от него с boot-ващата дискета на Knoppix, която може да създадете от флопи имиджа, който е на CDROM-диска. Не е необходимо да инсталирате нищо на хард диска (но при желание и това може да стане), дори може да няма и харддиск на машината. Авторът на тази инсталирана на CDROM система **ползва пакети от Debian**³, като освен това е добавил доста код от себе си за разпознаване на хардуера и решаването на задачи, специфични за системи, инсталирани на `read-only` медия, каквато е CDROM-дискът, като `root filesystem`, която се зарежда в `RamDisk`, т.е. в паметта, заключени потребителски акаунти и много други.

Повече обяснения за тази система върху CDROM ще намерите в **документацията**⁴. Добре ще е да се запознаете с цялата документация, за да можете да използвате повече възможности, предлагани от Knoppix LiveCD. Обърнете внимание на:

- **FAQ**⁵ (наличен и като **един файл**⁶), този документ ще е доста полезен, включително и ако ви се наложи да си създадете boot-ваща дискета, ако машината ви не може да зареди направо от CDROM и нямате `bootable network card`.
- ще е интересно да научите как може да изпълните **remote booting**⁷ за клиенти, които нямат CDROM-устройства, но пък разполагат с `bootable network card`, която се поддържа от Linux ядрото и са в мрежа, в която е достъпен терминален сървър.

Специфичните за Knoppix сорсове, отнасящи се до разпознаването на хардуера, можете да получите от <http://www.knopper.net/download/knoppix/> или погледнете в директория `/usr/src/`,

¹<http://www.knoppix.net>

²<http://www.debian.org/CD/faq/#live-cd>

³<http://download.linuxtag.org/knoppix/packages.txt>

⁴<http://www.knoppix.net/docs/>

⁵<http://www.knoppix.net/docs/KnoppixFaq>

⁶<http://download.linuxtag.org/knoppix/KNOPPIX-FAQ-EN.txt>

⁷<http://www.knoppix.net/docs/index.php/FaqPXE>

след като заредите от Knoppix CDROM-диска. Доста от този код на автора е оценен като полезен и се приема в официалния Debian архив. [Форумът](#)⁸ и [пощенския списък debian-knoppix](#)⁹ са добър източник на допълнителна потребителска и развойна информация.

След като заредите Knoppix от CDROM-диска, ще бъде направено опознаване на хардуера, който имате, и съответно ще бъдат заредени необходимите драйвери, като в крайна сметка ви се стартира графична сесия (може да промените поведението при зареждане с подаването на [cheatcodes](#)¹⁰, които може да разберете с F2, когато в началото ви се подава `boot:` промпта). Два акаунта (`root` и `knoppix`), с които системата идва по подразбиране, са заключени, но това не е проблем. Не е нужно дори да знаете техните пароли, за да ги смените, а всъщност те нямат пароли. Дори и като потребител `knoppix` ви е предоставена възможността да изпълните:

```
$ sudo su
```

и ставате `root` (`sudo(8)`, `su(1)`, `sudoers(5)`) и разгледайте файла `/etc/sudoers`), след което може да му смените паролата с:

```
# passwd root
```

(тя ще е валидна само за сесията, т.е. докато рестартирате, и само вие си я знаете, разбира се. Повече информация по този въпрос можете да намерите на CDROM-диска в `KNOPPIX/README_Security.txt`).

Оттук вече, ако имате желание или ви се налага, можете да работите със съществуващите файлови системи на хард дисковете — ако имате такива, да създавате нови дялове и да създавате в тях различни типове файлови системи, които после да монтирате където намерите за добре (т.е. доста мощно `rescue` решение). Без да инсталирате никъде нищо, можете просто да прочетете набързо документацията, специфична за Debian:

- Команди: `dpkg(8)`, `apt(8)`, `dselect(8)`, `aptitude(1)`
- Конфигурация: `sources.list(5)`, `apt.conf(5)`, `apt_preferences(5)`, `deb(5)`, файловете в `/etc/apt/` и `/etc/dpkg/`
- както и документацията, която идва с пакетите в `/usr/share/doc/<packagename>/`

Дори ви препоръчвам този документ да го четете от вашата Knoppix система (без да пипате нищо по хард дисковете), за да поглеждате в нея, докато четете.

Ако нямате възможност да си вдигнете мрежата и да четете този документ от мястото, където се хоства (т.е. от отдалечения `web server`), то го запишете на дискета, монтирайте я и четете от нея.

Достъп до Knoppix `.iso` файлове:

- [download](#)¹¹
- [mirrors](#)¹²
- [bg mirror](#)¹³
- [sources](#)¹⁴
- [KnoppixKDE](#)¹⁵

Достъп до Debian `.iso` файлове:

- [CD доставчици](#)¹⁶

⁸<http://www.knoppix.net/forum>

⁹<http://mailman.linuxtag.org/mailman/listinfo/debian-knoppix>

¹⁰<http://download.linuxtag.org/knoppix/knoppix-cheatcodes.txt>

¹¹<http://www.knoppix.net/get.php>

¹²<http://www.knopper.net/knoppix/index-en.html#mirrors>

¹³<http://mirrors.ludost.net/cd-images/linux/knoppix/>

¹⁴<http://www.knopper.net/download/knoppix/>

¹⁵<http://www.knoppix.net/docs/index.php/KnoppixKDE>

¹⁶<http://www.debian.org/CD/vendors/>

- CD iso файлове¹⁷

Българските потребители ще им е по-удобно да изтеглят Debian и Knoppix CD images от:

- официалния Debian mirror за България¹⁸
- друго mirror¹⁹: тук има Knoppix images и много други.
- още един mirror²⁰

Не бързайте да изтегляте .iso файлове на Debian:

- Може да се окаже, че за вас е по-подходящ някой друг начин на инсталация, като netinst²¹ или инсталация на Debian Base от Knoppix LiveCD със скрипта `/usr/local/bin/knx-hdinstall`
- Също така не е нужно да разполагате с всичките CD-та, първото и второто ще са ви достатъчни, пък и винаги може да доинсталирате и обновите каквото ви е необходимо впоследствие от Debian mirrors.

Ако нямате възможност или не знаете как да дръпнете и изпечете на CD тези images, тогава ви остава да намерите някой, който да го направи за вас, или безплатно и на приятелски начала, ако е ваш познат, или може да се наложи да заплатите разходите по изтеглянето, изпичането и самата празна CDROM бланка, ако не си носите такава. Помнете, че за самия софтуер на Debian и Knoppix не могат да се искат пари, но пък никой не е длъжен да ви пече колкото се сетите на брой CD-та за негова сметка, щото на вас така ви харесва, пък не можете да го постигнете сами. Така че е излишно да се хвърляте на повече от едно-две CD-та, след това много лесно може да се доинсталира и обнови каквото ви трябва, пък го е нямало на тези CD's.

6.1.2. ... към Debian на диска

След като понапреднете малко с материала и сметнете, че искате да имате инсталиран Debian на вашия харддиск (или харддискове), може да опитате да го инсталирате от Knoppix CDROM-диска с помоща на скрипта `/usr/local/bin/knx-hdinstall` и описаното на <http://www.freenet.org.nz/misc/knoppix-install.html> или както е описано в [Install Manual](#)²² за различните хардуерни архитектури. Не бързайте с инсталацията върху хард диск, тя няма да избяга, докато разучите Knoppix-а и документацията на Debian. Добра статия е и [The Very Verbose Debian 3.0 Installation Walkthrough](#)²³. Debian може да се инсталира по много начини, както ще прочетете в наръчника за инсталация, но пък знам, че първо ще се пита за CD's. Тук разнообразието е голямо и сами можете да се запознаете от <http://www.debian.org/CD/>. Там ще прочетете как да си изтеглите официални и неофициални CD images през HTTP или FTP и как по-ефективно да правите това с `jigdo`²⁴, също така са изброени и vendors, които могат да продават CD's (не се заплаща софтуера, а само носител!). Предоставя се и неофициален Net Install bootable CD image, официални такива засега няма. Освен CD images ще намерите и DVD images, като и двата вида могат да се изтеглят и обновяват с `jigdo`²⁵ от пакета `jigdo-file`.

В тази връзка впоследствие обърнете внимание на пакета `bootcd` (`bootcd(1)`). Може да изкопирате вашия running Debian на CDROM чрез скрипта `bootcdwrite(1)` от същия пакет. За генериране на Official Debian CD images си инсталирайте пакета `debian-cd`.

За автоматично разпознаване и конфигуриране на хардуера за така инсталирания Debian на диска има програми като `discover` и `kudzu`, които се ползват и в други дистрибуции. Knoppix

¹⁷<http://www.debian.org/distrib/cd>

¹⁸<ftp://ftp.bg.debian.org/debian-cd/>

¹⁹<http://mirrors.ludost.net/cd-images/>

²⁰<http://mirrors.unixsol.org/debian/>

²¹<http://www.debian.org/distrib/netinst>

²²<http://www.debian.org/releases/stable/installmanual>

²³http://www.osnews.com/story.php?news_id=2016

²⁴<http://www.dirac.org/linux/debian/jigdo/debian-jigdo-mini-howto.html>

²⁵<http://www.debian.org/CD/jigdo-cd/>

LiveCD, например, ползва собствени конфигуриращи хардуера скриптове заедно с модула `cloop`, който вече е в официалния Debian архив благодарение на автора на Knoppix Klaus Knopper — `cloop-src` и `cloop-utils`. Имайте предвид, че ако на вашата система някой драйвър не е компилиран като модул за ядрото или не е закомпилиран в самото ядро, то ще трябва да направите поне едно от двете, за да може да използвате съответния хардуер.

6.1.3. Custom Debian Knoppix

[knoppix-customize](#)²⁶

[Custom Debian Knoppix](#)²⁷;

[Teck2k - Gnome LiveCD, based on Knoppix \(Debian\)](#)²⁸. Изцяло експандирана система. Това е добра идея, ако някой се заеме да реализира `LANG=bg` за Knoppix-based Live CD (Gnome2 и/или KDE3.1), така на новите потребители няма да им се налага още в началото да изтрият старата си любима ОС.

6.2. Gibraltar LiveCD

<http://www.gibraltar.at> — A Debian-based router/firewall distribution, fully workable from a bootable, live CD-ROM. Log files can be stored on a hard disk, and configuration data is stored on a floppy disk and kept on a RAM disk during run-time. (bg mirror <http://mirrors.ludost.net/cd-images/linux/gibraltar>). `gibraltar-bootsupport` — This package contains the necessary support for the live Gibraltar CD-ROM. It will manage `/etc` and `/var` to be writeable and will take care of saving and restoring `/etc`. Therefore this package should be installed on the master copy that will be used as the live file system on the CD-ROM. Do not install it on a system that boots from a harddisk partition (e.g. the development system for creating the bootable CD-ROMs), it will break.

6.3. Public accessible Debian Machines

<http://www.debian.org/News/2003/20030102>

Hewlett-Packard (HP) offers public access to several machines running Debian GNU/Linux through their Test Drive program. Software authors and prospective users are offered an account on those machines in order to find out more about Debian GNU/Linux and a particular HP hardware. Four architectures are supported (Alpha, PA-RISC, IA-32 and IA-64). Compilers are installed to that software authors can test whether their software compiles on those platforms.

²⁶<http://download.linuxtag.org/knoppix/knoppix-customize/ANNOUNCE.txt>

²⁷<http://www.linuxgazette.com/issue87/sunil.html>

²⁸<http://metadistros.hispalinux.es/download.html>

Глава 7

По-user-friendly debian-installer and custom installer(s)

Доста сложна е задачата на инсталационния процес, като се има предвид броят поддържани хардуерни архитектури и начини на инсталация. Вече разбрахме, че като инсталатор на Debian за x86 машини може да се ползва и CDROM диска на Knoppix (`/usr/local/bin/knx-hdinstall`). FIXME: по-точни обяснения за installers...

7.1. Официалният debian-installer

Официалните сорсове можете да намерите на <http://cvs.debian.org/debian-installer/>.

Основните обвинения към официалния инсталатор са към единствения засега текстов интерфейс и скромния hardware autodetecting, но пък от друга страна има и потребители, които не искат и да чуят за повече от това. За да бъдат всички доволни, се подхожда колкото е възможно по-модулно и с „повече лица“, т.е. предлага се общ базов протокол за минималните и задължителни неща, които трябва да присъстват в един такъв инсталатор, справящ се с доста хардуерни архитектури и начини на инсталация, като отгоре му вече се избира опционално видът на интерфейса (text, dialog, debconf, slang, gtk и др.), дали да се прави hardware autodetecting и по какъв начин, с използването на програми като [kudzu](#), [discover](#) и т.н. Това е в процес на разработка, ето и някои примери:

Работа върху инсталационен инструмент¹

7.2. *pgi*: The Progeny Graphical Installer

Този инсталатор има [собствен сайт](#)², както и пакет `pgi`.

[Progeny](#)³ реши да предостави Debian 3.0 Woody i386 installer images, базирани на [PGI](#)⁴ 1.0.1 (свободен лизенз). ISO image (има и [bg mirror](#)⁵) съдържа само базова инсталация, като се прави hardware autodetection и ви се предлага да изберете групи от пакети. След което може да продължите с инсталиране на пакети от външен източник, например насочвайки apt към друго Debian CD, HTTP или FTP mirror и т.н.

¹<http://lists.debian.org/debian-devel/2002/debian-devel-200210/msg01516.html>

²<http://hackers.progeny.com/pgi/>

³<http://www.progeny.com>

⁴<http://archive.progeny.com/progeny/pgi/>

⁵<http://mirrors.ludost.net/cd-images/linux/debian/pgi/>

Има и [снимки](#)⁶ на това, как изглежда началото на инсталацията с PGI 0.9.6.

Дори можете да си създадете инсталатор по ваш вкус и желание, ползвайки като основа кода и документацията:

- [Creating Debian Installers with PGI](#)⁷
- [The Discover Hardware Detection System](#)⁸, която е включена и в Debian като `discover`;
- `autoinstall`: Progeny Debian auto-installation system
- `autoinstall-i386`: Progeny Debian auto-installation system — i386-specific files

7.3. *fai*: Fully Automatic Installation for Debian GNU/Linux

- [официален сайт](#)⁹
- пакет `fai`

Целевата група на FAI са системните администратори, които имат за задача да инсталират Debian на множество машини. Може да се използва за инсталиране на Beowulf cluster, rendering farm, web server farm или linux лаборатория или classroom. Също така large-scale linux мрежи с разнообразен хардуер и различни инсталационни изисквания не са проблем при използването на FAI. FAI е автоматизиран инсталационен инструмент за Debian GNU/Linux, подобен на, но по-добър (според автора) от инструменти като kickstart за Red Hat, yast и alice за SuSE, lui от IBM или Jumpstart за Solaris.

7.4. SystemInstaller, SystemImager, SystemConfigurator

`systeminstaller`

Creates Linux distribution images from a set of packages SystemInstaller creates Linux distribution images from a set of packages and specification files. Working in conjunction with SystemImager and SystemConfigurator, these images can then be installed to machines throughout your cluster/network. As a side-effect, it can be used as a tool for building chroot environments for many package based distributions. Further details can be found at <http://systeminstaller.sourceforge.net> and <http://sisuite.org>.

`systemimager-common`

SystemImager ramdisk for client nodes SystemImager is a set of utilities for installing GNU/Linux images to clients machines over the network. Images are stored in flat files on the server, making updates easy. rsync is used for transfers, making updates efficient. <http://www.systemimager.org/download/>

`systemconfigurator`

Unified Configuration API for Linux Installation Provides an API for various installation and configuration processes that are otherwise inconsistent between the many Linux distributions, and the many architectures they run on. For example, you can configure the bootloader on a system in a general way - you don't need to know anything about the particular boot loader on the system. You can update the network settings of a system, without knowing the distribution or the format of its network configuration files.

⁶<http://hackers.progeny.com/pgi/screenshots/>

⁷<http://hackers.progeny.com/pgi/guide.html>

⁸<http://hackers.progeny.com/discover/doc/guide.html>

⁹<http://www.informatik.uni-koeln.de/fai/>

7.5. Replicator

`debreplicator`: Автоматизирана инсталация през мрежа. Използвайки `nfs-root` файлова система и `rsync`, `replicator` ви позволява интерактивно да използвате като източник компютър с инсталирана система, която да пренесете върху друга машина през мрежата, като взима предвид различията в разделянето на дисковите дялове и в хардуера. Специално проектиран за кълстери, класни стаи и всякакви идентични помежду си компютри, това наистина е най-бързият метод на инсталация.

Който и източник на инсталация с даден `installer` да ползвате, все ще получите стандартен Debian. От липса на `installers` едва ли може да се оплачете ;-)

Глава 8

По-user-friendly десктоп

Като че ли си е достатъчно friendly, но все пак трябва да се угоди на колкото се може повече потребители.

8.1. Debian Desktop Project

[Debian Desktop Project](#)¹ е група в Debian, която работи за един Debian, който да е по-примамлив за desktop потребители.

```
# apt-get install desktop-base
# dpkg -L desktop-base
```

8.2. Debian Menu System

Разработена е прекрасна система за унифициране на менютата на различните графични среди, като maintainers трябва да решат какви менюта ще предоставят: *upstream*, *debian specific* или комбинация от двете, която потребителя може да избере. В самия пакет `menu` се използват програми на *shell* и *C++*, а като описателен език за описване на менютата се използва *XML*.

- [Menu packages](#)²
- [Debian Menu Manual](#)³
- [Debian Menu Sub-policy](#)⁴
- [Предложение за нова Debian Menu System](#)⁵
- Ето и едно още по-ново предложение за Debian Menu System. Enrico Zini [предполага](#)⁶, че настоящата система за десктоп менюта се нуждае от редизайн и би трябвало да се интегрира с вече съществуващи такива. Той предлага да се премине към формата на [Desktop Menu Specification](#)⁷ for desktop entries, и по този начин Debian да продължи да предоставя menu information за приложенията които нямат свои собствена такава.

¹<http://www.debian.org/devel/debian-desktop/>

²<http://packages.debian.org/menu>

³<http://www.debian.org/doc/packaging-manuals/menu.html/>

⁴<http://www.debian.org/doc/packaging-manuals/menu-policy/>

⁵<http://phys251.phy.olemiss.edu/cgi-bin/viewcvs.cgi/>

⁶<http://lists.debian.org/debian-devel-0304/msg00800.html>

⁷<http://www.freedesktop.org/standards/menu/draft/menu-spec/menu-spec.html>

Описание - provides **update-menus**(1L) functions for some applications The intent of this package is to streamline the menu's (like the fvwm2 ones) in debian. For this purpose, menu provides an **update-menus**(1L) command, that will read all installed menu files (as provided by other packages in **/usr/lib/menu**), and run the frontends for various window-managers in **/etc/menu-methods** to create startup files for the window managers (or **pdmenu**(1L)). The user and system admin can easily override the menu files on a *by-user* or *by-system* bases.

FIXME: да се опише по-детайлно... Документацията, както винаги, може да се намери в `/usr/share/doc/menu/`

8.3. Package Browser

Търсене на информация за пакетите и файловете, които предоставят те, може да стане по много начини, включително и през <http://packages.debian.org>. Ето едно алтернативно предложение: Erich Schubert [анонсира](#)⁸ нова версия на неговия **Package Browser**⁹, който да помогне категоризирането на Debian пакетите. Това ще подобри браузването на пакетите през йерархично подредени категории вместо сортирани в секции. Package Browser се базира на data sets от пакета [aptitude](#). Сега тези усилия са прехвърлени към проекта [Debian Package Tags](#)¹⁰.

⁸<http://lists.debian.org/debian-devel-0303/msg01371.html>

⁹<http://debian.vitavonni.de/packagebrowser/>

¹⁰<http://deb-usability.aliioth.debian.org/debtags/>

Глава 9

Интернационализация, Локализация, Българизация

9.1. Въведение

Подробен и достъпно написан наръчник¹ има в официалната документация на Дебиан². Освен това можете да разгледате архивите на списъка debian-i18n@lists.debian.org³. След което, например, може да изгълните

```
# apt-cache search bulgarian
```

за пакети, които имат отношение към българския език. Обърнете внимание на `console-cyrillic` и `language-env`. Ако имате желание дръпнете сорс пакетите в някоя временна директория:

```
# apt-get source console-cyrillic language-env
```

и разгледайте сорса и документацията, която идва с тях. Ще разберете, че локализирането (и, в частност, нас българите ни интересува българизирането) или установяването на дадена езикова среда, на която и да е система, е доста комплексен и сложен процес, зависещ дори и от това какво ще има инсталирано на нея в даден момент. Възможността за четене и писане с азбуката на даден език съвсем не е достатъчно като локализация, за това се добавят и бази данни от думи за правописни коректори, езикови речници, ако щете и счетоводни програми, специфични за законодателството на дадена страна. Подробност, която може би не всички знаят, е, че благодарение на това, че има и български разработчици, включени в проекта Debian, дистрибуцията е най-българизираната до момента сред всички останали. Това ще рече, че в официалния архив се поддържат пакети специално за българските потребители, които се инсталират и обновяват съвсем естествено с всички останали пакети от архива, така че всичко е в синхрон, което не може да се твърди за всички останали, поне към момента. Това, разбира се, няма да „българизира“ програмите, които „не разбират“ от локализация и/или **Unicode**⁴ към момента, но това е проблем на всички, който бавно, но сигурно се отстранява. Добра идея е да се следи и сайта и пощенския списък на <http://debian.gabrovo.com>, като българизирането на Debian съвсем не е единствената тема, разбира се. Списъкът се поддържа от българи за българи. За съжаление все още няма официален списък `debian-{l10n|user}-bulgarian` списък, т.е. няма достатъчно българи, които да спретнат и участват в един такъв списък. Някои ще кажат, че българизацията не е от голямо значение, но това съвсем не е така, защото има потребители, които не мислят така, и е много добре, когато всички са доволни при използването на дадена система.

¹<http://www.debian.org/doc/manuals/intro-i18n/>

²<http://www.debian.org/doc/>

³<http://lists.debian.org/debian-i18n/>

⁴<http://www.unicode.org>

Тук ще ви представим в дълбочина процеса на локализиране на Debian GNU/Linux Woody, за да проследите как наистина стават нещата „отвътре“. Това в изключително голяма степен ще важи и за следващите версии, така че ще се процедира аналогично.

Още статии по темата, но вече не ограничени до Дебиан, могат да се намерят в [секцията за кирилизирани](#)⁵ на [Linux-BG.org](#)⁶. Практически единственото пълно и всепризнато решение на проблема за българизирането на Линукс, независимо от дистрибуцията, е пакетът [bglinux](#)⁷ на [Антон Зиновиев](#)⁸. За щастие той е разработчик в Дебиан и всичко от пакета `bglinux` го има в дистрибуцията под формата на стандартните за нея пакети, така че вие ще ползвате тях.

9.2. По време на инсталацията на Дебиан

По време на инсталацията се задава въпрос за подредбата на клавиатурата. В списъка е и българската подредба, но изберете подразбиращата се подредба (`qwerty/us`). В частта за кирилизирани на конзолата се описва много по-гъвкав начин за кирилизирани на клавиатурата. Последното действие на инсталацията на Дебиан 3.0 е да се стартира програмата `tasksel`, от която бързо може да си инсталирате практически всичко необходимо за българизирана система. На тази програма е отделена секция в края на тази глава.

9.3. *locales*: Добавяне на български език

Инсталира се пакетът `locales` (CD1) и в настройките на `debconf` се задава генериране на българските настройки за `bg_BG`, както и че това е подразбиращият се локал. Поради дефект в пакета `locales` се налага да изпълните командата

```
# dpkg-reconfigure locales
```

което отново задава същите въпроси. (Ако ви се наложи да променят списъка на локалите, не редактирайте файла `/etc/locale.gen`, а използвайте същата тази команда за преконфигуриране на пакета.) По този начин, освен генерирането на информацията за българския, се задава глобалният локал на всички програми да е `bg_BG`. Той се запазва във файла `/etc/environment`, който се използва от [PAM-модула](#)⁹ `pam_env`¹⁰. Програмите, които в PAM конфигурацията си `/etc/pam.d/програма` използват този модул, ще използват този локал. Във файла `/etc/locale.alias` се добавя редът:

```
bulgarian          bg_BG.CP1251
```

9.4. *console-cyrillic*: Конзола

За конзолата е достатъчно да се инсталира пакетът `console-cyrillic` (CD2). На въпросите отговаряйте с подразбиращите се отговори, освен може би на тези въпроси:

Choose the keyboard layout	Bulgarian phonetic или Bulgarian BDS
How to toggle between Cyrillic and Latin letters	Alt+Shift или нещо друго
What is your encoding?	CP1251 (или UTF-8)
Do you want to setup Cyrillic on the console at boot-time?	Yes

⁵<http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&cmd=cat&cid=cyr>

⁶<http://www.Linux-BG.org/>

⁷<http://lml.bas.bg/~anton/linux/bglinux.html>

⁸<http://lml.bas.bg/~anton/>

⁹<http://www.kernel.org/pub/linux/libs/pam/>

¹⁰<http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam-6.html#ss6.5>

9.4.1. `cyr`: Команда за конфигуриране на конзолата

Подробна информация за параметрите, които могат да се предадат на командата `cyr(1)`, може да получите след изпълнение на

```
cyr --help
```

Ако се изпълни командата с опция `--save`, настройките от командния ред се и запазват във файла `~/.cyr_defaults`. Следващото изпълнение на `cyr` без параметри ще конфигурира конзолата според запазените настройки. Така, след като веднъж има запазени потребителски настройки, добавянето на командата

```
cyr 2> /dev/null
```

във файла `~/.bash_profile` ще конфигурира конзолата при всяко влизане (login) в системата. (`2> /dev/null` подтиска грешките при отваряне на терминал в X, като `xterm` или `gnome-terminal`.)

9.5. XFree86: Графична среда

9.5.1. XKB: Клавиатура

Следните текстове могат да ви помогнат ако искате да разберете повече за разширението XKB на X:

- [The XKB Configuration Guide¹¹](#) (част от XFree86 4.3)
- [How to further enhance XKB configuration¹²](#) (част от XFree86 4.3)
- [An Unreliable Guide to XKB configuration¹³](#) (от [Doug Palmer¹⁴](#))
- [X Keyboard Extension¹⁵](#) (от [Иван Паскал¹⁶](#))

`xserver-xfree86`: Конфигуриране при инсталиране

При инсталирането на пакета `xserver-xfree86` трябва да отговорите на следните въпроси:

Въпрос	Опция	Отговори за българска среда
Please select your keyboard layout	<i>XkbLayout</i>	- bg
Please select your keyboard variant	<i>XkbVariant</i>	- phonetic - bds
Please select your keyboard options	<i>XkbOptions</i>	- grp:shift_toggle - grp:ctrl_shift_toggle - grp:caps_toggle - grp:ctrl_alt_toggle - grp:alt_shift_toggle - grp:menu_toggle

Подробности за смисъла на стойностите на *XkbOptions* се намират в `/etc/X11/xkb/symbols/group`.

¹¹<http://www.xfree86.org/current/XKB-Config.html>

¹²<http://www.xfree86.org/current/XKB-Enhancing.html>

¹³<http://www.charvolant.org/~doug/xkb/html/index.html>

¹⁴<http://www.charvolant.org/~doug/>

¹⁵<http://www.tsu.ru/~pascal/other/xkb/>

¹⁶<http://www.tsu.ru/~pascal/>

Важна корекция

Проверете дали имате този файл, и ако не то непременно трябва да изпълните командата:

```
# touch /usr/lib/X11/locale/microsoft-cp1251/Compose
```

понеже някои програми не могат да тръгнат, ако този файл не съществува.

9.5.2. Шрифтове

Ако искате да разберете повече за шрифтовете в X, следните текстове може да ви помогнат:

- [Fonts in XFree86](#)¹⁷, от документацията на XFree86¹⁸
- [XFree86 Font De-uglification HOWTO](#)¹⁹

Имайте в предвид, че в X кодирането на знаците на нашата кирилица се именува `microsoft-cp1251`, по специално в имената на шрифтовете, и `windows-1251` на всички други места, като поща например. Последното име е и официалното на това кодиране.

В X широко се използват някои предефинирани имена на шрифтове, като `fixed` или `10x20` например. Версиите на тези шрифтове с кодировка `windows-1251` са с префикс `w-`, като `w-fixed` и `w-10x20` например. Префиксът `c-` се използва за шрифтове с Уникод кодиране, което в X се именува `iso10646-1`. Всички такива кратки имена (*alias*-и, псевдоними) могат да се видят с командата `xlsfonts`²⁰:

```
$ xlsfonts | grep ^w-
```

Пълен списък на всички псевдоними се намира във файловете `*.alias` от поддиректориите на `/etc/X11/fonts`. За да видите образите на знаците на някой шрифт, използвайте командата `xfd`²¹ от вида:

```
$ xfd -fn w-10x20
```

За търсене и разглеждане на шрифт използвайте `xfontsel`²². Една комбинация от двете команди, която използвам за Уникод²³ шрифтове (в X ги именуват с кодиране `iso10646-1`), е

```
$ xfd -fn "`xfontsel -print`"
```

Пакети с шрифтове, съдържащи кирилица

(FIXME: Тази част още не е прехвърлена.)

Препоръчвани пакети с шрифтове

Ето списък на най-често използваните пакети с кирилски шрифтове:

- `xfonts-base`
- `xfonts-cronyx-{cp125124, koi8r25}-{{7526, 10027}dpi, misc28}`

¹⁷<http://www.xfree86.org/current/fonts.html>

¹⁸<http://www.xfree86.org/>

¹⁹<http://feenix.burgiss.net/ldp/fdu/>

²⁰<http://www.xfree86.org/current/xlsfonts.1.html>

²¹<http://www.xfree86.org/current/xfd.1.html>

²²<http://www.xfree86.org/current/xfontsel.1.html>

²³<http://www.unicode.org/>

²⁴<http://packages.debian.org/xfonts-cronyx-cp1251-75dpi>

²⁵<http://packages.debian.org/xfonts-cronyx-koi8r-75dpi>

²⁶<http://packages.debian.org/xfonts-cronyx-75dpi>

²⁷<http://packages.debian.org/xfonts-cronyx-100dpi>

²⁸<http://packages.debian.org/xfonts-cronyx-misc>

- `xfonts-bolkhov`²⁹ - {cp1251, koi8r} - {75dpi, misc}
- `scalable-cyrfonts-x11`
- `scalable-terminus`
- `msttcorefonts`

Забележка Ако не знаете точното предназначение на кривите скоби, използвани тук, моля прочетете частта **Brace Expansion**³⁰ от секцията **Basic Shell Features**³¹ в **ръководството на Bash**³². Тук те няма да се обясняват, защото и без друго това е една от ценните възможности на Bash, която си струва да се знае.

TrueType шрифтове

Има два основни начина, с които една програма може да използва шрифтове. Първият е класическият: дават се команди на X сървъра да изобрази определени знаци от определен шрифт. Наричаме този начин *използване на X шрифтовете (X core fonts)*. Вторият начин е самата програма да направи изображение на знаците, които иска да покаже, и да прати тези готови образи на X сървъра чрез **разширението RENDER**³³ на **XFree86**³⁴. За тази цел се използва библиотеката **FreeType**³⁵. Самата библиотека не прави повече от построяване на образи на знаци. Задачата да се прехвърлят тези образи в X сървъра се изпълнява от библиотеката Xft, последната версия 2 на която е част от по-общата система **FontConfig**³⁶ за използване на шрифтове. Този начин наричаме *използване на Xft шрифтовете*. (В бъдеще по-добре е да се наричат *FontConfig шрифтове*, но в Дебиан 3.0 това няма да е адекватно, защото там има само Xft1.) От потребителска гледна точка вторият начин има няколко предимства: способността да се изглаждат знаците на шрифтовете (anti-aliasing, X шрифтовете нямат тази способност), много по-разбираемото именуване на шрифтовете (сравнете Lucida Sans 14 c -b&h-lucida-medium-r-normal-sans-14-100-100-100-p-80-iso10646-1) и способността за фино конфигуриране на Xft. Вторият начин е по-нов и само по-новите програми (например тези на KDE2 и GNOME2) са способни да го използват.

И по двата начина могат да се използват TrueType шрифтове. Като X шрифтове те се изобразяват чрез вариант на FreeType, или чрез друга библиотека за изобразяване на знаци — XFT, но това е само за съвместимост с по-старите програми.

Инсталиране на TrueType шрифтове Понякога се разпространяват TrueType шрифтове за определено кодиране, например windows-1251. Много вероятно е да имате проблеми с тях, защото те ще се възприемат като с кодиране iso8859-1. Използвайте Уникод шрифтове.

1. Вземете отнякъде TrueType шрифтове
2. Инсталирайте пакета `ttmkfdir`
3. Сложете `*.ttf` файловете в някаква временна директория.
4. Изпълнете там командата
`$ ttmkfdir -o име.scale`
където *име* е специфично за тази група шрифтове име.
5. Копирайте `*.ttf` файловете в директорията `/usr/lib/X11/fonts/TrueType`. Създайте я, ако не съществува.
6. Копирайте файла `име.scale` в директорията `/etc/X11/fonts/TrueType`. Създайте я, ако не съществува.

²⁹<http://packages.debian.org/xfonts-bolkhov-75dpi>

³⁰http://www.gnu.org/manual/bash-2.05a/html_chapter/bashref_3.html#SEC27

³¹http://www.gnu.org/manual/bash-2.05a/html_chapter/bashref_3.html

³²<http://www.gnu.org/manual/bash-2.05a/>

³³<http://keithp.com/~keithp/talks/>

³⁴<http://www.xfree86.org/>

³⁵<http://www.FreeType.org/>

³⁶<http://fontconfig.org/>

7. Изпълнете командите:

```
# update-fonts-scale TrueType
# update-fonts-dir TrueType
```

Използване като X шрифтове Ако използвате шрифтовия сървър `xfs`, трябва да добавите директорията `/usr/lib/X11/fonts/TrueType` към параметъра `catalogue` във файла `/etc/X11/fs/config`. В шрифтовия сървър `xfs-xtt` това е файлът `/etc/X11/fs-xtt/config`. За да видите резултат веднага след като инсталирате шрифтовете, изпълнете командата

```
# /etc/init.d/xfs reload
```

или ако използвате `xfs-xtt`:

```
# /etc/init.d/xfs-xtt reload
```

Ако не използвате шрифтов сървър, копирайте цялата секция `Files` на `/etc/X11/XF86Config-4` в края на файла, задължително след реда (ако го има):

```
### END DEBCONF SECTION
```

Добавете `/usr/lib/X11/fonts/TrueType` в началото на списъка с директориите.

Използване като Xft шрифтове Не е нужно да правите каквото и да е — директорията на шрифтовете е част от подразбиращата се конфигурация. Между другото във `FontConfig`, която система не е част от Дебиан 3.0, инсталирането на TrueType шрифтове се свежда до копиране на `*.ttf` файловете в директорията `~/.fonts`.

Възможно е обаче да искате да конфигурирате Xft. Подробности могат да се намерят в [XFree86 Font De-uglification HOWTO](#)³⁷. Обикновено най-желаната конфигурация се свежда до добавяне на следните редове към `/etc/X11/XftConfig`:

```
match
    any size > 8
    any size < 15
edit
    antialias = false;
```

Това забранява изглаждането на шрифтове с големина между 8 и 15 пункта.

9.5.3. `xfs`: Шрифтов сървър

Инсталирането на шрифтов сървър, като например `xfs` или `xfs-xtt`, е **препоръчително**³⁸. Използвайте `xfs`. За да получите по-добри резултати, в края на файла `/etc/X11/XF86Config-4`, задължително след реда (ако го има)

```
### END DEBCONF SECTION
```

вмъкнете следното:

```
Section "Files"
    FontPath      "unix:/7100"      # xfs port
EndSection "Files"
```

³⁷<http://feenix.burgiss.net/ldp/fdu/>

³⁸<https://listman.redhat.com/pipermail/roswell-list/2001-September/001816.html>

9.6. Справяне с някои „упорити“ програми

9.6.1. GNOME

GTK

Подразбиращите се шрифтове за GTK базираните програми (това включва всички програми на GNOME) могат да се променят чрез редактиране на файла `/etc/gtk/gtkrc.bg`.

AbiWord

Следващите инструкции са преработка на [едно писмо](#)³⁹ от [Георги Данчев](#)⁴⁰. Използват се TrueType шрифтове, например тези на Майкрософт, които могат да се инсталират чрез пакета `msttcorefonts`. В частта за шрифтове има повече информация за TrueType шрифтовете.

1. Създайте директория `/usr/share/AbiSuite/fonts/CP1251`. Всички следващи команди предполагат, че сте в тази директория.
2. Създайте препратки към `*.ttf` файловете, които искате да ползвате, например с командата (като последният аргумент е точка):

```
# ln -s /usr/lib/X11/fonts/TrueType/*.ttf .
```

3. Изпълнете командите (като пакетът `ttmkfdir` трябва да е инсталиран):

```
# ttmkfdir | grep microsoft-cp1251$ > fonts.list
# (wc -l fonts.list; cat fonts.list) > fonts.scale
# mkfontdir
```

4. За да проверите дали всичко е наред, изпълнете командата:

```
# ttftool -e print
```

В изхода на командата трябва да присъства кодирането CP1251.

5. Изпълнявате командата:

```
# ttfadmin.sh /usr/share/AbiSuite/fonts/CP1251 CP1251
```

В директорията с CP1251 шрифтове на AbiWord ще се появят файлове с разширения `.t42`, `.afm` и `.u2g`.

6. *(Използвайте това закърпване на положението само ако имате проблеми. В оригиналното писмо се настоява за тази промяна, но при мен всичко върви добре и с `LANG=bg_BG`.)* Когато се стартира AbiWord, трябва в променливата `LANG` да се съдържа стойността `bg_BG.CP1251`. Забележете, че при нормално кирилизирана система тази стойност обикновено е `bg_BG`. По принцип стойността `bg_BG.CP1251` е по-правилна, така че е оправдано променянето на глобалната стойност на `bg_BG.CP1251`. И така, има два варианта:

- *Промяна на глобалната стойност.* Редактирайте файла `/etc/environment` и променете `LANG=bg_BG` на `LANG=bg_BG.CP1251`. Вариант на този подход е да се добави в `~/ .bash_profile` реда

```
export LANG=bg_BG.CP1251
```

- *Създаване на „опакровка“.* Създайте файл `/usr/local/bin/abi` със следното съдържание:

```
#!/bin/sh
LANG=bg_BG.CP1251 exec abiword
```

Друг вариант е в `~/ .bash_profile` да добавите

³⁹<http://linux-bulgaria.org/lug-bg-list/archive/2003/Jan/0260.html>

⁴⁰<http://danchev.fccf.net/>

```
alias abi='LANG=bg_BG.CP1251 abiword'
```

Недостатъкът на всички тези подходи е, че тогава редакторът трябва да се стартира от команден ред или да се направи нещо като Shortcut.

7. Ако изцяло сте поверили X шрифтовете на `xfs` сървър, както е препоръчано по-горе, и ви се появява досадно съобщение за грешка при добавяне на Abiword шрифтовете към шрифтовете на X сървъра, изключете Modify Unix Font Path от частта Preferences Schemes от диалога Preferences, достъпен от менюто Tools.

GDM

Във файла `/etc/init.d/gdm` в началото се добавят следните команди:

```
LANG=bg_BG
export LANG
```

Във файла `/etc/locale.alias` се добавя реда

```
bulgarian          bg_BG.CP1251
```

Следните параметри от файла `/etc/X11/gdm/gdm.conf` трябва да се променят:

```
DefaultLocale=bg_BG
SystemMenu=true
Use24Clock=true
```

9.6.2. KDE

С KDE и въобще приложенията разчитащи на библиотеката Qt (защото има и доста такива които се разработват и извън проекта KDE⁴¹) няма да имате никакви проблеми. Но все пак ето някои бележки, които може да са непълни. Ако имате някакви предложения, казвайте ги в [debian-bg](#)⁴².

KDE изисква следните настройки: (май има и още неща⁴³):

- Kontrol panel — Personalization — Country & language — Charset се установява на `windows-1251`.
- В Kontrol panel — Look & feel — Fonts кодирането на всички шрифтове се прави `windows-1251`.

Вместо `windows-1251`, може да използвате UTF-8, ако сте компилирали и тези локали, и имате инсталирани Unicode шрифтове, т.е. ако се налага може да преконфигурирате изпълнявайки:

```
# dpkg-reconfigure locales
```

Ще се появи меню (от `debconf`) и ще може да посочите за `bg_BG` и `CP1251`, UTF-8 и т.н. Реално това което става е, че се извиква скрипта `/usr/sbin/locale-gen` чийто конфигурационен файл е `/etc/locale.gen`. От така компилираните локали ще се възползват всички програми разбиращи от тях, т.е. не само тези от KDE или разчитащи на Qt.

(Display Manager-ите `kdm` и `wdm` също **изискват**⁴⁴ някои допълнителни действия.)

⁴¹<http://www.kde.org>

⁴²<http://debian.gabrovo.com/mailling.php>

⁴³<http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=343148460>

⁴⁴<http://linux-bulgaria.org/lug-bg-list/archive/2002/Aug/0211.html>

9.6.3. Mozilla

Мозила използва FreeType2 библиотеката, и като следствие, не се съобразява с файла `/etc/X11/XftConfig`. За да използвате тази библиотека, трябва да инсталирате пакета `libfreetype6`. При инсталирането на Мозила се пита дали да се използва библиотеката: отговорете, че искате. В Sarge (бъдещата още неиздадената версия след Woody) не се задава такъв въпрос, но допълнително трябва да инсталирате пакета `mozilla-xft`. Във файла `/etc/mozilla/prefs.js` могат допълнително да се настройва употребата на FreeType2, като трябва там да добавите директорията `/usr/lib/X11/fonts/TrueType`. Допълнително може да зададете чрез настройката `font.antialias.min` какъв да е най-малкият размер, при който знаците да се изглаждат.

9.6.4. Midnight Commander

Стартирате `mc`, от менюто избирате **Options**, след което **Display Bits** и включвате *Full 8 bit input* и *Full 8 bit output*.

9.6.5. teTeX

Българизирането на **teTeX** е описано в главата **Работа с L^AT_EX 2₄**

9.7. *tasksel*: Бързо българизиране на Debian

Истината е, че не е нужно да се задълбочавате в подробности, за да може бързо да си българизирате вашия Дебиан. Достатъчно е да изпълните командата

```
# tasksel
```

която показва кратък списък от групи, съдържащи пакети. (Това е същата програма, която се изпълнява съвсем в края на инсталирането на Дебиан 3.0.) Интересни са групите „desktop environment“ и „Cyrillic environment“. На теория инсталирането на тези групи трябва наведнъж да направи всичко, но действителността, както обикновено, не се оказва толкова розова. По време на конфигурирането различни пакети ще ви задават въпроси. Имената на някои от тези пакети са включени в имена на секции от тази статия. Това е нарочно направено за справка по време на инсталирането. Ако използвате `language-env`, дейността по конфигурирането се упрости.

9.7.1. *language-env*: Автоматично конфигуриране на програми

Може да се инсталира и пакета `language-env` (CD2, част от „Cyrillic environment“ на `tasksel`), който е предназначен да създава точка-файлове, т.е. конфигурационни файлове, за разпространени програми. След инсталирането си този пакет не извършва никакви глобални конфигурации. Всеки потребител трябва сам да стартира командата

```
$ set-language-env
```

която задава въпроси за предпочитанията и създава различни конфигурационни файлове в потребителската директория. За да се обърнат промените, които програмата е извършила, се изпълнява командата

```
$ set-language-env -r
```

Предимствата на този пакет са две. Първо, не е нужно да се конфигурират пакета `console-cyrillic` и ХКВ метода за въвеждане, и второ, всеки потребител може да има различни настройки. Не трябва да се забравя, че пакетът `locales` трябва да е конфигуриран коректно.

9.8. Какво още може да се направи...

9.8.1. Превод на сайта <http://www.debian.org>

- Достъп до официалния превод:


```
cvcs -d :pserver:anonymous@cvcs.debian.org:/cvcs/webwml login
cvcs -d :pserver:anonymous@cvcs.debian.org:/cvcs/webwml checkout webwml/bulgarian
```
- Документи: Всяка подкрепа е добре дошла, би било чудесно ако се включат повече хора а ето и основната информация, която да прочетат, след като се направи **cvcs checkout** за българския и за английския текст:
 - <http://www.debian.org/devel/website/>
 - <http://www.debian.org/devel/website/translating>
 - <http://www.debian.org/devel/website/stats/bg.html>
- Преводачи: Към момента единствения преводач е Румен Кръстев (rkrastev at obs dot bg), с *write access* до cvcs.debian.org⁴⁵. Ако сте забелязали грешки при превода или имате идея как да помогнете при превода най-добре е да влезете във връзка с него.

9.8.2. Превод на официалните документи на <http://www.debian.org/doc>

За съжаление за сега са известни само:

- [Стария и превод на Maint Guide на Николай Христов](#)⁴⁶ - не е включен в официалната документация
- [Стария и превод на APT-HOWTO](#)⁴⁷ - не е включен в официалната документация

Това което би било добре да се направи е да се осъвременят преводите спрямо последните официални версии на тези документи и да се включат в официалната документация.

9.8.3. Превод на официалния *debian-installer*

FIXME: който се е захванал с превода да се обозначи, моля ;-)

9.8.4. Поддръжане на неофициално *apt* хранилище

За тези които постоянно питат и предлагат самостоятелна Българска дистрибуция, без да знаят за какво приказват и какво означава това като начинание. Та може да се поддържа неофициално хранилище с пакети които по една или друга причина не могат да влязат в официалната дистрибуция на Debian, но пък ни интересуват нас като българи и да могат лесно и бързо да се инсталират от българските потребители. Та абсолютно в реда на нещата е да се възползваме от това което е в официалния Debian и да добавим каквото ни липсва, въпреки, че е много трудно да се намери нещо което не е включено в официалния Debian, щото това не е просто най-долямата дистрибуция на GNU/Linux, а е най-голямата дистрибуция на Операционна Система въобще - 12 000 official deb's към момента хич не е скромно. Та решението е и ние българите (като всички останали) да се вместим някак си в тази интернационална и универсална дистрибуция на Операционна Система, щото да се опитваме да догоним възможностите на Debian почвайки от нулата е меко казано несериозно или неразбиране за какво се опитваме да говорим.

Ето някои случайни примери за такива пакети, интересувачи само нас българите:

- *bgtex-v2*, който го няма в официалната дистрибуция на *TeX* и естествено и в *tex* пакетите които са в Debian архива.

⁴⁵<http://cvcs.debian.org>

⁴⁶<http://debian.gabrovo.com/docs/maint-guide/>

⁴⁷<http://danchev.fccf.net/docs/linux/apt-howto-bg/>

- Програмата за Дневниците по ДДС
 - Може да се направи и пакет, който да преконфигурира и промени каквото намерите за добре
- FIXME: Проекта <http://bgdebian.sourceforge.net/> не беше ли създаден с цел unofficial bg repository ?
- Най-доброто решение разбира се е да имаме колкото може повече българи като official Debian maintainers**

Глава 10

Други

Какво трябва да се направи в Debian — <http://www.debian.org/devel/todo/>.

Част IV

Управление на софтуера

Глава 11

Общи положения

Качествен свободен софтуер вече има в неимоверно големи количества и продължава да се бълва такъв. Затвореният софтуер бива достиган в почти всяка ниша и безмилостно конкуриран, което е добре за развитието и на двата вида софтуер, въпреки, че не се харесва много на производителите на втория, особено на тези, които се изживяват като единствени и неповторими, но няма как да им се размине. При нарастването на софтуера в системата, нормално е известни трудности да бъдат срещани от потребителите при неговото управление (включая правилното му компилиране, инсталиране и впоследствие евентуално и чистото му и напълно премахване от потребителската система). Това особено се усеща и натежава, когато се налага да се консумира в големи и изключително големи количества и възможно най-бързо и надеждно. Понякога излишно се хабят усилия и време за съвсем рутинни и елементарни неща, като главоблъскане откъде да си вземем липсващия ни заглавен файл. Това трябва да се налага да се прави само при поискване от страна на потребителя и ако той има необходимото време и желание да го прави, и по подразбиране е добре да бъде автоматизирано с дадени системни инструменти. Съвсем нормално е да видите Debian система с инсталирани например около 7000 packages от настоящите Stable, Testing и Unstable пак и кой знае колко още Unofficials, която е била доведена от потребителя в това си състояние от първоначалната инсталация (initial install), която е била направена от някой стар release (2-3 release назад например). Дотук е доведена без преинсталация начисто след появата на следващия нов release, което си е съвсем в реда на нещата при upgrade до по-горна версия. С две думи, Debian се инсталира първоначално само веднъж и се upgrade докато crash-не медията, върху която е инсталиран. Няма такова понятие като преинсталация на чисто на цялата система за да се upgrade до по-горна версия, това е чиста загуба на време. Затова е много, добре когато се реализира всеобхватно, гъвкаво и надеждно управление на софтуера (binary & source), предоставян в различните версии на дистрибуцията. Нали това е грижа на дистрибуцията, наред с поддръжка на fast & safe security updates, Bug Tracking System, mailing lists, new groups, irc channels.

Нека да дадем и яснота и по някои понятия или термини, повечето от които са въведени от проекта Debian, но се ползват и от всички останали, къде с разбиране къде не:

- **upstream sources** - сорсове на дадена програма или приложение.
- **debian source package** - **upstream sources** към които е добавена директория **debian/** съдържаща кода които контролира билд процеса за получаването на **debian binary packages** (**.deb**'s) за съответните хардуерни архитектури.
- **upstream developer** - разработчик на софтуер, програмист - по принцип. Тези създават **upstream sources**.
- **upstream maintainer** - този който поддържа **upstream sources** на дадена програма към даден момент. Може и да съвпада с горното, т.е. и той да е и **upstream developer**.
- **debian maintainer**, използва се още и **debian developer** или за кратко **DD** - този който създава и поддържа **debian source package** или по-точно директорията **debian/** към дадени **upstream sources** към даден момент, като трябва да е запознат до известна степен и с нейните **upstream sources**, дори и да не се явява **upstream developer** или **upstream**

maintainer. Може да съвпада с горните две, т.е. да е и **upstream developer** и **upstream maintainer**.

- **Sponsor** - като цяло **debian maintainer** не се става лесно. Има установена процедура през която всички кандидати за **debian maintainers** се оценяват и евентуално се одобряват според техните технически умения и възможности (статус <http://nm.debian.org>). Има **Група от опитни debian developers**¹ които се занимават с това процедиране на новите кандидатури. Тези които все още не са станали официални **debian maintainers**, могат да си намерят **Sponsor**² - това е официално регистриран **debian maintainer**, който ще се съгласи на одитира и upload-ва в официалния архив на Debian така направения от вас пакет или пакети. Може да ви бъде от полза следното хранилище <http://mentors.debian.net>. Не бъркайте този вид **Developer Sponsorship**³ с **Partners**⁴ или **Donations**⁵ ;-)

Понеже Debian е платформа използвана много интензивно от програмисти или въобще разработчици на софтуер, в доста случаи един и същи човек е и **upstream developer** и **debian maintainer**. Има и доста софтуер които е разработен специално за проекта Debian, но може да се ползва и от всички останали разбира се.

¹<http://nm.debian.org/whoisam.php>

²<http://www.debian.org/devel/join/newmaint#Sponsor>

³<http://www.debian.org/devel/join/newmaint#Sponsor>

⁴<http://www.debian.org/partners>

⁵<http://www.debian.org/donations>

11.1. Официалният архив

Нека като за начало да хвърлим малко светлина (ама това наистина е много набързо) към неговата структура. Не е важно колко е актуална информацията в тази таблица (а тя не е, и едва ли има смисъл да е), важното е да се види структурата (за да не бъркаме понятията;-), която хич не е сложна и е логична (разгледайте някой официален Debian mirror за справка).

Забележка: Понеже таблицата е стара, имайте предвид, че сегашният stable е Woody, а не Potato. Също така сегашният testing е Sarge, а не Woody.

Origin	Debian	Debian	Debian
Label	Debian	Debian	Debian
Suite	Stable	Testing	Unstable
Codename	Potato	Woody	Sid
Components	main, contrib, non-free	main, contrib, non-free	main, contrib, non-free
Arch	alpha, arm, i386, m68k, powerpc, sparc	alpha, arm, i386, m68k, powerpc, sparc{64}, ia64 hppa	alpha, arm, i386, m68k, powerpc, sparc{64}, ia64, mips{64}..... hurd-i386
Date	Mon, 16 Apr 2001	Tue, 04 Sep 2001	Tue, 04 Sep 2001
Description	Debian 2.2r3 Released 16 Apr 2001 (обновява се много рядко - само за security и point releases, e.g. revisions)	Not Released (release-ва се директно от testing, евентуално след известен период на freeze. Release Critical bugs се свеждат до минимум, защото не се допуска release с такива)	Not Released (не се release-ва директно от Unstable, тук се правят разни по-сериозни package transitions и подобни. Всичко минава първо от тук)
Version	2.2r3	-	-
md5sum

- **Stable, Testing и Unstable** са ясни за какво са... Софтуерът (.deb файловете, както и сорсовете) влиза в <http://incoming.debian.org>, след това в Unstable, после в Testing и оттам — в Stable (има и [project/experimental](http://project.experimental)⁶, който е non-automatic, но това е за по-опасни експерименти). На тези Suites се присвояват кодови имена, като Unstable винаги остава с името Sid, другите се променят. Към момента Stable е Woody, Testing е Sarge, и Unstable винаги е Sid. (Ха да видим дали ще откриете на някой [Debian mirror](http://www.debian.org/mirror/)⁷ къде се намират съответните binary & source packages & lists files.) Официалния release е Stable, като освен кодово име, което е имал до сега, му се присвоява версия (2.1, 2.2, 3.0 и т.н.) и по-късно се правят само т.н. point releases на този release, или това са revisions (r1, r2, r3 и т.н.) оттук идва и пълното име, например Potato 2.2r3 и т.н. Като се издаде нов release, старият заминава в <http://archive.debian.org>, като се поддържа и още известно време (разбирай security updates, и т.н.).
- **Arch** — освен binary builds за съответните архитектури се предоставят и sources, които са архитектурно независими разбира се. Естествено, съответните binary packages (.deb файловете) се получават от тези source packages на [машините на проекта](http://buildd.debian.org)⁸ — <http://buildd.debian.org>, като потребителят също може да направи build от debian source packages. Съществуват и архитектурно зависими или архитектурно специфични packages, съдържащи програми, които са създадени специално за дадена архитектура и които са безпредметни за други: като `cpuid` специално и само за i386 (x86), `aboot` за Alpha и т.н. Каква част от пакетите имат build за всяка архитектура, може да видите на <http://buildd.debian.org/stats/>.
- **hurd-i386** — [The Hurd](http://www.gnu.org/software/hurd/hurd.html)⁹ е набор от сървърски програми, които работят върху микроядрото [GnuMach](http://www.gnu.org/software/hurd/gnumach.html)¹⁰ (поне за сега само върху това микроядро и само на x86, e.g. i386) и които реализират драйвер на файлова система, мрежови протоколи и всякакви други способности, които се срещат в стандартните монолитни Unix ядра. Усилено се говори (тъ-ъ, работи) по

⁶[ftp://ftp.debian.org/debian/project/experimental](http://ftp.debian.org/debian/project/experimental)

⁷<http://www.debian.org/mirror/>

⁸<http://db.debian.org/machines.cgi>

⁹<http://www.gnu.org/software/hurd/hurd.html>

¹⁰<http://www.gnu.org/software/hurd/gnumach.html>

използванете на микроядрото [L4Ka::Pistachio](http://l4ka.org/projects/pistachio/)¹¹, което чувствително ще подобри бързодействието на The Hurd. Още информация по-въпроса можете да намерите на:

- [GNU Project](http://www.gnu.org/)¹²
 - [Linux and GNU](http://www.gnu.org/gnu/linux-and-gnu.html)¹³
 - [GNU Hurd fans](http://hurd.gnufans.org/)¹⁴
 - [И естествено българското участие в лицето на Оги Кулев](http://debian.fmi.uni-sofia.bg/~ogi/hurd/ext3fs/)¹⁵
- Към момента единственото ядро, с което се правят издания (releases) на Debian, е [Linux](http://www.gnu.org/software/hurd/hurd.html)¹⁶, т.е. това е Debian GNU/Linux, с това ядро се поддържат и сървърите на проекта, които хич не са едни от най-разтоварените на света, но се работи и по портиране на дистрибуцията към [The Hurd](http://www.gnu.org/software/hurd/gnumach.html)¹⁷, който за сега стои само в Sid, работещ върху микроядро [GnuMach](http://www.debian.org/ports/netbsd/)¹⁸, при което се получава [Debian GNU/Hurd](http://www.debian.org/ports/freebsd/)¹⁹.
- Следват евентуално и монолитните *BSD ядра, при което се получават [Debian GNU/NetBSD](http://www.debian.org/ports/netbsd/)²⁰, [Debian GNU/FreeBSD](http://www.debian.org/ports/freebsd/)²¹, Debian GNU/OpenBSD (но последното като че ли е [временно изоставено](http://qa.debian.org)²²) за евентуално няколко хардуерни архитектури. Всъщност погледнете и [Debian on the Go](http://bugs.debian.org)²³ (laptops) и [Debian Beowulf](http://packages.qa.debian.org)²⁴ (MPI clusters).

11.1.1. Поддръжка

Както се досещате, това, което идва от официалния архив, идва директно и се поддържа от проекта Debian и може да бъде намерено по официалните огледални хостове. За този софтуер, освен че се поддържат builds за всички архитектури, поддържани от проекта, се възползва и от: [Bug Tracking System](http://bugs.debian.org)²⁵, [Package Tracking System](http://packages.qa.debian.org)²⁶, отчитат се [Release Critical Bugs](http://bugs.debian.org)²⁷ и [Quality Assurance](http://qa.debian.org)²⁸ за [Base](http://base.debian.net)²⁹ и [Standard](http://standard.debian.net)³⁰ и т.н.

11.2. Неофициалните архиви

Неофициалните архиви се поддържат от любители от цял свят, като в подобни начинания могат да участват и официални debian maintainers дори for fun & tests. Структурата на подобни неофициални архиви може да наподобява тази на официалния архив напълно или частично. Т.е. в най-простия случай може да е една директория с насипани вътре debian binary & source packages заедно с list files, че дори и само binary packages (но това би било нарушение на GPL например, ако софтуера е GPL'ed). Ето например <http://www.apt-get.org> е един източник, където се събират (и

¹¹<http://l4ka.org/projects/pistachio/>

¹²<http://www.gnu.org/>

¹³<http://www.gnu.org/gnu/linux-and-gnu.html>

¹⁴<http://hurd.gnufans.org/>

¹⁵<http://debian.fmi.uni-sofia.bg/~ogi/hurd/ext3fs/>

¹⁶<http://www.kernel.org>

¹⁷<http://www.gnu.org/software/hurd/hurd.html>

¹⁸<http://www.gnu.org/software/hurd/gnumach.html>

¹⁹<http://www.debian.org/ports/hurd/>

²⁰<http://www.debian.org/ports/netbsd/>

²¹<http://www.debian.org/ports/freebsd/>

²²<http://lists.debian.org/debian-bsd/2002/debian-bsd-200210/msg00063.html>

²³<http://www.debian.org/misc/laptops/>

²⁴<http://www.debian.org/ports/beowulf/>

²⁵<http://bugs.debian.org>

²⁶<http://packages.qa.debian.org>

²⁷<http://bugs.debian.org/release-critical/>

²⁸<http://qa.debian.org>

²⁹<http://base.debian.net>

³⁰<http://standard.debian.net>

проверяват) подобни неофициални източници. Подобни неофициални архиви могат и да се поддържат от проекти като [KDE](http://www.kde.org)³¹, [Mozilla](http://www.mozilla.org)³² и много други. Имайте предвид, обаче, че на подобни неофициални архиви обикновено се поддържат пакети само за x86 (i386) архитектурата, няма я официалната поддръжка на [Bug Tracking System](http://bugs.debian.org)³³, [Package Tracking System](http://packages.qa.debian.org)³⁴, електронен подпис на официалния debian maintainer в debian source package, дори може да няма и md5sum на файловете и т.н.. Така че вие си решавате кое да ползвате и по колко ;-).

³¹<http://www.kde.org>

³²<http://www.mozilla.org>

³³<http://bugs.debian.org>

³⁴<http://packages.qa.debian.org>

11.3. Контролиране на избора на пакети за инсталиране

Ето един конфигурационен файл `/etc/apt/sources.list` със списък на официални и някои (примерни) неофициални източници:

```
#####
# Official US & non-US, binary & source package entries start here
#####
# BINARY PACKAGES
deb ftp://ftp.bg.debian.org/debian stable main contrib non-free
deb ftp://ftp.bg.debian.org/debian testing main contrib non-free
deb ftp://ftp.bg.debian.org/debian unstable main contrib non-free
deb ftp://ftp.bg.debian.org/debian-non-US stable/non-US main contrib non-free
deb ftp://ftp.bg.debian.org/debian-non-US testing/non-US main contrib non-free
deb ftp://ftp.bg.debian.org/debian-non-US unstable/non-US main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://non-us.debian.org/debian-non-US testing/non-US main contrib non-free
deb http://non-us.debian.org/debian-non-US unstable/non-US main contrib non-free
# Proposed & security updates
deb http://security.debian.org stable/updates main contrib non-free
deb http://security.debian.org testing/updates main contrib non-free
deb ftp://ftp.bg.debian.org/debian proposed-updates main contrib non-free
deb ftp://ftp.bg.debian.org/debian testing-proposed-updates main contrib non-free
# SOURCE PACKAGES
deb-src ftp://ftp.bg.debian.org/debian stable main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian testing main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian unstable main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian-non-US stable/non-US main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian-non-US testing/non-US main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian-non-US unstable/non-US main contrib non-free
deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb-src http://non-us.debian.org/debian-non-US testing/non-US main contrib non-free
deb-src http://non-us.debian.org/debian-non-US unstable/non-US main contrib non-free
# Proposed & security updates
deb-src http://security.debian.org stable/updates main contrib non-free
deb-src http://security.debian.org testing/updates main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian proposed-updates main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian testing-proposed-updates main contrib non-
#####
# Unofficial APT entries start here - binary and/or source packages
#####
# Various unofficial sources for APT can be found here:
# http://www.internatif.org/bortzmeyer/debian/apt-sources/
# http://www.apt-get.org
#####
# Unofficial apt-build local repository - note compiler optimizations
#####
#deb file:/var/cache/apt-build/repository apt-build main
# 1) MY OWN LOCAL REPOSITORY
# =====
#deb file:/usr/local/src/Mplayer-dev/mplayer-netinst/MPlayer-CVS ./
# 2) GNOME 2 semi-officials
# =====
# after apt-get update, go examine list files in /var/lib/apt/lists/
# ex: apt-get install -t experimental gnome2
# ex: apt-get -t experimental install nautilus2 gnome-panel2
# gnome-applets2 gnome-terminal2 gnome-control-center2 fam
# msttcorefonts ...
deb http://ftp.de.debian.org/debian/ ../project/experimental main contrib non-free
deb-src http://ftp.de.debian.org/debian/ ../project/experimental main contrib non-
# apt-get install gnome2
#deb http://people.debian.org/~walters/debian/staging/ ./
```

11.3 Контролиране на избора на пакети за инсталиране

```
# 3) KDE 3.x unofficials
# =====
# SEE http://davidpashley.com/debian-kde/faq.html
#
# for KDE 3.1 unofficial deb's see http://wh9.tu-dresden.de/kde3/
# for stable backport from download.kde.org
#deb http://download.kde.org/stable/3.1.1/Debian stable main
#deb-src http://download.kde.org/stable/3.1.1/Debian stable main
# KOffice, kdeartwork, kdeaddons, kdedu, kdesdk, kdetools, kile,
# kbear, quanta, kcd, kcpuload, kdbg, knetload, konq-speaker, kprof
#deb http://people.debian.org/~bab/kde3 ./
# kdevelop
#deb http://people.debian.org/~njordan kde3.0/
# ksensors, kvim, kxicq2, krusader, yammi, arson
#deb http://ers.linuxforum.hu/kde3deb/ ./
# 4) Joey Hess's kitenet development network http://kitenet.net/programs/debs.cgi
# =====
# very powerful perl version of apt-src program + more ...
# deb http://kitenet.net/programs/code/debian /
# deb-src http://kitenet.net/programs/code/debian /
# 5) OPERA
# =====
deb http://www.opera.com/debian stable opera non-free
# 6) ICKLE
# =====
#deb http://stud3.tuwien.ac.at/~e0027500/debian/ ./
# 7) Pixie Plus
# =====
#deb http://arachni.kiwi.uni-hamburg.de/~harlekin/binary-i386/ ./
# 8) Blackdown java files
# =====
#deb ftp://metalab.unc.edu/pub/linux/devel/lang/java/blackdown.org/debian woody non-free
# 9) Jonas site - lame, pine binaries, etc ... , do in mind apt's preferences !
# =====
# deb http://debian.jones.dk/ unstable misc
# deb http://debian.jones.dk/ testing misc
# 10) snapshot.debian.net - Fumitoshi UKAI <ukai@debian.or.jp>
# =====
# Different from usual mirror site, it provides daily snapshot since
# 2002/06/04. It uses pdumpfs to backup debian & debian-non-US
# daily. This is useful in case you got broken version of package and
# you want to get old working version of package without searching
# around delayed debian mirror sites. Whenever you like, you can
# access debian archive on specific date. For example, you can get
# debian packages of June 20th from
# http://snapshot.debian.net/archive/2002/06/20/debian
# or apt-line
#deb http://snapshot.debian.net/archive/2002/06/20/debian unstable main contrib non-free
# You can also get debian packages on relative date, for instance, last week's
# debian from http://snapshot.debian.net/archive/date/last-week/debian/
# or apt-line
#deb http://snapshot.debian.net/archive/date/last-week/debian unstable main contrib non-free
# In place of 'last-week', you can use any datestr recognized by date(1).
# You can also access all version of debian package in snapshot.debian.net by using the
#deb http://snapshot.debian.net/archive pool packagename1 packagename2 ....
# This is useful if you don't know when specific version of package you want
# was installed in debian but know which version of package.
```

Вече трябва да сте се ориентирали как стоят нещата, и може да го ползвате, допълвайки го или коментирайки излишното. Ето ви и още няколко примерни конфигурационни файлове:

/etc/apt/apt-build.conf:

```
cc = gcc-2.95
```

```
Olevel = -O3
march = -march=i686
mcpu = -mcpu=i686
```

/etc/apt/apt.conf:

```
/* This file is a sample configuration file with a few harmless sample
options.
*/
APT
{
  // Options for apt-get
  Get
  {
    Download-Only "false";
  };
};
// Always show packages to be upgraded (-u)
APT::Get::Show-Upgrade "true";
// Options for the downloading routines
Acquire
{
  Retries "0";
};
// Things that effect the APT dselect method
DSelect
{
  Clean "auto"; // always|auto|prompt|never
};
DPkg
{
  // Probably don't want to use force-downgrade..
  Options {"--force-overwrite";}
}
  APT::Default-Release "testing";
  APT::Cache-Limit "25165824";
// All dpsyco packages should be updated after installation.
DPkg::Post-Invoke {"/usr/sbin/update-dpsyco || true";};
```

/etc/apt/apt-file.conf:

```
# cache directory. All Contents files will be stored in this directory
cache = /var/cache/auto-apt
# verbose. Run apt-file in verbose mode
verbose off
# arch. Processor architecture (defaults to host architecture)
# arch = i386
# sources-list. Where to find the 'sources-list' file
sources-list = /etc/apt/sources.list
# ftp-passive. Switch to passive ftp connection
ftp-passive on
# case-sensitive. Find files in case sensitive mode
case-sensitive on
# recursive. Search file in a recursive mode
recursive on
```

/etc/apt/preferences (I):

```
Package: *
Pin: release o=Jones
Pin-Priority: 99
```

/etc/apt/preferences (II):

```
Package: *
Pin: release a=stable
Pin-Priority: 200
Package: *
Pin: release a=testing
Pin-Priority: 300
Package: *
```

11.3 Контролиране на избора на пакети за инсталиране

```
Pin: release a=unstable
Pin-Priority: 400

  /etc/apt/preferences (III):
# OLD VALUES BEGIN
# Package: *
# Pin: release o=Jones
# Pin-Priority: 99
# OLD VALUES END
# Package: *
# Pin: release v=3.0*
# Pin-Priority: 1001
# Using APT with both Debian and non-Debian sources
# -----
#
# APT's Default-Release setting (aka "apt-get --target-release") is an
# extremely useful feature, but it has problems if you're using non-Debian
# entries in your /etc/apt/sources.list file. This file improves the
# situation a bit.
#
# Copy this file to /etc/apt/preferences and edit the following Pin:
# line, replacing "testing" with "stable" if that's your preferences.
# The rest of the file contains an explanation, you don't have to
# worry about anything other than this line if you don't care about
# the details.
#
# -----
Pin: release a=testing
# -----
#
# The above Pin: is your default release. The way it's set is the
# equivalent of the apt.conf APT::Default-Release setting. It's
# convenient to have it here instead so that all the pinning settings
# are in one place.
#
Package: *
Pin-Priority: 900
# Pin unstable at a lower than default priority. Here's an example to
# show why this is necessary. Consider a package which is available
# from 3 releases like this:
#
#   rel.      ver.      without pin      with pin
#   ----      -
#   testing   1.0       900              900
#   local     1.1       500              500
#   unstable  1.2       500              200
#
# Without this pin version 1.1 installed from local would be immediately
# upgraded to the 1.2 version from unstable, since they're both priority
# 500.
#
# The priority of this pin has to be > 100 (the priority of currently
# installed packages) else a package installed from unstable wouldn't
# track new versions from unstable.
#
# This isn't a complete solution. Say your apt.sources included
# Debian's testing and unstable, plus 3 external sources A, B, and
# C. It'd be nice to be able to install a package from B and have
# it always come from B (or from the default release, if a newer
# version gets there), but I don't see a way to do that without
# listing the other releases here explicitly. Since A, B, and C all
# have the same priority (500, the default priority) a package from
# B can be replaced by a newer one from C. If this is a problem for
# you, the best solution I can currently offer is to add a new pin
# here for each of your external sources. Even then I don't see a
# way to do per-release mutual exclusion, so you'll still have to
# order them. If they don't provide a Release file you should be
```

```
# able to use a specifier like "Pin: origin www.somesource.com".
Package: *
Pin: release o=Debian
Pin-Priority: 200
```

ФИХМЕ: да се обясни за apt-cdrom, и въобще за /usr/lib/apt/methods/

11.3.1. Избор на release, от който да се вземат пакети

Нека имаме един такъв разширен /etc/apt/sources.list, както е показан по-горе, с редове за official stable, testing, unstable, experimental и дужина unofficialстава е само пример за демонстрация, не е задължително винаги на подхождате така глобално). Освен това в /etc/apt/apt.conf можем да укажем например

```
APT::Default-Release "testing"
```

така че apt по подразбиране да точи от testing и само с изрично указана опция -t, --target-release, --default-release, подавана на apt, да се предприема теглене от stable, unstable и др. Ако няма указано нищо за APT: :Default-Release, не е подадена опция -t, и освен това няма промяна от потребителя в приоритета на пакетите чрез Package:, Pin: или Pin-Priority: в /etc/apt/preferences, което е с по-голяма тежест от APT: :Default-Release, apt ще предпочете най-голямата версия на пакета. Ето какво бихме получили:

Само този *пакет* да се тегли от unstable и нищо друго, ако има неудоволетворени зависимости apt ще каже:

```
# apt-get install пакет/unstable
```

Apt има разрешение освен за *пакет* от unstable да вземе и неговите зависимости, ако има такива, пак от там:

```
# apt-get install -t unstable пакет
```

Опцията -s е за симулация, т.е. ползвайте я, за да проверите какво би се получило, като нищо няма да бъде изтеглено и инсталирано, само за проверка.

```
# apt-get install пакет1/stable пакет2/testing пакет3/unstable -s
```

Дори може да се конкретизира и до версия на пакет и т.н.

```
# apt-get install пакет=версия
```

Имайте предвид, че по този начин, а и изброените по-горе, може и да downgrade даден инсталиран вече във вашата система пакет, като при ситуация на downgrade apt предупреждава изрично, че минавате към по-ниска версия на пакета.

```
# apt-get install пакет/stable
```

11.3.2. Възстановяване на стари версии на пакети

Състоянието на unstable много бързо се променя, Testing също, но по-бавно. Stable само със security updates, така че за кратко време влизат доста нови packages, като някои стари версии може да изпаднат от официалните архиви и т.н. Ако търсите някоя по-стара версия на даден пакет и я няма в Unstable или Testing към момента, а така също и във вашия локален кеш /var/cache/apt/archives/, но е била там преди известно време, то може да добавяте и редове във файла sources.list(5) към <http://snapshot.debian.net> за търсене на такива по-стари и изпаднали към момента версии на отделните пакети. На сайта си пише как се ползва.

Има предвидена и още една възможност в подобни ситуации. Ако във вашата система имате инсталирана версия на пакет, който е изпаднал към момента от официалните stable/testing/unstable, освен това сте го премахнали и от локалния си кеш на apt и отгоре на това не ви се търси точно тази версия в архивите на <http://snapshot.debian.net>, то може да използвате Perl скрипта dpkg-repack, който идва с едноименния пакет dpkg-repack. Преди да upgrade-вате така дефицитните версии на интересуващите ви пакети, изпълнете:

```
# dpkg-repack пакет
```

Инсталираният в системата пакет ще бъде събран пак като инсталационен `.deb` пакет и сега вече спокойно може да `upgrade`-вате към нови версии на дадените пакети, при което, ако те нещо не ви харесат, ги премахвате или форсирате `downgrade` за тях.

11.3.3. Приоритети на пакетите

`Apt` отчита вътрешно списък с приоритети на пакетите (`candidate version policy`, или начина, по които те да бъдат избирани от `apt`), описание на които ще намерите в `apt_preferences(5)`. Там е обяснено и какво са `non-automatic priorities` и как може да се променят подразбиращите се такива, а оттам и селективното поведение на `apt`. Когато правите такива промени, винаги използвайте `-s` опцията на `apt`, за да се уверите, че ще постигнете точно това, което желаете, избягвайки нежелателни изненади. Изпълнявайки:

```
# apt-cache policy пакет
```

ще получите информация за `Installed` и `Candidate` версиите и `Version Table`, в която за всяка достъпна версия на пакета се изписват приоритета и мястото, от където евентуално ще изтеглите този пакет.

FIXME: Страницата е вече променена. Един пример <http://people.debian.org/~walters/gnome2.html> за това как може да се промени приоритета на `packages` от `experimental` (трябва да имате `experimental entries` в `/etc/apt/sources.list`, и да сте изпълнили `apt-get update`), за да бъдат предпочетени те от `APT` пред тези от `unstable (Sid)`. В случая е даден пример с `GNOME 2 packages` (виж отговора на предпоследния въпрос). Забележете и интересно предложение за `debfooster -u`, което се дава. Чрез промяна на приоритета на пакетите може да се постигне и т.н. им „заковане“. **FIXME:** Реферира се `apt-howto-bg`.

11.3.4. Downgrade

`Downgrade`-ването на цялата система, да речем от настоящия `Testing` към настоящия `Stable`, може да се разглежда като специален случай на цялостен `upgrade` от `Stable` към `Testing`, който вие изпълнявате чрез

```
# apt-get dist-upgrade
```

Това може да стане чрез `pinning feature` на `apt-0.5.4`. Вижте [APT HOWTO³⁵](#) и `apt_preferences(5)` за повече подробности. На практика `debian packaging tools` са толкова развити, че спокойно може да се програмира върху тях, ползвайки ги като `stable API`. Например статията [How I Downgraded Testing to Stable³⁶](#) показва как може да постигнете горното по един такъв начин със създадени от вас приложения. Разбира се, трябва да знаете какво правите, имайки поне базови познания и опит с `debian packaging system`. Например, че ако „слизате“ от `Testing` към `Stable`, трябва да имате предвид какво ще правите с пакетите, които ги има в `Testing` и ги имате инсталирани на вашата система и които към момента ги няма в `Stable`. Известни познания за `shell` програмиране въобще не пречат ;-). Шелът ви дава достатъчно възможности, но с употребата на `Perl` или `Python` скриптиране за подобни ваши цели можете да направите нещата наистина сериозни. Всъщност добър пример за административни `Perl` скриптове са пакетите `debconf` и `debhelper`, чиито сорсове можете да разгледате. По подобен начин ако решите, че `apt-src`, `apt-build`, `pbuilder` и т.н. не ви предлагат достатъчно възможности за получаването на `binary packages` от съответните `source packages`, може да погледнете в кодовете им и да запрограмирате нещо аналогично и по-специфично за вашите цели, като отчитате, че ще се нуждаете само от `source packages` на тези `binary packages`, които вие имате инсталирани, плюс необходимото за техния успешен `build`.

³⁵<http://www.debian.org/doc/manuals/apt-howto/ch-apt-get.en.html#s-pin>

³⁶<http://www.debianplanet.org/node.php?id=880>

11.3.5. Виртуални и мета-пакети

Една от силните страни на пакетната система на Debian това е наличието на виртуални и мета-пакети. Те спестяват много време и усилия на потребителите.

Виртуални пакети - Има пакети, които предлагат същата или почти същата функционалност. В такъв случай е добра идея да се създаде виртуален пакет, който да обедини тези функционалности. Тези пакети се наричат "виртуални пакети". Виртуалните пакети съществуват само логически - т.е. те не са истински пакети (от там идва и името им - виртуални). Тази им функционалност е доста полезна за хората, които не са запознати с конкретни имена на програми, но знаят каква функционалност търсят. Например ако се търси ftp сървър не е нужно потребителят да знае конкретни имена на продукти и да търси не-ефективно (в отговорите на apt-cache ще има доста програми, които всъщност не са ftp сървъри, а просто имат някаква връзка), нужно е просто да се изпълни:

```
# apt-cache search ftp-server
```

Отговорът вече е повече от задоволителен:

```
bsd-ftpd - Port of the OpenBSD FTP server
ftpd - FTP server
ftpd-ssl - FTP server with SSL encryption support.
heimdal-servers - Servers for Heimdal Kerberos
kerberos4kth-servers - Servers for Kerberos4 From KTH
krb5-ftpd - Secure FTP server supporting MIT Kerberos
lukemftpd - The enhanced ftp daemon from NetBSD.
vsftpd - The Very Secure FTP Daemon
wu-ftpd - powerful and widely used FTP server
:::..
```

Информация за наличните виртуални пакети в Debian можете да намерите на адрес: doc/package-developer/virtual-package-names-list.text в най-близкия локален mirror на Debian.org³⁷ например: bg.debian.org³⁸

Ако имате инсталиран пакета `debian-policy` на вашата система можете да прочетете този списък и от: `/usr/share/doc/debian-policy/virtual-package-names-list.txt.gz`

Мета-пакети - Мета-пакетите са още един много полезен инструмент за начинаещите и за напредналите в Debian. Както добре се знае един от основните проблеми при запознаването с Linux, и при боравенето със софтуера впоследствие е как потребителя да е сигурен, че е инсталирал всички пакети, необходими му за работа или използване на всичките възможности на дадена програма. Мета-пакетите правят точно това. Най-добър пример за мета-пакети са тези на `kde`, `gnome` и `x-window-system`. С инсталирането на `x-window-system` ще се инсталира всичко необходимо за да стартирате графична среда във вашия Debian. Съответно инсталирането на мета-пакета `kde` ще ви снабди с KDE³⁹ и пълното му обкръжение от софтуер: `kdegraphics`, `koffice`, и т.н За да разберете с какви мета-пакети можете да боравите във вашата система изпълнете:

```
#apt-cache search metapackage
```

Системата ще върне списък със всички мета-пакети, които можете да използвате:

```
arts - Analog Realtime Synthesizer (aRts) metapackage
debian-reference - A metapackage to install all translations of Debian Reference
education-astronomy - DebianEdu astronomy related applications
education-chemistry - DebianEdu chemistry related applications
education-desktop-gnome - DebianEdu GNOME desktop applications
education-desktop-kde - DebianEdu KDE desktop applications
education-desktop-other - DebianEdu desktop applications (non-GNOME, non-KDE)
education-electronics - DebianEdu electronics related applications
education-geography - DebianEdu applications for geography
education-graphics - DebianEdu graphics related applications
:::..
```

³⁷<http://www.debian.org/>

³⁸<http://www.bg.debian.org/doc/package-developer/virtual-package-names-list.text>

³⁹<http://www.kde.org/>

11.3.6. Алтернативи на *dpkg* и *apt*

За пояснение, самата инсталация на packages се прави от **dpkg(8)** (аналогично на *rpm*), а *apt* се използва за внасяне на допълнителна логика върху всичко това и правене на по-специални магии, които не са работа на **dpkg(8)** да знае и може (той си има достатъчно друга работа), на него **apt-get(8)** му подава готов набор от пакети за обработка. Реално може да ползвате и само **dpkg(8)** (точно както и програмата *rpm*) и без надстройка като **apt-get(8)** (или **dselect(8)**), но ако има някакви неудоволетворени зависимости и/или конфликти, **dpkg(8)** само ще изреве и ще спре работа и ще се оплаква докато не му ги доставите и подадете ръчно в съответния ред, вместо да даде предложения за решения и т.н., което е от компетенцията на **apt-get(8)** (и **dselect(8)**). Такива надстройки има и за *rpm*, разбира се. Има и графични надстройки и над *apt* като *aptitude*, *synaptic*, *gsynaptic*, *stornpkg*, *deity*, *gnome-apt*, *kpackage*. Последните две май са лош пример за такива ;-)

Нека не се бъркат програмите пакетни менажери (като *dpkg* и *rpm*) със съответните пакетни бинарни формати (.deb и .rpm), с които те работят. Тези бинарни файлове (да речем, .deb) са просто един архив, който се разпознава и от програми като **ar(1)**, **tar(1)**, **file(1)**. Те се получават от съответните сорсове (source packages). Те са си .tgz или .tar.gz архив, като са конфигурирани по подходящ начин, за да се компилират и инсталират коректно на съответната система, т.е. спазва се някакъв стил и политика. Реално пакетният менажер *rpm* го има и в Debian, но не бива да се ползва директно за инсталиране на .rpm пакети, най-малкото понеже тези пакети не са подходящо конфигурирани и едва ли спазват стила и практиката на Debian при изготвянето на пакети, т.е. те не го спазват и това не им е работа, разбира се. Той е сложен за създаване на такива при добро желание от страна на потребителя. Има и един пакет *alien*, който е предназначен уж за подобно конвертиране на бинарните пакетни файлове, но трябва да се убедите и разгледате какво и как конвертира, защото не винаги го прави коректно. Има maintainer scripts, които едва ли се генерират при едно такова конвертиране, такива scripts просто се създават за debian source packages и са си специфични за Debian. Такива .rpm пакети са конфигурирани за Red Hat, Mandrake, SuSE и т.н., дори за съответните версии на тези дистрибуции, като хич не е добра идея .rpm пакет, конфигуриран за Red Hat, да се инсталира, особено форсирано, на нещо различно от Red Hat като Mandrake, SuSE и т.н. Не си чупете дистрибуциите по този смешен начин, гледайки на пакетните файлове и пакетните менажери като на ябълки и круши ... ;-). Това хич не е GNU/Linux way... Подобно поведение от страна на потребителите е лошо наследство от работата на сяло с предишна операционна система (затворена), която лесно се обозава и плаче за преинсталация на определен период от време. Това са смешни истории и мисля, че са безкрайно ясни.

FIXME: Да се обясни повече за *apt-listchanges*, *apt-listbugs*, *apt-show-source*, *apt-show-versions*.

Документация

Като нов потребител за начало ще е полезно да прочетете:

- [Install Manual за вашата хардуерна архитектура](#)⁴⁰ (пакет *install-doc*)
- [APT-HOWTO](#)⁴¹ или [стария превод на български](#)⁴² (пакет *apt-howto-en*)
- [quick-reference](#)⁴³, който е силно ориентирано към потребителя (пакет *debian-reference-en*)
- [apt-dpkg-ref](#)⁴⁴ — бърз справочник за *apt* и *dpkg* (пакет *apt-dpkg-ref*)
- [Wiki pages за APT](#)⁴⁵
- [Linux HOWTO's, mini-HOWTO's, FAQ's](#) (пакети *doc-linux-text* и *doc-linux-html*)

⁴⁰<http://www.debian.org/releases/stable/installmanual>

⁴¹<http://www.debian.org/doc/user-manuals#apt-howto>

⁴²<http://danchev.fccf.net/files/docs/linux/apt-howto-bg/>

⁴³<http://www.debian.org/doc/user-manuals#quick-reference>

⁴⁴<http://people.debian.org/~mrd/deb-ref/apt-dpkg-ref.html>

⁴⁵<http://www.spack.org/index.cgi/AptHelp>

11 Общи положения

- Освен това може да е удобно да публикувате документация, инсталирана във вашата система, във вашето уеб пространство за по-лесно и бързо браузване. Ето пакети, които правят това:
 - `dwww` показва документацията на адрес `http://localhost/dwww`
 - `dhelp` показва документацията на адрес `http://localhost/doc/HTML`
 - `doc-central` показва документацията на адрес `http://localhost/dc`
- Можете да разгледате как би изглеждала цялата документация на <http://www.fifi.org/documentation/>
- още препратки към [FAQ's](#)⁴⁶

⁴⁶<http://www.linux.mine.nu/debian-faq/>

Глава 12

Бързи инструкции за работа с *apt* и *dpkg*

Превод на [APT and Dpkg Quick Reference Sheet](http://people.debian.org/~mrd/deb-ref/apt-dpkg-ref.html)¹. Пакетът `apt-dpkg-ref` съдържа този документ. След инсталацията му може да намерите различните формати на документа в директорията `/usr/share/doc/apt-dpkg-ref/`.

12.1. *apt*

```
# apt-get install пакет
```

Изтегля *пакет* и всички негови зависимости, и ги инсталира или ъпгрейдва. Това също ще изкара пакета от състояние *hold*, ако преди това е било в него. Вижте по-долу за повече информация за *hold*.

```
# apt-get remove [--purge] пакет
```

Премахва *пакет* и всички пакети, които зависят от него. `-purge` определя пакетите да бъде напълно изтрити (*purged*), вижте `dpkg -P` за повече информация.

```
# apt-get update
```

Обновява списъците на пакети от мирорите. Трябва да бъде стартирана поне веднъж на ден, ако инсталирате нещо същия ден, а също и всеки път, когато `/etc/apt/sources.list` е променен.

```
# apt-get upgrade [-u]
```

Ъпгрейдва всички инсталирани пакети до най-новите им налични версии. Няма да инсталира нови или да премахва стари пакети. Ако някой пакет промени зависимостите си и изисква инсталация на нов пакет, тогава няма да бъде ъпгрейднат. Вместо това ще бъде третиран все едно е в състояние *hold*. `apt-get upgrade` няма да ъпгрейдне пакети, поставени в състояние *hold* (това е всъщност смисъла на *hold*). Вижте по-долу за ръчното поставяне на пакети в състояние *hold*. Препоръчваме ви и опцията `-u`, защото тогава можете да видите кои пакети ще бъдат ъпгрейднати.

¹<http://people.debian.org/~mrd/deb-ref/apt-dpkg-ref.html>

12 Бързи инструкции за работа с *apt* и *dpkg*

```
# apt-get dist-upgrade [-u]
```

Същото като `apt-get upgrade`, само че *dist-upgrade* ще инсталира или премахва пакети, за да удовлетвори зависимостите на новите версии на пакетите.

```
# apt-cache search шаблон
```

Търси *шаблон* в имената и описанията на пакетите. Могат да бъдат зададени няколко шаблона, като тогава трябва всеки от шаблоните да участва в името или описанието на всеки изведен пакет.

```
# apt-cache show пакет
```

Показва пълното описание на *пакет*.

```
# apt-cache showpkg пакет
```

Показва различни детайли за *пакет*, както и зависимостите му с други пакети.

dselect(8), [console-apt](#), [aptitude](#), [gnome-apt](#), [synaptic](#), [stormpkg](#)

Графични интерфейси за *APT*. *dselect* е най-мощният от тях, но и най-старият и най-труден за употреба. Най-добрият е [aptitude](#).

12.2. *dpkg*

```
# dpkg -i пакет.deb
```

Инсталира дебиански пакет. Например такъв, който ръчно сте изтеглили.

```
# dpkg -c пакет.deb
```

Показва съдържанието (списъкът на файловете, които ще се инсталират) на `пакет.deb` (`.deb` файл).

```
# dpkg -I пакет.deb
```

Показва разнообразна метаинформация, съдържаща се в `пакет.deb`.

```
# dpkg -r пакет
```

Премахва *пакет*. Не може да премахне пакетите, които зависят от *пакет*.

```
# dpkg -P пакет
```

Напълно изтрива (*purge*) *пакет*. Разликата между `-r` (`--remove`) и `-P` (`--purge`) е, че докато `--remove` изтрива файлове с данни и изпълними файлове, `--purge` допълнително изтрива всички конфигурационни файлове.

```
# dpkg -L пакет
```

Показва списък на всички файлове, инсталирани от *пакет*. Вижте също `dpkg -c` за начин, по който тази информация може да се извлече от `.deb` файл.

```
# dpkg -s пакет
```

Показва информация за инсталиран *пакет*. Вижте също `apt-cache show` за показване на информация за пакет от дебианския архив (местата, описани в `/etc/apt/sources.list`) и `dpkg -I` за показване на тази информация, извличайки я от `.deb` файл.

```
# dpkg-reconfigure <package>
```

Наново конфигурира инсталиран пакет, ако той използва `debconf` (`debconf` доставя консистентния конфигуриращ интерфейс по време на инсталация на пакет). Можете и да конфигурирате наново самия `debconf`, ако искате да смените потребителския интерфейс или да промените доколко детайлно да бъдете разпитвани от конфигурацията (`question priority`). Например, за да преконфигурирате `debconf` да използва диалоговия потребителски интерфейс (`dialog`), просто изпълнете:

```
# dpkg-reconfigure --frontend=dialog debconf
```

```
# echo 'пакет hold' | dpkg --set-selections
```

Поставя *пакет* в състояние *hold*.

```
# dpkg --get-selections пакет
```

Извежда какво е състоянието (`install`, `hold` и др.) на *пакет*.

```
# dpkg -S файл
```

Търси *файл* в базата данни с пакети, извеждайки в кои пакети се намира този файл.

12.3. Компилиране на Debian *binary packages* от *source packages*

```
# apt-get source [-b] пакет
```

Изтегля сorsa на *пакет*. Трябва да имате необходимите `deb-src` редове в `/etc/apt/sources.list`, за да работи тази команди. Ако добавите опцията `-b` и изпълните командата като `root`, пакетът ще бъде компилиран автоматично (ако това е възможно).

```
# apt-get build-dep пакет
```

Изтегля и инсталира пакетите, необходими за компилирането на *пакет* от сorsa. Тази способност е налична само в `apt` версия 0.5 или по-нова. `Woody` и всички следващи издания на `Debian` притежават тази способност. Често командата се използва в комбинация с `apt-get source -b`. Например (като `root`):

```
apt-get build-dep пакет
apt-get source -b пакет
```

Ще изтегли сorsa на пакета, ще инсталира всички зависимости, нужни за компилиране, и ще се опита да компилира сorsa.

```
# dpkg-source -x пакет.dsc
```

Ако сте изтеглили ръчно сорс-пакет за програма, която включва няколко файла, като `.orig.tar.gz` (`.tar.gz`, ако програмата поддържа Debian), `.dsc` и `.diff.gz` (кръпки, нужни ако програмата не поддържа Debian), то тази команда разпакетира сорса, използвайки `.dsc` файла.

```
# dpkg-buildpackage
```

Компилира дебиански пакет. Трябва да сте в главната директория на сорса. Примерна употреба:

```
dpkg-buildpackage -rfakeroot -uc -b
```

Където `-rfakeroot` инструктира командата да използва `fakeroot`, която симулира `root` права, `-uc` означава „Не подписвай changelog-a“, а `-b` означава „компилирай само binary package“.

```
# debuild
```

Удобна обвивка на `dpkg-buildpackage`, която ще поеме грижата по използването на `fakeroot`, както и стартирането на `lintian` и `gpg`.

12.4. Решаване на `dependencies` проблеми - *автоматично*

```
# dpkg --configure --pending
```

Ако `dpkg` прекъсне изпълнението си поради грешка докато се изпълнява някоя от командите

```
# apt-get install пакет
# apt-get upgrade
# apt-get dist-upgrade
```

опитайте тази команда:

```
# dpkg --configure --pending
```

за да конфигурирате пакетите, които все пак са разпакетирани, но не и конфигурирани. След това опитайте

```
# apt-get install -f
# apt-get upgrade -f
# apt-get dist-upgrade -f
```

И след това първата команда (без `-f`) отново. Продължавайте, колкото е необходимо. Обикновено голяма част от конфликтите могат да се решат по този начин. Ако се споменават проблемни пакети, можете да пробвате и да ги премахнете.

```
# apt-get install -f
# apt-get upgrade -f
# apt-get dist-upgrade -f
```

Опитват се да разрешат зависимостите докато се изпълнява операцията. Забележете, че `apt-get install -f` не иска други аргументи.

12.5. Справяне с проблеми - ръчно

12.5.1. Справяне с някои конфигурационни проблеми чрез maintainer's scripts

Това е пример с цел демонстрация и не е задължително посоченият пакет да има проблеми;-). Има ситуации, в които *автоматичното* конфигуриране и преконфигуриране на пакети няма да е достатъчно, за това ще се намесим и *самостоятелно* или *ръчно*.

Нека предположим, че ползвате пакети от Unstable (Sid), experimental или някое неофициално място. Освен това сте попаднали на пакети с недостатъчно изтествани и усъвършенствани maintainer scripts, поради които **apt-get(8)** се оказва, че изчаква **dpkg(8)**, който пък от своя страна изчаква някой такъв `post*` и/или `pre*` скрипт, идващ с дадения пакет. Може да се окаже, че скриптът е некоректен и въобще да чака във висящо състояние до безкрай, блокирайки **dpkg(8)**, а оттам и **apt-get(8)**. Те не са увиснали, а изчакват, например, даден лош `postinst` скрипт да завърши успешно своето изпълнение. Може да се окаже, разбира се, че това наистина не е по вина на самите скриптове, а на някоя програма, която те пък извикват или се опитват да стартират.

Предназначението на тези скриптове (както между другото е видно и от техните имена), намиращи се в `/var/lib/dpkg/info/` и извиквани от **dpkg(8)**, е следното:

- пакет.`preinst`: изпълнява се **ПРЕДИ ИНСТАЛАЦИЯТА** на пакета
- пакет.`postinst`: изпълнява се **СЛЕД ИНСТАЛАЦИЯТА** на пакета
- пакет.`prerm`: изпълнява се **ПРЕДИ ПРЕМАХВАНЕТО** на пакета
- пакет.`postrm`: изпълнява се **СЛЕД ПРЕМАХВАНЕТО** на пакета

Например, нека кажем, че такава ситуация се получава при:

```
# apt-get install scrollkeeper -t unstable
```

Чакате доста дълго време, наблюдавате процесите **apt-get(8)**, **dpkg(8)**, `scrollkeeper.postinst c ps auxfw` и `top`, и като че ли изглежда не се върши никаква полезна работа. След като наистина се убедите, че това е така, ще се опитаме да решиме нещата в ход. Хвърляте едно око на самия `postinst`-скрипт, който за пакета `scrollkeeper` е в `/var/lib/dpkg/info/scrollkeeper.postinst`. В случая виждаме, че този скрипт пък извиква друг такъв, `scrollkeeper-rebuilddb -q`, разглеждаме набързо неговата справочна страница **scrollkeeper-rebuilddb(8)** и се убеждаваме, че временно е напълно безопасно и да не бъде изпълнен, така че можем да го коментираме в `scrollkeeper.postinst`, за да мине инсталацията пред **dpkg(8)** и **apt-get(8)** успешно, пък после ще се опитаме да открием проблемите на самия **scrollkeeper-rebuilddb(8)**, стартирайки го например с опция `-v` (`verbose`). Коментираме и запазваме промените. Дори можем да прибавим в `postinst`-скрипта и едно:

```
echo "'this is a temporary fix to unblock dpkg & apt'"
```

Следва направо `kill` на `apt-get`, той от своя страна ще убие процесите `dpkg` и `scrollkeeper.postinst`. Забележете, че пред **dpkg(8)** все още не е регистрирана успешна инсталация на този пакет и при изпълнението на `apt-get install` или `dpkg -i` ще се препоръча да се изпълни първо:

```
# dpkg --configure -a
```

Ако го изпълним без да сме редактирали `postinst`-скрипта, ще получим пак същия неуспешен резултат, за това го правим след промени в него. Изпълняваме го и вече няма кое да спре завършването на скрипта `scrollkeeper.postinst`, оттам **dpkg(8)** и **apt-get(8)** завършват успешно работа и са разблокирани за по-нататъчни процедури. Започваме да търсим проблема на самия скрипт **scrollkeeper-rebuilddb(8)**, разглеждайки кода му и стартирайки го с разни аргументи, като може дори да се наложи да направите промени и в самия него, след което пак да откоментираме достъпа до него от `scrollkeeper.postinst` и тестваме как ще работи той, например с:

```
# dpkg-reconfigure skrollkeeper
```

И така, докато постигнем някакъв по-задоволителен резултат. Това не сте длъжни да го правите, но за спорта си заслужава. След това, ако имате желание, можете да дръпнете *debian source package* и да внесете съответните подобрения, да ги предложите на *maintainer*-а на пакета, или просто да регистрирате бърг срещу този пакет от дадено ниво (вкл. и *wishlist*) на <http://bugs.debian.org>, ако вече това не е направено, разбира се.

В случая е даден прост пример, но можете да се сблъскате с истински предизвикателства и след като разблокирате (освободите) **dpkg(8)** и **apt-get(8)**, от ход да потърсите и цялостно решение. Такива *preinst*-, *postinst*-, *prerm*-, *postrm*- скриптове, разбира се, ще намерите за всеки *debian binary package*, също така и в *debian source package*, от който е получен, както и в */var/lib/dpkg/info/пакет.скрипт*.

Аналогично, вместо **apt-get(8)** и **dpkg(8)** да изчакват скриптовете, които не са си свършили работата по някаква причина, може направо да връщат грешка, при което да получите нещо от вида на:

```
apt-get remove xscreensaver ..
dpkg: error processing xscreensaver (--remove):
 subprocess post-removal script returned error exit status 127
Errors were encountered while processing:
 xscreensaver
```

Процедурата е напълно аналогична на гореописаната, само че сега скриптовете направо приключват работа и уведомяват **dpkg(8)**, че излизат с даден статус. В този пример очевидно трябва да се разправим със скрипта */var/lib/dpkg/info/xscreensaver.postrm*, така че той да приключва работата си успешно. След което регистрираме това, изпълнявайки:

```
# dpkg-reconfigure xscreensaver
```

Глава 13

Конфигуриране на пакети - проблеми и решения

В Debian са предвидени добре обмислени и стандартизирани процедури за почти всякакви ситуации. Освен наличието на такива за правилното инсталиране на софтуера, съществуват и такива за неговото преконфигуриране в най-различни аспекти. Имплементацията наистина е много сериозна, но за съжаление не много потребители обръщат внимание на това, а това е прекрасен източник за придобиване на чужди знания и опит (особено за хакерите на Perl, а вече и Python). Ще е много добре, ако сте запознати със стандартните GNU development tools, като [autoconf](#), [automake](#), [make](#), [gcc](#), познания по shell и особено по Perl скритиране също ще са от голяма полза. Всичко това, разбира се, е част от скромния ни опит да погледнем по надълбоко в една Debian система.

13.1. Официалните документи

Официалните документи, които се съблюдават са:

- [Debian New Maintainers' Guide](#)¹
- [Debian Developer's Reference](#)²
- [Debian Policy Manual](#)³
- [Debian Perl Policy](#)⁴
- [Debian MIME Policy](#)⁵
- [Debian Java Policy](#)⁶

Това са правилата, които са създадени и се спазват от debian maintainers, като те могат да са и upstream developers на код, специфичен или неспецифичен за Debian.

Както и с документите:

- [Debian Package Management HOWTO Version 1.2](#)⁷

¹<http://www.debian.org/doc/maint-guide/>

²<http://www.debian.org/doc/developers-reference/>

³<http://www.debian.org/doc/debian-policy/>

⁴<http://www.debian.org/doc/packaging-manuals/perl-policy/>

⁵<http://www.debian.org/doc/packaging-manuals/mime-policy/>

⁶<http://www.debian.org/doc/packaging-manuals/java-policy/>

⁷<http://www.linuxorbit.com/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=535>

- [Debian New Maintainer's Guide](http://www.debian.org/doc/maint-guide/)⁸, още достъпен и като пакет `maint-guide`, или стария му превод на <http://debian.gabrovo.com/docs/maint-guide/>
- [How To Create Your Own Debian Package](http://www.kclee.com/clemens/unix/HowToCreateYourOwnDebianPackage.html)⁹
- Ако ползвате локални (персонални) apt repositories, ето този бърз пример: <http://www.symonds.net/~rajesh/localdeb.html>
- повече обяснения можете да намерите в [Debian Repository HOWTO](http://www.isotton.com/debian/docs/repository-howto/)¹⁰
- навярно ще ви е полезен и пакета `mini-dinstall`:

13.2. Многото лица на `debconf`

- [The many faces of Debconf](http://kitenet.net/programs/debconf/)¹¹
- `debconf(1)`
- `debconf`, `debconf-utils`, `debconf-devel`

13.2.1. Обработване на конфигурационните файлове на пакетите

Това става чрез `debconf`-скриптове, извиквани от maintainer scripts:

```
# apt-get install base-files
Preparing to replace base-files 3.0.5 (using ../base-files_3.0.6_i386.deb) ...
Unpacking replacement base-files ...
Setting up base-files (3.0.6) ...
Configuration file '/etc/profile'
==> Modified (by you or by a script) since installation.
==> Package distributor has shipped an updated version.
What would you like to do about it ? Your options are:
Y or I : install the package maintainer's version
N or O : keep your currently-installed version
D : show the differences between the versions
Z : background this process to examine the situation
The default action is to keep your current version.
*** profile (Y/I/N/O/D/Z) [default=N] ? I
Installing new version of config file /etc/profile ...
```

13.2.2. Манипулация на файлове от други пакети

Divertions на файлове е начин да се укаже на `dpkg(8)` да инсталира даден файл или файлове от даден пакет не точно на неговото място, а на друго, с цел добавяне на някаква допълнителна функционалност. Обикновено от maintainer scripts се извиква `dpkg-divert(8)`, така щото файлове от даден пакет да могат да бъдат модифицирани при инсталирането или премахването на този пакет. Това например се наблюдава при инсталирането и премахването на пакета `dpkg-cross`. Нека преди да инсталираме `dpkg-cross` да направим следната проверка, т.е. с кой пакет идва скрипта `/usr/bin/dpkg-buildpackage`:

```
# dpkg -S /usr/bin/dpkg-buildpackage
dpkg-dev: /usr/bin/dpkg-buildpackage
```

Нека сега да инсталираме `dpkg-cross` и да наблюдаваме неговото поведение:

⁸<http://www.debian.org/doc/maint-guide/>

⁹<http://www.kclee.com/clemens/unix/HowToCreateYourOwnDebianPackage.html>

¹⁰<http://www.isotton.com/debian/docs/repository-howto/>

¹¹<http://kitenet.net/programs/debconf/>

```
# apt-get install dpkg-cross
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
dpkg-cross
0 packages upgraded, 1 newly installed, 0 to remove and 19 not upgraded.
Need to get 43.7kB of archives. After unpacking 205kB will be used.
Get:1 ftp://ftp.de.debian.org stable/main dpkg-cross 1.13.1 [43.7kB]
Fetched 43.7kB in 27s (1607B/s)
Selecting previously deselected package dpkg-cross.
(Reading database ... 154258 files and directories currently installed.)
Unpacking dpkg-cross (from .../dpkg-cross_1.13.1_all.deb) ...
Adding `diversion of /usr/bin/dpkg-buildpackage to
/usr/bin/dpkg-buildpackage.orig by dpkg-cross'
Adding `diversion of /usr/bin/dpkg-shlibdeps to
/usr/bin/dpkg-shlibdeps.orig by dpkg-cross'
Setting up dpkg-cross (1.13.1) ...
Updating Debian Packages of System Configurations.
```

Обърнете внимание какво ни се съобщава с реда `Adding diversion` Файлът `/usr/bin/dpkg-buildpackage` е бил преименуван на `/usr/bin/dpkg-buildpackage.orig`, т.е. заместен с друга версия на този файл, идваща с пакета `dpkg-cross`. Това се прави от неговия `postinst`-скрипт, който освен в директория `debian` в сорс пакета можете да намерите и в `/var/lib/dpkg/info/dpkg-cross.postinst`, след неговото инсталиране, разбира се. Този файл ще бъде върнат в предишното си състояние при премахването на пакета `dpkg-cross`. Това се прави от неговия `prerm`-скрипт. Този *diversion*, извършван от `dpkg-cross`, в случая е необходим, за да се модифицира скрипта `dpkg-buildpackage`, така че да може да поддържа крос-компилиране. За повече детайли вижте документацията на `dpkg-cross`. Забележете, че `dpkg-cross` изисква наличието на пакета `dpkg-dev`, в който се съдържа оригиналният скрипт `/usr/bin/dpkg-buildpackage` и който ще бъде модифициран:

```
# apt-cache show dpkg-cross | grep Depends
Depends: dpkg-dev, perl5 | perl
```

След като сме инсталирали `dpkg-cross`, проверяваме какво ще ни каже `dpkg -S` за `/usr/bin/dpkg-buildpackage`:

```
# dpkg -S /usr/bin/dpkg-buildpackage
diversion by dpkg-cross from: /usr/bin/dpkg-buildpackage
diversion to: /usr/bin/dpkg-buildpackage.orig
dpkg-cross, dpkg-dev: /usr/bin/dpkg-buildpackage
```

Т.е. вече е регистриран факта за модификацията на оригиналният скрипт `/usr/bin/dpkg-buildpackage` и потребителя се информира за това, когато попита с `dpkg -S`.

Нека сега премахнем пакета `dpkg-cross` и пак да наблюдаваме какво става:

```
# apt-get --purge remove dpkg-cross
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
dpkg-cross*
0 packages upgraded, 0 newly installed, 1 to remove and 19 not upgraded.
Need to get 0B of archives. After unpacking 205kB will be freed.
Do you want to continue? [Y/n]
(Reading database ... 154276 files and directories currently installed.)
Removing dpkg-cross ...
Removing `diversion of /usr/bin/dpkg-buildpackage to
/usr/bin/dpkg-buildpackage.orig by dpkg-cross'
Removing `diversion of /usr/bin/dpkg-shlibdeps to
/usr/bin/dpkg-shlibdeps.orig by dpkg-cross'
```

```
Purging configuration files for dpkg-cross ...
Updating Debian Packages of System Configurations.
```

Обърнете внимание, че вече ни се съобщава за `Removing diversion`. Т.е. нещата се връщат наистина в изходна позиция. Естествено, пак ще попитаме:

```
# dpkg -S /usr/bin/dpkg-buildpackage
dpkg-dev: /usr/bin/dpkg-buildpackage
```

Вече се съобщава само за `dpkg-dev` и нямаме наличието на `diversion`, която беше добавена от `dpkg-cross`. Т.е. след тези гимнастики имаме чисто премахване, въпреки промените и по файлове от чужди пакети, осъществявани от `dpkg-cross`.

Реализацията с код на това не е сложна, разбира се. Важното е, че е предвиден и подобен начин на модификация на файловете, идващи с пакетите.

13.3. *dh-make*: Начално дебианизиране на програма

Дотук вече трябва да сте добили представа как се борави с `debian binary packages` (или `deb`-файлове). Нека видим как се получават те. Т.е. захващаме се да разберем какво има и в `debian source packages`. Само да напомним, че без да сте прочели [Debian New Maintainer's Guide](#)¹², още достъпен и като пакет `maint-guide`, или поне да сте прехвърлили с поглед [стария му превод](#)¹³, ще ви е доста трудно да разберете за какво ще се говори в следващата глава.

Пакетът `dh-make` съдържа скрипта `dh_make(8)`, който генерира `debian source package` от `regular source code archive` (или `upstream sources`), подготвя `control`-файловете, както и предоставя примерна конфигурация за `debhelper(1)` инструментите, за която конфигурация обикновено е необходимо само малко донастройка, за да пасне за конкретния случай. Или с други думи, `dh-make` се използва за създаване на скелета или общия вид на `debian source package`, след което може да доредактирате файловете в директория `debian/` както намерите за добре. По-опитните `maintainers` могат и без него, но силно се препоръчва да се използва от начинаещите потребители, при което по-лесно и бързо ще свикнат с процеса на конфигуриране на `debian source packages`. Използва се например така:

```
# cd program-source-directory
# dh_make -e your.maint@address -f ../programname-x.y.z.tar.gz
```

Разбира се, ще отговорите на няколко въпроса, като дали това ще е `single` или `multiple binary packages`, библиотеки и прочее. За повече се обърнете към [Debian New Maintainer's Guide](#)¹⁴, главата `First Steps`, т. 2.4 `Initial Debianization`.

13.4. *debhelper*: Инструменти за инсталиращите и конфигуриращите скриптове

Най-добре ще е първо да си инсталирате пакета `hello-debhelper` и да прочетете документацията му, както и да разгледате примерите.

Скриптовите `dh_*` от пакета `debhelper` се извикват от `maintainer scripts` в `debian source packages`, и по точно от файла `debian/rules`. Използват се за автоматизиране на процедурите по построяването на `debian binary packages`, а именно, като инсталиране на определени файлове в дадения пакет, компресиране, установяване на съответните права и собственост, интегриране с `debian menu system` и много други. Повечето `debian source packages` използват скриптовите на `debhelper` като част от техния `build` процес.

¹²<http://www.debian.org/doc/maint-guide/>

¹³<http://debian.gabrovo.com/docs/maint-guide/>

¹⁴<http://www.debian.org/doc/maint-guide>

Ето и какви Perl скриптове се съдържат във версия 4.0.2 на пакета `debhelper`. За да ги листнем, изпълняваме:

```
$ dpkg -L debhelper | grep usr/bin
```

Следват кратки обяснения за скриптовите от пакета `debhelper`:

- **dh_builddeb(1)**: извиква **dpkg(8)** да билдва пакети.
- **dh_clean(1)**: почиства build директорията на пакета.
- **dh_compress(1)**: компресира файлове и оправя символните връзки в билд директорията на пакета.
- **dh_fixperms(1)**: оправя правата на файловете в билд директорията на пакета.
- **dh_gencontrol(1)**: генерира и инсталира `control`-файла в билд директорията на пакета.
- **dh_install(1)**: инсталира файлове, които не се нуждаят от специална обработка в билд директорията на пакета. За някои по-специални файлове са предвидени и по-специални скриптове.
- **dh_installchangelogs(1)**: инсталира `changelog`-файловете в билд директорията на пакета
- **dh_installcron(1)**: инсталира `cron` скриптове в `etc/cron.*`
- **dh_installdeb(1)**: в DEBIAN директорията инсталира файловете:
 - `package.postinst`
 - `package.preinst`
 - `package.postrm`
 - `package.prerm`
 - `package.shlibs`
 - `package.conf files`
- **dh_installdebconf(1)**: инсталира файловете, използвани от **debconf(7)** в билд директорията на пакета.
- **dh_installdirs(1)**: създава поддиректориите в билд директорията на пакета.
- **dh_installdocs(1)**: инсталира документацията в билд директорията на пакета.
- **dh_installemacs(1)**: за някои пакети е възможно или е необходимо да се извършва `byte-compiling` по време на инсталацията. Такъв пример е Emacs (е то оставаше точно пък той да не е ;-). Ако вашият пакет се нуждае подобна функционалност, точно този `dh_*`-скрипт ще извикате от файла `debian/rules`.
- **dh_installexamples(1)**: инсталира `examples` файловете в билд директорията на пакета, или по-точно в `usr/share/doc/пакет/examples`. Файлът `debian/packages.examples` може да съдържа списък с други файлове, които да бъдат инсталирани.
- **dh_installinfo(1)**: инсталира `info`-файловете. Файлът `debian/packages.info` може да съдържа списък с други файлове, които да бъдат инсталирани.
- **dh_installinit(1)**: инсталира `init`-скриптове в билд директорията на пакета.
- **dh_installlogrotate(1)**: инсталира конфигурационните файлове за `logrotate`
- **dh_installman(1)**: инсталира `man`-файловете. Файлът `debian/packages.manpages` може да съдържа списък с други файлове, които да бъдат инсталирани
- **dh_installmanpages(1)**: това е стария аналог на предната команда, вместо него използвайте **dh_installman(1)**
- **dh_installmenu(1)**: инсталира `debian menu` файловете в билд директорията на пакета. Автоматично генерира `postinst` и `postrm`, необходими да взаимодействат в пакета `menu`. Ако има файл `debian/package.menu`, то той се инсталира в `usr/lib/menu/package` в билд директорията на пакета (това е `debian menu` файлът, **menufile(5L)**). Ако има файл `debian/package.menu-method`, то той се инсталира в `etc/menu-methods/пакет` в билд директорията на пакета. Това е `debian menu method` файл. За повече информация вижте **update-menus(1)**.

- **dh_installmime(1)**: инсталира MIME-файловете в билд директорията на пакета. Ако има файл `debian/package.mime`, то той се инсталира в `usr/lib/mime/packages/пакет` в билд директорията на пакета.
- **dh_installmodules(1)**: регистрира kernel-модули посредством `modutils`. Ако има файл `debian/package.modules`, то той ще бъде инсталиран в `etc/modutils/пакет`
- **dh_installpam(1)**: инсталира файловете за поддръжка на PAM. Ако има файл `debian/package.pam`, то той ще бъде инсталиран като `etc/pam.d/пакет`.
- **dh_installwm(1)**: регистрира window manager. Файлът `debian/package.wm` може да съдържа списък с други прозоречни манежери.
- **dh_installxaw(1)**: инсталира конфигурационните файлове на Xaw wrappers в билд директорията на пакета. Ако има файл `debian/package.xaw`, то той ще бъде инсталиран в `usr/lib/xaw-wrappers/config/пакет` в билд директорията на пакета.
- **dh_installxfonts(1)**: регистрира шрифтовете за X
- **dh_link(1)**: създава символни връзки в билд директорията на пакета
- **dh_listpackages(1)**: листва binary packages, за които ще се използва `debhelper`
- **dh_makeshlibs(1)**: генерира shlibs-файл
- **dh_md5sums(1)**: генерира DEBIAN/md5sums файл.
- **dh_movefiles(1)**: файлът `debian/package.files` съдържа списък с файловете, които да бъдат преместени от `debian/tmp` в `subpackages`. Използвайте **dh_install(1)**, която напълно я замества, като може и доста други работи.
- **dh_perl(1)**: генерира зависимостите на Perl скриптовете вкл. и Perl модулите, от които зависят
- **dh_shlibdeps(1)**: генерира зависимостите за споделените библиотеки (много яка проверка)
- **dh_strip(1)**: стрипва или премахва дебъг символите от изпълнимите и библиотечните файлове, за да се намали размера им
- **dh_suidregister(1)**: не ползвайте този скрипт! Оставен е само за обратна съвместимост. Вместо него използвайте **dpkg-statoverride(8)**.
- **dh_testdir(1)**: тества директорията преди да се билдва пакета. Проверява за наличието на `debian/control`, както и за някои други основни файлове.
- **dh_testroot(1)**: проверява за това дали пакетът се билдва от потребителя `root`. За справка **fakeroot(1)**.
- **dh_testversion(1)**: не ползвайте този скрипт! Проверява дали е инсталирана правилната версия на пакета `debhelper`. Вместо него се използват `dependencies` и `conflicts` от `control` файла.
- **dh_undocumented(1)**: създава символна връзка към `undocumented.7.gz`. В случай, че пакетът няма *man* страница, разбира се.

Разбира се, не при всички `debian source packages` ще се извикват всички `dh_*` скриптове. Това ще зависи от съответния пакет, както и от решенията на `maintainer-a`. Най-добре ще е първо да изчетете добре *man*-страниците за тези скриптове, както и четенето на самите скриптове (Perl) не пречи, разбира се. Вземете, например, сорса на `MPlayer` от CVS или някое негово издание и разгледайте файла `debian/rules` за това какви `dh_*` скриптове се извикват от него. Същото можете да направите и с който и да е `debian source package`, включен в официалния Debian архив. Другото, което е добре да направите, е да разгледате самия `debhelper` като `debian source package`, и по специално неговата директория `debian/` и естествено Perl-скриптовете, които предоставя. Изпълняваме:

```
# apt-get source debhelper
```

Още пакети по темата: [dh-kpatches](#), [dh-consoledata](#), [dpsyco-devel](#)

13.5. Разработка и поддръжка на Debian source packages

13.5.1. *devscripts*: Съвременният начин

Пакетът *devscripts* е предвиден да облекчи живота на Debian package maintainers. Съдържа следните скриптове, като dependencies/recommendations са показани в счупените скоби:

- **bts(1)**: Команда за комуникиране с Bug Tracking System [*www-browser*, *mailx*]
- **dch(1)**, **debchange(1)**: Автоматично добавя entries към *debian/changelog* файловете
- **debclean(1)**: Пречиства (*purge*) а Debian source tree [*fakeroot*]
- **debuild(1)**: Wrapper за билдване на пакети без да е необходимо да се изпълнява *su* или да се мисли как да се стартира **dpkg(8)** да билдва, използвайки **fakeroot(1)**. Също така се оправя с общите проблеми на средата, *umask* и т.н. [*fakeroot*, *lintian*, *gnupg*]
- **debdiff(1)**: Сравнява две версии на Debian package, за да провери за добавени или премахнати файлове [*wdiff*, *patchutils*]
- **debpkg(1)**: Dpkg wrapper за менажиране и тестове на пакети без *su* [*perl-suid*]
- **debi(1)**, **debc(1)**: скриптове за инсталиране на пакети и извличане на тяхното съдържание
- **debit(1)**: скрипт за инсталиране на пакети и тестването им с *debian-test* [*debian-test*]
- **debrelease(1)**: Wrapper за *dupload* или *dput* [*dupload* | *dput*, *ssh*]
- **dscverify(1)**: Проверка целостта на Debian package от файловете *.changes* или *.dsc* [*gnupg*, *debian-keyring*, *libdigest-md5-perl*]
- **debsign(1)**, **debrsign(1)**: подписване на двойката файлове *.changes/.dsc* без да е необходима останалата част от пакета; двойката файлове може да се подписва отдалечено или да се изтеглят файловете и да се подпишат локално [*gnupg*, *debian-keyring*, *ssh*]
- **dpkg-depcheck(1)**, **dpkg-genbuilddeps(1)**: Определя използваните пакети по време на build на даден Debian package; удобен за определяне на Build-Depends в *control* файла [*build-essential*, *strace*]
- **grep-excuses(1)**: **grep(1)** на файла *update_excuses.html* за дадени пакети [*libwww-perl*]
- **mergechanges(1)**: merge на *.changes* файловете от пакет, който е билднат за друга хардуерна архитектура
- **plotchangelog(1)**: показва графика на данните от *changelog* файла [*libtimedate-perl*, *gnuplot*]
- **uupdate(1)**: интегрира upstream промените в debian source package [*patch*]
- **uscan(1)**: сканира upstream сайтовете за нови издания. Също така са включени и няколко примерни mail filters за филтриране на пощата от пощенските списъци на Debian чрез *exim*, *prosmail* и др. [*libwww-perl*]

13.5.2. *debmake*: Старият начин

Пакетът *debmake* е още един от пакетите, предназначени за разработка и поддръжка на Debian source packages. Но този пакет се използва все по-рядко.

- **deb-make(1L)**: генерира debian source package от upstream source код. Настройва файловете *control*. Предоставя примерна конфигурация за *debstd*, която в повечето случаи е използвана с минимална нужда от доредктиране
- **debstd(1L)** разполага със следните възможности:
 - автоматизира компресирането и инсталирането на документацията
 - генерира multiple binaries от един-единствен debian source package
 - генерира maintainer scripts и ги инсталира на подходящите места с подходящите права.

13 Конфигуриране на пакети - проблеми и решения

- може да модифицира много от debian config files чрез генериране на подходящи maintainer scripts.
- извиква **dpkg-shlibdeps**(1) за всички ELF binaries и генерира коректен shlibs файл за дадените библиотеки автоматично.
- проверява symlinks за manpages / documentation и пренасочва в случай, че е необходимо.

Глава 14

Изграждане на дистрибуцията и средства за контрол

14.1. Packaging - *Debian official maintainer's way*

На практика след като се установи, че даден софтуер си заслужава да се включи в официалния архив от пакети на Debian, се изготвя *debian source package*, който включва *upstream source* и директорията с *maintainer scripts*. От този *debian source package* след това се получават *debian binary packages (deb-файлове)* за различните хардуерни архитектури, като се отчита и факта, че има и такъв софтуер, който не е пригоден или предвиден за всички хардуерни архитектури, а само за една или няколко. Обикновено за получаването на базовите или начални версии на *maintainer scripts* за *debian/* се използват Perl скриптовете от пакета *debhelper*, след което се донастройват специфичните за пакета неща. В крайна сметка се създава унифицирано управление на процеса по конфигуриране на сорса, компилация, свързване и евентуално последващо конфигуриране и преконфигуриране на софтуера за различните хардуерни архитектури. Разбира се, *maintainer scripts* не изместват стандартните *GNU devel tools*, а работейки преди или над тях ги използват по подходящ начин. Естествено, в крайна сметка се извикват стандартните *GNU devel tools* чрез файла *debian/rules*, който се явява стандартен файл за програмата *make*, само че не е именуван като *Makefile*, а започващ с *shebang* (`#!`) ред в началото си, указващ пътя до *make*. Използват се още доста файлове в *maintainer scripts* и пример за това, как точно стават нещата, е даден в [Debian New Maintainers' Guide](#)¹.

14.1.1. Средства за контрол

- `linda`
- `lintian`
- `debian-test`
- и други...

Пример 2.1: Пример за пакетирание на `bgtex-v2`

FIXME: пример step-by-step. Или пък с програмата за Дневници по ДДС?

¹<http://www.debian.org/doc/maint-guide/>

14.2. Packaging - *at home*

Пример 2.2: Пример за бързо и *dirty* получаване на *binary packages*

Нека да получим deb-файлове по-възможно най-краткия път за изключително кратка програма, която се получава от един единствен сорс файл, който изисква само наличието на основната C библиотека, която всички имат, и освен това не конфликттира с нищо, така че нашите *maintainer scripts* ще са възможно най-прости. По-лесен случай май не може да се измисли ;-).

Нека имаме [сорса на една такава програмка](#)², който сме компилирали с `gcc -o wsver wsver.c` и сме получили динамично свързан изпълним файл `wsver`, за който проверяваме и се убеждаваме, че наистина не е претенциозен:

```
# ldd ./wsver
libc.so.6 => /lib/libc.so.6 (0x40026000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
# dpkg -S /lib/libc.so.6 /lib/ld-linux.so.2
libc6: /lib/libc.so.6
libc6: /lib/ld-linux.so.2
# apt-cache show libc6 | grep Section
Section: base
```

Идеално... Всички имат `libc6`, защото е в `base`, така че спокойно можем да прескочим описването на каквито и да са зависимости и конфликти и минаваме направо, прескачайки целия [Debian New Maintainers' Guide](#)³.

```
# mkdir -p wsver/usr/bin wsver/DEBIAN
# cp wsver wsver/usr/bin
# echo "Package: wsver" > wsver/DEBIAN/control
# echo "Version: 1" >> wsver/DEBIAN/control
# echo "Architecture: i386" >> wsver/DEBIAN/control
# echo "Maintainer: You <you@some.net>" >> wsver/DEBIAN/control
# echo "Description: Check web server version" >> wsver/DEBIAN/control
# dpkg-deb -b wsver
dpkg-deb: building package 'wsver' in 'wsver.deb'.
# dpkg -i wsver.deb
# whereis wsver
wsver: /usr/bin/wsver
# apt-cache show wsver
Package: wsver
Status: install ok installed
Maintainer: You <you@some.net>
Version: 1
Description: Check web server version
# wsver localhost
```

Подобно пакетиране не се прави от *debian maintainers* и е само като пример, който едва ли ще срещнете някъде. Това е *fast & dirty home made packaging* на специално избран, възможно най-непретенциозен сорс код. Но понякога дори и *fast & dirty* може да се окаже, че е по-приемливо за самия потребител. Ако го компилирате за `i386`, можете спокойно да споделите този `deb`-файл с други `x86` потребители, както можете да го компилирате за произволна друга хардуерна архитектура. В случая не сме оформили *debian source package*, от който да се получават един или няколко *binary packages* за различните хардуерни архитектури, също така не сме направили проверка с програми като `lintian`, `linda` и т.н.

Пример 3: [Mplayer](#)⁴ — директорията `debian/` идва с `upstream sources`

Нека бъде от `current CVS` и отделните му `Releases`, които ако сте мързеливи, може да получите барабар с шрифтове, кожи и кодеци с помощта на този красив [Makefile](#)⁵, който да кажем сте съхранили в `/usr/local/src/MPlayer/` и сте разгледали какво точно прави, разбира се. Макар и все още невключен в официалния Debian архив, тези сорсове дефакто са оформени като

²<http://danchev.fccf.net/files/wsver/wsver.c>

³<http://www.debian.org/doc/maint-guide/>

⁴<http://www.mplayerhq.hu>

⁵<http://danchev.fccf.net/files/mplayer/Makefile>

debian source packages. В `main/` директорията разгледайте съдържанието на директорията `debian/`, за да добиете идея за *maintainer scripts*, които в момента са такива и които могат да бъдат промени впоследствие. Изпълнявайки от `main/`:

```
# fakeroot debian/rules binary
```

или:

```
# DEB_BUILD_OPTIONS="--compile-options-here" debian/rules binary
```

Ще получите съответния *binary package*, можете да генерирате *list files* за *apt repository* (`dpkg-scansources(8)`, `dpkg-scansources(8)` и т.н.) Нека за момент предположим, че нямате инсталирана библиотеката `SDL` и съответно сте получили изпълним файл на `Mplayer`, който е без такава поддръжка, което не е фатално, но пък може да се получи така, че да имате липса на нещо (най-често някоя библиотека), поради което компилацията на `Mplayer` няма да завърши успешно. Освен това нямате никакви *list files*, които да подсказват `Build-Depends` информация (т.е. какво е нужно като файлове и в кои пакети са те, за да се изкомпилират изходните кодове успешно) за `Mplayer` на командата `apt-get build-dep`, т.е. разполагаме само с `upstream` сорсовете на `Mplayer`, в които има и `debian/` директория, съдържаща *maintainer scripts*. Тогава ако нямаме `auto-apt`, го инсталираме и конфигурираме:

```
# apt-get install auto-apt
# auto-apt update updatedb update-local
# fakeroot auto-apt run debian/rules binary
```

Ще бъдете запитани за всичко, което бъде потърсено и ненамерено в момента във вашата система, и вие ще решите кое да бъде изтеглено и инсталирано и кое да бъде отказано... `debian/rules` е най-обикновен файл за програмата `make(1)`, разгледайте го.

Нещо повече: ако предположим, че имаме *upstream sources* на `Mplayer` без *maintainer scripts*, намиращи се в `debian/` директорията, то `auto-apt` пак ще свърши работа:

```
# auto-apt run ./configure --prefix=/usr/local/somewhere --more-options-here
# auto-apt run make
```

Отново, ако има липси, необходими за компилационния и свързващия процес, ще бъдете запитани дали искате да ги инсталирате. Това важи, разбира се, и за произволни сорсове, за които `auto-apt` ще може да намери необходимото в рамките на източниците от пакети, до които има достъп, за да търси съответните липсващи файлове. Как да инсталираме необходимото при поискване, е описано в [APT-HOWTO](#)⁶.

Може да продължите с по-сложни експерименти, следвайки горните официални документи, за сорсове, изискващи по-задълбочено конфигуриране. Добър пример е даден в [Debian New Maintainers' Guide](#)⁷. Ще видите, че нещата не винаги са толкова лесни и прости. Ето например следващото може да е доста полезно за тези „advanced“ или „power“ root потребители, които обичат да компилират и инсталират `system-wide`, без да разбират и осъзнават какво всъщност правят, и докарват своите *GNU/Linux* дистрибуции до състояние на омазан до безпомощност `виндовс`: [библиотеки](#)⁸, [поделени библиотеки](#)⁹, [Debian Library Packaging guide](#)¹⁰

Точно обратното, може да се наложи тотален контрол върху инсталирания софтуер в системата, така че да се предотвратява наличието на `broken stuff`. Налага се да се контролират зависимостите и конфликтите при многобройните парчета софтуер, които имате в системата, особено при споделените библиотеки. Например, защо се налага и как се пресмятат `shared library dependencies`: `dpkg-shlibdeps(1)`, `dh_shlibdeps(1)` или опитайте на <http://www.fifi.org/doc/HTML/>. Като че ли няма аналог извън `Debian` ;-) За това се грижи дистрибуцията или `package maintainers`, но вие ако знаете повече или искате някакво по-`custom` решение, разбира се, че ще се намесите. Ето няколко случайни примера:

⁶<http://debian.gabrovo.com/docs/apt-howto/>

⁷<http://www.debian.org/doc/maint-guide/>

⁸<http://www.debian.org/doc/debian-policy/ch-files.html#s11.2>

⁹<http://www.debian.org/doc/debian-policy/ch-files.html#s11.3>

¹⁰<http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html>

Пример 4: Промяна на Depends and Conflicts

Ако чувствате, че знаете какво правите, можете спокойно да дръпнете интересуващите ви *debian source packages*. От всеки такъв пакет се получават по един или няколко *debian binary packages* (deb-файлове), всеки от които доставящ по една или няколко изпълними програми и/или библиотеки и др., като разни *shared data files*. ... Можете да промените зависимостите и конфликтите между тях като редактирате файловете *debian/control*, *debian/rules* и други в директорията *debian/*. Може да се намесите и в самите *upstream sources* и след това да ги инсталирате. Трябва да имате предвид, че променянето на дадена опция при компилацията или свързването може да даде, или не, отражение върху работата на други програми. Променянето на *upstream sources* също може да доведе до такова поведение. Така че мислете глобално, когато се намесвате. Изключително голяма радост за очите е да се наблюдава как на разни системи „advanced“ потребители компилират и инсталират каквото им дойде на ума и както се сетят, без наличието на какъвто и да е мисловен процес. Важното е, че компилират нещо там (т.е. наблюдават как например gcc компилира) и се изживяват като „advanced“, но че после може да има breaks въобще на осъзнават, я усетили, я не. Debian учи как да не се правят подобни lame изпълнения и то на „сляпо“, а с механизмите за контрол, които предоставя, препоръчва нещата да се правят така, че да може да се наложи от потребителя прецизен и тотален контрол върху софтуера, бил той като *debian source* или *binary packages*. Разбирайки това, вече разбирате и колко много weakness и мъка може да има по този свят. Познаването на [Debian New Maintainers' Guide](#)¹¹ и [Debian Developer's Reference](#)¹² би било от изключителна полза, както и познанията за *upstream sources*, в които намесите са задължителни. След като направите промените в сорса и получите вашите *custom binary packages* и ги инсталирате, можете да се погрижите да ги заковете чрез *pinning feature* на *apt*, така че да не *upgrade*-нете вашата *custom* версия, без да усетите или забележите. Ако при бъдещи промени в системата, като *upgrades* или *downgrades*, все пак се наложи да *unpin* („отковете“) вашия *custom pin*-нат пакет, за да може да бъде *upgrade*-нат например, то тогава ще бъдете уведомен от *apt* и ще помислите кое от двете по ви изнася да изберете, като пак може можете да внесете вашите промени (мислейки!) в новата версия на вече бившия закован и *upgrade*-нат *package* и да го заковете отново.

Пример 5: Намеса в upstream sources, препакетиране и инсталиране

Нека ви се налага по някаква причина (и вие наистина знаете какво правите) да промените нещо в сорса на *glibc*. Да речем, съдържанието на някой заглавен файл: искате да увеличите стойността на някоя дефиниция в */usr/include/директория/файл.h* да прекомпилирате *glibc* и да го инсталирате. Прекомпиляция може да се наложи, за да се отрази тази промяна и в библиотечните файлове в */usr/lib/* и вероятно на още няколко места, такава може да прецените, че не се налага, и просто да се задоволите само с промяна на заглавния файл без прекомпиляция и инсталация. Да предположим, че все пак сте преценили, че се налага прекомпиляция на сорса на *glibc* с последваща инсталация. Няма да преоткриваме колелото и да обясняваме, че изтеглянето на чистите *upstream sources*, разпакетирането им, компилирането им и инсталирането им *system-wide* с *./configure; make; make install*, без да се проверява и осъзнава кой файл точно къде се инсталира, е пълна глупост и може да нанесе големи проблеми на системата най-малко поради факта, че трябва да има гаранция, че старите файлове ще бъдат напълно заменени от новите, както и че няма да останат стари *stalled files*, които да пречат по някакъв начин. Ако имате желание, може да се занимаете с въпроса, проверявайки кое къде отива при новата компилация и как да отстраните старата ;-). Ето как ще го направим бързо и безопасно като оставим горната рутинна и тежка работа по „бройкането“ на стари/нови файлове на *dpkg*. Той знае как да премахне старите и да инсталира новите файлове и няма смисъл ние да си бодем очите с подобна досадна и незаслужаваща времето ни задача. Няма да вадим корен квадратен от голямо число с лист и молив, я. Това, че знаем алгоритъма, не е основание да го правим с беден инструментариум. Първо, проверяваме в кой *binary package* е интересуващият ни файл */usr/include/директория/файл.h*:

```
# dpkg -S /usr/include/somedir/somefile.h
```

да кажем, че *dpkg* отговори, че този файл идва с пакета *libc6-dev*.

¹¹<http://www.debian.org/doc/maint-guide/>

¹²<http://www.debian.org/doc/developers-reference/>

Изтегляме съответния му *source package*, от който този *binary package* е получен, защото от един *source package* могат да се получат един или няколко *binary packages*, но това не е задължително, разбира се, винаги да е точно така, като преди това проверяваме дали имате необходимото, за да може да бъде компилиран успешно този *source package*:

```
# apt-get build-dep libc6-dev
# apt-get source libc6-dev
```

В текущата директория, забележете, получаваме: `glibc-2.3.1/, glibc_2.3.1-5.diff.gz, glibc_2.3.1-5.dsc` и `glibc_2.3.1.orig.tar.gz`. Няма да обясняваме кое какво е, в документацията си пише, например в [Debian New Maintainers' Guide](#)¹³.

Променяме сорса. В директорията `debian/` са конфигуриращите скриптове на `maintainer-a`, всичко останало са чистите *upstream sources*. Забележете, че наименованието на софтуера *upstream* може да не съвпада в имената на пакетите в GNU/Linux дистрибуцията, а също че *upstream* може да бъде разбит на няколко пакета. Трябва да знаем къде да търсим и променим това, което ни трябва, в *upstream sources* на *glibc*, променяме и версията на пакета в `debian/changelog`. Забележете, че в `debian/patches/` са предоставени и кръпките които са приложени към *upstream sources* на *glibc*, така че внимавайте вашите промени да се „понасят“ с тях. Не всички *upstream sources*, идващи с *debian source packages*, се закърпват. Такива кръпки, специфични за дистрибуцията, се прилагат от `maintainers` много внимателно и само когато това наистина се налага, така че да не се „вадят очи, вместо да се изписват вежди“. Понякога това може да се наложи поради изискванията, които се поставят от дистрибуцията. Например, за по-добра съвместимост с *File Hierachy Standard* или *Linux Standard Base*, по-добра съвместимост с различни хардуерни архитектури като IA-64, ARM, HPPA, Sparc64, S390x, . . . , и ядра като Hurd-on-GnuMach и т.н. и т.н. . . Тези кръпки най-вероятно след това влизат и в следващия официален *upstream release* на *glibc* или който и да е софтуер в дадения случай. По подобен начин дефакто Debian служи и се „експлоатира“ като платформа за пренасянето на XFree86 (това „86“, напомнящо x86 или PC в името, е дразнещо определено ;-) за GNU/Linux на доста хардуерни архитектури, вкл. и `hurd-i386`.

Компилираме и получаваме *binary package(s)*, кой(и)то инсталираме:

```
# debian/rules binary
# dpkg -i ../*.deb
```

Допълнение: тук дори можете ако имате желание да си направите *local apt repository* и да го добавите в `/etc/apt/sources.list`, така ще ползвате `apt` да смята зависимостите и конфликтите и да подава пакетите в определения ред на `dpkg`, вместо просто само `dpkg`. Един бърз пример как става това е [How to do apt-get install for local debs](#)¹⁴

Забележете, че от този *source package* се получават няколко *binary packages* (`deb`-файлове), които ние ще инсталираме. Не е задължително от всеки *source package* да се получават по няколко *binary* такива. Частен случай е, когато от един *source package* се получава един *binary package*. Това се контролира от `debian/control`, в зависимост от решенията на `maintainer-a`, които го е създал или от вас разбира се.

Така всичко ще е под пълен и бърз контрол, като рутинната и досадна работа по издирването на файловете при инсталацията сме оставили на `dpkg`. Това е случая, когато инсталирате вашия си *custom glibc build system-wide*, т.е. по-опасния случай, а иначе в `/usr/local/` в отделни директории може да имате компилирано и инсталирано *glibc* колкото пъти се сетите и както се сетите и във всеки отделен терминал да `export` различен `LD_LIBRARY_PATH`, за да ползвате различен *build* на *glibc*. . . ;-). Тази библиотека е основна градивна единица и няма да се спираме на това колко динамично свързани програми зависят от нея, така че внимателно с *custom-изацията*. Тук по-наблюдателните ще отправят основателен въпрос: как `dpkg`, който е динамично свързан и зависи от библиотечни файлове на *glibc*, ги премахва и докато инсталира новите, и все пак продължава да работи добре. Отговорът е, че докато прави тази операция, `dpkg` заедно със старите споделени библиотеки, с които е свързан динамично, е зареден в паметта. След като завърши тази операция, на следващото стартиране `dpkg` ще се свързва с новите такива споделени библиотеки. Точно поради това, че е зареден в паметта (е то няма иначе къде другаде;-), `dpkg`

¹³<http://www.debian.org/doc/maint-guide/>

¹⁴<http://www.symonds.net/~rajesh/localdeb.html>

може да `upgrade`-ва или `downgrade`-ва в момента инсталирания `dpkg` на диска (образно казано), така че `apt-get install dpkg apt` или `dpkg -i dpkg_*.deb apt_*.deb` са операции абсолютно в реда на нещата и няма място за опасения. Разбира се, може да имате и статично свързан `dpkg`, както и други важни програми. Ако изтеглите някой *source package* на `dpkg` от `ftp/http mirrors` или пък някой сорс (даден `tar`) на `dpkg` от cvs.debian.org¹⁵ и разгледате `debian/control`, ще видите, че може да се получи и е предвиден и *binary package*: `dpkg-static`. [FreeBSD](http://www.FreeBSD.org)¹⁶ държи някои такива статично свързани програми в отделна директория `/stand`, но те са си за `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin` и т.н. но именувани, например, с подходящ суфикс `-static` или подобен, за да е ясно за какво става дума, не че с `ldd(1)` не може да се провери кое е статично и кое е динамично свързано — въпрос на вкус.

Повечето потребители, наблюдавайки работата на `dpkg` и `apt` при себе си, остават с впечатлението, че заслугата е изцяло и единствено на тези програми. Някои дори не забелязват, че `apt`, след като си свърши своята част от работата, извиква `dpkg`, за да прави реално инсталацията. Разбира се, че това са корави програми, изпитани и разширявани постепенно с времето. Също така не е случаен и фактът, че функционалността е разпределена между тях (и други разбира се), а не набутана само в една от тях — това е по стара *Unix* традиция, за да не се достига до *bloatware* при претрупване на едно приложение с прекалено голям брой *features*, които е трудно да се поддържат реализирани в едно единствено приложение. Всичко това се вижда от потребителите на пръв поглед, но като се погледне малко по-обстойно на нещата, се разбира, че `dpkg` и `apt` се „захранват“ и „разполагат“ със страшно много и прецизна информация, без значение какво имате инсталирано в системата. Това са *list files* в `/var/lib/apt/lists/`, които вие обновявате при всеки `apt-get update` в зависимост от посочените от вас източници в `/etc/apt/sources.list`. Така те разполагат с пълна информация за пакетите, достъпни от споменатите от вас източници, освен информацията, която пакетите носят сами със себе си. Дръпнете някой *debian source package* с `apt-get source пакет`, разгледайте го, и по-специално директорията `debian/`, получите от него *binary package(s)*, след това разархивирайте един *binary package deb*-файл, изпълнявайки `ar -x пакет_версия.deb`, и след това разгледайте какво има в `control.tar.gz`, в `data.tar.gz` е самото приложение. След това разгледайте тези `*Packages`, `*Sources`, `*Release` файлове в `/var/lib/apt/lists/`, защо така са именувани при вас и какво съдържат. Грер-вайки небрежно, потърсете вашия пакет. Не редактирайте тези файлове, ако не разбирате какво правите, но ако ги омажете, можете да ги изтриете и да ги обновите пак. Тази мета-информация идва от файловете `Packages`, `Sources`, `Release` от Debian архива(ите), съответно избрани от вас и посочени в `/etc/apt/sources.list`. Тя пък е получена от контролните файлове в *debian source packages*, от които са получени съответните *debian binary packages*. Няма как да не се сетите, че *debian source packages*, и по-точно контролната информация, която носят те, се създава и поддържа от *debian maintainers* и потребителите, ако решат, могат да се намесват. Дотук говорихме за една и съща мета-информация, разпространявана независимо по няколко „канала“ и поради наличието на която може да се оценява коректността на работа на произволно избрана селекция от пакети, налични или неналични в потребителската система. Т.е. всичко в крайна сметка зависи от мета-информацията, която тръгва от *debian source packages* (тя трябва да е пълна и прецизна), отива в *debian binary packages* и *list files* в архива и впоследствие и в `/var/lib/apt/lists/` на потребителя. Разбира се, че последната не е задължителна и може да имате само *debian binary packages* или *debian source package*, от които да получите *debian binary package(s)*, и мета-информацията, идваща с тях. Това пак ще ви свърши някаква работа, но е доста скромнен случай, разбира се. Съвсем отделен е въпросът, че `dpkg` си поддържа своя база данни за състоянието/статуса на пакетите, която вие запитвате с `dpkg -l`, `dpkg -get-selections` и т.н.

Съществуват и т.н. *Contents files* (намиращи се на `огледало/debian/dists/издание/Contents-архитектура.gz`), които съдържат пълната информация за това, кой файл в кой пакет се намира. Всичко това е за дадената архитектура, разбира се, понеже между тях може да има разлики. Тази информация е полезна, когато знаете файла (или пътя до него) и ви интересува в кой пакет се намира той. Например, скриптовете `apt-file` (`apt-file update`) и `auto-apt` (`auto-apt update updatedb`)

¹⁵<http://cvs.debian.org>

¹⁶<http://www.FreeBSD.org>

update-local) изтеглят тези *Contents files*.

Споделят се и аналогични впечатления, като за `dpkg` и `apt` пакети, или пакети предоставящи набор от програми, и от работата на скрипта `auto-apt`, който прекъсва процеса на компилация или свързване за произволни дървета от сорс код, изнамира в кой пакет е липсващия *header/lib*, стига този пакет да е в обсега му (`sources.list`) разбира се, предлага го за инсталация, които вие може да откажете, и след това *resume*-ва процеса на компилацията и/или свързването, от там докдето е бил временно прекъснат. Няма магии тука, има добре обмислен дизайн, който служи като добра основа за създаване на едни или други инструменти с една или друга функционалност. Тези инструменти също трябва да са на ниво, разбира се.

Глава 15

Компилиране и инсталиране на софтуера - проблеми и решения

15.1. Local APT Repositories

15.1.1. Създаване и управление на локално apt-хранилище с готови deb-файлове

Ако разполагате с *.deb*-файлове в някоя локална директория, за да ги инсталирате, единият от начините е да се изпълни `dpkg -i` по реда на зависимостите, което може да бъде и досадно. За това ще е по-културно да се впрегне `apt` да свърши това вместо нас.

- Създайте файл `overridefile`, съдържащ списък на файловете в директория с пакети, т.е. списък на файловете *.deb* намиращи се там:

```
$ cd /usr/local/mypackages
$ find . -name "*.deb" > overridefile
```

Сега файлът `overridefile` ще има съдържание, подобно на това:

```
./aalib1_1.2-helix1_i386.deb
./abiword_0.7.10-helix1_i386.deb
./bonobo_0.23-helix4_i386.deb
./codecommander_0.9.7-helix1_i386.deb
./eog_0.5-helix1_i386.deb
```

Има, разбира се, и други опции, които можете да укажете при създаването на `overridefile`, които за момента не са ни необходими. За повече информация, прегледайте съответно `dpkg-scanpackages(8)` за *debian binary packages* и `dpkg-scansources(8)` за *debian source packages*.

- `$ dpkg-scanpackages . overridefile > Packages`
- Добавете тази директория като източник във файла `/etc/apt/sources.list`:
`deb file:/usr/local/mypackages ./`
- Обновете `apt`, така че и този източник да се прибави в списъка му:
`# apt-get update`
- Сега вече можете да инсталирате който и да е пакет от тази директория, като естествено трябва да бъдат достъпни всички пакети, от които той зависи и които ще бъдат предложени от `apt` автоматично:
`# apt-get install abiword`

debuid: binary пакети от source пакети

В пакета `devscripts` ще намерите една много полезна Perl програма наречена **debuid(1)**. Използва се за получаване на `debian binary packages (deb-файлове)` от `debian source packages`:

```
# debuid -b -uc -us
```

15.1.2. apt-build: Инсталиране на пакети от сорс

Накратко

Колкото и странно да звучи на някои, ако искате да имате винаги възможно най-актуалните версии на софтуера за *GUN/Linux*, най-лесният път към това е използването на **Debian**¹. Противно на общоприетото схващане, тази дистрибуция се снабдява най-пъргово с всички нови програми и ви предоставя достатъчно гъвав и лесен начин за тяхното управление с помощта на серия инструменти.

Добре е известно, че **Debian**² се дели на три — да ги наречем условно — издания: *stable*, *testing* и *unstable* (има и *project/experimental*, но там нещата са наистина за експерименти). В *stable* влиза проверен от времето (и хакерите) софтуер, на който можете да разчитате за сериозни задачи, изискващи максимална сигурност и стабилност. Логично е, софтуерът в *stable* да е по-старичък. Официалните ISO-имиджи на *stable* можете да изтеглите от Интернет и с тяхна помощ да си направите една базова инсталация, която впоследствие да надградите и актуализирате до *testing*, където влизат по-нови неща, подлежащи на усилено тестване и кандидати за *stable*, или направо да преминете към *unstable*, където буквално има всичко, което е излязло на бял свят към момента. Имате и друга възможност: част от пакетите да държите от *stable* (например, ядрото, базовата система), а друга част, която е по-клиентски ориентирана (като KDE и GNOME), да вземете от *unstable*. Много хора се оплакват, че *Debian stable* е толкова стар, че дори се инсталира с ядро 2.2.x. Така е по подразбиране, но ако четат внимателно, ще видят, че инсталацията на Debian предлага широк набор от ядра, между които и 2.4.x. В този текст няма да се занимаваме с основните команди на програмата **apt-get(8)**, с които би трябвало да е запознат всеки любител на тази дистрибуция, а ще обърнем внимание на един друг инструмент — **apt-build(1)**. С негова помощ можете напълно автоматизирано да си направите собствено *apt-хранилище* (или *local apt repository*), смисълът от което е познат на всеки почитател на сорс-базираните дистрибуции. Ползата от подобно хранилище е огромна: избирате компилатор по собствено желание и опции за оптимизация също по свой вкус, като така постигате максимална производителност за цялата система, управлявате хранилището със стотиците компилирани от вас пакети с **apt-get(8)**, подобно на всеки друг дебиански източник. Като собственик на преклонно стар компютър, аз не мога да си позволя лукса, предоставян например от **Gentoo**³, да компилирам от изходен код цялата си дистрибуция. А и това не е необходимо. Оставяме настрана ядрото, за което има специален инструмент **kernel-package(5)**. Съсредоточаваме се само върху най-често използваните потребителски приложения, за да не губим излишно време в безкрайни компилации, като съобразяваме нуждите си с възможностите на самия компютър.

Какво е необходимо

Първо трябва да се уверим, че във файла, в който се описват източниците на дебианския софтуер, сме въвели и пътя към сорсовете. Би трябвало в `/etc/apt/sources.list` да виждаме нещо такова:

```
deb ftp://ftp.bg.debian.org/debian stable main contrib non-free
deb ftp://ftp.bg.debian.org/debian unstable main contrib non-free
deb http://security.debian.org/ stable/updates main
deb http://security.debian.org/ testing/updates main
deb-src http://security.debian.org/ stable/updates main
deb-src http://security.debian.org/ testing/updates main
```

¹<http://www.debian.org>

²<http://www.debian.org>

³<http://www.gentoo.org>

```
deb-src ftp://ftp.bg.debian.org/debian stable main contrib non-free
deb-src ftp://ftp.bg.debian.org/debian unstable main contrib non-free
```

След това, трябва да инсталираме `apt-build`:

```
# apt-get install apt-build
```

При самата инсталация ще бъдем попитани къде искаме да бъде създадена директорията, в която ще се съхраняват пакетите от нашето хранилище, кой компилатор предпочитаме (например, `gcc-3.2`), дали искаме да се прилагат някакви специални опции за оптимизация. По подразбиране директорията на нашето хранилище е `/var/cache/apt-build/repository`. Накрая ще бъдем попитани дали искаме въпросната директория да бъде описана във файла с дебиански източници, на което ние отговаряме утвърдително.

Създаване на собствени пакети с `apt-build`

Можете да се изненадате колко е просто, но всъщност цялата процедура се свежда до една единствена команда. Пакетите със сорсове носят същото наименование като бинарните. Да речем, че искате да си инсталирате последната версия на [Mozilla](http://www.mozilla.org)⁴.

```
# apt-build install mozilla-browser
```

Оттук нататък `apt-build` ще се погрижи да си изтегли и инсталира всички необходими за компилацията пакети, за да компилира успешно `mozilla`, ще съхрани новите бинарита в хранилището, откъдето ще ги инсталира и вие ще можете да ги управлявате вече по стандартния начин с `apt-get(8)`.

Създаване на `.deb` пакети и добавяне в хранилището

Добре, няма нищо по-лесно от това да инсталирате от официалните източници на *Debian* пакети, като ги компилирате локално с `apt-build(1)`. Но, да речем, че искаме да пипнем тук-там в сорса или да компилираме пакет, който не влиза никъде в *Debian*, след което ще го добавим в нашето хранилище. В първия случай можем да изтеглим само сорса с `apt-get(8)`:

```
# apt-get source mozilla-browser
```

Сорсът ще се изтегли и разархивира в текущата директория. Можем да направим каквото искаме по него и след това да го компилираме с `dpkg-buildpackage(1)`. В резултат ще получим отново дебиански бинарита (`deb`-файлове), които можем да копираме в нашето хранилище. След като ги копираме, трябва да кажем на `apt-build(1)` да актуализира съдържанието на файловете `Packages.gz` и `Sources.gz`, от които чете пък `apt-get(8)`, за да се ориентира къде какви пакети има.

```
# apt-build update-repository
```

Полезни процедури с `apt-build`

– Изчистване на работната директория

Когато `apt-build(1)` компилира пакети, сваля на хард диска много *devel* библиотеки и сорсове. Бързо ще почувствате липсата на дисково пространство, ако не разчиствате редовно работното пространство на компилатора, което по подразбиране се намира в директорията `/var/cache/apt-build/build`:

```
# apt-build clean-build
```

По същия начин можете да разчиствате и локалния кеш на `apt-get(8)`:

```
# apt-get clean
```

⁴<http://www.mozilla.org>

- Премахване на *devel* библиотеките и разчистване на ненужния софтуер с **debfooster(8)**
Знаем, че за компилацията на софтуера **apt-build(1)** инсталира допълнително много хедърни файлове и *devel* библиотеки, които бързо запълват дисковото пространство, а в същото време нямаме нужда от тях, освен по време на самата компилация. За разрешаването на този проблем *Debian* предлага още един много удобен инструмент — **debfooster(8)**. Достатъчно е само да го стартирате, за да разберете какво прави: пита ви за всеки пакет, който е възможно да бъде премахнат безболезнено, и грижливо "измита" отпадъците.

15.1.3. *apt-src*, *pbuilder*

FIXME: Съдържание трябва да има тука ;-)

15.1.4. *kernel-package*: Компилиране на ядро по дебиански

Накратко

По желание на потребителя ядрото може да бъде и като *debian package*, взет наготово от Debian archive-а или получен при потребителя, който освен изтегляне на *kernel sources* от където пожелае, четене на *Documentation/Changes* за евентуален upgrade на някои *user space utils*, като *gcc*, *make*, *binutils* и т.н., конфигуриране, компилиране и инсталиране както намери за добре, може да използва и **kernel-package(5)** предоставящ *Perl scripts* чрез които може да се получи собствен (*custom*) *debian kernel package(s)* за *kernel images* и евентуално и *kernel modules* от *upstream kernel sources* (изтеглени например от kernel.org⁵) или от *debian packages* като *kernel-source-**, *kernel-patch-**, *kernel-image-**, *kernel-headers-**. За повече подробности се обърнете към *man kernel-package(5)* или документацията в */usr/share/doc/kernel-package/* както и статията http://www.osnews.com/story.php?news_id=2949

Ето как се използва инструмента **kernel-package(5)**, който е характерен за *Debian GNU/Linux*. Първо инсталираме самия **kernel-package(5)** разбира се:

```
# apt-get install kernel-package
```

След като се инсталира, можете да въведете личните си данни в */etc/kernel-package.conf*, така, че създаденият от вас *.deb* пакет с новото ядро ще носи информация за онзи, който го е създал (ставате *maintainer*:)). След като изтеглите сорса на ядрото и го компилирате и пакетирате в удобен за управление формат, можете да инсталирате новото ядрото по официалния за дистрибуцията начин, като по този начин го направя част от системата. Именно тези възможности ми даде инструментът **kernel-package(5)**. Разбира се това е единия вариант, който е опционален, а не задължителен.

Основни процедури

В Debian можете да си инсталирате сорса на избраното от вас ядро, леко пачнат от авторите на дистрибуцията (т.е. добавена е директория *debian/* с *maintainer scripts*, иначе сорса си е същия като *upstream*, или да си разпакетирате *upstream* архива от kernel.org⁶, т.е. официалния изходен код. Нека да речем, че го вземем от архива на Debian:

```
# apt-get install kernel-source-x.x.x
```

В */usr/src* ще се появи архивът със сорса, който вие трябва да разпакетирате. В резултат ще се появи нова директория */usr/src/kernel-source-x.x.x*, към която е добре да създадете символна връзка:

```
# ln -s /usr/src/kernel-source-x.x.x /usr/src/linux
# cd /usr/src/linux
```

⁵<http://www.kernel.org>

⁶<http://www.kernel.org>

Командата, с която можете да зададете комплексно цялата процедура по конфигурацията и компилацията, е следната:

```
# make-kpkg --config menuconfig kernel_image
```

Така, все едно сте изпълнили едновременно *make menuconfig*, *make dep bzImage modules*. След цялата процедура в `/usr/src` ще се появи *.deb* пакет с новото ядро, който можете да инсталирате по стандартния начин, а това ще ви спести и ровенето в `/etc/lilo.conf`, е разбира се, вие можете да погледнете все пак и в този файл какво е положението. Предишното ядро ще бъде описано със суфикса `.old` и ще можете да го заредите като резервен вариант, ако сте объркали настройките преди компилацията на новото ядро. В случай, че използвате **initrd(4)**, ще трябва да добавите още опция към командата:

```
# make-kpkg --config menuconfig --initrd kernel_image
```

Внимание! Ако ползвате оригиналния сорс, преди цялата процедура трябва създадете стандартната директория `debian/`, без която автоматизацията е немислима.

```
# make-kpkg debian
```

Инсталиране на драйвери за *ALSA* и *NVidia* с *kernel-package*

Нека да компилираме драйверите на *NVidia* и *ALSA* :

```
# apt-get install nvidia-glx-src nvidia-kernel-src alsa-source
```

Добре е да добавите в списъка и пакетите *alsa-base* (задължителен е!), *alsa-utils* и *alsacore* (ако случайно притрябва). При инсталацията на пакета *alsa-source* ще бъдем попитани дали искаме да се компилират всички драйвери или само за конкретна звукова платка.

В `/usr/src` се появява една директория `nvidia-glx-x.x.x`, в която има само поддиректорията `debian/` и нищо друго. Намираме и архив `nvidia-kernel-src.tar.gz`, който след като разархивираме, дава директория `modules`, в която сякаш също няма коя знае какво. Всъщност, това не са самите сорсове. Тези архиви съдържат само информацията, необходима за изтеглянето и компилирането на самите сорсове. А самите сорсове са все още на сървъра на *NVidia*. Те не могат да влязат в състава на дистрибуцията, защото конфликтират с **DFSG**⁷. Не е така с архива `alsa-driver.tar.gz`. Той съдържа необходимите сорсове, който разархивираме, за да ги добави в `/usr/src/modules`.

Следва рутинната процедура:

```
# cd /usr/src/nvidia-glx-x.x.x
# dpkg-buildpackage -us -uc
```

Тази команда дава като резултат *.deb* пакет с *glx* модула на *NVidia*. Обърнете внимание как инсталацията пакет ще се свърже със сървъра на *NVidia*, ще си изтегли сорса на модула и ще го компилира пред очите ви. Същото ще направи и с *kernel* драйвера.

```
# cd /usr/src/linux
# make-kpkg modules_image
```

Това е. След малко ще имате два *.deb* пакета с *ALSA* драйверите за избраната от вас звукова платка и с драйвера на *NVidia*. Инсталирате пакетите по стандартния начин и рестартирате системата или просто само скрипта `/etc/init.d/modutils restart`. Да, рестартирате. Не редактирате никакви конфигурационни файлове, не пипате `/etc/modules` и пр. — за всичко това се е погрижил вече Debian.

Можете да конфигурирате драйверите за *ALSA* (ако вече това не е станало при инсталирането на пакета) с помощта на инструмента **alsacore(1)**, който подобно на **sndconfig(1)** за *OSS* просто редактира `/etc/modules.conf` по индиректен начин, чрез добавяне на информация в `/etc/modutils`. В Debian `/etc/modules.conf` се редактира от `debconf`, и не се препоръчва да се редактира от потребителя, освен ако наистина знае какво прави, за ръчни указания относно заежданите модули за ядото е предвиден `/etc/modules`. Но това вече е друга тема.

Само за *NVidia* остава задължителното и познато на всеки редактиране на `/etc/X11/XF86Config-4`. Там, все пак, **kernel-package(5)** няма власт:)

⁷http://www.debian.org/social_contact#guidelines

15.2. *stow*: Управление на upstream sources

Колкото и голяма да е дадена дистрибуция, то може да се очаква, че не всичко може да бъде пакетирано за нея или в нейния формат към даден момент. Съвсем в реда на нещата е да инсталирате и софтуер който идва просто като upstream tar архив, без каквито и да се maintainer's scripts (обикновено като tar.gz или като tar.bz2 архив). Всичко това естествено може да се прави паралелно с инсталации на пакети които предлага дистрибуцията като всеки трябва да си знае мястото. В такива случаи може да опитате дали ще ви хареса програмата *stow*.

Програмата *stow* е част от проекта GNU⁸ и е пакетирана за много системи. Подобни програми за административни цели обикновено се пишат на Perl както е и в случая с *stow*.

- [Официален сайт](#)⁹
- [Savannah](#)¹⁰
- [FTP достъп](#)¹¹
- [дебиански пакет](#)¹²

upstream авторът (този който е създал софтуера) и debian maintainer за този debian package в случая е един и същи човек. Програмата естествено не е специфична за Debian, може и е добре да се ползва и от други Unix системи. За Debian инсталацията на едноимения пакет *stow* е безкрайно лесна:

```
# apt-get install stow
```

Освен `man stow(8)` може да разгледате и някои статии описващи работата със тази програмка:

- <http://www-106.ibm.com/developerworks/linux/library/l-stow/?ca=dgr-lnxw02STOW> ;
- http://linuxtoday.com/news_story.php3?ltsn=2002-02-08-007-20-PS ;
- <http://www.linuxgazette.com/issue75/peda.html>

⁸<http://www.gnu.org>

⁹<http://www.gnu.org/software/stow/>

¹⁰<http://savannah.gnu.org/projects/stow/>

¹¹<ftp://ftp.gnu.org/pub/gnu/stow/>

¹²<http://packages.debian.org/stow>

Глава 16

Сигурност и надеждност

16.1. Документи и пакети

Обърнете внимание на <http://security.debian.org>.

- [Security Team FAQ¹](#)
- [Securing Debian Manual²](#)

Имайте предвид, че при откриване на проблем в сигурността на даден софтуер Security Team винаги първо ще се погрижи за настоящия Stable Release, ако той е засегнат в дадения случай, като не винаги е задължително да предпочетат upgrade до по-новата upstream версия на софтуера. Много вероятно е старата уязвима версия да бъде закърпена, ако се счете, че промените, адресиращи и поправящи съответната уязвимост, е по-добре да бъдат backported и приложени към старата и уязвима версия. Вижте Security Team FAQ за повече обяснения. След това се гледа дали е засегнат и предишния Stable Release и настоящите Testing и Unstable. В повечето случаи се изкарват почти веднага поправки за всички засегнати версии (или Suites на Debian).

Предвидени са и специални пакети за анализ и оценка на сигурността, някои от които са:

- [harden](#)
- [harden-tools](#)
- FIXМЕ: някой ползвал ли е <http://people.debian.org/~ajt/apt-check-sigs> ?

Ако свалите някой debian source package чрез apt-get source пакет, то ще забележите, че в един от файловете, .dsc, се намира и подписът на съответния debian maintainer. Например:

```
-----BEGIN PGP SIGNED MESSAGE-----
Format: 1.0
Source: sysvinit
Version: 2.84-3
Binary: sysvinit
Maintainer: Miquel van Smoorenburg <miquels@cistron.nl>
Architecture: any
Standards-Version: 3.5.2.0
Files:
 57f11fc13458d8a59894df099449ddb 91130 sysvinit_2.84.orig.tar.gz
 4b90729bfad0c576e8e46250efb65e4d 42387 sysvinit_2.84-3.diff.gz
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org
iQB1AwUBPPNh6FiLscT2F1RZAQGw7QMAk0nUPS/Hsx6V5XD7Cjk54R9C8jvWPRkB
```

¹<http://www.debian.org/security/faq>

²<http://www.debian.org/doc/user-manuals#securing>

```
yxS7/GOpnUWPW9ND517mtNw7E0ZAocZb0oj50wW5PS0y1RIuQuilIaNv0comz0Rv
TACbvUv99j9eAcRTs+qYjnuX8MKTIVO7
=uKkZ
-----END PGP SIGNATURE-----
```

ФИХМЕ: да се обясни повече за Debian keyring, като [подписване на пакетите и проверка](#)³ и ползването от maintainer's на [debsigs](#).

16.2. Прилагане на security updates за множество машини

Пример 11: Настройка на apt-proxy

Ето например как можете да процедурите, ако имате желание да автоматизирате прилагането на security updates към многото ваши Stable Debian машини. Разбира се, ако предпочитате, това можете да правите и ръчно.

`apt-proxy` може да кешира всякакви пакети, но тук ще дадем пример с еднократното изтегляне на security updates за множество машини. Нека имате множество Debian машини, които имат в `/etc/cron.daily/` скрипт, който представлява:

```
#!/bin/bash
apt-get update && apt-get -y upgrade
```

В `/etc/apt/sources.list` трябва да имате следния ред:

```
deb http://security.debian.org/ stable/updates main
```

Представете си, ако излезе update на `glibc`, всичките машини ще теглят доста МВ-ти от [security.debian.org](#).

За да не се хаби излишно трафик, може да ползвате пакета `apt-proxy`. Всичко, което е нужно, е да го инсталирате на един от сървърите си:

```
# apt-get install apt-proxy
```

След което редактирайте `/etc/apt-proxy/apt-proxy.conf`. Ето един:

```
APT_PROXY_CACHE=/var/cache/apt-proxy
add_backend /main/ \
    $APT_PROXY_CACHE/debian/ \
    ftp.us.debian.org::debian/ \
    ftp.de.debian.org::debian/ \
    ftp2.de.debian.org::debian/ \
    ftp.uk.debian.org::debian/
add_backend /non-US/ \
    $APT_PROXY_CACHE/non-US/ \
    ftp.de.debian.org::debian-non-US/ \
    ftp2.de.debian.org::debian-non-US/ \
    ftp.uk.debian.org::debian/non-US/
add_backend /security/ \
    $APT_PROXY_CACHE/security/ \
    security.debian.org::debian-security/ \
    non-us.debian.org::debian-security/
```

Естествено е да сложите най-близкия до вас Mirror на Debian. По подразбиране порта, на който слуша `apt-proxy`, е 9999. Можете да го промените в `/etc/inetd.conf`.

За конфигурация на клиентите, които ще ползват `apt-proxy`-то, е нужно да редактирате `/etc/apt/sources.list`, као коментирате всички редове и да добавите следните:

```
deb http://apt-proxy-сървър:9999/non-US stable/non-US main contrib non-free
deb http://apt-proxy-сървър:9999/security stable/updates main contrib non-free
deb http://apt-proxy-сървър:9999/main stable main non-free contrib
```

³<http://www.debian.org/doc/manuals/securing-debian-howto/ch7.en.html#s-deb-pack-sign>

След това на клиентския компютър е достатъчно да изпълните:

```
# apt-get update
```

При тази конфигурация, при всяко ползване на `apt-get` за инсталация на даден пакет от Интернет самият пакет се пази в кеша на `apt-proxu-то` и при повторна заявка от друг сървър пакета не се тегли наново, а се взема от локалната директория. Наистина спестява доста трафик.

Глава 17

Some Nice Hints and Tricks - Special experience

Тук следват някои по-специфични предложения или възможности, които са (или не са) описани в официалната документация на Debian. Опитът е на разработчици или просто на потребители. Добре ще е първо да сте се запознали с [Ръководството за инсталиране за вашата хардуерна архитектура](#)¹, а също и с базовите инструменти за боравене в Unix среда и в частност Debian GNU/Linux.

17.1. Как да си направим custom installer за Debian Base и LiveCD image

FIXME: е трябва някой да го напише ;-) Базиран на PGI и ако е възможно да се използват модули от разработвания в момента нов debian installer.

17.2. Използване на вашата домашна директория със CVS

[Joey Hess shows you how to keep track of everything with CVS](#)²

На пръв поглед странен начин да се поддържа ред в своя \$HOME. След известна практика може да се окаже, че поне една част е удачно да се пази по такъв начин. Внимавайте с конфиденциалните данни, ако CVS хранилището е публично (включвайте `.cvsignore`, скривайте `.cvspass` ;-)

17.3. chroot-ed Debian

[Colin Walters shows how to chroot your Universal Operating System](#)³

Ако възнамерявате да ползвате голямо количество пакети от Sid + experimental + untrusted unofficial източници, било то и доказано или не, че са лоши такива, които да обновявате изключително често и едновременно с това да сте сигурни, че веднага можете да се върнете към rock solid system, то chrooted система за експерименти в рамките на стабилна система е

¹<http://www.debian.org/releases/stable/installmanual>

²<http://www.linuxjournal.com/article.php?sid=5976>

³<http://people.debian.org/~walters/chroot.html>

най-безопасният и безболезнен начин да си поиграе човек с огъня, и то да го прави с удоволствие, че ще се отърве ненаказан отгоре на това ;-).

17.4. Take Over Installations

- [Official cross-install howto](#)⁴
- [Remote install/convert към Debian изцяло през ssh](#)⁵
- [Chrooted Debian install from base image](#)⁶

17.5. Използване на Debian GNU/Linux при по-необикновени положения

- [Върху Sun Box](#)⁷
- [x86 чрез PXE](#)⁸
- [LVM \(вкл. root filesystem\)](#)⁹
- [Branden Robinson shows Installing Debian 3.0 onto an Apple iBook ... without using any physical media!](#)¹⁰ (fast Internet connection recommended)
- [Sony Vaio SRX87](#)¹¹
- [Running Debian on an Acer Tablet PC](#)¹². Dean Townsley managed to install Debian GNU/Linux on the Acer Travelmate C100 which is a tablet PC that can also act as normal notebook. Anyone who has setup a few systems and compiled their own kernel before should be able to install and set up Debian on such a machine. He [described](#)¹³ in detail how the system is booted from the network and how X needs to be configured in order to support the pen.
- Инсталация на Debian GNU/Linux върху MS X-BOX. Ed's Debian е пълна Debian-базирана Linux дистрибуция, която съдържа последните предимства на X-Box Linux разработването.
 - [Изтегляне на дистрибуцията](#)¹⁴
 - [Как да инсталираме Debian GNU/Linux на Xbox](#)¹⁵
 - За deb-пакетите поставете в `/etc/apt/sources.list` следният ред:

```
deb http://ftp.linux.pt/pub/xbox/debian woody main
```

17.6. Други платформи

⁴<http://www.debian.org/releases/stable/i386/ch-preparing.en.html#s-linux-upgrade>

⁵<http://trilldev.sourceforge.net/files/remotedeb.html>

⁶<http://lists.debian.org/debian-user/2002/debian-user-200204/msg01010.html>

⁷<http://www.tomun.org/docs/sunbox.html>

⁸<http://www.debianplanet.org/node.php?id=818>

⁹http://www.21chouse.com/deb_lvm.htm

¹⁰<http://people.debian.org/~branden/ibook.html>

¹¹<http://www.differentpla.net/~roger/hardware/vaio/linux/>

¹²http://global.acer.com/products/tablet_pc/tmc100.htm

¹³<http://prometheus.physics.ucsb.edu/~dean/TmC100/AcerTmC100.html>

¹⁴http://sourceforge.net/project/showfiles.php?group_id=54192

¹⁵<http://xbox-linux.sourceforge.net/articles.php?aid=2002248060056>

17.6.1. HP PA-RISC

Хардуер

Описаната тук процедура е проведена с:

```

ARCH: HP PA-RISC 2.0
MODEL: 9000/800/A400-44 (Crescendo DC-440)
CPU: PA8500 (PCX-W) 440 MHz (512 KB I-cache, 1024 KB D-cache)
RAM: 1024 MB
ETH0: Digital DS21143 Tulip rev 65
SCSI: 2x sym53c875, 2x sym53c896
SDA: SEAGATE ST336704LCV (36704 MB)
SDB: SEAGATE ST336704LCV (36704 MB)
SDC: SEAGATE ST318404LC (18210 MB)
SR0: HP DVD-ROM 6x/32x

```

Документация

- [Debian for PA-RISC](#)¹⁶
- [The PA-RISC Linux Project Web](#)¹⁷
- [Installing PA-RISC Linux](#)¹⁸
- [PALO PA-RISC/Linux Boot Loader](#)¹⁹
- [PA-RISC/Linux Boot HOWTO](#)²⁰
- [HP Systems Documentation](#)²¹
- [HP PA-RISC Architecture Reference Documents](#)²²
- [The OpenPA Project](#)²³

Начало на инсталацията

[ISO-та на Woody за HPPA](#)²⁴

Дърпате, записвате, боотвате hppa машината и прекъсвате boot процеса:

```

Processor is booting from first available device.
To discontinue, press any key within 10 seconds.
Boot terminated.

```

```

----- Main Menu -----
Command                                     Description
BObot [PRI|ALT|<path>]                       Boot from specified path
PAth [PRI|ALT] [<path>]                       Display or modify a path
SEARch [DIspay|IPL] [<path>]                 Search for boot devices
COntfiguration menu                          Displays or sets boot values
INformation menu                             Displays hardware information
SERvice menu                                 Displays service commands
DIspay                                        Redisplay the current menu
HElp [<menu>|<command>]                       Display help for menu or command
RESEt                                        Restart the system
-----

```

Main Menu: Enter command or menu >

¹⁶<http://www.debian.org/ports/hppa/>

¹⁷<http://parisc-linux.org/>

¹⁸<http://parisc-linux.org/software/install.html>

¹⁹<http://ftp.parisc-linux.org/cgi-bin/cvslite/palo/README.html>

²⁰<http://pateam.esiee.fr/parisc-linux-boot/doc.html>

²¹<http://docs.hp.com/hpux/hw/>

²²<http://h21007.www2.hp.com/dev/>

²³<http://www.openpa.net/>

²⁴ftp://gsyprf10.external.hp.com/debian-cd/3.0_r0/hppa/

Търсим за boot устройства:

```
Main Menu: Enter command or menu > sea
Searching for potential boot device(s)
This may take several minutes.
To discontinue search, press any key (termination may not be immediate).
  Path#   Device Path (dec)   Device Path (mnm)   Device Type
-----
P0       0/0/1/0.1             extscsia.1          Random access media
P1       0/0/1/0.0             extscsia.0          Random access media
P2       0/0/1/1.15            intscsia.15         Random access media
P3       0/0/2/0.1             extscsib.1          Random access media
Main Menu: Enter command or menu >
```

В този случай 0/0/2/0.1 ми е CD-ROMа от който искам да boot-на. Тъй като в Linux наименоването на устройствата започва от най-малкото SCSI ID, горните paths ще бъдат:

```
0/0/1/0.0   /dev/sda
0/0/1/0.1   /dev/sdb
0/0/1/1.15  /dev/sdc
0/0/2/0.1   /dev/sr0
```

Сега е момента да помислите кой диск(ове) ще дадете на Debian, и как ще се виждат под Linux, какви partitions ще има, и дали няма да изтриете някой друг диск по грешка. След което bootваме от CD-то:

```
Main Menu: Enter command or menu > bo p3
Interact with IPL (Y, N, or Cancel)?> n

Booting...
Boot IO Dependent Code (IODC) revision 1
HARD Booted.
palo ipl 1.0 root@palinux Mon Apr 1 10:02:53 MST 2002
Boot image contains:
  0/vmlinux32 3687647 bytes @ 0x649000
  0/vmlinux64 4719374 bytes @ 0x9cd800
  0/ramdisk 2663046 bytes @ 0xe4e000
Information: No console specified on kernel command line. This is normal.
PALO will choose the console currently used by firmware (serial).
Command line for kernel: 'ramdisk_size=8192 root=/dev/ram \
console=ttyS0TERM=vt102 palo_kernel=0/vmlinux'
Selected kernel: /vmlinux from partition 0
Selected ramdisk: /ramdisk from partition 0
Warning: kernel name doesn't end with 32 or 64 -- Guessing... \
Choosing 64-bit kernelELF64 executable
Entry 00100000 first 00100000 n 4
Segment 0 load 00100000 size 2568152 mediaptr 0x1000
Segment 1 load 00374000 size 722104 mediaptr 0x274000
Segment 2 load 00428000 size 463872 mediaptr 0x325000
Segment 3 load 0049c000 size 49152 mediaptr 0x397000
Loading ramdisk 2663046 bytes @ 3fd65000...
Branching to kernel entry point 0x00100000. If this is the last
message you see, you may need to switch your console. This is
a common symptom -- search the FAQ and mailing list at parisc-linux.org
```

След което boot-ва самия кернел. debian-hprra използва за boot loader PALO, а не LILO (както се вижда по горе), и виждаме познатия Debian Installer. (ФИКСМЕ: Проблеми)

Основни конфигурации

Избор на клавиатура няма, тъй като сме през конзола (в моя случай — през GSP), следва partitioning на дисковете. Тук имаме следните моменти:

- Debian GNU/Linux и HP-UX не могат да си делят един и същи диск, така, че ще трябва да заделите отделен за Debian

- по таблицата „Device Path <-> /dev/sd*“ която си направихме в началото, преценяваме с кой диск ще работим (това, че сте в Debian Installer, не значи, че терминала няма scrollback buffer, в който да видите все пак кернела кое как е детектал)
- веднъж след като се инсталирали Debian GNU/Linux във firmware-a на машината, можете на следващия reboot да кажете кой ще е primary boot path-a (от кой диск да bootва по default) (команда Path [PRI|ALT] [<path>] Display or modify a path)
- казахме, използва се PALO за boot loader, поради което имаме следните разлики:
 - PALO боотва кернела директно от Linux (ext2/ext3) дял, няма нужда се gunва palo, когато се променя кернела
 - hp firmware/PALO изискват един служебен partition тип `ef0`, в който ще се съдържат secondary boot loader image и recovery kernel, препоръчва се големина 16MB (аз лично направих моя 32 MB)
 - partition-a който ще съдържа обикновения kernel, който тръгва по default (а не recovery kernel-a), трябва да е във първите 2 GB от диска. И тук имаме два варианта — или `ef0` дяла и root дяла да са в първите 2 GB (което значи, че root ще ни е по-малко от 2 GB), или да имаме отделни root и /boot дялове. Аз лично обикновено си правя /boot дял, препоръчва се обем от около 32 MB.

Като се имат предвид горните забележки, правим си дяловете, и в моя случай накрая имаме следното:

```

                                cfdisk 2.11n
                                Disk Drive: /dev/sda
                                Size: 36703932928 bytes
                                Heads: 64   Sectors per Track: 32   Cylinders: 35003
-----
Name      Flags      Part Type  FS Type   [Label]      Size (MB)
-----
sda1          Primary  Linux/PA-RISC boot      32.51
sda2          Primary  Linux                  32.51
sda3          Primary  Linux                  1024.46
sda5          Logical  Linux swap             2047.87
sda6          Logical  Linux                  1024.46
sda7          Logical  Linux                  2047.87
sda8          Logical  Linux                  2047.87
sda9          Logical  Linux                  28445.77

```

Продължаваме по стандартния начин с debian installer (Initialize and Activate a Swap Partition, Initialize a Linux Partition), докато ги mount-нем всичките. root дялът — първи, лично аз избрах да ги направя ext3 а не ext2.

Довършителни процедури

Продължаваме с инсталацията (основно от CD, българските mirror-и на Debian не съдържат hppa, така че засега upgrade и инсталации могат да се правят само от international mirror-и). След конфигуриране на drivers (ако имате нужда) следва инсталацията на Base System по познатия начин, reboot, base-config, описваме си apt-sources, и аз лично се лишавам както от услугите на dselect, така и от тези на tasksel, като предпочитам нататък да си инсталирам на ръка. Завършваме с base-config-a по традиционния начин.

17.6.2. SGI (netboot)

Хардуер

Описаната тук процедура е проведена с:

```
ARCH: SGI Indy (SGI-IP22)
CPU: MIPS-R4600 V2.0 FPU V2.0 / 100.00 MHz (big endian)
RAM: 61100032 bytes
ETH0: SGI Seeq8003
SCSI0: SGI WD93 WD33c93B/13
SDA: SGI Model: IBMDSAS-3540 (548 MB)
SDB: HP Model: 1.050 GB #A1 (1050 MB)
```

По принцип би трябвало със машини от същия модел (Indy) да няма особенни разлики. Предстоят тестове с още няколко машини — Indy, O2 (FIXME)

SGI INDIGO (моето е с R4000 процесор) за момента не се поддържа (IP-20), но се работи по въпроса, да се надяваме че скоро... :-)

инсталацията засега е на фаза „console“ и не включва подкарване на X (FIXME)

Документация

- [Linux/MIPS HOWTO](#)²⁵
- [SGI Performance Comparisons](#)²⁶
- [SGI Technical Advice and Information](#)²⁷
- [Debian for MIPS](#)²⁸
- [How to install Debian GNU/Linux on Indy \(MIPS\)](#)²⁹
- [Indytech — Hardware technical information for the SGI Indy computer](#)³⁰
- [Google](#)³¹

Начало на инсталацията

Изтеглете [kernel-a](#)³². За съжаление [ftp.bg.debian.org](#)³³ не съдържа **disks-mips**)

boot server: Ще ви трябва още една машина (no matter arch), за да направите инсталацията (моята е Debian в/у i386, т.е. desktop PC-то ми), като на нея в кернела трябва да имате:

```
#
# Networking options
#
CONFIG_PACKET=y
CONFIG_FILTER=y
```

Пак на същата сервизна машина инсталвате *DHCP* и *TFTP* демони:

```
# apt-get install dhcp tftpd
```

По време на power-on boot на Indy-то има един кратък момент в който се вижда едно бутонче „stop for maintenance“. Натискате го и ви излиза системното меню на Indy-то със следните опции:

²⁵<http://howto.linux-mips.org/mips-howto.ps>

²⁶<http://futuretech.mirror.vuurwerk.net/perfcomp.html>

²⁷<http://futuretech.mirror.vuurwerk.net/sgi.html>

²⁸<http://www.debian.org/ports/mips/>

²⁹<http://www.linux-debian.de/howto/debian-mips-woody-install.html>

³⁰<http://www.reputable.com/indytech.html>

³¹<http://www.google.com>

³²<http://ftp.fi.debian.org/debian/dists/woody/main/disks-mips/current/r4k-ip22/tftpboot.img>

³³<http://ftp.bg.debian.org>

```

Start System
Install System Software
Run Diagnostics
Recover System
Enter Command Monitor
Select Keyboard Layout

```

Е, поне на моето Indy са тези, ще видим по другите SGI машини. (FIXME)

Диагностиката винаги е препоръчителна :) но в случая избирате „Enter Command Monitor“ и ще ви излезе един промпт, който предлага някои интересни възможности (вж. „help“)

пишете `printenv` и си записвате MAC адреса на мрежовата карта на Indy-то. В моя случай — `eaddr=08:00:69:08:28:CA`.

Забележка: Ако Indy-то има работеща операционна система преди нашата намеса (примерно някоя версия на **IRIX**), MAC адреса може да се вземе и като се `ping`-не машината и после се разгледа `agr`-таблицата .. въпрос на вкус.

На сервизната машина описваме в `/etc/dhcpd.conf` запис за Indy-то:

```

---
subnet 192.168.1.0 netmask 255.255.255.0 {}
# Entry for Indy-1 (zirakzigil)
host zirakzigil {
    hardware ethernet 08:00:69:08:28:ca;
    fixed-address 192.168.1.5;
    option host-name "zirakzigil";
    option domain-name-servers 192.168.1.1;
    option routers 192.168.1.1;
}
---
```

Като параметрите имат следния смисъл:

- *hardware ethernet* е MAC адреса на Indy-то
- *fixed-address* е IP адреса който ще има Indy-то
- *option host-name* е `hostname`-а на Indy-то
- *option domain-name-servers* е адреса на DNS сървъра който ще ползва Indy-то
- *option routers* е адреса на `gateway`-а на Indy-то.

За повече подробности вижте в `dhcpd.conf(8)`.

Забележка: В случая съм използвал фалшиви IP адреси, но това не значи, че не могат да се ползват и истински. И в двата случая обаче е добре да има изградена мрежова структура (т.е. на IP-то на DNS-а наистина да има DNS, и `gateway`-я да работи като `gateway` :)

Изпълнявате на сервизната машина:

```
# echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc
```

за подробности: [kernel.org/Documentation/filesystems/proc.txt](http://kernel.org/doc/Documentation/filesystems/proc.txt), [lkml](http://kernel.org/doc/Documentation/filesystems/lkml), [google](http://www.google.com)³⁴

След което си пускате DHCP сървъра:

```
# /etc/init.d/dhcpd start
```

Забележка: Ако в мрежата имате друг DHCP сървър, бъдете осторожни :)

Следваща стъпка: редактирате си `/etc/inetd.conf` файла и добавяте нещо от рода на:

```
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /tftpboot
```

Като `/tftpboot` е директорията, в която ще седят `boot-image`-ите (в нашия случай за Indy-то). Подробности: `inetd.conf(8)`, `tftpd(8)`, `tftp(1)`, `inetd(8)`.

След което (ре)стартирате `inetd`:

³⁴<http://www.google.com>

```
# /etc/init.d/inetd stop
# /etc/init.d/inetd start
```

или

```
# kill -HUP `ps aux |grep inetd |grep -v grep |awk '{print $2}'`
```

Ако трябва да рестартирате inetd на машина, която не е debian.. но след като сте тръгнали да четете тази документация, предполага се, че сте Debian Zealot :-)

Забележка: Лично мнение: с изключение на случаи като този (примерно инсталация на машина, която се нуждае от tftpd или някаква друга специфична inetd услуга) по-добре е да нямате пуснат inetd. Един service по-малко, малко по-спокоен съм :) Все пак, въпрос на личен избор и конкретна ситуация.

Копирате кернела за Indy-то в /tftpboot.

Обратно на Indy Command Monitor промпта пишете:

```
# setenv netaddr $IP
```

Като на мястото на \$IP слагате IP-то, което трябва да има Indy-то. В моя случай: 192.168.1.5.

```
bootp() /tftpboot/tftpboot.img
```

натискате Enter и сте в бизнеса :)

Забележка: добре е да хвърлите един поглед на /var/log/messages за възможни грешки, ако нещо не е наред, да проверите какви са permission-ите на /tftpboot и /usr/sbin/tcpd.

Основни конфигурации

След като кернелът вече е тръгнал и гледаме debian-installer-а на екрана, инсталацията продължава почти като на i386 архитектура. Можете да спрете вече dhcpd и inetd сървърите, няма да ви трябват повече.

Следва избор на клавиатура и partitioning на диска или дисковете. Лично аз съм фен на класическия вариант с command-line fdisk(8), така че отиваме на втората конзола (Alt-F2) и започваме:

Едно бързо dmesg, за да видим какво си е намерил кернела. Ако има нещо важно за процеса на инсталация, което е изпуснато (хардове, мрежова карта, etc.), ще трябва да си търсим или правим друг boot image, което ще го опишем в някоя от следващите версии (FIXME).

В моя случай съм малко зле с обема на дисковете (1x 500MB и 1x 1GB) така че ще разхвърляме различните partition-и на двата диска. След като изтрием (по стандартния за fdisk начин :) всички дялове останали от добрия стар IRIX, имаме нещо от рода на:

```
# fdisk -l /dev/sda
Disk /dev/sda (SGI disk label): 3 heads, 108 sectors, 3314 cylinders
Units = cylinders of 324 * 512 bytes
---- partitions ----
---- Bootinfo -----
Bootfile: /unix
---- Directory Entries ----
[...other info here... :-)]
#
```

тука идва tricky part: Indy-то е *Big-Endian* машина, така че нещата са малко по-различни от добрия стар i386 (BIOS, MBR, partition table) и в частност ще трябва да направим *disklabel*.

Първо една бърза сметка: 3 heads * 108 sectors * 3314 cylinders * 512 bytes = 549752832 bytes (524MB), колкото е обема на първия ми диск. Направете два дяла (root+swap) с големина по ваш избор, те не са важни, ще ги използваме само за пример. Ако се съгласите с default-ната стойност за начален цилиндър на първия дял (т.е. 5-ти цилиндър) и след това си направите още един дял, освен тези sda1 и sda2, ще имаме още два записа в „partition“ секцията на изхода от командата p на fdisk-a, а примерно:

Pt#	Device	Info	Start	End	Sectors	Id	System
9:	/dev/sda3		0	4	1620	0	SGI volhdr
11:	/dev/sda4		0	3313	1073736	6	SGI volume

- *SGI volume* дяла представлява целия харддиск. В моя случай харда има 3314 цилиндъра, номерирани от 0 до 3313, и /dev/sda4 го обхваща целия (1073736 сектора).
- *SGI volhdr* дяла (volume header) е сервизен partition за disklabel-a, и въпреки че е sda3 в нашия случай, физически той се намира в началото на диска (цилиндри от 0 до 4). Ако искате, можете да мислите за него като за един голям MBR :) Интересното е, че в него се записва кернела на машината и firmware-a го чете от там за да boot-не, което означава, че все пак трябва да е достатъчно голям, за да може да събере кернела в себе си. В по горния пример volume header-a е 1620 sectors * 512 bytes - 829440 bytes (810 KB), което е леко недостатъчно. Ако изберем 16MB за volume header дяла, това прави: $16 * 1024 * 1024 = 16777216$ bytes, или 32768 сектора, и тъй като в един цилиндър имаме 324 сектора, $32768 / 324 = 101$ цилиндъра трябва да е голям този дял.

Приятният момент е, че не трябва да правите на ръка тези два дяла. fdisk (донякъде) ще ги направи вместо вас. Изтриваме всички partition-и от таблицата и започваме наново:

n за нов partition и като ви попита за начален цилиндър на дяла, кажете 101. По този начин цилиндри от 0 до 100 ще бъдат заделени за volume header-a, което са си точно 101 цилиндъра. От там нататък създаваме този дял (sda1) по стандартния начин, като имате предвид че това трябва да ви бъде Linux native дяла (root), и чак след него трябва да е swap дяла. В моя случай на първия ми диск (524MB) нямам много място, така че ще се ограничи само с root+swap. Тук е момента за още малко математика:

Имам около 64 MB RAM и искам да си направя swap дял 128 MB. (Доброто старо правило swap-a да е 1x или 2x обема на паметта. Да не говорим че в някои други UNIX-и е почти задължително swap-a да е минимум колкото паметта на машината, иначе ако параметъра swaptm_on е сетнат на 0-a, което осигурява директен mapping м/у паметта и swapa, за да се избегнат по-тежките операции по адресиране на паметта и за да се спести памет от т.нар. псевдо swap, OS-a ще ползва толкова RAM, колкото е голям swap дяла. Long story short: правете си swap дяла поне два пъти колкото RAM-a, изгражда полезни навици.

И така: 128 MB = 134217728 bytes = 262144 sector-a ни трябва за този дял. $262144 / 324$ сектора на цилиндър = 809 цилиндъра. Дискът ни има 3314 цилиндъра и ако сложим swap дяла последен, той ще заема цилиндри от 2505 до 3314, което означава, че root дяла трябва да е до 2504-ти цилиндър, което въвеждаме на промпта на fdisk, който още ни чака :)

Следваща стъпка: изтриваме sda9 (volume header-a) и го правим наново пак като sda9 (9-ти partition) с цилиндри от 0 до 100. След това създаваме sda2 със цилиндри от 2505 до 3314, така че накрая имаме:

Pt#	Device	Info	Start	End	Sectors	Id	System
1:	/dev/sda1	boot	101	2504	778896	83	Linux native
2:	/dev/sda2	swap	2505	3313	262116	82	Linux swap
9:	/dev/sda3		0	100	32724	0	SGI volhdr
11:	/dev/sda4		0	3313	1073736	6	SGI volume

Лека проверка с v (нямаме неалокирани сектори), w (записваме таблицата на диска и излизаме). В моя случай продължавам с fdisk /dev/sdb, тъй като искам да си направя дялове за /var, /usr и /home. Тъй като този диск не е boot-able, тук нещата са по-прости: създавам си три дяла набързо и имам:

```
# fdisk -l /dev/hdb
Disk /dev/sdb: 33 heads, 61 sectors, 1019 cylinders
Units = cylinders of 2013 * 512 bytes
Device      Boot  Start  End  Blocks id System
-----
/dev/sdb1   1      261   262666 83 Linux      (за /home)
/dev/sdb2   262    522   262696 83 Linux      (за /var)
/dev/sdb3   523    1019  500230 83 Linux      (за /usr)
```

Записваме таблицата с w и се връщаме обратно на първа конзола (Alt-F1).

ВНИМАНИЕ: Описаните по-горе стойности на цилиндри, глави, сектори и така нататък се отнасят конкретно за моите дискове и служат само за пример как трябва да проведете нещата при

вас. С изключение на случая, когато имате **точно** същите дискове (много малко вероятно), тези стойности са напълно неприложими за вашата инсталация, и **трябва** да пресметнете всичко спрямо **ваши**те параметри.

Продължаваме инсталацията по нормалния начин (указвайки кои са ни root и swap дяловете). Инсталационната програма конкретно при мен не поиска да си намери дяловете на /dev/sdb, така че пак във втора конзола им направих файлови системи (ext3, както и root дяла), дори и така не пожела да ги mount-не през инсталера. Конкретно аз се отказах да пробвам да ги mount-на на ръка в /target, където им е мястото, и оставих тази част за после, инсталирайки само в sda1.

Довършителни процедури

Следващата стъпка е инсталация на kernel върху новата машина: избираме „from internet“ (FIXME: описание на инсталация от NFS). Настройваме „network settings“ (или си пускаме пак DHCP сървър на сервизната машина и му казваме да ползва DHCP). Въпрос на вкус и възможности е от къде ще дръпне kernela и драйверите, аз се съгласих с default-ната стойност <http://http.us.debian.org:80>. Добрата новина тук е, че <http://ftp.bg.debian.org> съдържа [binary-mips³⁵](#), така че нататък инсталацията ще върви по-бързичко :) За съжаление другият Debian mirror в България, <http://debian.ludost.net>, засега mirrorва само i386 и source. След конфигуриране на drivers (ако имате нужда) следва инсталацията на Base System по познатия начин от <http://ftp.bg.debian.org/debian>, като не трябва да се забравя проху-то (а тъй като аз имам машина с **apt-proxy**(8) и това е второто Indy което инсталирам, нещата вървят доста бързичко — въпрос на network setup и bandwidth, както винаги)

Следват `make system bootable`, и още малко настройки в Command Monitor-a на Indy-то:

```
setenv OSLoader linux
setenv SystemPartition scsi(0)disk(X)rdisk(0)partition(8)
setenv OSLoadPartition /dev/sda1
```

Където X е SCSI id-то на /dev/sda диска (това ще ви го каже накрая инсталационната програма).

Чрез едно `dmesg`(8) на втората конзола преди да reboot-нем виждаме (в моя случай):

```
Attached scsi disk sda at scsi0, channel 0, id 1, lun 0
```

Така че при мен настройките трябва да са:

```
setenv OSLoader linux
setenv SystemPartition scsi(0)disk(1)rdisk(0)partition(8)
setenv OSLoadPartition /dev/sda1
```

Boot-ваме, следва **base-config**(8), т.е. процедираме по стандартния начин за Debian, описваме си apt-sources, и аз лично се лишавам както от услугите на dselect, така и от тези на tasksel, като предпочитам нататък да си инсталирам на ръка. Завършваме с base-config-a по традиционния начин.

Препратки:

- Някои „хитрости“ от Рик Моен³⁶

³⁵<http://ftp.bg.debian.org/debian/dists/woody/main/binary-mips/>

³⁶<http://www.linuxmafia.com/debian/tips>

17.6.3. Mosix, OpenMosix и други

Разбира се, клъстерните технологии не са специфични за Debian, но понеже има подобни пакети, включени в архива на Debian, а както и реални примери на работещи клъстери, изградени от Debian GNU/Linux машини, ще споменем и тази тема.

Ето няколко примера:

- **Debian Beowulf Project**³⁷. На страницата **с потребителите на Debian**³⁸ можете да забележите доста примери на действащи клъстерни конфигурации с Debian GNU/Linux (от 2 до 256 nodes, а вероятно и повече)
- Широко прилагани и използвани в университетските и академичните среди:
 - **Artificial Intelligence Lab, Massachusetts Institute of Technology, USA**³⁹: In the MIT AI lab the „Officially Supported“ flavor of GNU/Linux is Debian. There are approximately 100 user workstations running Debian, as well as a small but growing **OpenMosix Cluster**⁴⁰ serving compute cycles. The switch to Debian was made official in September 2001. The largest reason being the ease of package management, which users love (users here get that kind of freedom) and also allows the sysadmins to semi-automate security updates (we like to check the updates on a test system before forcing them on the users, but to date there’s been no problems).
 - **Doshisha University, Japan**⁴¹: At Doshisha University, Debian „potato“ runs on **256-node Beowulf cluster**⁴². They are arranged on 16-node computer groups which have 1 diskfull and 15 diskless machines. Maintenance with Debian installed has been a pleasure with security updates being only an apt-get away, and the updates being very prompt.
- **Също така**⁴³ и в големи компютърни центрове като **SARA, Netherlands**⁴⁴
 - 168 nodes IA-32 Beowulf клъстер използван в университета на Амстердам.
 - 16 nodes Alpha клъстер използвам от различни потребители.
 - 2 nodes (4 CPU на машина) IA-64 клъстер, за пренасяне на 32-битови приложения към 64-битови.
 - 9 nodes IA-32 клъстер за тестване на приложения и инсталационни методи
- **Qli Linux Clusters**⁴⁵: Дори се и продават готови преинсталирани с Debian и Red Hat машини обединени в клъстер. Става въпрос за маниши от добре познатата ни x86 (IA-32) архитектура с процесори на Intel Xeon и AMD Athlon.

Тук ми се ще да дам и един такъв пример леко встрани. Не малко хора употребяват и свързват думата Enterprise само и единствено с т.н. Commercial OS vendors или по-точно със системите, които се предоставят от тях. Не винаги става ясно в какъв смисъл се употребява думата Enterprise, но почти винаги като че ли се има предвид използването на някаква технология за clustering, или пък дадена mission-critical задача. Доста едностранчиво и праволинейно е да се мисли, че това е единственият и неповторим подход. Нека погледнем от другата страна — Debian и Gentoo например са отявлени представители на т.н. некомерсиални General-purpose Distributions. Общото е, че те не се продават срещу заплащане, и проектите са sponsored by companies, а не owned by companies. Ето една статия от три части

- <http://www-1.ibm.com/servers/esdd/articles/openmosix.html?t=gr,lnxw02=OpenMosix1>
- <http://www-1.ibm.com/servers/esdd/articles/openmosixpart2.html?t=gr,lnxw02=OpenMozix2>

³⁷<http://www.debian.org/ports/beowulf/>

³⁸<http://www.debian.org/users/>

³⁹<http://www.ai.mit.edu/>

⁴⁰<http://www.ai.mit.edu/sysadmin/cluster.html>

⁴¹<http://www.doshisha.ac.jp/>

⁴²cambria.doshisha.ac.jp

⁴³<http://www.sara.nl/beowulf/>

⁴⁴<http://www.sara.nl>

⁴⁵<http://www.qfilinuxpc.com/products/clustering/>

– <http://www-1.ibm.com/servers/esdd/articles/openmosixpart3.html?t=gr,lnxw02=OpenMosix3>

която може да разясни някои недоразумения в този дух. В случая се визира clustering чрез **OpenMosix**⁴⁶ и примера е даден с Gentoo, но спокойно можете да си мислите същото и за Debian, защото Mosix (**kernel patches**⁴⁷ and **userland tools**⁴⁸), а впоследствие и OpenMosix (**kernel patches**⁴⁹ and **userland tools**⁵⁰) присъстват отдавна в неговия архив, но дори и да не присъстваха нямаше да бъде болка за умирање — на който му трябва clustering най-вероятно ще знае как да използва този софтуер и в upstream вид. Забележете каква е и историята на самия **OpenMosix проект**⁵¹ защо е GPL'ed, и защо е било необходимо да се основава той, наследявайки своя предшественик **Mosix**⁵². Въпросът не опира до каквато и да е религиозна или идеологическа основа, въпросът не е кой е по-комерсиален или по-неутрален в материалния смисъл на думите, въпросът опира чисто и просто само до техническото съвършенство на софтуера и възможностите за постигане на такова. Запазвайки колкото е възможно по-голям неутралитет, т.е. без каквито и да са притеснения как ще се продава това или онова, възможностите за техническо усъвършенстване се увеличават. Проектът Debian, а както и самия Linux kernel, са блестящи примери за запазване на собствен неутралитет. Както казва и самия Linus Torvalds, дори и той да бъде „купен“ от някоя компания, не е възможно това да стане с Linux. Създателят на проекта Debian впоследствие напусна този проект и започна да развива своя комерсиална дейност, което си е абсолютно нормално и в реда на нещата, но самият проект Debian винаги ще си остане неутрален и некомерсиален, поради гореспоменатите причини, иначе няма да е Debian. Много ползватели на свободен и/или отворен софтуер го ползват и защитават само поради идеологически и религиозни причини, несъзнавайки къде точно се крие неговото превъзходство, а това е много много жалко. Софтуерът не е основа за водене на безсмислени идеологически войни на тема „кой е по-велик“, софтуерът е за да върши някаква полезна работа. Аз например не ползвам Debian и Linux (e.g. Debian GNU/Linux), защото съм чул, че са безплатни (т.е. получават се без заплащане), ексцентрични и модни. Напротив, ползвам ги поради техническите им качества, последните оценявам технически, а не идеологически или религиозно, което хич не е сериозно да се прави. Естествено, че нямам нищо против да бъда убеден от даден Commercial OS vendor, че съществуват и технически по-съвършени аналози за General Computing usage, винаги съм готов да слушам, но технически аргументи, а не сухи рекламни трикове подвеждащи потребителя.

⁴⁶<http://www.openmosix.org>

⁴⁷<http://packages.debian.org/kernel-patch-mosix>

⁴⁸<http://packages.debian.org/mosix>

⁴⁹<http://packages.debian.org/kernel-patch-open-mosix>

⁵⁰<http://packages.debian.org/openmosix>

⁵¹<http://www.openmosix.org>

⁵²<http://www.mosix.com>

Глава 18

Анализи, оценки, предложения

Малко по-задълбочени сравнения и анализи има в [Advanced Package Management Comparisons](#)¹.

Debian/GNU Linux: [The Past, the Present and the Future](#)² — presented at the Free Software Symposium 2002 on October 22, 2002 15:30 at the University of Tokyo.

Като, че ли е пропуснато да се обясни малко повече за работа с `debian source packages` (FIXME: да се обясни още по-подробно):

- уникални инструменти като `auto-apt`, различни проверки с `dh_*`, като `dh_shlibdeps`, или направо проверете какво съдържат пакетите `dpkg-dev` и `debhelper`, разгледайте сorsa и документацията на съответните програми, идващи с тях.
- създаване на *local apt repository*, и подаване на *custom build options* глобално или поотделно за пакет (`DEB_BUILD_OPTIONS`, `apt-build`, `apt-src`, `chrooted builds` с `pbuilder`). Това са все важни неща, като се има предвид, че потребителя в общия случай ще иска (или му се налага) да има пакети с различни версии на дистрибуцията към момента. Може да са нужни и доста по-стари версии на пакети от <http://snapshot.debian.net>, както и следене на последните версии на всичко (за Debian това е Sid архива или пък `project/experimental`). Това не винаги е разумно решение, като освен това може да е добавено и *local repository* или въобще *unofficial repositories*, т.е. контролира се процеса на `build` от началото до края. Например `auto-apt` помага за:
 - откриването на липсващи хедъри и библиотеки, необходими за компилацията, свързането и намирането им в кой(и) пакет(и) са и предложение за инсталиране
 - подаване или установяване на *custom build options*
 - различни проверки с `dh_*`, особено проверка за коректност на получения *binary package*, в който най-вероятно ще има програми, които ще ползват поделени библиотеки (`soname check` и [Debian Library Packaging guide](#)³)

Подобен контрол е необходим и ми се струва уникален Debian при `builds`, за да не бъде `build` процеса „чуплив“, т.е. да прекъсва поради липси на това и онова, особено когато се предприема в смесена среда (напр. пакети от няколко издания на дистрибуцията). Трябва да има гаранции, че така полученото `binary` ще работи коректно. Ако има синтактични и/или логически програмни грешки в кода на приложението, естествено, че или вие трябва да ги оправите, или разработчиците. Така че е необходимо да има гъвкавост и потребителят да разполага с инструменти бързо и лесно да установява докъде се простира тя и от къде нататък започват да че чупят нещата и защо. Разбира се, това не пречи в `/usr/local/` да водите война, инсталирайки каквото ви дойде на ума, без да разваляте нещата *system-wide*, пък може да пипате и там, ако се чувствате сигурни, че знаете какво правите.

¹<http://u-os.org/upm.html>

²<http://u-os.org/tokyo.html>

³<http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html#AEN29>

Нова система за *build* на пакети се готви от Colin Walters. Тя се нарича [версия 2](#)⁴ на source пакетите и в нея се адресират сложността на сегашните `textttdebian/rules` файлове, както и многото излишък в тях. Новата система е силно повлияна от *и-ос* [пакетната система](#)⁵ на Christoph Lameter. Основната идея е да се направят нещата прости, а сложните неща — възможни.

Вече като малко по-напреднали трябва да четете:

- [Maint Guide](#)⁶ или [стария превод на български](#)⁷
- [Debian Developer's Reference](#)⁸
- [Debian Policy Manual](#)⁹
- и въобще [Debian Developers' Corner](#)¹⁰

⁴<http://lists.debian.org/debian-devel-0211/msg02630.html>

⁵<http://www.u-os.org/upm.html>

⁶<http://www.debian.org/doc/maint-guide/>

⁷<http://debian.gabrovo.com/docs/maint-guide/index.html>

⁸<http://www.debian.org/doc/developers-reference/>

⁹<http://www.debian.org/doc/debian-policy/>

¹⁰<http://www.debian.org/devel/>

Глава 19

More

Fwd: Re: lug-bg: Gentoo ebuilds¹ Не предоставя механизъм за надеждното използване на различни версии на софтуера предлаган в различните издания на дистрибуцията, т.е. изчакваме, например, за смислен аналог на проверка от вида

```
# apt-get install p1/stable p2/testing p3/unstable
```

За Analysis of library dependencies to insure accuracy of runtime dependencies и не споменаваме; 4–5 пъти по малък архив и горе долу толкова пъти по-малко на брой поддържани хардуерни архитектури.

Тези, които имат желание за собствени биулдове на Debian, да пообърнат внимание на [apt-build](#), [apt-src](#), [pbuilder](#), [auto-apt](#) и т.н.

Тук май е мястото да спомена, че критериите, които посочихме в началото, изпълняват и свободни операционни системи като [FreeBSD²](#), [NetBSD³](#) и [OpenBSD⁴](#). Много добра идея ще е да се погледне и как те могат да ви бъдат от полза. Специално начинаещите потребители могат да прочетат и превода на [FreeBSD Packages & Ports Collection⁵](#), въпреки че не е цялостно описание как се доставя и инсталира нов софтуер във вашата системата, а визира само Third Party software, но като за начало става. Не посмях да превода или да давам съвети как се upgrade-ва една такава система (имайки предвид всичките ѝ части), понеже както и при Gentoo, и тук няма определен механизъм за частичен или смесен upgrade на отделни части от системата (Base and Third Party) и единствено се взима под внимание и гарантира с успех цялостен upgrade на сорсовете на системата (вижте [Synchronizing Your Source⁶](#)) или евентуален binary upgrade (не се поддържат, например няма Conflicts, както при Debian).

Всички тези работи може би разработчиците и по-напредналите потребители ги знаят, имайки предвид статистиката за [Which of the following is your favorite distribution / operating system?⁷](#) от окончателното изследване и проучване [FLOSS⁸](#) на Международния Институт по Информатика към Университета на Маастрихт (Холандия) в партньорство с Berlecon Research, Berlin и Proactive International (Paris). Данните са приети от [European Commision⁹](#). Та защо и по-новите потребители да не се възползват от всичко това, отгърсвайки се от неправилни предубеждения и схващания, че дистрибуцията не е подходяща или не може да бъде удобна и за тях? Напротив, дистрибуцията учи на правилно използване и прилагане на софтуера.

¹<http://linux-bulgaria.org/lug-bg-list/archive/2002/Oct/0308.html>

²<http://www.freebsd.org>

³<http://www.netbsd.org>

⁴<http://www.openbsd.org>

⁵<http://www.freebsd-bg.org/html-br-up/>

⁶http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/synching.html

⁷http://floss1.infonomics.nl/floss1/stats_13.html

⁸<http://www.infonomics.nl/FLOSS/report/>

⁹http://www.europa.eu.int/comm/index_en.htm

Глава 20

Погрешно схващане

Доста погрешно схващане е, главно сред тези, които не са се докосвали до Debian, че щом операционната система се изготвя от некомерсиална организация, то това е неин недостатък, главно поради това, че няма конкретно лице, което да поеме персонална отговорност при евентуални проблеми със софтуера. Първо, **безплатна**¹, а така и **комерсиална**² поддръжка има. Комерсиалната е ясна — лице или компания се заема с поддръжката и гарантира, че всичко ще е както сте се договорили, че трябва да бъде, и при проблеми следват някакви взаимоотношения между вас, както е навсякъде. По-важното е, че безплатната поддръжка се осъществява и идва съвсем естествено до потребителите чрез многобройните пощенски списъци, новинарски групи, пряко взаимодействие с разработчиците и т.н, и всичко това е в процеса на разработка. Разбира се, не всички имат времето, знанията и уменията да се възползват от всичко това, но това е един прекрасен източник за придобиване на знанията и опита на другите, по-опитните и по-напредналите. Получава се точно обратното, главното предимство на организацията е, че е некомерсиална, и съответно операционната система е такава, т.е. изключително много се държи на техническия аспект на нещата, без влияние на каквито и да са външни странични фактори, които могат да отклонят нещата от чисто техническия нюанс. Това може би е и поради много тясната връзка с академичните среди, които се интересуват главно от техническата част на нещата. Това от своя страна води до налагане на много високо техническо ниво и задълбоченост, трудно достижимо от редица комерсиални аналози. На практика направо трябва да се погледне **списъка с регистриралите се потребители**³ (добавени там по своя инициатива), сред които има научни институции (изключително висока популярност), комерсиални организации, опганизации с идеални цели, а както и правителствени такива. Обърнете внимание на мотивите, които изтъкват за ползване на системата, а както и за какви цели се ползва.

¹<http://www.debian.org/support>

²<http://www.debian.org/consultants/>

³<http://www.bg.debian.org/users/>

Глава 21

Обобщение

Надявам се ви се поизясниха повечето от представените по-горе неща. Ето и една таблица, схематично представяща какво може да съдържа вашият Debian:

Debianized (<i>debian source & binary packages</i>)					Non-Debianized (<i>just upstream sources</i>)
Official .deb's <i>ftp.debian.org and mirrors</i>				Unofficials .deb's <i>www.apf-get.org</i>	
	Stable	Testing	Unstable	project/experimental	
Настоящо състояние на Official	Изключително консервативен release.	Ако Stable не ви стига точите и от тук.	Ако Testing не ви стига точите и от тук.	С това малко по-внимателно.	Изключително много рядко и трудно ще се разминете с успешен build от upstream sources (от скромност не казвам, че няма как да се разминете). Тук посочвам auto-art който изнамира в кой debian package са необходимите и липсващи към момента в системата на потребителя headers/libs необходими за компилацията и свързаното ... Заслугата не е само на auto-art, заслугата е на цялостния дизайн на debian packaging style който позволява това да бъде имплементирано в една такава сравнително скромна скрипт ui-ка, която изглежда, че прави чудеса, а такива няма ...
Стари моменти състояния на Official	<i>(само fast & safe security updates и point releases).</i>	<i>(more fancy stuff..)</i>	<i>(even more fancy stuff..)</i>	<i>(even more eventually dangerous stuff..)</i>	
	Минали моменти състояния на ftp.debian.org (и mirrors) snapshot.debian.org (snapshots на предишни състояния на official – stable, testing, unstable, project/experimental)				В общия случай като структура това са частни случаи на official. Ако все пак не намирате това което ви трябва в рамките на official, преглеждате какво може да притеглите от различните unofficial repositories.

Ползването само на компилирани пакети от Stable (със security updates), както се вижда от таблицата, е изключително скромно и непретенциозно поведение от страна на потребителя, най-вероятно понеже потребителят не е достатъчно информиран, че има и други блага, от които може да избира и да миксира. Така че няма смисъл от излишна скромност, ако се налага не се стеснявайте да точите, откъдето намерите и сметнете за добре. Системата е достатъчно гъвкава и не е нужно да инсталирате всичко от дадено издание, така че за да работи коректно, спокойно може да миксирате от няколко издания, т.е. да инсталирате само това, което искате, като ще бъдете уведомени за това, от което то зависи или конфликттира. От тук идва и голямата надеждност. Ако мислите, че нещо трябва да е по друг начин, който на вас по ви изнася, естествено, че ще се намесите компетентно ;-). Нещата не винаги са толкова прости, колкото изглеждат, така че преди да пипате клавиатурата, помислете защо така е било направено.

Накрая мисля, че е уместно да цитираме Christoph Lameter: „Do not ask me: Does Debian support this and that. Debian supports everything.“

Christoph Lameter is a faculty member of the University of Phoenix. He is currently teaching graduate and undergraduate classes at the San Jose extension of the University of Phoenix. Subjects taught vary from Critical Thinking (PHL251) to Introduction to Unix (POS/420), Programming Concepts using C++ (POS/370), Networks and Telecommunications (NTC/410) to classes on management of programming teams Managing Programming (CMGT/576). Christoph has been a developer of the Debian Project since 1996. In 1997 Christoph was elected to the Board of Directors of the Debian Project. Christoph has contributed more than 150 packages to Debian among those alien (Package format converter), deb-make (rapid packaging tool, precursor to helper). Since 1994 Christoph has contributed to the Open Source community various patches to utilities and the Linux kernel as well as started some open source projects. Christoph has been speaking at various conferences on technical and nontechnical issues since 1999. Christoph is serving as the representative of the Debian Project on the Advisory Council of the Linux Professional Institute since 2000. Christoph is currently working on a temporary basis at the

21 Обобщение

University of Phoenix and is looking for permanent opportunities for teaching and/or employment. You can reach Christoph at Christoph@Lameter.com. Academic Credentials: Master of Computer Science, University of Bremen, Germany, 1986 with a thesis on Compiler construction titled A transpiler from ADA to Pascal using LALR(1)-Syntax transformation. Master of Divinity, Fuller Theological Seminary, Pasadena, California, 1994. Associate Fellow of CRIS (Azusa Pacific University, California), 1999. Ph.D. Candidate, Fuller Theological Seminary, 2003(?). Divine action in the context of Scientific Thinking: From Quantum Mechanics to Divine Action.

Аз бих добавил: „Може и нескромно да звучи, но като че ли не много (комерсиални) анализи биха издържали на темпото, налагано от Debian.“

Част V

Често задавани въпроси за Debian

Въпроси, които се повтарят отново и отново или такива, които винаги сте искали да зададете, но не сте задали за да не ви помислят за *lamer* ;-)

– **V1: Какво все пак означава наименованието Debian (Дебиан)?**

O1: Дебиан е проект, започнат през 1993-та година от Иан Мърдок. Името Дебиан идва от Деб + Иан (Деб идва от Дебора, съпругата на Иан Мърдок)

– **V2: Кои сървъри са най-подходящи за включване в /etc/apt/sources.list за България?**

O2: ftp.bg.debian.org или debian.ludost.net. Например:

```
deb http://ftp.bg.debian.org/debian stable main contrib non-free
deb http://ftp.bg.debian.org/debian-non-US stable/non-US main contrib non-free
deb-src http://ftp.bg.debian.org/debian stable main contrib non-free
deb-src http://ftp.bg.debian.org/debian-non-US stable/non-US main contrib non-free
```

Препратка: http://www.debian.org/mirrors/mirrors_full#BG

– **V3: Има ли пощенски списъци (mailing lists) за Debian в България?**

O3: Да, има такъв¹ в <http://debian.gabrovo.com>.

– **V4: Има ли свързани с Debian български софтуерни проекти?**

O4: Погледнете проекта [bgdebian](#)².

– **V5: Къде в България мога да си закупя Debian CD, книги, програми и т.н**

O5: Дистрибуционни CD-та на Debian можете да си закупите от [CDforME](#)³. Книги, CD-та и други материали, могат също така да се закупят от каталога на [Amazon.com](#)⁴, като съответно съществува по-приемлив вариант за разплащане и доставка в България⁵ — фирмата срещу съответен процент доставя стоката по ISBN номер.

– **V6: Необходимо ли е да си изтегля (download) всичките 5/7/9 Debian CD-та?**

O6: Ако разполагате с връзка към Интернет, достатъчно е да имате само първото CD. Оттам може да се инсталира минималната система, достатъчна да довършите мрежова инсталацията (да свалите всичко останало от Интернет)

– **V7: Защо при apt-get update получавам грешка от вида:**

```
Reading Package Lists... Error!
E: Dynamic MMap ran out of room
```

O7: Списъка с Debian пакети за инсталиране е твърде голям (дълъг лист от сървъри в /etc/apt/sources.list) и apt няма достатъчно памет за да го обработи. Трябва да разрешите на apt да ползва повече памет. За целта коригирайте стойността на Cache-Limit в /etc/apt/apt.conf:

```
APT::Cache-Limit 25165824
```

– **V8: Как мога да премина от KDE2 към KDE3 без пълно преинсталиране?**

O8: <http://lists.debian.org/debian-kde/2002/debian-kde-200210/msg00073.html>

– **V9: А някакви журнални файлови системи да се ползват в Debian?**

O9: Може би е добре да започнете от тук: <http://people.debian.org/~blade/>

– **V10: Има ли начини за автоматично разпознаване и конфигуриране на хардуера?**

O10: За целта има програми като *discover* и *kudzu*, които се ползват и в други дистрибуции. Knoppix LiveCD, например, ползва собствени конфигуриращи хардуера скриптове заедно с модула *cloop*, който вече е в официалния Debian архив благодарение на Klaus Knopper като пакети *cloop-src* и *cloop-utils*. Имайте предвид, че ако на вашата система някой драйвър не е компилиран като модул за ядрото или не е закомпилиран в самото ядро, то ще трябва да направите поне едно от двете, за да може да използвате съответния хардуер.

¹<http://debian.gabrovo.com/mailling.php>

²<http://sf.net/projects/bgdebian/>

³http://www.cdforme.net/default.php?cPath=20_24

⁴<http://www.Amazon.com>

⁵http://shop.global.bg/helpbg_n.asp

-
- **B11: При инсталиране или премахване на *packages*, получавам съобщение от вида:**

```
apt-get remove xscreensaver
dpkg: error processing xscreensaver (--remove):
 subprocess post-removal script returned error exit status 127
Errors were encountered while processing:
 xscreensaver
```

O11: Както ни се съобщава, нещата няма да станат автоматично в команди от вида на:

```
# apt-get install -f
```

Също така не бихме постигнали и успех с *force* опции. Трябва да редактираме скриптовете за съответния пакет в директория `/var/lib/dpkg/info/`, така че след като бъдат извикани от **dpkg(8)**, те да приключат своята работа успешно, а не да връщат съобщение за грешка. Това може да бъде по тяхна вина или по вина на програма, която пък те извикват. Скриптовете имат вида: `/var/lib/dpkg/info/пакет.скрипт`, където **скрипт** е `preinst`, `postinst`, `prerm` или `postrm`. В този случай не завършва изпълнението си поради някаква причина и трябва да бъде редактиран, дори и времето *post removal* скрипта `/var/lib/dpkg/info/xscreensaver.postrm` за пакета `xscreensaver`. В книгата е даден пример за самостоятелно (т.е. неавтоматично) справяне с подобни проблеми, като там е взет за демонстрация пакета `scrollkeeper`.

- **B12: Защо е необходим този Perl horror с *debconf*, *debhelper*, *kernel-package* и т.н.**

O12: Всичко това е необходимо за да се унифицира и изгради интелигентен процес по конфигурирането, компилирането и инсталирането на `binary` и `source packages` така, че всичко да е под пълен контрол. **Perl**⁶ често е наричан „швейцарската резачка на Unix“ и е съвсем в реда на нещата да е застъпено по-широкото му използване за административни цели в тези системи. Гореспоменатите пакети точно това правят и това е една от много силните страни на Debian.

Други страници с отговори на въпроси:

- [Официалната Debian страница с отговори](#)⁷
- [Често задавани въпроси във freenode #debian IRC канал](#)⁸
- [Въпроси/отговори за Debian CD-тата](#)⁹
- [Работа с Java в Debian](#)¹⁰
- [KDE3 FAQ](#)¹¹
- [Руска страница с отговори на въпроси, свързани предимно с кирилизацията на Debian](#)¹²
- [Управление на rpm пакети с apt](#)¹³

⁶<http://www.perl.com>

⁷<http://www.debian.org/doc/FAQ/>

⁸<http://www.linuxs.mine.nu/debian-faq/>

⁹<http://www.debian.org/CD/faq/>

¹⁰<http://www.debian.org/doc/manuals/debian-java-faq/>

¹¹<http://davidpashley.com/debian-kde/faq.html>

¹²<http://linux.perm.ru/doc/distro/debian/3.0/>

¹³<http://apt4rpm.sourceforge.net/faq.html>

Част VI

Участие в писането на книгата

Глава 22

За този документ и авторите

22.1. Замисъл и стил на писане

- Да е полезна като съдържание и начин на дистрибутиране за начинаещи и напреднали. Това ще рече, че в началото на книгата матариала трябва да е лек и достъпен за колкото се може повече потребители, като сложността се увеличава с всяка следваща глава.
- Да отразява по-рядко срещани и познати възможности на Debian, дори нехарактерни за официалната документация (например custom installers, защо не и да поддържахме сорсове на такива в *utils/*, non-x86 personal arch experience и др.)

22.2. Препоръки към авторите

FIXME: Тази секция може би се нуждае от обсъждане.

Всеки, който добавя някакъв код, би трябвало да:

- Добавя името си по азбучен ред във файла AUTHORS, както и в полетата `\pdfauthor` и `\author` в `debian-book.tex`.
- Добавя кратки обяснения за промените си в файла `ChangeLog`.
- Обсъжда всички по-структурни промени в списъка debian-bg@lists.gabrovo.com¹.

Книгата е голяма, за това не може да се очаква всички да я прочетат от началото до края, като при това запомнят всяко нещо къде е. Поради тази причина трябва да отделяте специално внимание къде точно слагате своите материали и какво заглавие им слагате. Ръководното правило е да се поставят на мястото на читателя. Поставете се на негово място, погледнете дългото съдържание и решете къде бихте търсили вашия материал.

Следващото нещо е заглавието на материала. Съобразете се с мястото, което сте избрали. Не претоварвайте заглавието, но и не го правете неясно. Ако материалът може да се опише с една ключова дума, сложете я като първа в заглавието (в `\textit`) и я отделете с думеточие. Идеалният вариант е тази ключова дума да е име на пакет, име на команда или име на файл.

Използвайки този начин, позволявате на по-напредналите потребители по-бързо да намират информацията, която им е нужна, защото тези ключови думи може да им говорят повече от самото заглавие. Начинаещите потребители също печелят, защото самото съдържание на книгата става като справочник за пакети, команди и файлове. Ето пример за секция:

```
\chapter{\textit{stow}: Инсталиране на не-дебиански софтуер}
```

¹<http://debian.gabrovo.com/mailling.php>

Избягвайте да пишете команди, вградени в самите абзаци на текста. Отделяйте ги във `\begin{verbatim}`. По този начин човек, без да чете текста, добива представа какво трябва да се прави. В крайна сметка в командите, които препоръчвате, се съдържа цялата практическа страна на вашия материал.

Поради същата причина не вграждайте примерни редове от файлове в текста, а използвайте `\begin{verbatim}`.

Добре е първото изречение на абзац да въвежда в темата на абзаца. Така се дава по-добра възможност да се пресява нужното за читателя от ненужното.

Използвайте богатството на \LaTeX . Не се опитвайте да правите таблици или списъци в среда `verbatim`. Команди, имена на файлове, извадки от файлове — всички тези е добре да бъдат опаковани в `\texttt`.

Ако искате да следите промените, които се извършват в `debian-book`, абонирайте се за списъка [lug-bg-all-cvs](#)².

22.3. Какво остава да се направи

Състояние към момента:

- Трансформирането към \LaTeX все още не е завършено. Кодът се компилира обаче успешно с `latex`, `pdflatex`, `dvips`.
- `latex2html` също генерира задоволителен резултат, въпреки, че поради неговата невъзможност на се справи към момента с `multirow` едната таблица трябваше да влезе като PNG. Алтернатива на `latex2html` може да бъде [hevea](#), при все, че ще се наложи редактиране на някои от файловете на книгата. Повече информация за която можете да намерите на <http://para.inria.fr/~maranget/hevea/>³.
- Който има желание от \LaTeX сорса, може да компилира DVI, PS и каквото още намери за добре.

Предвидено или добре е да се направи:

- Да се премахнат всички FIXME, т.е. където са поставени, означава, че има за допълване или поправяне.
- Да се добави `personal non-x86 arch experience` и `custom made installers`.
- Да се дадат подробни `step-by-step` примери за `official debian packaging beyond Debian' New Maintainers Guide`. Подходящ пример е софтуера от проекта <http://biona.sf.net>⁴
- Евентуално да се включват някъде в съдържанието някои хвърчащи писания, които сега са във вид на самостоятелни статии, както и такива, изчакващи в директория `queue/`.
- По-красиви картинки, ако ще заменят таблиците.
- Евентуално на `Nightly CVS Mirror & Build Locations` да се постави `search engine`, като [namazu2](#)

²<http://lists.sourceforge.net/lists/listinfo/lug-bg-all-cvs>

³<http://para.inria.fr/~maranget/hevea/>

⁴<http://biona.sf.net>

Глава 23

Регистрация в SourceForge

За да бъдете добавен в проекта, трябва да имате регистрация в SourceForge. Името на вашата регистрация трябва да изпратите на администратора на debian-book, който в момента е Огнян Кулев <ogi@fmi.uni-sofia.bg>.

Глава 24

Работа с L^AT_EX

Преди да комитвате в CVS хранилището, проверявайте дали кодът се компилира при вас. Винаги тествайте с PDF.

24.1. Инсталиране на L^AT_EX

L^AT_EX е мощна система за изготвяне на документи, позволяваща форматиране на по-ниско и високо ниво и конвертирането в различни формати за четене и печат. Само част от поддържаните формати са DVI, PostScript, PDF, HTML. Ако сте инсталирали някоя дистрибуция на Линукс, можете да проверите дали имате инсталирани L^AT_EX (или, по-точно, една от реализациите му teTeX), като използвате някоя от командите `latex`, `tex`, `pdflatex`. Ако нито една от тях не работи, тогава ще ви се наложи да инсталирате. Най-добре прочетете инструкциите за инсталация на вашата дистрибуция и изберете съответните необходими пакети.

Ако разполагате с Debian:

```
# apt-get install tetex-extra
```

За съжаление стандартната дистрибуция на L^AT_EX не е достатъчно добре кирилизана и не е много наясно с българския език. Именно затова, преди да се впуснем в не особено сложния процес на създаване на PDF, PostScript и т.н. изходи от L^AT_EX, е нужно да инсталираме няколко пакета.

bgtex-v2 е първият пакет, който ни трябва за българизация на teTeX. Можете да го изтеглите от <ftp://lml.bas.bg/home/anton/tex/> и след като го разпакетирате, внимателно прочетете и следвайте инструкциите за инсталация.

scalable-cyrfonts-tex е пакет, който добавя нови шрифтове в teTeX. Тези шрифтове са напълно безплатни и нямат ограничения за разпространението си. Можете да дръпнете пакета от: <ftp://lml.bas.bg/home/anton/tex/> и след като го разпакетирате внимателно прочетете, следвайте инструкциите за инсталация.

Ако разполагате с Debian:

```
# apt-get install scalable-cyrfonts-tex
```

Не забравяйте да добавите следните ред в съответните файлове:

Файл	Ред
<code>/etc/texmf/dvips/config.ps</code>	<code>p +cyrfonts.map</code>
<code>/etc/texmf/pdftex/pdftex.cfg</code>	<code>map +cyrfonts.map</code>

Ако не можете да намерите някои от тези два файла, пробвайте със следните алтернативни имена:

<code>/etc/texmf/dvips/config.ps</code>	<code>/usr/share/texmf/dvips/config/config.ps</code>
<code>/etc/texmf/pdftex/pdftex.cfg</code>	<code>/usr/share/texmf/pdftex/config/pdftex.cfg</code>

24.2. Писане на L^AT_EX

Тук следват леки насоки за езика L^AT_EX и как го ползваме в нашия документ. Добри източници за езика L^AT_EX са [latexhelp.html](http://www.latexhelp.html), които се намира в главна директория, [Hypertext Help with L^AT_EX](#)¹ и книгата „L^AT_EX Companion“². Нека се спазва установеният L^AT_EX стил, както и стилът на поднасяне на информацията — ясно, точно и ако може да се приведат и примери. Където има съмнения или все още не е довършено нещо, нека се поставя FIXME. Добра идея ще е за повече подробности относно L^AT_EX кода да се обърнете към кода на самия документ в директорията `src/`, и ще добиете представа, че хич не е страшно:

L^AT_EX е мощен език, но ние ползваме много малка част от него. Например:

За части, глави, подглави и т.е. разделението е такова:

```
\part{заглавие}
\chapter{заглавие}
\chapter{заглавие}
\section{заглавие}
\subsection{заглавие}
```

24.2.1. Форматиране на текста

```
\par
Откриваме параграф.
\par
\textbf{тук има bold текст} \\
\textit{тук има italic текст}, може и комбинирано: \\
\textbf{\textit{хем bold, хем italic}} тука ще е само bold текст} \\
\begin{verrbatim}
Изход на някаква команда или просто текст, който искаме да
представим, точно както сме го написали. Нарочно използваме двойно
"\"r"\" шото не можем да го escape-нем, иначе се ползва с едно ;-)
Cheating \ thru \textbackslash doesn't help either :-( FIXME:
\end{verrbatim}
```

Откриваме параграф.

тук има bold текст

тук има italic текст, може и комбинирано:

хем bold, хем italic тука ще е само bold текст

```
Изход на някаква команда или просто текст, който искаме да
представим, точно както сме го написали. Нарочно използваме двойно
"\"r"\" шото не можем да го escape-нем, иначе се ползва с едно ;-)
Cheating \ thru \textbackslash doesn't help either :-( FIXME:
```

24.2.2. Списъци

```
\begin{itemize}
\item Debian
\item GNU
\item Linux
\end{itemize}
```

- Debian
- GNU
- Linux

¹<http://www.giss.nasa.gov/latex/>

²<http://www.amazon.com/exec/obidos/tg/detail/-/0201541998/102-5934600-1253723?vi=glance>

```
\begin{enumerate}
\item Едно
\item Две
\item Три
\end{enumerate}
```

1. Едно
2. Две
3. Три

24.2.3. Таблици

```
\begin{tabular}{|l|r|c|}
\hline \textbf{Първи стълб} & \textbf{Втори стълб} & \textbf{Трети стълб} \\
\hline ляво & дясно & централно \\
\hline още ляво & още дясно & още централно \\
\hline
% Последният \hline задължително трябва да бъде точно на
% следващия ред след последното \\.
\end{tabular}
```

Първи стълб	Втори стълб	Трети стълб
ляво	дясно	централно
още ляво	още дясно	още централно

24.2.4. Специални макроси

В книгата се използват макроси, които не са част от стандартния L^AT_EX. Използвайте ги винаги, когато е възможно.

L ^A T _E X	Изглед
<code>\linkTo{http://www.debian.org}</code>	http://www.debian.org
<code>\hlink{Заглавна страница}{http://www.debian.org}</code>	Заглавна страница ³
<code>\deb{vsftpd}</code>	vsftpd
<code>\man{dpkg}{8}</code>	dpkg(8)
<code>\manx{apt-get}{8}</code>	apt-get(8)

В HTML изходите `\man` генерира хипервръзка към сайт със справочни страници. Той обаче не съдържа всички справочни страници на Debian, а само основните. В случай, че хипервръзката е нежелана, използвайте `\manx`.

24.2.5. Специални знаци

Някои знаци в L^AT_EX са специални и трябва да ги напишете по определен начин, за да ги включите в документа, който пишете:

Знак	Заместител
#	<code>\#</code>
\$	<code>\\$</code>
%	<code>\%</code>
&	<code>\&</code>
~	<code>\textasciitilde</code>
^	<code>\textasciicircum</code>
\	<code>\textbackslash</code>
{	<code>\{</code>
}	<code>\}</code>

Някои от тези специални знаци, като `\textbackslash`, ще ви принудят да добавите интервал след тях. L^AT_EX ще премахне тези интервали в PDF изхода, но latex2html ще го направи само ако няма нов ред между тези интервали. Ако има знак за нов ред, latex2html ще добави интервал, който вероятно не желаете. Това се случва понякога, когато е активиран режимът Word

W^гар на текстовия редактор. Най-доброто решение е да добавите знака % непосредствено преди знака за нов ред.

Ако искате точно обратното поведение: да има интервал след знака, използвайте знака ~, който означава празно място, което не може да се пренася (като в HTML).

Всичко това е вярно и за всяка друга \команда.

```
% Няма интервал между знака и следващата част:  
\textbackslash texttt  
\textbackslash      texttt  
\textbackslash%  
  texttt  
% Има интервал:  
\textbackslash~texttt
```

24.2.6. Кавички

За кавички винаги използвайте " ` и " ' . Така текстът изглежда „по-български“, защото ще бъде по правописните правила на българския език. Не използвайте за кавички ` ` и ' ' , както е описано в книгите за L^AT_EX.

24.2.7. Тирета

Има два основни вида тирета: късо тире и дълго тире. Късото тире се използва между думи — например в „по-бързо“. Дългото тире се използва между части на изречението, като например в предното изречение.

В L^AT_EX за късо тире се използва просто знака -. За дълго тире се използва последователност от три обикновени тирета: ---. Има и още едно междинно тире, което се представя с две обикновени тирета. В английския език се използва между две числа, които са граници на интервал:

Пробвайте 2--3 пъти --- ако не стане, значи няма да стане.

В някои случаи се налага да представите последователност от две тирета. Това се налага при вмъкване на опции на команди. В такива случаи използвайте \verb:

Опциите \verb#--help# и \verb#--version# трябва да се обработват от всяка GNU програма.

24.3. Структура и съдържание - организация на файловете

./ / главна директория. Всичко, на което не може да се намери място, е тук.

src/ / L^AT_EX сорсовете на самата книга и всички изходни файлове, като **src/debian-book.tex** е главният L^AT_EX файл. Започва с всички общи и по-сложни настройки за документа и накрая включва останалите файлове. Ако искате да включите нов файл, се прави по-този начин, ако не, то просто редактирате някой съществуващ. Забележете, че разширението .tex при включване се пропуска.

dists/ / bzip2 tarballs на изходите са в тази директория.

queue/ / ако някой има проблеми с L^AT_EX, то може да предостави своите писания в чист текст, но на български език.

utils/ / ако някой е написал и ползва неофициално някоя собствена програмка, може да я сподели тук.

Глава 25

Работа със CVS

25.1. Достъп до изходните кодове чрез CVS

Естествено, трябва да имате инсталиран CVS. Ако разполагате с Debian:

```
# apt-get install cvs
```

За да изтеглите последните сорсове на книгата от CVS хранилището изпълнете:

```
$ export CVSROOT=:pserver:anonymous@cvs.lug-bg.sourceforge.net:/cvsroot/lug-bg
$ cvs login
$ cvs -z9 checkout -P debian-book
```

Тези, които имат желание да се включат в разработката на книгата и да commit-ват в CVS хранилището, трябва да имат инсталиран SSH. Те могат да изтеглят книгата по следния начин:

```
$ export CVSROOT=:ext:developername@cvs.lug-bg.sourceforge.net:/cvsroot/lug-bg
$ export CVS_RSH=ssh
$ cvs -z9 checkout -P modulename
```

където *developername* е вашето потребителско име, регистрирано в SourceForge (паролата си я знаете ;-), а *modulename*, в случая за книгата, е *debian-book*, като впоследствие ще бъдат включени и други модули към хранилището.

25.2. Бързи инструкции за CVS

Работата със CVS е подробно описана в документацията на SourceForge. Добри източници за CVS са: <http://cvshome.org> и <http://www.loria.fr/~molli/cvs-index.html>. Ако не искате да се задълбочавате чак толкова, тук са представени основните понятия и действия при работата със CVS в SourceForge.

За да направите каквато и да е промяна, трябва да имате *работно копие* на книгата. Преди всяка промяна синхронизирате работното копие със CVS хранилището:

```
$ cvs update -PdR
```

Извършвате промяната и записвате промените и в CVS хранилището:

```
$ cvs commit -m 'описание на промените' file1 path/to/file2
```

Всяко действие със CVS хранилището се извършва чрез командата `cvs`, която получава като параметър името на конкретното действие и евентуално имената на файловете, които са намесени. За по-голямо удобство е нужно променливата `CVSROOT` да съдържа с кое хранилище се работи, а `CVS_RSH` да съдържа `ssh`. Следните команди на Bash настройват обкръжението както трябва (където трябва да замените *име* с името на регистрацията ви в SourceForge):

25 Работа със CVS

```
export CVSROOT=:ext:име@cvs.sourceforge.net:/cvsroot/lug-bg
export CVS_RSH=ssh
```

След като всичко това е настроено, може да изтеглите работно копие на книгата с помощта на командата (опцията `-P` пропуска празните директории, които не са малко в книгата):

```
$ cvs checkout -P debian-book
```

Тази команда създава начално работно копие. За обновяване на съдържанието на работното копие използвайте командата

```
$ cvs update -Pdr
```

която задължително трябва да се изпълнява в директория на работното копие.

`TeX` файловете се намират в директорията `src`. Ако имената на файловете не са достатъчни, за да ви ориентират кой какво съдържа, погледнете във файла `debian-book.tex`.

След като свършите с промените, трябва се обнови и файлът `ChangeLog`. Използвайте следната последователност (пакетът `cvs2cl` трябва да е инсталиран):

```
$ cvs2cl -P
$ cvs commit -m'' ChangeLog
```

25.2.1. CVS session with project-x

```
$ cd # move to the work area
$ cvs co project-x # get sources from CVS to local
$ cd project-x
... make changes to the content ...
$ cvs diff -u # similar to diff -u repository/ local/
$ cvs ci -m "Describe change" # save local sources to CVS
$ vi newfile_added
$ cvs add newfile_added
$ cvs ci -m "Added newfile_added"
$ cvs up # merge latest version from CVS
... watch out for lines starting with "C filename"
... unmodified code is moved to `.#filename.version'.
... Search "<<<<<<" and ">>>>>>" in filename.
$ cvs tag Release-1 # add release tag
... edit further ...
$ cvs tag -d Release-1 # remove release tag
$ cvs ci -m "more comments"
$ cvs tag Release-1 # re-add release tag
$ cd # move back to the work area
$ cvs co -r Release-initial -d old project-x
... get original version to old directory
$ cd old
$ cvs tag -b Release-initial-bugfixes # create branch (-b) tag
... Now you can work on the old version (Tag=sticky)
$ cvs update
... Source tree now has sticky tag "Release-initial-bugfixes"
... Work on this branch
$ cvs up # sync with files modified by others on this branch
$ cvs ci -m "check into this branch"
$ cvs update -kk -A
... Remove sticky tag and forget contents
... Update from main trunk without keyword expansion
$ cvs update -kk -j Release-initial-bugfixes
... Merge from Release-initial-bugfixes branch into the main
... trunk without keyword expansion. Fix conflicts with editor.
$ cvs ci -m "merge Release-initial-bugfixes"
$ cd
$ tar -cvzf old-project-x.tar.gz old # make archive, -j for bz2
$ cvs release -d old # remove local source (optional)
```

25.2.2. Добавяне на файлове

Добавянето на файлове се извършва посредством две стъпки: Първо стартирате командата `add`, а след това – `commit`. Файлът няма да се появи в хранилището, докато не се изпълни `commit`:

```
$ cvs add newfile.c
cvs add: scheduling file 'newfile.c' for addition
cvs add: use 'cvs commit' to add this file permanently
$ cvs ci -m "added newfile.c" newfile.c
RCS file: /usr/local/cvs/myproj/newfile.c,v
done
Checking in newfile.c;
/usr/local/cvs/myproj/newfile.c,v <- newfile.c
initial revision: 1.1
done
```

25.2.3. Добавяне на директории

Unlike adding a file, adding a new directory is done in one step; there's no need to do a commit afterwards:

```
$ mkdir c-subdir
$ cvs add c-subdir
Directory /usr/local/cvs/myproj/c-subdir added to the repository
```

If you look inside the new directory in the working copy, you'll see that a CVS subdirectory was created automatically by `add`:

```
$ ls c-subdir
CVS/
$ ls c-subdir/CVS
Entries      Repository  Root
```

Now you can add files (or new directories) inside it, as with any other working copy directory.

25.2.4. Премахване на файлове

Removing a file is similar to adding one, except there's an extra step: You have to remove the file from the working copy first:

```
$ rm newfile.c
$ cvs remove newfile.c
cvs remove: scheduling 'newfile.c' for removal
cvs remove: use 'cvs commit' to remove this file permanently
$ cvs ci -m "removed newfile.c" newfile.c
Removing newfile.c;
/usr/local/cvs/myproj/newfile.c,v <- newfile.c
new revision: delete; previous revision: 1.1
done
```

Notice how, in the second and third commands, we name `newfile.c` explicitly even though it doesn't exist in the working copy anymore. Of course, in the `commit`, you don't absolutely need to name the file, as long as you don't mind the `commit` encompassing any other modifications that may have taken place in the working copy.

25.2.5. Премахване на директории

As I said before, CVS doesn't really keep directories under version control. Instead, as a kind of cheap substitute, it offers certain odd behaviors that in most cases do the "right thing". One of these odd behaviors is that empty directories can be treated specially. If you want to remove a directory from a project, you first remove all the files in it

```
$ cd dir
$ rm file1 file2 file3
$ cvs remove file1 file2 file3
(output omitted)
$ cvs ci -m "removed all files" file1 file2 file3
(output omitted)
```

and then run update in the directory above it with the -P flag:

```
$ cd ..
$ cvs update -P
(output omitted)
```

The -P option tells update to "prune" any empty directories – that is, to remove them from the working copy. Once that's done, the directory can be said to have been removed; all of its files are gone, and the directory itself is gone (from the working copy, at least, although there is actually still an empty directory in the repository). An interesting counterpart to this behavior is that when you run a plain update, CVS does not automatically bring new directories from the repository into your working copy. There are a couple of different justifications for this, none really worth going into here. The short answer is that from time to time you should run update with the -d flag, telling it to bring down any new directories from the repository.

25.2.6. Преименуване на файлове и директории

Renaming a file is equivalent to creating it under the new name and removing it under the old. In Unix, the commands are:

```
$ cp oldname newname
$ rm oldname
```

Here's the equivalent in CVS:

```
$ mv oldname newname
$ cvs remove oldname
(output omitted)
$ cvs add newname
(output omitted)
$ cvs ci -m "renamed oldname to newname" oldname newname
(output omitted)
$
```

For files, that's all there is to it. Renaming directories is not done very differently: create the new directory, cvs add it, move all the files from the old directory to the new one, cvs remove them from the old directory, cvs add them in the new one, cvs commit so everything takes effect, and then do cvs update -P to make the now-empty directory disappear from the working copy. That is to say:

```
$ mkdir newdir
$ cvs add newdir
$ mv olddir/* newdir
mv: newdir/CVS: cannot overwrite directory
$ cd olddir
$ cvs rm foo.c bar.txt
$ cd ../newdir
$ cvs add foo.c bar.txt
$ cd ..
$ cvs commit -m "moved foo.c and bar.txt from olddir to newdir"
$ cvs update -P
```

Note: the warning message after the third command. It's telling you that it can't copy olddir's CVS/ subdirectory into newdir because newdir already has a directory of that name. This is fine, because you want olddir to keep its CVS/ subdirectory anyway. Obviously, moving directories around can get a bit cumbersome. The best policy is to try to come up with a good layout when you initially import your project so you won't have to move directories around very often. Later, you'll learn about a more drastic method of moving directories that involves making the change directly in the repository. However, that method is best saved for emergencies; whenever possible, it's best to handle everything with CVS operations inside working copies.

25.2.7. CVS и бинарните файлове

Until now, I've left unsaid the dirty little secret of CVS, which is that it doesn't handle binary files very well (well, there are other dirty little secrets, but this definitely counts as one of the dirtiest). It's not that CVS doesn't handle binaries at all; it does, just not with any great panache. All the files we've been working with until now have been plain text files. CVS has some special tricks for text files. For example, when it's working between a Unix repository and a Windows or Macintosh working copy, it converts file line endings appropriately for each platform. For example, Unix convention is to use a linefeed (LF) only, whereas Windows expects a carriage return/linefeed (CRLF) sequence at the end of each line. Thus, the files in a working copy on a Windows machine will have CRLF endings, but a working copy of the same project on a Unix machine will have LF endings (the repository itself is always stored in LF format). Another trick is that CVS detects special strings, known as RCS keyword strings, in text files and replaces them with revision information and other useful things. For example, if your file contains this string

```
$Revision: 1.38 $
```

CVS will expand on each commit to include the revision number. For example, it may get expanded to

```
$Revision: 1.38 $
```

CVS will keep that string up to date as the file is developed. (The various keyword strings are documented in *Advanced CVS* and *Third-Party Tools*.) This string expansion is a very useful feature in text files, as it allows you to see the revision number or other information about a file while you're editing it. But what if the file is a JPG image? Or a compiled executable program? In those kinds of files, CVS could do some serious damage if it blundered around expanding any keyword string that it encountered. In a binary, such strings may even appear by coincidence.

Therefore, when you add a binary file, you have to tell CVS to turn off both keyword expansion and line-ending conversion. To do so, use `-kb`:

```
$ cvs add -kb filename
$ cvs ci -m "added blah" filename
(etc)
```

Also, in some cases (such as text files that are likely to contain spurious keyword strings), you may wish to disable just the keyword expansion. That's done with `-ko`:

```
$ cvs add -ko filename
$ cvs ci -m "added blah" filename
(etc)
```

Note that you can't meaningfully run `cvs diff` on two revisions of a binary file. Diff uses a text-based algorithm that can only report whether two binary files differ, but not how they differ. Future versions of CVS may provide a way to diff binary files.

25.2.8. Заключение

В крайна сметка правете само неща, които наистина разбирате ;-)

Преди да комитвате в CVS хранилището проверявайте дали кода се компилира при вас, винаги тествайте с PDF.

25.3. Използване на общодостъпен ключ за достъп до CVS

По подразбиране всяка команда `cvs` изисква парола за достъп. Това е досадно, но решение има - използването на общодостъпен ключ (*public key*) вместо парола.

Отново, всичко това е описано подробно в документацията на SourceForge, но тук ще бъдат представени кратки инструкции, засягащи темата единствено в най-общия случай.

1. Изпълнете командата `ssh-keygen -t dsa`. Въведете за парола добре измислена парола, която да е поне толкова неразбиваема, колкото паролата ви в SourceForge.

2. Отидете в личната ви страница в SourceForge, обикновено достъпна чрез my.sf.net от горната хоризонтална лента на сайта.
3. Оттам отидете в страницата Account Options, а после в Edit Keys (намира се най-отдолу на страницата).
4. В текстовото поле Authorized keys копирайте единствения ред, който се намира във файла `.ssh/id_dsa.pub`, генериран от командата `ssh-keygen`. Внимавайте да не добавите излишни нови редове. Потвърдете с бутона Update.

С това всичко е подготвено за почти безпаролна работа със CVS хранилищата на SourceForge. Трябва, обаче, да изчакате към 6 часа, за да може общодостъпния ключ да стигне до сървърите на SourceForge.

За да използвате ключа, веднъж в цялата X (или конзолна) сесия трябва да изпълните командата `ssh-add`, която пита за паролата, която дадохте на `ssh-keygen`. По този начин в текущата сесия се зарежда личния ключ (*private key*), който се използва при всяко извикване на командата `ssh`. (Всички команди към CVS хранилището минават през `ssh`, затова и трябва да се установи променливата `CVS_RSH`.)

Ако командата `ssh-add` се оплаче, че не може да намери `ssh-agent`, значи дистрибуцията, която ползвате, не е направена, както трябва. В Дебиан няма такива проблеми. Тук няма да се показва решение на този проблем, а ако го имате, обърнете се към справочника относно командата `ssh-agent`.

Глава 26

Как да генерираме PDF, DVI, Postscript, HTML

За да получите **PDF** изхода, изпълнете:

```
$ make pdf
```

DVI и Postscript за сега не са предвидени в Makefile-a, но можете да ги получите чрез командите:

```
$ latex debian-book.tex  
$ dvips debian-book.tex
```

Внимание: поради това, че обема на книгата доста нарастна, може би ще се наложи да разрешите на програмите от пакета `TeX` да ползват по-голямо количество памет от зададено по-подразбиране. Това става от `pool_size` стойността във файла `texmf.cnf`. Намира се в `/etc/texmf/texmf.cnf` или го потърсете с някоя от командите:

```
$ locate texmf.cnf  
$ find / -name texmf.cnf
```

Намерете в него реда

```
pool_size = 125000
```

И увеличете тази стойност няколко пъти, примерно на 625000.

Дръпнете си последната версия на `latex2html` от:

<http://saftsack.fs.uni-bayreuth.de/~latex2ht/current/latex2html-2K.1beta.tar.gz>

Ако разполагате с Debian:

```
# apt-get install latex2html
```

За да получите **HTML** изходите, изпълнете:

```
$ cd debian-book/src  
$ make html  
или  
$ make htmsplit
```

Забележка: Когато пишете на `LaTeX` и не сте убедени, че сте се справили много добре, е хубаво да коментирате първият ред в `debian-book.tex`, т.е. да изключите `batch` режима. Така ще можете да виждате своите грешки.

Ако искате да компилирате всичко:

```
$ cd src/  
$ make clean dists-clean  
$ make dists  
$ make pdf html htmsplit
```


Част VII

Мотивация и благодарности

Това е резултат на наблюдения с течение на времето, как едни или други потребители на GNU/Linux измъчват себе си или измъчват GNU/Linux дистрибуцията, с която работят (без значение коя). Достигнахме до убеждението, че не е лошо да събираме тези неща на едно място и някога да ги приведем в по-удобен вид за четене, което може би ще спомогне да се оцени с какво може или не може да им бъде полезен Debian GNU/Linux. Дали след това въобще потребителите и ще опитат ползването на една такава дистрибуция, това е без значение. Това, разбира се, в никакъв случай не е опит за подценяване качествата на другите дистрибуции или опит за оценяване на нещата от позицията на всезнаещи и всеможещи. Напротив, като най-обикновени потребители приемаме всякакви мнения и схващания различни от нашите, от което можем да се възползваме и преоценявайки нещата, евентуално да придобием чужд опит и знания (едно от хитрите неща, които човек може да направи;-). От друга страна пък е удоволствие, когато след подобно представяне на дистрибуцията получаваме като обратна връзка благодарности от неподозиращи силата на Debian потребители (мда, случвало се е и доста по-напреднали потребители да остават благодарни, че сме им обърнали внимание за съществуването на дистрибуцията, но по-голямо удоволствие е, когато някой нов потребител оцени нещата и евентуално се възползва от тях, понеже по-напредналите не са за жалене, те си знаят ;-).

Специални благодарности:

- Christoph Lameter - за задълбочения обзор по проблемите на управление на софтуера, потребяван в особено големи количества.
- Антон Зиновиев - за всичко българско, включено в Debian.
- Валентин Вълчев и Минко Марков - за консултациите относно LaTeX.
- и разбира се многобройните debian support channels



Част VIII

Лиценз

Chapter 27

GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

27.1 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any

mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, \LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

27.2 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 27.3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

27.3 COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the

copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

27.4 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 27.2 and 27.3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

1. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
2. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
3. State on the Title page the name of the publisher of the Modified Version, as the publisher.
4. Preserve all the copyright notices of the Document.
5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
6. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
8. Include an unaltered copy of this License.
9. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

10. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
11. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
12. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
13. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
14. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
15. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

27.5 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 27.4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

27.6 COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

27.7 AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 27.3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

27.8 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 27.4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 27.4) to Preserve its Title (section 27.1) will typically require changing the actual title.

27.9 TERMINATION

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

27.10 FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.