Книга за Debian GNU/Linux

16 декември 2004 г.

Версия 0.3cvs

Съдържание

Част I

Въведение

Глава 1

За този документ и потребителите

1.1. Достъп до документа

Официалния сайт на книгата е http://www.debianbookbg.org/[1], където ще се публикуват официалните издания. Проекта е преместен на https://openfmi.net/projects/debian-book-bg[2] като кода се поддържа в Subversion[3] хранилище. Най-добре би било, ако всички потребители използват направо кода от това SVN-хранилище, синхронизирайки своето локално копие от него и компилирайки по свое собствено желание различните изходи – HTML, при който всичко в един файл, или HTML, разбит по глави, PDF и т.н. Така всички бързо и лесно ще се сдобиват с последните промени в хранилището и ще могат да добавят свои допълнения, когато намерят за добре.

Цялата документация и информация, необходима на потребителите, за да генерират документа на своите машини, както и такава за участие в проекта и лицензa, под който се разпространява, идва с изходния код.

1.2. Накратко за съдържанието на книгата

Тази книга се разработва в свободното време с помощта на LaTeX и SVN. Предназначена 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. Предполага се, че имате поне основна представа от структурата и основните особености на Debian, можете да работите с най-важните инструменти за управление на системата. Интересно е да се споделя 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 и дизайнерска гледна точка, така че подобна помощ, а както и всякакви конструктивни критики и коментари по съдържанието, се приемат с удоволствие. Така ще е поне докато махнем и последното FIXME, включително и това. Целта не е точно описание на конкретен Release на Debian, това е преходно и е описано на доста места, затова ще гледаме малко по-глобално. Важното е да се посочат идеята и дизайнът, 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[4] и GNU/Linux[5], но все пак силно препоръчително е да се прочете внимателно и задълбочено Linux kernel mailing list FAQ[6], където се изясняват неща по принцип, като не е нужно да се записвате в този списък, разбира се. Ако не разбирате или не ви е ясно нещо от гореспоменатото, то се препоръчва да започнете първо с:

- http://linux-book.hit.bg[7]
- http://linux-book.logos-bg.net[8]

които дават 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 инсталирана Debian подръка, можете да заредите на произволна x86 машина (PC) Knoppix[9] от CDROM (без да предприемате каквито и да било интервенции по хард дисковете, 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

[1] http://www.debianbookbg.org/
[2] https://openfmi.net/projects/debian-book-bg
[3] http://subversion.tigris.org
[4] http://www.kernel.org
[5] http://www.gnu.org/gnu/linux-and-gnu.html
[6] http://www.kernel.org/pub/linux/docs/lkml/
[7] http://linux-book.hit.bg
[8] http://linux-book.logos-bg.net
[9] http://www.knoppix.org

прочетете Историята на проекта Debian[10] (от пакета `debian-history`) и отговорите на Често Задаваните Въпроси за Debian[11] (от пакета `doc-debian`).

Също така е добра идея да се запишете в:

- http://lists.uni-sofia.bg/cgi-bin/mailman/listinfo/debian[12] пощенския списък в който се обсъжда Debian от български участници
- http://www.linux-bulgaria.org/public/mail_list.html[13] пощенския списък в който се обсъжда GNU/Linux от български участници

1.3. Други книги за Debian на английски език

Следните книги могат свободно да се четат през Интернет:

- Debian Reference[14]
- The Debian Universe[15]
- Guide to Debian GNU/Linux Desktop Survival[16]

[10] http://www.debian.org/doc/manuals/project-history/
[11] http://www.debian.org/doc/FAQ/
[12] http://lists.uni-sofia.bg/cgi-bin/mailman/listinfo/debian
[13] http://www.linux-bulgaria.org/public/mail_list.html
[14] http://qref.sourceforge.net/
[15] http://www.debianuniverse.com/
[16] http://www.togaware.com/linux/survivor/

Глава 2

Защо Debian GNU/Linux

Debian GNU/Linux[1] 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 по-нататъшно 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
[2] http://www.debian.org/social_contract
[3] http://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 системи*. 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 пакетират програмите, може би даже по различен начин.

[1] http://www.redhat.com/
[2] http://www.rpm.org/
[3] http://www.mandrakelinux.com/

3.2.3. Някои термини и понятия

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:

- **upstream sources** - сорсове на дадена програма или приложение.
- **debian source package** - **upstream sources**, към които е добавена директория **debian/**, съдържаща кода, който контролира процеса на изграждане (билд процеса) за получаването на **debian binary packages** (**.deb's**) за съответните хардуерни архитектури.
- **upstream developer** - разработчик на софтуер, по принцип програмист. Тези създават **upstream sources**.
- **upstream maintainer** - този които 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 по-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[4]). Има Група от опитни debian developers[5], които се занимават с това процедиране на новите кандидатури. Тези, които все още не са станали официални **debian maintainers**, могат да си намерят Sponsor[6] - 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[7]. Не бъркайте този вид Developer Sponsorship[8] с Partners[9] или Donations[10]

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, който е разработен специално за проекта Debian, но може да се ползва и от всички останали, разбира се.

[4] http://nm.debian.org
[5] http://nm.debian.org/whoisam.php
[6] http://www.debian.org/devel/join/newmaint#Sponsor
[7] http://mentors.debian.net
[8] http://www.debian.org/devel/join/newmaint#Sponsor
[9] http://www.debian.org/parthners
[10] http://www.debian.org/donations

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. В следващите глави на тази книга ще се опитаме да обясним как се решава всичко това в света на Debian.

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 Debian GNU/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

Част III

Решения и предложения за подобрения

Глава 6

Бърз преглед, без инсталация

6.1. Knoppix LiveCD

6.1.1. От Knoppix в паметта...

Knoppix[1] 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 готови LiveCD[2], то се препоръчва като такова да се ползва именно Knoppix. По-добро решение трудно ще бъде измислено, като освен това може да послужи и като инсталатор. От друга страна има пакет, с чиято помощ можете сами да си направите LiveCD, съдържащо точно това което ви трябва. Това е пакета `dfsbuild`, като за разлика от готовите Knoppix имиджи, с него можете да си направите и LiveCD и за архитектури различни от x86. За да се запоснаете с Knoppix ви трябва само x86 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://developer.linuxtag.net/knoppix/sources/[8], тези сорсове са пакетирани като пакети за Debian (`.debs`) за i386 и са достъпни от http://developer.linuxtag.net/knoppix/i386/[9]. Доста от този код на автора е оценен като полезен и е приет в официалния Debian архив. Форумът[10] и пощенския списък `debian-knoppix`[11] 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 променяте поведението при зареждане с подаването на cheatcodes[12], които може да разберете с F2, когато в началото ви се подава `boot:` промпта). Двата 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:

[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
[8] http://developer.linuxtag.net/knoppix/sources/
[9] http://developer.linuxtag.net/knoppix/i386/
[10] http://www.knoppix.net/forum
[11] http://mailman.linuxtag.org/mailman/listinfo/debian-knoppix
[12] http://download.linuxtag.org/knoppix/knoppix-cheatcodes.txt

```
$ 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
```

тази парола 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.

Трябва да отбележим, че от Knoppix 3.3 освен *knx-hdinstall* вече има и още един хард-диск инсталатор наречен *knoppix-installer*. Той предлага два режима на инсталация *knoppix-mode* при който зареждането на инсталираната върху хард-диска система ще прави автоматично разпознаване на хардуера точно както това става на LiveCD-то и *debian-mode*, който инсталира традиционния Debian, като неговия официален инсталатор. Ако при инсталацията на хард-диска използвате *knoppix-mode* след това не би трябвало да има проблеми ако обновявате от официалния Debian архив, понеже файловете специфични за Knoppix няма да бъдат закачани от **dpkg**, поне докато не влязат подобни пакети доставящи точно същите файлове точно на същите места във вашата система, при което **dpkg** ще предупреди за **overwrite** и ще преустанови инсталацията на тези пакети докато не го принудите с някоя от *force* опциите му. За повече, подробности http://www.knoppix.net/forum/viewtopic.php?t=5297[13] и http://www.knoppix.net/forum/viewtopic.php?t=5017[14], където ще намерите още по-подробен ChangeLog на Knoppix 3.3 съдържащ нови *cheatcodes* както и коментари за *knoppix-mode* и *debian-mode*.

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 помощта на скрипта `knx-hdinstall` и описаното на http://www.freenet.org.nz/misc/knoppix-install.html[15] или с новия `knoppix-installer` от Knoppix 3.3 или както е описано в Install Manual[16] 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[17].

Има и още един вариант да инсталирате оригиналния истински Debian чрез зареждане на Knoppix LiveCD. Това е чрез bootstraping на Debian от Knoppix е описано тук: http://www.inittab.de/manuals/debootstrap.html[18] Накратко ако не ви задоволяват `boot-floppies`, буутвате с Knoppix и използвате `debootstrap` за да инсталира Debian Base система.

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/[19]. 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`, което може да научите от Debian Jigdo mini-HOWTO[20], също така са изброени и vendors, които могат да продават CD's (не се заплаща софтуера, а само носителя!). 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[21] 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`, който се ползват и в други дистрибуции. Knoppix LiveCD, например, ползва собствени конфигуриращи хардуера скриптове заедно с модула cloop, който вече е в официалния

[13] http://www.knoppix.net/forum/viewtopic.php?t=5297
[14] http://www.knoppix.net/forum/viewtopic.php?t=5017
[15] http://www.freenet.org.nz/misc/knoppix-install.html
[16] http://www.debian.org/releases/stable/installmanual
[17] http://www.osnews.com/story.php?news_id=2016
[18] http://www.inittab.de/manuals/debootstrap.html
[19] http://www.debian.org/CD/
[20] http://www.tldp.org/HOWTO/Debian-Jigdo/index.html
[21] http://www.debian.org/CD/jigdo-cd/

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.

6.2. Достъп до Knoppix и Debian CD и DVD images

6.2.1. Knoppix images

- http://www.knopper.net/knoppix/[22] - официален сайт.
- http://www.knopper.net/knoppix-mirrors/index-en.html[23] - download.
- http://www.knopper.net/download/knoppix/[24] - специфичните за Knoppix binary и source packages.

6.2.2. Debian images

- http://www.debian.org/CD/vendors/[25] - CD доставчици.
- http://www.debian.org/distrib/cd[26] - CD iso images.
- ftp://cdimage.debian.org/debian-cd/[27] - Официални CD images на Stable Releases през ftp, rsync, jigdo за всички архитектури (*за сега не е мирорнато у нас с изключение на x86 CD images, вероятно поради убийствените изисквания за intl bandwidth и local storage*).
- rsync -avz cdimage.debian.org::debian-cd/
- ftp://ftp.fsn.hu/pub/CDROM-Images/debian/[28]
- ftp://ftp.fsn.hu/pub/CDROM-Images/debian-unofficial/[29] - Неофициални CD и DVD images на Stable, Testing и Unstable през ftp, rsync, jigdo за x86 архитектурата (*за сега не е мирорнато у нас без изключения, вероятно поради убийствените изисквания за intl bandwidth и local storage*).

6.2.3. Български миръри на Knoppix и Debian

За българските потребители ще е по-удобно да изтеглят Debian Knoppix от български миръри: Images:

- ftp://ftp.bg.debian.org/debian-cd/[30] - официалния Debian mirror за България
- ftp://ftp.uni-sofia.bg/cd-images/[31] - вторичен мирър - Knoppix и Debian images и много други.
- rsync -auv debian.ludost.net::cd-images/linux/ - вторичен мирър - Knoppix и Debian images и много други.
- jigdo-lite ftp://debian.ludost.net/debian-jigdo/..jigdo - jigdo[32] files за сглобяване на images при потребителя.

Binary и Source Packages - тези може да ги описвате в `/etc/apt/sources.list`. В директорията `utils/home/` на книгата може да разгледате такива конфигурационни файлове:

- ftp://ftp.bg.debian.org[33]
- ftp://ftp.uni-sofia.bg[34]
- http://debian.ludost.net[35]

Не бързайте да изтегляте Debian images:

- Debian поддържа отчайващо много официални и неофициални начини на инсталация и може да се окаже, че за вас е по-подходящ някой друг начин на инсталация, като netinst[36] или инсталация на Debian Base от Knoppix LiveCD чрез инсталаторите `knx-hdinstall` и knoppix-installer[37] или пък чрез debootstrap[38]. Ето и едно сравнение между някои от тях[39] като по принцип е добра идея да се прегледа първо официалния инсталатор на Debian[40] и документацията на Knoppix[41] за други интересни и алтернативни начини на инсталация.

[22] http://www.knopper.net/knoppix/
[23] http://www.knopper.net/knoppix-mirrors/index-en.html
[24] http://www.knopper.net/download/knoppix/
[25] http://www.debian.org/CD/vendors/
[26] http://www.debian.org/distrib/cd
[27] ftp://cdimage.debian.org/debian-cd/
[28] ftp://ftp.fsn.hu/pub/CDROM-Images/debian/
[29] ftp://ftp.fsn.hu/pub/CDROM-Images/debian-unofficial/
[30] ftp://ftp.bg.debian.org/debian-cd/
[31] ftp://ftp.uni-sofia.bg/cd-images/
[32] http://atterer.net/jigdo/
[33] ftp://ftp.bg.debian.org
[34] ftp://ftp.uni-sofia.bg
[35] http://debian.ludost.net
[36] http://www.debian.org/distrib/netinst
[37] http://www.knoppix.net/docs/index.php/KnoppixInstaller
[38] http://www.inittab.de/manuals/debootstrap.html
[39] http://www.knoppix.net/forum/viewtopic.php?t=6785
[40] http://www.debian.org/devel/debian-installer/
[41] http://www.knoppix.net/docs/

– Също така не е нужно да разполагате с всичките CD-та, първото и второто ще са ви предостатъчни, 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.

6.2.4. Примери за сглобяване на ISO files с jigdo

Примерни огледални сайтове

Ако по българските миръри няма ISO файла който търсите (за дадена архитектура или по-нова в момента разработвана и неиздадена версия) то можем да ползваме *.deb's* които ги има по нашите миръри и да не теглим ISO files от чужбина, а само т.н. *.jigdo* и *.template* files. Българските миръри могат да ги изтеглят от:

```
rsync -avz cdimage.debian.org::debian-cd/
rsync -avz us.cdimage.debian.org::jigdo-area/
rsync -avz non-us.cdimage.debian.org::debian-jigdo/
```

Много добър пример за *ISO* и *jigdo* files, а така и *rsync* сървър е **mirrors.kernel.org**. В поддиректории на **debian-cd/** се съхраняват:

– **jigdo-area/** съдържаща съответните **.jigdo** и **.template** files.

– **jigdo/** и символични връзки <*архитектура*> -> *jigdo/*<*архитектура*> за **ISO** files.

```
# rsync -avz mirrors.kernel.org::debian-cd/
drwxrwxr-x        4096 2003/12/02 16:49:56 jigdo-area
drwxr-xr-x        4096 2003/01/15 09:47:59 jigdo-area/3.0_r0
...
drwxr-xr-x        4096 2003/12/02 16:49:56 jigdo-area/3.0_r2
drwxr-xr-x        4096 2003/12/04 12:20:13 jigdo-area/3.0_r2/jigdo
drwxr-xr-x        4096 2003/12/02 16:51:22 jigdo-area/3.0_r2/jigdo/hppa
-rw-r--r--         454 2003/11/27 02:23:38 jigdo-area/3.0_r2/jigdo/hppa/MD5SUMS
-rw-r--r--       32791 2003/11/27 02:14:19 jigdo-area/3.0_r2/jigdo/hppa/woody-hppa-1.jigdo
-rw-r--r--       14515668 2003/11/27 02:15:17 jigdo-area/3.0_r2/jigdo/hppa/woody-hppa-1.temp
-rw-r--r--       39198 2003/11/27 02:15:38 jigdo-area/3.0_r2/jigdo/hppa/woody-hppa-1_NONUS.j
-rw-r--r--       14640001 2003/11/27 02:15:41 jigdo-area/3.0_r2/jigdo/hppa/woody-hppa-1_NONU
:::
drwxr-xr-x        4096 2003/03/24 07:43:53 jigdo
drwxr-xr-x        4096 2003/03/24 05:21:57 jigdo/ia64
-rw-r--r--       607846400 2003/03/24 05:08:06 jigdo/ia64/debian-30r1-ia64-binary-1.iso
...
lrwxrwxrwx          11 2003/03/24 05:20:29 alpha -> jigdo/alpha
lrwxrwxrwx          9 2003/03/24 05:20:32 arm -> jigdo/arm
lrwxrwxrwx          10 2003/03/24 05:20:37 hppa -> jigdo/hppa
lrwxrwxrwx          10 2003/03/21 07:14:47 i386 -> jigdo/i386
lrwxrwxrwx          10 2003/03/24 05:20:45 ia64 -> jigdo/ia64
lrwxrwxrwx          10 2003/03/24 05:54:13 m68k -> jigdo/m68k
lrwxrwxrwx          10 2003/03/24 05:54:16 mips -> jigdo/mips
lrwxrwxrwx          12 2003/03/24 08:29:08 mipsel -> jigdo/mipsel
lrwxrwxrwx          13 2003/03/24 08:29:13 powerpc -> jigdo/powerpc
lrwxrwxrwx          10 2003/03/24 08:29:19 s390 -> jigdo/s390
lrwxrwxrwx          12 2003/03/24 08:29:23 source -> jigdo/source
lrwxrwxrwx          11 2003/03/24 08:29:27 sparc -> jigdo/sparc
```

Домашен потребител не е нужно да мирорва горното, достатъчно е да знае какво ISO иска и да си избере един *.jigdo* file , съответния *.template* file ще бъде изтеглен автоматично.

Сглобяване на `ISO image` който го няма по българските миръри с изтегляне на даден `.jigdo` и `.template` file

– Намираме и изтегляме един *.jigdo* файл по наше желание от някой чужд мирър:

```
# jigdo-lite ftp://ftp.fsn.hu/pub/CDROM-Images/debian-unofficial/sid/jigdo/sid-i386-1.j
```

Ако имаме `.jigdo` файл в текущата директория можем просто да го подаден на `jigdo-lite`:

```
# jigdo-lite sid-i386-1.jigdo
```

– Ще ни се даде възможност да спестим теглене на пакети ако вече ги имаме:

```
If you already have a previous version of the CD you are
downloading, jigdo can re-use files on the old CD that are also
present in the new image, and you do not need to download them
again. Mount the old CD ROM and enter the path it is mounted under
(e.g. '/mnt/cdrom').
Alternatively, just press enter if you want to start downloading
the remaining files.
Files to scan:
```

Имайте предвид, че освен CDROM-дискове, може да монтирате и ISO-файлове:

```
# mount -o loop /path/to/file.iso /mnt/loop1
```

– Ще бъде изтеглен съответния *.template* файл (ако не бъде намерен в текущата директория) за кореспондиращия му *.jigdo* файла посочен по-горе.

– Монтираната медия ще бъде сканирана, при което ще бъдете информирани колко файла са намерени от тези които се изискват от `.template` файла.

– Предлага ни се отново да монтираме някоя медия за да бъде сканирана:

```
If you already have a previous version of the CD you are
downloading, jigdo can re-use files on the old CD that are also
present in the new image, and you do not need to download them
again. Mount the old CD ROM and enter the path it is mounted under
(e.g. '/mnt/cdrom').
Alternatively, just press enter if you want to start downloading
the remaining files.
You can also enter a single digit from the list below to
select the respective entry for scanning:
  1: /mnt/loop1
  2: /mnt/loop2
Files to scan:
```

Можем да посочим път до нещо което е монтирано което и го няма в списъка, да изберем нещо съществуващо от списъка или просто не избираме нищо и натискаме само `Enter`.

– Въвеждаме някой български FTP/HTTP мирър на Debian, т.е. пътя до директориите *debian/* и *debian-non-US/* откъдето ще бъдат изтеглени binary *.deb's* или source packages, прави им се *checksum* проверка:

```
Debian mirror: ftp.bg.debian.org/debian/
Debian non-US mirror: ftp.bg.debian.org/debian-non-US/
```

След което ще бъде сглобен ISO image. В най-лошия случай ако това се провали можете да започнете с **jigdo-lite**(1) отначало като ще се опита да продължи от там докъдето е стигнал. Ако горното не помогне продължавате с **rsync**(1), стига да намерите някъде online ISO-то което ви трябва. Махаме разширението *.tmp* и изпълняваме (т.е. *rsync* ще пише във файла на който махнахме разширението *.tmp*):

```
# rsync rsync://server.org/path/binary-i386-1.iso binary-i386-1.iso
```

Сглобяване на `ISO image` който го няма по българските миръри с изтегляне на всички `.jigdo` и `.template` files

За разлика от горния пример тук мирорваме локално всички `.jigdo` и `.template` файлове със скрипт подобен на този от `utils/desync.sh`, необходимото място е около 1GB. След което процедираме по следния начин (например ще сглобяваме CD 1 на Sarge за Alpha):

```
cd debian-cd-unofficial/sarge/jigdo/
jigdo-lite sarge-alpha-1.jigdo
```

jigdo-lite ще ни съобщи, че е намерил съответния `.template` в текущата директория (особено удобно ако в `.jigdo`, секция [Image], Template= е зададено с абсолютен път, вместо с относителен):

```
Not downloading .template file - 'sarge-alpha-1.template' already present
```

За `debian/` и `debian-non-US/` миръри на пакети посочваме първо българските, като дори и на тях да няма пакети за `alpha`, то най-малко ще бъдат изтеглени архитектурно независимите пакети и `jigdo-lite` ще ни съобщи колко няма да могат да бъдат изтеглени, при което имаме следните възможности:

– jigdo-lite ще опита сам да изтегли ненамерените файлове от алтернативни източници. Например:

```
-----------------------------------------------------------------
4 files not found in previous pass, trying
alternative download locations:
--14:39:42--  ftp://ftp.fsn.hu/pub/debian-superseded/aOK458Q62Cmdpj2HW1ObrQ
          => 'debian-20031230-i386-binary-2.iso.tmpdir/ftp.fsn.hu/pub/debian-superseded
```

Вижте също и fallback servers[42].

– да опитаме отново от същите сървъри.

[42] http://lists.debian.org/debian-cd/2003/debian-cd-200312/msg00157.html

– да опитаме с други `debian` и `debian-non-US` сървъри - при което посочваме някои извън страната и стартираме пак `jigdo-lite` при което ще продължи от там докъдето е стигнато.
– продължаваме с `rsync` на съответния недоизтеглен `ISO image` след като го намерим на някой сървър.

Подробности на: http://www.debian.org/CD/jigdo-cd/#faq[43] и http://atterer.net/jigdo/[44]

6.3. *Tilix*: българизиран Knoppix

Tilix[45] ftp://ftp.uni-sofia.bg/cd-images/linux/tilix/[46]

6.4. Други Knoppix-live LiveCD's

Появиха се цял рояк малко или много променени Knoppix LiveCD-та, това е т.н. knoppix ефект ;-)

– Gnoppix[47] - KDE е заменено с GNOME.
– Knoppix MiB Privacy Edition[48]
– ClusterKnoppix[49] Custom Debian Knoppix[50] - как да променим Knoppix image. knoppix-customize[51] - пакет улесняващ промяната на Knoppix image.

6.5. Gibraltar LiveCD

Съществуват и други проекти за LiveCD's, но целта им може да бъде по-специализирана. Такъв например е проекта http://www.gibraltar.at[52]. Това е базирана на Debian router/firewall дистрибуция инсталирана на CDROM, т.е. имаме LiveCD. Големите файлове могат да се съхраняват на хард-диска, а конфигурационните данни могат да се записват на флопи и да се съхраняват в RAM паметта по време на работа. Българския мирър за ISO файлове е http://mirrors.ludost.net/cd-images/linux/gibraltar[53]).

В официалния архив на Debian има пакети които могат да са ви от полза ако използвате това LiveCD. Такъв например е пакета `gibraltar-bootsupport`. С негова помощ можете да управлявате съдържанието на директориите `/etc` и `/var` внасяйки промени в тях, запазвайки ги и впоследствие може да възстановите от там. Поради тази причина този пакет трябва да се инсталира на master copy което ще бъде използвано като live filesystem на CDROM. Не инсталирайте този пакет на система която се зарежда от дял на някой хард-диск (т.е. на системата на която разработвате и създавате bootable CDROMs).

6.6. Публично достъпни Debian машини

Има и още един начин да погледнете в Debian система, без да е инсталирана на вашия компютър. Можете да погледнете в някоя публично достъпна система като тези обявени в тази новина http://www.debian.org/News/2003/20030102[54]. Чрез програмата Test Drive, Hewlett-Packard (HP) предлага публичен достъп до няколко машини на които има Debian GNU/Linux. Потребителите могат да получат акаунт, да надникнат в тези машини и да придобият представа как се представя Debian GNU/Linux на хардуердуерните архитертури от HP - поддържат се четири такива архитектури - Alpha, PA-RISC, IA-32 и IA-64 (може би това ще е уникална възможност да наблюдавате Debian GNU/Linux на платформа различна от познатото до болка x86 PC). Има инсталирани компилатори на тези платформи, така, че потребителите ще могат да тестват дали даден софтуер се компилира на тези платформи.

[43] http://www.debian.org/CD/jigdo-cd/#faq
[44] http://atterer.net/jigdo/
[45] http://tilix.slaveinostudios.com
[46] ftp://ftp.uni-sofia.bg/cd-images/linux/tilix/
[47] http://www.gnoppix.org
[48] http://www.bouissou.net/knoppix-mib/doc-html/Knoppix-Mib.html
[49] http://bofh.be/clusterknoppix/
[50] http://www.linuxgazette.com/issue87/sunil.html
[51] http://download.linuxtag.org/knoppix/knoppix-customize/ANNOUNCE.txt
[52] http://www.gibraltar.at
[53] http://mirrors.ludost.net/cd-images/linux/gibraltar
[54] http://www.debian.org/News/2003/20030102

Глава 7

Инсталация на Debian GNU/Linux Woody от CD-ROM на x86 PC

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 CD-ROM диска на Knoppix (`/usr/local/bin/knx-hdinstall`).

Следва описание на инсталационната процедура от CD-ROM на Debian GNU/Linux Woody върху x86 PC. Може би това е най-често срещаната ситуация при начинаещите потребители, но това далеч не е единствения метод за инсталиране на Debian.

7.1. Стартиране на инсталационния процес

Първо да укажем на компютъра да зарежда от CD-ROM-а при стартиране:

Включваме компютъра и още в началото докато се инициализира BIOS-а (Basic Input/Output System) в долният край на екрана се появява надписът **Press DEL to enter Setup**. Натиснете клавиша **Delete** докато се вижда надписа, за да влезете в менюто с настройките на **BIOS**. Ако сте с марков компютър (IBM,Dell,HP и т.н) то най-вероятно тази клавишна комбинация няма да е валидна за вас. Опитайте например с **F1** или **F2**, понякога може да се наложи и да упорствате. В самото меню навигацията се осъществява с клавишите със стрелките и Enter (изписано е на самото меню). След като сте влезли в менюто на BIOS-а изберете опцията **BIOS FEATURES SETUP** (или подобно). В новопоказалото се меню отидете на **Boot Sequence** или мястото, от където се указва подредбата на устройствата, от които зарежда компютъра. Там задайте първо да зарежда от CD-ROM-а (или **First Booting Device** да е **CD-ROM**). Сложете CD-ROM диска с Debian Woody в CD-ROM устройството След това натиснете **Esc**, за да отидете отново в главното меню и там изберете **SAVE AND EXIT SETUP**. Потвърдете с **Y(es)** и **Enter**. Компютърът ще се рестартира и ще опита да зареди първо от CD-ROM устройството.

Компютърът зарежда от CD-ROM и се стига до **boot:** запитването. Ако натиснете **Enter** ще започнете инсталация на Debian с **kernel** (ядро) версия **2.2.20** (считано от някои за старо). Ако искате по-модерната версия **2.4** въведете на **boot:** bf24 (можете да разгледате наличните ядра с клавиша **F3** и да получите още разнообразна информация с клавиши от **F1** до **F10**).

В тази примерна инсталация ще продължим с ядро версия **2.4**, така, че на **boot:** задаваме **bf24**. След натискането на **Enter** указаното ядро бива заредено и първият въпрос, които се показва пред нас е за избор на език на който да протече инсталацията.

7.2. Choose The Language

Тук се указва на какъв език да продължи инсталацията. В този пример ще изберем *[en] (English)*. Показва се следният въпрос:

Choose language Variant

Избираме *English(United States)*

Следват ***Release Notes***.

Просто натиснете Enter :-)

Ето ни в главното инсталационно меню на Debian GNU/Linux…

7.3. Debian GNU/Linux Installation Main Menu

Навигацията се осъществява със "стрелките"на клавиатурата и **Enter**. Нека да започнем работа, сега ;-). Изберете от менюто (маркирайте със стрелките и натиснете Enter):

7.3.1. 1.Configure Keyboard

Изберете **qwerty/us**

7.3.2. 2.Partition a Hard Drive

Select Disk Drive

Ако имате един IDE твърд диск най-вероятно ще бъде изписано **/dev/hda**. Ако е например /dev/sda - това означава, че сте късметлия и разполагате със SCSI твърд диск.В този пример ще приемем, че има само един - натискаме Enter. Ако обаче имате повече от един твърд диск ще трябва да зададете на кой ще инсталирате Debian.

LILO Limitations

Enter

Note on additional space for the ReiserFS Journal

Enter

… зарежда се програмата **cfdisk**. CFDISK е еквивалента на страшната команда fdisk в DOS - помните я нали? Разбира се и тук има команда fdisk, но повярвайте ми - работите ли веднъж със cfdisk - ще забравите за fdisk. Навигацията е със стрелките <- и -> за менюто долу, а с другите две в таблицата с дяловете (partition table) на твърдия диск. **Enter** селектира маркираната опция/дял. Ето как изглежда, ако предположим, че твърдия диск Ви е с големина 20GB и има два дяла (означени под Windows като C: и D:)

cfdisk 2.11u
Disk Drive: /dev/hda
Size:20560412672 bytes
Heads:255 Sectors per Track:63 Cylinders:2499

Name	Flags	Part. Type	FS(File System)	Type	[Lablel]	Size(MB)
hda1	Boot	Primary	Win95	FAT32		6259.45
hda2		Primary	Win95	FAT32		14295.56

[Bootable] [Delete] [Help] [Maximize] [Print]
[Quit] [Type] [Units] [Write]

Поради всевъзможните различия в големината на твърдите дискове и разделението им на дялове, не може да се даде пример, валиден за всеки един отделен случай. За това най-добре разгледайте **[Help]** секцията на **cfdisk**. И все пак, за да можем да продължим примерната инсталация ще приемем, че PC-то не е виждало що е то GNU/Linux - има два Primary (главни) дяла (като на схемата на cfdisk по-горе). (ако преди това сте инсталирали на компютъра си някоя друга GNU/Linux дистрибуция и дяла й все още е на твърдия диск, Debian ще го засече и ще Ви предложи да използвате вече съществуващия дял, без да е нужно да правите нов. По този начин си спестявате частта Partition a Hard Drive и продължавате с останалите стъпки.)

Сега да продължим по сценария - един 20GB твърд диск с два дяла под Windows… За да създадем дял за Debian ще трябва да изтрием втория дял (на първия евентуално се намира операционната система Windows, която не желаем да затрием.. поне за сега ;-)) и от освободилото се място (Free Space както го показва cfdisk) да направим няколко нови дяла - един за Windows, един за GNU/Linux и един swap дял (играе ролята на виртуална RAM памет, само че доста по-бавна, тъй като е създадена на твърдия диск). Маркираме с помощта на стрелките втория дял с име (Name в cfdisk) hda2. След като е маркиран дяла hda2 избираме от менюто на cfdisk опцията **[Delete]** (така изтриваме втория дял и мястото, което той е заемал се освобождава като Free Space; **имайте предвид, че при това изтриване ще изгубите всичката информация съхранявана на втория дял**; ако сте задали само **[Delete]** не се безпокойте, промените не се записват докато не се зададе **[Write]** от менюто на cfdisk. Така че можете просто да дадете **[Quit]**, без това да се отрази по някакъв начин на информацията на Вашия твърд диск.) В нашия пример ще предположим, че всички ценни данни от втория дял са били предварително прехвърлени на първия. Така втория дял е изтрит и е освободено мястото, което той е заемал - точно 14295.56MB. Сега да създадем новите дялове: първо Windows дяла - селектираите Free Space-а от таблицата на cfdisk и от менюто изберете **[New]**. Трябва да укажете колко MB желаете да е дяла, например 9000MB. Въвеждате числото 9000 и натиснете Enter. Докато е маркиран новонаправения дял изберете **[Type]** от менюто. Показва се таблица с различните файлови системи (FS), които могат да бъдат указани с cfdisk и съответстващия им код (номер). Ако в долния край на екрана пише *Press any key to continue* натиснете един произволен клавиш. Показва се краят на таблицата и следният текст: *Enter filesystem type:82*. Тук се въвежда кодът съответсващ на файловата система, която искаме да укажем за новия дял. Най-вероятно ще е зададен кодът **82**, съответстващ на файловата система за Linux - **ext2**. Но ние желаем да зададем дял за Windows, за това ще въведем съответстващият код - **0B** (нула B). Новият дял е готов. Сега по същата схема да направим дял и за Debian - маркиране на Free Space, **[New]**, задаване на големината на дяла - 5000MB, после **[Type]** - код **82**. И за swap дяла - **[New]**, задайте цялото оставащо място (по принцип и 100-200MB са предостатъчни), **[Type]** - код **83**. Това е само пример как се работи с cfdisk. Можете да си поиграете с дяловете на Вашия твърд диск, но без да задавате **[Write]** освен, ако не сте сигурни, че искате да запазите новосъздадения ред в таблицата с дялове. Така след като сме си свършили работата с дяловете и искаме да запазиме новата подредба избираме **[Write]** и отговаряме на въпроса *Do you want (to) write the partition table to disk?* с *Yes*, за да бъдат записани промените. След това **[Quit]** и отиваме на:

7.3.3. 3.Initialize and Activate a Swap Partition

Scan for Bad Blocks

Задайте No, освен ако Вашият твърд диск е от по–старичките или просто искате да бъде направена проверката за лоши блокове.

Are you sure?
Yes

7.3.4. 4.Initialize a Linux Partition

Choose Filesystem Type
Ако нямате представа за какво става въпрос, просто изберете **ext2** ;-)
Select Partition
Тук изберете дяла, който създадохте с cfdisk. От примера - **/dev/hda3:Linux native**
Scan for Bad Blocks?
Както при swap-а.
Are you sure?
Yes ;-) След това се създава файловата система и се появява въпросът:
Mount as the Root Filesystem?
Yes

7.3.5. 5.Install Kernel and Driver Modules

Found a Debian CD-ROM
Yes и изчакайте да свърши копирането.

7.3.6. 6.Configure Device Driver Modules

Note about loaded drivers
Enter
Select Category
Ако не сте сигурни просто изберете **Exit(Finished. Return to previous menu)**

7.3.7. 7.Configure the host name

Choose the Hostname
Тук задавате името на машината Ви или така напечения hostname. По подразбиране е debian.

7.3.8. 8.Install the Base System

Select Installation Media
cdrom:CD-ROM drive . . . ако имате повече от едно CD-ROM устройство ще се появи следния въпрос:
Select CD-ROM drive
Ако не знаете кое устройство да зададете, изберете едно и потвърдете с Enter. И след като на следващия въпрос:
Please insert the CD-ROM
натиснете Enter се появи: ***Mount failed*** значи сте избрали грешния CD-ROM (!диска с Debian Woody трябва да е в едно от CD-ROM устройствата!). Не се притеснявайте — дайте пак Enter и се връщате обратно в главното инсталационно меню. Там пак изберете ***Install the Base System*** като продължите по същия път, само че сега задайте правилното CD-ROM устройство.
Select Archive path
Enter . . . започва инсталацията на най-важните пакети, тоест Base System-а. Ако всичко е протеклно нормално, пак ще се озовете в ***Installation Main Menu***.

7.3.9. 9.Make System Bootable

Select Archive path
Препоръчително е да изберете **/dev/hda:Install LILO in MBR**.
Other bootable partitions
Изберете **Include all into the menu**. Така ще можете да зареждате всички операционни системи, инсталирани на компютъра Ви.
Securing LILO
Enter

7.3.10. 10.Make a Boot Floppy?

В тази част от инсталацията имате възможност да създадете дискета, от която евентуално можете да зареждате Debian-а си.

Change Disk

Ако искате да направите подобна дискета, сложете във флопито една (!ако има някаква информация на нея, то тя ще бъде загубена!) и натиснете Enter. Ако ли не — само натиснете Enter и на изскочилия

Problem

просто натиснете Enter. Това по никакав начин не пречи на инсталацията Ви. Ето ни обратно в ***Installation Main Menu***.

7.3.11. 11.Reboot the System

Ще рестартираме системата.

Reboot the system?

Ако има някакви дискети или CD-ROM-и, извадете ги от съответните устройства и задайте Yes. Reboot…

След рестартирането се появява LILO Boot Menu-то. От тук избирате коя операционна система (OS) да бъде заредена. Избираме Linux и Enter. Започва се конфигурирането…

7.4. Debian System Configuration

Enter

7.4.1. Time Zone Configuration

Is the hardware clock set to GMT?

Ако имате няколко различни OS на компютъра по-добре изберете No, защото може да се получи една хубава каша…

What area do you live in?

None of the above FIXME: виж малко по-надолу

Select a city or time zone:

Изберете **UTC**. FIXME. Препоръчително е тъй като се намирате на 2h време от GMT, да изберете България/София, или просто GMT+2. Имайте предвид, че дори и да изберете град със същата часова зона, daylight saving time (лятното часово време) не е същото. Имайте го предвид ако си настроите Кайро, Истанбул или нещо от сорта, че все пак не е София ;-)

7.4.2. Password setup

Shall I enable md5 passwords?

Yes

Shall I enable shadow passwords?

Yes

Enter a password for root:

Въведете парола за **root** (това е най-важния потребител на системата или още администратора; за него няма ограничения в действията - може да отвори/изтрие/модифицира всеки файл/директория на системата)

Re-enter password to verify:

Въведете отново същата парола за потвърждение.

Shall I create a normal user account now?

За създаване на обикновен потребителски акаунт, който се използва обикновено вместо **root** акаунта (използването на **root** по време на irc сесии, разглеждане на сайтове и тем подобни се счита за рисково относно сигурността на системата)

Enter a username for your account:

Въведете потребителско име за новия акаут.

Enter a username for your account:

"Въведете пълно име за новия потребител"… може да въведете и произволен текст, а може и нищо да не въвеждате - само да натиснете Enter.

Enter a password for the new user:

Паролата… и след това още веднъж…

… ето ни пак в ***Debian System Configuration***.

Shall I remove the pcmcia packages

Yes (в този пример се предполага, че към системата няма pcmcia устройства, както би било с някой най-обикновен компютър за домашна употреба)

Do you want to use a PPP connection to install the system?

No

7.4.3. Apt Configuration

Choose the method apt should use to access the Debian archives

За сега задайте само **cdrom**. Ако диска не е бил поставен в CD-ROM-а — поставете го.

Enter CD-ROM device file:

Само поставете диска и натиснете Enter. Диска бива сканиран и следва въпроса:

Scan another CD

. . . дали желаете да се сканира друг диск. Ако разполагата само с диска, с който извършихте инсталацията изберете No. Ако имате още Debian дискове, можете да зададете Yes и те да бъдат сканирани, за да знае системата и за намиращите се в тях пакети.

Add another apt source?

Засега No. Оставете това за по-късно, запомнете командата apt-setup, тя ще ви пести copy&paste ;-)

Use security updates from security.debain.org?

За сега No. По-късно като се вържем към Интеренет и ще нагласим тези настройки.

Ето ни обратно в ***Debian System Configuration***.

Run tasksel?
No
Run dselect?
No
Do you want to continue?
Yes
Do you want to erase any previously downloaded deb files?
Yes
Press enter to continue
Enter. Следва конфигурирането на exim:

I can do some. . . bla bla. . .

Enter. За да не се занимавате с конфигурацията на exim, можете да зададете 5 и да натиснете Enter. Ако не - четете менютата. . .

Thank you for choosing Debian

Честит Ви новичък Debian Woody :). Сега можете да влезете в системата като root или "обикновения"потребител, който създадохте по време на инсталацията. Няколко подробности: ако желаете да рестартирате машината, въведете следната командата като root:

```
# shutdown -r now
```

Същият ефект можете да постигнете със популярната клавишна комбинация **Ctrl+Alt+Del**.

За изключване на компютъра:

```
# shutdown -h now
```

За повече информация относно **shutdown**(8). За информация относно някоя команда:

```
$ man команда
```

Глава 8

Разработка на официалния debian-installer и други installer(s)

8.1. Официалният debian-installer

До Woody 3.0 официалния инсталатор се наричаше `boot-floppies`, за следващото стабилно издание Sarge вече има нов проект (или по-точно голямо разширение на предния) който се нарича `debian-installer`. CVS хранилище на двата проекта можете да намерите на cvs.debian.org[1]. Официалната страница на новия инсталатор можете да намерите на http://www.debian.org/devel/debian-installer/[2].

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 и ще бъде официалния инсталатор за Sarge.

8.2. *pgi*: The Progeny Graphical Installer

Този инсталатор има собствен сайт[3], както и пакет `pgi`.

А както и хранилище на alioth.debian.org/projects/pgi[4].

Progeny[5] реши да предостави Debian 3.0 Woody i386 installer images, базирани на PGI[6] 1.0.1 (свободен лизенз). ISO image (има и bg mirror[7]) 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.

Има и снимки[8] 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[9]
- The Discover Hardware Detection System[10], която е включена и в Debian като `discover`;
- `autoinstall`: Progeny Debian auto-installation system
- `autoinstall-i386`: Progeny Debian auto-installation system — i386-specific files

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

- официален сайт[11]

[1] cvs.debian.org
[2] http://www.debian.org/devel/debian-installer/
[3] http://hackers.progeny.com/pgi/
[4] alioth.debian.org/projects/pgi
[5] http://www.progeny.com
[6] http://archive.progeny.com/progeny/pgi/
[7] http://mirrors.ludost.net/cd-images/linux/debian/pgi/
[8] http://hackers.progeny.com/pgi/screenshots/
[9] http://hackers.progeny.com/pgi/guide.html
[10] http://hackers.progeny.com/discover/doc/guide.html
[11] http://www.informatik.uni-koeln.de/fai/

– 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. Това 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.

8.4. 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[12] and http://sisuite.org[13].

`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/[14]

`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.

8.5. Replicator

`replicator`: 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 клъстери, класни стаи и всякакви идентични помежду си компютри, това наистина е най-бързият метод на инсталация.

Който и източник на инсталация с даден инсталатор да ползвате, все ще получите стандартен Debian Base (т.е. стандартна базова инсталация на Debian).

[12] http://systeminstaller.sourceforge.net
[13] http://sisuite.org
[14] http://www.systemimager.org/download/

Глава 9

Първоначално запознаване с Debian

9.1. Процес на стартиране на Debian GNU/Linux

Когато включите компютъра си, първото нещо, което той прави, е да провери дали всичко е наред с хардуера. След това програма, наречена "bootstrap loader търси boot-sector (сектор за начално зареждане). Boot-секторът е първият сектор от твърдия диск и в него има малка програма, която може да зарежда операционни системи. Тази програма не може да да бъде по–голяма от 512 байта. Boot секторът може също да се намира на дискета или CD. Когато bootstrap-програмата намери boot-сектор, го зарежда в паметта и се изпълнява програмата, която зарежда операционната система. Обикновено, когато на компютъра е инсталиран GNU/Linux, тази програма е `LILO` (**LI**nux **LO**ader). Debian използва `LILO`. Освен `LILO` можете да използвате и по-малко известния `grub` (друга програма за зареждане на операционни системи).

Когато компютърът зареди boot-сектор на нормална GNU/Linux система, той всъщност зарежда част от `LILO`, наречена "first stage boot loader (първо ниво)". Неговата работа е да зареди "second stage boot loader (второ ниво)". Второто ниво Ви запитва каква операционна система искате да се зареди.

Ако изберете Linux, се зарежда ядрото (обикновено `/vmlinuz`).FIXME! Ядрото всъщност е самата операционна система. То отговаря за комуникацията на софтуера с хардуера. Когато ядрото се зареди напълно, първата програма, която изпълнява, е `/sbin/init`. За подробности може да погледнете **init**(8) man-страницата. Просто напишете:

```
$ man 8 init
```

или само

```
$ man init
```

Ако обаче не намери `/sbin/init`, ядрото (говорим за ядра, версия 2.4.x) търси `/etc/init`, `/bin/init`. Ако и тези не съществуват, ядрото изпълнява шел `/bin/sh`. Ако обаче и /bin/sh не съществува, системата не се стартира, а вместо това виждате следното съобщение: "`No init found. Try passing init= option to kernel.`".

Ако получите такова съобщение, можете да стартирате ядрото, задавайки следните параметри на `LILO`:

```
LILO: Linux init=/bin/bash
```

По този начин казвате на ядрото да не се изпълнява `/sbin/init`, а `/bin/bash`, което ще Ви осигури шел (shell) веднага след зареждане на ядорото в паметта. Имайте предвид, че системата няма да се държи както обикновено (след малко ще разберете защо). Най-простият пример за това е, че не можете да видите процесите на системата (ако се чудите защо — защото `/proc` файловата система не е монтирана. За да я монтирате просто напишете:

```
mount /proc
```

На практика можете да замените `/sbin/init` с каквато решите програма, но не е препоръчително, освен ако **наистина** знаете какво правите :-).

Работата на `init` е да прочете файла `/etc/inittab` и в зависиост от съдържанието му да изпълни определени скриптове след зареждане на ядрото, наречени **rc** скриптове. Те имат за цел да стартират определени услуги (програми) при стартиране на системата (например графичната среда, уебсървъра, демона за водене на отчетни файлове, виртуалните терминали и други) и да ги спират при изключване на системата. Когато се стартират тези скриптове, системата може да премине в различни нива (runlevels), а те са: еднопотребителски режим, многопотребителски режим, спиране на системата или рестартиране. Нивата на изпълнение са описани в `/etc/inittab`:

```
# Runlevel 0 is halt. (спиране на системата)
# Runlevel 1 is single-user. (еднопотребителски режим)
# Runlevels 2-5 are multi-user. (многопотребителски режим)
# Runlevel 6 is reboot. (рестартиране на системата)
```

Ако искате да смените режима, в който системата по принцип влиза при стартиране — търсете ред в `/etc/inittab`, подобен на този:

```
id:ниво:initdefault:
```

Стойностите на "ниво"могат да бъдат 1, 2, 3, 4 или 5. В противен случай системата няма да се стартира.

Еднопотребителският режим е по-специален. Когато системата се стартира в този режим, тя всъщност иска паролата на root-потребителя и му предоставя шел, ако е вярна. В този режим не се отварят виртуални терминали и никой друг не може да влезе в системата. Този режим се използва главно, когато нещо в системата не е наред и трябва да се оправи или при сериозна промяна в системната конфигурация като реорганизиране на дяловете, форматиране на файлови системи и въобще мащабни операции, а също и при провеждане на тестове върху системата.

Когато една Debian-система се стартира, тя преминава в runlevel 2 (многопотребителски режим). Тогава се изпълняват скриптовете в директория `/etc/rc2.d/`, които пък сами по себе си са символни връзки (за повече информация прочетете man страницата на **ln**(1)) към скриптовете в `/etc/init.d/`. В `/etc/init.d/` се намират всички **rc** скриптове. Различните нива имат директории `/etc/rc`**`ниво`**`.d/`. Така че, ако искате да промените скриптовете, които се изпълняват при стартиране на системата — отивате в директория `/etc/rc2.d/`.

За по–любознателните — кодът, който определя коя програма да се изпълни след зареждане на ядрото, се намира във файла `/usr/src/linux/init/main.c` (последните няколко реда).

9.2. Вход и изход от системата

9.2.1. Първоначална идентификация

Когато стартирате своята Debain-система, първото, което трябва да направите след като системата е заредила успешно, е да се идентифицирате. Debian ще очаква да направите това, като ви предостави login prompt, който изглежда примерно така:

```
Debian GNU/Linux testing/unstable shodan tty1
shodan login:
```

При вас разбира се ще изглежда малко по-различно (най-малкото, заради името на хоста), а също така е възможно и този процес при вас да се извършва в графичен режим, вместо текстово като при мен, при положение че системата ви е конфигурирана с XWindow (kdm, gdm или xdm).

След въвеждането на потребителското име, ще трябва да се идентифицираме и с парола:

```
Debian GNU/Linux testing/unstable shodan tty1
shodan login:manchev
Password:
```

След успешната идентификация, Debian ще ни предостави своя prompt, който при мен изглежда така:

```
manchev@shodan:~$
```

Важно е да отбележим, че описанията и примерите, които ще даваме и обясняваме, в случая засягат работата на Debian през конзола, така че ако използвате графична среда, сега е моментът да отворите терминална емулация на конзолата.

Преди да се захванем с най-основите команди, които може да изпълнявате, трябва да обърнем внимание на два основни аспекта. След като вече сте получили оторизация за използването на системата, може да се наложи да оставите компютъра без надзор, което е опасно от гледна точка на сигурността. Освен това, трябва да знаете как правилно да изключвате системата. И така, когато трябва да стане от работното си място, добра идея е да се изключите от системата, за да не може някои да използва вашето име и вашите права в нея с недобронамерени цели. За изключване от системата напишете на промпта:

```
manchev@shodan:~$logout
```

Това ще ни върне в стартовото състояние и на екрана ще се изведе частта, очакваща идентификация:

```
Debian GNU/Linux testing/unstable shodan tty1
shodan login:
```

Когато искате да изключите компютъра изобщо, трябва да изпълните друга командата — shutdown. Важно е да се отбележи, че това е задължителна стъпка, понеже директното щракване на копчето крие рискове от загуба на данни — Linux поддържа кеш спрямо дисковете си, като времето на опресняване е от порядъка на 30 секунди. Спрете ли системата направо от захранването, рискувате загуба на информация, която все още не е прехвърлена към дисковата система.

9.2.2. shutdown

Командата **shutdown**(8) се използва за изключване на системата чисто и безопасно. Когато изпълните командата shutdown всички включени в системата потребители (локални или отдалечени) получават нотификация, че системата прекратява работа. Освен това се поставя и забрана за включване на нови потребители. Следващата стъпка, която **shutdown**(8) предприема е да изпрати сигнала `SIGTERM` до всички работещи към момента процеси. Това дава възможност на програмите, които го разпознават да приключат нормално работата си (например даден текстови редактор разбира, че системата се готви за изключване и може да запази данни от файла, който в момента се редактира чрез него). По-нататък **shutdown**(8) се свързва с процеса **init**(8), като заявява смяна на **runlevel**(8), като стойността по подразбиране е runlevel 1. (FIXME: Да се обясни) Има няколко важни параметъра, които можем да укажем на командата, а именно:

-r Параметърът изисква от системата да се рестартира, след като приключи изпълнението на процесите, които shutdown изисква.

-h Блокира всякакъв достъп към системата. Обикновено точно този параметър се използва, когато смятаме да изключим изцяло компютъра.

Освен това, командата shutdown изисква и един задължителен параметър time който казва кога да се извърши изключването. Той може да се замени с константата "now което инструктира shutdown да започне процеса незабавно.

От казаното по-горе можем да конструираме два от основните формати на извикване на shutdown, а именно:

– Преди да изгасим компютъра:

```
manchev@shodan:~$shutdown now -h
```

– Или когато искаме просто да го рестартираме:

```
manchev@shodan:~$shutdown now -r
```

Ако сега се опитам да изпълня една от двете команди на моята система, това което ще се случи няма да е очакваното:

```
manchev@shodan:~$shutdown now -r
bash: shutdown: command not found
```

Bash (FIXME:Две думи за Bourne и C shell) ни съобщава, че не е намерил изисканата от нас команда. Това се случва по простата причина, че ние като обикновен потребител нямаме в пътя си достъп до `/sbin` директорията от файловата система, която съдържа в команди достъпни единствено за администраторите на системата. Дори и да отидем в самта директория и да изпълним командата там (което е постижимо), ще видим нещо от рода на:

```
shutdown: you must be root to do that!
```

Ако помислим логично — тази защита е съвсем уместна. Ние не бихме искали всеки потребител на когото сме дали някакъв достъп до системата да има възможност да я изключва и рестартира. Което ни води до следващата крачка: Ако не използваме системата като root, но въпреки това администрираме системата, как да извършим операциите по изключването и рестартирането й?

Тук е момента да споменем още една основна команда — **su**(1). Тя ни дава възможност да сменим своята идентификация и в частност да се превърнем `superuser`, който има права да извърши дадената операция.

9.2.3. su

Както вече споменахме, su ни дава възможност да се превърнем в друг потребител по време на нашата собствена сесия. Ако извикаме командата без име на потребител, чиито права искаме да придобием, тогава по подразбиране su предполага, че искаме да се превърнем в root. Важно е да обърнем внимание на параметъра " който се използва за да придобием и обкръжението, което потребителят в който се превръщаме би получил ако се включи директно към системата. Например, ако се превръщаме в нормален потребител, стойността на на променливата (FIXME: DOLLAR)PATH ще се промени на `/bin:/usr/bin` или ако се идентифицираме като super user, нейната стойност ще бъде `/sbin:/bin:/usr/sbin:/usr/bin`.

Друг итнересен параметър на командата е с"чрез който направо изпълняваме дадена команда с правата на потребителя, в който се превръщаме.

От тук насетне имаме два подхода: Първият е да се превърнем в super user и да изпълним командата **shutdown**(8), а вторият — да извикаме **su**(1) и направо да поискаме той да изпълни **shutdown**(8) с права на `superuser`.

Първият подход изглежда така:

```
manchev@shodan:~$su -
Password:
```

След като желаем да се превърнем в `root`, системата изисква да напишем паролата за акаунта на този потребител. След като я въведем правилно ще получим нов промпт:

```
shodan:~#
```

Обърнете внимание в промяната на знака $ в #. Именно заради тази промяна винаги можем да знаем като какъв потребител използваме конзолата — нормален когато промпта е $ или `root` когато промпта е #.

Изпълняваме **shutdown**(8):

```
shodan:~#shutdown now -h
Broadcast message from root (tty1)
The system is going down for system halt NOW!
INIT: Switching to runlevel: 0
INIT: Sending processes the TERM signal
INIT: Sending processes the KILL signal
Stopping periodic command scheduler: cron.
Stopping initernet superserver: inetd.
Stopping OpenBSD Secure Shell server: sshd.
Saving the System Clock time to the Hardware Clock...
Hardware Clock updated to Tue Aug 19 13:23:37 EEST 2003.
Stopping deferred execution scheduler: atd.
Stopping kernel log daemon: klogd.
Stopping system log daemon: syslogd.
```

```
Sending all processes the TERM signal... done.
Sending all processes the KILL signal... done.
Saving random seed... done.
Unmounting remote filesystems... done.
Deconfiguring network interfaces.... done.
Deactivating swap... done.
Unmounting local filesystems... mount: proc already mounted
done.
flushing ide devices: hda hdc
Power down.
```

Е, сега е безопасно да изключите компютъра си, ако той сам не се е изключил разбира се.

Другият вариант беше да напишем направо:

```
manchev@shodan:~$su -c "shutdown now -h"
Password:
```

И след въвеждането на паролата, щеше да последва същата поредица от действия (FIXME: Да не забравя за sudo).

Когато говорим за изключване или рестартиране на системата, трябва да обърнем внимание и на командите **halt**(8), **reboot**(8) и **poweroff**(8). Какво правят те е видно от самите им имена. Да вземем за пример действието на **halt**(8). Ако изпълним **halt**(8) и не се намираме в **runlevel**(8) 0 или 6 (тоест — нормално функционираща система) Debian ще изпълни `shutdown -h` (респективно `shutdown -r`, ако извикаме **reboot**(8)). С една дума може да гледаме на **halt**(8) като синоним на `shutdown -h`. В по-старите версии на `sysvinit` не беше разрешно **halt**(8) да се извиква директно. Така или иначе употребата на **shutdown**(8) е по-елегантния начин за изключване/рестарт и най-добре се придържайте към него.

След като видяхме как правилно можем да се включим, изключим и как безопасно да спрем системата, вече е време да започнем да се забавляваме с основните команди на Debian GNU/Linux.

9.3. Разглеждане на файлове с ls

Първото нещо, което веднага ни идва на ум е да видим какво има в системата, след като веднъж сме се идентифицирали в нея и притежаваме определни права. Тук е моментът да се запознаем с командата **ls**(1). Това е една от основните команди на Linux. Това, което тя прави е да покаже списък (list — ls, връзката е ясна) от файлове. Когато изпълняваме **ls**(1) без параметри по подразбиране се показва списъка с файловете от текущата директория. Нека опитаме:

```
manchev@shodan:~$ls
manchev@shodan:~$
```

Малко неочаквано, но разбираемо. Като нов потребител на системата в моята директория все още няма никакви файлове. Все пак, за да проверим дали **ls**(1) върши изобщо нещо, нека я изпълним с параметър "/".

```
manchev@shodan:~$ls /
bin   cdrom  etc     home    lib         mnt  proc  root-n  tmp  var
boot  dev    floppy  initrd  lost+found  opt  root  sbin    usr
```

Така нещата изглеждат по-нормални. Това което накарахме **ls**(1) да ни покаже е основната директория на нашата система. Основната директория е пределно позната концепция от множество операционни системи. Тя е коренът, от който дървовидно се разраства цялата файлова система на Debian.

Видяхме, че можем да предаваме като параметър на **ls**(1) коя част от файловата структура искаме да разгледаме. Можем например да опитме да видим директориите на регистрираните в системата потребители. Всеки акаунт на потребител който добавяме към системата автоматично получава и собствена директория в нея, където той може да държи файловете си. Тези директории по подразбиране се създават в `/home`. Да погледнем:

```
manchev@shodan:~$ls /home
manchev
manchev@shodan:~$
```

Тук виждаме само една единствена директория — моята собствена. Това е така, понеже в момента аз съм единствения регистриран ползвател на моя Debian.

Ако все пак сте любопитни, къде е директорията на потребителя `root`, в който се превърнахме за да изключим системата, отговорът е, че `root` има собствена директория извън `home`, която е разположена в основната директория. Можете да я видите под името `root` на изхода от изпълнението на командата `ls /` по-горе.

Нека все пак отбележим, че не всички акаунти имат свои директории в `home`. Има различни служебни потребители, за които няма смисъл да се създават такива. Например `mail`, чиито права се използват от частта по обслужване на пощенската система.

Освен простото изписване на имената, **ls**(1) може да показва още много допълнителна информация. Командата е изключително мощна и богата на параметри, както междувпрочем и повечето команди на Debian.

Нека вземем за пример параметъра F". Да изпълним:

```
manchev@shodan:~$ls -F /
bin/   cdrom/  etc/     home/    lib/        mnt/  proc/  root-n*  tmp/  var/
boot/  dev/    floppy/  initrd/  lost+found  opt/  root/  sbin/    usr/
```

Сега получаваме повече информация за отделните файлове в основната директория. Параметърът F"кара **ls**(1) да постави допълнителен идентификатор до всеки запис, който ни показва какъв е неговия тип. Знакът "/"е указател за директория, "*"показва че записът е изпълним файл, "@"обозначава символни линкове (FIXME: За symlinks/hard links да се каже) , -"се прикрепя към сокети и "за FIFO (FIXME: За FIFO да се каже).

Има още няколко основни параметъра, които е добре да разгледаме. Един много важен и често използван параметър на **ls**(e) l който кара командата да изведе списъка в по-разширен формат. Да опитаме:

```
manchev@shodan:~$ls -l /
total 100
drwxr-xr-x    2 root     root         4096 Aug 17 18:19 bin
drwxr-xr-x    2 root     root         4096 Aug 19 08:53 boot
drwxr-xr-x    2 root     root         4096 Aug 17 17:37 cdrom
drwxr-xr-x    9 root     root        24576 Aug 19 18:25 dev
drwxr-xr-x   49 root     root         4096 Aug 19 19:38 etc
drwxr-xr-x    2 root     root         4096 Aug 17 17:37 floppy
drwxrwsr-x    3 root     staff        4096 Aug 19 19:38 home
drwxr-xr-x    2 root     root         4096 Aug 17 17:37 initrd
drwxr-xr-x    5 root     root         4096 Aug 17 22:14 lib
drwx------    2 root     root        16384 Aug 17 16:32 lost+found
drwxr-xr-x    2 root     root         4096 Feb  8  2002 mnt
drwxr-xr-x    2 root     root         4096 Aug 17 17:37 opt
dr-xr-xr-x   34 root     root            0 Aug 19 18:25 proc
drwxr-xr-x    9 root     root         4096 Aug 19 20:12 root
-rwxr-xr-x    1 root     root            0 Aug 18 19:18 root-n
drwxr-xr-x    2 root     root         4096 Aug 17 18:19 sbin
drwxrwxrwt    5 root     root         4096 Aug 19 20:11 tmp
drwxr-xr-x   12 root     root         4096 Aug 18 00:07 usr
drwxr-xr-x   13 root     root         4096 Aug 17 17:37 var
```

Изходът от изпълнението започва с реда "total <блокове>". Стойността на тези блокове показва размера на дисковото пространство, което заемат файловете в директорията. По подразбиране един блок се равнява на 1024 байта, но тази стойност може да се променя с подхящи параметри на **ls**(1).

По-нататък за всеки отделен запис се извежда информация както следва: права, брой на твърдите връзки на файла или ако е директория — броя на твърдите връзки на файловете в нея, име на притежателя на файла, име на групата, размер и времето на последна модификация. Смисълът на тези данни ще разгледаме по-късно, когато видим употребата на различните команди по раздаване на права, смяна на собственост на файлове и т.н.

Последното, което ще спомена за **ls**(1) е параметъра h". Той предоставя същата информация но във вид, който е удобен за четене от потребителя. Разбира се, чудесно е да знаем размера на всеки файл до байт, но е къде по-лесно за възприемане, ако той се конвертира автоматично до KB, MB или GB (в зависимост от размера си), защото така е по-удобен за възприемане.

Като последен пример за **ls**(1), нека изпълним командата с два параметъра — l"и h". Ето как изглежда изхода от изпълнението:

```
manchev@shodan:~$ls -l -h
total 100K
drwxr-xr-x    2 root     root         4.0K Aug 17 18:19 bin
drwxr-xr-x    2 root     root         4.0K Aug 19 08:53 boot
drwxr-xr-x    2 root     root         4.0K Aug 17 17:37 cdrom
drwxr-xr-x    9 root     root          24K Aug 19 18:25 dev
drwxr-xr-x   49 root     root         4.0K Aug 19 19:38 etc
drwxr-xr-x    2 root     root         4.0K Aug 17 17:37 floppy
drwxrwsr-x    3 root     staff        4.0K Aug 19 19:38 home
drwxr-xr-x    2 root     root         4.0K Aug 17 17:37 initrd
drwxr-xr-x    5 root     root         4.0K Aug 17 22:14 lib
drwx------    2 root     root          16K Aug 17 16:32 lost+found
drwxr-xr-x    2 root     root         4.0K Feb  8  2002 mnt
drwxr-xr-x    2 root     root         4.0K Aug 17 17:37 opt
dr-xr-xr-x   34 root     root            0 Aug 19 18:25 proc
drwxr-xr-x    9 root     root         4.0K Aug 19 20:12 root
-rwxr-xr-x    1 root     root            0 Aug 18 19:18 root-n
drwxr-xr-x    2 root     root         4.0K Aug 17 18:19 sbin
drwxrwxrwt    5 root     root         4.0K Aug 19 20:11 tmp
drwxr-xr-x   12 root     root         4.0K Aug 18 00:07 usr
drwxr-xr-x   13 root     root         4.0K Aug 17 17:37 var
```

Нека обърнем внимание на още един факт. В **ls**(1), както и в повечето Debian команди може да комбинирате параметри и да ги подавате заедно към командата, за която се отнасят. Например, вместо "ls -l -h"можехме да напишем "ls -lh".

Командата **ls**(e) много мощен инструмент. Тя може да извежда различна информация в множество различни формати. Да сортира, да филтрира, да навлиза и показва съдържанието на директории, да поддържа контролни знаци. Изобщо — каквото ви дойде наум. Това е валидно за почти всички команди на Debian — изключително богатство от функции, достъпни чрез определени параметри и гъвкаво управление на изпълнението.

Разбира се, ние не можем да опишем всички параметри и възможности на всяка команда, която разглеждаме. Това, което можем да покажем обаче е как да получите помощ за всяка една от тях. Това е следващата тема, която ще разгледаме.

Преди това, нека спомена и още една команда, с аналогични функции на **ls**(1). Това е командата **dir**(1). Когато говорим за системата за помощ на Debian ще разберете как да получите повече информация за нея (ако разбира се ви интересува). Така или иначе, **ls**(1) и **dir**(1) вършат едно и също, въпреки че имат някои разлики.

По-голямата част от потребителите на Debian, които познавам се придържат твърдо към **ls**(1). Както и аз самият. Моят личен е избор е продиктуван от две неща :

1. свикнал съм с ls, защото в първия Linux който ползвах нямше dir
2. dir изгражда лошия навик да се стряскам когато по инерция в DOS конзола напиша "dir -la"и видя празна диркетория.

9.4. Когато се нуждаем от помощ

9.4.1. man, whatis, apropos, info. . .

man(1) е най-често използвания инструмент за бързо достъпване на помощна информация за командите в Debian. Всъщност с man можем да видим доста повече информация, касаеща също така и системни извиквания, файлови формати и различни програми. Най-простото използване на **man**(1) е когато подаваме един единствен аргумент — името на командата (функцията, програмата и т.н.) за която искаме да получим информация. От там насетне, задачата на **man**(1) е да открие страницата съдържаща съответния текст и да ни я покаже.

Нека опитаме да получим помощна информация за една друга важна команда в Debian — **cat**(1).

```
manchev@shodan:~$man cat
```

Екранът ще се изчисти и **man**(1) ще изведе страницата с помощна информация за cat (FIXME: да се добави за разликата, когато man работи с **cat**(1) и с **less**(2): 1. чрез cat може да се чете само "надолу"; 2. чрез less може да се използва PageUp, PageDown) , която ще изглежда така:

```
Reformatting cat(1), please wait...
CAT(1)                          User Commands                          CAT(1)
NAME
       cat - concatenate files and print on the standard output
SYNOPSIS
       cat [OPTION] [FILE]...
DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.
       -A, --show-all
              equivalent to -vET
       -b, --number-nonblank
              number nonblank output lines
       -e     equivalent to -vE
       -E, --show-ends
              display $ at end of each line
       -n, --number
--More--
```

Когато текста предоставян ни от man не се събира на един единствен екран (както е в по-голямата част от случаите), **man**(1) показва първата страница, а на последния ред извежда инверсно "`--More--`". Това ни показва, че `man` очаква команда, след като прочетем изведената информация. Това, което правим в повечето случаи е:

- натискаме space, което извежда следващата страница
- натискаме Enter, което скролира текста с един единствен ред
- натискаме q, което спира работата на man и ни връща в промпт

(FIXME: Като се говори за more, да се ref-не тук)

Нека обърнем внимание на още един факт: **man**(1) не само съдържа описание на информацията, а я държи и в структуриран формат. Когато изпълняваме **man**(1) и получаваме помощ, в горната лява и дясна част от изхода на командата, виждаме изписано името на параметъра който сме подали (респективно, за който получаваме информация - в нашия случай — **cat**(1)). В скоби до него виждаме стойност, която ни показва от кой раздел получаваме информация. Ето и списък на отделните раздели:

1. Програми и команди на shell-а
2. Системни функции (които се предоставят от kernel-а)
3. Функции, реализирани в различни библиотеки
4. Файлове със специално предназначение (например тези в /dev)
5. Файлови формати и конвенции
6. Игри
7. Други
8. Команди свързани с администрирането на системата (в повечето случаи, достъпни само за root)
9. Специални функции на kernel-а

Ключов момент е, че **man**(1) претърсва за подадения параметър разделите в точно определения ред и извежда първата страница, която открие. Възможно е параметъра, който подаваме на **man**(1) да същесвува едновременно в няколко различни раздела. Именно затова **man**(1) резрешава да зададем в явен вид кой точно от разделите да се претърси за параметъра, без да се поглежда в другите части на документацията. Форматът на командата в този случай е:

```
man [име_на_раздел] какво_търсим
```

Ако **man**(1) не открие търсенето в посочения раздел, той не претърсва останалите. Ако зададем, например:

```
manchev@shodan:~$ man 1 cat
```

това ще работи коректно, защото така или иначе информацията за **cat**(1) е разположена в частта с програмите и различните `shell` команди. За сметка на това, ако зададем друга стойност на раздела (например 2) ще получим следното:

```
manchev@shodan:~$ man 2 cat
No manual entry for cat in section 2
See "man 7 undocumented" for help when manual pages are not available.
```

Веднага изниква въпросът дали има възможност да видим в кой раздел е разположена информацията която търсим и дали за параметъра, съгласно който искаме информация от **man**(1), не съществуват записи в повече от един раздел. Например, да предположим, че ни трябва описание на функцията `clear`, която е част от библиотеката `curses` (поддържа функции за работа с екрана, които са терминално независими и следователно много полезни). За съжаление, ако просто изпълним `man clear` ще видим информация за командата (а не функцията) `clear`. Командата **clear**(1) се използва за изчистване на екрана и няма нищо общо с това, което на нас ни трябва.

В този случай може да ни помогне командата **whatis**(1). Нека се върнем за момент към примерния изход от изпълнението на `man cat` показа по-горе. Още в самото начало на страницата, в частта `NAME` имаме името на командата, за която сме потърсили помощ (cat) и веднага след нея — кратко нейно описание.

Това, което **whatis**(1) прави е да претърси всички раздели за страници отговарящи на името, което му подадем като параметър. После **whatis**(1) извежда списък, съдържащ съответно името, раздела и споменатото кратко описание. Да се върнем към случая с `clear` и да използваме `whatis`, за да си помогнем:

```
manchev@shodan:~$whatis clear
clear (1)          - clear the terminal window
clear (3ncurses)   - clear all or part of a curses window
```

Ето защо получаваме не това, което ни трябва - командата **clear**(1) е разположена в по-преден раздел от функцията clear и съответно е първото нещо, което **man**(1) ни показва. Сега, след като се уверихме че това, което ни трябва е в третия раздел, можем да го достъп по вече познатия начин:

```
manchev@shodan:~$man 3 clear
```

Което ще ни даде точно това, което и търсехме.

Следващия въпрос, който е логично да си зададем е: След като имаме възможност да търсим по имена и да виждаме описание, не можем ли да направим търсенето така, че да бъде по думи споменати в описанието и така да получим самите имена, които после да разгледаме с **man**(1)?

Разбира се, че е възможно. И тук на помощ ще дойде командата **apropos**(1). Нека вземем следния пример — искам да намеря такава команда, която да конвертира аудио диск в MP3 файлове, които да запиша в home директорията си и после да прослушвам с някакъв MP3 player.

Няма смисъл да си губим времето с **whatis**(1). Да опитаме чрез **apropos**(1):

```
manchev@shodan:~$apropos mp3
cdda2mp3 (1)         - extract audio CD audio tracks and encode them
```

Звучи точно като това, което търсим. Представете си, колко щяхме да налучкваме с **whatis**(1). От тук насетне можем веднага да погледнем:

```
manchev@shodan:~$man cdda2mp3
```

и да видим как правилно да използваме **cdda2mp3**(1). По въпроса за player-а можем да постъпим аналогично:

```
manchev@shodan:~$apropos player
xmms (1)             - an audio player for X.
xlsfonts (1x)        - server font list displayer for X
xprop (1x)           - property displayer for X
xwud (1x)            - image displayer for X
```

На пръв поглед, първият запис май ни върши работа. Въпросът е, че зад него са се наредили още няколко, които нямат никакво отношение към нашия случай. Това се е случило, защото **apropos**(1) търси стринга и в самите думи и е открил "player"в "displayer". В случая не е критчино, но когато имаме нужда от търсене точно на това което сме подали като параметър, можем да използваме опцията е"(exact — точно съвпадение). Да опитаме:

```
manchev@shodan:~$apropos -e player
xmms (1)             - an audio player for X.
```

Така нещата стоят по-нормално. Разбира се, в нашия случай не беше толкова критично, но е полезно да имаме е"предвид, когато apropos ни върне неподходящи резултати.

Освен гореспоменатите начини за набавяне на помощна информация, също така може да се използва и `info`. GNU проектът разпространява голяма част от ръководствата си в "info формат който може да бъде прочетен, използвайки "info reader (FIXME:четец)". За повече информация относно `info` можете да изпълните командата:

```
$ info info
```

FIXME: Да се добави повече инфо за info, например, за "браузването"в info и т.н.

Приключвайки с темата за man трябва да запомним едно основно правило: **man**(1), **whatis**(1) и **apropos**(1) няма да ни помогнат много, когато инструмента или командата която ни е нужна не е инсталирана. Ако на вашата система нямате инсталиран `xmms`, `apropos -e player` просто ще ви каже:

```
player: nothing appropriate
```

Още повече, направете си труда да видите какво връща `apt-cache search player`, за да видите колко много и разнообразни player-и могат да бъдат инсталирани на вашия Debian. Така че, ако не намерите инструмента който ви трябва по някой от описаните по-горе начини.

9.4.2. info

Друга основна команда, която ни е нужна когато търсим помощна информация е **info**(1). Тя се използва за четене на документи в Info формат. GNU разпространява голяма част от ръкводствата си в Info формат, затова е добре да можем да се справяме с инструмента за четене на този формат, а именно - Info.

Сега да опитаме нещо интересно — да извикаме **info**(1) и да поискаме от него да ни покаже собствената си документацията. Ще процедираме точно като с **man**(1) — предаваме един единствен параметър, показващ коя документация искаме да видим:

```
manchev@shodan:~$  info info
```

Ето какво ще видим на екрана, след изпълнението на "info info"

```
File: info,  Node: Top,  Next: Getting Started,  Up: (dir)
Info: An Introduction
*********************
   Info is a program, which you are using now, for reading
documentation of computer programs.  The GNU Project distributes most
of its on-line manuals in the Info format, so you need a program called
"Info reader" to read the manuals.  One of such programs you are using
now.
   If you are new to Info and want to learn how to use it, type the
command `h' now.  It brings you to a programmed instruction sequence.
   To learn advanced Info commands, type `n' twice.  This brings you to
`Info for Experts', skipping over the `Getting Started' chapter.
* Menu:
* Getting Started::             Getting started using an Info reader.
* Advanced Info::               Advanced commands within Info.
* Creating an Info File::       How to make your own Info file.
--zz-Info: (info.gz)Top, 24 lines --Top----*** Tags out of Date ***-------------
Welcome to Info version 4.6. Type ? for help, m for menu item.
```

Документацията във формат Info е разделена на различни части (nodes), според темата коят засягат. Да погледнем погледнем първия ред (или хедъра) на този node.Там изписан node-а, в който се намираме, както и този който следва след него. Виждаме, че различните раздели, които info ни показва са логически свързани един с друг. След като се запознаем със съдържанието на node "Top"(който в нашия случай е интродукция), следва да преминем към частта "Getting started".

Из различните страници, които info извежда можем да навигираме с клавишите "n"(next - следваща) и "p"(previous - предходна). Ако натиснем "n"пред нас ще се изведе раздела "Getting started". Респективно ще се промени и хедъра, така че секция "Next"ще бъде "Advanced Info а ще се появи и секция Prev — "Top която показва кой е разделът, предшестващ "Getting started".

Да натиснем "p за да се върнем в "Top". Ако погледнем долната част на екрана, можем да разберем, че тази страница е по-дълга от един екран. Ако информацията влизаше в рамките на екрана, долу щеше да бъде изписано "`--All----`"вместо "`--Top----`". За придвижване между страниците на текущия раздел използваме клавишите "PageUp"и "PageDown". Можем вместо тях да употребяваме и space и backspace, но ако с някой от тях достигнем границите на страницата, те автоматично ще ни прехвърлят към следващия(респективно предишния) раздел.

Освен линейното придвижване (чрез "n"и "p"), в info можем да използваме и менюта, чиито елементи водят директно към други раздели. Менюто можем да разпознаем по надписа "* Menu:". Всеки следващ елемент след "* Menu:"е връзка към друг раздел. В нашия случай (раздел "Top") имаме меню, както и определен брой елементи закачени към него:

```
* Menu:
* Getting Started::             Getting started using an Info reader.
* Advanced Info::               Advanced commands within Info.
* Creating an Info File::       How to make your own Info file.
```

Основно има два подхода за навигация из менюто. Единият е просто да се придвижим до елемента със стрелките на клавиатурата (или Tab - елемент по елемент) и да натиснем Enter. Другият, макар и да изглежда по-неинтуитивен е много мощен и бърз.

Другата възможност е да натиснем "m което ще ни вкара в режим за работа с менюто. В долната част на екрана ще видим:

```
Menu item:
```

В този режим можем да напишем името на елемнта от менюто, а натискането на Enter ще ни прати в раздела, асоцииран с този елемент. Освен това:

- когато работим в този режим, можем да използваме клавиша Tab за автоматично дописване. Например, ако напишем "get"и натиснем Tab, info автоматично ще го превърне в "Getting Started". Ако има повече от една възможност, info ще ни го покаже над статусния ред, за да конкретизираме заявката си;
- напускането на режима за навигация става чрез комбинацията Ctrl-g.

Последното, което ще спомена е, че когато навлизаме навътре в дървото от node-ове можем да използваме клавиша "u който ни връща обратно нагоре по структурата.

Казаното до тук е една съвсем малка част от възможностите на **info**(1). Ние можем да обхождаме различните раздели по име, да навигираме из подтеми по индекс, да редактираме info документи, да извършваме сложни търсения и още много други неща. Всички тези възможности може да научим направо от документацията на самото **info**(1) (`info info`). Не трябва да забравяме, че ако търсим нещо по-специфично, което не сме открили в **man**(1), добре е да погледнем в **info**(1). Неговите раздели в повечето случаи са доста по-пълни и по-подробни.

9.5. Навигация във файловата система

Когато разгледахме командата **ls**(1) видяхме част от файловата структура на Debian. Тя е организирана в дървовидно — концепция позната от множество операционни системи. Преди да видим как можем да се придвижваме из различните части на структурата (директориите), нека се запознаем с една друга команда - **pwd**(1). Тя е изключително лесна и предназначението й е да покаже името на директорията, в която се намираме. Да опитаме:

```
manchev@shodan:~$ pwd
/home/manchev
manchev@shodan:~$
```

В случая, командата показва директорията на потребителя като текуща, понеже до сега не сме навигирали извън нея.

Ако погледнем описанието на **man**(1) за **pwd**(1) ще видим, че командата има две опции — "`--version`"и "`--help`". Нека опитаме някоя от тях:

```
manchev@shodan:~$ pwd --help
-bash: pwd: --: invalid option
pwd: usage: pwd [-PL]
manchev@shodan:~$
```

Изглежда "`--help`"не е валидна опция за **pwd**(1). Същото ще се случи, ако опитаме и с "`--version`". Това е така, понеже повечето shell-ове като **bash**(1) имат собствена имплементация на **pwd**(1), която се различава от описанията на man.

В случая с `bash`, man няма да ни даде информация за реализацията на командата. За сметка на това, ако погледнем в info (info bash) ще видим описание на pwd. От там ще разберем, че pwd на `bash` има две валидни опции: P"и L които контролират дали извежданото име на директория да съдържа или не символни линкове.

След като веднъж знаем къде се намираме, нека опитаме да сменим текущата си директория. Това можем да постигнем чрез командата `cd`.

9.5.1. cd

`cd` променя текущата директория и приема един аргумент — име на директорията, в която да се преместим (и тя да стане текуща). Ако извикаме `cd` без аргументи, тя по подразбиране ни връща в нашата `home` директория.

Да опитаме да се преместим в основната директория и после да се върнем в нашата собствена.

```
manchev@shodan:~$ cd /
manchev@shodan:/$ pwd
/
manchev@shodan:/$ cd
manchev@shodan:~$ pwd
/home/manchev
manchev@shodan:~$
```

Видяхме как заедно се използват `cd` и `pwd`. При втората промяна на директория не използвахме параметър. В този случай:

```
manchev@shodan:/$ cd
```

и

```
manchev@shodan:/$ cd /home/manchev
```

се държат еквивалентно, така че съкратихме писането на аргумента. Командата `cd` е изключително проста. Всъщност, освен аргумента за директория, тя приема само още два параметъра — P"и L които са аналогични на тези от **pwd**(1). Не трябва да забравяме, че ако все пак търсим помощна информация за `cd` трябва да погледнем чрез `man bash` или `info bash`, защото `cd` е също вградена команда за shell-а.

Докато се "разхождаме"из директориите виждаме, че при смяната на текущата директория bash prompt-ът се променя, отразявайки новата директория в която се намираме. Ако бяхме изпълнили `cd /home`, той щеше да добие вида:

```
manchev@shodan:/home$
```

В този случай бихме могли да се запитаме, дали тази функционалност не прави **pwd**(1) излишна. Е, определено не е така. Първо, командата може вместо да бъде веведена от нас, да бъде изпълнена в някакъв script (вътрешния език на shell-а) и тогавата нещата стоят малко по-различно. Освен това, дадена Debian система може да е така конфигурирана, че да не извежда подобна информация на prompt-а си, а ние трябва винаги лесно да можем да разберем къде се намираме.

Друго, което следва да се запитаме, защо тогава, когато сме в собствената си директория (`/home/manchev`) prompt-ът изглежда по друг начин, вместо да ни показва името на директорията. В този случай знакът -"има специално значение и то е да отговаря на собствената директория на конкретния потребител. Смисълът на тази функционалност ще видим след малко.

Има още една особеност на `cd`, която принципно важи и за много други команди : тя позволява подаваната чрез аргумент директория да бъде описана абсолютно или относително. Какъв е смисълът? Идеята е, че ако навигираме до директория `/home`, след което пожелаем да се озовем в `/home/manchev` имаме два подхода:

```
manchev@shodan:/home$ cd /home/manchev
```

и

```
manchev@shodan:/home$ cd manchev
```

Във втория случай, когато аргументът не започва със знака "/"(който обозначава основната директория на дървото) командата предполага, че директорията която указваме е относителна спрямо текущото ни местоположение. Това предимство не е толкова явно в този случай, но ако си представим, че се намираме например в `/usr/X11R6/lib/modules/drivers` и искаме да се озовем в `/usr/X11R6/lib/modules/drivers/linux` спестява доста писане. Нещо повече — `cd` разрешава да указваме не само име на директория, а и цялостен път, който също може да бъде абсолютен (от основната директория) или относителен (от текущото ни местоположение). В нашия пример ако в директория `manchev` имаше да кажем директория `backup`, вместо:

```
manchev@shodan:/home$ cd manchev
manchev@shodan:~$ cd backup
```

можем да използваме:

```
manchev@shodan:/home$ cd manchev/backup
```

Тук идва реда да се върнем към знака -"и да видим колко полезен може да ни бъде, когато навигирме из файловата система на Debian.

Да предположим, че се намираме в директория `/etc` и искаме да достигнем до поддиректория на нашата собствена (пак ще използваме измислената backup). От казаното до сега следва, че имаме два похода:

```
manchev@shodan:/etc$ cd
manchev@shodan:~$ cd backup
```

или:

```
manchev@shodan:/etc$ cd /home/manchev/backup
```

А ето как можем да използваме -"и да си съкратим дългото писане или двете последователни изпълнения на `cd`:

```
manchev@shodan:/etc$ cd ~/backup
```

Сега, нека приключим с `cd` и да видим как може да направим измислената директория `backup` съвсем реална.

Глава 10

По-user-friendly десктоп

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

10.1. 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
```

10.2. 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++*. Използва се свой собствен описателен език за конструиране на менютата.

- 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.

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 frontents for various window-managers in **/etc/menu-methods** to create startup files for the window managers (or **pdmenu**(1)). 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/`

Друга подобна меню система е pdmenu[8], която е пакетирана и налична и в Debian архива като едноименния пакет `pdmenu`. Въз основа на **pdmenu**(1) може да ползвате **apteryx**[9], който обединява **apt-get**(8) и **apt-cache**(8) и техните опции поднасяйки ги чрез **pdmenu**(1).

Подобен на `apteryx` инструмент извикващ **apt-get**(8) и **apt-cache**(8) е **apt-iselect**[10]. Той се основана на http://www.engelschall.com/sw/iselect/[11] (дебианския пакет е `iselect`) и ако последния не е инсталиран при първото стартиране на `apt-iselect` ще бъде инсталиран автоматично.

[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
[8] http://www.kitenet.net/programs/pdmenu/
[9] http://www.internetlab.org/apteryx/
[10] http://www.rot13.org/~dpavlin/apt-iselect.html
[11] http://www.engelschall.com/sw/iselect/

10.3. Debian Usability Research

Debian Usability Research[12] е под-проект на проекта Debian[13] занимаващ се с използваемостта на софтуера специфичен за операционните системи на Debian. Основните цели са:

- състоянието на package metadata: кое е добре, какво липсва и как може да бъде добавено
- техники за package searching и browsing, така щото да се превъзмогнат проблемите при постоянното нарастване броя на пакетите
- как debian package subsets адрасиращи специфични нужди може да помогнат да се изгради дистрибуция която по-добре адресира потребностите на различните общности и как те по-лесно могат да бъдат създавани
- как да се обработват usability bug reports

Освен това се работи и по Debian Package Manager brainstorming session[14]. Целта е да се набира информация относно пакетните менажери и как те да бъдат използвани и подобрявани. Това проучване ще се използва като основа за по-нататъшни анализи и заключения относно дизайна на пакетните менажери в Debian.

Търсене на информация за пакетите и файловете, които предоставят те, може да стане по много начини, включително и през http://packages.debian.org[15]. Ето едно алтернативно предложение на Erich Schubert, който анонсира[16] нова версия на неговия Package Browser[17], 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[18].

На практика вече има и пакети като `debtags` и `synaptic-debtags`

[12] http://deb-usability.alioth.debian.org/
[13] http://www.debian.org
[14] http://deb-usability.alioth.debian.org/pkgman-brainstorm.html
[15] http://packages.debian.org
[16] http://lists.debian.org/debian-devel-0303/msg01371.html
[17] http://debian.vitavonni.de/packagebrowser/
[18] http://deb-usability.alioth.debian.org/debtags/

Глава 11

Интернационализация, Локализация, Българизация

11.1. Въведение

Подробен и достъпно написан наръчник[1] има в официалната документация на Дебиан[2]. Освен това можете да разгледате архивите на списъка debian-i18n@lists.debian.org[3]. След което, например, може да изпълните

```
# 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://lists.zadnik.org/cgi-bin/mailman/listinfo/debian[5], 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.

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 кирилизиране[6] на Linux-BG.org[7]. Практически единственото пълно и всепризнато решение на проблема за българизирането на ГНУ/Линукс, независимо от дистрибуцията, е пакетът bglinux[8] на Антон Зиновиев[9]. 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.

---

[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
[5] http://lists.zadnik.org/cgi-bin/mailman/listinfo/debian
[6] http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&cmd=cat&cid=cyr
[7] http://www.Linux-BG.org/
[8] http://lml.bas.bg/~anton/linux/bglinux.html
[9] http://lml.bas.bg/~anton/

11.2. 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 кирилизиране на конзолата се описва много по-гъвкав начин за кирилизиране 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.

11.3. *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 променяте 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[10] pam_env[11]. 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
```

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

11.4. *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

11.4.1. *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`.)

11.5. *XFree86*: Графична среда

11.5.1. *XKB*: Клавиатура

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

- The XKB Configuration Guide[12] (част от XFree86 4.3)
- How to further enhance XKB configuration[13] (част от XFree86 4.3)
- An Unreliable Guide to XKB configuration[14] (от Doug Palmer[15])
- X Keyboard Extension[16] (от Иван Паскал[17])

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	*XkbLayout*	– `bg`
Please select your keyboard variant	*XkbVariant*	– `phonetic` – `bds`
Please select your keyboard options	*XkbOptions*	– `grp:shift_toggle` – `grp:ctrl_shift_toggle` – `grp:caps_toggle` – `grp:ctrl_alt_toggle` – `grp:alt_shift_toggle` – `grp:menu_toggle`

Списък на всички опции, задавани чрез *Option*, може да се намери в `/etc/X11/xkb/rules/xfree86.lst`.

XF86Config: Редактиране на конфигурацията на X

Редактирайте файла `/etc/X11/XF86Config-4`, като копирате секцията InputDevice за клавиатурата преди реда

```
### BEGIN DEBCONF SECTION
```

ако го има. След това в копието се добавят или променят следните настройки:

```
Section "InputDevice"
        ::
        Option  "XkbLayout"     "bg"
        Option  "XkbVariant"    "phonetic"
        Option  "XkbOptions"    "grp:alt_shift_toggle,grp_led:scroll"
EndSection
```

Различните възможности за стойностите на опциите могат да се видят от предишната секция.

Ако ползвате XFree86 версия 4.3 или по-голяма, трябва да употребите малко по-друг синтаксис, който позволява повече от две подреждания (layouts). Примерите по-долу засягат само *XkbLayout* и *XkbVariant*, защото останалите неща са същите.

```
# едновременно фонетична и БДС наредба
        Option "XkbLayout" "us,bg,bg"
        Option "XkbVariant" ",phonetic,bds"
# или само фонетична
        Option "XkbLayout" "us,bg"
        Option "XkbVariant" ",phonetic"
```

[12] http://www.xfree86.org/current/XKB-Config.html
[13] http://www.xfree86.org/current/XKB-Enhancing.html
[14] http://www.charvolant.org/~doug/xkb/html/index.html
[15] http://www.charvolant.org/~doug/
[16] http://www.tsu.ru/~pascal/other/xkb/
[17] http://www.tsu.ru/~pascal/

setxkbmap: Различна подредба за отделен потребител

Командата **setxkbmap**(1) позволява конфигуриране на клавиатурата от команден ред или скрипт. Например:

За фонетична подредба с превключване на кирилица/латиница с десен Alt:

```
# setxkbmap bg phonetic_enhanced grp:toggle,grp_led:scroll
```

За БДС подредба подредба с превключване на кирилица/латиница с десен Alt:

```
# setxkbmap bg bds_enhanced grp:toggle,grp_led:scroll
```

(Не забравяйте, че от XFree86 4.3 нататък има малко по-друг синтаксис, както е показано по-горе.)

Такъв скрипт може да се използва при начално зареждане на потребителска X сесия, променяйки глобалните настройки за подредба на клавиатурата. Файлът `~/.xsession` (в Дебиан вместо `~/.xinitrc` трябва да се използва `~/.xsession`) е подходящ за тази цел. Забележете, че този файл може и да не се изпълни от display manager-и, различни от `xdm`, като `kdm` или `gdm` например. Използвайки горните променливи, командата изглежда по следния начин (в квадратни скоби се отбелязват части, които могат да се пропуснат):

```
# setxkbmap [ -layout XkbLayout ]
            [ -variant XkbVariant ]
            [ -option XkbOptions ]
            [ -compat XkbCompat ]
```

Като графични превключватели на подредбата на клавиатурата може да ползвате `kxkb`, `xxkbd` и др.

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.

11.5.2. Шрифтове

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

- Fonts in XFree86[18], от документацията на XFree86[19]
- XFree86 Font De-uglification HOWTO[20]

Имайте впредвид, 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[21]:

```
$ 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[22] от вида:

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

За търсене и разглеждане на шрифт използвайте xfontsel[23]. Една комбинация от двете команди, която използвам за Уникод[24] 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.)

[18] http://www.xfree86.org/current/fonts.html
[19] http://www.xfree86.org/
[20] http://feenix.burgiss.net/ldp/fdu/
[21] http://www.xfree86.org/current/xlsfonts.1.html
[22] http://www.xfree86.org/current/xfd.1.html
[23] http://www.xfree86.org/current/xfontsel.1.html
[24] http://www.unicode.org/

Препоръчвани пакети с шрифтове

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

- `xfonts-base`
- `xfonts-cronyx-{cp1251`[25]`,koi8r`[26]`}-{{75`[27]`,100`[28]`}dpi,misc`[29]`}`
- `xfonts-bolkhov`[30]`-{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[31] от секцията Basic Shell Features[32] в ръководството на Bash[33]. 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[34] на XFree86[35]. За тази цел се използва библиотеката FreeType[36]. 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[37] 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` с `-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 — XTT, 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.
7. Изпълнете командите:

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

[25] http://packages.debian.org/xfonts-cronyx-cp1251-75dpi
[26] http://packages.debian.org/xfonts-cronyx-koi8r-75dpi
[27] http://packages.debian.org/xfonts-cronyx-75dpi
[28] http://packages.debian.org/xfonts-cronyx-100dpi
[29] http://packages.debian.org/xfonts-cronyx-misc
[30] http://packages.debian.org/xfonts-bolkhov-75dpi
[31] http://www.gnu.org/manual/bash-2.05a/html_chapter/bashref_3.html#SEC27
[32] http://www.gnu.org/manual/bash-2.05a/html_chapter/bashref_3.html
[33] http://www.gnu.org/manual/bash-2.05a/
[34] http://keithp.com/~keithp/talks/
[35] http://www.xfree86.org/
[36] http://www.FreeType.org/
[37] http://fontconfig.org/

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[38]. 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.

11.5.3. *xfs*: Шрифтов сървър

Инсталирането на шрифтов сървър, като например `xfs` или `xfs-xtt`, е препоръчително[39]. 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"
```

[38] http://feenix.burgiss.net/ldp/fdu/

[39] https://listman.redhat.com/pipermail/roswell-list/2001-September/001816.html

11.6. Справяне с някои „упорити“ програми

11.6.1. 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[40] от Георги Данчев[41]. 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 „опаковка“*. 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

     ```
     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.

[40] http://linux-bulgaria.org/lug-bg-list/archive/2003/Jan/0260.html
[41] http://danchev.fccf.net/

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
```

11.6.2. KDE

С KDE и въобще приложенията разчитащи на библиотеката Qt (защото има и доста такива които се разработват и извън проекта KDE[42]) няма да имате никакви проблеми. Но все пак ето някои бележки, които може да са непълни. Ако имате някакви предложения, казвайте ги в http://lists.zadnik.org/cgi-bin/mailman/listinfo[43].

KDE изисква следните настройки: (май има и още неща[44]):

- 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[45] някои допълнителни действия.)

11.6.3. 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.

11.6.4. Midnight Commander

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

11.6.5. teTex

Българизирането на **teTeX** е описано в главата **Работа с LaTeX** 29

[42] http://www.kde.org
[43] http://lists.zadnik.org/cgi-bin/mailman/listinfo
[44] http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=343148460
[45] http://linux-bulgaria.org/lug-bg-list/archive/2002/Aug/0211.html

11.7. *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 упростява.

11.7.1. *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 XKB 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.

11.8. Какво още може да се направи...

11.8.1. Превод на сайта *http://www.debian.org*

Месец май 2003 беше обявено, че основната част от официалния сайт на Debian GNU/Linux[46] е преведена на български език. Ако предварително не е направена, след малка настройка[47] може съвсем комфортно да се разглеждат "българските"уеб страници на любимата свободна[48] операционна система.

– Достъп до официалния превод:

```
cvs -d :pserver:anonymous@cvs.debian.org:/cvs/webwml login
cvs -d :pserver:anonymous@cvs.debian.org:/cvs/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 **cvs checkout** за българския и за английския текст:

  – http://www.debian.org/devel/website/[49]
  – http://www.debian.org/devel/website/translating[50]
  – http://www.debian.org/devel/website/stats/bg.html[51]

– Ръководител: Тук[52] може да се види кой ръководи проекта за българския превод (в момента това е Румен Кръстев - rkrastev at obs dot bg ; той има *write access* до cvs.debian.org[53]). Ако сте забелязали грешки при превода или имате идея как да помогнете, най-добре е да влезете във връзка с него.

11.8.2. 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[54] - не е включен в официалната документация

– Стария и превод на APT-HOWTO[55] - 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.

11.8.3. Превод на официалния *debian-installer*

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

11.8.4. Поддържане 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 - 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. 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, защото това не е просто най-голямата дистрибуция на GNU/Linux, а е и най-голямата дистрибуция на операционна система (ОС) въобще - около 12 000 официални debian пакети към момента - това изобщо не е скромно. Решението е и ние българите (както всички останали) да се вместим някакси в тази интернационална и универсална дистрибуция на операционна система, защото да се опитваме да догоним възможностите на Debian започвайки от нулата (from scratch) е меко казано несериозно или неразбиране за какво се опитваме да говорим. Би било прекрасно някои хора да престанат да подскачат и да крещят: "О, тази дистрибуция така... онази пък така... на тази пише, че е направена в България, на онази пък, че е създадена специално за нас, на тази пише Advanced... на другата Professional... ". Загърбете рекламните трикове, точно така, както правете и с телевизионните реклами. Вместо това вкопчете се в някоя дистрибуция (разбира се след като опитате от всичко преди това), която Ви е по сърце и така докато наистина не откриете нещо генерално по-добро и аргументирано да го замените.

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

[46] http://www.debian.org
[47] http://www.bg.debian.org/intro/cn.bg.html#setting
[48] http://www.debian.org/intro/free
[49] http://www.debian.org/devel/website/
[50] http://www.debian.org/devel/website/translating
[51] http://www.debian.org/devel/website/stats/bg.html
[52] http://www.debian.org/devel/website/translation_coordinators
[53] http://cvs.debian.org
[54] http://debian.gabrovo.com/docs/maint-guide/
[55] http://danchev.fccf.net/docs/linux/apt-howto-bg/

- bgtex-v2, който го няма в официалната дистрибуция на teTex и естествено и в tetex пакетите които са в Debian архива.
- 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/[56] 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

[56] http://bgdebian.sourceforge.net/

Глава 12

Връзка към Internet

FIXME: В Debian GNU/Linux Internet Server[1] в лаконичен стил е описано конфигурирането на Интернет сървър. В Guide to IP Layer Network Administration with Linux[2] ще намерите добро описание на маршрутизиращите способности на Linux, както и на употребата на `iproute`. В Traffic Control HOWTO[3] се намира много добра документация за управление на трафика под Linux.

12.1. PPP връзка към Internet

12.1.1. *pppconfig*

Ако нямате инсталиран **pppconfig**(8), то поставете CDROM диска и го добавете с:

```
# apt-cdrom add
```

Инсталирайте пакета `pppconfig`.

```
# apt-get update && apt-get install pppconfig
```

Като root изпълнете командата:

```
# pppconfig
```

Ако не сте root в момента, можете да използвате командата:

```
$ su
```

и след това да въведете паролата на суперпотребителя. Забележете, че така наречения *prompt* се променя от

```
$
```

за обикновения потребител (най-често използваният символ, но може да се срещнат и други), на

```
#
```

за root.

След изпълняването на pppconfig се зарежда главното меню:

Main Menu
bla bla bla bla bla
bla bla bla bla bla

Create	Create a connection
Change	Change a connection
Delete	Delete a connection
Quit	Exit this utility

Избираме **Create**, за да създадеме нова връзка към някой доставчик на Интернет.

1.Provider name

Тук се въвежда името на доставчика, който ще бъде използван. По подразбиране това име е *provider* (това ще използваме и в този пример). Може да бъде използвано и друго име, и свързването към този доставчик ще се осъществява с командата:

```
#  pon име_на_конекцията
```

Затова е препоръчително конекцията (или връзката - connection), която ще използвате най-често, да се казва provider, за да може свързването да се осъществи само с командата pon. (!името на конекцията не трябва да съдържа интервали!)

2.Configure Nameservers (DNS)

Изберете **Dynamic** (върши работа в повечето случаи) — маркирайте с клавиша за интервал (Space) и след това чрез клавиша Tab отидете на **<Ok>** и потвърдете с Enter. (!за повече информация четете текста под заглавието на менютата!)

[1] http://documents.made-it.com/Debian_Internet_Server/Debian_Internet_Server.html
[2] http://linux-ip.net/
[3] http://linux-ip.net/articles/Traffic-Control-HOWTO/

3.Authentication Method for provider

Най-често това е **PAP**. Ако знаете със сигурност кой метод използва Вашият доставчик, изберете го.

4.User name

Изтрийте примерния текст и въведете Вашето потребителско име (user name).

5.Password

Изтрийте примерния текст и въведете Вашата парола (password).

6.Speed

Няма проблем да оставите вече указаната стоиност (115200). В някои случаи даже е по-добре.

7.Pulse or Tone

Тук се указва вида на набиране - пулсово или тоново. Повечето телефони използват тоново, така че ако не сте сигурни, че Вашият използва пулсово, маркирайте с помощта на клавиша Space опцията **Tone**. След това чрез Tab маркирайте **<Ok>** и Enter.

8.Phone Number

Изтрийте примерния текст и на негово място въведете номера на Вашия доставчик (!без тирета, например, 823746).

9.Choose Modem Config Method

Тук се указва на кой порт е свързан модема Ви. Под Windows наименованията са COM1, COM2. . . , а съответсващите им под Linux са ttyS0, ttyS1 (!имайте предвид, че Debian прави разлика между главни и малки букви, тоест между ttyS1 и ttys1!). Ако под Windows модема Ви е закачен за COM1, то под Linux можете да го намерите на /dev/ttyS0. Сега да зададем към кой порт е свързан модемът. Ако отговорите на това меню с **<Yes>**, ще бъде направена автоматична проверка, тоест Debian ще опита сам да открие модема Ви. Ако отговорите с **<No>**, ще имате възможност ръчно да зададете порта.

В този пример ще изберем **<Yes>**. След няколко секунди се появява следното на екрана:

Select Modem Port
( ) /dev/ttyS0
(*) /dev/ttyS1
( ) Manual

Debian е открил модема на /dev/ttyS1 (или COM2), което в нашия пример е правилният порт ;-) Чрез Tab отидете на **<Ok>** и натиснете Enter, за да преминете нататък.

Няма да разглеждаме случайте, когато към компютъра е свързан някои от "измислените"вътрешни модеми, за които е нужно да се инсталира допълнително софтуер, за да могат да работят. Ако притежавате такъв модем, по-добре спестете малко пари и си купете един нормален външен модем (ще си спестите и доста неприятности).

10.Properties of provider

След преминаване през гореизброените стъпки се стига до това меню. Тук са показани въведените за конекцията данни. Ако желаете да промените нещо от въведената информация, просто го селектираите и Enter. Така можете да модифицирате тези данни.

Ако изберете **Advanced Options**, ще имате достъп до някои по-интересни опции, като например опцията **Add-User - Add a ppp user**. Тук можете да добавите някой потребител, който по този начин ще добие правото да набира или спира конекциите към доставчиците на Интернет. По този начин не е нужно да сте root, за да наберете някоя конекция. За да добавите потребител, изберете опцията **Add-User** и въведете потребителското име, което желаете.

След като е добавен потребителя, със стрелката слезте надолу към опцията **Previous - Return to previous menu** и Enter. Върнахме се в **Properties of provider** и от там изберете **Finished - Write files and return to main menu**, за да бъде записана новонаправената конекция.

След натискането на Enter се озоваваме обратно в **Main Menu**. Ако желаете да създадете нова конекция, изберете **Create**. За модифициране на съществуваща конекция — **Change**, за премахване на конекция — **Delete** и за. изход от pppconfig — **Quit**. И сега, за да се свържете към Интернет трябва да изпълните командата:

```
# pon    (за конекцията provider)
```

или

```
# pon    име_на_конекцията
```

като потребител root, или добавения чрез **Advanced Options** потребител.

12.1.2. wvdial

Друга много добра програма за `dial-up` е `wvdial`. Инсталирайте го от CDROM диска, както `pppconfig`. В този пакет има самата изпълнима програма **wvdial**(1), програмата **wvdialconf**(1) за получаване на конфигурационния файл и самия конфигурационен файл **wvdial.conf**(5). Ето един кратък конфигурационен файл - **/etc/wvdial.conf**:

```
[Dialer Defaults]
# Серийният порт, на който се намира модема
Modem = /dev/ttyS1
# Скоростта на връзката, имайте предвид, че
# трябва да е равна на тази в /etc/ppp/options
Baud = 38400
# Инициализиращият стринг за модема; зависи от модела на
# модема, така че е добре да погледнете в документацията му
Init1 = ATX3
```

```
# Ако сте е pppd > 2.4.0
New PPPD = yes
# Телефонният номер, към който ще dial-up-вате
Phone = 9300904
# Вашето потребителско име
Username = YourUserName
# Вашата парола
Password = YourPassword
# При прекъсване автоматично ще опитаме отново
Auto Redial = yes
# ATDT за тонално, ATDP за пулсово набиране
Dial Command = ATDT
```

Не забравайте да проверите дали в `/etc/resolv.conf` сте добавили ключовата дума `nameserver` и след нея `IP` на работещ `DNS` сървър.

Изпълнявате като root:

```
# wvdial
```

12.1.3. kppp

За разлика от `pppconfig` и `wvdial`, които са конзолни програми, `kppp` е програма с графичен интерфейс и е доста лесна и интуитивна за настройване на Вашата `dial-up` връзка към `ISP`. Тук просто я споменаваме като представител на графичните аналози, а Вие със сигурност ще се справите с нея сами. Ако не можете да я намерите в менютата на графичната среда, която сте стартирали под **XFree86**, то стартирайте някой терминален емулатор, като `xterm`, `konsole`, `gnome-terminal` и т.н. и изпълнете:

```
# kppp
```

Не забравайте да проверите дали в `/etc/resolv.conf` сте добавили ключовата дума `nameserver` и след нея `IP` на работещ `DNS` сървър.

12.2. Ethernet връзка към Internet

За да можете да осъществите връзка по Ethernet, добре би било да имате основни познания по `TCP/IP`. За целта можете да прочетете например **The Linux Network Administrator's Guide, Second Edition**[4] (FIXME: link към pdf-а на български, пуснат от softpress) съсредоточите основно върху главите, които дават добра начална информация и представа за мрежовата инфраструктура на GNU/Linux.

От Debian GNU/Linux 2.2 Potato насам мрежовите настройки се пазят изключително в `/etc/network`, а самото пускане и спиране на мрежовата свързаност става с `/etc/init.d/networking`. Изключение правят само **PCMCIA (PC-Card** базираните мрежови карти. Нека първо обърнем внимание на `/etc/network/interfaces`, където става същинското конфигуриране на интерфейсите. Тук конфигурираме eth0 с IP адрес 192.168.1.120, мрежова маска 255.255.255.0 и шлюз по подразбиране 192.168.1.1; същевременно подсигуряваме автоматичното му вдигане с директивата `auto`:

```
auto eth0
iface eth0 inet static
        address 192.168.1.120
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
```

Понякога се налага да се изпълнят допълнителни команди преди вдигането на интерфейс с цел неговата по-специална конфигурация. Поради това `iface` частта поддържа клаузите `pre-up` и `post-down`. Както **interfaces**(5) обяснява, ролята на тези клаузи е следната:

Клаузата `pre-up` изпълнява командите, намиращи се непосредствено след нея преди даденият интерфейс да бъде вдигнат. Това е изключително удобно, ако например ви се налага да смените хардуерният (още наричан MAC) адрес на вашата мрежова карта. Това може да стане преди вдигането на съответният интерфейс, и се осъществява от следната команда (поставена някъде в `iface` блока - например непосредствено след редът `iface eth0 inet static` от примерът по-горе):

```
        pre-up ifconfig eth0 hw ether AB:CD:EF:AB:CD:EF
```

Така написана тази команда сменя хардуерният адрес на адаптерът, асоцииран с `eth0` на `AB:CD:EF:AB:CD:EF`. Това е изключително удобно, ако например ви се е наложило да смените мрежовата си карта, но ЛАН доставчикът контролира достъпът на машината по MAC адрес.

ВНИМАНИЕ! Смяната на MAC адрес, макар и в повечето случаи безобидна, може да доведе до сериозни мрежови проблеми в случай, че дублира вече съществуващ такъв адрес.

Клаузата `post-down` е подобна като действие на `pre-up` с разликата, че командите след нея се изпълняват след свалянето на даденият интерфейс.

И двете клаузи могат да се появят повече от един път за всеки интерфейс и изискват командите, които изпълняват, да завършат успешно. За повече информация вижте **interfaces**(5).

Ако вашият Интернет доставчик разполага с **DHCP** сървър, който автоматично снася на клиентите мрежови настройки, то в `/etc/network/interfaces` можете да напишете:

[4] http://www.tldp.org/LDP/nag2/index.html

```
auto eth0
iface eth0 inet dhcp
```

За конфигурирането на IP свързаност съществуват и други методи за конфигуриране на интерфейси, сред които bootp, ppp и wvdial, но за тях няма да говорим тук. За по-подробна информация относно възможностите за конфигуриране на **/etc/network/interfaces** може да погледнете в **interfaces**(5), а също **ifup**(8) и **ifdown**(8).

Във файла `/etc/network/options` също има някои интересни настройки. Например така се определя дали да се активира прехвърлянето на пакети между интерфейсите, като се установи във включено положение `/proc/sys/net/ipv4/ip_forward`:

```
ip_forward=yes
```

Така се установява т.нар "филтър на обратния път", чрез който се проверява дали изходния адрес на пакетите съответства с нашата маршрутизираща таблица и се подсигурява пакетите с точно този изходен адрес да получат отговор през мрежовия интерфейс, от който са влезли.

```
spoofprotect=no
```

С последната опция се установява защита срещу една доста популярна в миналото атака, наречена `syn flooding`. За целта трябва да имате необходимата поддръжка в ядрото.

```
syncookies=no
```

Разбира се, за да можете да достъпвате уеб, пощенски и всякакви други сървъри по техните имена, например `www.yahoo.com`, а не по адреси от вида `216.109.117.205`, ще е необходимо да редактирате файла `/etc/resolv.conf`:

```
search <домейн-име на вашия доставчик>
nameserver <IP на сървър за имена 1>
nameserver <IP на сървър за имена 2>
```

12.3. *iptables*: GNU/Linux като маршрутизатор

Една от честите употреби на GNU/Linux е да замести маршрутизатор Cisco или друг специализиран хардуер с евтино старо PC от класа на 486 или (по-добре) Pentium. Това в никакъв случай не значи по-малко възможности, а напротив — GNU/Linux като маршрутизатор е способен на почти всичко, което може да се появи в практиката.

Системата в ядрото, която се грижи за всички тези неща, се нарича Netfilter[5]. Тя се управлява чрез командата **iptables**(8) от пакета `iptables`. Повече информация може да намерите в сайта на Netfilter.

След като конфигурирате Netfilter с нужните ви правила, остава въпросът как те да се възстановяват при рестартиране на компютъра. За целта използвайте следната команда:

```
# /etc/init.d/iptables save active
```

FIXME: Това няма да работи под Sarge. В `/usr/share/doc/iptables/README.Debian.gz` има повече информация.

12.3.1. Примерни правила за филтриране на пакети

Следните прости правила са взети от SecurityFocus[6]. Те са само „скелет" с възможност да допълвате с още правила. Дадени са и две примерни правила.

```
INTIF=eth0                      # Вътрешна мрежа
EXTIF=eth1                      # Интернет
TCP_SERVICES="22,80"            # Разрешени услуги
# Изтриване на всички таблици
iptables -F INPUT
iptables -F FORWARD
iptables -F OUTPUT
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING
### Таблица INPUT ##########################################
# Подразбираща се политика.
iptables -P INPUT DROP
# Пакети от активни TCP връзки се разрешават.  Пакети-отговори също се
# разрешават, например отговор на DNS заявка (по UDP) или отговор на
# ping request (по ICMP).
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# Всеки от вътрешната мрежа, както и от локалния интерфейс, може да
# започне нова връзка.
iptables -A INPUT -i $INTIF -m state --state NEW -j ACCEPT
iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
# Тук е мястото за още правила.
```

[5] http://www.netfilter.org/

[6] http://www.securityfocus.com/infocus/unix?topic=fwrules

```
# Пример: Разрешава нови TCP връзки от Интернет към някой от
# $TCP_SERVICES на който и да е компютър от вътрешната мрежа.
iptables -A INPUT -i $EXTIF -m state --state NEW \
        -p tcp -m multiport --dport $TCP_SERVICES -j ACCEPT
# Хроникиране на отказаните пакети.
iptables -A INPUT -j LOG --log-prefix "FW_INPUT  "
### Таблица FORWARD ######################################
# Подразбираща се политика.
iptables -P FORWARD DROP
# Пакет от вътрешната мрежа, отиващ към Интернет.
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT
# За да се върне пакетът, обаче, трябва да е част от някоя активна
# TCP връзка.
iptables -A FORWARD -i $EXTIF \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
# Тук е мястото за още правила.
# Хроникиране на отказаните пакети.
iptables -A FORWARD -j LOG --log-prefix "FW_FORWARD  "
### Таблица OUTPUT #######################################
# Подразбираща се политика
iptables -P OUTPUT ACCEPT
### Таблица POSTROUTING ##################################
# Пример: Маскарадинг за отиващите към Интернет пакети.
iptables -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE
```

Глава 13

Периферни устройства

13.1. Принтери

FIXME: Има да се пълни тук.

13.2. Скенери

13.2.1. Подкарване на скенер Acer S2W 3300U

Понеже скенерът използва USB-порт, трябва при компилиране към ядрото да се добави `USB support`, а също така `USB-scanner support` като модул. След тези процедури трябва да бъде инсталиран SANE[1] - "Scanner Access Now Easy".

```
# apt-get install sane sane-utils
```

Препоръчително е да се инсталира и `libsane-extras`, където е включен и въпросният скенер:

```
# apt-get install libsane-extras
```

След това в `/etc/sane.d/snapscan.conf` трябва да редактирате реда, започващ с `firmware` и да опишете пътя до съответния `bin` файл. На snapscan.sourceforge.net[2] има таблица, в която е описано кой `bin` файл кога да се използва. За да разберете при Вас кой е, изпълнете последователно като root следните команди:

```
# modprobe scanner
# sane-find-scanner
```

Би трябвало да се получи нещо подобно:

```
found USB scanner (vendor=0x04a5, product=0x20de) at /dev/usb/scanner0
```

`Bin` файловете можете да намерите на диска към скенера или от тук[3]. Също така трябва да се укаже и на кой порт е закачен скенерът. Както се вижда в случая - `/dev/usb/scanner0` и съответно се описва.

С това се приключва с `/etc/sane.d/snapscan.conf`. Сега трябва да се инсталира и frontend към SANE. `xsane` е подходящо решение. Ако не Ви харесва, разгледайте и kooka, който е част от KDE.

```
apt-get install xsane xsane-common
```

Вече можете да използвате пълноценно Вашия скенер. Може да добавите в `/etc/modules.conf` следния ред, взет след изпълнението на командата `sane-find-sannner`:

```
options scanner vendor=0x04a5 product=0x20de
```

Ето и един адрес, на който е описано достатъчно добре как се подкарва този скенер и има доста интересни и полезни препратки http://www.acronymchile.com/scanner.html[4]

[1] http://www.mostang.com/sane/
[2] snapscan.sourceforge.net
[3] http://majesty.host.sk/Programs/Acer-bins/index.html
[4] http://www.acronymchile.com/scanner.html

Глава 14

Преобразуване на Woody към Sarge

Тук ще бъде описан процесът на преминаване към Sarge при положение, че току-що е инсталиран Woody. Понеже Sarge още не е издаден, тук се използва името на издание `testing`, което в момента на писането е бъдещият Sarge.

14.1. */etc/apt/sources.list*

Първото нещо е да укажете на `apt` да използва пакетите на Sarge вместо тези на Woody. Просто в `/etc/apt/sources.list` трябва да замените `stable` (или `woody`) с `testing`.

Може да направите това и по време на самото инсталиране на Woody, когато пита за източниците. Ще трябва обаче да зададете ръчно редактиране на файла `sources.list`.

Ето примерно съдържание на `sources.list`:

```
deb http://ftp.bg.debian.org/debian/ testing main non-free contrib
```

14.2. *dist-upgrade*

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

С това се извършва същинското преминаване от Woody към Sarge.

14.3. *grub*: Замяна на LILO с GRUB

```
# apt-get install grub
# grub-install /dev/hda
# update-grub
# dpkg -P lilo                    # Пълно изтриване на LILO
```

Това е въпрос на личен избор, но `grub` в почти всички случаи е по-добър.

Файлът **kernel-img.conf**(5) се използва при инсталиране и деинсталиране на ядра. Следната конфигурация в `/etc/kernel-img.conf` прави `grub` винаги да поддържа списък на всички инсталирани ядра автоматично.

```
postinst_hook = /sbin/update-grub
postrm_hook = /sbin/update-grub
warn_initrd = no
```

14.4. *kernel-image-2.6-686*

```
# apt-get install kernel-image-2.6-686
```

В почти всички случаи искате да имате ядро 2.6. Конкретното предложено име на пакет избира последното налично готово ядро от серията 2.6, което е компилирано за процесори Pentium II или по-нови. Вместо `686` може да сложите `386` или `k7`. Ако системата Ви е многопроцесорна, добавете `-smp` към името на пакета.

Ако сте настроили **kernel-img.conf**(5) както трябва, просто рестартирайте и ще сте с новото ядро.

14.5. *udev*: Замяна на devfs с udev

Целта му е да замени досегашната реализация на `devfs` от `kernel-space` прехвърляйки нещата към `user-space` използвайки `sysfs` и `/sbin/hotplug`. Директорията `/dev` ще бъде динамично запълнена с файловете на устройствата в зависимост от вашата конфигурация. Подробна информация за `udev` може да намерите на неговия сайт[1]. Хвърлете й един поглед за да сте в час какви промени се случват. Това което трябва да направите е:

Освен това преди да инсталирате каквото и да е изтеглете сорса на `udev` и `hotplug` и прегледайте поне документацията идваща с него.

```
cd /tmp
apt-get source udev hotplug
```

Трябва да имате 2.6 ядро компилирано със следните конфигурационни опции:

```
CONFIG_HOTPLUG=y
CONFIG_PROC_KCORE=y
CONFIG_SYSFS=y
# Ако включите CONFIG_DEVFS_FS=y , то при зареждане трябва да подадете
# devfs=nomount за да може да се използва udev
# Не включвайте CONFIG_DEVFS_MOUNT=y
CONFIG_DEVPTS_FS=y # ако я имате, защото след 2.6.4 е активирана
                   # директно в сорса на ядрото и не е като опция
CONFIG_TMPFS=y
CONFIG_RAMFS=y
```

Внимание: не редактирайте директно config файла на ядрото за да включите някоя опция, използвайте някой от конфигураторите (make menuconfig, make xconfig и т.н.). Това е защото включването на една опция може да изисква включване на друга, което може да се окаже малко трудно да го разберете веднага на момента.

```
apt-get install udev hotplug
```

Ако някои файлове за особени устройства в `/dev` не бъдат създадени, то ги добавяте в `/etc/udev/links.conf`. Например:

```
M nvidia0 c 195 0
M nvidia1 c 195 1
M nvidiactl c 195 255
```

Където **M** означава, че device nodes ще бъдат създадени с `/sbin/MAKEDEV`.

14.6. *screen*, *less*, *vim*: За работа в терминал

```
# apt-get install screen less vim
# dpkg -P nvi
```

Това са някои пакети, които често се използват при работа в терминал.

В `/etc/profile` добавете следния ред, който ще направи `less` да може да показва `gzip` и `bzip2` файлове:

```
eval $(lesspipe)
```

Към `/etc/vim/vimrc` добавете следните редове, които включват оцветяването на файлове и генериране на backup файлове (което винаги е добра идея):

```
set backup
syntax on
```

14.7. */etc/inetd.conf*: Изключване на ненужни услуги

От файла **inetd.conf**(5) изключете ненужните услуги `discard`, `time` и `daytime`. Презаредете **inetd**(8):

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

14.8. *postfix*: Замяна на Exim с Postfix

```
# apt-get install postfix
```

Както и при GRUB, замяната на Exim с Postfix е въпрос на предпочитание. Инсталирането на пакета `postfix` ще премахне пакетите на Exim!

14.9. *ssh*, *dnsutils*: Отдалечен достъп

```
# apt-get install ssh dnsutils
```

Ако сте в мрежа, сигурно ще искате тези пакети.

[1] http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html

Глава 15

Изготвяне на резервни копия - backup

15.1. Общи сведения

Има два типа хора - такива на които има е трябвал бакъп, и такива на които всеки момент ще им потрябва.

Александър Велин

Като идея изготвянето на резервни копия е много добро нещо. Така както и използването на RAID и LVM ако можем да си го позволим разбира се. Това не е специфично за Debian, така, че ще е полезно и ще може да се използва и от други системи. Начини за изготвяне на резервни копия има много и то с най-различни цели и подходи. Дори всеки може да си измисли технология за резервиране на важните за него файлове. Разбира се говорим за неща малко по-сложни от `cp -ra`, като инкрементирано обновяване на бакъпа, автоматизирано, отдалечено, върху различни носители и прочее. Ще се опитаме да предложим всякакви варианти.

15.2. Прости програми за дублиране или резервиране

15.2.1. dd - копиране на файлове

Програмата **dd**(1) е част от пакета `coreutils`.

```
apt-get install coreutils
```

Най-общо се използва така. Ако например искаме да направим огледално копие на първия primary дял на първия логически дял на същия IDE disk. Може и на друг disk разбира се, както от disk на tape и обратно. Не използвайте тази команда преди да сте прочели ман-страницата ѝ и внимавайте да не презапишете нещо върху важни за вас данни.

```
dd if=/dev/hda1 of=/dev/hda5
```

15.2.2. raw - свързва linux raw device към block device

Програмата **raw**(8) е част от пакета `util-linux`.

```
apt-get intall util-linux
```

Най-общо се използва така

15.2.3. dump - dump и restore за ext2/3 файлови системи

Програмите **dump**(8), **restore**(8), **rmt-dump**(8), **rdump**(8), **rrestore**(8) са част от пакета `dump`.

```
apt-get intall dump
```

Най-общо се използва така

15.2.4. pcopy - large disk(partition) to disk(partition) copying tool

Програмата `pcopy` е аналогична на **dd**(1) и към момента я няма в официалния архив на Debian. Можете да я намерите тук[1]

[1] http://sf.www.lysator.liu.se/~pen/pcopy/

15.3. По-сложни подходи и системи за изготвяне на бакъп

15.3.1. revision control systems - удобно за текстови файлове

Използване на вашата домашна директория или /etc 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` ;-)

Естествено подобно нещо може да се прави и чрез аналозите на CVS, като `subversion`, `arch`, `tla`, `rcs`, `cssc` и т.н.

15.3.2. rsync - бърза и ефективна програма за отдалечено синхронизиране

Инсталираме пакета `rsync`.

Много добра статия по въпроса има на: http://www.mikerubel.org/computers/rsync_snapshots/[3]

15.3.3. dirvish - filesystem-базиран бакъп чрез rsync

15.3.4. backuppc - disk-базиран бакъп с много възможности

15.3.5. bacula - network-базиран бакъп, възстановяване и верификация

15.3.6. partimage - partitions-базиран бакъп в компресирани image files

15.3.7. mondo - CD-базиран бакъп

15.3.8. amanda - клиент/сървър-базиран автоматичен advanced network disk archiver

15.3.9. cdrw-taper - дабавка за amanda за CD-RW и DVD+RW бакъпи

15.3.10. multicd - CD-базиран бакъп

15.3.11. faubackup - filesystem-базиран бакъп

15.3.12. dar - многоцелеви архиватор, дифенциални бакъпи, компресия, ssh

15.3.13. rdiff-backup - deltas-базиран бакъп

15.3.14. pdumpfs - filesystem-базиран бакъп, използващ ruby

15.3.15. storebackup - рекурсивно копиране на директорийни дървета

15.3.16. ibackup - бакъп за /etc, включително автоматизирано и отдалечено

15.3.17. afbackup - клиент/сървър-базиран бакъп

15.3.18. kbackup - компресиране, криптиране, multi-volume archives, и много други

15.3.19. cdbackup - CD-R(W) бакъп

[2] http://www.linuxjournal.com/article.php?sid=5976
[3] http://www.mikerubel.org/computers/rsync_snapshots/

Част IV

Управление на софтуера

Глава 16

Общи положения

Качествен свободен софтуер вече има в неимоверно големи количества и продължава да се бълва такъв. Затвореният софтуер бива настиган 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 потреблява 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 за 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 пък 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. Тук няма да такова животно наречено рестарт за/след upgrade. Тук няма и нужда до ходите до определена точка на планетата само за да upgrade-нете към по-новата версия на SSH. Скриптовете за preinstall,install и postinstall се грижат за това, за което другите дистрибуции ги е страх да мислят. 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 подръжка на fast & safe security updates, Bug Tracking System, mailing lists, news groups, irc channels.

16.1. 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 (разгладейте 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. Unstable винаги ще си е Sid. (виж малко по-надолу)

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, sparc64, ia64, hppa ...	alpha, arm, i386, m68k, powerp sparc64, ia64, mips, mips64 . i386 ...
Date	Mon, 16 Apr 2001	Tue, 04 Sep 2001	Tue, 04 Sep 2001
Description	Debian 2.2r3 Released 16 Apr 2001 (обновява се само за security и point releases, т.е. revisions)	Not Released (release-ва се директно от Testing, евентуално след известен период на freeze, след решаване на всички Release Critical Bugs)	Not Released (не се release-ва тно от Unstable, тук се правя по-сериозни package transitio добни. Всичко минава първо
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[1], след това в Unstable, после в Testing и оттам — в Stable (има и `project/experimental`[2], 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[3] 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[4], 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[5] — Debian autobuild system http://buildd.debian.org[6], http://www.buildd.net[7] 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/[8]. Интересно би било да погледнете и **The Common Debian Build System** намираща се на: http://build-common.alioth.debian.org/[9] и http://alioth.debian.org/projects/build-common/[10].
- **hurd-i386** — The Hurd[11] е набор от сървърски програми, които работят върху микроядрото GnuMach[12] (поне за сега само върху това микроядро и само на x86) и които реализират драйвер на файлова система, мрежови протоколи и всякакви други способности, които се срещат в стандартните монолитни Unix ядра. Усилено се говори (ъ-ъ, работи) по използването на микроядрото L4Ka::Pistachio[13], което чувствително ще подобри бързодействието на The Hurd. Още информация по-въпроса можете да намерите на:
  - GNU Project[14]
  - Linux and GNU[15]
  - GNU Hurd fans[16]

[1] http://incoming.debian.org
[2] ftp://ftp.debian.org/debian/project/experimental
[3] http://www.debian.org/mirror/
[4] http://archive.debian.org
[5] http://db.debian.org/machines.cgi
[6] http://buildd.debian.org
[7] http://www.buildd.net
[8] http://buildd.debian.org/stats/
[9] http://build-common.alioth.debian.org/
[10] http://alioth.debian.org/projects/build-common/
[11] http://www.gnu.org/software/hurd/hurd.html
[12] http://www.gnu.org/software/hurd/gnumach.html
[13] http://l4ka.org/projects/pistachio/
[14] http://www.gnu.org/
[15] http://www.gnu.org/gnu/linux-and-gnu.html
[16] http://hurd.gnufans.org/

  - И естествено българското участие в лицето на Оги Кулев[17]

- Към момента единственото ядро, с което се правят издания (releases) на Debian, е Linux[18], 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[19], който за сега стои само в Sid, работещ върху микроядро GnuMach[20], при което се получава Debian GNU/Hurd[21]. Следват евентуално и монолитните *BSD ядра, при което се получават Debian GNU/NetBSD[22], Debian GNU/FreeBSD[23], Debian GNU/OpenBSD (но последното като че ли е временно изоставено[24]) за евентуално няколко хардуерни архитектури. Всъщност погледнете и Debian on the Go[25] (laptops) и Debian Beowulf[26] (MPI clusters).

16.1.1. 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[27], Package Tracking System[28], отчитат се Release Critical Bugs[29] и Quality Assurance[30] за Base[31] и Standard[32] и т.н.

16.2. Неофициалните архиви

Неофицалните 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). Например:

- http://www.apt-get.org[33] е един източник, където се събират (и проверяват) подобни неофициални източници.
- http://apt.heanet.ie[34] - HEAnet aptable debian package repository - Ireland's National Education & Research Network.
- http://openfmi.net/projects/debian-addons-bg[35] Поддържа се от български участници.
- Подобни неофициални архиви могат и да се поддържат от проекти като KDE[36], Mozilla[37] 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[38], Package Tracking System[39], 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 ;-).

[17] http://debian.fmi.uni-sofia.bg/~ogi/hurd/ext3fs/
[18] http://www.kernel.org
[19] http://www.gnu.org/software/hurd/hurd.html
[20] http://www.gnu.org/software/hurd/gnumach.html
[21] http://www.debian.org/ports/hurd/
[22] http://www.debian.org/ports/netbsd/
[23] http://www.debian.org/ports/freebsd/
[24] http://lists.debian.org/debian-bsd/2002/debian-bsd-200210/msg00063.html
[25] http://www.debian.org/misc/laptops/
[26] http://www.debian.org/ports/beowulf/
[27] http://bugs.debian.org
[28] http://packages.qa.debian.org
[29] http://bugs.debian.org/release-critical/
[30] http://qa.debian.org
[31] http://base.debian.net
[32] http://standard.debian.net
[33] http://www.apt-get.org
[34] http://apt.heanet.ie
[35] http://openfmi.net/projects/debian-addons-bg
[36] http://www.kde.org
[37] http://www.mozilla.org
[38] http://bugs.debian.org
[39] http://packages.qa.debian.org

Глава 17

Инструкции за работа с *dpkg*, *dselect* и *apt*

17.1. Въведение

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.

17.1.1. 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, заедно с `MD5` сумите на следващите два файла и `PGP` подписа на този който го е създал
- *`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*. Какво представлява формата на тези файлове ще намерите в **deb**(5) и на http://www.dpkg.org[1]

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

[1].

17.1.2. *dpkg*: A medium-level package manager

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.

17.1.3. *dselect*: Debian package management frontend

`dselect`, е стандартния за Debian механизъм за управление и инсталация на пакети. Въпреки че частично функционалността му може да бъде постигната чрез apt или директно чрез dpkg, dselect разполага с някой функционалности, които не трябва да бъдат пропускани.

Основното, което прави **dselect**(8) уникален като механизъм за управление на пакетите в debian, е това, че той присъства стандартно във всяка инсталация, предлага текстово меню базиран интерфейс, и следи зависимостите между пакетите много по-дълбоко, отколкото го прави apt, и това му помага да решава проблеми, които `apt` не е способен за сега да реши.

Основи на dselect

`dselect` трябва да се стартира от командният ред в текстов режим, с права на **root** и без никакви параметри. Макар че е възможно да се стартира от името и на друг потребител, при типично стандартно инсталирана дистрибуция няма да може да се прави нищо съществено с него. `dselect` разполага с няколко командни параметъра, но те се използват много рядко и като цяло управляват debug, или цветови режим.

Менюто на `dselect` се състои от няколко основни части:

1. **[A]ccess**
2. **[U]pdate**
3. **[I]nstall**
4. **[C]onfig**
5. **[R]emove**
6. **[Q]uit**

Всяка буквичка от меню заградена в квадратни скоби `[]` може да бъде използвана за директен достъп чрез натискане на съответния клавиш от клавиатурата. Също така е възможно да се използва и съпоставеният номер пред името на реда, за директен достъп. В противен случай можете да използвате и да се позиционирате на съответното място чрез стрелките.

Въпреки че не е било винаги така, в съвременните Debian дистрибуции **dselect**(8) използва **dpkg**(8) и **apt**(8) като подсистеми за конкретно управление на пакетите, по макар и невидим за потребителите начин. Все пак е важно да се знае това, с цел в процеса на работата с дистрибуцията и управлението на пакети да се осъзнае нейната логика и свързаност между съставните и компоненти.

Когато управлявате инсталацията на пакети основните въпроси, на които трябва да си отговорите са:

1. От къде се извличат пакетите, когато желаем да ги инсталираме? - С това се занимава конфигурацията в менюто **[A]ccess**
2. Какви точно пакети има на мястото, от което ги взимаме преди да ги инсталираме и какви са техните изисквания и зависимости? - С това се занимава менюто **[U]pdate**
3. Какво точно искаме да направим (да инсталираме, премахнем, преконфигурираме) и с кой пакет? - С това се занимава менюто **[S]elect**
4. Да изпълним съответните действия зададени от **[S]elect**. - С това се занимават менюто **[I]nstall**, **[C]onfig** и **[R]remove**.

Следователно, за да направим инсталация на пакет през **dselect**(,) трябва последователно да преминем и конфигурираме всяко едно от менютата - **[A]ccess**, **[U]pdate**, **[S]elect**, **[I]nstall**. От друга страна, ако веднъж сме конфигурирали **[A]ccess** например, и не желаем да променяме конфигурацията, не е нужно да преминаваме от там. Същото се отнася като цяло и за другите менюта.

[A]ccess Целта на това меню е да бъде конфигуриран и избран методът на достъп до мястото, където се намират пакетите, които желаем евентуално да инсталираме на системата. Веднъж конфигурирано, не е необходимо това меню да бъде достъпвано повече (освен, ако не желаете да направите някаква промяна).

В съвременния **dselect**(8) след избиране на **[A]ccess** следва да посочите метод на достъп до пакетите. Възможностите са много, поради факта, че методите са всъщност малки модули, които разширяват функционалността на dselect. Всеки, както и вие самият можете да си напишете такъв модул.

Типичните налични методи са:

1. multi_cd
2. multi_nfs
3. mounted
4. floppy
5. apt

Метод за достъп **apt**

Това е метод за достъп, който е базиран на стандартната работа на командата **apt**(8). Целта е достъпът и управлението на пакетите да бъде унифициран, без значение дали се използва **apt**(8) или **dselect**(8). И понеже **dselect**(8) и **apt**(8) записват информацията за наличните пакети и техните зависимости в различни файлове, е силно препоръчително в **dselect**(8) да се използва метод за достъп **apt**(8), с цел информацията между двете да си съвпада (единствената съществена причина за това е да се улесни работата на администраторите и да не се извлича два пъти информация от оригиналния източник).

Когато изберете метод **apt**(8) ви се предлага един диалог, в който можете да промените конфигурацията на apt файла sources. Там можете да изберете мястото, където се намират пакетите и как да се достъпват според правилата на apt. Менюто е точен еквивалент на извикването на командата **apt-setup**(8). Конфигурацията на достъпа се намира във файла **/etc/apt/sources.list** (**sources.list**(5)). Ако имате proxy то трябва да го зададете по правилата на apt (или lynx) чрез променлива:

```
# export http_proxy=http://gateway:3128/
# dselect
```

Метод за достъп **multi_cd**

Това е мощен но вече сравнително стар метод за достъп до пакетите на debian. Основната му цел е да позволи инсталация на дистрибуция от един или няколко CDROM-а като например Debian Binary CD.

Когато започнете да го използвате ще бъдете запитани за директорията, в която се монтира CDROM-а, мястото на Debian дистрибуцията и на пакетите в нея.

Метод за достъп **multi_nfs**, **mounted**

Това са методи, чиято цел е да се опише, че пакетите са достъпни през монтирана през NFS или по друг начин директория на диска.

Метод за достъп **floppy**

Това е непрепоръчителен метод за инсталация на дистрибуцията от floopy или ZIP диск.

[U]pdate Целта на това меню е да стартира процес, който да извлече от мястото, където се намират пакетите списък с техните имена и зависимости. Какво точно прави зависи и е специфично за всеки различен избран метод за достъп. Ако е избран препоръчителният метод за достъп **apt**, се стартира процес, който е еквивалентен на изпълнението на командата **apt-get update**. След приключване на изпълнението на apt-get update, **dselect**(8) изчита обновеният списък с пакетите на apt, и изгражда собствената си база с данни. В базата на **dselect**(8) се следят много по-сложни зависимости.

В типичните ситуации е рядкост често да се променя методът за достъп (менюто [A]ccess). Но е силно препоръчително при всяко стартиране на dselect да се извика поне веднъж [U]pdate, за да бъде сигурно, че информацията за пакетите е актуална. Това не се прави автоматично с единствената идея, че администраторът на машината може да не я е свързал в интернет или да не е сложил cdrom-ът с пакетите.

[S]elect Това е менюто, в което се върши реалната работа по управлението на пакетите. След като е направен [U]pdate, за да сме сигурни, че информацията в базата с пакетите е актуална трябва да влезем тук.

След избиране на [S]elect се вижда една страница с кратка помощ как точно ще изглежда списъкът с пакетите и как да различим кои са нови, кои са инсталирани, за кои има по-нови версии и в кои има някакъв проблем. Също така с помощта на клавиша интервал може да се разходите и да разгледате останалата част от помощната страница, както и клавишите за достъп до функциите, които се предлагат. С клавиша Enter се влиза директно в списъка с пакетите.

Това, което е важно да се запомни е, че винаги, когато натиснете клавиша “?” ще бъде показана кратка помощ за клавишите и как се използва менюто като цяло.

Ако терминалната емулация ви е конфигурирана както трябва, то трябва да виждате всички пакети, за които **dselect**(8) знае от къде да ги извлече или са инсталирани на системата. Менюто изглежда горе долу така:

Пакетите са разделени на секции (стари, графични, инсталирани, свободни, платени и т.н.). Пред всяко име на пакет има 4 знака, които за инсталираните пакети изглеждат най-често като "*". Тези три знака определят какво е състоянието на пакета спрямо системата. Това се нарича EIOM. Или по точно:

Под E стои знак, който определя дали е имало грешка при инсталирането или достъпването на пакета. Ако тук пише "R"значи е имало сериозен проблем.

Под I стои знак, който определя дали пакетът е инсталиран вече. Знаците значат както следва:

- * - пакета е инсталиран
- - - пакета не е инсталиран, но конфигурационните му файлове присъстват (може да е бил деинсталиран без изтриване на конфигурационните файлове)
- U - разархивиран, но без стартиран конфигурационен скрипт
- C - полуинсталиран, поради грешки в конфигурационният скрипт
- I - полуинсталиран, поради някаква грешка

Под O стои знак, който определя каква е била предишната ни директива/команда към пакета (например да се инсталира).

Под M стои знак, който определя текущата директива (последното, което сме избрали и което ще бъде изпълнено при **[I]nstall**). Знаците значат както следва:

- * - ще бъде инсталиран и упгрейдван
- - - ще бъде изтрит
- = - пакета е заключен. Няма да бъде променян от автоматично действие.
- _ - пакета е маркиран за пълно изтриване (включително с конфигурационните файлове)
- n - пакета е нов и не е имал предишно зададена команда

Възможно е да желаете да виждате информацията за пакетите в по-подробен вид. Това се превключва чрез клавиша "V Verbose.

Основните клавиши, които трябва да се знаят:

- + избира пакетът да бъде инсталиран. При натискането веднага се проверяват всички познати зависимости между пакетите и се прави предложение да бъдат изпълнени.
- - избира пакетът да бъде деинсталиран. Проследяват се зависимостите и се прави предложение със списък на всички засегнати пакети, дали и те да се деинсталират.
- = заключва пакета. Така той няма да бъде променян или предлаган за upgrade при автоматична проверка.
- : отключва пакета, ако е бил заключен.
- Q излиза от менюто или под менюто, без да прави проверка на зависимостите.
- Enter излиза от менюто с проверка на зависимостите (препоръчителният начин).
- X излиза без да запомня последната промяна.
- CTRL+C излиза от dselect директно.
- / търси текст в името на пакетите.
- търси последният търсен текст отново.
- e се мести от началото до края на списъка.
- i дава допълнителна информация за пакета.

[I]nstall Това е менюто, което изпълнява зададените в Select команди. Ако се използва метод APT, тук се и изтриват зададените за изтриване пакети.

[C]onfigure Това е менюто, което се опитва да конфигурира всички неконфигурирани пакети, или пакети, на които не са сработили инсталационните скриптове, чрез повторното им изпълнение.

[R]emove Това е менюто, което изтрива зададените в [S]elect за изтриване пакети.

17.1.4. *apt*: Advanced Package Tool

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
```

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 командата 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[2]. 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 подбрани 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
```

[2] http://www.debian.org/security/

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

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

17.1.5. *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.

17.1.6. *synaptic*: Друг графичен начин

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

17.2. *apt-dpkg-ref*: Справочник на опциите на *apt* и *dpkg*

Превод на APT and Dpkg Quick Reference Sheet[3]. 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/`.

17.2.1. *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.

```
# 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`.

[3] http://people.debian.org/~mrd/deb-ref/apt-dpkg-ref.html

17.2.2. *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.

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

```
# apt-get source [-b] пакет
```

Изтегля сорса 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 сорс. 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 сорса на пакета, ще инсталира всички зависимости, нужни за компилиране, и ще се опита да компилира сорса.

```
# 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-а“, 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`.

17.2.4. 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 конфликите 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.

17.3. 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-free
#####################################################################
# 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-free
# apt-get install gnome2
#deb http://people.debian.org/~walters/debian/staging/ ./
# 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, kdeedu, kdesdk, kdetoys, 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-fre
# 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 follow
#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-Upgraded "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: *
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
```

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

17.3.1. 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 дузина unofficials, това 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
```

17.3.2. 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 изпадат 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[4] 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[5], 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:

[4] http://snapshot.debian.net

[5] http://snapshot.debian.net

```
# 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.

17.3.3. 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[6] 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 `debfoster -u`, което се дава. Чрез промяна на приоритета на пакетите може да се постигне и т.н. им „заковаване“. FIXME: Реферира се apt-howto-bg.

17.3.4. 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 HOWTO[7] 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 Stable[8] 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.

17.3.5. 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.
```

[6] http://people.debian.org/~walters/gnome2.html
[7] http://www.debian.org/doc/manuals/apt-howto/ch-apt-get.en.html#s-pin
[8] http://www.debianplanet.org/node.php?id=880

```
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.org[9] например: bg.debian.org[10]

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 KDE[11] 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
:::::·
```

17.3.6. 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`, `stormpkg`, `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 конфигурани 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 сляпо 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:

[9] http://www.debian.org/

[10] http://www.bg.debian.org/doc/package-developer/virtual-package-names-list.text

[11] http://www.kde.org/

- Install Manual за вашата хардуерна архитектура[12] (пакет `install-doc`)
- APT-HOWTO[13] или стария превод на български[14] (пакет `apt-howto-en`)
- quick-reference[15], който е силно ориентирано към потребителя (пакет `debian-reference-en`)
- apt-dpkg-ref[16] — бърз справочник за `apt` и `dpkg` (пакет `apt-dpkg-ref`)
- Wiki pages за APT[17]
- Linux HOWTO's, mini-HOWTO's, FAQ's (пакети `doc-linux-text` и `doc-linux-html`)
- 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/[18]
- още препратки към FAQ's[19]

[12] http://www.debian.org/releases/stable/installmanual
[13] http://www.debian.org/doc/user-manuals#apt-howto
[14] http://danchev.fccf.net/files/docs/linux/apt-howto-bg/
[15] http://www.debian.org/doc/user-manuals#quick-reference
[16] http://people.debian.org/~mrd/deb-ref/apt-dpkg-ref.html
[17] http://www.spack.org/index.cgi/AptHelp
[18] http://www.fifi.org/documentation/
[19] http://www.linuks.mine.nu/debian-faq/

17.4. Справяне с проблеми - *ръчно*

17.4.1. 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` с `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 бъг срещу този пакет от дадено ниво (вкл. и `wishlist`) на http://bugs.debian.org[20], 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
```

[20] http://bugs.debian.org

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 18

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.

18.1. 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]
- Debian New Maintainer's Guide[8], още достъпен и като пакет `maint-guide`, или стария му превод на http://debian.gabrovo.com/docs/maint-guide/[9]
- How To Create Your Own Debian Package[10]
- Ако ползвате локални (персонални) apt repositories, ето този бърз пример: http://www.symonds.net/~rajesh/localdeb.html[11]
- повече обяснения можете да намерите в Debian Repository HOWTO[12]
- навярно ще ви е полезен и пакета `mini-dinstall`:

[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
[8] http://www.debian.org/doc/maint-guide/
[9] http://debian.gabrovo.com/docs/maint-guide/
[10] http://www.kclee.com/clemens/unix/HowToCreateYourOwnDebianPackage.html
[11] http://www.symonds.net/~rajesh/localdeb.html
[12] http://www.isotton.com/debian/docs/repository-howto/

18.2. Многото лица на *debconf*

- The many faces of Debconf[13]
- **debconf**(1)
- `debconf`, `debconf-utils`, `debconf-devel`

Като `frontend` към пакетите ползващи `debconf` като конфигиратор може да използвате пакета `configure-debian`. При стартирането на едноименната команда се появява списък на тези пакети групирани тематично.

18.2.1. 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 ...
```

18.2.2. 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:

```
# 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 оригиналния скрипт `/usr/bin/dpkg-buildpackage` и който ще бъде модифициран:

[13] http://kitenet.net/programs/debconf/

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

Т.е. вече е регистриран факта за модификацията на оригиналния 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, осъществявяни 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.

Глава 19

Изграждане на дистрибуцията и средства за контрол

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

19.1.1. Бързо запознаване с `hello` и `hello-debhelper`

Целта на този процес е да се организира изграждането на сорс и бинарни пакети, които да бъдат обработвани от инструментите за пакетиране, инсталиране, премахване и преконфигуриране. Целта е в системата да има установен ред и правила при изпълнението на тези дейности с което наистина ще държим нещата под наш контрол, независимо колко е голяма инсталираното от системата, знаейки какво е инсталирано, как е инсталирано, с чисто премахване или надграждане на нещо вече инсталирано. `Debian` е известен с легендарните си способности за чисто инсталиране и надграждане на пакетите на системата. В тази глава ще се постараем да обясним къде е генезиса на всичко това и то да не изглежда просто като легенда, а като рутинна практика при всекидневното използване на системата. Като реален пример може да се посочи десктопа на един от авторите който е система първоначално инсталирана преди около 5 години (към момента на писане на тази глава). Системата разбира се е подложена на всекидневни упгрейди, инсталации на нови пакети или тяхното премахване и други по-необичайни дейности, при които проблеми може и да има, но те са решими. През повечето време системата изглежда като току-що инсталирана. Чиста. Разбира се, това съвсем не значи, че в някой директории като `/usr/local/` и в домашните директории на потребителите не може да се води война от инсталиране и премахване на каквото ви душа иска. Напротив, това не е територия на системата и е изцяло на вашата воля, какво ще правите там. Е, с глупави и опасни програми не си губете времето, внимавайте какво инсталирате и къде особено когато използвате суперпотребителя `root`. Научете `Debian` отвътре за да подчините и експлоатирате възможностите му за вашите потребителски цели.

- За да разберем какво съдържа един дебиански пакет, вземаме кой да е такъв и пробваме командите `dpkg-deb -x` и `dpkg-deb -e` върху него.
- За да разберем какво представлява изходния код на един дебиански пакет, вземаме изходния код на пакета `hello`, поглеждаме набързо `.dsc` файла, а останалите два файла разпакетираш по два начина:
  - с `dpkg-source`
  - ръчно с `tar` и `patch`
- Разглеждаме съдържанието на директорията `debian` в изходния код на пакета `hello`. Ползваме Debian Policy Manual[1] като справочник или това е в пакета `debian-policy`.
- Когато нещата станат ясни, правиме същото с пакета `hello-debhelper`, използвайки **debhelper**(7) като справочник. Имайте на предвид, че `debhelper` улеснява значително създаването на дебиански пакети автоматизирайки извършването на много рутинни и повтарящи се дейности.
- През свободното време четем:
  - Debian Policy Manual[2]
  - Debian Developer's Reference[3]
  - Debian New Maintainer' Guide[4]
- Вземаме програмата която искаме да пакетираме и копираме в изходния й код директорията `debian` от пакета `hello-debhelper`, като пригодяваме нещата за да генерираме нашия `deb`-пакет. Следващия път няма да използваме `hello-debhelper`, а някой вече съществуващ изготвен от нас пакет.
- Когато разберем какво значи **Intend To Package**, изпращаме **ITP** съобщение, за да не би някой друг да реши да пакетира същата програма и да си дублирате усилията. Ако все още не сме одобрени за `upload` в официалното хранилище, можем да използваме Mentors Public Package Repository[5]

[1] http://wwww.debian.org/doc/debian-policy/
[2] http://wwww.debian.org/doc/debian-policy/
[3] http://wwww.debian.org/doc/debian-policy/
[4] http://www.debian.org/doc/maint-guide/
[5] http://mentors.debian.net

19.1.2. Същото с `dh-make` и `devscripts`

- Начално дебианизиране с инструмента **dh_make**(8) от пакета `dh-make`

  ```
  dh_make -e your.maintainer@address -f ../program-1.2.3.tar.gz
  ```

- Цикъл на поддръжка с инструментите от пакета `devscripts`
  - Влизаме в сорс директорията на пакета
  - Правим промени по файловете
  - Документиране на промените с: `dch -i "I changed this"`
  - Компилираме изпълнявайки: `debuild`. Ако се върне грешка, се връщаме пак към промяна на файловете. Също така можем да стартираме компилацията със съответната част на `debian/rules` (като `debian/rules build binary patch unpatch` и `т.н.` каквото има предвидено)
  - Проверяваме дали пакета е в норми с: `debc`
  - Инсталираме пакета за да го изтестваме лично с: `debi`
  - Ако всичко е наред издаваме пакета с: `debrelease`

19.1.3. По-дълги обяснения

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 фактът, 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**[6]. Едно доста добро обяснение като за начало би бил документът на IBM - *Learn how to build easy-to-distribute packages for Debian users*[7].

19.1.4. Инструменти улесняващи пакетирането

Ще обясним накратко кои пакети и кои програми идващи с тях обикновено се използват при този процес. Тук ще дадем само някои насоки, което не изключва прочитането на **man**(1) страниците на програмите и документацията в директориите `/usr/share/doc/име_на_пакет/`. Имайте предвид, че постоянно могат да се появяват такива, но възлагайте на тях само рутинната част по създаването на файловете необходими за пакетиране, след което лично се убедете в тяхното съдържание и нанесете съответните корекции, така щото сорс пакета който създавате да се получат съответните бинарни пакети (**deb**(5)) в съответствие с всички правила и норми. Процеса може да бъде много прост и рутинен, но и много сложен и необичаен, всичко зависи от това какъв сорс пакетирате и как точно сте решили да го направите.

Пакета `dh-make`

Съдържа програмата **dh_make**(8), която се използва за първоначалното създаване на директорията `debian` и файловете в нея. С други думи спестява ви тази рутинна дейност, но задължително трябва да разгледате файловете и да внесете необходимите промени за точно този сорс пакет който дебианизирате. Дотук вече трябва да сте добили представа как се борави с debian `binary packages` (или **deb**(5)). Нека видим как се получават те. Захващаме се да разберем какво има и в `debian source packages`. Само да напомним, че без да сте прочели Debian New Maintainer's Guide[8], още достъпен и като пакет `maint-guide`, или поне да сте прехвърлили с поглед стария му превод[9], 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 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:

[6] http://www.debian.org/doc/maint-guide/
[7] http://www-106.ibm.com/developerworks/linux/library/l-debpkg.html?ca=dgr-lnxw01DebianLinux
[8] http://www.debian.org/doc/maint-guide/
[9] http://debian.gabrovo.com/docs/maint-guide/

```
# 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 Guige[10], главата First Steps, т. 2.4 Initial Debianization.

Пакета `debhelper`

Съдържа колекция от мощни `Perl` скриптове които можете да извиквате от `debian/rules`, който обикновено е файл за програмата **make**(1), както и от други скриптове като `.preinst`, `.postinst`, `.prerm`, `.postrm`. Скриптовете са много, просто изпълнете: `dpkg -L denhelper` за да ги видите. Скриптовете `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 скриптовете на `debhelper` като част от техния build процес.

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 скриптовете 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.conffiles`
- **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_installemacsen**(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).

[10] http://www.debian.org/doc/maint-guide

- **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 някой 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
```

Пакета `debconf`, `debconf-utils`, `po-debconf`

Унифициран начин пакетите да задават въпроси на потребители при първоначално инсталиране, последващо преконфигуриране и премахване.

Пакета `devscripts`

Съвременният начин. `zless /usr/share/doc/devscripts/README.gz` Предвиден е 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) a 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 привери 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 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`, `procmail` и др. [`libwww-perl`]

Пакета `debmake`

Старият начин. Това 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 използваема с минимална нужда от доредактиране
- **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.
  - 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.

Пакета `dpatch`, `patchutils`, `dh-kpatches`

Пакета `cdbs`

Пакета `cvs-buildpackage`

Пакета `svn-buildpackage`

Пакета `arch-buildpackage`

Пакета `tla-buildpackage`

19.1.5. Инструменти за проверка и контрол

Пакета `linda`

Пакета `lintian`

Пакета `debian-test`

Пакета `debbugs`

Пакета `reportbug`

19.1.6. Примерни програми за пакетиране

FIXME: да се започне и довърши step-by-step пример за:

- http://mironcho.sf.net[11]

[11] http://mironcho.sf.net

– http://bebocd.sf.net[12]
– http://opensource.netuser.cc/sgmixer/[13]
– http://biona.sf.net[14]
– http://netacct-mysql.gabrovo.com[15] (upstream debianized)

Примерни apt хранилища с binary и source пакети:

– http://danchev.fccf.net/debian/[16]
– http://ftp.logos-bg.net/debian/[17]

[12] http://bebocd.sf.net
[13] http://opensource.netuser.cc/sgmixer/
[14] http://biona.sf.net
[15] http://netacct-mysql.gabrovo.com
[16] http://danchev.fccf.net/debian/
[17] http://ftp.logos-bg.net/debian/

19.2. 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 малка програма, която се получава от един единствен сорс-файл, изискващ само наличието на основната C-библиотека, която всички имат, и освен това не конфликтира 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[18]

```
/*
 * In terms of GNU GPL
 * by <zimage@delbg.com>
 * 1995-2000 Davidov Electric Ltd.
 */
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>
#define DATASIZE 1024
int main (int argc, char *argv[]){
  int sockfd, bytesRecv, bytesSent;
  char *sendHeader, rData[DATASIZE];
  char *addr;
  struct hostent *wsHost;
  struct sockaddr_in wsAddr;
  struct in_addr wsIP;
  sendHeader = "HEAD / HTTP/1.0\n\r\n\r\n\r";
  if (argc < 2)
  {
    printf ("wsver v1.1 by zImage <zimage@delbg.com>\n\n\t");
    printf ("Usage: %s <ipaddress | hostname>\n\n", argv[0]);
    exit (0);
  }
  addr = argv[1];
  if((wsIP.s_addr = inet_addr(addr)) == -1)
  {
        printf("Lookin' up %s...\t", addr);
        fflush(stdout);
        if((wsHost = gethostbyname(addr)) == NULL)
        {
          printf("Failed.\n");
          herror(NULL);
          exit(1);
        }
        wsIP = *(struct in_addr *)wsHost->h_addr_list[0];
        printf("%s\n", inet_ntoa(wsIP) );
  }
  if ((sockfd = socket (PF_INET, SOCK_STREAM, 0)) == -1)
  {
    printf ("Error: %i", errno);
    perror(NULL);
    exit (1);
  } else {
    printf ("Connecting to %s...\t", inet_ntoa(wsIP));
    fflush(stdout);
  }
  wsAddr.sin_family = PF_INET;
  wsAddr.sin_port = htons (80);
  wsAddr.sin_addr.s_addr = wsIP.s_addr;
  bzero (&(wsAddr.sin_zero), 8);
 if ((connect(sockfd, (struct sockaddr *) &wsAddr, sizeof (struct sockaddr))) == -1)
  {
      printf ("Failed.\n");
      perror (NULL);
      exit (1);
  } else {
      printf ("OK.\n");
  }
  printf ("Sending header...\t");
  fflush(stdout);
  if ((bytesSent = send (sockfd, sendHeader, strlen (sendHeader), 0)) == -1)
  {
      printf ("Failed.\n");
      perror (NULL);
      exit (1);
  } else {
```

[18] http://danchev.fccf.net/files/wsver/wsver.c

```
      printf ("OK - %i bytes sent.\n", bytesSent);
  }
  printf ("Reading header...\t");
  fflush(stdout);
  if ((bytesRecv = recv (sockfd, rData, DATASIZE, 0)) == -1)
  {
    printf ("Failed.\n");
    perror (NULL);
    exit (1);
  } else {
    printf ("OK - %i bytes read\n", bytesRecv);
  }
  rData[bytesRecv] = '\0';
 printf (" \n- - - Header received - - -\n%s- - - end of header - - -\n", rData);
  return 0;
}
```

която се компилира 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 напряко, прескачайки целия Debian New Maintainers' Guide[19].

```
# 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 подбран, възможно най-непретенциозен сорс-код. 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[20] — 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[21], 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 *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
```

[19] http://www.debian.org/doc/maint-guide/

[20] http://www.mplayerhq.hu

[21] http://danchev.fccf.net/files/mplayer/Makefile

Имайте предвид, че в глава 10.1 на debian-policy е указано, че променливата DEB_BUILD_OPTIONS е запазена за други цели и официално приема *noopt* и *nostrip* стойности. Така, че ако MPlayer ще влиза в официалния Debian архив, то в неговия *debian/rules* ще се ползва друга променлива за целта за която сега се ползва DEB_BUILD_OPTIONS.

Ще получите съответния *binary package*, можете да генерирате *list files* за *apt repository* (**dpkg-scanpackages**(8), **dpkg-scansources**(8) и т.н.) Нека за момент предположим, че нямате инсталирана библиотеката SDL и съответно сте получите 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 подскажат `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[22].

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[23]. Ще видите, че нещата невинаги 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[24], поделени библиотеки[25], Debian Library Packaging guide[26]

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/[27]. 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:

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[28] и Debian Developer's Reference[29] би било от изключителна полза, както и познанията за upstream souces, 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

[22] http://debian.gabrovo.com/docs/apt-howto/
[23] http://www.debian.org/doc/maint-guide/
[24] http://www.debian.org/doc/debian-policy/ch-files.html#s11.2
[25] http://www.debian.org/doc/debian-policy/ch-files.html#s11.3
[26] http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html
[27] http://www.fifi.org/doc/HTML/
[28] http://www.debian.org/doc/maint-guide/
[29] http://www.debian.org/doc/developers-reference/

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. Прекомпилация 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 прекомпилация и инсталация. Да предположим, че все пак сте преценили, че се налага прекомпилация 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 задачка. Няма да вадим корен крадратен от голямо число с лист и молив, я. Това, че знаем алгоритъма, не е основание да го правим с беден инструментариум. Първо, проверяваме в кои 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`.

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[30].

Променяме сорса. В директорията `debian/` са конфигуриращите скриптове на maintainer-а, 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[31]

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-а, 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

[30] http://www.debian.org/doc/maint-guide/

[31] http://www.symonds.net/~rajesh/localdeb.html

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` 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 таг) на `dpkg` от cvs.debian.org[32] и разгледате `debian/control`, ще видите, че може да се получи и е предвиден и *binary package*: `dpkg-static`. FreeBSD[33] 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/`, получете 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. Grep-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 скромен 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 архитуктура, 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 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 докъдето 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.

[32] http://cvs.debian.org
[33] http://www.FreeBSD.org

Глава 20

Компилиране и инсталиране на софтуера - проблеми и решения

20.1. Local APT Repositories

20.1.1. 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
```

debuild: debian binary пакети от source пакети

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

```
# debuild -b -uc -us
```

sbuild: debian binary пакети от source пакети

За разлика от `debuild` програмката от едноименния пакет `sbuild` разбира и от `source dependencies`.

cvs-buildpackage: debian binary пакети от CVS хранилище

Управлява build процеса за получаване на `deb` пакет от сорсове съхранявани в произволно CVS хранилище (да не си помисли някой, че сорса трябва да е на cvs.debian.org;-). Много яка утилка.

20.1.2. *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-пъргаво 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
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[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).

[1] http://www.debian.org
[2] http://www.debian.org
[3] http://www.gentoo.org
[4] hppt://www.mozilla.org

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
```

Можете да посочите и конктерна версия на сорса на пакета, например:

```
# apt-get source diff=2.8.1-6
```

където `diff` е името на пакета, а `2.8.1-6` е неговата версия. Общия формат на версията е: `Epoch:UpstreamVersion-DebianPackageVersion` или например texttt4:1.2.3-8. `Epoch` и `DebianPackageVersion` са опционални и не винаги се ползва. Наличните версии можете да намирате чрез:

```
# apt-cache policy package
```

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 актулизира 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) комипилира 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
  ```

– *Премахване на devel библиотеките и разчистване на ненужния софтуер с **debfoster**(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 — **debfoster**(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.

20.1.3. *apt-src*

FIXME: Съдържание трябва да има тука ;-)

20.1.4. *pbuilder*

FIXME: Съдържание трябва да има тука ;-)

20.1.5. *apt-fu*

Описание[5]

```
deb http://www.yhbt.net/normalperson/debian ./
deb-src http://www.yhbt.net/normalperson/debian ./
```

[5] http://www.yhbt.net/normalperson/debian/html/

20.1.6. *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.org[6]) 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[7]

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

```
# 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 предоставя инструментът **kernel-package**(5). Разбира се, това е един 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.org[8], т.е. официалния изходен код. 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
```

За нетърпеливите

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
```

За любознателните

Самата процедура на изграждане на ядрото е следната:

1. Редактира се файла `.config`, намиращ се в директорията със сорса на ядрото. Това може да се извърши по няколко начина:
   - Първият е като ръчно се редактира файла - рядко използван.
   - Вторият е да изпълните командата

     ```
     # make config
     ```

     Ще Ви бъдат зададени цял куп въпроси, а при това не можете да се връщате назад, за да промените някой отговор. Не се препоръчва също така, защото трябва да преминете през всяка една опция, което е ненужно.
   - Третият е да изпълните

[6] http://www.kernel.org
[7] http://www.osnews.com/story.php?news_id=2949
[8] http://www.kernel.org

```
# make menuconfig
```

Разполагате с подредено меню в текстов режим. Според мен това е най-удобният начин. (нужен е пакета `libncurses5-dev` и тези, от които зависи... освен ако ги имате предварително инсталирани).

– Четвъртият е като използвате

```
# make xconfig
```

Графичен конфигуратор, предназначен за употреба под X. Прилича до някаде на `menuconfig`, но има малко объркваща подредба (може да Ви се наложи да инсталирате пакета `tk8.3`).

2. Следва компилирането и пакетирането. То може да бъде изпълнено с командата:

```
# make-kpkg kernel-image
```

Ядрото и избраните от Вас модули ще бъдат компилирани и след това ще бъде изграден Debian пакет с име `kernel-image-<версия>_<архитектура>.deb`, който ще се появи в `/usr/src/`.

За повече информация относно конфигурирането на ядрото и значението на най-използваните и нужни опции, можете да погледнете статията на Никола Антонов и съответните й части на адрес - http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=340742097[9].

Следва да инсталирате пакета с новото Ви, намиращ се в `/usr/src`:

```
# dpkg -i kernel-image-<версия>_<архитектура>.deb
```

Най-вероятно ще бъдете запитани дали искате да бъде направена boot дискета с новото ядро (не е проблем да откажете). Също така, ако използвате `LILO` за зареждане на системата (подразбиращия се в Debian Woody), ще Ви се предложи да бъде направен нов запис според съществуващия вече `/etc/lilo.conf`. Не се притеснявайте да отговорите с "Yes тъй като **kernel-package**(5) ще се погрижи новото ядро да бъде добавено в `/etc/lilo.conf`, а старото също така да е налично, в случай, че сте объркали нещо с конфигурацията на новото ядро и то откаже да зареди. Така при проблем можете да заредите старото (по подразбиране е именувано LinuxOLD) и пак да имате функционираща система.

Допълнителна информация

Освен описаните до тук основни функции на **kernel-package**(5), нужни за да компилирате набързо едно ядро, има и някои други тънкости. Информация за тях можете да намерите във файла `/usr/share/kernel-package/README` или като изпълните командата *man* **make-kpkg**(1). Също така можете да хвътлите един поглед на *man* **kernel-img.conf**(5) и *man* **kernel-pkg.conf**(5).

Също така можете да обърнете внимание на следните две неща:

1. "Проблемът който се получава при използването на автоматичната редакция на `/etc/lilo.conf`: при този метод скриптовете на **kernel-package**(5) се грижат `/vmlinuz` да сочи към най-новото ядро, а `/vmlinuz.old` към следващото по "новост". При система с повече от две компилирани ядра се получава така, че не са налични всички ядра в списъка на LILO за зареждане, а само последните две. Всеки индивидуално може да намери решение на "проблема"(било то да се направи друга символна връзка към по-старо ядро, което обезателно трябва да присъства в списъка на LILO освен най-новите, или нещо друго)
2. Използването на опцията *–append_to_version* (разгледайте и *–revision*). Да речем, че имате сорса на ядро 2.4.18 и искате да компилирате няколко нови ядра от този сорс. Ако не се направят някакви промени, модулите на стари и нови ядра ще се озоват в една и съща директория (`/lib/modules/2.4.18`) и ще се получи бъркотия от различни модули. Едно от решенията на проблема е да се използва следната процедура за компилиране:

```
# make-kpkg clean
# make menuconfig
# rm -rf include/linux/version.h
# make-kpkg --append_to_version -1mykernel kernel_image
```

По този начин се избягва горепосоченият проблем, а новото ядро ще бъде именувано `vmlinuz-2.4.18-1mykernel`. Можете да сменяте стойността на *–append_to_version* при всяко компилиране, но трябва да се съобразявате с вече указаната процедура. Повече информация относно този проблем можете да намерите в съответните секции на `/usr/share/kernel-package/README` и *man* **make-kpkg**(1).

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 `alsaconf` (ако случайно притрябва). При инсталацията на пакета `alsa-source` ще бъдем попитани дали искаме да се компилират всички драйвери или само за конкертна звукова платка.

В `/usr/src` се появава 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

[9] http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=340742097

конфликтират с DFSG[10]. 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 инсталиращия 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 конфигуруцаионни 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 **alsaconf**(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:)

20.2. *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 случай може да опитате дали ще ви хареса програмата `stow`.

Програмата `stow` е част от проекта GNU[11] 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[12]
- Savannah[13]
- FTP достъп[14]
- дебиански пакет[15]

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:

- Manage packages using stow[16]
- Simple package management using stow[17]

[10] http://www.debian.org/social_contat#guidelines
[11] http://www.gnu.org
[12] http://www.gnu.org/software/stow/
[13] http://savannah.gnu.org/projects/stow/
[14] ftp://ftp/gnu.org/pub/gnu/stow/
[15] http://packages.debian.org/stow
[16] http://www-106.ibm.com/developerworks/linux/library/l-stow/?ca=dgr-lnxw02STOW
[17] http://www.linuxgazette.com/issue75/peda.html

Глава 21

Сигурност и надеждност

21.1. Документи и пакети

Обърнете внимание на http://security.debian.org[1].

- Security Team FAQ[2]
- Securing Debian Manual[3]

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 предишният Stable Release и настоящите Testing и Unstable. В повечето случаи се изкарват почти веднага поправки за всички засегнати издания (или Suites на Debian).

21.2. Пакети за анализ и оценка на сигурността

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

- `harden`
- `harden-tools`

21.3. Автоматизиран контрол върху правата на изпълнимите файлове

Подобно на всяка уважаваща себе си дистрибуция Debian предлага автоматизирана възможност за управление правата на изпълнимите файлове.

```
# apt-get install suidmanager
```

Принципът на управление е лесен. Преди всичко, трябва да подложите на щателен анализ системата си - кои програми имат suid-бит и се изпълняват с администраторски права, кой какво има право да изпълнява. Например, да допуснем, че има програми, които искат suid, за да се ползват нормално, но от друга страна не искате всеки да може да ги изпълнява, а само членовете на групата users. Тогава на помощ идва командата `dpkg-statoverride` от пакета `suidmanager`. Решаваме да променим правата върху файла `/usr/bin/cdrecord` така, че да се изпълнява само от членовете на групата users. Ето какво ни казва и помощната информация за начина на използване на тази команда:

```
# dpkg-statoverride [options] --add <owner> <group> <mode> <file>
```

Значи трябва да изпълним само:

```
# dpkg-statoverride --update --add root users 4754 /usr/bin/cdrecord
```

[1] http://security.debian.org
[2] http://www.debian.org/security/faq
[3] http://www.debian.org/doc/user-manuals#securing

Така казахме на `dpkg` да промени групата на файла и да разреши само на членовете на тази група да го изпълняват независимо от задължителния suid-бит, който е необходим на cdrecord, за да се стартира от потребители без администраторски привилегии. Сега можем да бъдем спокойни, че и след поредния ъпгрейд на пакета, този изпълним файл ще получава именно тези права от `dpkg`. Всъщност, тази информация се записва във файла `/var/lib/dpkg/statoverride` и има съвсем разбираем вид, дори може да бъде редактиран ръчно.

Бърз преглед на опциите на командата `dpkg-statoverride`:

```
debian-nikola:~# dpkg-statoverride --help
Debian dpkg-statoverride 1.10.18.
Copyright (C) 2000 Wichert Akkerman.
This is free software; see the GNU General Public Licence version 2 or later
for copying conditions.  There is NO warranty.
Usage:
  dpkg-statoverride [options] --add <owner> <group> <mode> <file>
  dpkg-statoverride [options] --remove <file>
  dpkg-statoverride [options] --list [<glob-pattern>]
  Options:
  --update                 immediately update file permissions
  --force                  force an action even if a sanity check fails
  --quiet                  quiet operation, minimal output
  --help                   print this help screen and exit
  --admindir <directory>   set the directory with the statoverride file
```

Тази програма може да ви послужи и не само за изпълнимите файлове, а и за всеки файл или директория, чиито права искате да бъдат специфични.

21.4. Подписване на пакети - Debian keyring

GPG/PGP ключовете на debian maintainers можете да получите по следните начини:

- keyring.debian.org[4], на този хост има и `rsync` сървър, module `keyring`
- http://ftp.debian.org/debian/doc/debian-keyring.tar.gz[5]
- пакета `debian-keyring`
- finger `maintainer`@db.debian.org

21.4.1. Debian source packages - подписване на сорса

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:
 57f11fc13458d8a59894df099449ddbe 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
yxs7/GOpnUWPW9ND517mtnW7E0ZAOcZb0oj50wW5PS0ylRIuQui1IaNv0comzoRv
TACbvUv99j9eAcRTs+qYjnuX8MKTIVO7
=uKkZ
-----END PGP SIGNATURE-----
```

Можете да проверите сигнатурата чрез **dscverify**(1), което се намира в пакета `devscripts`.

21.4.2. Debian binary packages (deb's) - per-deb подписване

http://dpkg-sig.turmzimmer.net[6] - индивидуално подписване.

[4] http://keyring.debian.org
[5] http://ftp.debian.org/debian/doc/debian-keyring.tar.gz
[6] http://dpkg-sig.turmzimmer.net

21.4.3. Debian `Release.gpg` files - per-Archive подписване

http://monk.debian.net/apt-secure/[7] - групово подписване.

FIXME: да се обясни повече за Debian keyring, като подписване на пакетите и проверка[8] и ползването от maintainer's на `debsigs`. FIXME: някой ползвал ли е http://people.debian.org/~ajt/apt-check-sigs[9]?

21.5. Прилагане на security updates за множество машини

Пример 11: **Настройка на apt-proxy**

Ето например как можете да процедирате, 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 MB-ти от security.debian.org[10].

За да не се хаби излишно трафик, можете 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 портът, 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
```

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-proxy-то и при повторна заявка от друг сървър пакетът не се тегли наново, а се взема от локалната директория. Наистина спестява доста трафик.

[7] http://monk.debian.net/apt-secure/

[8] http://www.debian.org/doc/manuals/securing-debian-howto/ch7.en.html#s-deb-pack-sign

[9] http://people.debian.org/~ajt/apt-check-sigs

[10] security.debian.org

21.6. Контрол върху правата на файловете и директориите с помощта на `acl`

От известно време насам всички разпространени файлови системи в Линукс поддържат разширените свойства за контрол над правата, известни като `ACLs` (Access Control Lists) или "Списъци за контрол на достъпа". Въпросните разширения са базирани на стандарта POSIX 1003.1e. Накратко ще обясним какъв е техният смисъл. Както всеки от вас знае, т. нар. Unix permissions или просто стандартните права в една Unix/Linux-система са организирани съвсем просто: те се дефинират по собственик, група и останали потребители. В голяма част от случаите тази организация е достатъчна. Когато имаме усложнени нужди обаче, тя се оказва неудобна. Представете си, че имате директория, която може да се чете само от собственика и на която сме дали `chmod 700`. Искате да разрешите на друг да влиза в нея. Тогава правите нова група, в която включвате двамата потребители и променяте правата на директорията така, че освен от собственика да може да се чете и от групата, което става с командата `chmod 750`. Чудесно. А ако искате да направите същото и с други директории и други потребители? Трябва да създавате нови групи и така докато стигнете до момент, в който вече просто изпускате нишката. Ето тук на помощ идват "Списъците за контрол на достъпа". С тях можем лесно да делегираме разнообразни права на всеки върху всичко, без да пипаме нито потребителите, нито групите, нито да променяме стандартните Unix permissions на директорията/файла, т.е. без да си играем с команди като `chmod` или `chown`. Важно е да отбележим, че използването на ACLs не пречи с нищо на прилагането на стандартните команди за управление на Unix-правата. Двата метода съжителстват съвсем мирно.

Като за начало трябва изпълним командата:

```
apt-get install acl
```

Този пакет съдържа двата основни инструмента за управление на ACLs, които ще се превърнат в нашата дясна ръка.

Следва да редактираме файла `/etc/fstab`, за да кажем на ядрото да включи поддръжката на ACLs за съответните дискови дялове. Това става лесно, с добавяне на опция `acl` и премонтиране на съответния дял. Тук е мястото да отбележим, че Линукс поддържа `acl` за Ext2/Ext3 едва в последните версии на ядрата от серията 2.4 и в 2.6. Ако ползвате файловата система XFS, не е необходимо да пипате нищо в fstab, защото тази файлова система идва по подразбиране с вградена поддръжка на ACLs. Ако искате поддръжка на ACLs в Линукс, препоръчително е да се спрете на XFS или Ext2/Ext3 със съответната версия на ядрото, която поддържа опцията `acl` (тя е нужна само за Ext2/Ext3).

Пример за `fstab` за Ext2/Ext3.

```
/dev/hda2     /home     ext3     rw,usrquota,nosuid,nodev,noatime,acl     0     0
```

С командата `getfacl` извличаме информация за acl-статуса на дадения файл или директория.

```
# getfacl shots/
# file: shots
# owner: nikola
# group: nikola
user::rwx
group::r-x
other::r-x
```

Така изглеждат правата на всяка директория по подразбиране. В момента тази директория е собственост на nikola, но всеки може да влиза и да чете в нея.

```
chmod 700 shots/
ls -l shots
drwx------    5 nikola   nikola        111 2004-06-13 09:57 shots/
```

Сега вече никой друг не може да влиза в тази директория освен нейния собственик. Това е чудесно, но да речем, че искаме да публикуваме директорията в уеб. Тогава трябва да разрешим на уебсървъра да я чете. Това ще стане с командата `setfacl`. В Debian потребителят на уебсървъра е www-data.

```
setfacl -m u:www-data:r-x shots/
getfacl shots/
# file: shots
# owner: nikola
# group: nikola
user::rwx
user:www-data:r-x
group::---
group::---
mask::r-x
other::---
```

В този случай ние разрешихме на потребителя (флагът за потребител е `u`, а за група - `g`) www-data да чете и да изпълнява (флаговете `r` и `x` са познати от командата `chmod`) в тази директория.

Тук решихме една проста и банална задача, с което искахме да обясним простичко и накратко какъв е смисълът от използването на ACLs. Всички опции и възможности на командата `setfacl` са описани подробно в съответната man-страница.

Глава 22

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.

22.1. Да си направим сами LiveCD images

22.1.1. dfsbuild

Вместо да използвате готови LiveCD images като Knoppix, можете да използате тази програмка за да изградите свой LiveCD image: Debian From Scratch[2] Огледален сървър в България е: ftp://ftp.uni-sofia.bg/cd-images/linux/debian/dfs/[3] Както обичайно документацията е на сайта на автора и в `/usr/share/doc/dfsbuild/`. Програмата е доста гъвкава и предлага множество опции в конфигурационния си файл. Примерен конфигурационен файл може да намерите на:

http://www.debianbookbg.org/utils/home/danchev/etc/dfsbuild/[4]

22.1.2. debix

debix: създаване на `live filesystems` от съществуваща такава или отначало.

http://alioth.debian.org/projects/debix[5]

22.2. Debian GNU/Linux chrooted install with `debootstrap`

Colin Walters shows how to chroot your Universal Operating System[6]

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 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 ;-).

22.3. Debian GNU/Linux с LVM (вкл. root filesystem)

Използване на Debian GNU/Linux с LVM (вкл. root filesystem)[7]

[1] http://www.debian.org/releases/stable/installmanual
[2] http://people.debian.org/~jgoerzen/dfs/
[3] ftp://ftp.uni-sofia.bg/cd-images/linux/debian/dfs/
[4] http://www.debianbookbg.org/utils/home/danchev/etc/dfsbuild/
[5] http://alioth.debian.org/projects/debix
[6] http://people.debian.org/~walters/chroot.html
[7] http://www.21chouse.com/deb_lvm.htm

22.4. Debian GNU/Linux TakeOver Installations

- Official cross-install howto[8]
- Remote install/convert към Debian изцяло през ssh[9]
- Chrooted Debian install from base image[10]

22.5. Debian GNU/Linux върху x86 чрез PXE

Използване на Debian GNU/Linux вурху x86 чрез PXE[11]

22.6. Debian GNU/Linux върху Apple iBook

Branden Robinson shows Installing Debian 3.0 onto an Apple iBook without using any physical media! (fast Internet connection recommended) Използване на Debian GNU/Linux върху Apple iBook[12]

22.7. Debian GNU/Linux върху Sun Sparc Station or X terminal (netboot)

Използване на Debian GNU/Linux върху Sun Box[13]

22.8. Debian GNU/Linux върху Sony Vaio SRX87

Sony Vaio SRX87[14]

22.9. Debian GNU/Linux върху Acer Tablet PC

Running Debian on an Acer Tablet PC[15]. 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[16] in detail how the system is booted from the network and how X needs to be configured in order to support the pen.

22.10. Debian GNU/Linux върху MS X-Box

Инсталация на Debian GNU/Linux върху MS X-Box. Ed's Debian е пълна Debian-базирана GNU/Linux дистрибуция, която съдържа последните предимства на X-Box Linux разработването.

- Изтегляне на дистрибуцията[17]
- Как да инсталираме Debian GNU/Linux на Xbox[18]
- За deb-пакетите поставете в `/etc/apt/sources.list` следният ред:

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

[8] http://www.debian.org/releases/stable/i386/ch-preparing.en.html#s-linux-upgrade
[9] http://trilldev.sourceforge.net/files/remotedeb.html
[10] http://lists.debian.org/debian-user/2002/debian-user-200204/msg01010.html
[11] http://www.debianplanet.org/node.php?id=818
[12] http://people.debian.org/~branden/ibook.html
[13] http://www.tomun.org/docs/sunbox.html
[14] http://www.differentpla.net/~roger/hardware/vaio/linux/
[15] http://global.acer.com/products/tablet_pc/tmc100.htm
[16] http://prometheus.physics.ucsb.edu/~dean/TmC100/AcerTmC100.html
[17] http://sourceforge.net/project/showfiles.php?group_id=54192
[18] http://xbox-linux.sourceforge.net/articles.php?aid=2002248060056

22.11. Debian/GNU Linux върху HP PA-RISC

22.11.1. 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
```

22.11.2. Документация

- Debian for PA-RISC[19]
- The PA-RISC Linux Project Web[20]
- Installing PA-RISC Linux[21]
- PALO PA-RISC/Linux Boot Loader[22]
- PA-RISC/Linux Boot HOWTO[23]
- HP Systems Documentation[24]
- HP PA-RISC Architecture Reference Documents[25]
- The OpenPA Project[26]

22.11.3. Начало на инсталацията

ISO-та на Woody за HPPA[27]

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
     -------                              -----------
     BOot [PRI|ALT|<path>]                Boot from specified path
     PAth [PRI|ALT] [<path>]              Display or modify a path
     SEArch [DIsplay|IPL] [<path>]        Search for boot devices
     COnfiguration menu                  Displays or sets boot values
     INformation menu                     Displays hardware information
     SERvice menu                         Displays service commands
     DIsplay                              Redisplay the current menu
     HElp [<menu>|<command>]              Display help for menu or command
     RESET                                Restart the system
____
Main Menu: Enter command or menu >
```

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 (mnem)  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 >
```

[19] http://www.debian.org/ports/hppa/
[20] http://parisc-linux.org/
[21] http://parisc-linux.org/software/install.html
[22] http://ftp.parisc-linux.org/cgi-bin/cvslite/palo/README.html
[23] http://pateam.esiee.fr/parisc-linux-boot/doc.html
[24] http://docs.hp.com/hpux/hw/
[25] http://h21007.www2.hp.com/dev/
[26] http://www.openpa.net/
[27] ftp://gsyprf10.external.hp.com/debian-cd/3.0_r0/hppa/

В този случай 0/0/2/0.1 ми е CD-ROMa от който искам да boot-на. Тъй като в Linux наименованието 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-hppa използва за boot loader PALO, а не LILO (както се вижда по горе), и виждаме познатия Debian Installer. (FIXME: Проблеми)

22.11.4. 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 детектнал)
- веднъж след като се инсталирали Debian GNU/Linux във firmware-а на машината, можете на следващия reboot да кажете кой ще е primary boot path-а (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 runва palo, когато се променя кернела
  - hp firmware/PALO изискват един служебен partition тип `f0`, в който ще се съдържат secondary boot loader image и recovery kernel, препоръчва се големина 16MB (аз лично направих моя 32 MB)
  - partition-а който ще съдържа обикновенния кернел, който тръгва по default (а не recovery kernel-а), трябва да е във първите 2 GB от диска. И тук имаме два варианта — или `f0` 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 обикновенно 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       (recovery дял)
  sda2                 Primary  Linux                           32.51       (/boot)
  sda3                   Primary  Linux                           1024.46       (/)
```

```
sda5                Logical   Linux swap                  2047.87       (swap)
sda6                Logical   Linux                       1024.46       (/tmp)
sda7                Logical   Linux                       2047.87       (/var)
sda8                Logical   Linux                       2047.87       (/usr)
sda9                Logical   Linux                      28445.77       (/home)
```

Продължаваме по стандарния 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.

22.11.5. 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-а по традиционния начин.

22.12. Debian GNU/Linux върху SGI MIPS (netboot)

22.12.1. 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)

22.12.2. Документация

- Linux/MIPS HOWTO[28]
- SGI Performance Comparisons[29]
- SGI Technical Advice and Information[30]
- Debian for MIPS[31]
- How to install Debian GNU/Linux on Indy (MIPS)[32]
- Indytech — Hardware technical information for the SGI Indy computer[33]
- How to install Debian GNU/Linux on Indy (MIPS)[34]
- Google[35]

22.12.3. Начало на инсталацията

Изтеглете **kernel-a**[36]. За съжаление ftp.bg.debian.org[37] 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. десктоп 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 които се вижда едно бутонче „stop for maintenance“. Натискате го и ви излиза системното меню на Indy-то със следните опции:

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

[28] http://howto.linux-mips.org/mips-howto.ps
[29] http://futuretech.mirror.vuurwerk.net/perfcomp.html
[30] http://futuretech.mirror.vuurwerk.net/sgi.html
[31] http://www.debian.org/ports/mips/
[32] http://www.linux-debian.de/howto/debian-mips-woody-install.html
[33] http://www.reputable.com/indytech.html
[34] http://www.pvv.org/~pladsen/Indy/HOWTO.html
[35] http://www.google.com
[36] http://ftp.fi.debian.org/debian/dists/woody/main/disks-mips/current/r4k-ip22/tftpboot.img
[37] ftp://ftp.bg.debian.org

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 arp-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: `<kernelsource>/Documentation/filesystems/proc.txt`, `lkml`, google[38]

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:

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

или

```
# kill -HUP `ps auwx |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 tfptd или някаква друга специфична inetd услуга) по-добре е да нямате пуснат inetd. Един service по-малко, малко по-спокоен сън :) 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`.

[38] http://www.google.com

22.12.4. 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 стоиност 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-а, 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-а, 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-а 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-а е 1620 sectora * 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-а, 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 ограничa 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-а да е 1x или 2x обема на паметта. Да не говорим че в някои други UNIX-и е почти задължително swap-а да е минимум колкото паметта на машината, иначе ако параметъра swapmem_on е сетнат на 0-а, което осигурява директен mapping м/у паметта и swapа, за да се избегнат по-тежките операции по адресиране на паметта и за да се спести памет от т.нар. псевдо swap, OS-а ще ползва толкова RAM, колкото е голям swap дяла. Long story short: правете си swap дяла поне два пъти колкото RAM-а, изгражда полезни навици.

И така: 128 MB = 134217728 bytes = 262144 sector-а ни трябват 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-а) 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 **вашите** 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`.

22.12.5. Довършителни процедури

Следващата стъпка е инсталация на кернел върху новата машина: избираме „from internet“ (FIXME: описание на инсталация от NFS). Настройваме „network settings“ (или си пускаме пак DHCP сървъра на сервизната машина и му казваме да ползва DHCP). Въпрос на вкус и възможности е от къде ще дръпне кернела 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-mips[39], така че нататък инсталацията ще върви по-бързичко :) За съжаление другият Debian mirror в България, http://debian.ludost.net[40], 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[41], като не трябва да се забравя proxy-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-а 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 стандарния 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-а по традиционния начин.

Препратки:

– Някои „хитрости“ от Рик Моен[42]

[39] http://ftp.bg.debian.org/debian/dists/woody/main/binary-mips/
[40] http://debian.ludost.net
[41] http://ftp.bg.debian.org/debian
[42] http://www.linuxmafia.com/debian/tips

22.13. Debian/GNU Linux върху AMD64

Докато влезе официално в Sid софтуера и документацията ще са достъпни от:

- http://www.jukie.net/~bart/debian/amd64/[43]
- http://debian-amd64.alioth.debian.org/[44]

FIXME: да се даде пример за инсталация.

[43] http://www.jukie.net/~bart/debian/amd64/
[44] http://debian-amd64.alioth.debian.org/

22.14. Debian GNU/Linux в клъстер чрез 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 спомена и тази тема.

Ето няколко примера:

- Debian Beowulf Project[45]. На страницата с потребителите на Debian[46] 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[47]: 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[48] 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[49]: At Doshisha University, Debian „potato“ runs on 256-node Beowulf cluster[50]. 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[51] и в големи компютърни центрове като SARA, Netherlands[52]
  - 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[53]: 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

- OpenMosix 1[54]
- OpenMosix 2[55]
- OpenMosix 3[56]

която може да разясни някои недоразумения в този дух. В случая се визира clustering чрез OpenMosix[57] и примера е даден с Gentoo, но спокойно можете да си мислите същото и за Debian, защото Mosix (kernel patches[58] and userland tools[59]), а впоследствие и OpenMosix (kernel patches[60] and userland tools[61]) присъстват отдавна в неговия архив, но дори и да не присъстваха нямаше да бъде болка за умиране — 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[62] защо е GPL'ed, и защо е било необходимо да се основава той, наследявайки своя

[45] http://www.debian.org/ports/beowulf/
[46] http://www.debian.org/users/
[47] http://www.ai.mit.edu/
[48] http://www.ai.mit.edu/sysadmin/cluster.html
[49] http://www.doshisha.ac.jp/
[50] http://cambria.doshisha.ac.jp
[51] http://www.sara.nl/beowulf/
[52] http://www.sara.nl
[53] http://www.qlilinuxpc.com/products/clustering/
[54] http://www-1.ibm.com/servers/esdd/articles/openmosix.html?t=gr,lnxw02=OpenMosix1
[55] http://www-1.ibm.com/servers/esdd/articles/openmosixpart2.html?t=gr,lnxw02=OpenMozix2
[56] http://www-1.ibm.com/servers/esdd/articles/openmosixpart3.html?t=gr,lnxw02=OpenMosix3
[57] http://www.openmosix.org
[58] http://packages.debian.org/kernel-patch-mosix
[59] http://packages.debian.org/mosix
[60] http://packages.debian.org/kernel-patch-open-mosix
[61] http://packages.debian.org/openmosix
[62] http://www.openmosix.org

предшественик Mosix[63]. 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-съвършенни аналози за General Computing usage, винаги съм готов да слушам, но технически аргументи, а не сухи рекламни трикове подвеждащи потребителя.

[63] http://www.mosix.com

Глава 23

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 сорса 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[3], 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, свързването 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[4]

  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.

Нова система за *build* на пакети се готви от Colin Walters. Тя се нарича версия 2[5] на source пакетите и в нея се адресират сложността на сегашните `debian/rules` файлове, както и многото излишък в тях. Новата система е силно повлияна от *u-os* пакетната система[6] на Christoph Lameter. Основната идея е да се направят нещата прости, а сложните неща — възможни.

В тази връзка доста се спекулира с т.н. source-based distributions, като Gentoo, при които софтуерът се компилира направо на машината на потребителя. Естествено, това може да стане и с debian source packages, въпреки че малко потребители го знаят;-). Основната идея е, че така могат да се използват по-добре оптимизациите на компилатора за съответния процесор. В това има доза истина, обаче се отнася само до т.н. CPU-intensive приложения като аудио и видео плеъри, библиотеки в които са реализирани някакви тежки математически или криптографски алгоритми, въобще програми, които могат да се възползват от конкретните за дадения процесор инструкции. Може да се окаже дори, че разликата между процесорите от x86 архитектурата не е забележима, докато при процесорите от Sparc архитектурата например едно такова оптимизиране дава сериозно отражение върху производителността. Затова не е лошо някои CPU-intensive приложения да се прекомпилират на място и да се оптимизират от source packages, ebuilds, ports и т.н., но да се

[1] http://u-os.org/upm.html
[2] http://u-os.org/tokyo.html
[3] http://snapshot.debian.net
[4] http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html#AEN29
[5] http://lists.debian.org/debian-devel-0211/msg02630.html
[6] http://www.u-os.org/upm.html

очаква, че всички приложения ще се възползват от едно такова оптимизиране, е меко казано несериозно. Тук са замесени възможностите на съответното CPU, възможностите на съответния компилатор да оптимизира за него, както и самият на приложениетоу което се опитваме да оптимизираме. Ако последното е много зле, то никакъв компилатор и оптимизатор не може да ви помогне, затова най-добре се оптимизаира от изходния код на самото приложение. Ето едно сравнение на Debian за i386, Mandrake за i586 и Gentoo[7]. Или с казано накратко, подобни оптимизации са определящи от гледана точка на производителността в много специфични случаи и не е казано, че само source-based distros могат да компилират своите пакети на потребителската машина. Debian предлага впечатляващо много инструменти по въпроса, като доста от тях са разгледани в предните глави на тази книга.

Вече като малко по-напреднали трябва да четете:

- Maint Guide[8] или стария превод на български[9]
- Debian Developer's Reference[10]
- Debian Policy Manual[11]
- и въобще Debian Developers' Corner[12]

[7] http://articles.linmagau.org/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=227&page=1
[8] http://www.debian.org/doc/maint-guide/
[9] http://debian.gabrovo.com/docs/maint-guide/index.html
[10] http://www.debian.org/doc/developers-reference/
[11] http://www.debian.org/doc/debian-policy/
[12] http://www.debian.org/devel/

Глава 24

More

Fwd: Re: lug-bg: Gentoo ebuilds[1] 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 обърнат 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 FreeBSD[2], NetBSD[3] и OpenBSD[4]. 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 Collection[5], 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 преведа или да давам съвети как се upgrade-ва една такава система (имайки предвид всичките й 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 Source[6]) 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 FLOSS[8] 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 Commision[9]. Та защо и по-новите потребители да не се възползват от всичко това, освобождавайки 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/
[6] http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/synching.html
[7] http://floss1.infonomics.nl/floss1/stats_13.html
[8] http://www.infonomics.nl/FLOSS/report/
[9] http://www.europa.eu.int/comm/index_en.htm

Глава 25

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 регистрираните 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 26

Обобщение

Надявам се ви се поизясниха повечето от представените по-горе неща. Ето и една таблица, схематично представяща какво може да съдържа вашият Debian:

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 благини, 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 конфликтира. 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 a the San Jose extension of the University of Phoenix. Subjects taught vary from Critical Thinking (PHL/251) to Introduction to Unix (POS/420), Programming Concepts using C++ (POS/370), Networks and Telecommunications (NTC/410) to classes on management of programing 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 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) аналози 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* ;-)

- **B1: 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)
- **B2: Къде в България мога да си изтегля или закупя Debian, Knoppix и т.н. CD-та?**
  O2: Можете да изтеглите ISO-файлове от няколко сървъра в България:
  - ftp://ftp.bg.debian.org/debian-cd/[1]
  - http://mirrors.ludost.net/cd-images/[2]
  - ftp://ftp.uni-sofia.bg/debian-iso/[3]
  - ftp://ftp.uni-sofia.bg/cd-images/[4]
  - http://d.linux-bg.org[5]

  Ако имате стари CD/DVD дискове, можете да ги upgrade-нете с `jigdo`. Т.е. ще се изтеглят само новите файлове, старите ще се вземат от старите ви дискове. Дистрибуционни CD-та на Debian, книги, програми и т.н можете да си закупите от CDforME[6]. Книги, CD-та и други материали, могат също така да се закупят от каталога на Amazon.com[7], като съответно съществува по-приемлив вариант за разплащане и доставка в България[8] — фирмата срещу съответен процент доставя стоката по ISBN номер.
- **B3: Необходимо ли е да си изтегля (download) всичките 5/7/9 Debian CD-та?**
  O3: Ако разполагате с връзка към Интернет, достатъчно е да имате само първото CD **binary-1_NONUS.iso**. Оттам може да се инсталира минималната система, достатъчна да довършите мрежова инсталацията (да свалите всичко останало от Интернет).
- **B4: Кои сървъри са най-подходящи за включване в /etc/apt/sources.list за България?**
  O4: Няколко са:
  - ftp.bg.debian.org[9]
  - debian.ludost.net[10]
  - ftp.uni-sofia.bg[11]

  Или в **/etc/apt/sources.list** може да имате:
  ```
  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[12]
- **B5: Не намирам някои пакети в официалния архив, а преди време ги имаше. Къде да ги търся?**
  O5: Snapshots на официалния Debian архив можете да претърсвате тук: http://snapshot.debian.net[13]. На сайта е описано как се правят тези snapshots, как да се претърсват и ползват с **apt-get**(8) от **sources.list**(5).
- **B6: Има ли пощенски списъци (mailing lists) за Debian в България?**
  O6: Да, има такива http://lists.zadnik.org/cgi-bin/mailman/listinfo[14].
- **B7: Има ли свързани с Debian български софтуерни проекти?**
  O7: Погледнете проекта bgdebian[15].
- **B8: Защо при *apt-get update* получавам грешка от вида:**
  ```
  Reading Package Lists... Error!
  E: Dynamic MMap ran out of room
  ```
  O8: 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`:

[1] ftp://ftp.bg.debian.org/debian-cd/
[2] http://mirrors.ludost.net/cd-images/
[3] ftp://ftp.uni-sofia.bg/debian-iso/
[4] ftp://ftp.uni-sofia.bg/cd-images/
[5] http://d.linux-bg.org
[6] http://www.cdforme.net/default.php?cPath=20_24
[7] http://www.Amazon.com
[8] http://shop.global.bg/helpbg_n.asp
[9] ftp.bg.debian.org
[10] debian.ludost.net
[11] ftp.uni-sofia.bg
[12] http://www.debian.org/mirrors/mirrors_full#BG
[13] http://snapshot.debian.net
[14] http://lists.zadnik.org/cgi-bin/mailman/listinfo
[15] http://sf.net/projects/bgdebian/

```
APT::Cache-Limit 25165824
```

– **B9: Как да намирам и инсталирам при поискване съответните пакети съдържащи `development libraries and header files` необходими за компилацията и свързването на приложения от `upstream sources`?**
O9: Можете да wrap-нете процеса по търсене на съответните файлове в системата, компилирането на сорса и свързването на обектните файлове. При липсващи файлове, ще бъдете попитани за разрешение за инсталирането на съответните пакети, ако са достъпни от посочените от вас източници. Даден файл може да се среща в няколко пакета, при което ще избирате. Запознайте се с **auto-apt**(1):

```
# apt-get install auto-apt
# auto-apt update updatedb update-local
# cd arbitrary-src-dir/
# auto-apt run ./configure --your-options-here
# auto-apt run make
```

– **B10: 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
```

O10: 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 времено *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 денонстрация пакета `scrollkeeper`.

– **B11: Има ли начини за автоматично разпознаване и конфигуриране на хардуера?**
O11: За целта има програми като `discover` и `kudzu`, който 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.

– **B12: А някакви журнални файлови системи да се ползват в Debian?**
O12: Да, като навсякъде другаде, зависи от ядрото което, ползвате, и няколко user-space utils. Може би е добре да започнете оттук: http://people.debian.org/~blade/[16]

– **B13: Защо е необходим този Perl horror с debconf, debhelper, kernel-package и т.н.**
O13: 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[17] често е наричан „щвейцарската 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.

– **B14: Мога ли и аз да upload-вам пакети в Debian?**
O14: Може би. За официалния Debian архив се обърнете към http://nm.debian.org[18]. Иначе неофициален архив всеки може да си направи.

– **B15: Как мога да помогна на проекта Debian?**
O15: Обърнете към http://www.debian.org/devel/join/[19].

– **B16: Има ли българи участници в проекта Debian?**
O16: Има. Обърнете се към http://db.debian.org[20] и http://www.debian.gr.jp/~kitame/maint.cgi[21].

– **B17: Аааа Debian може ли . . . ?**
O17: Цитираме Christoph Lameter: **Do not ask me: Does Debian support this and that. Debian supports everything.**

Други страници с отговори на въпроси:

– Официалната Debian страница с отговори[22]

– Често задавани въпроси във freenode `#debian` IRC канал[23]

[16] http://people.debian.org/~blade/
[17] http://www.perl.com
[18] http://nm.debian.org
[19] http://www.debian.org/devel/join/
[20] http://db.debian.org
[21] http://www.debian.gr.jp/~kitame/maint.cgi
[22] http://www.debian.org/doc/FAQ/
[23] http://www.linuks.mine.nu/debian-faq/

- Въпроси/отговори за Debian CD-тата[24]
- Работа с Java в Debian[25]
- http://www.debian-administration.org[26]
- http://www.debianplanet.org[27]
- http://planet.debian.org[28]
- http://www.maximumdebian.org[29]
- http://www.debian-enterprise.org[30]
- KDE3 FAQ[31]
- Руска страница с отговори на въпроси, свързани предимно с кирилизацията на Debian[32]
- Управление на *rpm* пакети с `apt`[33]

[24] http://www.debian.org/CD/faq/
[25] http://www.debian.org/doc/manuals/debian-java-faq/
[26] http://www.debian-administration.org
[27] http://www.debianplanet.org
[28] http://planet.debian.org
[29] http://www.maximumdebian.org
[30] http://www.debian-enterprise.org
[31] http://davidpashley.com/debian-kde/faq.html
[32] http://linux.perm.ru/doc/distro/debian/3.0/
[33] http://apt4rpm.sourceforge.net/faq.html

Част VI

Участие в писането на книгата

Глава 27

За този документ и авторите

27.1. Замисъл и стил на писане

- Книгата трябва да е полезна като съдържание и начин на дистрибутиране, както за начинаещи, така и за напреднали. Това означава, че в началото на книгата матариалът трябва да е лек и достъпен за колкото може повече потребители, като сложността се увеличава с всяка следваща глава.
- Книгата трябва да отразява по-рядко срещани и познати възможности на Debian, дори нехарактерни за официалната документация (например custom installers, защо не и да поддържаме сорсове на такива в *utils/*, non-x86 personal arch experience и др.)

27.2. Препоръки към авторите

FIXME: Тази секция може би се нуждае от обсъждане.

Всеки, който добавя някакъв код, би трябвало да:

- Добави 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-book@lists.zadnik.org[1]>.

Книгата е голяма, затова не може да се очаква всички да я прочетат от началото до края, като при това да запомнят всяко нещо къде е. Поради тази причина трябва да се отделя специално внимание къде точно поставяте своите материали и какво заглавие им слагате. Ръководното правило е да се поставяте 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 двоеточие. 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}
```

Избягвайте да пишете команди, вградени в самите абзаци на текста. Отделяйте ги в `\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 практична 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 и директории е добре да бъдат опаковани в `\texttt`
- Термините и понятията е добре да бъдат опаковани в `\textit`
- Имена на менюта и на текст, съдържащ предупреждение, на което трябва да се наблегне, е добре да се опакова в `\textbf`
- Извадки от съдържание на файлове е добре да бъдат опаковани в среда `verbatim`

[1] http://zadnik.org/cgi-bin/mailman/listinfo/debian-book

Глава 28

Регистрация за авторите

За да бъдете добавен в проекта, трябва да имате регистрация като потребител на системата, която хоства проекта. Името на вашата регистрация трябва да изпратите на администратора на debian-book, който в момента е Никола Антонов <nikola@linux-bg.org>.

Въпреки че проектът има администратор, това не трябва да ви спира да давате всякакви предложения относно развитието на проекта, като се очаква и да можете да осъществявате тези предложения. В този смисъл функцията на администратора е по-скоро техническа. Ако покажете способности в това отношение и имате достатъчно време, може и да ви изберем (ние, участниците в книгата) да бъдете „глава“ на проекта.

Глава 29

Работа с LaTeX

Преди да комитвате в SVN хранилището, проверявайте LaTeX кода за коректност:

- Един добър инстарумент е `lacheck`. Подавате му главния файл `debian-book.tex` като аргумент и всички други файлове включвани от него ще бъдат проверени за коректност.
- Също така се убедете, че се компилира при вас. Винаги тествайте с PDF изхода.

29.1. Инсталиране на LaTeX

LaTeX 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, PostSrypt, PDF, HTML. Ако сте инсталирали някоя дистрибуция на GNU/Linux, можете да проверите дали имате инсталирани LaTeX (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 LaTeX не е достатъчно добре кирилизирана и не е много наясно с българския език. Именно затова, преди да се впуснем в не особенно сложния процес на създаване на PDF, PostScript и т.н. изходи от LaTeX, е нужно да инсталираме няколко пакета.

bgtex-v2 е първият пакет, който е нужен за българизация на teTeX. Можете да го изтеглите от ftp://lml.bas.bg/home/anton/tex/[1] 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 рапространението си. Можете да дръпнете пакета от: ftp://lml.bas.bg/home/anton/tex/[2] 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
`/etc/texmf/dvips/config.ps`	`p +cyrfonts.map`
`/etc/texmf/pdftex/pdftex.cfg`	`map +cyrfonts.map`

Ако не можете да намерите някой от тези два файла, пробвайте със следните алтернативни имена:

`/etc/texmf/dvips/config.ps`	`/usr/share/texmf/dvips/config/config.ps`
`/etc/texmf/pdftex/pdftex.cfg`	`/usr/share/texmf/pdftex/config/pdftex.cfg`

29.2. Писане на LaTeX

Тук следват леки насоки за езика LaTeX и как го ползваме в нашия документ. Добри източници за езика LaTeX са:

- A beginner's introduction to typesetting with LaTeX [3] За да разберете както е TeX ,LaTeX и `teTeX` започнете от тук
- Comprehensive TeX Archive Network[4]

[1] ftp://lml.bas.bg/home/anton/tex/
[2] ftp://lml.bas.bg/home/anton/tex/
[3] http://www.ctan.org/ctan/tex-archive/info/beginlatex/
[4] http://www.ctan.org

- TeX Users Group[5]
- LaTeX Project[6]
- teTeX[7]
- The TeX catalogue online[8]
- UK List of TeX FAQ[9]
- Справочник latexhelp.html[10]
- Hypertext Help with LaTeX [11]

Нека се спазва установеният LaTeX стил, както и стилът на поднасяне на информацията — ясно, точно и ако може да се привеждат и примери. Където има съмнения или все още не е довършено нещо, нека се поставя FIXME. Добра идея ще е за повече подробности относно LaTeX кода да се обърнете към кода на самия документ в директорията `src/`, и ще добиете представа, че хич не е страшно:

LaTeX е мощен език, но ние ползваме много малка част от него. Например:

За части, глави, подглави и т.н. разделението е такова:

```
\part{заглавие}
\chapter{заглавие}
\section{заглавие}
\subsection{заглавие}
\subsubsection{заглавие}
```

29.2.1. Форматиране на текста

```
Абзаците се пишат направо.  Между отделните абзаци трябва да има поне
един празен ред.
\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"' щото 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"' щото не можем да го escape-нем, иначе се ползва с едно ;-)
  Cheating \ thru \textbackslash doesn't help either :-( FIXME:
```

29.2.2. Списъци

```
\begin{itemize}
\item Debian
\item GNU
\item Linux
\end{itemize}
```

- Debian
- GNU
- Linux

```
\begin{enumerate}
\item Едно
\item Две
\item Три
\end{enumerate}
```

1. Едно
2. Две
3. Три

[5] http://www.tug.org
[6] http://www.latex-project.org
[7] http://www.tug.org/tetex
[8] http://datamining.csiro.au/tex/
[9] http://www.tex.ac.uk/cgi-bin/texfaq2html
[10] http://www.debianbookbg.org/utils/latexhelp.html
[11] http://www.giss.nasa.gov/latex/

29.2.3. 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	централно	дясно
още ляво	още централно	още дясно

29.2.4. Специални макроси

В книгата се използват макроси, които не са част от стандартния LaTeX. Използвайте ги винаги, когато е възможно.

LaTeX	**Изглед**
`\hlink{http://www.debian.org}{http://www.debian.org}`	http://www.debian.org[12]
`\hlink{Заглавна страница}{http://www.debian.org}`	Заглавна страница[13]
`\deb{vsftpd}`	`vsftpd`
`\man{dpkg}{8}`	**dpkg**(8)
`\manx{apt-get}{8}`	**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`.

29.2.5. Специални знаци

Някои знаци в LaTeX са специални и трябва да ги напишете по определен начин, за да ги включите в документа, който пишете:

Знак	**Заместител**
#	`\#`
$	`\$`
%	`\%`
&	`\&`
~	`\textasciitilde`
_	`\_`
^	`\textasciicircum`
\	`\textbackslash`
{	`\{`
}	`\}`

Някои от тези специални знаци, като `\textbackslash`, ще ви принудят да добавите интервал след тях. LaTeX 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 Wrap 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
```

29.2.6. Кавички

От версия 0.1 в LaTeX кода на `debian-book` вече не се използва шрифта `teams`, а `times` от пакета `scalable-cyrfonts-tex`. Когато пишете на LaTeX имайте предвид следното:

За кавички винаги използвайте `"`` този начин `"'`. Така текстът изглежда „по-български“, защото ще бъде по правописните правила на българския език. Българските кавички ще изглеждат „така“. За сега поне българските кавички излизат правилно само в PDF изхода.

Не използвайте за кавички ``` `` ``` по този начин `''`, както е описано в книгите за LaTeX. Английските кавички ще изглеждат “така”

Няма проблем да използвате в LaTeX единични кавички ’така’ и двойни "така". Шрифта който използваме в LaTeX поддържа тези символи така, че те ще се изобразяват точно така в HTML и PDF изходите, но няма да е по правилата на българския език, т.е. отварящите кавички на са като деветки долу вляво, а затварящите като шестици горе в дясно.

29.2.7. 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 LaTeX 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 случай се налага да представите последователност от две тирета. Това се налага при вмъкване на опции на команди. В такива случай използвайте `\verb`:

```
Опциите \verb#--help# и \verb#--version# трябва да се обработват от
всяка GNU програма.
```

29.3. Структура и съдържание - организация на файловете

`./` главна директория. Всичко, на което не може да се намери място, е тук.

`src/` LaTeX сорсовете на самата книга и всички изходни файлове, като `src/debian-book.tex` е главният LaTeX 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 LaTeX, 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 30

Работа със SVN

FIXME: да се завърши

30.1. Достъп до изходните кодове чрез SVN

Естествено, трябва да имате инсталиран SVN. Ако разполагате с Debian:

```
# apt-get install subversion
```

За да изтеглите последните сорсове на книгата от CVS хранилището, изпълнете:

```
$ svn checkout http://svn.openfmi.net/debian-book-bg
```

Тези, които имат желание да се включат в разработката на книгата и да commit-ват в SVN-хранилището, трябва да имат и права за писане в проекта. Авторите трябва да изтеглят книгата по следния начин:

```
$ svn checkout --username име https://svn.openfmi.net/debian-book-bg
```

Където *име* е Вашето потребителско име.

30.2. Бързи инструкции за SVN

Работата със SVN е подробно описана в документацията на http://svnbook.red-bean.com[1]. Ако не искате да се задълбочавате чак толкова, тук са представени основните понятия и действия при работата със SVN. Имайте предвид, че командите много наподобяват на тези на CVS, така, че ще ви е лесно ако сте имали опит с тази система.

За да направите каквато и да е промяна, трябва да имате *работно копие* на книгата. Преди всяка промяна синхронизирате работното копие със SVN-хранилището:

```
$ svn up
```

Извършвате промяната и записвате промените и в SVN хранилището:

```
$ svn commit -m 'описание на промените' file1 path/to/file2
```

LaTeX файловете се намират в директорията `src`. Ако имената на файловете не са достатъчни, за да ви орентират кой какво съдържа, погледнете във файла `debian-book.tex`.

След като свършите с промените, трябва да се обнови и файла `ChangeLog`. Използвайте скрипта `changelog.up`, намиращ се в главната директория debian-book. За да го използвате обаче, трябва да имате инсталирани пакетите `cvs2cl` и `txt2html`.

30.2.1. 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`:

```
$ svn add newfile.c
$ svn ci -m "added newfile.c" newfile.c
```

[1] http://svnbook.red-bean.com

30.2.2. Добавяне на директории

За разлика от добавянето на файл, добавянето на нова директория се извършва посредством една стъпка; не е нужно да изпълнявате commit след това:

```
$ mkdir subdir
$ svn add subdir
```

30.2.3. Премахване на файлове

Премахването на файл е подобна на добавянето:

```
$ rm newfile.c
$ svn remove newfile.c
$ svn ci newfile.c
% $ -- затваряне на долара във verbatim
Забележете, че във втората и третата команда изрично именуваме \texttt{newfile.c},
въпреки че не съществува вече в работното копие. Разбира се, при commit
не е задължително да именувате файла, стига да нямате нищо против commit
да включи всички други промени, които са се състояли в работното копие.
\subsection{Премахване на директории}
SVN за разлика от CVS поддържа контрол на версиите (version control) на директории.
Директориите в SVN хренилището могат да бъдат трити, преименувани, копирани дори и
рекурсивно:
\begin{verbatim}
$ cd dir
$ rm file1 file2 file3
$ svn remove file1 file2 file3
(output omitted)
$ svn ci -m "removed all files" file1 file2 file3
(output omitted)
```

30.2.4. Преименуване на файлове и директории

Преименуването на файл е еквивалентно на създаването му под ново име и премахването му под старото. Под Unix командите са:

```
$ cp oldname newname
$ rm oldname
```

Ето еквивалента при 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)
$
```

Ето еквивалента при SVN:

```
$ svn move file1 file2
```

SVN разполага с команди за copy, delete, move, за повече информация изпълнете:

```
$ svn help move
$ svn help copy
$ svn help delete
```

Относно файловете - това е всичко. Преименуването на директории е аналогично.

30.2.5. SVN и бинарните файлове

SVN управлява binary files доста по-интелигентно отколкото CVS прави това. Поради това, че CVS използва RCS, той може да съхранява успешно пълни копия на променения файл. Но вътрешно SVN прави разлика между файловете използвайки binary-differencing алгоритъм, независимо дали те съдържат текстови (textual) или двоични (binary) данни. Това означава, че всички файлове се съхраняват диференцирано (компресирани) в хранилището и малки разлики винаги се изпращат по мрежата (от клиента към сървъра).

При CVS двоични (binary) файлове трябва да се маркират с флаг `-kb` за да се предотврати модификацията върху тях. Обаче това понякога доста лесно за забравя.

SVN притежава по-параноичен подход:

Първо, никога не изпълнява какъвто и да е вид keyword или line-ending translation докато изрично не му се укаже.

По подразбиране, SVN третира всички файлове като literal byte strings и файловете винаги се съхраняват в хранилището като untranslated state.

Второ, SVN поддържа вътрешно информацията дали файла е текстови или бинарен (text или binary data), но тази информация е налична само в работното копие. При изпълнение на `svn update`, SVN ще направи contextual merges върху модифицираните локално текстови файлове, но няма да се опита да направи това за двоичните файлове.

За да определи дали contextual merge е възможен, SVN изследва `svn:mime-type` информацията. Ако файла няма такава `svn:mime-type` или има `mime-type` който е текстови (например text/*), то се приема, че това е текст. В противен случай, се приема, че файла е двоичен (binary).

SVN също така използва binary-detection алгоритъм при `svn import` и `svn add` командите. Тези команди ще направят опит за правилно опознаване и евентуално ще поставят `svn:mime-type` за двоичен файл при добавянето на му. Ако SVN направи грешно разпознаване, то потребителя винаги може да премахне или постави сам тази информация.

Преди да комитвате в SVN хранилището, проверявайте дали кодът се компилира при вас, винаги тествайте с PDF.

Глава 31

Работа със CVS

Премина се към работа със SVN, но тези инструкции за CVS ще останат тук поради исторически причини.

31.1. 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@photo-forum.net:/home/cvsroot
$ 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 и права за писане в проекта, които могат да получат, ако изпратят писмо до `debian-book@lists.zadnik.org`. Авторите трябва да изтеглят книгата по следния начин:

```
$ export CVSROOT=:ext:име@photo-forum.net:/home/cvsroot
$ export CVS_RSH=ssh
$ cvs -z9 checkout -P debian-book
```

Където *име* е Вашето потребителско име.

31.2. Бързи инструкции за CVS

Работата със CVS е подробно описана в документацията на SourceForge. Добри източници за CVS са: http://cvshome.org[1] и http://www.loria.fr/~molli/cvs-index.html[2]. Ако не искате да се задълбочавате чак толкова, тук са представени основните понятия и действия при работата със CVS.

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 потребителското си име):

```
export CVSROOT=:ext:име@photo-forum.net:/home/cvsroot
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 -z9 checkout -P debian-book
```

Тази команда създава начално работно копие. За обновяване на съдържанието на работното копие използвайте командата

[1] http://cvshome.org

[2] http://www.loria.fr/~molli/cvs-index.html

```
$ cvs -z9 update -PdR
```

Тази команда задължително трябва да се изпълнява в директория на работното копие.

LaTeX 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 да се обнови и файла `ChangeLog`. Използвайте скрипта `changelog.up`, намиращ се в главната директория debian-book. За да го използвате обаче, трябва да имате инсталирани пакетите `cvs2cl` и `txt2html`.

31.2.1. 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)
```

31.2.2. 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
```

31.2.3. Добавяне на директории

За разлика от добавянето на файл, добавянето на нова директория се извършва посредством една стъпка; не е нужно да изпълнявате commit след това:

```
$ mkdir c-subdir
$ cvs add c-subdir
Directory /usr/local/cvs/myproj/c-subdir added to the repository
```

Ако погледнете в новата директория на Вашето работно копие ще видите, че чрез add автоматично бива създадена CVS поддиректория:

```
$ ls c-subdir
CVS/
$ ls c-subdir/CVS
Entries     Repository  Root
```

Сега в нея можете да добавяте файлове (или нови директории), както и при която и да е друга директория в работното копие.

31.2.4. Премахване на файлове

Премахването на файл е подобна на добавянето, освен че има една допълнителна стъпка: Първо трябва да премахнете файла от работното копие:

```
$ 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
```

Забележете, че във втората и третата команда изрично именуваме `newfile.c`, въпреки че не съществува вече в работното копие. Разбира се, при commit не е задължително да именувате файла, стига да нямате нищо против commit да включи всички други промени, които са се състояли в работното копие.

31.2.5. Премахване на директории

Както вече беше споменато, CVS всъщност не дръжи под контрол версиите (version control) на директории. Вместо това, като един вид евтин заместител, предлага определени странни функции, които в повечето случаи вършат работа. Една от тези функции е, че празните директории могат да бъдат третирани по по–особен начин. Ако искате да премахнете директория от даден проект, първо трябва да премахнете всички файлове, които съдържа

```
$ cd dir
$ rm file1 file2 file3
$ cvs remove file1 file2 file3
(output omitted)
$ cvs ci -m "removed all files" file1 file2 file3
(output omitted)
```

и след това да стартирате update в директорията над нея с опцията -P:

```
$ cd ..
$ cvs update -P
(output omitted)
```

Опцията -P указва на update да "съкрати"(prune) всички празни директории — тоест да ги премахне от работното копие. Щом това е направено, може да се каже, че директорията е премахната; всички файлове са премахнати, както и самата директория (поне от работното копие, въпреки че все още има празна директория в хранилището). Интересно съответствие (counterpart) на тази функция е, че когато пуснете само update, CVS не сваля автоматично нови директории от хранилището към Вашето работно копие. Съществуват няколко оправдавания за това поведение, но нито едно не заслужава да бъде обсъдено тук. Краткият отговор е, че от време на време трябва да пускате update с опцията -d, която указва да бъдат свалени всички нови директории от хранилището.

31.2.6. Преименуване на файлове и директории

Преименуването на файл е еквивалентно на създаването му под ново име и премахването му под старото. Под Unix командите са:

```
$ cp oldname newname
$ rm oldname
```

Ето еквивалента при 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)
$
```

Относно файловете — това е всичко. Преименуването на директории не е много по–различно: създайте новата директория, добавете я чрез cvs, преместете всички файлове от старата директория в новата, чрез cvs remove ги премахнете от старата директория, чрез cvs add ги добавете в новата, след това cvs commit, за да има всичко ефект, и накрая направете cvs update -P, за да накарате сега празната директория да изчезне от работното копие. Това е все едно да:

```
$ 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
```

Забележка: съобщението след третата команда. То Ви казва, че не може да копира поддиректорията CVS/ на olddir в newdir, защото newdir вече съдържа директория с такова име. Това е хубаво, защото и без това искате olddir да запази своята CVS/ поддиректория. Очевидно, преместеането на директории може да бъде не особено леко. Най-добрият подход е да опитате да измислите добра подредба на проекта Ви още в началото на разработването му, така че да не Ви се налага често да премествате директории. По-късно ще научите по-драстичен начин за преместване на директории, който включва правенето на промяната направо в хранилището. Най-добре е обаче този начин да бъде запазен за спешни случаи; когато е възможно, най-добре е да управлявате всичко с CVS операции вътре в работните копия.

31.2.7. CVS и бинарните файлове

Досега оставих неизказана малката мръсна тайна: CVS не може да се оправя особено добре с бинарни файлове. Не че CVS изобщо не може да се справя с бинарни файлове; справя се, но не с особена самоувереност. Всички файлове, с които работихме до сега бяха такива съдържащи обикновен текст. CVS има някои специални трика/похвата за текстови файлове. Например, когато работи между Unix хранилище и Windows или Macintosh работно копие, то конвертира завършека на реда както е подходящо за всяка платформа. Например, Unix конвенцията е да се използва само linefeed (LF), докато Windows очаква carriage return/linefeed (CRLF) предица в края на всеки ред. По този начин работното копие на машина използваща Windows ще има CRLF завършеци, а работното копие на същия проект върху Unix машина ще има LF завършеци (самото хранилище винаги се записва във формат LF). Друг трик е, че CVS засича специални стрингове (string - поредица от символи), познати още като RCS ключови стрингове, в текстови файлове и ги замества с revision information и други полезни неща. Например, ако Вашият файл съдържа стринга

```
$Revision: 1.4 $
```

CVS will expand on each commit to include the revision number. For example, it may get expanded to

```
$Revision: 1.4 $
```

CVS ще държи този стринг актуален, докато файлът бива разработван. (Различните ключови стрингове са документирани в Advanced CVS и Third-Party Tools.) This string expansion е много полезна функция при текстови файлове, като Ви позволява да видите the revision number или друга информация относно файла, докато го редактирате. Но какво ще стане, ако файлът е JPG изображение? Или компилирана изпълнима програма? При тези видове файлове CVS може доста да навреди, ако се blundere around expanding който и да е ключов стринг, който срещне. При бинарни файлове, по случайност могат да се появят подобни стрингове.

По тази причина, когато добавяте бинарен файл, трябва да кажете на CVS да изключи както keyword expansion, така и преустройванто на завършека на реда (line-ending conversion). За да направите това, използвайте -kb:

```
$ cvs add -kb filename
$ cvs ci -m "added blah" filename
(etc)
```

Също така, в някои случаи (като текстови файлове, които е вероятно да съдържат фалшифицирани ключови стрингове), може да поискате да изключите само the keyword expansion. Това се прави с -ko:

```
$ cvs add -ko filename
$ cvs ci -m "added blah" filename
(etc)
```

Обърнете внимание, че не можете пълноценно да пуснете cvs diff на два revisions of a binary file. Diff използва текстово-базиран алгоритъм, който може само да съобщи дали двата бинарни файла се различават, но не и как точно се различават. Бъдещи версии на CVS може да предлагат начин за изпълнение на diff върху бинарни файлове.

31.2.8. Работа зад защитна стена (firewall)

31.2.9. Заключение

В крайна сметка правете само неща, които наистина разбирате ;-)

Преди да комитвате в CVS хранилището, проверявайте дали кодът се компилира при вас, винаги тествайте с PDF.

31.3. Използване на общодостъпен ключ за достъп до CVS

По подразбиране всяка команда **cvs**(1) изисква парола за достъп. Това е досадно, но решение има – използването на общодостъпен ключ (*public key*) вместо парола.

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 истинската Ви парола.
2. `scp .ssh/id_dsa.pub photo-forum.net:.`
3. `photo-forum.net$ cat id_dsa.pub » .ssh/authorized_keys`

С това всичко е подготвено за почти безпаролна работа със CVS хранилището.

За да използвате ключа, веднъж в цялата X (или конзолна) сесия трябва да изпълните командата **ssh-add**(1), която пита за паролата, която дадохте на **ssh-keygen**(1). 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**(1).

Глава 32

Как да генерираме PDF, DVI, Postscript, HTML

За да получите **PDF-изхода**, в директорията `src/` изпълнете:

```
$ make pdf
```

DVI и Postscript засега не са предвидени в Makefile-а, но можете да ги получите чрез командите:

```
$ latex debian-book.tex
$ dvips debian-book.tex
```

Внимание: поради това, че обемът на книгата доста нарасна, може би ще се наложи да разрешите на програмите от пакета teTex да ползват по-голямо количествхо 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[1]

Ако разполагате с Debian:

```
# apt-get install latex2html
```

За да получите **HTML изходите**, изпълнете:

```
$ cd debian-book/src
$ make html
```

или

```
$ make htmlsplit
```

Забележка: Когато пишете на LaTeX и не сте убедени, че сте се справили много добре, е хубаво да коментирате първия ред в `debian-book.tex`, т.е. да изключите batch режима. Така ще можете да виждате грешки си.

Ако искате да компилирате всичко:

```
$ cd src/
$ make clean dists-clean
$ make dists
$ make all
```

[1] http://saftsack.fs.uni-bayreuth.de/~latex2ht/current/latex2html-2K.1beta.tar.gz

Част VII

Мотивация и благодарности

Тази книга е резултат на наблюдения на това как с течение на времето 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, 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. От друга страна пък е удоволствие, когато след подобно представяне на дистрибуцията получаваме като обратна връзка благодарности от потребители, неподозиращи силата на Debian. 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 33

GNU Free Documentation License

Version 1.2, November 2002

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. Secondarily, 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.

33.1 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.

33.2 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 33.3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

33.3 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.

33.4 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 33.2 and 33.3 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.

33.5 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 33.4 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".

33.6 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.

33.7 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 33.3 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.

33.8 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 33.4. 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 33.4) to Preserve its Title (section 33.1) will typically require changing the actual title.

33.9 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.

33.10 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.