

Debian Live Manual

Debian Live Project [\[debian-live@lists.debian.org\]](mailto:debian-live@lists.debian.org)

2015-08-23

Debian Live Manual

Debian Live Project debian-live@lists.debian.org

Copyright © 2006-2015 Live Systems Project, Copyright © 2016-2025 The Debian Live team

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

The complete text of the GNU General Public License can be found in `/usr/share/common-licenses/GPL-3` file.

Contents	Utente	10
Debian Live Manual	i Installazione	11
A proposito	3	3. Installazione 11
A proposito di questo manuale	4	3.1 Requisiti 11
1. A proposito di questo manuale	4	3.2 Installare live-build 11
1.1 Per gli impazienti	4	3.2.1 Dal repository Debian 11
1.2 Glossario	4	3.2.2 Da sorgenti 11
1.3 Autori	5	3.3 Installare live-boot e live-config 12
1.4 Contribuire a questo documento	6	3.3.1 Dal repository Debian 12
1.4.1 Applying changes	6	3.3.2 Da sorgenti 12
1.4.2 Traduzione	6	Nozioni di base 14
About the Debian Live Project	8	4. Nozioni di base 14
2. About the Debian Live Project	8	4.1 Cos' un sistema live? 14
2.1 Motivazioni	8	4.2 Scaricare immagini precompilate 15
2.1.1 Cosa c' di sbagliato con gli attuali sistemi live	8	4.3 Primi passi: creare un'immagine ISO ibrida 15
2.1.2 Perch creare il proprio sistema live?	8	4.4 Utilizzare un'immagine ISO live ibrida 15
2.2 Filosofia	8	4.4.1 Masterizzare un'immagine ISO su un supporto fisico 15
2.2.1 Only unchanged packages from Debian main and non-free-firmware	8	4.4.2 Copiare un'immagine ISO ibrida su una penna USB 16
2.2.2 Nessun pacchetto di configurazione per il sistema live	9	4.4.3 Usare lo spazio rimanente su una penna USB 16
2.3 Contatti	9	4.4.4 Avviare il supporto live 16
		4.5 Utilizzare una macchina virtuale per le prove 17
		4.5.1 Provare un'immagine ISO con QEMU 17
		4.5.2 Testing an ISO image with VirtualBox 17
		4.6 Creare e utilizzare un'immagine HDD 18

4.7 Creare un'immagine netboot	19	Personalizzazione dei contenuti	27
4.7.1 Server DHCP	19	7. Panoramica sulla personalizzazione	27
4.7.2 Server TFTP	20	7.1 Configurazione in fase di compilazione e di avvio	27
4.7.3 Server NFS	20	7.2 Fasi della creazione	27
4.7.4 Come provare una netboot	20	7.3 Integrare la configurazione di lb con dei file	27
4.7.5 Qemu	20	7.4 Personalizzazione dei compiti	28
4.8 Webbooting	21	Personalizzare l'installazione dei pacchetti	29
4.8.1 Getting the webboot files	21	8. Personalizzare l'installazione dei pacchetti	29
4.8.2 Booting webboot images	21	8.1 Sorgenti dei pacchetti	29
Panoramica degli strumenti	23	8.1.1 Distribuzione, le aree di archivio e le modalit�	29
5. Panoramica degli strumenti	23	8.1.2 Mirror delle distribuzioni	30
5.1 Il pacchetto live-build	23	8.1.3 Mirror delle distribuzioni usati in fase di com- pilazione	30
5.1.1 Il comando lb config	23	8.1.4 Mirror delle distribuzioni usate durante l'esecuzione	30
5.1.2 Il comando lb build	24	8.1.5 Repository aggiuntivi	30
5.1.3 Il comando lb clean	24	8.2 Scegliere i pacchetti da installare	31
5.2 Il pacchetto live-boot	24	8.2.1 Elenchi di pacchetti	31
5.3 Il pacchetto live-config	24	8.2.2 Usare metapacchetti	31
Gestire una configurazione	25	8.2.3 Elenchi locali dei pacchetti	32
6. Gestire una configurazione	25	8.2.4 Elenchi locali di pacchetti binari	32
6.1 Gestire i cambiamenti di configurazione	25	8.2.5 Elenchi di pacchetti generati	32
6.1.1 Perch� utilizzare gli script automatici? Cosa fanno?	25	8.2.6 Usare condizioni all'interno degli elenchi di pac- chetti	32
6.1.2 Esempi d'uso di script automatici	25	8.2.7 Removing packages at install time	33
6.2 Clonare una configurazione pubblicata tramite Git.	26	8.2.8 Summary	33
		8.2.9 Task per desktop e lingua	33
		8.2.10 Tipi e versioni del kernel	34
		8.2.11 Kernel personalizzati	34

8.3 Installare pacchetti modificati o di terze parti	35	10.3 Persistenza	44
8.3.1 Utilizzare packages.chroot per installare pacchetti personalizzati	35	10.3.1 Il file persistence.conf	46
8.3.2 Utilizzare un repository APT per installare pacchetti personalizzati	36	10.3.2 Utilizzare pi di un'archiviazione persistente	46
8.3.3 Pacchetti personalizzati e APT	36	10.3.3 Using persistence with encryption	47
8.4 Configurare APT in fase di compilazione	36	Personalizzare l'immagine binaria	49
8.4.1 Scegliere apt o aptitude	37	11. Personalizzare l'immagine binaria	49
8.4.2 Utilizzare un proxy con APT	37	11.1 Bootloader	49
8.4.3 Modificare APT per risparmiare spazio	37	11.2 Metadati ISO	49
8.4.4 Passare opzioni ad apt o aptitude	38		
8.4.5 APT pinning	38		
Personalizzazione dei contenuti	40	Personalizzare l'Installatore Debian	50
9. Personalizzazione dei contenuti	40	12. Personalizzare l'Installatore Debian	50
9.1 Include	40	12.1 Tipologie dell'Installatore Debian	50
9.1.1 Live/chroot include locali	40	12.2 Personalizzare il Debian Installer con la preconfigurazione	51
9.1.2 Include locali binari	40	12.3 Personalizzare il contenuto dell'Installatore Debian	51
9.2 Hook	41	Progetto	52
9.2.1 Chroot local hooks	41		
9.2.2 Hook binari locali	41		
9.2.3 Hook in fase di avvio	41		
9.3 Preconfigurare le domande di Debconf	41		
Personalizzare i comportamenti durante l'esecuzione	43	Contribuire al progetto	53
10. Personalizzare i comportamenti durante l'esecuzione	43	13. Contribuire al progetto	53
10.1 Personalizzare l'utente live	43	13.1 Translation of man pages	53
10.2 Personalizzare la localizzazione e la lingua	43		

Segnalare bug	54	Esempi	61
14. Segnalare bug	54	16. Esempi	61
14.1 Problemi noti	54	16.1 Usare gli esempi	61
14.2 Fare la ricerca	54	16.2 Tutorial 1: un'immagine predefinita	61
14.3 Ricompilare da zero	54	16.3 Tutorial 2: servizio browser web	61
14.4 Usare pacchetti aggiornati	55	16.4 Tutorial 3: un'immagine personalizzata	62
14.5 Raccogliere informazioni	55	16.4.1 Prima revisione	62
14.6 Se possibile isolare il caso non andato a buon fine	55	16.4.2 Seconda revisione	63
14.7 Segnalare il bug del pacchetto giusto	56	16.5 Un client Kiosk VNC	64
14.7.1 Durante la compilazione mentre esegue il boot-	56	16.6 A minimal image for a 512MB USB key	64
strap	56	16.7 Un desktop GNOME localizzato e l'installatore	65
14.7.2 Durante la compilazione mentre installa i pac-	56	Appendice	67
chetti	56	Style guide	68
14.7.3 In fase di avvio	56	17. Style guide	68
14.7.4 In fase di esecuzione	56	17.1 Guidelines for authors	68
14.8 Dove segnalare i bug	56	17.1.1 Linguistic features	68
Lo stile nello scrivere codice	57	17.1.2 Procedures	69
15. Lo stile nello scrivere codice	57	17.2 Guidelines for translators	71
15.1 Compatibilit	57	17.2.1 Translation hints	71
15.2 Rientri	57	SiSU Metadata, document information	73
15.3 Ritorno a capo	57		
15.4 Variabili	58		
15.5 Varie	59		
Esempi	60		

₁ Debian Live Manual

₂ A proposito

A proposito di questo manuale

1. A proposito di questo manuale

This manual serves as a single access point to all documentation related to the Debian Live Project and in particular applies to the software produced by the project for the Debian bookworm release. An up-to-date version can always be found at <https://live-team.pages.debian.net/live-manual/>

While live-manual is primarily focused on helping you build a live system and not on end-user topics, an end user may find some useful information in these sections: **The Basics** covers downloading pre-built images and preparing images to be booted from media or the network, either using the web builder or running live-build directly on your system. **Customizing run time behaviours** describes some options that may be specified at the boot prompt, such as selecting a keyboard layout and locale, and using persistence.

Alcuni dei comandi menzionati nel testo devono essere eseguiti con i privilegi di super-utente che possono essere ottenuti diventando utente root tramite su oppure usando sudo. Per distinguere i comandi che possono essere eseguiti come utente normale da quelli che necessitano dei privilegi di super-utente, i comandi sono preceduti rispettivamente da \$ o #. Questi simboli non fanno parte del comando.

1.1 Per gli impazienti

Sebbene crediamo che ogni cosa in questo manuale sia importante almeno per alcuni dei nostri utenti, ci rendiamo conto che c'è tanto materiale da trattare e che si potrebbe voler provare il software

prima di entrare nei dettagli; pertanto suggeriamo di leggerlo nel seguente ordine.

First, read this chapter, **About this manual**, from the beginning and ending with the **Terms** section. Next, skip to the three tutorials at the front of the **Examples** section designed to teach you image building and customization basics. Read **Using the examples** first, followed by **Tutorial 1: A default image**, **Tutorial 2: A web browser utility** and finally **Tutorial 3: A personalized image**. By the end of these tutorials, you will have a taste of what can be done with live systems.

We encourage you to return to more in-depth study of the manual, perhaps next reading **The basics**, skimming or skipping **Building a netboot image**, and finishing by reading the **Customization overview** and the chapters that follow it. By this point, we hope you are thoroughly excited by what can be done with live systems and motivated to read the rest of the manual, cover-to-cover.

1.2 Glossario

Live system : An operating system that can boot without installation to a hard drive. Live systems do not alter local operating system(s) or file(s) already installed on the computer hard drive unless instructed to do so. Live systems are typically booted from media such as CDs, DVDs or USB sticks. Some may also boot over the network (via netboot images, see **Building a netboot image**), and over the Internet (via the boot parameter fetch=URL, see **Webbooting**).

Supporto Live : diversamente dal sistema live, si riferisce a CD, DVD o penna USB dove viene scritto il binario prodotto da live-build e usato per l'avvio del sistema live. Più in generale, il termine si riferisce anche a qualsiasi posto in cui risiede il binario allo scopo di avviare il sistema, come il percorso per i file di avvio da rete.

Debian Live Project : The project which maintains, among others, the live-boot, live-build, live-config, live-tools and live-manual packages.

Sistema host : l'ambiente utilizzato per creare il sistema live.

Sistema di destinazione : l'ambiente usato per eseguire il sistema live.

live-boot : una raccolta di script usati per avviare sistemi live.

live-build : A collection of scripts used to build customized live systems.

live-config : una raccolta di script usati per configurare un sistema live durante il processo di avvio.

live-tools : una raccolta di script aggiuntivi usati per eseguire utili compiti in un sistema live avviato.

live-manual : questo documento inserito nel pacchetto chiamato live-manual.

Debian Installer (d-i) : il sistema d'installazione ufficiale per la distribuzione Debian.

Boot parameters : parametri che possono essere immessi nel prompt del boot loader per modificare il comportamento del kernel o di live-config.

chroot : il programma chroot, chroot(8), rende possibile eseguire diverse istanze dell'ambiente GNU/Linux su un singolo sistema simultaneamente senza riavviare.

Binary image : A file containing the live system, such as live-image-amd64.hybrid.iso or live-image-amd64.img.

Distribuzione di destinazione : la distribuzione su cui sar basato il sistema live. Pu differire dalla distribuzione presente sul proprio computer.

stable/testing/unstable : The stable distribution, currently codenamed bookworm , contains the latest officially released distribution of Debian. The testing distribution, temporarily codenamed trixie , is the staging area for the next stable release. A major advantage of using this distribution is that it has more recent versions of software relative to the stable release. The unstable distribution, permanently codenamed sid , is where active development of Debian occurs. Generally, this distribution is run by developers and those who like to live on the edge. Throughout the manual, we tend to use codenames for the releases, such as trixie or sid , as that is what is supported by the tools themselves.

1.3 Autori

Lista degli autori (in ordine alfabetico):

Ben Armstrong

Brendan Sleight

Carlos Zuferri

Chris Lamb

Daniel Baumann

Franklin Piat

Jonas Stein

Kai Hendry

Marco Amadori

Mathieu Geli

Matthias Kirschner

Richard Nelson

Roland Clobus

Trent W. Buck

1.4 Contribuire a questo documento

Questo manuale pensato come un progetto comunitario e ogni suggerimento e contributo benvenuto. Si veda [Contribuire al progetto](#) per informazioni dettagliate su come prelevare la chiave SSH ed eseguire buoni commit.

1.4.1 Applying changes

Per apportare modifiche alla versione inglese del manuale necessario modificare i file giusti in `manual/en/`, ma prima di sottoporre il proprio contributo si prega di visionare l'anteprima del proprio lavoro. Per ottenere l'anteprima di `live-manual`, assicurarsi di avere installato i pacchetti necessari per la sua compilazione eseguendo:

```
# apt-get install make po4a ruby ruby-nokogiri sisu-complete
```

Si pu compilare il `live-manual` dalla directory superiore del checkout di Git eseguendo:

```
$ make build
```

Since it takes a while to build the manual in all supported languages, authors may find it convenient to use one of the fast proofing shortcuts when reviewing the new documentation they have added to the English manual. Using `PROOF=1` builds `live-manual` in html format, but without the segmented html files, and using `PROOF=2` builds `live-manual` in pdf format, but only the A4 and letter portraits. That is why using either of the `PROOF=` possibilities can save up a considerable amount of time, e.g:

```
$ make build PROOF=1
```

When proofing one of the translations it is possible to build only one language by executing, e.g:

```
$ make build LANGUAGES=de
```

inoltre possibile compilare in base al tipo di documento, esempio:

```
$ make build FORMATS=pdf
```

O entrambi:

```
$ make build LANGUAGES=de FORMATS=html
```

Dopo aver revisionato il proprio lavoro e assicurato che tutto funzioni, non usare `make commit` a meno che nel commit non si stiano aggiornando delle traduzioni, in tal caso non mescolare nello stesso le modifiche al manuale inglese e le traduzioni ma eseguire un commit per ognuna. Per maggiori dettagli vedere la sezione [Traduzione](#).

1.4.2 Traduzione

Note: For the translation of the man pages see [Translation of man pages](#)

In order to translate `live-manual`, follow these steps depending on whether you are starting a translation from scratch or continue working on an already existing one:

Start a new translation from scratch

Translate the `about`manual.ssi.pot` , `about`project.ssi.pot` and `index.html.in.pot` files in `manual/pot/` to your language with your favourite editor (such as poedit) and send the translated .po files to the mailing list to check their integrity. live-manual's integrity check not only ensures that the .po files are 100% translated but it also detects possible errors.

Once checked, to enable a new language in the autobuild it is enough to add the initial translated files to `manual/-po/$-LANGUAGE"/` and edit `manual/`sisu/home/index.html` adding the name of the language and its name in English between brackets. And then, add the folder `manual/-po/$-LANGUAGE"/` to the file `.gitignore`. Finally, run `make commit`.

Continue with an already started translation

If your target language has already been added, you can randomly continue translating the remaining .po files in `manual/po/$-LANGUAGE"/` using your favourite editor (such as poedit) .

Do not forget that you need to run `make commit` to ensure that the translated manuals are updated from the .po files and then you can review your changes launching `make build` before `git add .`, `git commit -m Translating...` and `git push`. Remember that since `make build` can take a considerable amount of time, you can proofread languages individually as explained in [Applying changes](#)

Dopo aver eseguito `make commit` si vedrà del testo scorrere. Questi sono messaggi informativi sullo stato del processo e alcuni suggerimenti su cosa si può fare per migliorare live-manual. A meno che non si ottenga un errore fatale si può procedere e inviare il proprio contributo.

live-manual comes with two utilities that can greatly help translators to find untranslated and changed strings. The first one is `make translate`. It launches a script that tells you in detail how many untranslated strings there are in each .po file. The second one, the `make fixfuzzy` target, only acts upon changed strings but it helps you to find and fix them one by one.

da considerare che nonostante queste utilità possono davvero risultare utili per tradurre dalla riga di comando, si raccomanda l'uso di uno strumento specifico come poedit. inoltre una buona idea leggere la documentazione sulla localizzazione in Debian (110n) e, specifiche per live-manual, le [Linee guida per i traduttori](#).

Nota: si può usare `make clean` per pulire l'albero del repository git locale prima del push. Grazie al file `.gitignore` questo passo non è obbligatorio ma una buona abitudine che evita di fare involontariamente il commit di certi file.

About the Debian Live Project

2. About the Debian Live Project

2.1 Motivazioni

2.1.1 Cosa c'è di sbagliato con gli attuali sistemi live

When Debian Live Project was initiated (around 2006), there were already several Debian based live systems available and they are doing a great job. From the Debian perspective most of them have one or more of the following disadvantages:

Non sono progetti Debian, per cui non sono supportati da Debian.

Mischiano differenti distribuzioni come ad esempio: testing e unstable .

Supportano solamente i386.

Modificano l'aspetto e il comportamento dei pacchetti snellendoli per risparmiare spazio.

Includono pacchetti esterni all'archivio Debian.

Forniscono un kernel con patch addizionali che non appartengono a Debian.

Sono grandi e lenti a causa delle loro dimensioni e non adatti per operazioni di salvataggio.

Non sono disponibili in diversi formati come CD, DVD, penne USB e immagini netboot.

2.1.2 Perché creare il proprio sistema live?

Debian il Sistema Operativo Universale, ha un sistema live per

mostrare e rappresentare accuratamente il sistema con i seguenti vantaggi:

un sottoprogetto di Debian.

Riflette lo stato (attuale) di una distribuzione.

Gira su più architetture possibili.

costituito solo da pacchetti Debian non modificati.

Non contiene nessun pacchetto che non sia presente nell'archivio di Debian.

Usa un kernel Debian inalterato senza patch addizionali.

2.2 Filosofia

2.2.1 Only unchanged packages from Debian main and non-free-firmware

Verranno usati solo pacchetti dal repository Debian della sezione main. La sezione non-free non fa parte di Debian perciò non possono essere utilizzati per le immagini ufficiali del sistema live.

Starting with Debian 12 bookworm we added the [non-free-firmware](#) section for better support of modern hardware.

Non verrà cambiato nessun pacchetto. Nel caso in cui sarà necessario cambiare qualcosa sarà fatto in coordinazione con il maintainer del pacchetto Debian.

In via eccezionale i nostri pacchetti come live-boot, live-build o live-config possono temporaneamente essere usati dal nostro repository per ragioni di sviluppo (ad esempio per creare istantanee). Verranno caricati regolarmente in Debian.

2.2.2 Nessun pacchetto di configurazione per il sistema live 101

In questa fase non saranno disponibili n esempi di installazione n 102
configurazioni alternative. Tutti i pacchetti vengono usati con la
loro configurazione predefinita cos come accade con una regolare
installazione di Debian.

103 Nel caso in cui serva una configurazione predefinita differente,
sar fatto in coordinazione con il maintainer del pacchetto in De-
bian.

104 A system for configuring packages is provided using debconf allow-
ing custom configured packages to be installed in your custom pro-
duced live system images, but for the **prebuilt live images** we choose
to leave packages in their default configuration, unless absolutely
necessary in order to work in the live environment. Wherever possi-
ble, we prefer to adapt packages within the Debian archive to work
better in a live system versus making changes to the live toolchain
or **prebuilt image configurations**. For more information, please see
Customization overview.

2.3 Contatti 105

106 Mailing list : The primary contact for the project is the mail-
ing list at <https://lists.debian.org/debian-live/>. You can email the list
directly by addressing your mail to debian-live@lists.debian.org. The
list archives are available at <https://lists.debian.org/debian-live/>.

107 IRC : molti utenti e sviluppatori sono presenti sul canale #debian-
live su irc.debian.org (OFTC). Quando si pone una domanda su
IRC, si prega di essere pazienti nell'ottenere una risposta; se non
si riceve risposta scrivere alla mailing list.

108 BTS : il **Segnalare bug**.

Utente

Installazione

3. Installazione

3.1 Requisiti

Building live system images has very few system requirements for the host system:

Accesso come utente root

Una versione aggiornata di live-build

Una shell POSIX-compliant, come bash o dash

debootstrap

Linux 2.6 or newer

A mount point with dev and exec rights.

```
# mount -i your`mount`point -o dev,exec,remount
```

Si noti che usare Debian o una distribuzione derivata Debian non richiede - live-build funzioner sostanzialmente su qualsiasi distribuzione che soddisfi i requisiti di cui sopra.

3.2 Installare live-build

Si pu installare live-build in diversi modi:

Dal repository Debian

Da sorgenti

Da istantanee

Se si sta usando Debian, il metodo raccomandato di installare live-build attraverso il repository Debian.

3.2.1 Dal repository Debian

Installare live-build come qualsiasi altro pacchetto:

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

3.2.2 Da sorgenti

live-build sviluppato usando il sistema di controllo versione Git. Sui sistemi basati su Debian fornito dal pacchetto git. Per scaricare il codice aggiornato, eseguire:

```
$ git clone https://salsa.debian.org/live-team/live-build.git
```

possibile costruirsi ed installarsi il proprio pacchetto Debian eseguendo:

```
$ cd live-build
$ dpkg-buildpackage -b -uc -us
$ cd ..
```

Si installino ora i file .deb appena generati ai quali si interessati, ad esempio:

```
# dpkg -i live-build`4.0-1`all.deb
```

Si pu anche installare live-build direttamente sul proprio sistema eseguendo:

```
# make install
```


e disinstallarlo con:

```
# make uninstall
```

3.3 Installare live-boot e live-config

Note: You do not need to install live-boot or live-config on your system to create customized live systems. However, doing so will do no harm and is useful for reference purposes. If you only want the documentation, you may now install the live-boot-doc and live-config-doc packages separately.

3.3.1 Dal repository Debian

Sia live-boot che live-config sono disponibili dai repository Debian come per l'[installazione di live-build](#).

3.3.2 Da sorgenti

Per utilizzare i sorgenti pi recenti da Git si pu seguire il procedimento seguente. Assicurarsi di conoscere i termini menzionati nel [Glossario](#).

Scaricare i sorgenti di live-boot e live-config

```
$ git clone https://salsa.debian.org/live-team/live-boot.git
$ git clone https://salsa.debian.org/live-team/live-config.git
```

Consultare la pagine man di live-boot e live-config per i dettagli sulla personalizzazione se questa il motivo per compilare questi pacchetti dai sorgenti.

140 Costruire un .deb di live-boot e live-config

You must build either on your target distribution or in a chroot containing your target platform: this means if your target is trixie then you should build against trixie .

Use a personal builder such as pbuilder or sbuilder if you need to build live-boot for a target distribution that differs from your build system. For example, for trixie live images, build live-boot in a trixie chroot. If your target distribution happens to match your build system distribution, you may build directly on the build system using dpkg-buildpackage (provided by the dpkg-dev package):

```
$ cd live-boot
$ dpkg-buildpackage -b -uc -us
$ cd ../live-config
$ dpkg-buildpackage -b -uc -us
```

Usare il .deb di live-boot generato

As live-boot and live-config are installed by live-build system, installing the packages in the host system is not sufficient: you should treat the generated .deb files like any other custom packages. Since your purpose for building from source is likely to test new things over the short term before the official release, follow [Installing modified or third-party packages](#) to temporarily include the relevant files in your configuration. In particular, notice that both packages are divided into a generic part, a documentation part and one or more back-ends. Include the generic part, only one back-end matching your configuration, and optionally the documentation. Assuming you are building a live image in the current directory and have generated all .deb files for a single version of both packages in the directory above, these bash commands would copy all of the relevant packages including default back-ends:

```
$ cp ../live-boot-*, -initramfs-tools, -doc "*.deb config/↵  
    packages.chroot/  
$ cp ../live-config-*, -sysvinit, -doc "*.deb config/packages.↵  
    chroot/
```

Nozioni di base

4. Nozioni di base

This chapter contains a brief overview of the build process and instructions for using the three most commonly used image types. The most versatile image type, iso-hybrid, may be used on a virtual machine, optical medium or USB portable storage device. In certain special cases, as explained later, the hdd type may be more suitable. The chapter includes detailed instructions for building and using a netboot type image, which is a bit more involved due to the setup required on the server. This is an slightly advanced topic for anyone who is not already familiar with netbooting, but it is included here because once the setup is done, it is a very convenient way to test and deploy images for booting on the local network without the hassle of dealing with image media.

The section finishes with a quick introduction to **webbooting** which is, perhaps, the easiest way of using different images for different purposes, switching from one to the other as needed using the internet as a means.

In tutto il capitolo faremo spesso riferimento ai nomi dei file predefiniti creati da live-build. Se invece si **scarica un'immagine pre-compilata**, i nomi possono variare.

4.1 Cos' un sistema live?

Per sistema live generalmente si intende un sistema operativo che pu essere avviato da un supporto rimovibile, come un CD-ROM o una chiavetta USB oppure da una rete, pronto per l'uso senza alcuna installazione su hard disk con una configurazione automatica fatta durante l'esecuzione (vedere **Glossario**).

With live systems, it's an operating system, built for one of the supported architectures (currently amd64 and arm64). It is made from the following parts:

Immagine del kernel Linux , comunemente chiamata vmlinuz*

Initial RAM disk image (initrd) : un disco RAM creato per il boot di Linux, contenente i moduli potenzialmente necessari per montare l'immagine di sistema e alcuni script per farlo.

System image : The operating system's filesystem image. Usually, a SquashFS compressed filesystem is used to minimize the live system image size. Note that it is read-only. So, during boot the live system will use a RAM disk and 'union' mechanism to enable writing files within the running system. However, all modifications will be lost upon shutdown unless optional persistence is used (see **Persistence**).

Bootloader : una piccola porzione di codice predisposto per l'avvio dal supporto scelto, che presenta un prompt o un menu per la selezione di opzioni/configurazioni. Carica il kernel Linux ed il suo initrd da eseguire con un filesystem associato. Possono essere usate diverse soluzioni, in base al supporto di destinazione ed al formato del filesystem contenenti le componenti precedentemente citate: isolinux per il boot da CD o DVD nel formato ISO9660, syslinux per supporti HDD o USB che si avviano da una partizione VFAT, extlinux per le partizioni ext/2/3/4 e btrfs, pxelinux per il netboot PXE, GRUB per partizioni ext2/3/4, ecc.

You can use live-build to build the system image from your specifications, set up a Linux kernel, its initrd, and a bootloader to run them, all in one medium-dependent format (ISO9660 image, disk image, etc.).

4.2 Scaricare immagini precompilate

You can download one of the prebuilt images from <https://www.debian.org/CD/live/>. For many of the popular desktop environments (GNOME, Xfce, KDE, etc.) a specific live image is prepared.

If you are unsure which file to download, use the ‘Live GNOME’ image from the ‘stable’ release. You can then skip reading the next sections and run the image in a **virtual machine**.

No parameters are passed to these commands, so defaults for all of their various options will be used. See **The lb config command** for more details.

Ora che si ha una gerarchia config/ si pu generare l’immagine con il comando lb build:

```
# lb build
```

4.3 Primi passi: creare un’immagine ISO ibrida

Indipendentemente dal tipo di immagine, per crearne una necessario eseguire ogni volta la stessa procedura. Come primo esempio si creer una directory di lavoro, vi si entrerà e si eseguirà la seguente sequenza di comandi di live-build per creare un’immagine ISO ibrida di base contenente un sistema live predefinito senza X.org. adatta per essere masterizzata su CD o DVD e anche per essere copiata su una penna USB.

Il nome della directory di lavoro arbitrario ma guardando gli esempi usati da live-manual una buona idea utilizzare un nome che aiuta a identificare in qualsiasi directory l’immagine su cui si sta lavorando, in particolar modo se si stanno sperimentando tipi di immagine differenti. In questo caso stiamo creando un sistema predefinito quindi chiamiamolo ad esempio live-default.

```
$ mkdir live - default && cd live - default
```

Then, run the lb config command. This will create a config/ hierarchy in the current directory for use by other commands:

```
$ lb config
```

This process can take a while, depending on the speed of your computer and your network connection. When it is complete, there should be a live-image-amd64.hybrid.iso image file, ready to use, in the current directory.

Note: If you are building on an amd64 system the name of the resulting image will be live-image-amd64.hybrid.iso. Keep in mind this naming convention throughout the manual.

4.4 Utilizzare un’immagine ISO live ibrida

After either building or downloading an ISO hybrid image the usual next step is to prepare your medium for booting, either CD-R(W) or DVD-R(W) optical media or a USB stick.

4.4.1 Masterizzare un’immagine ISO su un supporto fisico

Masterizzare un’immagine ISO semplice, basta installare xorriso e utilizzarlo da riga di comando; ad esempio:

```
# apt-get install xorriso
$ xorriso -as cdrecord -v dev=/dev/sr0 blank=as-needed live -<
image-amd64.hybrid.iso
```

4.4.2 Copiare un'immagine ISO ibrida su una penna USB

ISO images prepared with xorriso, can be simply copied to a USB stick with the cp program or an equivalent. Plug in a USB stick with a size large enough for your image file and determine which device it is, which we hereafter refer to as `$-USBSTICK`". This is the device file of your key, such as `/dev/sdb`, not a partition, such as `/dev/sdb1`! You can find the right device name by looking in dmesg's output after plugging in the stick, or better yet, `ls -l /dev/-disk/by-id`.

Once you are certain you have the correct device name, use the cp command to copy the image to the stick. This will definitely overwrite any previous contents on your stick!

```
$ cp live-image-amd64.hybrid.iso $-USBSTICK"
$ sync
```

Note: The sync command is useful to ensure that all the data, which is stored in memory by the kernel while copying the image, is written to the USB stick.

4.4.3 Usare lo spazio rimanente su una penna USB

After copying the live-image-amd64.hybrid.iso to a USB stick, the first partition on the device will be filled up by the live system. To use the remaining free space, use a partitioning tool such as gparted or parted to create a new partition on the stick.

```
# gparted $-USBSTICK"
```

Dopo aver creato la partizione, dove `$-PARTITION`" il nome della

partizione, ad esempio `/dev/sdb2`, si deve creare su di essa un filesystem. Una scelta possibile potrebbe essere ext4.

```
# mkfs.ext4 $-PARTITION"
```

Nota: se si desidera utilizzare lo spazio extra con Windows pare che questo sistema operativo non possa accedere a nessuna partizione eccetto la prima. Alcune soluzioni a questo problema sono state discusse sulla nostra [mailing list](#), ma non sembrano esserci risposte semplici.

Remember: Every time you install a new live-image-amd64.hybrid.iso on the stick, all data on the stick will be lost because the partition table is overwritten by the contents of the image, so back up your extra partition first to restore again after updating the live image.

4.4.4 Avviare il supporto live

La prima volta che si avvia il supporto live, CD, DVD, penna USB o PXE, pu essere necessario impostare il BIOS del computer, ma giacch questi variano parecchio in opzioni e scorciatoie, non siamo in grado di descriverli. Alcuni BIOS offrono un menu per selezionare il device in fase di boot, in caso sia disponibile nel vostro sistema il modo pi semplice. Altrimenti necessario accedere alla sua configurazione e modificare l'ordine di avvio per posizionare la periferica di boot del sistema live prima di quella usuale.

Avviando il supporto si otterr un menu, premendo il tasto enter il sistema partir utilizzando la voce Live e le opzioni predefinite. Per ulteriori informazioni sulle opzioni di boot, si veda la voce help nel menu e le pagine di manuale di live-boot e live-config all'interno del sistema.

Assuming you've selected Live and booted a default desktop live image, after the boot messages scroll by, you should be automatically logged into the user account and see a desktop, ready to use. If you have booted a console-only image, you should be automatically logged in on the console to the user account and see a shell prompt, ready to use.

4.5 Utilizzare una macchina virtuale per le prove

Per lo sviluppo delle immagini live, pu essere un notevole risparmio di tempo eseguirle in una macchina virtuale (VM). Non senza qualche raccomandazione:

Eseguire una VM richiede un quantitativo sufficiente di RAM sia per il sistema ospitato che per quello ospitante; consigliato un processore che gestisca la virtualizzazione a livello hardware.

Ci sono alcune limitazioni inerenti, quali uno scarso rendimento video e una scelta limitata di hardware emulato.

Quando si sviluppa per un hardware specifico non vi alcun sostituto migliore del proprio hardware.

Occasionalmente possono esserci dei bug relativi al solo utilizzo di una VM. Nel dubbio si provi l'immagine direttamente sul proprio hardware.

A condizione che si possa lavorare entro questi vincoli, cercare il software disponibile per la virtualizzazione e scegliere quello adatto alle proprie necessit.

4.5.1 Provare un'immagine ISO con QEMU

Il programma pi versatile in Debian QEMU. Se il processore gestisce la virtualizzazione hardware utilizzare il pacchetto qemu-kvm; la descrizione elenca brevemente i requisiti.

Per prima cosa installare qemu-kvm o altrimenti qemu, nel qual caso il nome del programma nei successivi sar qemu invece di kvm. Il pacchetto qemu-utils inoltre utile per creare immagini di dischi virtuali con qemu-img.

```
# apt-get install qemu-kvm qemu-utils
```

Avviare un'immagine ISO semplice:

```
$ kvm -cdrom live-image-amd64.hybrid.iso -m 4G
```

Per maggiori dettagli si vedano le pagine di manuale.

Note: For live systems containing a desktop environment that you want to test with qemu, you may wish to include the spice-vdagent package in your live-build configuration. This will automatically adjust the resolution and enable the clipboard between the virtual machine and the host.

```
$ echo "spice-vdagent" >> config/package-lists/spice.list &↵
$ chroot
```

4.5.2 Testing an ISO image with VirtualBox

Per provare la ISO con virtualbox:

```
# apt-get install virtualbox virtualbox-qt virtualbox-dkms
$ virtualbox
```

Create a new virtual machine, change the storage settings to use

live-image-amd64.hybrid.iso as the CD/DVD device, and start the machine.

Nota: per sistemi live contenenti X.org che si vogliono provare con virtualbox, si pu voler includere il pacchetto dei driver per X.org di VirtualBox, virtualbox-guest-dkms e virtualbox-guest-x11, nella configurazione di live-build. In caso contrario la risoluzione limitata a 800x600.

```
$ echo "virtualbox-guest-dkms virtualbox-guest-x11" && config↵
/package-lists/my.list.chroot
```

Per far funzionare il pacchetto dkms vanno anche installati gli header per il kernel utilizzato nell'immagine. Anzich indicare manualmente il pacchetto linux-headers adeguato nell'elenco dei pacchetti creato prima, la selezione pu essere fatta automaticamente da live-build.

```
$ lb config --linux-packages "linux-image linux-headers"
```

4.6 Creare e utilizzare un'immagine HDD

Building an HDD image is similar to an ISO hybrid one in all respects except you specify -b hdd and the resulting filename is live-image-amd64.img which cannot be burnt to optical media. It is suitable for booting from USB sticks, USB hard drives, and various other portable storage devices. Normally, an ISO hybrid image can be used for this purpose instead, but if you have a BIOS which does not handle hybrid images properly, you need an HDD image.

Nota: se si crea un'immagine ISO ibrida con gli esempi precedenti, occorre pulire la directory di lavoro con il comando lb clean (vedere [Il comando lb clean](#)):

```
# lb clean --binary
```

Eeguire il comando lb config come prima, questa volta specificando per il tipo di immagine HDD:

```
$ lb config -b hdd
```

Creare ora l'immagine con il comando lb build:

```
# lb build
```

When the build finishes, a live-image-amd64.img file should be present in the current directory.

The generated binary image contains a VFAT partition and the syslinux bootloader, ready to be directly written on a USB device. Once again, using an HDD image is just like using an ISO hybrid one on USB. Follow the instructions in [Using an ISO hybrid live image](#), except use the filename live-image-amd64.img instead of live-image-amd64.hybrid.iso.

Likewise, to test an HDD image with Qemu, install qemu as described above in [Testing an ISO image with QEMU](#). Then run kvm or qemu, depending on which version your host system needs, specifying live-image-amd64.img as the first hard drive.

```
$ kvm -hda live-image-amd64.img
```

4.7 Creare un'immagine netboot

La seguente sequenza di comandi creerà un'immagine netboot di base contenente un sistema live predefinito senza X.org. adatta per il boot tramite rete.

Nota: se qualcuno tra gli esempi precedenti è stato seguito, bisogna pulire la directory di lavoro con il comando `lb clean`:

```
# lb clean
```

In this specific case, a `lb clean --binary` would not be enough to clean up the necessary stages. The cause for this is that in netboot setups, a different `initramfs` configuration needs to be used which `live-build` performs automatically when building netboot images. Since the `initramfs` creation belongs to the `chroot` stage, switching to netboot in an existing build directory means to rebuild the `chroot` stage too. Therefore, `lb clean` (which will remove the `chroot` stage, too) needs to be used.

Per configurare l'immagine per l'avvio da rete, eseguire il comando `lb config` come segue:

```
$ lb config -b netboot --net-root-path "/srv/debian-live" --net-root-server "192.168.0.2"
```

Diversamente dalle immagini ISO e HDD, il boot via rete non fornisce un'immagine del filesystem al client, perciò i file devono essere forniti via NFS. Con `lb config` si possono scegliere filesystem di rete differenti. Le opzioni `--net-root-path` e `--net-root-server` specificano, rispettivamente, il percorso e il server del server NFS dove l'immagine del filesystem sarà situata all'avvio. Accertarsi che questi siano impostati su valori adeguati alla propria rete.

Creare ora l'immagine con il comando `lb build`:

```
# lb build
```

In un avvio tramite rete, il client esegue una piccola parte di software che normalmente risiede sulla EPROM della scheda Ethernet. Questo programma invia una richiesta DHCP per ottenere un indirizzo IP e le informazioni su cosa fare in seguito. In genere il passo successivo è ottenere un bootloader di livello superiore attraverso il protocollo TFTP. Questi potrebbero essere `pxelinux`, `GRUB`, o anche avviare direttamente un sistema operativo come Linux.

For example, if you unpack the generated live-image-`amd64.netboot.tar` archive in the `/srv/debian-live` directory, you'll find the filesystem image in `live/filesystem.squashfs` and the kernel, `initrd` and `pxelinux` bootloader in `tftpboot/`.

We must now configure three services on the server to enable netbooting: the DHCP server, the TFTP server and the NFS server.

4.7.1 Server DHCP

Si deve configurare il server DHCP della rete per essere sicuri di fornire un indirizzo IP al sistema client che si avvia tramite rete, e notificare la posizione del bootloader PXE.

Ecco un esempio, scritto per un server DHCP ISC `isc-dhcp-server` nel file di configurazione `/etc/dhcp/dhcpd.conf`:

```
# /etc/dhcp/dhcpd.conf - configuration file for isc-dhcp-server

ddns-update-style none;
```



```

option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

log-facility local7;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.1 192.168.0.254;
    filename "pxelinux.0";
    next-server 192.168.0.2;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
}

```

4.7.2 Server TFTP

Fornisce al sistema il kernel e il ramdisk iniziale in fase di esecuzione.

Si installi il pacchetto `tftpd-hpa`, che mette a disposizione tutti i file contenuti in una directory `root`, di solito `/srv/tftp`. Affinché si possa disporre dei file contenuti in `/srv/debian-live/tftpboot`, eseguire il seguente comando come utente `root`:

```
# dpkg-reconfigure -plow tftpd-hpa
```

e inserire la nuova directory del server tftp quando richiesto.

4.7.3 Server NFS

Una volta che il computer ospite ha scaricato e avviato un kernel Linux e caricato il suo `initrd`, cercherà di montare l'immagine del filesystem Live tramite un server NFS.

Bisogna installare il pacchetto `nfs-kernel-server`.

Quindi, rendere disponibile l'immagine del filesystem via NFS aggiungendo una riga come la seguente in `/etc/exports`:

```
/srv/debian-live *(ro,async,no'root'squash,no'subtree'check)
```

e comunicare il nuovo export al server NFS con il seguente comando:

```
# exportfs -rv
```

Setting up these three services can be a little tricky. You might need some patience to get all of them working together. For more information, see the `syslinux` wiki at <https://wiki.syslinux.org/wiki/index.php?title=PXELINUX> or the Debian Installer Manual's TFTP Net Booting section at <https://www.debian.org/releases/stable/amd64/ch04s05.en.html>. They might help, as their processes are very similar.

4.7.4 Come provare una netboot

La creazione di immagini netboot resa semplice da `live-build`, ma provare le immagini su una macchina reale può essere davvero dispendioso in termini di tempo.

Per semplificarsi la vita si può usare la virtualizzazione.

4.7.5 Qemu

Installare `qemu`, `bridge-utils`, `sudo`.

Modificare `/etc/qemu-ifup`:

```
#!/bin/sh
sudo -p "Password for $0:" /sbin/ifconfig $1 172.20.0.1
echo "Executing /etc/qemu-ifup"
echo "Bringing up $1 for bridged mode..."
sudo /sbin/ifconfig $1 0.0.0.0 promisc up
echo "Adding $1 to br0..."
sudo /usr/sbin/brctl addif br0 $1
sleep 2
```

Procurarsi o compilare grub-floppy-netboot.

```
Lanciare      qemu      con      -net      nic,vlan=0      -net
tap,vlan=0,ifname=tun0
```

4.8 Webbooting

Webbooting is a convenient way of retrieving and booting live systems using the internet as a means. The requirements for webbooting are very few. On the one hand, you need a medium with a boot-loader, an initial ramdisk and a kernel. On the other hand, a web server to store the squashfs files which contain the filesystem.

4.8.1 Getting the webboot files

As usual, you can build the images yourself or use the **prebuilt files**. Using prebuilt images would be handy for doing initial testing until one can fine tune their own needs. If you have built a live image you will find the files needed for webbooting in the build directory under binary/live/. The files are called vmlinuz, initrd.img and filesystem.squashfs.

It is also possible to extract those files from an already existing iso image. In order to achieve that, loopback mount the image as follows:

```
# mount -o loop image.iso /mnt
```

The files are to be found under the live/ directory. In this specific case, it would be /mnt/live/. This method has the disadvantage that you need to be root to be able to mount the image. However, it has the advantage that it is easily scriptable and thus, easily automated.

But undoubtedly, the easiest way of extracting the files from an iso image and uploading it to the web server at the same time, is using the midnight commander or mc. If you have the genisoimage package installed, the two-pane file manager allows you to browse the contents of an iso file in one pane and upload the files via ftp in the other pane. Even though this method requires manual work, it does not require root privileges.

4.8.2 Booting webboot images

While some users will prefer virtualization to test webbooting, we refer to real hardware here to match the following possible use case which should only be considered as an example.

In order to boot a webboot image it is enough to have the components mentioned above, i.e. vmlinuz and initrd.img in a usb stick inside a directory named live/ and install syslinux as bootloader. Then boot from the usb stick and type fetch=URL/PATH/TO/-FILE at the boot options. live-boot will retrieve the squashfs file and store it into ram. This way, it is possible to use the downloaded compressed filesystem as a regular live system. For example:

```
append boot=live components fetch=http://192.168.2.50/images/↵
webboot/filesystem.squashfs
```

Use case: You have a web server in which you have stored two 293 squashfs files, one which contains a full desktop, like for example gnome, and a standard one. If you need a graphical environment for one machine, you can plug your usb stick in and webboot the gnome image. If you need one of the tools included in the second type of image, perhaps for another machine, you can webboot the standard one.

Panoramica degli strumenti

5. Panoramica degli strumenti

This chapter contains an overview of the three main tools used in building live systems: live-build, live-boot and live-config.

5.1 Il pacchetto live-build

live-build is a collection of scripts to build live systems. These scripts are also referred to as commands.

L'idea dietro live-build è di essere un'infrastruttura che utilizza una directory di configurazione per automatizzare totalmente e personalizzare tutti gli aspetti della creazione di un'immagine live.

Molti concetti sono simili a quelli applicati per creare pacchetti Debian con debhelper:

The scripts have a central location for configuring their operation. In debhelper, this is the `debian/` subdirectory of a package tree. For example, `dh_install` will look, among others, for a file called `debian/install` to determine which files should exist in a particular binary package. In much the same way, live-build stores its configuration entirely under a `config/` subdirectory.

Gli script sono indipendenti, vale a dire che sempre sicuro eseguire ogni comando.

Unlike debhelper, live-build provides the tools to generate a skeleton configuration directory. This could be considered to be similar to tools such as `dh-make`. For more information about these tools, read on, since the remainder of this section discusses the four most important commands. Note that the preceding `lb` is a generic wrapper for live-build commands.

`lb config` : Responsible for initializing a Live system configuration directory. See [The `lb config` command](#) for more information.

`lb build` : responsabile di iniziare la creazione di un sistema live. Si veda [Il comando `lb`](#) per maggiori informazioni.

`lb clean` : responsabile della rimozione di parti della creazione di un sistema live. Si veda [Il comando `lb clean`](#) per maggiori informazioni.

5.1.1 Il comando `lb config`

As discussed in [live-build](#), the scripts that make up live-build read their configuration with the source command from a single directory named `config/`. As constructing this directory by hand would be time-consuming and error-prone, the `lb config` command can be used to create the initial skeleton configuration tree.

Issuing `lb config` without any arguments creates the `config/` subdirectory which is populated with some default settings in configuration files, and two skeleton trees named `auto/` and `local/`.

```
$ lb config
[2025-02-15 12:34:56] lb config
P: Using http proxy: http://127.0.0.1:3142
P: Creating config tree for a debian/testing/amd64 system
P: Symlinking hooks...
```

Using `lb config` without any arguments would be suitable for users who need a very basic image, or who intend to provide a more complete configuration via `auto/config` later (see [Managing a configuration](#) for details).

Normally, you will want to specify some options. For example, to specify which package manager to use while building the image:

```
$ lb config --apt aptitude
```

possibile specificare molte opzioni, come:

```
$ lb config --binary-images netboot --bootappend-live "boot=↵
live components hostname=live-host username=live-user" ↵
...
```

Una lista completa delle opzioni disponibile nel manuale di `lb-config`.

5.1.2 Il comando `lb build`

Il comando `lb build` legge la configurazione dalla directory `config/` ed esegue a un livello inferiore i comandi necessari a costruire il sistema live.

5.1.3 Il comando `lb clean`

It is the job of the `lb clean` command to remove various parts of a build so subsequent builds can start from a clean state. By default, `chroot`, `binary` and `source` stages are cleaned, but the cache is left intact. Also, individual stages can be cleaned. For example, if you have made changes that only affect the `binary` stage, use `lb clean --binary` prior to building a new binary. If your changes invalidate the bootstrap and/or package caches, e.g. changes to `--mode`, `--architecture`, or `--bootstrap`, you must use `lb clean --purge`. See the `lb clean` man page for a full list of options.

5.2 Il pacchetto `live-boot`

`live-boot` is a collection of scripts providing hooks for the `initramfs-`

tools, used to generate an `initramfs` capable of booting live systems, such as those created by `live-build`. This includes the live system ISOs, netboot tarballs, and USB stick images.

At boot time it will look for read-only media containing a `/live/` directory where a root filesystem (often a compressed filesystem image like `squashfs`) is stored. If found, it will create a writable environment, using `OverlayFS`, for Debian like systems to boot from.

More information on initial ramfs in Debian can be found in the Debian Linux Kernel Handbook at <https://kernel-team.pages.debian.net/kernel-handbook/> in the chapter on `initramfs`.

5.3 Il pacchetto `live-config`

`live-config` costituito da script eseguiti all'avvio dopo `live-boot` per configurare automaticamente il sistema live. Gestisce attività quali impostare l'hostname, localizzazione e fuso orario, creare l'utente live, inibire compiti automatizzati tramite cron ed eseguire il login automatico dell'utente live.

Gestire una configurazione

6. Gestire una configurazione

Questo capitolo spiega come gestire una configurazione per una live sin dalla creazione iniziale, attraverso le successive revisioni e rilasci sia del software live-build che della stessa immagine.

6.1 Gestire i cambiamenti di configurazione

Le configurazioni live sono di rado perfette al primo tentativo. Pu andar bene passare le opzioni di lb config a riga di comando per eseguire una compilazione ma tipico rivedere queste opzioni e compilare finch non si soddisfatti. Per gestire le modifiche c' bisogno di script automatici che assicurano che la propria configurazione sia coerente.

6.1.1 Perch utilizzare gli script automatici? Cosa fanno?

The lb config command stores the options you pass to it in config/* files along with many other options set to default values. If you run lb config again, it will not reset any option that was defaulted based on your initial options. So, for example, if you run lb config again with a new value for `--binary-images`, any dependent options that were defaulted for the old image type may no longer work with the new ones. Nor are these files intended to be read or edited. They store values for over a hundred options, so nobody, let alone yourself, will be able to see in these which options you actually specified. And finally, if you run lb config, then upgrade live-build and it happens to rename an option, config/* would still contain variables named after the old option that are no longer valid.

Per queste ragioni gli script nella directory auto/* faciliteranno il lavoro; sono semplici wrapper ai comandi lb config, lb build e lb clean designati per aiutare a gestire la configurazione. Gli script in auto/config memorizzano i comandi di lb config con le opzioni desiderate, quelli in auto/clean rimuovono i file contenenti i valori delle variabili di configurazione, mentre gli script in auto/build tengono un build.log di ogni compilazione. Ognuno di questi script viene eseguito automaticamente ogni qualvolta si esegue il comando lb corrispondente; utilizzandoli la vostra configurazione sar pi semplice da leggere e verr mantenuta coerente da una revisione all'altra. Inoltre sar molto pi facile identificare e sistemare le opzioni che necessitano di modifiche quando si aggiorna live-build dopo aver letto la documentazione aggiornata.

6.1.2 Esempi d'uso di script automatici

Per comodit live-build fornito di esempi di script automatici da copiare e modificare. Inizializzare una nuova configurazione predefinita quindi copiare gli esempi in essa:

```
$ mkdir mylive && cd mylive && lb config
$ mkdir auto
$ cp /usr/share/doc/live-build/examples/auto/* auto/
```

Modificare auto/config aggiungendo qualsiasi opzione vi serva, esempio:

```
#!/bin/sh
lb config noauto "
    --distribution stable "
    --binary-images hdd "
    --mirror-bootstrap http://ftp.ch.debian.org/debian/ "
    --mirror-binary http://ftp.ch.debian.org/debian/ "
"$@"
```

Ogni volta che verr usato `lb config`, `auto/config` ripristiner la configurazione in base a queste opzioni; quando si vogliono apportare modifiche baster modificare le opzioni in questo file invece di passarle a `lb config`. Utilizzando `lb clean`, `auto/clean` pulir i file in `config/*` insieme a qualsiasi altro creato dalla compilazione. Infine, quando si usa `lb build`, verr scritto da `auto/build` un file di log della compilazione in `build.log`.

Nota: il parametro speciale `noauto` viene qui usato per impedire un'ulteriore chiamata di `auto/config`, impedendo quindi infinite chiamate ricorsive; assicurarsi di non rimuoverlo facendo modifiche. Quando si dividono comandi lunghi di `lb config` su pi righe per agevolarne la leggibilit, non dimenticare il backslash (alla fine di ogni riga che continua sulla successiva, come mostrato poc'anzi nell'esempio di script.

6.2 Clonare una configurazione pubblicata tramite Git.

Use the `lb config --config` option to clone a Git repository that contains a live system configuration. If you would like to base your configuration on one maintained by the Debian Live Project, look at <https://salsa.debian.org/live-team/> for the repository named `live-images` in the category Subgroups and projects. This repository contains the configurations for the live systems **prebuilt images**.

For example, to build a standard image, use the `live-images` repository as follows:

```
$ mkdir live-images && cd live-images
$ lb config --config https://salsa.debian.org/live-team/live-images.git::debian
$ cd images/standard
```

Modificare `auto/config` e qualsiasi altro file presente in `config` nec-

essario alle proprie esigenze. Ad esempio, le immagini non-free precompilate non ufficiali sono create semplicemente aggiungendo `--archive-areas main contrib non-free`.

possibile definire una scorciatoia nella configurazione di Git aggiungendo quanto segue al file `$HOME"/.gitconfig`:

```
[url "https://salsa.debian.org/live-team/"]
    insteadOf = lso:
```

This enables you to use `lso:` anywhere you need to specify the address of a `salsa.debian.org` git repository. If you also drop the optional `.git` suffix, starting a new image using this configuration is as easy as:

```
$ lb config --config lso:live-images::debian
```

Clonando l'intero repository `live-images` si ottengono configurazioni usate per svariate immagini. Se dopo aver terminato la prima si vuole creare un'immagine differente, baster cambiare `directory` e opzionalmente fare di nuovo le modifiche necessarie alle proprie esigenze.

In ogni caso ricordarsi che ogni volta si dovr creare l'immagine come utente `root`: `lb build`

Personalizzazione dei contenuti

7. Panoramica sulla personalizzazione

This chapter gives an overview of the various ways in which you may customize a live system.

7.1 Configurazione in fase di compilazione e di avvio

La configurazione del sistema live è divisa in opzioni applicate in fase di compilazione e al momento dell'avvio. Le opzioni di compilazione sono ulteriormente divise in quelle che si verificano prima dell'avvio, applicate dal pacchetto live-boot, e quelle dopo l'avvio, applicate da live-config. Qualsiasi opzione in fase di avvio può essere modificata dall'utente specificandola al prompt di avvio. L'immagine può inoltre essere costruita con i parametri di avvio predefiniti in modo che quando tutti i valori predefiniti sono adatti gli utenti possano avviare direttamente il sistema senza specificare alcuna opzione. In particolare, l'argomento di `lb -bootappend-live` costituito da tutte le opzioni da riga di comando del kernel predefinite in un sistema live, come persistenza dei dati, layout di tastiera o fuso orario. Per gli esempi si veda [Personalizzare localizzazione e lingua](#).

Build-time configuration options are described in the `lb config` man page. Boot-time options are described in the man pages for `live-boot` and `live-config`. Although the `live-boot` and `live-config` packages are installed within the live system you are building, it is recommended that you also install them on your build system for easy reference when you are working on your configuration. It is safe to do so, as none of the scripts contained within them are executed unless the system is configured as a live system.

7.2 Fasi della creazione

Il processo di creazione è diviso in due fasi, con varie personalizzazioni applicate in sequenza a ciascuna di esse. La prima consiste nell'avvio, questa è la fase iniziale di popolamento della directory di `chroot` con i pacchetti atti a creare un sistema Debian di base. Viene quindi seguita dalla fase `chroot` che completa la costruzione della directory `chroot` e la popola con tutti i pacchetti elencati nella configurazione insieme a qualsiasi altro materiale; la maggior parte della personalizzazione dei contenuti avviene in questa tappa. La parte finale della preparazione dell'immagine è la fase binaria che genera un'immagine avviabile utilizzando i contenuti della directory `chroot` per costruire il file system principale per il sistema live, includere l'installatore e ogni altro materiale aggiuntivo sul supporto di destinazione al di fuori del file system del sistema live. Una volta che l'immagine è pronta viene creato, se abilitato, l'archivio dei sorgenti nella fase sorgenti.

All'interno di ciascuna di queste fasi c'è una sequenza particolare in cui vengono applicati i comandi, sono organizzati in modo da assicurare che le personalizzazioni siano ragionevolmente stratificate. Ad esempio, nella fase `chroot` i `preseed` vengono applicati prima che qualsiasi pacchetto sia installato, i pacchetti vengono installati prima che qualsiasi file incluso localmente venga copiato e gli `hook` eseguiti dopo che tutto il materiale è a posto.

7.3 Integrare la configurazione di lb con dei file

Although `lb config` creates a skeletal configuration in the `config/` directory, to accomplish your goals, you may need to provide additional files in subdirectories of `config/`. Depending on where the files are stored in the configuration, they may be copied into the live system's filesystem or into the binary image filesystem, or may provide build-time configurations of the system that would be cum-

bersome to pass as command-line options. You may include things such as custom lists of packages, custom artwork, or hook scripts to run either at build time or at boot time, boosting the already considerable flexibility of debian-live with code of your own.

7.4 Personalizzazione dei compiti

I capitoli seguenti sono costituiti dai tipi di compito personalizzato che gli utenti eseguono solitamente: **personalizzare l'installazione dei pacchetti**, **personalizzare i contenuti** e **personalizzare localizzazione e lingua** coprono solo alcune delle cose che si potrebbero desiderare.

Personalizzare l'installazione dei pacchetti

8. Personalizzare l'installazione dei pacchetti

Perhaps the most basic customization of a live system is the selection of packages to be included in the image. This chapter guides you through the various build-time options to customize live-build's installation of packages. The broadest choices influencing which packages are available to install in the image are the distribution and archive areas. To ensure decent download speeds, you should choose a nearby distribution mirror. You can also add your own repositories for backports, experimental or custom packages, or include packages directly as files. You can define lists of packages, including metapackages which will install many related packages at once, such as packages for a particular desktop or language. Finally, a number of options give some control over apt, or if you prefer, aptitude, at build time when packages are installed. You may find these handy if you use a proxy, want to disable installation of recommended packages to save space, or need to control which versions of packages are installed via APT pinning, to name a few possibilities.

8.1 Sorgenti dei pacchetti

8.1.1 Distribuzione, le aree di archivio e le modalit

The distribution you choose has the broadest impact on which packages are available to include in your live image. Specify the code-name, which defaults to `testing`. Any current distribution carried in the archive may be specified by its codename here. (See [Terms](#)

for more details.) The `--distribution` option not only influences the source of packages within the archive, but also instructs live-build to enable other sources.

For example, to build against the stable release, with security, updates (enabled per default) and additionally proposed-updates and backports, specify:

```
$ lb config --distribution stable --proposed-updates true --backports true
```

Similarly, for the unstable release, `sid`, which has neither security nor updates, specify:

```
$ lb config --distribution sid
```

All'interno dell'archivio dei pacchetti, le aree sono le principali divisioni dello stesso. In Debian queste sono `main`, `contrib` e `non-free`; soltanto `main` contiene il software che parte di Debian, perci questa la predefinita. Possono essere specificati uno o pi valori:

```
$ lb config --archive-areas "main contrib non-free"
```

Attraverso l'opzione `--mode` disponibile un supporto sperimentale per alcune derivate di Debian; per impostazione predefinita, questa opzione impostata su `debian` solo se si sta costruendo su un sistema Debian o sconosciuto. Invocando `lb config` su una delle derivate supportate, verr creata un'immagine di quella derivata in modo predefinito. Se `lb config` viene ad esempio eseguito in modalit `ubuntu`, saranno gestiti i nomi della distribuzione e le aree di archivio per la derivata specificata e non quelli di Debian. La modalit cambia anche il comportamento di live-build per adattarlo alle derivate.

Note: The projects for whom these modes were added are primarily responsible for supporting users of these options. The Debian Live Project, in turn, provides development support on a best-effort basis only, based on feedback from the derivative projects as we do not develop or support these derivatives ourselves.

8.1.2 Mirror delle distribuzioni

L'archivio Debian replicato attraverso una vasta rete di mirror in tutto il mondo cosicch chiunque in ogni nazione pu selezionare il mirror pi vicino per una migliore velocit di scaricamento. Ciascuna delle opzioni `--mirror-*` determina quale mirror della distribuzione usato nei vari stadi della compilazione. Ricordando dalle **Fasi della creazione** che la fase di avvio quando il chroot inizialmente popolato da `debootstrap` con un sistema minimale e quella di `chroot` quando viene creato il chroot usato per costruire il file system del sistema live. Perci per queste fasi vengono usati i corrispondenti cambi di mirror, e in seguito, nella fase binaria vengono usati i valori di `--mirror-binary` e `--mirror-binary-security` sostituendo qualsiasi altro mirror usato nelle fasi iniziali.

8.1.3 Mirror delle distribuzioni usati in fase di compilazione

To set the distribution mirrors used at build time to point at a local mirror, it is sufficient to set `--mirror-bootstrap` and `--mirror-chroot-security` as follows.

```
$ lb config --mirror-bootstrap http://localhost/debian/ "
--mirror-chroot-security http://localhost/debian-↵
security/
```

Il mirror `chroot`, specificato da `--mirror-chroot`, impostato al valore di `--mirror-bootstrap` in modo predefinito.

8.1.4 Mirror delle distribuzioni usate durante l'esecuzione

The `--mirror-binary*` options govern the distribution mirrors placed in the binary image. These may be used to install additional packages while running the live system. The defaults employ `deb.debian.org`, a service that chooses a geographically close mirror based, among other things, on the user's IP family and the availability of the mirrors. This is a suitable choice when you cannot predict which mirror will be best for all of your users. Or you may specify your own values as shown in the example below. An image built from this configuration would only be suitable for users on a network where mirror is reachable.

```
$ lb config --mirror-binary http://mirror/debian/ "
--mirror-binary-security http://mirror/debian-↵
security/ "
--mirror-binary-backports http://mirror/debian-↵
backports/
```

8.1.5 Repository aggiuntivi

Si possono aggiungere altri repository, ampliando cos la scelta dei pacchetti al di l di quelli disponibili nella distribuzione di destinazione. Questi possono essere, per esempio, pacchetti di backport, sperimentali o personalizzati. Per configurare repository aggiuntivi, creare i file `config/archives/vostro-repository.list.chroot`, o `config/archives/vostro-repository.list.binary`. Come per le opzioni `--mirror-*`, queste controlleranno i repository usati nella fase `chroot` quando si compila l'immagine, e nella fase `binary`, ad esempio per usarli quando il sistema live avviato.

Per esempio, `config/archives/live.list.chroot` permette di installare pacchetti dal repository snapshot `debian-live` al momento della creazione del sistema live.

```
deb http://debian-live.alieth.debian.org/sid-snapshots main contrib non-free
```

Se si aggiunge la stessa riga in `config/archives/live.list.binary`, il repository verrà aggiunto alla directory `/etc/apt/sources.list.d/` del sistema live.

Se questi file esistono saranno prelevati automaticamente.

You should also put the ASCII-armored GPG key used to sign the repository into `config/archives/your-repository.key-binary,chroot` files.

Se si necessita di personalizzare il pinning di APT, le sezioni di APT preferences possono essere inserite nei file `config/archives/mio-repository.pref-binary,chroot` e verranno automaticamente aggiunte nella directory `/etc/apt/preferences.d/` del sistema live.

Similarly, if you need custom APT·AUTH.CONF(5) authentication configuration, this can be placed in `config/archives/your-repository-auth-binary,chroot` files and will be automatically added to your live system's `/etc/apt/auth.conf.d/` directory

8.2 Scegliere i pacchetti da installare

Ci sono diversi modi per scegliere quali pacchetti live-build installer nell'immagine, coprendo una gamma di esigenze diverse. Si possono richiamare i singoli pacchetti da un elenco, usare i metapacchetti o selezionarli tramite il file control. E infine inserire i file dei pacchetti nell'albero `config/`, che ben si adatta a provare pacchetti nuovi o sperimentali prima che siano disponibili in un repository.

8.2.1 Elenchi di pacchetti

Gli elenchi di pacchetti sono un potente mezzo per esprimere quali pacchetti devono essere installati. La sintassi gestisce sezioni condizionali rendendo semplice la creazione di elenchi e adattarli per l'uso in molteplici configurazioni. I nomi dei pacchetti possono inoltre essere inseriti nell'elenco utilizzando script shell in fase di compilazione.

Nota: quando si specifica un pacchetto che non esiste, il comportamento di live-build è determinato dalla scelta delle utilità di APT. Per ulteriori dettagli si veda [Scegliere apt o aptitude](#).

8.2.2 Usare metapacchetti

Il metodo più semplice per popolare una lista di pacchetti utilizzare un metapacchetto task mantenuto dalla distribuzione. Ad esempio:

```
$ lb config
$ echo task-gnome-desktop & config/package-lists/desktop.list &
  .chroot
```

This supersedes the older predefined list method supported in live-build 2.x. Unlike predefined lists, task metapackages are not specific to the Live System project. Instead, they are maintained by specialist working groups within the distribution and therefore reflect the consensus of each group about which packages best serve the needs of the intended users. They also cover a much broader range of use cases than the predefined lists they replace.

Tutti i metapacchetti task iniziano per task-, un modo per determinare quali siano disponibili (sebbene possa contenere alcuni falsi positivi che corrispondono al nome ma non sono metapacchetti) di controllare il nome del pacchetto con:

```
$ apt-cache search --names-only ^task -
```

In aggiunta a questi si trovano altri metapacchetti per vari scopi. Alcuni sono dei sottoinsiemi dei pacchetti task generici, come `gnome-core`, mentre altri sono parti individuali di un Debian Pure Blend, come il metapacchetto `education-*`. Per elencarli tutti installare il pacchetto `debtags` e usare il tag `role::metapackage` come segue:

```
$ debtags search role::metapackage
```

8.2.3 Elenchi locali dei pacchetti

Se si richiede l'elenco di metapacchetti, pacchetti individuali o una combinazione di entrambi tutte le liste dei pacchetti locali vengono salvate in `config/package-lists/`. Giacch possibile usare pi di una lista, ci si presta bene a progetti modulari. Si pu ad esempio decidere di dedicare un elenco ad un particolare desktop, un altro ad un insieme di pacchetti correlati utilizzabili con desktop differenti. Questo permette di sperimentare diverse combinazioni di insiemi di pacchetti con il minimo sforzo condividendo gli elenchi tra progetti live differenti.

Per essere processati, gli elenchi dei pacchetti che si trovano in questa directory devono avere un suffisso `.list` e un suffisso `.chroot` o `.binary` aggiuntivo per indicare per quale fase sia l'elenco.

The packages in the `.list.chroot` install list are present both in the live system and in the installed system.

Nota: se non si specifica il suffisso l'elenco sar usato per entrambe le fasi. Normalmente preferibile specificare `.list.chroot` in modo che i pacchetti vengono installati solo nel filesystem live evitando di avere una copia extra del `.deb` sul dispositivo.

8.2.4 Elenchi locali di pacchetti binari

Per creare un elenco di binari inserire un file con suffisso `.list.binary` in `config/package-lists/`; questi pacchetti non sono installati nel filesystem ma inclusi sul dispositivo live sotto `pool/`. Solitamente questo elenco si usa con una delle varianti non-live dell'installatore; come detto sopra, se si vuole che questo sia identico all'elenco della fase `chroot`, usare semplicemente il suffisso `.list`.

8.2.5 Elenchi di pacchetti generati

Talvolta succede che il modo migliore per ottenere un elenco di generarlo con uno script. Ogni riga che inizia con un punto esclamativo indica un comando da eseguire nel `chroot` quando viene creata l'immagine. Ad esempio si potrebbe includere la riga `! grep-aptavail -n -sPackage -FPriority standard —sort` in una lista di pacchetti per produrne una contenente i pacchetti con `Priority: standard` disponibili.

Infatti selezionare i pacchetti con il comando `grep-aptavail` (presente nel pacchetto `dctrl-tools`) talmente utile che `live-build` fornisce uno script `Packages` per comodit; accetta due argomenti: `field` e `pattern`. Per cui si pu creare un elenco con il seguente contenuto:

```
$ lb config
$ echo '! Packages Priority standard' & config/package-lists/↵
  standard.list.chroot
```

8.2.6 Usare condizioni all'interno degli elenchi di pacchetti

Ognuna delle variabili di configurazione di `live-build` situate in `config/*` (senza il prefisso `LB`) possono essere utilizzate per istruzioni

condizionali nell’elenco dei pacchetti. In genere questo significa qualsiasi opzione di lb config in maiuscolo e con trattini cambiati in trattini bassi; ma in pratica la sola ad influenzare la selezione dei pacchetti che abbia senso, come DISTRIBUTION, ARCHITECTURES o ARCHIVE’AREAS.

Per esempio, per installare ia32-libs se specificata `–architectures amd64`:

```
#if ARCHITECTURES amd64
ia32-libs
#endif
```

Si pu provare per ognuna di una serie di valori, ad esempio per installare memtest86+ specificando sia `–architectures i386` sia `–architectures amd64`:

```
#if ARCHITECTURES i386 amd64
memtest86+
#endif
```

possibile provare altre variabili che contengano pi di un valore, ad esempio per installare vrms specificando sia da contrib sia da non-free tramite `–archive-areas`:

```
#if ARCHIVE’AREAS contrib non-free
vrms
#endif
```

Le condizioni nidificate non sono supportate.

8.2.7 Removing packages at install time

You can list packages in files with `.list.chroot`live` and `.list.chroot`in-`

stall suffixes inside the `config/package-lists` directory. If both a live and an install list exist, the packages in the `.list.chroot`live` list are removed with a hook after the installation (if the user uses the installer). The packages in the `.list.chroot`install` list are present both in the live system and in the installed system. This is a special tweak for the installer and may be useful if you have `–debian-installer live` set in your config, and wish to remove live system-specific packages at install time.

8.2.8 Summary

The table below shows which configuration files are required to achieve the desired availability of the package.

	X.chroot	X.chroot`- X live	X	X.binary
Package is installed in the live system	Yes	Yes	Yes	No
Package is removed after installing the live system	No	Yes	No	N/A
Package can be installed from the live system without network	N/A	N/A	Yes *1	Yes

*1: Because the installer needs this package

X = `config/package-lists/custom`name.list`

8.2.9 Task per desktop e lingua

I task per i desktop e la lingua sono un caso particolare che ne-

cessita di ulteriori pianificazioni e configurazioni e in questo senso le immagini live sono diverse da quelle dell'Installatore Debian. Nell'Installatore Debian, se il supporto stato preparato per un particolare ambiente desktop, il corrispondente task verr automaticamente installato. Perci ci sono task gnome-desktop, kde-desktop, lxde-desktop e xfce-desktop interni, nessuno dei quali offerto nel menu di tasksel. Allo stesso modo, non c' nessuna voce nel menu per i task delle lingue, ma la scelta della lingua dell'utente durante l'installazione influenza la selezione dei corrispondenti task della lingua.

Sviluppando un'immagine live per desktop, questa si avvia direttamente su un'area di lavoro, le scelte del desktop e della lingua predefinita sono state fatte al momento della compilazione e non al volo come nel caso dell'installatore Debian. Questo non per dire che un'immagine live non possa essere creata con un supporto per desktop o lingue multipli per offrire all'utente una scelta, ma che non il comportamento predefinito nella creazione di una live.

Poich automaticamente non viene fatta alcuna preparazione sui task della lingua, i quali includono cose come caratteri specifici per la lingua e pacchetti per i metodi di input, se li si vogliono, vanno specificati nella configurazione. Per esempio, un'immagine del desktop GNOME contenente il supporto per il tedesco pu includere questi metapacchetti task:

```
$ lb config
$ echo "task-gnome-desktop task-laptop" & & config/package- <-
  lists/my.list.chroot
$ echo "task-german task-german-desktop task-german-gnome- <-
  desktop" & & config/package-lists/my.list.chroot
```

8.2.10 Tipi e versioni del kernel

A seconda dell'architettura, nell'immagine verranno inclusi uno o pi tipi di kernel in modo predefinito. possibile scegliere tipi differenti tramite l'opzione `--linux-flavours`, ognuno ha come suffisso `linux-image` che costituisce il nome del metapacchetto che a sua volta dipende dall'esatto pacchetto del kernel da inserire nell'immagine.

Thus by default, an amd64 architecture image will include the `linux-image-amd64` flavour metapackage, and an i386 architecture image will include the `linux-image-586` metapackage.

When more than one kernel package version is available in your configured archives, you can specify a different kernel package name stub with the `--linux-packages` option. For example, supposing you are building an amd64 architecture image and add the experimental archive for testing purposes so you can install the `linux-image-3.18.0-trunk-amd64` kernel. You would configure that image as follows:

```
$ lb config --linux-packages linux-image-3.18.0-trunk
$ echo "deb http://deb.debian.org/debian/ experimental main" <-
  & & config/archives/experimental.list.chroot
```

8.2.11 Kernel personalizzati

Si pu compilare e includere i propri kernel personalizzati a patto che siano integrati nel sistema di gestione dei pacchetti di Debian. Il sistema live-build non supporta i kernel n crea pacchetti `.deb`.

La maniera corretta e raccomandata per collocare i propri pacchetti di seguire le istruzioni nel `kernel-handbook`. Ricordarsi di modificare i suffissi per ABI e tipologia in modo appropriato quindi

includere una compilazione completa del pacchetto linux e del corrispondente linux-latest nel repository.

Se si opta per creare i pacchetti del kernel senza i metapacchetti corrispondenti, bisogna specificare un suffisso `-linux-packages` appropriato come discusso in **Tipi e versioni del kernel**. Come spiegato in **Installare pacchetti modificati o di terze parti**, meglio includere i propri pacchetti del kernel nel proprio repository, sebbene funzionino anche le alternative discusse in tale sezione.

Fornire suggerimenti sul come personalizzare il proprio kernel va oltre lo scopo di questa documentazione, tuttavia necessario assicurarsi che la configurazione soddisfi almeno i seguenti requisiti minimi:

Utilizzare un ramdisk iniziale;

Include the union filesystem module (i.e. usually OverlayFS).

includere qualsiasi altro modulo del filesystem necessario alla configurazione (solitamente squashfs).

8.3 Installare pacchetti modificati o di terze parti

While it is against the philosophy of a live system, it may sometimes be necessary to build a live system with modified versions of packages that are in the Debian repository. This may be to modify or support additional features, languages and branding, or even to remove elements of existing packages that are undesirable. Similarly, third-party packages may be used to add bespoke and/or proprietary functionality.

This section does not cover advice regarding building or maintaining modified packages. Joachim Breitner's 'How to fork privately' method from <http://www.joachim-breitner.de/blog/archives/>

[282-How-to-fork-privately.html](#) may be of interest, however. The creation of bespoke packages is covered in the Debian New Maintainers' Guide at <https://www.debian.org/doc/manuals/maint-guide/> and elsewhere.

Ci sono due modi per installare pacchetti personalizzati:

`packages.chroot`

utilizzare repository APT personalizzati

Usando `packages.chroot` pi semplice da ottenere e utile per una personalizzazione una tantum ma ha una serie di svantaggi, mentre un repository APT personalizzato pi laborioso da configurare.

8.3.1 Utilizzare `packages.chroot` per installare pacchetti personalizzati

Per installare un pacchetto personalizzato copiarlo nella directory `config/packages.chroot/`; i pacchetti al suo interno verranno installati automaticamente durante la creazione del sistema live, non necessario specificarli altrove.

I pacchetti devono essere nominati nel modo prescritto, un metodo semplice per farlo usare `dpkg-name`.

L'utilizzo di `packages.chroot` per l'installazione di pacchetti personalizzati presenta degli svantaggi:

non possibile usare secure APT,

necessario installare i pacchetti adeguati nella directory `config/packages.chroot/`,

It does not lend itself to storing live system configurations in revision control.

8.3.2 Utilizzare un repository APT per installare pacchetti personalizzati

A differenza di `packages.chroot`, quando si usa un repository APT personalizzato necessario assicurarsi di specificare altrove i pacchetti. Per i dettagli si veda [Scegliere i pacchetti da installare](#).

Sebbene creare un repository APT possa sembrare uno sforzo inutile, l'infrastruttura pu facilmente essere riutilizzata in un secondo momento per offrire aggiornamenti dei pacchetti modificati.

The APT repository does not necessarily need to be online, you can use a local repository instead. However, in both cases the repository needs to be signed.

Example:

```
$ gpg --armor --output config/archives/custom`repo`.gpg.key$-↵
  EXTENSION" --export-options export-minimal --export $-↵
  SIGNING`KEY"
$ cat ;; EOF ; config/archives/custom`repo`.list$-EXTENSION"
deb [signed-by=/etc/apt/trusted.gpg.d/custom`repo`.gpg.key$-↵
  EXTENSION".asc] $-URI" $-SUITE" $-COMPONENTS"
EOF
$ echo "$-PACKAGESFROM`REPOSITORY`" ; config/package-lists/↵
  custom`repo`.list$-EXTENSION"
```

Where:

`$-EXTENSION`": the optional stage suffix, see the [summary](#)

`$-SIGNING`KEY`": the keyID of the signature of the repository

`$-URI`": the URI to the repository, e.g. `http://deb.debian.org/debian/` or `file://$(pwd)/my`local`-repository`

`$-SUITE`": the suite within the repository, e.g. `my-debian-based-distro`

`$-COMPONENTS`": the components within the repository, e.g. `main`

`$-PACKAGESFROM`REPOSITORY``": the names of the packages to install (dependencies will automatically be installed as well)

8.3.3 Pacchetti personalizzati e APT

live-build utilizza APT per installare tutti i pacchetti nel sistema live in modo da ereditare i comportamenti di questo programma. Un esempio rilevante che (considerando una configurazione predefinita) dato un pacchetto disponibile in due repository differenti con numeri di versione diversi, APT sceglie di installare quello con il numero di versione pi alto.

A causa di questo si pu voler incrementare il numero della versione nei file `debian/changelog` dei pacchetti personalizzati per accertare che la propria versione avr la precedenza sui repository Debian ufficiali. anche ottenibile modificando le preferenze del APT pinning del sistema live, si veda [APT pinning](#) per maggiori informazioni.

8.4 Configurare APT in fase di compilazione

APT configurabile tramite una serie di opzioni applicate solo in fase di costruzione (la configurazione di APT utilizzata nel sistema live in esecuzione pu essere configurata nel solito modo, ovvero includendo le impostazioni appropriate attraverso `config/includes.chroot/`). Per un elenco completo, cercare nel manuale di `lb`config` le opzioni che iniziano con `apt`.

8.4.1 Scegliere apt o aptitude

Per installare pacchetti in fase di compilazione si pu optare sia per apt sia per aptitude, l'argomento `--apt` di `lb config` determina quale usare. Sceglie il metodo implementando il comportamento preferito per l'installazione dei pacchetti, la notevole differenza come vengono gestiti quelli mancanti.

`apt`: se viene specificato un pacchetto mancante, l'installazione avr esito negativo; questo l'impostazione predefinita.

`aptitude`: se viene specificato un pacchetto mancante, l'installazione avr successo.

8.4.2 Utilizzare un proxy con APT

One commonly required APT configuration is to deal with building an image behind a proxy. You may specify your APT proxy with the `--apt-http-proxy` option as needed, e.g.

```
$ lb config --apt-http-proxy http://proxy/
```

8.4.3 Modificare APT per risparmiare spazio

Si pu aver bisogno di risparmiare dello spazio sul supporto dell'immagine, in tal caso una o entrambe delle seguenti opzioni possono essere d'interesse.

possibile non includere gli indici di APT con:

```
$ lb config --apt-indices false
```

Questo non influenzer le voci in `/etc/apt/sources.list`, determina

solo se `/#-var/lib/apt/#` contiene o meno i file degli indici. Il compromesso che APT necessita di quegli indici per operar enel sistema live, perci prima di eseguire `apt-cache search` o `apt-get install`, per esempio, l'utente deve usare prima `apt-get update` per crearli.

In caso si trovi che l'installazione dei pacchetti raccomandati appesantisca troppo l'immagine, a patto si preparati ad affrontare le conseguenze discusse prima, si pu disabilitare l'opzione predefinita di APT con:

```
$ lb config --apt-recommends false
```

The most important consequence of turning off recommends is that live-boot and live-config themselves recommend some packages that provide important functionality used by most Live configurations.

Two packages which you most probably will want to add again are:

`user-setup` which live-config recommends is used to create the live user.

`sudo` which live-config recommends is used to obtain root access in the live-image, which is needed to shutdown the computer.

```
$ lb config --apt-recommends false
$ echo "user-setup sudo" & config/package-lists/recommends.<-
list.chroot
```

In all but the most exceptional circumstances you need to add back at least some of these recommends to your package lists or else your image will not work as expected, if at all. Look at the recommended packages for each of the live-* packages included in your build and

if you are not certain you can omit them, add them back into your package lists.

The more general consequence is that if you don't install recommended packages for any given package, that is, packages that would be found together with this one in all but unusual installations (**APT pinning**).

8.4.4 Passare opzioni ad apt o aptitude

Se non esiste un'opzione di `lb config` per modificare il comportamento di APT come si desidera, utilizzare `--apt-options` o `--aptitude-options` per passare qualsiasi argomento tramite lo strumento APT scelto. Per i dettagli consultare le pagine di manuale di apt e aptitude. Notare che entrambe le opzioni hanno valori predefiniti che servir mantenere in aggiunta a qualsiasi altra fornita. Per cui supponendo di aver incluso qualcosa da `snapshot.debian.org` per fare dei test e volendo specificare `Acquire::Check-Valid-Until=false` per soddisfare APT con il vecchio file Release, si proceder come nell'esempio riportato di seguito, appendendo la nuova opzione al valore predefinito `--yes`:

```
$ lb config --apt-options "--yes -oAcquire::Check-Valid-Until=false"
```

Per apprendere a pieno queste opzioni e sapere quando usarle consultare i manuali. Questo solo un esempio e non va interpretato come il modo per configurare la propria immagine, non sarebbe appropriato per il rilascio finale.

Per configurazioni di APT pi complesse che comportano l'uso di opzioni in `apt.conf` si pu voler creare invece il file `config/apt/apt.conf`. Vedere anche le altre opzioni `apt-*` per alcune comode scorciatoie di operazioni di uso frequente.

8.4.5 APT pinning

Si prega di leggere prima il manuale di `apt-preferences(5)`. Il pinning pu essere configurato sia in fase di costruzione sia di esecuzione; per la prima creare `config/archives/*.pref`, `config/archives/*.pref.chroot`, e `config/apt/preferences` mentre per l'ultima creare `config/includes.chroot/etc/apt/preferences`.

Let's say you are building a trixie live system but need all the live packages that end up in the binary image to be installed from `sid` at build time. You need to add `sid` to your APT sources and pin the live packages from it higher, but all other packages from it lower, than the default priority. Thus, only the packages you want are installed from `sid` at build time and all others are taken from the target system distribution, trixie . The following will accomplish this:

```
$ echo "deb http://mirror/debian/ sid main" > config/archives/sid.list.chroot
$ cat >> config/archives/sid.pref.chroot << EOF
Package: live-*
Pin: release n=sid
Pin-Priority: 600

Package: *
Pin: release n=sid
Pin-Priority: 1
EOF
```

Un valore negativo della `priority` evita che un pacchetto venga installato, come nel caso in cui non se ne voglia uno raccomandato da un altro. Supponiamo di costruire un'immagine di LXDE utilizzando l'opzione `task-lxde-desktop` in `config/package-lists/desktop.list.chroot` ma non si desidera che all'utente venga richiesto di salvare la password del wifi nel portachiavi. Questo metapacchetto dipende da `lxde-core` che raccomanda `gksu` e che a sua volta racco-

manda gnome-keyring, in questo caso si vorrà omettere il pacchetto gnome-keyring aggiungendo a `#-config/apt/preferences` la seguente istruzione:

519

```
Package: gnome-keyring
Pin: version *
Pin-Priority: -1
```

Personalizzazione dei contenuti

9. Personalizzazione dei contenuti

This chapter discusses fine-tuning customization of the live system contents beyond merely choosing which packages to include. Includes allow you to add or replace arbitrary files in your live system image, hooks allow you to execute arbitrary commands at different stages of the build and at boot time, and preseeding allows you to configure packages when they are installed by supplying answers to debconf questions.

9.1 Include

While ideally a live system would include files entirely provided by unmodified packages, it is sometimes convenient to provide or modify some content by means of files. Using includes, it is possible to add (or replace) arbitrary files in your live system image. live-build provides two mechanisms for using them:

Include locali del chroot: permettono di aggiungere o sostituire file al file system chroot/Live. Vedere [Live/chroot include locali](#) per maggiori informazioni.

Include locali binari: permettono di aggiungere o sostituire file nell'immagine binaria. Vedere [Include locali binari](#) per maggiori informazioni

Si consulti il [Glossario](#) per ulteriori informazioni sulla distinzione tra immagini Live e binarie.

9.1.1 Live/chroot include locali

Chroot local includes can be used to add or replace files in the chroot/Live filesystem so that they may be used in the Live system. A typical use is to populate the skeleton user directory (/etc/skel) used by the Live system to create the live user's home directory. Another is to supply configuration files that can be simply added or replaced in the image without processing; see [Chroot local hooks](#) if processing is needed.

Per includere i file si aggiungano semplicemente alla directory config/includes.chroot. Questa corrisponde alla directory root / del sistema live. Per esempio, per aggiungere un file /var/www/index.html nel sistema live, si usi:

```
$ mkdir -p config/includes.chroot/var/www
$ cp /path/to/my/index.html config/includes.chroot/var/www
```

La configurazione avrà quindi il seguente schema:

```
-- config
[... ]
    |-- includes.chroot
    |   |-- var
    |       |-- www
    |           |-- index.html
[... ]
```

Gli include locali del chroot vengono installati dopo l'installazione dei pacchetti in modo che tali file vengano in seguito sovrascritti.

9.1.2 Include locali binari

Si possono utilizzare include locali binari per inserire sul filesystem

tem del supporto materiale come documentazione o video affinché sia immediatamente accessibile dopo l'inserimento dello stesso senza avviare il sistema live. Ci funziona in modo simile agli include locali del chroot; supponendo che i file `~/video'demo.*` siano video dimostrativi del sistema descritti da e collegati a una pagina HTML indice, basta copiare il materiale in `config/includes.binary/` come segue:

```
$ cp ~/video'demo.* config/includes.binary/
```

Questi file appariranno nella directory principale del supporto live.

9.2 Hook

Hooks allow commands to be run in the chroot and binary stages of the build in order to customize the image. Depending on whether you are building a live image or a regular system image you have to place your hooks in `config/hooks/live` or `config/hooks/normal` respectively. These are frequently referred to as local hooks because they are executed inside the build environment.

There are also boot-time hooks that allow you to run commands once the image has already been built, during the boot process.

9.2.1 Chroot local hooks

To run commands in the chroot stage, create a hook script with a `.hook.chroot` suffix containing the commands either in the `config/hooks/live` or `config/hooks/normal` directories. The hook will run in the chroot after the rest of your chroot configuration has been

applied, so remember to ensure your configuration includes all packages and files your hook needs in order to run. See the example chroot hook scripts for various common chroot customization tasks provided in `/usr/share/doc/live-build/examples/hooks` which you can copy or symlink to use them in your own configuration.

9.2.2 Hook binari locali

To run commands in the binary stage, create a hook script with a `.hook.binary` suffix containing the commands either in the `config/hooks/live` or `config/hooks/normal` directories. The hook will run after all other binary commands are run, but before binary checksums, the very last binary command. The commands in your hook do not run in the chroot, so take care not to modify any files outside of the build tree, or you may damage your build system! See the example binary hook scripts for various common binary customization tasks provided in `/usr/share/doc/live-build/examples/hooks` which you can copy or symlink to use them in your own configuration.

9.2.3 Hook in fase di avvio

Per eseguire comandi all'avvio, è possibile fornire degli hook a live-config come spiegato nella sezione Customization del suo manuale. Controllare gli hook di live-config in `/lib/live/config/` e notare i numeri sequenziali; fornire quindi i propri hook con una sequenza numerica appropriata, sia come include locali del chroot in `config/includes.chroot/lib/live/config/`, sia come pacchetto personalizzato come discusso in **Installare pacchetti modificati o di terze parti**.

9.3 Preconfigurare le domande di Debconf

I file nella directory `config/preseed/` con suffisso `.cfg` seguiti dalla

fase (.chroot o .binary) sono considerati file di preconfigurazione di debconf e sono installati da live-build usando debconf-set-selections durante la fase corrispondente.

550 Per ulteriori informazioni su debconf, vedere debconf(7) nel pacchetto debconf.

Personalizzare i comportamenti durante l'esecuzione

10. Personalizzare i comportamenti durante l'esecuzione

Tutte le configurazioni durante l'esecuzione sono eseguite da live-config. Vengono qui presentate alcune delle opzioni di live-config pi comuni alle quali gli utenti sono interessati; una lista completa pu essere trovata nel suo manuale.

10.1 Personalizzare l'utente live

Un'importante considerazione che l'utente live viene creato all'avvio da live-boot e non da live-build durante la compilazione. Questo non solo influenza dove viene introdotto il materiale relativo all'utente nella creazione, come discusso in [Live/chroot include locali](#), ma anche ogni gruppo e permesso associato all'utente live.

You can specify additional groups that the live user will belong to by using any of the possibilities to configure live-config. For example, to add the live user to the fuse group, you can either add the following file in config/includes.chroot/etc/live/config.conf.d/10-user-setup.conf:

```
LIVE'USER'DEFAULT'GROUPS="audio cdrom dip floppy video ←  
plugdev netdev powerdev scanner bluetooth fuse"
```

o utilizzare live-config.user-default-groups=audio,cdrom,dip,floppy,video,plugdev,netdev,powerdev,scanner,bluetooth,fuse come parametro di boot.

inoltre possibile modificare facilmente il nome utente user e la password live predefiniti.

Per cambiare il nome utente specificare quanto segue nella configurazione:

```
$ lb config --bootappend-live "boot=live components username=live-user"
```

Un modo per cambiare la password tramite un hook come descritto in [Hook in fase di avvio](#). Si pu usare l'hook passwd da /usr/share/doc/live-config/examples/hooks, antepoendolo di conseguenza (ad esempio, 2000-passwd) e aggiungerlo al file config/includes.chroot/lib/live/config/

10.2 Personalizzare la localizzazione e la lingua

Quando il sistema live si avvia, la lingua inserita in due fasi:

generazione della localizzazione

impostare la configurazione della tastiera

Quando si crea un sistema live la localizzazione predefinita locales=en'US.UTF-8. Per definire quale generare, si usi il parametro locales nell'opzione --bootappend-live di lb config:

```
$ lb config --bootappend-live "boot=live components locales=de'CH.UTF-8"
```

Possono essere specificate pi lingue separate da una virgola.

Questo parametro, cos come quelli della tastiera indicati pi avanti, possono essere usati anche dalla riga di comando del kernel specificando una lingua con language'country (nel qual caso verr usata la codifica

predefinita) o l'intera stringa language`country.encoding. In /usr/share/i18n/SUPPORTED possibile trovare un elenco delle lingue supportate e la codifica per ognuna di esse.

Sia la configurazione della tastiera in console sia di X sono eseguite da live-config con il pacchetto console-setup. Per fare ci usare i parametri keyboard-layouts, keyboard-variants, keyboard-options e keyboard-model tramite l'opzione --bootappend-live. Le opzioni valide si trovano in /usr/share/X11/xkb/rules/base.lst. Per ottenere i layout e le varianti di una data lingua, provare a cercare il loro nome inglese o il paese in cui usata, esempio:

```
$ egrep -i '(!—german.*switzerland)' /usr/share/X11/xkb/rules/base.lst
! model
! layout
! variant
! option
```

Notare che ogni variante mostra nella descrizione il layout alla quale viene applicata.

Spesso c'è bisogno di configurare solo il layout. Ad esempio per ottenere i file di localizzazione per il layout di tastiera tedesco e svizzero-tedesco in X:

```
$ lb config --bootappend-live "boot=live components locales=de`CH.UTF-8 keyboard-layouts=ch"
```

Tuttavia per casi molto particolari si vorrà includere altri parametri.

Ad esempio per configurare un sistema in francese con un layout Dvorak (chiamato Bepo) su una tastiera USB TypeMatrix EZ-Reach 2030:

```
$ lb config --bootappend-live "
boot=live components locales=fr`FR.UTF-8 keyboard-
layouts=fr keyboard-variants=bepo keyboard-model=
tm2030usb"
```

Per ogni opzione keyboard-* si possono specificare più valori separati da una virgola, con l'eccezione di keyboard-model che ne accetta uno solo. Consultare la pagina di manuale di keyboard(5) per dettagli ed esempi delle variabili XKBMODEL, XKBLAYOUT, KKBVARIANT e KKBOPTIONS. Se vengono forniti più valori per keyboard-variants, questi verranno combinati uno ad uno con quelli di keyboard-layouts (vedere l'opzione -variant in setxkbmap(1)). Sono permessi valori vuoti, ad esempio per definire due layout, US QWERTY come predefinito e US Dvorak, usare:

```
$ lb config --bootappend-live "
boot=live components keyboard-layouts=us,us keyboard-
variants=,dvorak"
```

10.3 Persistenza

Uno dei paradigmi di un cd live è un sistema preinstallato eseguito da un supporto in sola lettura, come un cdrom, dove le modifiche non sopravvivono ai riavvii dell'hardware della macchina ospitante.

Un sistema live è una generalizzazione di questo paradigma e di conseguenza oltre ai CD gestisce altri supporti; ma comunque, nel suo comportamento predefinito, deve essere considerato in sola lettura e

tutte i cambiamenti fatti durante l'esecuzione del sistema verranno persi allo spegnimento.

Persistenza il nome comune per differenti tipi di soluzioni per salvare alcune o tutte queste modifiche con i riavvii. Per capire come funziona potrebbe essere utile sapere che sebbene il sistema venga avviato ed eseguito da un dispositivo in sola lettura, le modifiche a file e directory vengono scritte su uno scrivibile, tipicamente un ram disk (tmpfs) e i dati sui ram disk non sopravvivono ai riavvii.

I dati immagazzinati su questo ramdisk andrebbero salvati un supporto scrivibile persistente come un supporto di memorizzazione locale, una condivisione di rete o anche una sessione di un CD/DVD riscrivibile multisessione. Tutti questi supporti sono gestiti in modi differenti e tutti tranne l'ultimo richiedono un parametro d'avvio speciale da specificare all'avvio: persistence.

Se il parametro di boot persistence impostato (e non lo noperistence), i supporti di memorizzazione locali (hard disk, dispositivi USB) saranno rilevati come volumi persistenti durante l'avvio. possibile selezionare quali tipi utilizzare specificando certi parametri di avvio descritti nella manpage di live-boot(7). Un volume persistente uno dei seguenti:

una partizione, identificata dal suo nome GPT (GUID Partition Table).

un filesystem, identificato dalla sua label.

un file immagine situato nella directory radice di un qualsiasi filesystem leggibile (anche una partizione NTFS di un sistema estraneo), identificato dal nome del file.

La label del volume per le stratificazioni deve essere persistence ma verr ignorata a meno che non sia presente nella directory radice un file chiamato persistence.conf che viene usato per personalizzare la persistenza del volume, in altre parole, specificare le directory che

si vogliono salvare dopo un riavvio. Per maggiori dettagli vedere [Il file persistence.conf](#).

Ecco alcuni esempi per preparare un volume da utilizzare per la persistenza. Pu ad esempio essere una partizione ext4 su un hard disk o una penna USB creata con:

```
# mkfs.ext4 -L persistence /dev/sdb1
```

Vedere anche [Usare lo spazio rimanente su una penna USB](#).

Se si possiede gi una partizione sul dispositivo basta solo cambiare l'etichetta con una delle seguenti:

```
# tune2fs -L persistence /dev/sdb1 # per filesystem ext2,3,4
```

Un esempio di come creare un file immagine ext4 da utilizzare per la persistenza:

```
$ dd if=/dev/null of=persistence bs=1 count=0 seek=1G # for a↵
1GB sized image file
$ /sbin/mkfs.ext4 -F persistence
```

Una volta che il file immagine stato creato, ad esempio per rendere /usr persistente salvando solo le modifiche fatte a quella directory e non tutto il contenuto di /usr, si pu usare l'opzione union. Se l'immagine situata nella propria home copiarla nella radice del filesystem sul disco e montarla in /mnt come segue:

```
# cp persistence /
# mount -t ext4 /persistence /mnt
```

Creare quindi il file `persistence.conf` aggiungendovi il contenuto e smontare il file immagine.

```
# echo "/usr union" && /mnt/persistence.conf
# umount /mnt
```

Ora riavviare il dispositivo live con il parametro d'avvio `persistence`.

10.3.1 Il file `persistence.conf`

Un volume con la label `persistence` deve essere configurato mediante il file `persistence.conf` per creare directory persistenti arbitrarie. Tale file, situato nella directory radice del filesystem del volume, controlla quali rendere persistenti e in che modo.

Nella manpage di `persistence.conf(5)` descritto dettagliatamente come configurato il mount degli strati personalizzati, ma un semplice esempio dovrebbe essere sufficiente per la maggior parte degli usi. Supponendo di voler creare la directory `home` e quella della cache di APT in modo persistente in un filesystem `ext4` sulla partizione `/dev/sdb1`:

```
# mkfs.ext4 -L persistence /dev/sdb1
# mount -t ext4 /dev/sdb1 /mnt
# echo "/home" && /mnt/persistence.conf
# echo "/var/cache/apt" && /mnt/persistence.conf
# umount /mnt
```

Quindi riavviare. Durante il primo avvio il contenuto di `/home` e `/var/cache/apt` saranno copiati nel volume persistente e da allora tutte le modifiche a queste directory risiederanno in modo persistente sul volume. C'è da considerare che tutti i path elencati nel file `persistence.conf` non possono contenere spazi o i caratteri speciali

599e ..., inoltre `n /lib, /lib/live` (o una delle sue sottodirectory) `n /` pu essere resa persistente tramite i mount personalizzati. Come workaround a questa limitazione possibile aggiungere `/ union` al file `persistence.conf` file per ottenere la persistenza completa.

10.3.2 Utilizzare più di un'archiviazione persistente

Ci sono tre metodi differenti di utilizzare persistenze multiple per differenti casi d'uso. Ad esempio l'utilizzo di svariati volumi contemporaneamente o selezionandone uno solo per scopi molto specifici.

Possono essere utilizzati svariati volumi di stratificazione personalizzati (con i rispettivi file `persistence.conf`) allo stesso tempo ma se questi creano la stessa directory persistente, ne verrà usata solo una. Se due directory montate sono nidificate (una la sottodirectory dell'altra), la superiore sarà montata per prima, per cui nessuna operazione di mount verrà sovrastata dall'altra. I mount nidificati personalizzati sono problematici se sono elencati nello stesso file `persistence.conf`. Se si ha davvero la necessità (in genere non si dovrebbe averla), consultare la manpage di `persistence.conf(5)` per sapere come gestire questo caso.

One possible use case: If you wish to store the user data i.e. `/home` and the superuser data i.e. `/root` in different partitions, create two partitions with the persistence label and add a `persistence.conf` file in each one like this, `# echo /home & persistence.conf` for the first partition that will save the user's files and `# echo /root & persistence.conf` for the second partition which will store the superuser's files. Finally, use the persistence boot parameter.

Se un utente avesse bisogno di spazi di archiviazione multipli dello stesso tipo per posizioni differenti o per test, come privato e lavoro, il parametro d'avvio `persistence-label` usato in congiunzione con `persistent` permetterà supporti persistenti multipli ma univoci.

Un esempio potrebbe essere un utente che vuole usare una partizione etichettata come privato per dati personali come i preferiti del browser o di altro tipo, questi user i parametri d'avvio persistence persistence-label=privato. E per archiviare dati inerenti il lavoro, come documenti, ricerche e altro, verranno usati i parametri d'avvio persistence persistence-label=lavoro.

importante ricordare che ognuno di questi volumi, privato e lavoro, necessitano anche di un file persistence.conf nella propria radice. Il manuale di live-boot contiene altre informazioni su come utilizzare queste etichette con nomi usati in versioni precedenti.

10.3.3 Using persistence with encryption

Using the persistence feature means that some sensible data might get exposed to risk. Especially if the persistent data is stored on a portable device such as a usb stick or an external hard drive. That is when encryption comes in handy. Even if the entire procedure might seem complicated because of the number of steps to be taken, it is really easy to handle encrypted partitions with live-boot. In order to use luks, which is the supported encryption type, you need to install cryptsetup both on the machine you are creating the encrypted partition with and also in the live system you are going to use the encrypted persistent partition with.

To install cryptsetup on your machine:

```
# apt-get install cryptsetup
```

To install cryptsetup in your live system, add it to your package-lists:

```
$ lb config
$ echo "cryptsetup cryptsetup-initramfs" & config/package-<←
lists/encryption.list.chroot
```

Once you have your live system with cryptsetup, you basically only need to create a new partition, encrypt it and boot with the persistence and persistence-encryption=luks parameters. We could have already anticipated this step and added the boot parameters following the usual procedure:

```
$ lb config --bootappend-live "boot=live components <←
persistence persistence-encryption=luks"
```

Let's go into the details for all of those who are not familiar with encryption. In the following example we are going to use a partition on a usb stick which corresponds to /dev/sdc2. Please be warned that you need to determine which partition is the one you are going to use in your specific case.

The first step is plugging in your usb stick and determine which device it is. The recommended method of listing devices in live-manual is using ls -l /dev/disk/by-id. After that, create a new partition and then, encrypt it with a passphrase as follows:

```
# cryptsetup --verify-passphrase luksFormat /dev/sdc2
```

Then open the luks partition in the virtual device mapper. Use any name you like. We use live here as an example:

```
# cryptsetup luksOpen /dev/sdc2 live
```

The next step is filling the device with zeros before creating the

filesystem:

638

Let's summarize the process. So far, we have created an encryption capable live system, which can be copied to a usb stick as explained in [Copying an ISO hybrid image to a USB stick](#). We have also created an encrypted partition, which can be located in the same usb stick to carry it around and we have configured the encrypted partition to be used as persistence store. So now, we only need to boot the live system. At boot time, live-boot will prompt us for the passphrase and will mount the encrypted partition to be used for persistence.

627

```
# dd if=/dev/zero of=/dev/mapper/live
```

628

Now, we are ready to create the filesystem. Notice that we are adding the label persistence so that the device is mounted as persistence store at boot time.

629

```
# mkfs.ext4 -L persistence /dev/mapper/live
```

630

To continue with our setup, we need to mount the device, for example in /mnt.

631

```
# mount /dev/mapper/live /mnt
```

632

And create the persistence.conf file in the root of the partition. This is, as explained before, strictly necessary. See [The persistence.conf file](#).

633

```
# echo "/ union" > /mnt/persistence.conf
```

634

Then unmount the mount point:

635

```
# umount /mnt
```

636

And optionally, although it might be a good way of securing the data we have just added to the partition, we can close the device:

637

```
# cryptsetup luksClose live
```

Personalizzare l'immagine binaria

```
include menu.cfg
default vesamenu.c32
prompt 0
timeout 50
```

11. Personalizzare l'immagine binaria

11.1 Bootloader

live-build usa syslinux e alcuni dei suoi derivati (a seconda del tipo di immagine) come bootloader predefiniti. Si possono facilmente personalizzare per soddisfare le proprie esigenze.

Per utilizzare un tema completo, copiare `/usr/share/live/build/-bootloaders` in `config/bootloaders` e modificare i file. Se non si vogliono modificare tutte le configurazioni dei bootloader supportati sufficiente fornire la copia locale di uno di essi, ad esempio `isolinux` in `config/bootloaders/isolinux` pu bastare, dipende dalle esigenze.

When modifying one of the default themes, if you want to use a personalized background image that will be displayed together with the boot menu, add a `splash.png` picture of 640x480 pixels. Then, remove the `splash.svg` file.

Quando si tratta di fare modifiche ci sono varie possibilit. Per esempio i derivati di `syslinux` sono configurati con un timeout impostato a 0 (zero) in modo predefinito, significa che resteranno in pausa al loro splash screen fino a quando non si preme un tasto.

Per modificare il timeout di avvio di un'immagine iso-hybrid modificare un file `isolinux.cfg` predefinito specificando il timeout in unit di 1/10 di secondo. Un file `isolinux.cfg` modificato per effettuare il boot dopo cinque secondi sarebbe simile a questo:

11.2 Metadati ISO

Quando si crea un'immagine binaria ISO9660, si possono usare le seguenti opzioni per aggiungere vari metadati testuali. Questo pu aiutare a identificare facilmente la versione o la configurazione di un'immagine senza avviarla.

`LB'ISO'APPLICATION/`–iso-application NAME: descrive l'applicazione che sar nell'immagine. La lunghezza massima per questo campo di 128 caratteri.

* `LB'ISO'PREPARER/`–iso-preparer NAME: descrive il costruttore dell'immagine, solitamente con alcuni dettagli per contattarlo. L'impostazione predefinita la versione di live-build che si sta usando, il quale potr essere utile in seguito per il debugging. La lunghezza massima per questo campo di 128 caratteri.

`LB'ISO'PUBLISHER/`–iso-publisher NAME: descrive l'editore dell'immagine, solitamente con qualche dettaglio per contattarlo. La lunghezza massima lunghezza per questo campo di 128 caratteri.

`LB'ISO'VOLUME/`–iso-volume NAME: specifica l'ID del volume dell'immagine. Questa utilizzata come etichetta visibile all'utente su alcune piattaforme, come Windows e Apple Mac OS. La lunghezza massima per questo campo di 128 caratteri.

Personalizzare l'Installatore Debian

12. Personalizzare l'Installatore Debian

Live system images can be integrated with Debian Installer. There are a number of different types of installation, varying in what is included and how the installer operates.

In questa sezione si presti attenzione all'uso delle lettere maiuscole quando si fa riferimento all'Installatore Debian, quando usato ci si riferisce esclusivamente all'installatore ufficiale Debian. Spesso abbreviato come d-i.

12.1 Tipologie dell'Installatore Debian

I tre principali tipi dell'installer sono:

Normal Debian Installer : This is a normal live system image with a separate kernel and initrd which (when selected from the appropriate bootloader) launches into a standard Debian Installer instance, just as if you had downloaded a CD image of Debian and booted it. Images containing a live system and such an otherwise independent installer are often referred to as combined images.

In queste immagini, Debian installata prendendo e installando i pacchetti .deb usando debootstrap, da supporti locali o dalla rete, risultante in un sistema Debian standard installato sul disco rigido.

L'intero processo pu essere preimpostato e personalizzato in diversi modi; per ulteriori informazioni si vedano le corrispondenti pagine del manuale dell'Installatore Debian. Una volta che si ha un file

preimpostato funzionante, live-build pu inserirlo automaticamente nell'immagine e abilitarlo.

Live Debian Installer : This is a live system image with a separate kernel and initrd which (when selected from the appropriate bootloader) launches into an instance of the Debian Installer.

L'installazione proceder nello stesso modo di un'installazione Normale come descritto sopra, ma nella fase dell'installazione del pacchetto, invece di usare debootstrap per prelevare e installare i pacchetti, l'immagine del filesystem live viene copiata sulla destinazione. Questo si ottiene con uno speciale udeb chiamato live-installer.

Dopo questa fase, l'Installatore Debian continua normalmente, installando e configurando elementi come bootloader e utenti locali, ecc.

Nota: per supportare nel bootloader sia la voce normale che quella live dell'installatore sullo stesso supporto si deve disabilitare live-installer preconfigurando live-installer/enable=false.

Installatore Debian Desktop : indipendentemente dal tipo di Installatore Debian incluso, d-i pu essere lanciato cliccando un'icona sul desktop, in alcune situazioni pi semplice per l'utente. Per poterne usufruire deve essere incluso il pacchetto debian-installer-launcher.

Si noti che live-build non include l'Installatore Debian nell'immagine in modo predefinito, necessita di essere espressamente abilitato con lb config. Inoltre, affinch l'installatore Desktop funzioni, il kernel del sistema live deve corrispondere a quello usato dal d-i per l'architettura specificata. Per esempio:

```
$ lb config --debian-installer live
$ echo debian-installer-launcher && config/package-lists/my.<
list.chroot
```

12.2 Personalizzare il Debian Installer con la preconfigurazione

As described in the Debian Installer Manual, Appendix B at <https://www.debian.org/releases/stable/amd64/apb.en.html>, Preseeding provides a way to set answers to questions asked during the installation process, without having to manually enter the answers while the installation is running. This makes it possible to fully automate most types of installation and even offers some features not available during normal installations. This kind of customization is best accomplished with live-build by placing the configuration in a preseed.cfg file included in config/includes.installer/. For example, to preseed setting the locale to en`US:

```
$ echo "d-i debian-installer/locale string en`US" "  
    `ll config/includes.installer/preseed.cfg
```

12.3 Personalizzare il contenuto dell'Installatore Debian

For experimental or debugging purposes, you might want to include locally built d-i component udeb packages. Place these in config/packages.binary/ to include them in the image. Additional or replacement files and directories may be included in the installer initrd as well, in a similar fashion to **Live/chroot local includes**, by placing the material in config/includes.installer/.

Progetto

Contribuire al progetto

13. Contribuire al progetto

When submitting a contribution, please clearly identify its copyright holder and include any applicable licensing statement. Note that to be accepted, the contribution must be licensed under the same license as the rest of the documents, namely, GPL version 3 or later.

Contributions to the project, such as translations and patches, are greatly welcome. Anyone can send merge requests. The projects are hosted on Salsa: <https://salsa.debian.org/live-team> follow Salsa's documentation for instructions on how to contribute.

Anche se tutti i commit possono essere corretti, chiediamo di usare il buon senso ed eseguire buoni commit con dei buoni messaggi.

Si scrivano messaggi costituiti da frasi in inglese esaurienti e utili, inizianti con una lettera maiuscola e terminanti con un punto. Solitamente cominceranno con la forma Fixing/Adding/-Removing/Correcting/Translating/....

Scrivere buoni messaggi nei commit. La prima riga deve contenere un sunto accurato del contenuto del commit in quanto verrà incluso nel changelog. Se si necessita di aggiungere ulteriori spiegazioni, scriverle sotto lasciando una riga vuota dopo la prima e quindi un'altra vuota dopo ogni paragrafo. Le righe non devono superare gli 80 caratteri.

Eseguire commit atomici, ovvero non mescolare cose non inerenti tra loro nello stesso commit ma farne uno per ogni modifica apportata.

13.1 Translation of man pages

You can also contribute to the project working on the translation of the man pages for the different live-* packages that the project maintains. The procedure is different depending on whether you are starting a translation from scratch or continue working on an already existing one:

Working on an already existing translation

If you want to maintain the translation of an already existing language you have to make your changes to your manpages/po/\$-LANGUAGE"/*.po file or files and then run make rebuild from inside the manpages/ directory. This will update the actual man pages in manpages/\$-LANGUAGE"/*

Starting a new translation from scratch

In order to add a new translation of any of the project's man pages you have to follow a similar procedure. It could be summarized as follows:

Open the manpages/pot/ file or files in your favourite editor, such as poedit, and save it as a .po file in manpages/po/\$-LANGUAGE"/. (You will have to create your \$-LANGUAGE"/ directory).

Run make rebuild from inside the manpages/ directory to create the manpages/\$-LANGUAGE"/ files which will contain the actual man pages.

Remember that you will have to add all the directories and files, then make the commit and finally push to the git server.

Segnalare bug

14. Segnalare bug

Live systems are far from being perfect, but we want to make it as close as possible to perfect - with your help. Do not hesitate to report a bug. It is better to fill a report twice than never. However, this chapter includes recommendations on how to file good bug reports.

Per gli impazienti

First check whether the bugs has been reported already. You can see the full list of bugs that are assigned to the live-team at <https://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=debian-live%40lists.debian.org>.

Before submitting a bug report always try to reproduce the bug with the most recent versions of the packages of live-build, live-boot, live-config and live-tools that you're using.

Si cerchi di fornire informazioni il pi dettagliate possibile riguardo il bug. Questo comprende (almeno) la versione di live-build, live-boot, live-config e live-tools utilizzata e la distribuzione del sistema live che si sta creando.

14.1 Problemi noti

Currently known issues are listed in the BTS at <https://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=debian-live%40lists.debian.org>.

Note: Since Debian testing and Debian unstable distributions are moving targets, when you specify either of them as the target system distribution, a successful build may not always be possible.

Se questo causa troppe difficolta, non creare un sistema basato su

testing o unstable ma usare piuttosto stable . live-build si basa su stable in modo predefinito.

It is out of the scope of this manual to train you to correctly identify and fix problems in packages of the development distributions, however, you can always try the following: If a build fails when the target distribution is testing , try unstable . If unstable does work, revert to testing and pin the newer version of the failing package from unstable (see [APT pinning](#) for details).

14.2 Fare la ricerca

Prima di riportare il bug si prega di cercare sul web il messaggio d'errore o il sintomo ottenuti. Poich altamente improbabile essere l'unica persona ad incontrare un certo problema, c' sempre la possibilit che sia stato discusso altrove e che siano stati proposte una soluzione, una patch o soluzione temporanea.

You should pay particular attention to the live systems mailing list, as well as the homepage, as these are likely to contain the most up-to-date information. If such information exists, always include the references to it in your bug report.

In aggiunta bisogna controllare l'attuale elenco dei bug riguardanti live-build, live-boot, live-config e live-tools per vedere se sia gi stato segnalato qualcosa di simile.

14.3 Ricompilare da zero

Per essere certi che un particolare bug non sia causato dalla creazione di un sistema non pulito, ricostruire sempre l'intero sistema da zero per vedere se il bug sia riproducibile.

14.4 Usare pacchetti aggiornati

Using outdated packages can cause significant problems when trying to reproduce (and ultimately fix) your problem. Make sure your build system is up-to-date and any packages included in your image are up-to-date as well. If possible, try to reproduce the bug with the newest code from source, see [Installation](#) for details.

14.5 Raccogliere informazioni

Nella segnalazione si invita a fornire informazioni sufficienti. Dovrebbe almeno contenere l'esatta versione di live-build nella quale si trovato il bug e i passi per riprodurlo. Con un po' di buon senso si pu includere qualsiasi altro dettaglio rilevante che si ritiene utile per la risoluzione del problema.

Affinch la segnalazione del bug sia migliore possibile, si richiedono almeno le seguenti informazioni:

Architettura del sistema ospitante

Distribution of the host system

Versione di live-build sul sistema ospitante

Version of debootstrap on the host system

Architettura del sistema live

Distribuzione del sistema live

Versione di live-boot sul sistema ospitante

Versione di live-config sul sistema live

Versione di live-tools sul sistema ospitante

possibile generare un registro del processo di costruzione usando il comando tee. Si raccomanda di farlo automaticamente con uno

script auto/build; (si veda [Gestire una configurazione](#) per i dettagli).

```
# lb build 2i&1 — tee build.log
```

All'avvio, live-boot e live-config conservano i loro registri in /var/log/live/. Controllarvi gli errori.

Inoltre, per escludere altri errori sempre una buona idea creare un tar della propria directory config/ e caricarlo da qualche parte (non inviarlo come allegato alla mailing list), in modo che sia per noi possibile riprodurre gli errori incontrati. Se ci causa problemi (ad esempio a causa della dimensione) si pu utilizzare l'output di lb config -dump che produce un sommario dell'albero di configurazione (elenca i file nelle sottodirectory di config/ ma non le include).

Ricordarsi che i file di registro da inviare vanno creati con l'impostazione della lingua inglese, ad esempio eseguendo il comando live-build preponendo LC'ALL=C oppure LC'ALL=en'US.

14.6 Se possibile isolare il caso non andato a buon fine

Se possibile, isolare il caso non andato a buon fine alla variazione pi piccola che lo causa. Non sempre facile da fare, perci non preoccupatevi se non riuscite a gestirlo per la vostra segnalazione. Tuttavia, se si pianifica bene il ciclo di sviluppo adottando piccole modifiche per ogni iterazione, si riuscir ad isolare il problema creando una configurazione semplificata che si avvicina all'attuale con l'aggiunta delle sole modifiche problematiche. Se si incontrano serie difficolt nel trovare la causa, potrebbe essere che sono stati inseriti troppi cambiamenti in una sola volta e bisogna cambiare approccio.

14.7 Segnalare il bug del pacchetto giusto

In genere bisogna riportare gli errori in fase di compilazione verso il pacchetto live-build, quelli di avvio verso live-boot e quelli in fase di esecuzione a live-config. Se non siete certi di quale sia il pacchetto appropriato o serve maggiore aiuto prima della segnalazione, inviate una segnalazione per lo pseudo-pacchetto debian-live. Ce ne occuperemo riassegnandolo dove pi  appropriato.

Tuttavia vi saremmo grati se tentate di restringere il campo in base a dove appare il bug.

14.7.1 Durante la compilazione mentre esegue il bootstrap

live-build first bootstraps a basic Debian system with debootstrap. If a bug appears here, check if the error is related to a specific Debian package (most likely), or if it is related to the bootstrapping tool itself.

In both cases, this is not a bug in the live system, but rather in Debian itself and probably we cannot fix it directly. Please report such a bug against the bootstrapping tool or the failing package.

14.7.2 Durante la compilazione mentre installa i pacchetti

live-build installa pacchetti aggiuntivi dall'archivio Debian e pu  fallire a seconda della distribuzione Debian e lo stato dell'archivio giornaliero. Se il bug appare a questo punto, controllare che l'errore sia riproducibile su un sistema normale.

If this is the case, this is not a bug in the live system, but rather in Debian - please report it against the failing package. Running debootstrap separately from the Live system build or running lb bootstrap -debug will give you more information.

Se si verifica un problema utilizzando un mirror locale o un qualsiasi

tipo di proxy bene riprodurlo avviando da un mirror ufficiale.

14.7.3 In fase di avvio

Se l'immagine non si avvia segnalarlo alla mailing list con le informazioni richieste in [Raccogliere informazioni](#). Non dimenticare di menzionare esattamente come e quando l'immagine fallisce, utilizzando la virtualizzazione o hardware reale. Se si utilizza un qualsiasi sistema di virtualizzazione provare sempre su hardware reale prima di segnalare un bug; anche fornire un'istantanea dello schermo pu  essere molto utile.

14.7.4 In fase di esecuzione

If a package was successfully installed, but fails while actually running the Live system, this is probably a bug in live-config.

14.8 Dove segnalare i bug

The Debian Live Project keeps track of all bugs in the Bug Tracking System (BTS). For information on how to use the system, please see <https://bugs.debian.org/>. You can also submit the bugs by using the reportbug command from the package with the same name.

Si noti che i bug trovati nelle distribuzioni derivate da Debian (come Ubuntu e altre) non vanno segnalati a Debian BTS a meno che non siano riproducibili anche su un sistema Debian utilizzando pacchetti ufficiali Debian.

Lo stile nello scrivere codice

15. Lo stile nello scrivere codice

This chapter documents the coding style used in live systems.

15.1 Compatibilit

Avoid bashisms, the codebase must be POSIX compliant and thus universally compatible.

Furthermore it must comply with the version of the POSIX specification chosen by the current Debian Policy.

possibile verificare i propri script con `sh -n` e `checkbashisms`.

Assicurarsi che tutto il codice giri con `'set -e'`.

15.2 Rientri

Usare sempre i tab piuttosto che gli spazi.

Keep case branch terminators (`;;`) aligned with the content of the branch, rather than the branch entry.

Corretto:

```
case "$1" in
    foo)
        foobar
        ;;
    bar)
        foobar
        ;;
esac
```

15.3 Ritorno a capo

Generally, lines should be 80 chars at maximum.

Placement of keywords like `then` and `do` should be chosen with good judgement with respect to clutter and readability. For small bits of code in particular it should be preferred to have them on the same line as the prior keyword they relate to (`if`; `for`; etc). Only place on the next line where it makes good sense to do so; typically this might only be to comply with maximum line length restrictions. One situation where they should always be placed on the next line is where what they follow is broken up onto multiple lines, and thus it being on a new line creates clear separation between that and the body of code following it. I.e. :

Preferred:

```
if foo; then
    bar
fi

for FOO in $ITEMS; do
    bar
done

if [ "$MY'LOCATION'VARIABLE" = "something" ] && [ -e "$MY'OUTPUT'FILE" ]
then
    MY'OTHER'VARIABLE="$(some`bin $-FOOBAR" — awk -F` ' —`"
    print $1 "`)"
fi

if [ "$MY'FOO'" = "something" ] && [ -e "path/$-FILE`1" ] &&
[ "$MY'BAR'" = "something`else" ] && [ $-ALLOW = "true" ]
then
    foobar
fi
```

Less ideal:

```
if [ "$MYLOCATIONVARIABLE" = "something" ] && [ -e "$MYOUTPUTFILE" ]; then
    MYOTHERVARIABLE="$(some`bin $-FOOBAR" — awk -F' ' —
    print $1 " ")"
fi
```

```
Foo ()
-
    bar
"
```

Horrible:

```
if [ "$MYLOCATIONVARIABLE" = "something" ] && [ -e "$MYOUTPUTFILE" ] — [ "$MYLOCATIONVARIABLE" = "something-else" ] && [ -e "$MYOUTPUTFILE2" ]; then
    MYOTHERVARIABLE="$(some`bin $-FOOBAR" — awk -F' ' —
    print $1 " ")"
fi
```

Prefer placing the opening brace of a function on a new line (for consistency with established style), and keep the braces aligned with the function name:

Corretto:

```
Foo ()
-
    bar
"
```

Bad (inconsistent with existing style):

```
Foo () -
    bar
"
```

Awful:

15.4 Variabili

Le variabili vanno sempre scritte in maiuscolo.

Config variables used in live-build should start with an LB` prefix.

Local function variables should be restricted to local scope.

Le variabili in live-config relative ai parametri di avvio iniziano con LIVE`.

Tutte le altre variabili in live-config iniziano con il prefisso `.

Intorno alle variabili utilizzare le graffe; ad esempio scrivere \$-FOO" invece di \$FOO.

Always protect variables with quotes to respect potential whitespaces (except where necessary to achieve correct word splitting): write \$-FOO" not \$-FOO".

Per coerenza usare sempre le virgolette quando si assegnano valori alle variabili:

Sbagliato:

```
FOO=bar
```

Corretto:

```
FOO="bar"
```

If multiple variables are used, prefer quoting the full expression:

Typically bad:

792

793

```
if [ -f "$-FOO"/foo/"$-BAR"/bar ]; then
    foobar
fi
```

Corretto:

794

795

```
if [ -f "$-FOO"/foo/"$-BAR"/bar ]; then
    foobar
fi
```

15.5 Varie

796

Prefer `—` (without the surround quotes) as a separator in calls to `sed`, e.g. `sed -e 's—'` (without `"`).

797

Don't use the test command for comparisons or tests, use `[` and `]` (without `"`); e.g. `if [-x /bin/foo]; ...` and not `if test -x /bin/foo; ...`

798

Use case wherever it makes code more readable than conditional checks (`if foo; ...` and tests without the actual `if` keyword, e.g. `[-e $-FILE"] —exit 0`).

799

Use Foo·bar style names for functions, i.e. a capital first letter, then all lowercase, with sensible use of underscores for better readability.

800

Esempi

Esempi

16. Esempi

This chapter covers example builds for specific use cases with live systems. If you are new to building your own live system images, we recommend you first look at the three tutorials in sequence, as each one teaches new techniques that will help you use and understand the remaining examples.

16.1 Usare gli esempi

Per usare questi esempi è necessario un sistema per costruirveli sopra che soddisfi i requisiti elencati in [Requisiti](#) e avere live-build installato come descritto in [Installare live-build](#).

da notare che per brevità in questi esempi non specifichiamo un mirror locale da usare per la costruzione. Usando un mirror locale, si possono accelerare considerevolmente i download. Si possono specificare le opzioni quando si usa `lb config`, come descritto in [Mirror delle distribuzioni usati in fase di compilazione](#) o, più convenientemente, impostare il predefinito per il proprio sistema in `/etc/live-build.conf`. Si crei semplicemente questo file e si impostino in esso le corrispondenti variabili `LB_MIRROR*` per il mirror desiderato. Tutti gli altri mirror utilizzati nella costruzione avranno questi valori, ad esempio:

```
LB_MIRROR_BOOTSTRAP="http://mirror/debian/"
LB_MIRROR_CHROOT_SECURITY="http://mirror/debian-security/"
LB_MIRROR_CHROOT_BACKPORTS="http://mirror/debian-updates/"
```

16.2 Tutorial 1: un'immagine predefinita

Caso d'uso: creazione di una prima semplice immagine, imparando i fondamenti di live-build.

In this tutorial, we will build a default ISO hybrid live system image containing only base packages (no Xorg) and some live system support packages, as a first exercise in using live-build.

Non può essere più semplice:

```
$ mkdir tutorial1 ; cd tutorial1 ; lb config
```

Esaminare i contenuti della directory `config/`; si noteranno uno scheletro di configurazione pronto per essere personalizzato o, in questo caso, usato immediatamente per costruire un'immagine predefinita.

Ora, come utente `root`, generare l'immagine salvando un log con `tee`.

```
# lb build 2i&1 -- tee build.log
```

Assuming all goes well, after a while, the current directory will contain `live-image-amd64.hybrid.iso`. This ISO hybrid image can be booted directly in a virtual machine as described in [Testing an ISO image with Qemu](#) and [Testing an ISO image with VirtualBox](#), or else imaged onto optical media or a USB flash device as described in [Burning an ISO image to a physical medium](#) and [Copying an ISO hybrid image to a USB stick](#), respectively.

16.3 Tutorial 2: servizio browser web

Caso d'uso: creazione di un'immagine per servizio browser web, imparando come applicare le personalizzazioni.

In this tutorial, we will create an image suitable for use as a web browser utility, serving as an introduction to customizing live system images.

```
$ mkdir tutorial2
$ cd tutorial2
$ lb config
$ echo "task-lxde-desktop firefox-esr" && config/package-<
lists/my.list.chroot
```

La scelta di LXDE per questo esempio riflette il desiderio di fornire un ambiente desktop minimale, dato che il punto focale dell'immagine il singolo uso che abbiamo in mente, il browser web. Potremmo anche spingerci oltre e fornire una configurazione predefinita per il browser web in config/includes.chroot/etc/iceweasel/profile/, o pacchetti aggiuntivi di supporto per la fruizione di vari tipi di contenuti web, ma lasciamo questo come esercizio per il lettore.

Generare l'immagine, ancora come utente root, conservando un log come in [Tutorial 1](#):

```
# lb build 2&1 — tee build.log
```

Di nuovo, verificare che l'immagine sia a posto e collaudarla, come in [Tutorial 1](#).

16.4 Tutorial 3: un'immagine personalizzata

Caso d'uso: creazione di un progetto per costruire un'immagine personalizzata che contiene i pacchetti preferiti da portare con sè in una chiavetta USB ovunque si vada, e che evolve in revisioni successive allorché i bisogni o le preferenze cambino.

Dal momento che la nostra immagine personalizzata cambierà con le successive revisioni e che vogliamo tener traccia di questi cambiamenti, andando per tentativi ed eventualmente tornando indietro se qualcosa non funziona, conserveremo la nostra configurazione nel popolare sistema di controllo di versione git. Useremo anche le migliori pratiche di auto-configurazione tramite gli script auto come descritto in [Gestire una configurazione](#).

16.4.1 Prima revisione

```
$ mkdir -p tutorial3/auto
$ cp /usr/share/doc/live-build/examples/auto/* tutorial3/auto<
/
$ cd tutorial3
```

Modificare auto/config come segue:

```
#!/bin/sh

lb config noauto "
--distribution stable "
"$@"
```

Eeguire lb config per generare l'albero di configurazione utilizzando lo script auto/config appena creato:

```
$ lb config
```

Popolare ora l'elenco locale dei pacchetti:

```
$ echo "task-lxde-desktop spice-vdagent hexchat" && config/<
package-lists/my.list.chroot
```

First, `--distribution stable` ensures that `stable` is used instead of the default `--testing`. Second, we have added `spice-vdagent` for easier testing the image in `qemu`. And finally, we have added an initial favourite package: `hexchat`.

Costruire quindi l'immagine:

```
# lb build
```

Notare che diversamente dai primi due tutorial non occorre più digitare `2i&1 --tee build.log` dato che questo ora incluso in `auto/build`.

Una volta che l'immagine stata collaudata (come in [Tutorial 1](#)) e che si sicuri che funzioni correttamente, il momento di inizializzare il repository `git`, aggiungendo solo gli script `auto` appena creati, e di fare poi il primo commit:

```
$ git init
$ cp /usr/share/doc/live-build/examples/gitignore .gitignore
$ git add .gitignore auto config
$ git commit -m "Initial import."
```

16.4.2 Seconda revisione

In this revision, we're going to clean up from the first build, replace the `smplayer` package with `vlc` package, rebuild, test and commit.

Il comando `lb clean` ripulir tutti i file ottenuti con la precedente generazione eccetto la cache, che ci evita un nuovo download dei pacchetti. Ci assicura che il successivo `lb build` eseguir di nuovo tutti i passaggi per rigenerare i file dalla nuova configurazione.

```
# lb clean
```

Now install the `vlc` package before the `lxde` package chooses between `smplayer`, `vlc` and `mplayer-gui` in our local package list in `config/package-lists/my.list.chroot`:

```
$ echo "vlc task-lxde-desktop spice-vdagent hexchat" >>
config/package-lists/my.list.chroot
```

Rigenerare nuovamente:

```
# lb build
```

Verificare e, quando soddisfatti, eseguire il commit della revisione successiva:

```
$ git commit -a -m "Replacing smplayer with vlc."
```

Ovviamente sono possibili cambiamenti alla configurazione più complicati, magari aggiungendo file in sottodirectory di `config/`. Quando si esegue il commit di nuove revisioni, si faccia solo attenzione a non modificare manualmente o fare un commit dei file al livello superiore di `config` che contengono le variabili `LB*`, giacché sono anche prodotti dell'assemblaggio, e che sono sempre ripuliti da `lb clean` e ricreati con `lb config` attraverso i loro rispettivi script `auto`.

We've come to the end of our tutorial series. While many more kinds of customization are possible, even just using the few features explored in these simple examples, an almost infinite variety of different images can be created. The remaining examples in this section cover several other use cases drawn from the collected experiences of users of live systems.

16.5 Un client Kiosk VNC

Caso d'uso: creazione di un'immagine con live-build per avviare direttamente un server VNC.

Creare una directory per la compilazione e una configurazione di base al suo interno disabilitando i raccomandati per ottenere un sistema minimale. Quindi creare due elenchi di pacchetti: il primo generato con uno script fornito da live-build chiamato Packages (vedere [Elenchi di pacchetti generati](#)) e il secondo che include xorg, gdm3, metacity e xvnc4viewer.

```
$ mkdir vnc-kiosk-client
$ cd vnc-kiosk-client
$ lb config --apt-recommends false
$ echo '! Packages Priority standard' > config/package-lists/standard.list.chroot
$ echo "xorg gdm3 metacity xtightvncviewer" > config/package-lists/my.list.chroot
```

Come spiegato in [Modificare APT per risparmiare spazio](#) potrebbe essere necessario riaggiungere alcuni pacchetti raccomandati al fine di far funzionare l'immagine correttamente.

Un modo semplice per elencare i raccomandati usare apt-cache, ad esempio:

```
$ apt-cache depends live-config live-boot
```

In questo esempio abbiamo scoperto che dobbiamo iserire nuovamente svariati pacchetti raccomandati da live-config e live-boot: user-setup perch il login automatico funzioni e sudo come programma essenziale per spegnere il sistema. Oltretutto pu essere comodo aggiungere live-tools per poter copiare l'immagine in RAM e eject per espellere il supporto live alla fine. Quindi:

```
$ echo "live-tools user-setup sudo eject" > config/package-lists/recommends.list.chroot
```

Successivamente creare la directory /etc/skel in config/includes.chroot e inserirvi un .xsession personalizzato per l'utente predefinito che lancer metacity e avvier xvncviewer connesso alla porta 5901 su un server con indirizzo 192.168.1.2:

```
$ mkdir -p config/includes.chroot/etc/skel
$ cat > config/includes.chroot/etc/skel/.xsession ; EOF
#!/bin/sh

/usr/bin/metacity &
/usr/bin/xvncviewer 192.168.1.2:1

exit
EOF
```

Compilare l'immagine:

```
# lb build
```

Buon divertimento.

16.6 A minimal image for a 512MB USB key

Use case: Create a default image with some components removed in order to fit on a 512MB USB key with a little space left over to use as you see fit.

When optimizing an image to fit a certain media size, you need to understand the tradeoffs you are making between size and functionality. In this example, we trim only so much as to make room for additional material within a 512MB media size, but without doing

anything to destroy the integrity of the packages contained within, such as the purging of locale data via the `localepurge` package, or other such intrusive optimizations. Of particular note, we use `--debootstrap-options` to create a minimal system from scratch and `--binary-image hdd` to create an image that can be copied to a USB key.

```
$ lb config --binary-image hdd --apt-indices false --apt-recommends false --debootstrap-options "--variant=minbase" --firmware-chroot false --memtest none
```

Affinch l'immagine funzioni correttamente dobbiamo riaggiungere almeno due pacchetti raccomandati lasciati fuori dall'opzione `--apt-recommends false`. Vedere [Modificare APT per risparmiare spazio](#)

```
$ echo "user-setup sudo" && config/package-lists/recommends.list.chroot
```

Additionally, you'll want to have network access, so another two recommended packages need to be re-added:

```
$ echo "ifupdown isc-dhcp-client" && config/package-lists/recommends.list.chroot
```

Costruire quindi l'immagine nel modo consueto:

```
# lb build 2i&1 --tee build.log
```

On the author's system at the time of writing this, the above configuration produced a 298MiB image. This compares favourably with

the 380MiB image produced by the default configuration in [Tutorial 1](#), when `--binary-image hdd` is added.

Leaving off APT's indices with `--apt-indices false` saves a fair amount of space, the tradeoff being that you need to do an `apt-get update` before using `apt` in the live system. Dropping recommended packages with `--apt-recommends false` saves some additional space, at the expense of omitting some packages you might otherwise expect to be there. `--debootstrap-options --variant=minbase` bootstraps a minimal system from the start. Not automatically including firmware packages with `--firmware-chroot false` saves some space too. And finally, `--memtest none` prevents the installation of a memory tester.

Note: A minimal system can also be achieved using hooks, like for example the `stripped.hook.chroot` hook found in `/usr/share/doc/live-build/examples/hooks`. It may shave off additional small amounts of space and produce an image of 277MiB. However, it does so by removal of documentation and other files from packages installed on the system. This violates the integrity of those packages and that, as the comment header warns, may have unforeseen consequences. That is why using a minimal `debootstrap` is the recommended way of achieving this goal.

16.7 Un desktop GNOME localizzato e l'installatore

Caso d'uso: creazione di un'immagine con il desktop GNOME, localizzato in svizzero e che includa l'installatore.

We want to make an iso-hybrid image using our preferred desktop, in this case GNOME, containing all of the same packages that would be installed by the standard Debian installer for GNOME.

Il problema iniziale di scoprire i nomi dei task della lingua appropriati, attualmente, `live-build` non aiuta in questo. Si pu essere for-

tunati o arrivarci con vari tentativi, ma c'è uno strumento `grep-dctrl` il quale può essere utilizzato per scavare nelle descrizioni in `tasksel-data`, perciò assicurarsi di avere entrambi questi pacchetti:

```
# apt-get install dctrl-tools tasksel-data
```

```
--debian-installer live
$ echo '! Packages Priority standard' && config/package-lists/<↵
  standard.list.chroot
$ echo task-gnome-desktop task-german task-german-desktop && <↵
  config/package-lists/desktop.list.chroot
$ echo debian-installer-launcher && config/package-lists/<↵
  installer.list.chroot
```

Ora si possono cercare i task appropriati:

```
$ grep dctrl -FTTest-lang de /usr/share/tasksel/descs/debian-<↵
  tasks.desc -sTask
Task: german
```

Note that we have included the `debian-installer-launcher` package to launch the installer from the live desktop. 893

Con questo comando, si è chiaramente scoperto che il task si chiama `german`. Ora per trovare i task correlati:

```
$ grep dctrl -FEnhances german /usr/share/tasksel/descs/<↵
  debian-tasks.desc -sTask
Task: german-desktop
Task: german-kde-desktop
```

Durante il boot verrà generata la localizzazione `de`CH.UTF-8` e selezionato il layout di tastiera `*-ch`, mettiamo ora insieme questi pezzi. Ricordando che i metapacchetti task iniziano con `task-` (come descritto in [Usare metapacchetti](#)), specifichiamo questi parametri d'avvio per la lingua, quindi aggiungiamo i pacchetti con priorità standard e tutti i metapacchetti task al nostro elenco in questo modo:

```
$ mkdir live-gnome-ch
$ cd live-gnome-ch
$ lb config "
  --bootappend-live "boot=live components locales=de`CH.<↵
  UTF-8 keyboard-layouts=ch" "
```

Appendice

Style guide

17. Style guide

17.1 Guidelines for authors

This section deals with some general considerations to be taken into account when writing technical documentation for live-manual. They are divided into linguistic features and recommended procedures.

Note: Authors should first read [Contributing to this document](#)

17.1.1 Linguistic features

Use plain English

Keep in mind that a high percentage of your readers are not native speakers of English. So as a general rule try to use short, meaningful sentences, followed by a full stop.

This does not mean that you have to use a simplistic, naive style. It is a suggestion to try to avoid, as much as possible, complex subordinate sentences that make the text difficult to understand for non-native speakers of English.

Variety of English

The most widely spread varieties of English are British and American so it is very likely that most authors will use either one or the other. In a collaborative environment, the ideal variety would be International English but it is very difficult, not to say impossible, to decide on which variety among all the existing ones, is the best to use.

We expect that different varieties may mix without creating misunderstandings but in general terms you should try to be coherent and before deciding on using British, American or any other English flavour at your discretion, please take a look at how other people write and try to imitate them.

Be balanced

Do not be biased. Avoid including references to ideologies completely unrelated to live-manual. Technical writing should be as neutral as possible. It is in the very nature of scientific writing.

Be politically correct

Try to avoid sexist language as much as possible. If you need to make references to the third person singular preferably use they rather than he or she or awkward inventions such as s/he, s(he) and the like.

Be concise

Go straight to the point and do not wander around aimlessly. Give as much information as necessary but do not give more information than necessary, this is to say, do not explain unnecessary details. Your readers are intelligent. Presume some previous knowledge on their part.

Minimize translation work

Keep in mind that whatever you write will have to be translated into several other languages. This implies that a number of people will have to do an extra work if you add useless or redundant information.

Be coherent

As suggested before, it is almost impossible to standardize a collaborative document into a perfectly unified whole. However, every

effort on your side to write in a coherent way with the rest of the authors will be appreciated.

Be cohesive

Use as many text-forming devices as necessary to make your text cohesive and unambiguous. (Text-forming devices are linguistic markers such as connectors).

Be descriptive

It is preferable to describe the point in one or several paragraphs than merely using a number of sentences in a typical changelog style. Describe it! Your readers will appreciate it.

Dictionary

Look up the meaning of words in a dictionary or encyclopedia if you do not know how to express certain concepts in English. But keep in mind that a dictionary can either be your best friend or can turn into your worst enemy if you do not know how to use it correctly.

English has the largest vocabulary that exists (with over one million words). Many of these words are borrowings from other languages. When looking up the meaning of words in a bilingual dictionary the tendency of a non-native speaker of English is to choose the one that sounds more similar in their mother tongue. This often turns into an excessively formal discourse which does not sound quite natural in English.

As a general rule, if a concept can be expressed using different synonyms, it is a good advice to choose the first word proposed by the dictionary. If in doubt, choosing words of Germanic origin (Usually monosyllabic words) is often the right thing to do. Be warned that these two techniques might produce a rather informal discourse but at least your choice of words will be of wide use and generally accepted.

Using a dictionary of collocations is recommended. They are extremely helpful when it comes to know which words usually occur together.

Again it is a good practice to learn from the work of others. Using a search engine to check how other authors use certain expressions may help a lot.

False friends, idioms and other idiomatic expressions

Watch out for false friends. No matter how proficient you are in a foreign language you cannot help falling from time to time in the trap of the so called false friends, words that look similar in two languages but whose meanings or uses might be completely different.

Try to avoid idioms as much as possible. Idioms are expressions that may convey a completely different meaning from what their individual words seem to mean. Sometimes, idioms might be difficult to understand even for native speakers of English!

Avoid slang, abbreviations, contractions...

Even though you are encouraged to use plain, everyday English, technical writing belongs to the formal register of the language.

Try to avoid slang, unusual abbreviations that are difficult to understand and above all contractions that try to imitate the spoken language. Not to mention typical irc and family friendly expressions.

17.1.2 Procedures

Test before write

It is important that authors test their examples before adding them

to live-manual to ensure that everything works as described. Testing on a clean chroot or VM can be a good starting point. Besides, it would be ideal if the tests were then carried out on different machines with different hardware to spot possible problems that may arise.

Examples

When providing an example try to be as specific as you can. An example is, after all, just an example.

It is often better to use a line that only applies to a specific case than using abstractions that may confuse your readers. In this case you can provide a brief explanation of the effects of the proposed example.

There may be some exceptions when the example suggests using some potentially dangerous commands that, if misused, may cause data loss or other similar undesirable effects. In this case you should provide a thorough explanation of the possible side effects.

External links

Links to external sites should only be used when the information on those sites is crucial when it comes to understanding a special point. Even so, try to use links to external sites as sparsely as possible. Internet links are likely to change from time to time resulting in broken links and leaving your arguments in an incomplete state.

Besides, people who read the manual offline will not have the chance to follow those links.

Avoid branding and things that violate the license under which the manual is published

Try to avoid branding as much as possible. Keep in mind that other downstream projects might make use of the documentation you write. So you are complicating things for them if you add certain specific material.

live-manual is licensed under the GNU GPL. This has a number of implications that apply to the distribution of the material (of any kind, including copyrighted graphics or logos) that is published with it.

Write a first draft, revise, edit, improve, redo if necessary

- Brainstorm!. You need to organize your ideas first in a logical sequence of events.

- Once you have somehow organized those ideas in your mind write a first draft.

- Revise grammar, syntax and spelling. Keep in mind that the proper names of the releases, such as trixie or sid , should not be capitalized when referred to as code names. In order to check the spelling you can run the spell target. i.e. make spell

- Improve your statements and redo any part if necessary.

Chapters

Use the conventional numbering system for chapters and subtitles. e.g. 1, 1.1, 1.1.1, 1.1.2 ... 1.2, 1.2.1, 1.2.2 ... 2, 2.1 ... and so on. See markup below.

If you have to enumerate a series of steps or stages in your description, you can also use ordinal numbers: First, second, third ... or First, Then, After that, Finally ... Alternatively you can use bulleted items.

Markup

And last but not least, live-manual uses [SiSU](#) to process the text files and produce a multiple format output. It is recommended to take a look at [SiSU's manual](#) to get familiar with its markup, or else type:

```
$ sisu --help markup
```

Here are some markup examples that may prove useful:

- For emphasis/bold text:

```
*-foo"* or !-foo"!
```

produces: **foo** or **foo** . Use it to emphasize certain key words.

- For italics:

```
/-foo"/
```

produces: *foo*. Use them e.g. for the names of Debian packages.

- For monospace:

```
#-foo"#
```

produces: `foo`. Use it e.g. for the names of commands. And also to highlight some key words or things like paths.

- For code blocks:

```
code-
$ foo
# bar
"code
```

produces:

```
$ foo
# bar
```

Use `code-` to open and `"code` to close the tags. It is important to remember to leave a space at the beginning of each line of code.

17.2 Guidelines for translators

This section deals with some general considerations to be taken into account when translating the contents of live-manual.

As a general recommendation, translators should have read and understood the translation rules that apply to their specific languages. Usually, translation groups and mailing lists provide information on how to produce translated work that complies with Debian quality standards.

Note: Translators should also read [Contributing to this document](#). In particular the section [Translation](#)

17.2.1 Translation hints

Comments

The role of the translator is to convey as faithfully as possible the meaning of words, sentences, paragraphs and texts as written by the original authors into their target language.

So they should refrain from adding personal comments or extra bits of information of their own. If they want to add a comment for other translators working on the same documents, they can leave it in the space reserved for that. That is, the header of the strings in the po files preceded by a number sign `#` . Most graphical translation programs can automatically handle those types of comments.

TN, Translator's Note

981	It is perfectly acceptable however, to include a word or an expression in brackets in the translated text if, and only if, that makes the meaning of a difficult word or expression clearer to the reader. Inside the brackets the translator should make evident that the addition was theirs using the abbreviation TN or Translator's Note.	Newlines	990
		The translated texts need to have the exact same newlines as the original texts. Be careful to press the Enter key or type <code>\n</code> if they appear in the original files. These newlines often appear, for instance, in the code blocks.	991
982	Impersonal sentences	Make no mistake, this does not mean that the translated text needs to have the same length as the English version. That is nearly impossible.	992
983	Documents written in English make an extensive use of the impersonal form you. In some other languages that do not share this characteristic, this might give the false impression that the original texts are directly addressing the reader when they are actually not doing so. Translators must be aware of that fact and reflect it in their language as accurately as possible.	Untranslatable strings	993
		Translators should never translate:	994
		- The code names of releases (which should be written in lower-case)	995
984	False friends	- The names of programs	996
985	The trap of false friends explained before especially applies to translators. Double check the meaning of suspicious false friends if in doubt.	- The commands given as examples	997
		- Metadata (often between colons :metadata:)	998
986	Markup	- Links	999
987	Translators working initially with pot files and later on with po files will find many markup features in the strings. They can translate the text anyway, as long as it is translatable, but it is extremely important that they use exactly the same markup as the original English version.	- Paths	1000
988	Code blocks		
989	Even though the code blocks are usually untranslatable, including them in the translation is the only way to score a 100% complete translation. And even though it means more work at first because it might require the intervention of the translators if the code changes, it is the best way, in the long run, to identify what has already been translated and what has not when checking the integrity of the .po files.		

SiSU Metadata, document information

Titolo: Debian Live Manual

Autore: Debian Live Project `jdebian-live@lists.debian.org`

Diritti del lettore: Copyright: Copyright (C) 2006-2015 Live Systems Project,
Copyright (C) 2016-2025 The Debian Live team

License: This program is free software: you can redistribute it and/or modify it under
the terms of the GNU General Public License as published by the Free Software
Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
for more details.

You should have received a copy of the GNU General Public License along
with this program. If not, see <http://www.gnu.org/licenses/>.

The complete text of the GNU General Public License can be found in `/usr/-
share/common-licenses/GPL-3` file.

Casa editrice: Debian Live Project `jdebian-live@lists.debian.org`

Data: 2025-02-26

Version Information

Sorgente: `live-manual.ssm.sst`

Filetype: SiSU text 2.0, Unicode text, UTF-8 text, with very long lines (745)

Source Digest: SHA2-256(`live-manual.ssm.sst`)=`f07f50085bcd7b94523015b6-
a898f924e414d57e8927d1f3a03791ebffc19aa6`

Generated

Data di ultima generazione (ao metaverse): 2025-02-27 00:00:20 +0000

Generato da: SiSU 7.3.0 of 2023w44/1 (2023-10-30)

Ruby versione: ruby 3.3.7 (2025-01-15 revision `be31f993d7`) [x86_64-linux-gnu]