

# 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;](mailto: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	Uytownik	10
Debian Live Manual	i Instalacja	11
O tym podrzniku	3	3. Instalacja
O tym podrzniku	4	3.1 Wymagania . . . . .
1. O tym podrzniku	4	3.2 Instalowanie live-build . . . . .
1.1 Dla niecierpliwych . . . . .	4	3.2.1 Z repozytorium Debiana . . . . .
1.2 Definicje . . . . .	4	3.2.2 Ze rda . . . . .
1.3 Autorzy . . . . .	5	3.3 Instalowanie live-boot i live-config . . . . .
1.4 Wnoszenie wkadu do tego dokumentu . . . . .	6	3.3.1 Z repozytorium Debiana . . . . .
1.4.1 Nanoszenie zmian . . . . .	6	3.3.2 Ze rda . . . . .
1.4.2 Tumaczenie . . . . .	7	Podstawy
O Debian Live Project	8	4. Podstawy
2. O Debian Live Project	8	4.1 Co to jest system live? . . . . .
2.1 Motywacja . . . . .	8	4.2 Pobieranie prekompilowanych obrazw . . . . .
2.1.1 Co jest nie tak w moim dotychczasowym systemie live? . . . . .	8	4.3 Pierwsze kroki: budowanie obrazu ISO-hybrydy . . . . .
2.1.2 Czemu tworzy nasz wasny system live? . . . . .	8	4.4 Korzystanie z hybrydowego obrazu ISO live . . . . .
2.2 Filozofia . . . . .	8	4.4.1 Wypalanie obrazu ISO na fizycznym noniku . . . . .
2.2.1 Only unchanged packages from Debian main and non-free-firmware . . . . .	8	4.4.2 Kopiowanie obrazu ISO-hybrydy na nonik USB . . . . .
2.2.2 Bez konfiguracji pakietw systemu live . . . . .	9	4.4.3 Wykorzystanie przestrzeni pozostaej na noniku USB . . . . .
2.3 Kontakt . . . . .	9	4.4.4 Uruchamianie nonika live . . . . .
		4.5 Uywanie wirtualnej maszyny do testowania . . . . .
		4.5.1 Testowanie obrazu ISO z uyciem QEMU . . . . .
		4.5.2 Testowanie obrazu ISO z uyciem VirtualBox'a . . . . .
		4.6 Budowanie i uywanie obrazu HDD . . . . .
		4.7 Budowanie obrazu netboot . . . . .
		4.7.1 Serwer DHCP . . . . .

4.7.2 Serwer TFTP . . . . .	20	Dostosowywanie zawartosci . . . . .	26
4.7.3 Serwer NFS . . . . .	20	7. Opis dostosowywania . . . . .	26
4.7.4 Netboot testing HowTo . . . . .	20	7.1 Konfiguracja podczas kompilacji vs. podczas uruchamiania systemu . . . . .	26
4.7.5 Qemu . . . . .	20	7.2 Etapy kompilacji . . . . .	26
4.8 Webbooting . . . . .	21	7.3 Uzupenienie lb config plikami . . . . .	26
4.8.1 Getting the webboot files . . . . .	21	7.4 Zadania dostosowywania . . . . .	27
4.8.2 Uruchamianie obrazow webboot . . . . .	21		
Przegld narzdzi . . . . .	22	Dostosowywanie instalacji pakietow . . . . .	28
5. Przegld narzdzi . . . . .	22	8. Dostosowywanie instalacji pakietow . . . . .	28
5.1 Pakiet live-build . . . . .	22	8.1 rda pakietu . . . . .	28
5.1.1 Polecenie lb config . . . . .	22	8.1.1 Dystrybucja, dziaay archiwum i tryb . . . . .	28
5.2 Wcicia . . . . .	22	8.1.2 Serwery lustrzane dystrybucji . . . . .	29
5.2.1 Polecenie lb build . . . . .	23	8.1.3 Serwery lustrzane dystrybucji uywane podczas budowania obrazu . . . . .	29
5.2.2 Polecenie lb clean . . . . .	23	8.1.4 Serwery lustrzane dystrybucji uyte podczas uruchomienia . . . . .	29
5.3 Pakiet live-boot . . . . .	23	8.1.5 Dodatkowe repozytoria . . . . .	29
5.4 Pakiet live-config . . . . .	23	8.2 Wybieranie pakietow do instalacji . . . . .	30
Zarzdzanie konfiguracj . . . . .	24	8.2.1 Lista pakietow . . . . .	30
6. Zarzdzanie konfiguracj . . . . .	24	8.2.2 Uywanie metapakietow . . . . .	30
6.1 Radzenie sobie ze zmianami konfiguracji . . . . .	24	8.2.3 Lokalna lista pakietow . . . . .	31
6.1.1 Czemu uywa automatycznych skryptow? Co one robi? . . . . .	24	8.2.4 Lokalna lista pakietow binarnych . . . . .	31
6.1.2 Uyj przykadowych automatycznych skryptow . . . . .	24	8.2.5 Wygenerowana lista pakietow . . . . .	31
6.2 Klonowanie konfiguracji opublikowanej przez Git . . . . .	25	8.2.6 Uywanie instrukcji warunkowych w listach pakietow. . . . .	31
		8.2.7 Usuwanie pakietu podczas instalacji . . . . .	32
		8.2.8 Summary . . . . .	32
		8.2.9 Pulpit i zadania jzykowe . . . . .	32
		8.2.10 Rodzaj jdra i wersja . . . . .	33

8.2.11 Niestandardowe jdra . . . . .	33	Dostosowywanie zdarze podczas uruchamiania systemu . . . . .	40
8.3 Instalowanie zmodyfikowanych pakietw lub pakietw innych firm . . . . .	34	10. Dostosowywanie zdarze podczas uruchamiania systemu . . . . .	40
8.3.1 Uywanie packages.chrootdo instalacji niestandardowych pakietw . . . . .	34	10.1 Personalizacja uytkownika live . . . . .	40
8.3.2 Uywanie repozytorium APT aby zainstalowa niestandardowe pakiety . . . . .	34	10.2 Ustawianie lokalizacji i jzyka . . . . .	40
8.3.3 Niestandardowe pakiety i APT . . . . .	35	10.3 Persistence . . . . .	41
8.4 Konfigurowanie APT podczas kompilacji . . . . .	35	10.3.1 Plik persistence.conf . . . . .	43
8.4.1 Wybieranie apt lub aptitude . . . . .	35	10.3.2 Uywanie wiecej ni jednego magazynu persistence . . . . .	43
8.4.2 Uywanie serwera proxy z APT . . . . .	36	10.3.3 Using persistence with encryption . . . . .	44
8.4.3 Podkrcanie APT celu zaoszczdzenia miejsca . . . . .	36	Dostosowywanie obrazu binarnego . . . . .	46
8.4.4 Przekazywanie opcji do apt lub aptitude . . . . .	36	11. Dostosowywanie obrazu binarnego . . . . .	46
8.4.5 Pinning APT . . . . .	37	11.1 Programy adujce (ang. Bootloadery) . . . . .	46
Dostosowywanie zawartoci . . . . .	38	11.2 Metadane ISO . . . . .	46
9. Dostosowywanie zawartoci . . . . .	38	Dostosowywanie Instalatora Debiana . . . . .	47
9.1 Uwzglndnianie . . . . .	38	12. Dostosowywanie Instalatora Debiana . . . . .	47
9.1.1 Lokalnie uwzglndniane w chroot/live . . . . .	38	12.1 Typy Instalatora Debiana . . . . .	47
9.1.2 Lokalnie uwzglndniane dane binarne . . . . .	38	12.2 Dostosowywanie Instalatora Debiana przez preseedng . . . . .	48
9.2 Haki . . . . .	39	12.3 Dostosowywanie zawartoci Instalatora Debiana . . . . .	48
9.2.1 Chroot local hooks . . . . .	39	Projekt . . . . .	49
9.2.2 Lokalne haki binarne . . . . .	39		
9.2.3 Haki podczas uruchamiania . . . . .	39		
9.3 Wstpne ustawienie pyta Debconfa (Preseeding) . . . . .	39		

Wnoszenie wkładu do tego projektu	50	Przykłady	58
13. Wnoszenie wkładu do tego projektu	50		
13.1 Translation of man pages	50	Przykłady	59
Zgłaszanie błądów	51	16. Przykłady	59
14. Zgłaszanie błądów	51	16.1 Używanie przykładów	59
14.1 Znane problemy	51	16.2 Samouczek 1: Domyślny obraz	59
14.2 Spróbuj wykonać parę kroków	51	16.3 Samouczek 2: Narzędzie przeglądania	60
14.3 Przebuduj od zera	51	16.4 Samouczek 3: Spersonalizowany obraz	60
14.4 Używaj aktualnych pakietów	52	16.4.1 Pierwsza zmiana	60
14.5 Zbierz potrzebne informacje	52	16.4.2 Druga zmiana	61
14.6 Wyizoluj prawdopodobne wady, jeśli to możliwe	52	16.5 Kiosk-klient serwera VNC	62
14.7 Wybierz odpowiedni pakiet dla którego zgłaszasz błąd	53	16.6 A minimal image for a 512MB USB key	62
14.7.1 W czasie budowania podczas adowania początkowego (bootstrapping)	53	16.7 Pulpit GNOME w lokalnym języku oraz instalator	63
14.7.2 W czasie budowania podczas instalacji pakietów	53		
14.7.3 W czasie uruchamiania	53	Dodatek	65
14.7.4 W czasie gdy system jest już uruchomiony	53	Przewodnik redakcyjny	66
14.8 Gdzie zgłasza błąd	53	17. Przewodnik redakcyjny	66
Styl Kodowania	55	17.1 Wytyczne dla autorów	66
15. Styl Kodowania	55	17.1.1 Funkcje językowe	66
15.1 Kompatybilność	55	17.1.2 Procedury	67
15.2 Wcześnie	55	17.2 Wytyczne dla tłumaczy	69
15.3 Zawijanie	55	17.2.1 Wskazówki tłumaczenia	69
15.4 Zmienne	56	SiSU Metadata, document information	71
15.5 Różne	57		

---

<sub>1</sub> Debian Live Manual

---

<sub>2</sub> O tym podręczniku



# O tym podręczniku

## 1. O tym podręczniku

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/>

Podczas, gdy podręcznik live-manual skupia się przede wszystkim na pomocy w budowaniu systemu live, a nie na tematach użytkownika koczowego. To użytkownik koczowy również może znaleźć przydatne informacje w następujących sekcjach: **Podstawy** obejmuje pobieranie skompilowanych obrazów i przygotowanie obrazów tak aby były uruchamiane z nośnika przenośnego lub z sieci, równocześnie używanie web builder lub uruchamianie live-build bezpośrednio w systemie. **Dostosowywanie zachowania w czasie działania systemu** opisuje kilka opcji, które mogą zostać określone podczas startu, na przykład wybieranie układu klawiatury i ustawienia regionalne, używając opcji persistence.

Niektóre z poleceń zawartych w tekście muszą być wykonywane z uprawnieniami superużytkownika, które mogą być uzyskane przez stanie się użytkownikiem root poprzez su lub używając sudo. Aby odrzucić te polecenia, które mogą być wykonywane przez nieuprzywilejowanego użytkownika i te wymagające praw administratora, polecenia zostaną poprzedzone przez odpowiednio by \$ or # . Symbol ten nie jest częścią polecenia.

### 1.1 Dla niecierpliwych

Wierzmy, że wszystko w tym podręczniku jest ważne, przynajmniej dla niektórych z naszych użytkowników. Zdajemy sobie sprawę również, że zawiera

on dużo materiału do sprostania. I może chciałoby się dowiedzieć szybkich postępów w używaniu oprogramowania przed zagłębieniem się w szczegóły. Dlatego sugerujemy czytanie w następującej kolejności.

Najpierw przeczytaj rozdział, **O tym podręczniku**, od początku, kończąc na sekcji **Warunki**. Następnie przejdź do trzech więcej na początku sekcji **Przykłady** mającej na celu nauczyć Cię podstaw budowania obrazu i dostosowywania. Najpierw przeczytaj **Używanie przykładów**, następnie **Tutorial 1: Domyślny obraz**, **Tutorial 2: Narzędzie przeglądania** i wreszcie **Tutorial 3: Spersonalizowany obraz**. Pod koniec tych tutoriali, będziesz mieć ogólny zarys tego, co można zrobić z systemami live.

Zachcemy do powrotu i bardziej dogłębnej analizy podręcznika, by możemy następnie razem czytać **Podstawy**, przejrzeć lub pominąć **Budowanie obrazu netboot**, a skoczywszy czytać **Odmienienie dostosowywania** i rozdziałów, które po nim następują. W tym momencie, mamy nadzieję, że zostaniesz zainteresowany tym, co można zrobić z systemami live i zmotywowany, aby przeczytać resztę tego podręcznika, od deski do deski.

## 1.2 Definicje

System live : System operacyjny, który można uruchomić bez instalacji na dysku twardym. Systemy live nie zmieniają lokalnego systemu(-w) lub pliku(-w) już zainstalowanego na dysku twardym komputera, chyba że zostało to specjalnie ustawione. Systemy live są zwykle uruchamiane z nośników takich jak płyty CD, DVD lub pamięci USB. Niektóre mogą się także uruchamiać z sieci (za pośrednictwem obrazów netboot, patrz **Budowanie obrazu netboot**) i przez Internet (za pomocą parametru startowego fetch=URL, patrz **Webbooting**). webbooting).

Nonik live : W odróżnieniu od systemu live, nonik live odnosi się do dysku CD, DVD lub pamięci USB, gdzie znajdują się pliki binarne stworzone przez live-build, które są używane do uruchamiania systemu live. Szerzej,

termin ten odnosi si rwnie do kadego miejsca, w ktym znajduj si te pliki binarne, ktre s potrzebne do uruchomienia systemu live, rwnie takich miejsc jak miejsca dla plikw startowych w sieci.

Debian Live Project : Projekt, ktory dostarcza, midzy innymi pakiety: live-boot, live-build, live-config, live-tools i live-manual.

System hosta : rodowisko uyte do stworzenia systemu live.

System docelowy : rodowisko uyte do uruchomienia systemu live.

live-boot : Zbir skryptw wykorzystywanych do uruchamiania systemw live.

live-build : Zbir skryptw wykorzystywanych do budowy niestandardowych systemw live.

live-config : Zbir skryptw uywane do konfiguracji systemu live w czasie procesu bootowania.

live-tools : Zbir dodatkowych skryptw wykorzystywanych do wykonywania poytecznych zada w ramach uruchomionego systemu live.

live-manual : Dokument ten jest tworzony w pakiecie o nazwie live-manual.

Debian Installer (d-i) : Oficjalny system instalacyjny dystrybucji Debian.

Parametry startowe : Parametry, ktre mog by wprowadzone w wierszu bootloadera, aby wpyn na jdro lub live-config.

chroot : Program chroot, chroot(8), pozwala na uruchomienie rnych instancji rodowiska GNU / Linux na jednym systemie bez ponownego uruchomienia go.

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

Dystrybucja docelowa : dystrybucja, na ktrej opiera si bdzie system ywo. To moe si rni od dystrybucji systemu hosta.

stable/testing/unstable : Dystrybucja stable , obecna nazwa kodowa to bookworm , zawiera najnowsze oficjalnie wydanie dystrybucji Debian. Dystrybucja testing , tymczasowo nadana nazwa kodowa to trixie , to obszar postou przed nastpnym wydaniem stable . Gwn zalet korzystania z tej dystrybucji jest to, e zawiera nowsze wersje oprogramowania w stosunku do wydania stable . Dystrybucja unstable , trwale nazwana sid , to ta gdzie zachodzi aktywny rozwj Debian. Oglne rzecz biorc, to dystrybucja prowadzona jest przez deweloperw i tych, ktrzy lubi ycie na krawdzi. W tym podrzniku, mamy tendencj do korzystania z nazw kodowych wyda, takich jak trixie lub sid , bo gwnie w tych wydaniach jest zawarte to co aktualnie zawieraj narzdzia same w sobie.

### 1.3 Autorzy

Lista Autorw (w kolejnoci alfabetycznej):

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

z jednego z szybko korygujących skrtw podczas zamieszczania nowej dokumentacji, ktr dodali do podręcznika angielskiego. Uywanie PROOF=1 tworzy live-manual w formacie HTML, ale bez posegmentowanych plikw HTML, a przy uyciu PROOF=2 tworzy si live-manual w formacie PDF, ale tylko sformatowane jako A4 i listowy pionowy. Dlatego korzystajc z jednej z moliwoci opcji PROOF= mona zaoszczdzi sporo czasu, np.:

## 1.4 Wnoszenie wkadu do tego dokumentu

53

Intencj niniejszy podręcznik jest dzianie jako projekt spoeczny, a wic wszelkie propozycje usprawnie i zmian s bardzo mile widziane. Prosz przejrzj sekcj **Przyczynianie si do projektu** w celu uzyskania szczegowych informacji na temat sposobu pobrania klucza do wysyiania zmian i jak wnosi dobre zmiany.

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

Gdy zatwierdzasz jedno z tumacze moliwe jest zbudowanie tylko dla jednego jzyka, przez wykonanie, np.:

54

```
$ make build LANGUAGES=pl
```

55

### 1.4.1 Nanoszenie zmian

W celu dokonania zmian w podręczniku angielskim musisz edytowa odpowiednie pliki w manual/en/, ale przed zoeniem swojego wkadu, nalej przejrze swoj prac. Aby wywietli podgld live-manual, upewnij si, e pakiety potrzebne do budowy to s zainstalowane przez wykonanie:

Jest te moliwe, aby zbudowa po typie dokumentu, np:

56

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

57

Lub kombinacja obu, np:

58

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

59

Moesz zbudowa live-manual z gwnego katalogu swojego zapytania GIT przez wykonanie:

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

```
$ make build
```

Po rewizji swojej pracy i upewnieniu si, e wszystko jest w porzdku, nie naley uywa make commit chyba aktualizujesz ju istniejce tumaczenia, i w tym przypadku, nie naley miesza zmian do instrukcji angielskiej i tumacze w tej samej wysanej zmianie, ale naley uy osobnych zgosze zmian dla kadego tumaczenia. Zobacz sekcj **Tumaczenie**, aby uzyska wiecej szczegw.

60

Poniewa zbudowanie podręcznika we wszystkich obsugiwanych jzykach zajmuje troch czasu, autorzy mog zdecydowa, e wygodniej bdzie skorzysta

### 1.4.2 Tłumaczenie

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

W celu przetłumaczenia live-manual, wykonaj następujące kroki, w zależności od tego, czy rozpoczynasz tłumaczenie od zera czy też kontynuujesz pracę na już istniejącym tłumaczeniu:

Rozpocznij nowe tłumaczenie od zera

Przetłumacz pliki `about`manual.ssi.pot`, `about`project.ssi.pot` i `index.html.in.pot` w `manual/pot/` na swój język używając swojego ulubionego edytora (np. `poedit`) i wylij przetłumaczony plik `.po` do listy mailingowej, aby sprawdzić ich integralność. Sprawdzanie integralności live-manual sprawdza, nie tylko czy pliki `.po` są w 100% przetłumaczone, ale również wykrywa ewentualne błędy.

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

Kontynuuj pracę z już rozpoczętym tłumaczeniem

Jeli Twój język docelowy został już dodany, można losowo kontynuować tłumaczenia pozostałych plików `.po` w `manual/po/-${LANGUAGE}/` za pomocą dowolnego edytora (np. `poedit`).

Nie zapomnij, że musisz uruchomić `make commit` w celu zapewnienia, że przetłumaczone podręczniki są aktualizowane z plików `.po` i że możesz przeglądać swoje zmiany uruchamiając `make build` przed `git add`, następnie `git commit -m Translating...` (Tłumaczenie...) i `git push`. Pamiętaj, że polecenie `make build` może zająć dużo czasu, możesz wybrać

indywidualnie korygowane języki jak wyjaśniono w [Zatwierdzanie zmian](#)

Po uruchomieniu `make commit` zobaczysz jaki przewijający się tekst. Są to przede wszystkim informacyjne komunikaty o statusie przetwarzania, a także niektóre wskazówki na temat tego, co mogłoby być zrobione, aby poprawić live-manual. Chyba, że widzisz błąd krytyczny, zazwyczaj w takim wypadku możesz kontynuować i wyśłać swój wkład w tłumaczenie.

live-manual składa się z dwóch narzędzi, które mogą w znacznym stopniu pomóc tłumaczom znaleźć nieprzetłumaczone i zmienione ciągi znaków. Pierwszy z nich jest `make translate`. Uruchamia skrypt, który mówi dokładnie ile nie przetłumaczonych ciągów jest w każdym pliku `.po`. Drugi, `make fixfuzzy`, działa tylko na zmienionych ciągach znaków, ale to pomaga znaleźć je i edytować jeden po drugim.

Należy pamiętać, że nawet jeśli te narzędzia mogą być bardzo pomocne do pracy z tłumaczeniami w linii poleceń, to korzystanie z wyspecjalizowanego narzędzia jak `poedit` jest zalecanym sposobem, aby wykonać zadanie. Jest to również dobry pomysł, aby zapoznać się z dokumentacjami Debian o lokalizacjach (l10n) i tymi specyficznymi dla live-manual: [Poradnik dla tłumaczy](#).

Uwaga: Możesz użyć `make clean` aby oczyścić swoje drzewo git przed zamieszczeniem. Ten krok nie jest obowiązkowy, dzięki plikowi `.gitignore`, ale dobrą praktyką jest unikanie zatwierdzania plików odruchowo.

74	O Debian Live Project	87	2.1.2 Czemu tworzy nasz wasny system live?	
		88	Debian to Uniwersalny system operacyjny: Debian bdzie posiada system live do przetestowania i dokadnego zaprezentowania systemu Debian z nastpujnymi gwnymi zaletami:	
75	2. O Debian Live Project		Bdzie pod-projektem Debian.	89
76	2.1 Motywacja		Bdzie odzwierciedla stan aktualnej dystrybucji.	90
77	2.1.1 Co jest nie tak w moim dotychczasowym systemie live?		Bdzie wspiera tak wiele architektur jak to tylko moliwe.	91
78	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:		Bdzie si skada tylko z niezmiennionych pakietw Debian.	92
			Nie bdzie zawiera adnych pakietw, ktre nie s w archiwum Debian.	93
			Bdzie korzysta z niezmiennionego jdro Debian bez dodatkowych poprawek.	94
79	Nie s to projekty Debian i dlatego nie posiadaj take wsparcia ze strony Debian.		2.2 Filozofia	95
80	Miesza rzne dystrybucje, np.: testing i unstable .		2.2.1 Only unchanged packages from Debian main and non-free-firmware	96
81	Wspieraj tylko i386.		Bdziemy uywa tylko pakietw z repozytorium Debian w sekcji main.	97
82	Modyfikuj zachowanie i/lub wygld pakietw przez obcinanie ich w celu zaoszczdzenia miejsca.		Sekcja non-free nie jest czci Debian, a zatem nie moe by uywana do budowania oficjalnych obrazw systemu live.	
83	Zawieraj pakiety z poza archiwum Debian.		Starting with Debian 12 bookworm we added the <a href="#">non-free-firmware</a> section for better support of modern hardware.	98
84	* S dostarczane z jdrem systemu zawierajcym dodatkowe aty, ktre nie s czci Debian.		Nie bdziemy zmienia adnych pakietw. Zawsze, gdy bdziemy musieli co zmienić, zrobimy to w porozumieniu z opiekunem tego pakietu w Debianie.	99
85	S due i powolne a ze wzgldu na swój zwyk wielko, a zatem nie nadaj si do zagadnie odzyskiwania.		W drodze wyjtku, nasze wasne pakiety, takie jak live-boot, live-build lub live-config mog by zastosowane tymczasowo z wasnego repozytorium z przyczyn rozwojowych (np. do tworzenia zrzutw rozwojowych). Zostan one przesane do Debian na biego.	100
86	Nie s one dostpne na rnych nonikach, np. CD, DVD, USB-stick czy obrazy netboot.			

### 2.2.2 Bez konfiguracji pakietów systemu live

Na tym etapie nie będziemy dostarczać lub też instalować przykładowych lub alternatywnych konfiguracji pakietów. Wszystkie pakiety będą użyte w ich podstawowych konfiguracjach takich jakich są po normalnej instalacji Debian.

Za każdym razem, gdy potrzebujemy innej domyślnej konfiguracji, zrobimy to w porozumieniu z opiekunem pakietu w Debianie.

System do konfigurowania pakietów jest dostarczony przez użycie `debconf'a`; umożliwiając instalację niestandardowo skonfigurowanych pakietów w Twoim niestandardowo stworzonym obrazie systemu live. A dla **prekompilowanych obrazów live** zdecydowaliśmy, aby pozostawić pakiety w swojej domyślnej konfiguracji, chyba że będzie to absolutnie niezbędne do pracy w środowisku live. Wszędzie tam, gdzie to możliwe, wolimy dostosowywać pakiety w archiwum Debian, aby lepiej pracowały w systemie live w porównaniu do dokonywania zmian w live toolchain lub –konfiguracji obrazu prekompilowanego” # klonuj-konfiguracja-przez-git. Aby uzyskać więcej informacji, zobacz **Omnienie dostosowywania**.

zawiera informacje o błądach zgłoszonych przez użytkowników i deweloperów. Każdy błąd ma przypisany swój numer i jest przechowywany, dopóki nie zostanie on oznaczony jako rozwiązany. Aby uzyskać więcej informacji, zobacz **Zgłaszanie błędów**.

## 2.3 Kontakt

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

IRC : Wielu użytkowników i deweloperów jest obecnych na kanale `#debian-live` na `irc.debian.org` (OFTC). Kiedy zadajesz pytanie na IRC, prosimy o cierpliwie czekać na odpowiedź. W przypadku, gdy odpowiedź nie pojawi się, napisz na list mailingowy.

BTS : –Debian Bug Tracking System” <https://www.debian.org/Bugs/> (BTS)

---

# Uytkownik

# Instalacja

Jeli używasz Debiana, zalecanym sposobem jest zainstalowanie live-build 127  
poprzez repozytorium Debiana.

### 3. Instalacja

### 3.2.1 Z repozytorium Debiana

### 3.1 Wymagania

Zwyczajnie zainstaluj live-build jak kad inn paczk: 129

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

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

Dostęp do konta (root) super-użytkownika

Aktualna wersja live-build

Powoka zgodna z POSIX, taka jak bash lub dash

3.2.2 Ze rda 131

debootstrap

live-build jest opracowana z wykorzystaniem systemu kontroli wersji Git. 132  
W systemach opartych na Debianie, jest on dostarczany przez pakiet git.  
Aby sprawdzić najnowszy kod, wykonaj:

Linux 2.6 or newer

A mount point with dev and exec rights.

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

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

Naley pamita, e uycie Debiana lub dystrybucji pochodzcej od Debian nie jest wymagane - live-build bdzie dziaaa na prawie kadej dystrybucji speniajcej powysze wymagania.

Możesz zbudować i zainstalować własną paczkę Debiana wykonując:

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

### 3.2 Instalowanie live-build

Możesz zainstalować live-build na wiele różnych sposobów:

Teraz zainstaluj ktrkolwiek wiewo zbudowan paczk #-deb", wedle wyboru, 136  
np.

# Z repozytorium Debiana

Ze rda

## Ze zrzutw deweloperskich

```
# dpkg -i live-build_4.0-1_all.deb
```



Możesz również zainstalować live-build bezpośrednio w swoim systemie wykonując:

```
# make install
```

i odinstalować go wykonując:

```
# make uninstall
```

### 3.3 Instalowanie live-boot i live-config

Uwaga: Nie musisz instalować live-boot lub live-config w systemie do tworzenia niestandardowych systemów ywych. Jednak ten sposób nie zaszkodzi i jest przydatny do celów porównawczych. Jeśli chcesz tylko przejrzeć dokumentację, możesz zainstalować pakiety live-boot-doc i live-config-doc oddzielnie.

#### 3.3.1 Z repozytorium Debiana

Zarówno live-boot oraz live-config są dostępne w repozytorium Debiana, tak jak w [Instalacji live-build](#).

#### 3.3.2 Ze rda

Aby używać najnowszych rde z repozytorium GIT, użyj poniższego polecenia. Proszę upewnić się, że zapoznałeś się z warunkami wymienionymi w [Warunkach](#).

Sklonuj rda live-boot i live-config

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

Porad się podręcznikiem man pakiet live-boot i live-config aby uzyskać szczegółowe informacje na temat dostosowywania, jeżeli to jest Twój powód do budowania tych pakietów ze rde.

Zbuduj pliki .deb live-boot i live-config

Musisz budować obraz albo na dystrybucji docelowej lub w środowisku chroot zawierającym platformę docelową: oznacza to, czy celem jest trixie to obraz należy budować na trixie .

Używaj osobistych konstruktorów takich jak pbuilder lub sbuilder, jeżeli istnieje potrzeba zbudowania live-boot na dystrybucji docelowej, która różni się od systemu budowania. Na przykład, dla obrazów trixie live, zbuduj live-boot w środowisku chroot trixie . Jeśli dystrybucja docelowa zgadza się z dystrybucją systemu kompilacji, można wtedy zbudować bezpośrednio na wbudowanym systemie, używając dpkg-buildpackage (dostarczanego przez pakiet dpkg-dev) :

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

Użyj mających wygenerowanych plików .deb

Przez to, że live-boot i live-config są instalowane przez system live-build, instalacja pakietów w systemie gospodarza nie jest wystarczająca: należy traktować wygenerowane pliki deb jak inne pakiety niestandardowe.. Ponieważ z reguły celem budowania ze rda jest testowanie nowych rzeczy w krótkim okresie przed oficjalną premierą, poinformuj się [Instalowanie zmodyfikowanych paczek innych firm](#), aby tymczasowo umieścić odpowiednie pliki

w konfiguracji. W szczeglnoci nalezy zauway, e oba pakiety s podzielone na rodzajowe czci, cz dokumentacji i jeden lub wiecej czci dodatkowych. Obejmuj cz rodzajow, tylko jeden back-end (cz dodatkowa) dopasowana do konfiguracji i ewentualnie cz dokumentacji. Zakadajc, e budujesz obraz live w biecym katalogu i wszystkie wygenerowane paczki .deb dla pojedynczej wersji obu pakietw znajduj si w katalogu powyzej, te polecenia bash skopiuj wszystkie odpowiednie pakiety, w tym domylne dla nich back-endy:

157

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

---

# Podstawy

## 4. Podstawy

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.

Throughout the chapter, we will often refer to the default filenames produced by live-build. If you are **downloading a prebuilt image** instead, the actual filenames may vary.

### 4.1 Co to jest system live?

A live system usually means an operating system booted on a computer from a removable medium, such as a CD-ROM or USB stick, or from a network, ready to use without any installation on the usual drive(s), with auto-configuration done at run time (see **Terms**).

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:

Obraz jdra Linuxa , zazwyczaj nazwany vmlinuz\*

Initial RAM disk image (initrd) : a RAM disk set up for the Linux boot, containing modules possibly needed to mount the System image and some scripts to do it.

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 : A small piece of code crafted to boot from the chosen medium, possibly presenting a prompt or menu to allow selection of options/configuration. It loads the Linux kernel and its initrd to run with an associated system filesystem. Different solutions can be used, depending on the target medium and format of the filesystem containing the previously mentioned components: isolinux to boot from a CD or DVD in ISO9660 format, syslinux for HDD or USB drive booting from a VFAT partition, extlinux for ext2/3/4 and btrfs partitions, pxelinux for PXE netboot, GRUB for ext2/3/4 partitions, etc.

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 Pobieranie prekompilowanych obrazów

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**.

### 4.3 Pierwsze kroki: budowanie obrazu ISO-hybrydy

Regardless of the image type, you will need to perform the same basic steps to build an image each time. As a first example, create a build directory, change to that directory and then execute the following sequence of live-build commands to create a basic ISO hybrid image containing a default live system without X.org. It is suitable for burning to CD or DVD media, and also to copy onto a USB stick.

The name of the working directory is absolutely up to you, but if you take a look at the examples used throughout live-manual, it is a good idea to use a name that helps you identify the image you are working with in each directory, especially if you are working or experimenting with different image types. In this case you are going to build a default system so let’s call it, for example, 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
```

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.

Now that the config/ hierarchy exists, build the image with the lb build command:

```
# lb build
```

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 Korzystanie z hybrydowego obrazu ISO live

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 Wypalanie obrazu ISO na fizycznym nośniku

Burning an ISO image is easy. Just install xorriso and use it from the command-line to burn the image. For instance:

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

#### 4.4.2 Kopiowanie obrazu ISO-hybrydy na nonik 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 Wykorzystanie przestrzeni pozostaej na noniku 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"
```

After the partition is created, where `$-PARTITION`" is the name of the

partition, such as `/dev/sdb2`, you have to create a filesystem on it. One possible choice would be ext4.

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

Note: If you want to use the extra space with Windows, apparently that OS cannot normally access any partitions but the first. Some solutions to this problem have been discussed on our [mailing list](#), but it seems there are no easy answers.

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 Uruchamianie nonika live

The first time you boot your live medium, whether CD, DVD, USB key, or PXE boot, some setup in your computer's BIOS may be needed first. Since BIOSes vary greatly in features and key bindings, we cannot get into the topic in depth here. Some BIOSes provide a key to bring up a menu of boot devices at boot time, which is the easiest way if it is available on your system. Otherwise, you need to enter the BIOS configuration menu and change the boot order to place the boot device for the live system before your normal boot device.

Once you've booted the medium, you are presented with a boot menu. If you just press enter here, the system will boot using the default entry, Live and default options. For more information about boot options, see the help entry in the menu and also the live-boot and live-config man pages found within the live system.

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.

examples. The qemu-utils package is also valuable for creating virtual disk images with qemu-img.

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

## 4.5 Uywanie wirtualnej maszyny do testowania

It can be a great time-saver for the development of live images to run them in a virtual machine (VM). This is not without its caveats:

Running a VM requires enough RAM for both the guest OS and the host and a CPU with hardware support for virtualization is recommended.

There are some inherent limitations to running on a VM, e.g. poor video performance, limited choice of emulated hardware.

When developing for specific hardware, there is no substitute for running on the hardware itself.

Occasionally there are bugs that relate only to running in a VM. When in doubt, test your image directly on the hardware.

Provided you can work within these constraints, survey the available VM software and choose one that is suitable for your needs.

Uruchamianie obrazu ISO jest proste:

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

Zobacz podrzchniki man, aby uzyska więcej szczegw.

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 Testowanie obrazu ISO z uyciem VirtualBox'a

W celu przetestowania ISO w VirtualBox'ie:

```
# 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.

Note: For live systems containing X.org that you want to test with virtualbox, you may wish to include the VirtualBox X.org driver package, virtualbox-guest-dkms and virtualbox-guest-x11, in your live-build configuration. Otherwise, the resolution is limited to 800x600.

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

In order to make the dkms package work, also the kernel headers for the kernel flavour used in your image need to be installed. Instead of manually listing the correct linux-headers package in above created package list, the selection of the right package can be done automatically by live-build.

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

## 4.6 Budowanie i uywanie obrazu 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.

Note: if you created an ISO hybrid image with the previous example, you will need to clean up your working directory with the lb clean command (see [The lb clean command](#)):

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

Run the lb config command as before, except this time specifying the HDD image type:

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

A teraz zbuduj obraz uywajc polecenia 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 Budowanie obrazu netboot

The following sequence of commands will create a basic netboot image

containing a default live system without X.org. It is suitable for booting over the network.

Note: if you performed any previous examples, you will need to clean up your working directory with the `lb clean` command:

```
# 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.

Run the `lb config` command as follows to configure your image for netbooting:

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

In contrast with the ISO and HDD images, netbooting does not, itself, serve the filesystem image to the client, so the files must be served via NFS. Different network filesystems can be chosen through `lb config`. The `--net-root-path` and `--net-root-server` options specify the location and server, respectively, of the NFS server where the filesystem image will be located at boot time. Make sure these are set to suitable values for your network and server.

A teraz zbuduj obraz uywajc polecenia `lb build`:

```
# lb build
```

In a network boot, the client runs a small piece of software which usually resides on the EPROM of the Ethernet card. This program sends a DHCP request to get an IP address and information about what to do next. Typically, the next step is getting a higher level bootloader via the TFTP protocol. That could be `pxelinux`, `GRUB`, or even boot directly to an operating system like 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

We must configure our network's DHCP server to be sure to give an IP address to the netbooting client system, and to advertise the location of the PXE bootloader.

Here is an example for inspiration, written for the ISC DHCP server `isc-dhcp-server` in the `/etc/dhcp/dhcpd.conf` configuration file:

```
# /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;
"
```

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

and tell the NFS server about this new export with the following command:

```
# 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.

## 4.7.2 Serwer TFTP

This serves the kernel and initial ramdisk to the system at run time.

You should install the tftpd-hpa package. It can serve all files contained inside a root directory, usually /srv/tftp. To let it serve files inside /srv/debian-live/tftpboot, run as root the following command:

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

and fill in the new tftp server directory when being asked about it.

## 4.7.3 Serwer NFS

Once the guest computer has downloaded and booted a Linux kernel and loaded its initrd, it will try to mount the Live filesystem image through a NFS server.

Musisz zainstalowa pakiet nfs-kernel-server.

Then, make the filesystem image available through NFS by adding a line like the following to /etc/exports:

## 4.7.4 Netboot testing HowTo

Netboot image creation is made easy with live-build, but testing the images on physical machines can be really time consuming.

Aby uatwi sobie ycie moemy uy wirtualizacji.

## 4.7.5 Qemu

Zainstaluj qemu, bridge-utils, sudo.

Edytuj /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
```

Zainstaluj, lub zbuduj grub-floppy-netboot.

Uruchom qemu z -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 bootloader, 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 Uruchamianie obrazw webboot

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 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.

# Przegląd narzędzi

## 5. Przegląd narzędzi

Ten rozdział zawiera przegląd trzech głównych narzędzi stosowanych w budowie systemów live: live-build, live-boot i live-config.

### 5.1 Pakiet live-build

Live-build to zbiór skryptów do budowania systemów live. Skrypty te są również określane jako polecenia.

Pomyślem stojącym za live-build jest bycie oparą, która używa struktury katalogów jako konfiguracji, aby całkowicie zautomatyzować i dostosować wszystkie aspekty budowania obrazu live.

Wiele jest podobnych do tych używanych do budowania pakietów Debiana z użyciem debhelper'a:

Skrypty posiadają centralną lokalizację dla konfiguracji ich działania. Dla debhelper'a jest to podkatalog drzewa pakietów debian/. Na przykład, dh\_install będzie szukał, spośród innych, pliku o nazwie debian/install do określenia, które pliki powinny zawierać się w określonym pakiecie binarnym. W taki sam sposób, live-build przechowuje swoją konfigurację w caoci w podkatalogu config/.

Skrypty są niezależne - to znaczy, że zawsze jest bezpieczne uruchomienie poszczególnych poleceń.

W przeciwieństwie do debhelpera, live-build zapewnia narzędzia do generowania szkieletu katalogów konfiguracyjnych. Może to być uznane za podobne do narzędzi takich jak dh-make. Aby uzyskać więcej informacji na temat tych narzędzi, kontynuuj czytanie, ponieważ pozostaje jeszcze tego

rozdziału omawia cztery najważniejsze polecenia. Należy zauważyć, że głównym wrapperem dla polecenia live-build jest lb.

lb config : Odpowiedzialny za inicjowanie katalogu konfiguracji systemu live. Zobacz **Polecenie lb config**, aby uzyskać więcej informacji.

lb build : Odpowiedzialny za rozpoczęcie kompilacji systemu live. Zobacz **polecenie lb build** aby uzyskać więcej informacji.

lb clean : Odpowiedzialny za czyszczenie kompilacji systemu live. Zobacz **polecenie lb clean** aby uzyskać więcej informacji.

#### 5.1.1 Polecenie lb config

Jak omówiono w **live-build**, skrypty, które składają się na live-build czytają swoją konfigurację przy użyciu polecenia source z katalogu o nazwie config/. Budowanie tego katalogu rzadziej byłoby czasochłonne i podatne na błąd, polecenie lb config może być używane do tworzenia początkowej konfiguracji drzewa katalogów.

Wykonanie lb config bez żadnych argumentów tworzy podkatalog config, w którym zapisane są niektóre domyślne ustawienia, w plikach konfiguracyjnych, oraz dwa szkielety drzew o nazwach auto/ i #-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...
```

#### 5.2 Wcześniejsze

Normalnie, pewnie będziesz chciał określić niektóre opcje. Na przykład,

aby okreli, jakiego menadiera pakierw chcesz uy podczas budowania obrazu:

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

Jest moliwe ustalenie wielu opcji, takich jak:

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

Pena lista opcji dostpna jest w podręczniku man pakietu lb'config.

### 5.2.1 Polecenie lb build

Polecenie lb build czyta konfiguracj z katalogu config/. A nastpnie uruchamia polecenia niszego poziomu potrzebne do budowy Twojego systemu live.

### 5.2.2 Polecenie lb clean

Zadaniem polecenia lb clean, jest to aby usun rne czci kompilacji tak aby mona byo zacz od czystego stanu. Domylnie etapy chroot, binary and source s sprztane, ale cache pozostaje nienaruszone. Ponadto, tylko poszczególne etapy mog by oczyszczane. Na przykad, jeli zostay wprowadzone zmiany, ktre wpywaj tylko na etap binarny, naley uy lb clean --binary przed budowaniem nowych plikw binarnych. Jeli zmiany uniewaniaj proces bootstrap i/lub zmieniaj cache pakietw, np. po zmianie opcji --mode, --architecture, lub --bootstrap, trzeba uy lb clean --purge. Zobacz podręcznik man pakietu lb'clean aby uzyska list wszystkich opcji.

## 5.3 Pakiet live-boot

live-boot to zbir skryptw zapewniajcych haki do initramfs-tools, wykonywane do wytwarzania plikw initramfs, ktre s w stanie uruchomi system live, takich jak te stworzone przez live-build. Obejmuje to obrazy ISO systemw live, archiwa netboot i obrazw dysku USB.

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.

## 5.4 Pakiet live-config

live-config zawiera skrypty, ktre s uruchamiane przy starcie systemu live po live-boot, tak aby automatycznie skonfigurowa system live. Obsuguje on takie zadania jak ustawienie nazwy hosta, lokalizacji i strefy czasowej, tworzenie uytkownika live, zatrzymywanie zada crona i autologowanie uytkownika live.

# Zarządzanie konfiguracją

## 6. Zarządzanie konfiguracją

Ten rozdział wyjaśnia, jak zarządza konfiguracją live od początku jej tworzenia, przez kolejne zmiany i kolejne wersje oprogramowania live-build i obrazu live.

### 6.1 Radzenie sobie ze zmianami konfiguracji

Konfiguracje live rzadko są idealne na pierwszy próbie. Powinno się dodać opcje `lb config` z linii poleceń do wykonania pojedynczej kompilacji, ale bardziej typowe jest sprawdzenie tych opcji i kompilowanie ponownie, aby uzyskać satysfakcję. Aby poradzić sobie z tymi zmianami, potrzeba automatycznych skryptów, które zapewnią, że konfiguracja przechowywana jest w stanie spójnym.

#### 6.1.1 Czemu używa automatycznych skryptów? Co one robi?

Polecenie `lb config` zapisuje opcje, które są wprowadzane do plików w `config/*#` oraz wielu innym opcjom przypisuje wartości domyślne. Jeśli uruchomisz `#-lb config` ponownie, nie skasuje to żadnych opcji, która została zapisana na podstawie początkowych opcji. Tak więc, na przykład, jeśli by uruchomić `lb config` ponownie z nową wartością dla `-binary-images`, wszelkie opcje zależne, które zostały przypisane jako domyślne dla poprzedniej dystrybucji mogą już nie współgrać z nowym ustawieniem. Pliki te nie są przeznaczone do odczytu lub edycji. Przechowuj one wartości dla ponad stu opcji, więc nikt, nie mówiąc już o robieniu tego w pojedynkę, nie będzie mógł zobaczyć, które z tych opcji są faktycznie przypisane. I wreszcie, po uruchomieniu `lb config`, a następnie uaktualnieniu live-build, a zdarza się, że zmieniają się nazwy

opcji, `config/*` nadal będzie zawierał zmienne nazwane po starym, które nie są już aktualne.

Z tych wszystkich powodów, skrypty `auto/*` czyni Twoje życie łatwiejszym. Są proste wrappery do poleceń `lb config`, `lb build` i `lb clean`, które są zaprojektowane, aby pomóc w zarządzaniu konfiguracją. Skrypt `auto/config` przechowuje to polecenie `lb config` ze wszystkimi podanymi opcjami, skrypt `auto/clean` usuwa pliki zawierające wartości zmiennych konfiguracyjnych, a skrypt `auto/build` zachowuje log `build.log` każdej kompilacji. Kiedy z tych scenariuszy jest uruchamiany automatycznie przy każdym uruchomieniu odpowiedniego polecenia `lb`. Korzystając z tych skryptów, konfiguracja jest bardziej czytelna i jest przechowywana w sposób wewnętrznie spójny z jedną wersją do następnej. Ponadto, będzie o wiele łatwiej zidentyfikować opcje, które należy zmienić po uaktualnieniu live-build i po przeczytaniu dokumentacji aktualizacji.

#### 6.1.2 Użyj przykładowych automatycznych skryptów

Dla Twojej wygody, live-build jest dostarczany z przykładowymi skryptami powłoki do automatycznego kopiowania i edycji. Rozpocznij nową, domyślną konfigurację, a następnie skopiuj do niej przykłady:

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

Edytuj `auto/config`, dodając wszelkie opcje, jakie uważasz. Przykładowo:

```
#!/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/ "
"$-@"
```

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

340 Teraz, za każdym razem kiedy korzystasz z `lb config`, `auto/config` skasuje konfigurację w oparciu o te opcje. Gdy chcesz wprowadzić zmiany do nich, należy edytować opcje w tym pliku zamiast przekazywać je do `lb config`. Podczas korzystania z `lb clean`, `auto/clean` oczyszczy pliki w `config/*` wraz z innymi produktami kompilacji. I wreszcie, kiedy używasz `lb build`, log kompilacji zostanie zapisany przez `auto/build` w `build.log`.

346 Edytuj `auto/config` i wszelkie inne rzeczy, których wymagasz do własnych potrzeb w drzewie katalogów `config`. Na przykład, nieoficjalne prekompilowane obrazy tworzy się dodając po prostu `--archive-areas main contrib non-free`.

347 Opcjonalnie można zdefiniować skrót w konfiguracji Git przez dodanie następujących opcji do `$HOME/.gitconfig`:

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

341 Uwaga: Specjalny parametr `noauto` jest użyty tutaj, aby powstrzymać kolejne zapytania do `auto/config`, zapobiegając w ten sposób nieskończonej rekurencji. Upewnij się, że przypadkowo nie została usunięta podczas dokonywania zmiany. Również należy zadbać o to, aby podczas dzielenia polecenia `lb config` na wiele linii dla czytelności, jak pokazano w powyższym przykładzie, nie zapomnieć odwrotnego ukośnika (na końcu każdej linii, która kontynuuje polecenie w następnej linii).

348 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
```

## 342 6.2 Klonowanie konfiguracji opublikowanej przez Git

343 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**.

351 Klonowanie całego repozytorium `live-images` (obrazów live) służy konfiguracji używanej dla kilku różnych obrazów. Jeśli masz ochotę na budowę innego obrazu po zakończeniu pracy z pierwszym, przejdź do innego katalogu i ponownie w miarę potrzeb dokonaj zmian.

352 W każdym przypadku należy pamiętać, że za każdym razem trzeba będzie budować obraz jako super-użytkownik: `lb build`

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

```
$ mkdir live-images && cd live-images
```

# Dostosowywanie zawartoci

## 7. Opis dostosowywania

Ten rozdzia zawiera przegld rnych sposobw, w jaki mona dostosowa system live.

### 7.1 Konfiguracja podczas kompilacji vs. podczas uruchamiania systemu

Opcje konfiguracji systemu live s podzielone na opcje w czasie budowania, ktre s stosowane w czasie kompilacji i na opcje w czasie rozruchu, ktre s stosowane podczas uruchamiania systemu. Opcje podczas uruchamiania s podzielone na te wystpujce wczenie podczas uruchamiania, zastosowane przez live-boot, i na te, wystpujce pniej, zastosowane przez live-config. Kady parametr rozruchu moe zosta zmodyfikowany przez uytkownika poprzez ustalenie go podczas startu. Obraz moe by rwnie zbudowany z domylnymi parametrami startowymi, dziki czemu uytkownicy mog normalnie tylko uruchomi bezporednio systemu live bez podawania adnych parametrw, gdy wszystkie opcje domylne s odpowiednie. W szczeglnoci argument `lb --bootappend-live` skada si z wszelkich opcji wiersza polece domylnych dla jdra systemu live, takich jak `trwao` (ang. `persistence`), ukad klawiatury, lub strefa czasowa. Zobacz [Dostosowywanie lokalizacji i jzyka](#), dla przykadw.

Opcje konfiguracyjne w czasie budowania s opisane w podrzniku `man` na stronie `lb config`. Opcje konfiguracyjne w czasie rozruchu opisane s w podrzniku `man` na stronach `live-boot` i `live-config`. Chocia pakiety startowe `live-boot` i `live-config` s zainstalowane w systemie live, ktry budujesz, zaleca si rwnie zainstalowa je w systemie budowania dla atwego odniesienia podczas Twojej pracy przy konfiguracji. Jest to bezpieczne, poniewa

aden z zawartych w nich skryptw nie bdzie wykonywany, chyba e system zostanie skonfigurowany jako system live.

### 7.2 Etapy kompilacji

Proces kompilacji jest podzielony na etapy, w kadym z nich z zastosowanymi w kolejnoci rnymi dostosowaniami. Pierwszym etapem do uruchomienia jest etap `bootstrap`. Jest to wstpna faza wypeniania katalogu `chroot` pakietami aby stworzy kadub systemu Debian. Nastpnym etapem jest `chroot`, ktry koczy budow katalogu `chroot`, wypeniania go wszystkimi pakietami wymienionymi w konfiguracji, wraz z innymi materiaami. Najwcej dostosowywania zawartoci odbywa si w tym etapie. Ostatnim etapem przygotowania obrazu live jest etap binarny (ang. `binary`), ktry tworzy moliwy do uruchomienia obraz, uywajc zawartoci katalogu `chroot` do budowy gwnego systemu plikw w systemie live, a tym instalatora i wszelkich innych dodatkowych materiaw na noniku docelowym poza system plikw na systemie live. Po skompilowaniu obrazu live, jeli wczo, archiwum `rdowe tarball` jest budowane podczas etapu `source` (ang. `rdo`).

W kadym z tych etapw istnieje szczeglna sekwencja, w ktrej stosuje si polecenia. S one usytuowane w taki sposb, aby zapewni modyfikacjom bycie uoonym w rozsdy sposb. Na przykad, w etapie `chroot`, prekonfiguracja (ang. `preseeding`) jest stosowana, zanim zostan zainstalowane jakiegokolwiek pakiety, pakiety s instalowane zanim jakiegokolwiek lokalnie zawarte pliki zostan kopiowane, a haki s wprowadzane pniej, gdy wszystkie materiay s ju na miejscu.

### 7.3 Uzupenie lb config plikami

Mimo, e `lb config` tworzy konfiguracj katalogw w `config/`, aby osign swoje

cele, moe by konieczne udostpnienie dodatkowych plikw w podkatalogach config/. W zalenoci od tego, gdzie pliki s przechowywane w konfiguracji, mog by skopiowane do systemu plikw systemu live lub do binarnego obrazu systemu plikw, lub mog zosta zapewnione konfiguracje w czasie budowy systemu, ktre byoby kopotliwie do przekazania jako opcje wiersza polecenia. Mona zawrze rzeczy takie jak niestandardowe listy pakietw, niestandardowa grafika lub inny skrypt do uruchomienia zarwno w czasie kompilacji jak i w czasie startu systemu, zwikszajc ju znaczn elastyczno debian-live swoim wasnych kodem.

#### 364 7.4 Zadania dostosowywania

365 Kolejne rozdziaiy s podzielone na rodzaje zada dostosowywania, ktry uytkownicy zazwyczaj wykonuj: **Dostosowywanie instalacji pakietu**, **Dostosowywanie zawartoci** i **Dostosowywanie ustawie regionalnych i jzyka** obejmuj tylko niektre z rzeczy, ktre moesz zrobi.



# Dostosowywanie instalacji pakietów

## 8. Dostosowywanie instalacji pakietów

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 rda pakietu

#### 8.1.1 Dystrybucja, dziay archiwum i tryb

The distribution you choose has the broadest impact on which packages are available to include in your live image. Specify the codename, 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
```

Within the distribution archive, archive areas are major divisions of the archive. In Debian, these are main, contrib and non-free. Only main contains software that is part of the Debian distribution, hence that is the default. One or more values may be specified, e.g.

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

Experimental support is available for some Debian derivatives through a `--mode` option. By default, this option is set to debian only if you are building on a Debian or on an unknown system. If `lb config` is invoked on any of the supported derivatives, it will default to create an image of that derivative. If `lb config` is run in e.g. ubuntu mode, the distribution names and archive areas for the specified derivative are supported instead of the ones for Debian. The mode also modifies live-build behaviour to suit the derivatives.

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 Serwery lustrzane dystrybucji

The Debian archive is replicated across a large network of mirrors around the world so that people in each region can choose a nearby mirror for best download speed. Each of the `--mirror-*` options governs which distribution mirror is used at various stages of the build. Recall from [Stages of the build](#) that the `bootstrap` stage is when the `chroot` is initially populated by `debootstrap` with a minimal system, and the `chroot` stage is when the `chroot` used to construct the live system's filesystem is built. Thus, the corresponding mirror switches are used for those stages, and later, in the `binary` stage, the `--mirror-binary` and `--mirror-binary-security` values are used, superseding any mirrors used in an earlier stage.

### 8.1.3 Serwery lustrzane dystrybucji uywane podczas budowania obrazu

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/
```

Serwer lustrzany `chroot`, określony przez `--mirror-chroot`, domylnie do wartoci `--mirror-bootstrap`.

### 8.1.4 Serwery lustrzane dystrybucji uyte podczas uruchomienia

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 Dodatkowe repozytoria

You may add more repositories, broadening your package choices beyond what is available in your target distribution. These may be, for example, for backports, experimental or custom packages. To configure additional repositories, create `config/archives/your-repository.list.chroot`, and/or `config/archives/your-repository.list.binary` files. As with the `--mirror-*` options, these govern the repositories used in the `chroot` stage when building the image, and in the `binary` stage, i.e. for use when running the live system.

For example, `config/archives/live.list.chroot` allows you to install packages from the `debian-live` snapshot repository at live system build time.

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

If you add the same line to `config/archives/live.list.binary`, the repository will be added to your live system's `/etc/apt/sources.list.d/` directory.

Jeeli takie pliki istniej to zostan wybrane automatycznie.

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

Should you need custom APT pinning, such APT preferences snippets can be placed in `config/archives/your-repository.pref.-binary,chrout` files and will be automatically added to your live system's `/etc/apt/preferences.d/` directory.

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

## 8.2 Wybieranie pakietw do instalacji

There are a number of ways to choose which packages live-build will install in your image, covering a variety of different needs. You can simply name individual packages to install in a package list. You can also use metapackages in those lists, or select them using package control file fields. And finally, you may place package files in your `config/` tree, which is well suited to testing of new or experimental packages before they are available from a repository.

### 8.2.1 Lista pakietw

Listy pakietw s skutecznym sposobem wyraenia, ktre pakiety powinny zosta zainstalowane. Skadnia list obsuguje instrukcje warunkowe, ktre uatwiaj budowanie list i dostosowywanie ich do wykorzystania w wielu

konfiguracjach. Nazwy pakietw mog by take wstrzykiwane do listy za pomoc pomocnikw powoki w czasie kompilacji.

Note: The behaviour of live-build when specifying a package that does not exist is determined by your choice of APT utility. See [Choosing apt or aptitude](#) for more details.

### 8.2.2 Uywanie metapakietw

Najprostszym sposobem, aby wypeni swoj list pakietw jest uycie metapakietu zada dostarczanych przez dystrybucj. Na przykad:

```
$ lb config
$ echo task-gnome-desktop & config/package-lists/desktop.list.<↵
  chrout
```

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.

All task metapackages are prefixed task-, so a quick way to determine which are available (though it may contain a handful of false hits that match the name but aren't metapackages) is to match on the package name with:

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

Oprcz tych, znajdziesz jeszcze inne metapakiety do rnych celw. Niektre s

podzbiorami szerszych pakietów, jak `gnome-core`, podczas gdy inne są indywidualne specjalistyczne części czystego Debiana mieszanki, takie jak metapakiety `education-*`. Aby wyświetlić wszystkie metapakiety w archiwum, należy zainstalować pakiet `debtags` i wyświetlić wszystkie pakiety z tagiem `role::metapackage` w następujący sposób:

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

### 8.2.3 Lokalna lista pakietów

Whether you list metapackages, individual packages, or a combination of both, all local package lists are stored in `config/package-lists/`. Since more than one list can be used, this lends itself well to modular designs. For example, you may decide to devote one list to a particular choice of desktop, another to a collection of related packages that might as easily be used on top of a different desktop. This allows you to experiment with different combinations of sets of packages with a minimum of fuss, sharing common lists between different live image projects.

Package lists that exist in this directory need to have a `.list` suffix in order to be processed, and then an additional stage suffix, `.chroot` or `.binary` to indicate which stage the list is for.

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

Note: If you don't specify the stage suffix, the list will be used for both stages. Normally, you want to specify `.list.chroot` so that the packages will only be installed in the live filesystem and not have an extra copy of the `.deb` placed on the medium.

### 8.2.4 Lokalna lista pakietów binarnych

To make a binary stage list, place a file suffixed with `.list.binary` in `config/package-lists/`. These packages are not installed in the live filesystem, but are included on the live medium under `pool/`. You would typically use such a list with one of the non-live installer variants. As mentioned above, if you want this list to be the same as your chroot stage list, simply use the `.list` suffix by itself.

### 8.2.5 Wygenerowana lista pakietów

It sometimes happens that the best way to compose a list is to generate it with a script. Any line starting with an exclamation point indicates a command to be executed within the chroot when the image is built. For example, one might include the line `! grep-aptavail -n -sPackage -FPriority standard` —sort in a package list to produce a sorted list of available packages with Priority: standard.

In fact, selecting packages with the `grep-aptavail` command (from the `dctrl-tools` package) is so useful that live-build provides a Packages helper script as a convenience. This script takes two arguments: field and pattern. Thus, you can create a list with the following contents:

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

### 8.2.6 Używanie instrukcji warunkowych w listach pakietów.

Any of the live-build configuration variables stored in `config/*` (minus the `LB` prefix) may be used in conditional statements in package lists. Generally, this means any `lb config` option uppercased and with dashes changed

to underscores. But in practice, it is only the ones that influence package selection that make sense, such as DISTRIBUTION, ARCHITECTURES or ARCHIVE`AREAS.

Na przykad, aby zainstalowa ia32-libs jeeli wybrano `-architectures amd64`:

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

Mona przetestowa dowolny szereg wartoci, na przykad zainstalowa memtest86+, jeeli ustalono `-architectures i386` lub `-architectures amd64`:

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

You may also test against variables that may contain more than one value, e.g. to install vrms if either contrib or non-free is specified via `-archive-areas`:

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

Zagniedanie instrukcji warunkowych jest nie obsugiwane.

### 8.2.7 Usuwanie pakietu podczas instalacji

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

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`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 Pulpit i zadania jzykowe

Desktop and language tasks are special cases that need some extra plan-

ning and configuration. Live images are different from Debian Installer images in this respect. In the Debian Installer, if the medium was prepared for a particular desktop environment flavour, the corresponding task will be automatically installed. Thus, there are internal gnome-desktop, kde-desktop, lxde-desktop and xfce-desktop tasks, none of which are offered in tasksel's menu. Likewise, there are no menu entries for tasks for languages, but the user's language choice during the install influences the selection of corresponding language tasks.

When developing a desktop live image, the image typically boots directly to a working desktop, the choices of both desktop and default language having been made at build time, not at run time as in the case of the Debian Installer. That's not to say that a live image couldn't be built to support multiple desktops or multiple languages and offer the user a choice, but that is not live-build's default behaviour.

Because there is no provision made automatically for language tasks, which include such things as language-specific fonts and input-method packages, if you want them, you need to specify them in your configuration. For example, a GNOME desktop image containing support for German might include these task metapackages:

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

### 8.2.10 Rodzaj jdra i wersja

One or more kernel flavours will be included in your image by default, depending on the architecture. You can choose different flavours via the

–linux-flavours option. Each flavour is suffixed to the default stub linux-image to form each metapackage name which in turn depends on an exact kernel package to be included in your image.

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 Niestandardowe jdra

You can build and include your own custom kernels, so long as they are integrated within the Debian package management system. The live-build system does not support kernels not built as .deb packages.

The proper and recommended way to deploy your own kernel packages is to follow the instructions in the kernel-handbook. Remember to modify the ABI and flavour suffixes appropriately, then include a complete build of the linux and matching linux-latest packages in your repository.

If you opt to build the kernel packages without the matching metapackages, you need to specify an appropriate –linux-packages stub as discussed in **Kernel flavour and version**. As we explain in **Installing modified or**

**third-party packages**, it is best if you include your custom kernel packages in your own repository, though the alternatives discussed in that section work as well.

It is beyond the scope of this document to give advice on how to customize your kernel. However, you must at least ensure your configuration satisfies these minimum requirements:

Uyj startowego dysku RAM.

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

Include any other filesystem modules required by your configuration (i.e. usually squashfs).

### 8.3 Instalowanie zmodyfikowanych pakietów lub pakietów innych firm

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.

Istnieją dwa sposoby instalacji niestandardowych pakietów:

packages.chroot

Używanie niestandardowego repozytorium APT

Using packages.chroot is simpler to achieve and useful for one-off customizations but has a number of drawbacks, while using a custom APT repository is more time-consuming to set up.

#### 8.3.1 Używanie packages.chroot do instalacji niestandardowych pakietów

To install a custom package, simply copy it to the config/packages.chroot/ directory. Packages that are inside this directory will be automatically installed into the live system during build - you do not need to specify them elsewhere.

Packages must be named in the prescribed way. One simple way to do this is to use dpkg-name.

Korzystanie z packages.chroot do instalacji niestandardowych pakietów ma wady:

Nie jest możliwe użycie bezpiecznego APT.

You must install all appropriate packages in the config/packages.chroot/ directory.

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

#### 8.3.2 Używanie repozytorium APT aby zainstalować niestandardowe pakiety

Unlike using packages.chroot, when using a custom APT repository you must ensure that you specify the packages elsewhere. See **Choosing packages to install** for details.

While it may seem unnecessary effort to create an APT repository to install custom packages, the infrastructure can be easily re-used at a later date to offer updates of the modified packages.

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 ;i EOF i 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 "$-PACKAGES`FROM`REPOSITORY"" i config/package-lists i custom`repo`.list$-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

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

### 8.3.3 Niestandardowe pakiety i APT

live-build uses APT to install all packages into the live system so will therefore inherit behaviours from this program. One relevant example is that (assuming a default configuration) given a package available in two

different repositories with different version numbers, APT will elect to install the package with the higher version number.

Because of this, you may wish to increment the version number in your custom packages' debian/changelog files to ensure that your modified version is installed over one in the official Debian repositories. This may also be achieved by altering the live system's APT pinning preferences - see [APT pinning](#) for more information.

## 8.4 Konfigurowanie APT podczas kompilacji

You can configure APT through a number of options applied only at build time. (APT configuration used in the running live system may be configured in the normal way for live system contents, that is, by including the appropriate configurations through config/includes.chroot/.) For a complete list, look for options starting with apt in the lb`config` man page.

### 8.4.1 Wybieranie apt lub aptitude

You can elect to use either apt or aptitude when installing packages at build time. Which utility is used is governed by the `-apt` argument to `lb config`. Choose the method implementing the preferred behaviour for package installation, the notable difference being how missing packages are handled.

apt: Uywajc tej metody, jeli zaznaczono brakujcy pakiet, instalacja pakietu zakoczy si niepowodzeniem. To jest domylne ustawienie.

aptitude: Uywajc tej metody, jeli zaznaczono brakujcy pakiet, instalacja pakietu zakoczy si powodzeniem.



### 8.4.2 Uywanie serwera proxy z APT

502

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 Podkrwanie APT celu zaoszczdzenia miejsca

You may find yourself needing to save some space on the image medium, in which case one or the other or both of the following options may be of interest.

Jeli nie chcesz zawiera indeksu APT w obrazie, mona je pomin z:

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

This will not influence the entries in `/etc/apt/sources.list`, but merely whether `/var/lib/apt` contains the indices files or not. The tradeoff is that APT needs those indices in order to operate in the live system, so before performing `apt-cache search` or `apt-get install`, for instance, the user must `apt-get update` first to create those indices.

If you find the installation of recommended packages bloats your image too much, provided you are prepared to deal with the consequences discussed below, you may disable that default option of APT with:

```
$ 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 Przekazywanie opcji do apt lub aptitude

If there is not a `lb config` option to alter APT's behaviour in the way you need, use `--apt-options` or `--aptitude-options` to pass any options through to your configured APT tool. See the man pages for `apt` and `aptitude` for details. Note that both options have default values that you will need to retain in addition to any overrides you may provide. So, for example, suppose you have included something from `snapshot.debian.org` for testing purposes and want to specify `Acquire::Check-Valid-Until=false`

to make APT happy with the stale Release file, you would do so as per the following example, appending the new option after the default value `--yes`:

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

Please check the man pages to fully understand these options and when to use them. This is an example only and should not be construed as advice to configure your image this way. This option would not be appropriate for, say, a final release of a live image.

For more complicated APT configurations involving `apt.conf` options you might want to create a `config/apt/apt.conf` file instead. See also the other `apt-*` options for a few convenient shortcuts for frequently needed options.

#### 8.4.5 Pinning APT

For background, please first read the `apt-preferences(5)` man page. APT pinning can be configured either for build time, or else for run time. For the former, create `config/archives/*.pref`, `config/archives/*.pref.chroot`, and `config/apt/preferences`. For the latter, create `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
```

Negative pin priorities will prevent a package from being installed, as in the case where you do not want a package that is recommended by another package. Suppose you are building an LXDE image using `task-lxde-desktop` in `config/package-lists/desktop.list.chroot`, but don't want the user prompted to store wifi passwords in the keyring. This metapackage depends on `lxde-core`, which recommends `gksu`, which in turn recommends `gnome-keyring`. So you want to omit the recommended `gnome-keyring` package. This can be done by adding the following stanza to `config/apt/preferences`:

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

# Dostosowywanie zawartoci

## 9. Dostosowywanie zawartoci

Ten rozdzia omawia dokadn regulacj dostosowywania zawartoci systemu poza samym wybieraniem pakietw do wyboru, ktre bd zawarte. Uwzglndianie pozwala doda lub wymieni dowolne pliki w obrazie systemu live, haki pozwalaj na wykonanie dowolnego polecenia na rnych etapach produkcji oraz w czasie rozruchu a opcji preseeding pozwala skonfigurowa pakiety, gdy s zainstalowane poprzez dostarczenie odpowiedzi na pytania debconfa.

### 9.1 Uwzglndnianie

Chocia najlepiej byoby gdyby system live zawiera si wycznie z plikw dostarczonych przez cakowicie niemodyfikowane pakiety, to czasami wygodniejszym jest, dostarczenie lub zmodyfikowanie pewnych treci za pomoc plikw. Korzystajc z uwzglndniania, mona doda (lub wymieni) dowolne pliki w systemie obrazu live. live-build dostarcza dwa takie mechanizmy wykorzystania:

Lokalne uwzglndnianie chroot: Pozwoli Ci doda lub zastpi pliki w systemie plikw chroot/live. Zobacz **Lokalne uwzglndnianie chroot/live**, aby uzyska wiecej informacji.

Lokalne uwzglndnianie danych binarnych: Pozwoli Ci doda lub zastpi pliki w obrazie binarnym. Zobacz **Lokalnych uwzglndnianie danych binarnych**, aby uzyska wiecej informacji.

Prosz zobaczy **Definicje** aby uzyska wiecej informacji na temat rnicy midzy obrazami binarnymi a obrazami live.

#### 9.1.1 Lokalnie uwzglndniane w chroot/live

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.

Aby doczy pliki, po prostu dodaj je do katalogu config/includes.chroot. Ten katalog odnosi si do katalogu root / systemu live. Na przykad, aby doda plik /var/www/index.html do systemu live, uuj:

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

Konfiguracja bdzie mie nastpujcy ukad:

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

Lokalnie uwzglndnione w chroot s instalowane po instalacji pakietw, tak e pliki zainstalowane przez pakiety s zastpowane.

#### 9.1.2 Lokalnie uwzglndniane dane binarne

Aby zawiera materiay takie jak filmy lub dokumentacja na systemie plikw nonika tak, aby byy one dostpne od razu po woeniu nonika bez uruchamiania systemu live, moesz uy lokalnego uwzglndniania danych binarnych.

Dziaa to w podobny sposb do lokalnego uwzglndniania w chroot/live. Zamy na przykad, e pliki ~/video`demo.\* to filmy demonstacyjne systemu live opisanego przez i dowizane przez stron indeksu HTML. Wystarczy skopiowa materia z config/includes.binary/ w nastpujcy sposb:

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

Pliki te pojawi si teraz w katalogu gwnym nonika live.

## 9.2 Haki

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 Lokalne haki binarne

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 Haki podczas uruchamiania

Aby wykona polecenia przy starcie systemu, moesz dostarczy haki live-config, jak wyjaniono w jego podrzniku man w sekcji Customization (ang. dostosowywanie). Przeanalizuj wasne haki live-config dostarczone w /lib/live/config/, zwracajc uwag na numery sekwencji. Nastpnie dodaj swj wasne hak z odpowiednim prefiksem numeru sekwencji, albo jako lokalnie uwzglndnione w chroot w config/includes.chroot/lib/live/config/, lub w postaci niestandardowego pakietu omwione w **Instalowanie zmodyfikowanych pakietw lub pakietw innych firm**.

## 9.3 Wstpne ustawienie pyta Debconfa (Preseeding)

Pliki w katalogu config/preseed/ z rozszerzeniem .cfg w nastpujcy etapie (.chroot lub .binary) s brane pod uwag jako pliki preseed debconfa i s instalowane za pomoc live-build przez debconf-set-selections podczas odpowiedniego etapu.

Aby uzyska wicej informacji o debconf, prosz przeczytaj debconf(7) z pakietu debconf.

# Dostosowywanie zdarze podczas uruchamiania systemu

## 10. Dostosowywanie zdarze podczas uruchamiania systemu

Caa konfiguracja, ktra odbywa si w czasie pracy systemu jest wykonywana przez live-config. Oto niektre z najbardziej popularnych opcji live-config, ktrymi mog by zainteresowani uytownicy. Pen list wszystkich moliwoci mona znale w podreczniku man pakietu live-config.

### 10.1 Personalizacja uytownika live

Jednym wanyim czynnikiem jest to, e uytownik jest tworzony przez live-boot w czasie startu systemu, a nie live-build w czasie kompilacji. To wpywa nie tylko, na to gdzie materiay dotyczce uytownika live s wprowadzone w kompilacji, jak to opisano w [Uwzglndnianie lokalne Live/chroot](#), ale rwnie na wszelkie grupy uprawnienia zwizane z uytownikiem 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"
```

lub uyj live-config.user-default-groups=audio,cdrom,dip,floppy,video,plugd

jako parametru startowego.

Moliwe jest rwnie, aby zmieni domyln nazw uytownika user i domylne haso live. Jeli chcesz to zrobi, z jakiegokolwiek powodu, mona to atwo osign w nastpujcy sposb:

Aby zmieni domyln nazw uytownika naley po prostu okreli j w konfiguracji:

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

Jednym z moliwych sposobw zmiany domylnego hasa jest uycie odpowiedniego haka, jak opisano w [Haki podczas uruchamiania systemu](#). W tym celu mona uy haka passwd z /usr/share/doc/live-config/examples/hooks, przedrostkiem jest odpowiednio (np. 2000-passwd), naley go doda do config/includes.chroot/lib/live/config/

### 10.2 Ustawianie lokalizacji i jzyka

Podczas uruchamiania systemu live, jzyk jest definiowany przez dwa etapy:

generowanie plikw lokalizacji

ustawienie konfiguracji klawiatury

Domylnie ustawieniem lokalnym podczas budowania systemu live jest locales=en`US.UTF-8. Aby okreli ustawienia regionalne, ktre powinny by wygenerowane, uyj parametru locales w opcji --bootappend-live polecenia lb config, np.

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

Wiele lokalizacji moe by okrelone w postaci listy rozdzielonej przecinkami.

Parametr ten, jak rwnie parametr konfiguracyjny klawiatury jak wskazano poniej, moe by rwnie uywany w linii polece jdra. Mona okreli ustawienia regionalne poprzez language-country (w tym przypadku uywane jest kodowanie domylne) lub penej nazwy z kodowaniem language-country.encoding. Lista obsugiwanych lokalizacji i kodowa mona znale w /usr/share/i18n/SUPPORTED.

Zarwno konfiguracja konsoli i klawiatury X wykonywana jest przez live-config przy pomocy pakietu console-setup. Aby je skonfigurowa, ustaw parametry startowe keyboard-layouts, keyboard-variants, keyboard-options i keyboard-model przez opcj --bootappend-live. Prawidowe wartoi opcje mona znale w /usr/share/X11/xkb/rules/base.lst. Aby znale ukady i warianty klawiatury dla danego jzyka, sprbuj wyszuka nazw w jzyku angielskim i/lub kraj, gdzie mwi si danym jzykiem, np.:

```
$ egrep -i '(!—german.*switzerland)' /usr/share/X11/xkb/rules/↵
base.lst
! model
! layout
! layout
  ch                German (Switzerland)
! variant
  legacy            ch: German (Switzerland, legacy)
  de-nodeadkeys     ch: German (Switzerland, eliminate dead keys)
  de-sundeadekeys   ch: German (Switzerland, Sun dead keys)
  de-mac            ch: German (Switzerland, Macintosh)
! option
```

Naley pamita, e kady wariant wymienia ukad, ktrego dotyczy w opisie.

Czsto tylko ukad klawiatury musi by skonfigurowany. Na przykad, aby uzyska list plikw lokalizacyjnych dla niemieckiego i szwajcarskiego niemieckiego ukadu klawiatury w systemie X uyj:

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

Jednak dla bardzo konkretnych przypadkw uycia, mona doda inne parametry. Na przykad, aby ustawi francusko jzyczny system z ukadem klawiatury French-Dvorak (zwany Bepo) na klawiaturze typu USB Type-Matrix EZ-Reach 2030, uyj:

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

Wiele wartoci oddzielonych przecinkami moe by przypisane do kadego z parametrw keyboard-\*, z wyjatkiem keyboard-model, ktry przyjmuje tylko jedn warto. Prosz przejrze podrncznik man keyboard(5) aby uzyska wiecej szczegw i przykadw zmiennych XKBMODEL, XKBLAYOUT, XKBVARIANT i XKBOPTIONS. Jeli podano wiele keyboard-variants (ang warianty klawiatury), bd one dopasowane jeden do drugiego przez warto keyboard-layouts (patrz opcja setxkbmap(1) -variant). Puste wartoci s dozwolone; na przykad aby zdefiniowa dwa ukady, domylny US QW-ERTY oraz drugi US Dvorak, zastosuj:

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

### 10.3 Persistence

Odmian Live CD jest preinstalowany system, ktry uruchamia si z non-

ikw tylko do odczytu, takich jak cdrom, gdzie operacje zapisu i modyfikacje nie przetrwaj restartu sprzutowych hosta, na ktrym jest uruchomiony.

System live jest uoglnieniem tego paradygmatu, a tym samym wspiera media inne prcz pyt CD; ale dalej jako jego domylne zachowanie, naley uwaa operacje tylko do odczytu a wszekie zmiany, w czasie dziaania s tracone podczas zamykania systemu.

‘Persistence’ (ang. trwaao) to wsplna nazwa dla rnych rodzajw rozwiza dla zapisania niektrych lub wszystkich danych midzy restartami podczas uywania i wprowadzania zmian do systemu. Aby zrozumie, jak to dziaa to przydatna bya by wiedza, e nawet wtedy, gdy system jest uruchamiany i dziaa z nonika tylko do odczytu, to modyfikacje plikw i katalogw s zapisywane na zapisywalnych nonikach, typowo dysk RAM (tmpfs) a dane w pamici RAM nie przetrwaj restartu.

Dane przechowywane na tym ramdysku powinny by przechowywane na zapisywalnym trwajm noniku takim jak lokalny dysk, lokalny udzia sieciowy lub nawet na sesji wielosesyjnego dysku CD-RW/DVD-RW. Wszystkie te noniki s obsugiwane w systemach live na rne sposoby i wszystkie, oprcz ostatniej wymagaj specjalnego parametru startowego okrelonego w czasie startu systemu: persistence.

Jeli ustawiony jest parametr startowy persistence (a nopersistence nie jest ustawiony), lokalne noniki (np. dyski twarde, napdy USB) bd przeszukane w celu znalezienia woluminw trwaoci podczas startu systemu. Moliwe jest ograniczenie, jakiego typu woluminy trwaoci bd wykorzystane przez okalenie pewnych parametrw startowych opisanych w podrzniku man live-boot(7). Wolumin trwaoci moe by kadym z wymienionych:

partycja, identyfikowana po nazwie GPT.

system plikw, identyfikowany po etykiecie.

plik obrazu zlokalizowany na kadym obsugiwany systemie plikw (nawet na partycji NTFS innego systemu), identyfikowany po nazwie pliku.

Etykiet dla woluminu musi by persistence, ale bdzie ignorowane, dopki nie bdzie zawarty w katalogu gwnym plik o nazwie persistence.conf, ktry jest uywany by peni dostosowa wolumin persistence, to znaczy, e wskazuje si w nim katalogi, w ktrych chcesz zapisa na woluminie zmiany przy restarcie. Zobacz [Plik persistence.conf](#), aby uzyska wiecej szczegw.

Oto kilka przykadw, jak przygotowa wolumin, aby mg by on uyty z opcj persistence. Moe to by, na przykad, partycja ext4 na dysku twardym lub na noniku wymiennym stworzona przez, np.:

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

Zobacz rwnie [Wykorzystanie przestrzeni pozostaej na noniku USB](#).

Jeli masz ju partycj na urzdzeniu, mona po prostu zmieni jego etykiet uywajc nastpujce polecenia:

```
# tune2fs -L persistence /dev/sdb1 # for ext2,3,4 filesystems
```

Oto przykad, jak stworzy plik obrazu opartego na ext4 do zastosowania z opcj persistence:

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

Po utworzeniu pliku obrazu, na przykad, aby sprawi by katalog /usr by prway, ale tylko zapisywa zmiany wprowadzone w tym katalogu, a

nie ca zawarto /usr, mona u opcji union. Jeli plik obrazu znajduje si w katalogu domowym, naley skopiowa go do katalogu gwnego systemu plikw na dysku twardym i zamontowa go w /mnt w nastpujcy sposb:

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

Nastpnie utwrx plik persistence.conf dodajc zawarto i odmontowujc plik obrazu.

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

Teraz uruchom ponownie i wybierz nonik live, a nastpnie uruchom dodajc parametr startowy persistence.

### 10.3.1 Plik persistence.conf

Partycj z etykiet persistence naley skonfigurowa za pomoc pliku persistence.conf, aby dowolne katalogi stay si trwae. Ten plik, znajdujcy si w gwnym katalogu systemu plikw partycji, kontroluje ktre katalogi s trwae i w jaki sposb.

To jak niestandardowe wierzchnie zamontowania s skonfigurowane jest opisane w szczegach w podrzniku man persistence.conf(5), ale ten prosty przykad powinien by wystarczajcy dla wikszoci zastosowa. Powiedzmy, e chcemy, aby nasz katalog domowy i cache APT byo trwae w pamici podrznej systemu plikw ext4 na partycji /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
```

Nastpnie uruchamiamy ponownie komputer. Podczas pierwszego uruchomienia zawarto /home i /var/cache/apt zostanie skopiowana do woluminu trwaoci i od tej pory wszystkie zmiany w tych katalogach bd przechowywane na tym woluminie. Naley pamita, e wszelkie ciek wymienione w pliku persistence.conf nie mog zawiera spacji lub specjalnych komponentw ciek: . i .. . Ponadto, ani /lib, /lib/live (lub ktrykolwiek z jego podkatalogw) ani / nie moe zosta utrwalony za pomoc wasnych punktww montowania. Jako obejcie tego ograniczenia mona doda / union do pliku persistence.conf w celu osignicia penej trwaoci.

### 10.3.2 Uywanie więcej ni jednego magazynu persistence

Istniej rne sposoby korzystania z wielu magazynw trwaoci (ang. persistence) dla rnych zastosowa. Na przykad, przy uywanie kilku magazynw w tym samym czasie lub wybranie tylko jednego, spord rnych, do bardzo specyficznych zastosowa.

Kilka rnych niestandardowych woluminw-nakadek (z wasnymi plikami persistence.conf) moe by uywane w tym samym czasie, ale jeeli kilka woluminw tworzy ten sam katalog trwaym, bdzie uywany tylko jeden z nich. Jeli jakie dwa punkty montowania s zagniedone (np. jeden jest podkatalogiem drugiego) to katalog parent (ang. rodzic) zostanie zamontowany przed katalogiem child (ang. dziecko) tak e jeden punkt montowania nie bdzie ukryty przed innym. Zagniedone niestandardowe woluminy s problematyczne, jeeli s wymienione w tym samym pliku persistence.conf. Zobacz podrznik man persistence.conf(5), jak radzi sobie z tym przypadkiem, jeli naprawd potrzebujesz (Wskazwka: zazwyczaj nie potrzebujesz).



Jednym z moliwych przypadkw uycia: Jeli chcesz przechowywa dane uytownika np. /home i dane superuytownika tj. /root na rnych partycjach, utwz dwie partycje z etykiet persistence i dodaj plik persistence.conf na kad z nich, tak #-# echo /home i persistence.conf na pierwszej partycji, przez co bdz zapisywane pliki uytownika i # echo /root i persistence.conf na drugiej partycji, ktra bdzie przechowywa pliki superuytownika. Wreszcie, naley uy parametru startowego persistence.

Jeli uytownik bdzie potrzebowa wiele magazynw trwaoci tego samego typu dla rnych miejsc lub dla celw testowych, takich jak magazyny private i #-work"# parametr startowy persistence-label uyty w poczeniu z parametrem persistence pozwoli na wiele unikatowych magazynw trwaoci. Przykadem moe by, jeli uytownik chciaby uy partycji trwaoci oznaczonej private dla prywatnych danych, takich jak zakadki w przeglдарce lub innych typw danych, to mgby uy parametrw startowych: #-persistence"# persistence-label=private. A do przechowywania danych zwizanych z prac, takich jak dokumenty, projekty badawcze lub inne rodzaje, mgby skorzysta z parametrw startowych: persistence persistence-label=work.

Wane jest, aby pamita, e kada z tych partycji, private i #-work"#, take potrzebuje pliku persistence.conf. Podrznik man pakietu live-boot zawiera wicej informacji o tym, jak korzysta z tych etykiet z zapisanymi nazwami.

### 10.3.3 Using persistence with encryption

Korzystanie z funkcji trwaoci (ang. persistence) oznacza, e niektre poufne dane mog zosta naraone na ryzyko. Zwazacza jeli trwae dane s przechowywane na urzdzeniu przenonym, takim jak pamici USB lub zewntrzne dyski twarde. To jest miejsce, gdzie przydatne staje si szyfrowanie. Nawet jeli caa procedura moe wydawa si skomplikowana, ze wzgldu na

liczb krokw, ktre naley podj, to jest bardzo atwo obsugiwa szyfrowane partycje z live-boot. Aby mc korzysta z luks, ktry jest obsugiwany typem szyfrowania, musisz zainstalowa cryptsetup zarwno na maszynie tworzenia zaszyfrowanych partycji, a take w systemie live, ktry bdzie uywa szyfrowanej trwaej partycji.

Aby zainstalowa cryptsetup na twoim komputerze:

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

Aby zainstalowa cryptsetup na twoim systemie live dodaj go do listy pakietw:

```
$ lb config
$ echo "cryptsetup cryptsetup-initramfs" i config/package-lists/↵
  encryption.list.chroot
```

Gdy Twój system live posiada cryptsetup, to w zasadzie wystarczy, ju tylko utworzy now partycj, zaszyfrowa j i uruchomi z parametrami persistence i persistence-encryption=luks. Moglimy ju przewidzie ten krok i doda parametry startowe w czasie kompilacji przestrzegajc nastpujcej procedury:

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

przejdmy do szczegw dla tych wszystkich, ktrzy nie s zaznajomieni z szyfrowaniem. W poniszym przykadmie mamy zamiar uy partycji na dysku USB, ktra odpowiada /dev/sdc2. Naley zaznaczy, e naley ustali, ktra partycja jest jeden t, ktr masz zamiar uywa w tym konkretnym przypadku.

Pierwszym krokiem jest podczenie dysku USB i okalenie, ktrym jest urzdzeniem. Zalecan metod tworzenia listy urzdze w live-manual jest ls -l /dev/disk/by-id. Nastpnie utworzymy now partycj, a nastpnie zaszyfrujemy j hasem w nastpujcy sposb:

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

Nastpnie otwieramy partycj LUKS w wirtualnym elemencie odwzorowujcym urzdzenia /dev/mapper . Mona tu uy dowolnej nazww. Uywamy live jako przykad:

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

Nastpnym krokiem jest wypenienie urzdzenia zerami przed utworzeniem systemu plikw:

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

Teraz jestemy gotowi do stworzenia systemu plikw. Warto zauway, e dodajemy etykiet persistence tak, aby urzdzenie zotao zamontowane w jako persistence store (magazyn persistence) w czasie startu systemu.

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

Aby kontynuowa nasz konfiguracj, musimy zamontowa urzdzenie, na przykad w /mnt.

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

12 Stwórz plik persistence.conf w katalogu gwnym partycji. To jest, jak wyjaniono wyej, absolutnie konieczne. Zobacz [Plik persistence.conf](#).

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

Potem odmontuj punkt montowania:

```
# umount /mnt
```

I opcjonalnie, cho moe to by dobry sposb na zabezpieczenie danych, ktre wanie dodalimy do partycji, moemy zamkn urzdzenie:

```
# cryptsetup luksClose live
```

Podsumujemy proces. Do tej pory stworzylimy system live z moliwoci szyfrowania, ktry mona skopiowa na nonik usb, jak wyjaniono w [kopiowaniu hybrydowego obrazu ISO na nonik pamici USB](#). Stworzylimy rwnie zaszyfrowan partycj, ktra moe znajdowa si na tym samym noniku usb, aby mona byo go nosi ze sob wszdzie i mamy skonfigurowan zaszyfrowan partycj, stosowan jako magazyn persistence. Wic teraz, musimy tylko uruchomi system live. W czasie startu systemu, na live-boot poprosi nas o wpisanie hasa i zamontuje zaszyfrowan partycj uywan przez opcj persistence.

# Dostosowywanie obrazu binarnego

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

## 11. Dostosowywanie obrazu binarnego

### 11.1 Programy adujce (ang. Bootloadery)

live-build używa syslinux i niektórych jego pochodnych (w zależności od typu obrazu) w domylnym programie ładującym (ang. bootloader). Można je łatwo dostosować do własnych potrzeb.

W celu wykorzystania pełnego motywu, skopiuj `/usr/share/live/build/bootloaders` do `config/bootloaders` i edytuj tam te pliki. Jeśli nie chcesz się martwić modyfikacją wszystkich obsługiwanych konfiguracji programu ładującego (ang. bootloader), tylko zapewnienie lokalnego zmodyfikowanego kopii jednego z typów programów, np. `* -isolinux` w `# -config / programy ładujące / isolinux` wystarczy, że, w zależności od przypadku użycia.

Podczas modyfikacji jednego z domyślnych motywów, jeśli chcesz skorzystać z indywidualnego obrazu tła, który będzie wyświetlany razem z menu startowym, dodaj obraz `splash.png` 640x480 pikseli. Następnie usuń plik `splash.svg`.

Istnieje wiele możliwości, jeśli chodzi o wprowadzanie zmian. Na przykład, pochodne syslinux mają domyślnie skonfigurowany limit czasowy (ang. timeout) na 0 (zero), co oznacza, że wstrzymaj się one na czas nieokreślony na ich ekranie powitalnym a do naciśnięcia klawisza.

Aby zmienić limit czasowy podczas rozruchu w domylnym obrazie iso-hybrid wystarczy zmienić domyślny plik `isolinux.cfg` określając limit czasu (ang. timeout) w jednostkach 1/10 sekundy. Zmodyfikowany `isolinux.cfg` uruchamiający rozruch po pięciu sekundach byłby podobny do tego:

### 11.2 Metadane ISO

Podczas tworzenia binarnego obrazu ISO9660, można korzystać z następujących opcji, aby dodać metadane tekstowe do obrazu. To może pomóc w identyfikacji wersji lub konfiguracji obrazu bez uruchamiania go.

`LB`ISO`APPLICATION/–iso-application NAZWA:` Pole to powinno opisywać zastosowanie obrazu. Maksymalna długość tego pola wynosi 128 znaków.

`LB`ISO`PREPARER/–iso-preparer NAME:` Pole to powinno opisywać przygotowującego obraz, zwykle z kilkoma danymi kontaktowymi. Domyślną wartością tej opcji jest używana wersja live-build, która może pomóc przy debugowaniu. Maksymalna długość tego pola wynosi 128 znaków.

`LB`ISO`PUBLISHER/–iso-publisher NAME:` Pole to powinno opisywać wydawcę obrazu, zwykle z kilkoma danymi kontaktowymi. Maksymalna długość tego pola wynosi 128 znaków.

`LB`ISO`VOLUME/–iso-volume NAME:` Pole to powinno opisywać ID woluminu obrazu. Jest używane jako etykieta widoczna dla użytkownika na niektórych platformach, takich jak Windows i Apple Mac OS. Długość maksymalna dla tego pola to 32 znaków.

---

# Dostosowywanie Instalatora Debiana

## 12. Dostosowywanie Instalatora Debiana

Obrazy systemów live mogą być zintegrowane z Instalatorem Debiana. Istnieje wiele różnych typów instalacji, różniących się tym, co jest zawarte w obrazie i jak działa instalator.

Proszę zwrócić uwagę na używanie wielkich liter, podczas odnoszenia się do Debian Installer (ang. Instalatora Debiana) w tej sekcji - kiedy jest stosowany tak, to odnosi się wyrażenie do oficjalnego instalatora dla systemu Debian, a nie cokolwiek innego, to jest często postrzegana w skrócie jako d-i.

### 12.1 Typy Instalatora Debiana

Trzy główne rodzaje instalatora to:

Instalator Debiana Normal : To jest normalny obraz systemu live z oddzielnym jądrem i initrd, który (gdy wybrany z odpowiedniego menu programu adjącego) uruchamia do standardowej instancji Instalatora Debiana, tak jakby pobrano obraz pyty Debiana i uruchomiono go. Obrazy zawierające system typu live i inny niezależny instalator są często określane jako połączone obrazy (ang. combined images).

Na takich obrazach, Debian jest instalowany przez pobieranie i instalowanie pakietów .deb za pomocą debootstrap, z lokalnych mediów lub jakiegoś opartego na sieci, w wyniku czego domyślny system Debian jest instalowany na dysku twardym.

Czy proces może być zapisany w pliku preseed i dostosowany na wiele

różnych sposobów, przeczytaj odpowiednie strony w instrukcji Instalatora Debiana, aby uzyskać więcej informacji. Kiedy utworzysz działający plik preseed, live-build może automatycznie umieścić go w Twoim obrazie i wczytać go dla Ciebie.

Instalator Debiana Live : To jest obraz systemu live z oddzielnym jądrem i initrd, który (gdy wybrany z odpowiedniego menu programu adjącego) uruchamia do instancji Instalatora Debiana.

Instalacja będzie przebiegać w identyczny sposób do instalacji normalnej opisanej powyżej, ale na tym samym etapie instalacji, zamiast używać debootstrap aby pobrać i zainstalować pakiety, systemu plików obrazu live zostanie skopiowany do celu. Jest to możliwe dzięki specjalnym pakietom udeb zwanym live-installer.

Po tym etapie, Instalator Debiana kontynuuje normalnie, instalując i konfigurując elementy, takie jak programy adjące i lokalnych użytkowników, itp.

Uwaga: aby zapewnić wsparcie dla wpisów zarówno normalnych i instalatora live w bootloaderze (ang. programie adjącym) na tym samym noniku live, należy wczytać instalatora live przez wpis w pliku preseed live-installer/enable=false.

Instalator Debiana Desktop : Niezależnie od wybranego rodzaju Instalatora Debiana, ten może zostać uruchomiony z pulpitu, klikając na jego ikonę. Jest bardziej przyjazny użytkownikowi w niektórych sytuacjach. W celu skorzystania z tej opcji, pakiet debian-installer-launcher musi zostać uwzględniony.

Należy pamiętać, że domyślnie, live-build nie obejmuje obrazów Instalatora Debiana w obrazach, musi to być sprecyzowane przez lb config. Również należy pamiętać, że aby działał Desktop Installer (ang. instalator uruchamiany z poziomu pulpitu) to jądro systemu live musi być zgodne z jądrem wykorzystywanym przez d-i dla określonej architektury. Na przykład:

669

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

## 670 12.2 Dostosowywanie Instalatora Debiana przez preseeding

671 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:

672

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

## 673 12.3 Dostosowywanie zawartosci Instalatora Debiana

674 Do celw dowiadczalnych lub debugowania, moesz zechcie doczy lokalnie zbudowany element di w paczce udeb. Naley umieci je w config/packages-binary/, aby wczy je do obrazu. Rwnie dodatkowe lub zamienne pliki i katalogi mog by zawarte w initrd instalatora, w sposb podobny do [Uwzgl-](#)  
[nianie lokalne live/chroot](#), poprzez umieszczenie materiaw w config/-includes.installer/.

# Projekt

---

# Wnoszenie wkładu do tego projektu

## 13. Wnoszenie wkładu do tego projektu

Wnosząc swój wkład należy jasno określić posiadacza jego praw autorskich i zawrzeć wszelkie stosowane oświadczenie licencjonowania. Należy pamiętać, aby zmiany były zaakceptowane, wkład musi być licencjonowany na tej samej licencji co reszta dokumentów, a mianowicie, GPL w wersji 3 lub nowszej.

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.

Nawet jeśli wszystkie zmiany mogą być później zweryfikowane, prosimy aby używać zdrowego rozsądku i tworzyć dobre zmiany opisane dobrym komentarzem.

Pisz komentarze do zmian, które składają się z paragrafów, sensownych zdań w języku angielskim, począwszy od dużej litery, a kończąc kropką. Zwykle te komentarze zaczynają się od Fixing/Adding/Removing/Correcting/Translating/.... (Naprawianie/Dodawanie/Usuwanie/Korygowanie/Tłumaczenie/ itd. ... .)

Pisz dobre komentarze do zmian. Pierwsza linia musi być dokładne podsumowanie treści paragrafu, która zostanie uwzględniona w liście zmian (ang. changelog). Jeśli musisz zrobić kilka wyjaśnień, napisz je nie pozostawiając pustych wierszy po pierwszej linii, a następnie kolejny pusty wiersz po każdym akapicie. Linie każdego akapitu nie powinny przekraczać 80 znaków.

\* Wysyłaj zmiany osobno. To znaczy; nie mieszaj niepowiązanych ze sobą rzeczy w tej samej zmianie. Dodaj osobno zmiany dla każdej rzeczy, którą zmieniasz.

### 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.

# Zgaszanie bdw

## 14. Zgaszanie bdw

Systemy live s dalekie od doskonaoci, ale chcemy, uczyni je jak najblisze perfekcji - z Wasz pomoc. Nie wahaj si zgosi bd. Lepiej jest wypeni raport dwa razy ni wcale. Ten rozdzia zawiera zalecenia dotyczce sposobu skadania raportw o bdach.

Dla niecierpliwych:

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.

Sprbuj poda jak najwiecej specyficznych informacji, jak to tylko moliwe o bdzie. Obejmuje to (co najmniej) wersj uywanych live-build, live-boot, live-config i live-tools oraz dystrybucj systemu live ktr kompilujesz.

### 14.1 Znane problemy

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.

Jeli budowanie systemu opartego na testing lub unstable powoduje u

Ciebie zbyt wiele trudnoci, raczej wykorzystaj wydanie stable . live-build zawsze ustawia domylnie wydanie stable .

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 Sprbuj wykona par krokw

Przed zgoseniem bdu, prosz poszukaj w internecie danego komunikatu o bdzie lub objawu jaki otrzymujesz. Poniewa jest mao prawdopodobne, e jeste jedyn osob, ktra zмага si ze szczeglnym problem. Zawsze jest szansa, e zosta on ju omwiony w innym miejscu i zaproponowano ju moliwe rozwizanie, obejcie lub patch.

Naley zwrci szczegln uwag na list mailingow systemw live, jak rwnie strona gwna, poniewa zawieraj one prawdopodobnie najbardziej aktualne informacje. Jeli takowa informacja istnieje, zawsze naley zawrze odniesienie do niej w swoim raporcie o bdzie.

Ponadto, naley sprawdzi aktualne wykazy bdw dla live-build, live-boot, live-config i live-tools, aby zobaczy, czy co podobnego nie zostao ju zgoszone.

### 14.3 Przebuduj od zera

Aby upewni si, e dany bd nie jest spowodowany nie w peni wyczyszczonym budowanym systemem, prosz zawsze odbudowa cay system live od podstaw, aby sprawdzi, czy bd jest powtarzalny.



## 14.4 Używaj aktualnych pakietów

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 Zbierz potrzebne informacje

Prosz dostarczyć wystarczającą ilość informacji z raportem. Obejmujące co najmniej, dokładną wersję live-build, gdzie napotkano błąd i kroki, jak go odtworzyć. Prosz użyć zdrowego rozsądku i przedstawić wszelkie inne istotne informacje, jeśli uważasz, że może to pomóc w rozwiązaniu problemu.

Aby w pełni wykorzystać Twój raport o błędzie, będziemy potrzebować przynajmniej następujących informacji:

Architektura systemu hosta

Dystrybucja systemu hosta

Wersja live-build na systemie hosta

Version of debootstrap on the host system

Architektura systemu live

Dystrybucja systemu live

Wersja live-boot na systemie live

Wersja live-config na systemie live

Wersja live-tools na systemie live

Można wygenerować log procesu kompilacji przy użyciu polecenia `tee`. Polecamy robić to automatycznie przy użyciu skryptu `auto/build` (patrz [Zarządzanie konfiguracją](#) w celu uzyskania szczegółów).

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

W czasie uruchamiania live-boot i live-config przetrzymuj swoje logi w `/var/log/live/`. Sprawdź je w poszukiwaniu błędów.

Dodatkowo, aby wykluczyć inne błędy, zawsze dobrym pomysłem jest, aby spakować do archiwum tar swój katalog `config/` i przesłać go gdzieś (nie należy wysłać go jako załącznika do listy), tak aby można spróbować odtworzyć napotkane błędy. Jeśli sprawia to trudność (np. ze względu na rozmiar) można użyć polecenia `lb config --dump`, które tworzy podsumowanie drzewa konfiguracyjnego (czyli np. list plików w podkatalogach `config/`, ale bez zaczynania ich).

Pamiętaj, aby wysłać wszystkie dzienniki i logi, które zostały wyprodukowane z ustawień regionalnych angielskich, np. uruchom polecenie `live-build` ze zmiennymi `LC_ALL=C` lub `LC_ALL=en_US`.

## 14.6 Wyizoluj prawdopodobne wady, jeśli to możliwe

Jeśli to możliwe, należy wyizolować przypadek do najmniejszej zmiany, która generuje błąd. Nie zawsze jest łatwo to zrobić, więc jeśli nie możesz dodać takiej informacji do raportu, nie martw się. Jednakże, jeśli użytkownik dobrze planuje swój cykl rozwoju, przy użyciu małych zestawów zmian na iterację, można być w stanie wyizolować problem poprzez budowę prostszej konfiguracji bazy, która blisko pasuje do rzeczywistej konfiguracji oraz tylko uszkodzony zestaw zmian do niej dodany. Jeśli masz trudności z sortowaniem, które z Twoich zmian wygenerowały błąd, może to znaczyć, że uwzględniono za dużo w każdym zestawie zmian i powinno się raczej kształcić w mniejszych krokach.

	14.7 Wybierz odpowiedni pakiet dla ktrego zgaszasz bd	bootstrap niezalenie od samej kompilacji systemu live lub uruchomienie lb bootstrap - debug daje wiecej informacji.	
733	Oglne rzecz biorc, naley zgasza bdy podczas kompilacji: jako dotyczce pakietu live-build, bdy podczas uruchamiania: live-boot oraz bdy w czasie dziaania systemu live: jako dotyczce live-config. Jeli nie jeste pewien, ktry pakiet bdzie odpowiedni lub potrzebujesz wiecej pomocy, przed zoeniem zgoshenia bdu, zgo raport dotyczcy pseudo-pakietu debian-live. Zajmiemy si wtedy nim i przypiszemy do odpowiedniego pakietu.	Ponadto, w przypadku korzystania z lokalnego serwera lustrzanego i/lub jakichkolwiek serwerw proxy i dowiadczania problemw prosimy zawsze najpierw sprbowa odtworzy czynoci z uyciem oficjalnego serwera lustrzanego.	741
734	Jednak bdziemy wdzieczni, jeli postarasz si zawzi pole moliwoci, w ktrych moe pojawia si bd.	14.7.3 W czasie uruchamiania	742
735	14.7.1 W czasie budowania podczas adowania pocztkowego (bootstrapping)	Jeli obraz nie uruchamia si, naley zgosi go do listy wraz z informacjami wymaganymi w <b>Zbierz potrzebne informacje</b> . Nie zapomnij wspomnie, jak/kiedy dokadnie obraz nie zadziaa, czy by uyty za pomoc wirtualizacji lub rzeczywistego sprztu. Jeli korzystasz z technologii wirtualizacji w jakiejkolwiek formie, prosz zawsze uruchomi go na prawdziwym sprzcie przed zgosheniem bdu. Zapewnienie zrzutu ekranu awarii jest rwnie bardzo pomocne.	743
736	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.		
737	W obu przypadkach nie jest to bd w systemie live, ale w samym Debianie i prawdopodobnie nie moemy naprawi go bezporednio. Prosz zgosi taki bd jako dotyczcy narzdzia adowania pocztkowego (ang. bootstrapping tool) lub uszkodzonego pakietu.	14.7.4 W czasie gdy system jest ju uruchomiony	744
		If a package was successfully installed, but fails while actually running the Live system, this is probably a bug in live-config.	745
738	14.7.2 W czasie budowania podczas instalacji pakietw	14.8 Gdzie zgasza bdy	746
739	live-build instaluje dodatkowe pakiety z archiwum Debiana i w zalenoci od uywanej dystrybucji Debian i codziennego stanu archiwum, moe si to nie powie. Jeli bd pojawi si w tym miejscu, naley sprawdzi, czy bd jest powtarzalny w normalnym systemie.	Debian Live Project ledzi wszystkie bdy w systemie ledzenia bdw (BTS). Aby uzyska informacje na temat korzystania z tego systemu, zobacz <a href="https://bugs.debian.org/">https://bugs.debian.org/</a> . Moesz te przesa bdy uywajc polecenia <code>#-reportbug</code> z pakietu o tej samej nazwie.	747
740	Jeli jest to przypadek, gdzie bd nie wystpuje w systemie live, ale w Debianie - zgo go jako dotyczcy wadliwego pakietu. Uruchomienie de-	Naley pamita, e bdy znalezione w dystrybucji pochodzcych od Debiana (takich jak Ubuntu, i inne) nie powinny by zgaszane do BTS Debiana,	748

chyba e bdy te mog by odtworzone w Debianie przy uyciu oficjalnych pakietw Debiana.

# Styl Kodowania

## 15. Styl Kodowania

Rozdział ten dokumentuje styl kodowania używany w systemach live.

### 15.1 Kompatybilność

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.

Możesz sprawdzić swoje skrypty używając 'sh -n' i 'checkbashisms'.

Upewnij się, że cały kod powokładka zaczyna się od 'set -e'.

### 15.2 Wcięcia

Zawsze używaj tabulatorów zamiast spacji.

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

Dobrze:

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

esac

### 15.3 Zawijanie

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
```

Awful:

776

Less ideal:

777

```
if [ "$MY_LOCATION_VARIABLE" = "something" ] && [ -e "$MY_OUTPUT_FILE" ]; then
    MY_OTHER_VARIABLE="$(some`bin $-FOOBAR" — awk -F' ' — ↵
    print $1 " ")"
fi
```

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

Horrible:

```
if [ "$MY_LOCATION_VARIABLE" = "something" ] && [ -e "$MY_OUTPUT_FILE" ] — [ "$MY_LOCATION_VARIABLE" = "something" ↵
-else" ] && [ -e "$MY_OUTPUT_FILE'2'" ]; then
    MY_OTHER_VARIABLE="$(some`bin $-FOOBAR" — awk -F' ' — ↵
    print $1 " ")"
fi
```

## 15.4 Zmienne

778

Zmienne wystpuj zawsze zapisane drukowanymi literami.

779

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

780

Local function variables should be restricted to local scope.

781

Zmienne dotyczce parametrw startowych live-config zaczynaj si od LIVE`.

782

Wszystkie inne zmienne w live-config zacznij przedroskiem `.

783

Uywaj nawiasw wok zmiennych; na przykad napisz `\$-FOO` zamiast \$FOO.

784

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

785

` \* Dla zachowania spjnoci, naley zawsze uywa znakw cytatu podczas przypisywania wartoci do zmiennych:

786

le:

787

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

Bad (inconsistent with existing style):

788

```
FOO=bar
```

Dobrze:

789

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

790

```
FOO="bar"
```

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

Typically bad:

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

Dobrze:

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

## 15.5 Rne

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

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; ...`

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`).

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

---

# Przykłady

# Przykłady

## 16. Przykłady

W tym rozdziale omwiono przykłady budowania dla konkretnych przypadków uycia z systemów live. Jeli jeste nowy w budowaniu wasnych obrazów systemów live, zaleca si najpierw zapoznanie z trzema kolejnymi samouczkami, a kady z nich nauczy Ci nowych technik, ktore pomog Ci uywa i rozumie pozostae przykłady.

### 16.1 Uywanie przykadów

Aby skorzysta z tych przykadów potrzebujesz systemu, ktory spenia wymagania wymienione w **wymaganiach** i ma zainstalowane live-build, jak opisano w **instalacji live-build**.

Naley zauway, e, ze wzgldu na zwizo, w tych przykadach nie okrełono lokalnego serwera uywanego do kompilacji. Mona znacznie przyspieszy pobra przypadku korzystania z lokalnego serwera lustrzanego. Mona okreli te opcje podczas korzystania z lb config, jak opisano w **Serwery lustrzane dystrybucji uywane w czasie kompilacji** lub dla wikszej wygody, ustawi domylna opcj dla systemu kompilacji w /etc/live/build.conf. Wystarczy utworzy ten plik, a w nim, ustawi odpowiedni zmienn LB\*MIRROR\* dla preferowanego serwera lustrzanego. Wszystkie inne serwery lustrzane stosowane podczas kompilacji, bd domylnie ustawione od tych wartoci. Na przykad:

```
LB*MIRROR*BOOTSTRAP="http://mirror/debian/"
LB*MIRROR*CHROOT*SECURITY="http://mirror/debian-security/"
LB*MIRROR*CHROOT*BACKPORTS="http://mirror/debian-backports/"
```

### 16.2 Samouczek 1: Domylny obraz

Przykad uycia: Stwz pierwszy prosty obraz aby nauczy si podstaw live-build.

W tym samouczku, bdziemy budowa domylny obraz live ISO-hybrid zawierajcy tylko pakiety podstawowe (bez Xorg'a) i kilka pakietów systemu live, jako pierwsze wiczenie w uyciu live-build.

Nie mona tego zrobi atwiej ni tak:

```
$ mkdir samouczek1 ; cd samouczek1 ; lb config
```

Zbadaj zawarto katalogu config/, jeli chcesz. Zobaczysz tam konfiguracje przechowywane w szkieletowych katalogach, gotowe do dostosowywania lub, w tym przypadku, uyte natychmiast, aby zbudowa domylny obraz.

A teraz jako super-uytkownik, zbuduj obraz zapisujc przy tym log podczas budowania uywajc 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 Samouczek 2: Narzdzie przegldarka

Przykad uycia: Stwrr obraz z przegldark internetow, uczc si jak wprowadza modyfikacje.

W tym samouczku, stworzymy obraz odpowiedni do wykorzystania jako narzdzie przegldarki internetowej, sucz jako wstp do dostosowywania obrazw systemu live.

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

Nasz wybr LXDE na tym przykadzie odzwierciedla nasze pragnienie, aby zapewni minimalne rodowisko pulpitu, poniewa celem obrazu jest jednorazowy uytek, ktry mamy na myli, czyli przegldarka internetowa. Moemy pj dalej i zapewni domyln konfiguracj dla przegldarki w config/includes.chroot/etc/iceweasel/profile/, lub dodatkowe pakiety wsparcia dla wywietlania rnego rodzaju treci internetowych, ale pozostawiamy to jako wiczenie dla czytelnika.

Zbuduj ponownie obraz jako super-uytkownik, zachowujc log jak to opisano w [Samouczku 1](#):

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

Jeszcze raz, zweryfikuj czy obraz jest OK i przetestuj go jak to opisano w [Samouczku 1](#).

## 16.4 Samouczek 3: Spersonalizowany obraz

Przykad uycia: Stwrr projekt spersonalizowanego obrazu zawierajcego

twoje ulubione oprogramowanie tak aby mg go zabra ze sob gdziekolwiek pjdiesz i zapisujcy sukcesywnie zmiany, kiedy tego potrzebujesz oraz zmiany w konfiguracji.

Poniewa bdziemy zmienia nasz indywidualny obraz wprowadzajc wiele zmian, chcemy, ledzi te zmiany, prbujc rzeczy eksperymentalnych i ewentualnie przywracajc je, jeli co nie wyjdzie, bdziemy trzymajc nasz konfiguracj w popularnym systemie kontroli wersji git. Bdziemy rwnie wykorzystywa najlepsze praktyki autokonfiguracji poprzez skrypty auto jak opisano w [Zarzdzanie konfiguracji](#).

### 16.4.1 Pierwsza zmiana

```
$ mkdir -p samouczek3/auto
$ cp /usr/share/doc/live-build/examples/auto/* samouczek3/auto/
$ cd samouczek3
```

Edtuj auto/config tak, aby zawiera:

```
#!/bin/sh

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

Wykonaj lb config, aby wygenerowa drzewo konfiguracyjne, uywajc wanie utworzonego skryptu w auto/config:

```
$ lb config
```

Teraz uzupenij swoj lokaln list pakietw:

```
$ 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`.

A teraz, zbuduj obraz:

```
# lb build
```

Naley zauway, e w przeciwiestwie do dwch pierwszych samouczkw, nie musimy ju wpisywa `2&1` —tee `build.log` bo jest to obecnie zawarte w `auto/build`.

Po tym jak przetestowalimy obraz (jak to jest w [Samouczku 1](#)) i jestemy zadowoleni, e dziaa, to jest to czas, aby zainicjowa nasze repozytorium `git`, dodajc tylko automatyczne skrypty przed chwil stworzone, a nastpnie dokona pierwszych zmian:

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

#### 16.4.2 Druga zmiana

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

Polecenie `lb clean` oczyi wszystkie wygenerowane pliki z poprzedniej

kompilacji z wytkiem pamici podrznej (cache), co oszczdza koniecznoci ponownego pobierania pakietw. To gwarantuje, e kolejne polecenie `lb build` ponownie uruchomi wszystkie etapy regeneracji pliki z naszej nowej konfiguracji.

```
# 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
```

Zbuduj ponownie:

```
# lb build
```

Przetestuj i jeeli jeste usatysfakcjonowany wprowad nastpn zmian:

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

Oczywicie, moliwe s bardziej skomplikowane zmiany w konfiguracji, prawdopodobnie dodajce pliki w podkatalogach `config/`. Kiedy wprowadzasz nowe zmiany, tylko uwaa, aby nie edytowa rcznie lub zmienia plikw najwyszego poziomu w `config` zawierajcych zmienn `LB*`, poniewa s to take efekty budowania i s zawsze sprztane przez `lb clean` i tworzone ponownie przez `lb config` przez odpowiednie automatyczne skrypty.

Doszlimy do koca naszej serii samouczka. Chocia moliwe jest o wiele wicej

rodzajw personalizacji, nawet tylko za pomoc kilku funkcji pokazanych w tych prostych przykadach, moe by stworzona niemal nieskoczona rnorodno obrazw. Pozostae przykady w tym rozdziale obejmuje kilka innych przypadkw uycia zaczerpnite z zebranych dowiadczce uytownikw systemw live.

## 16.5 Kiosk-klient serwera VNC

Przykad uycia: Stwrr obraz za pomoc live-build, ktry podczas uruchamiania, czy si automatycznie z serwerem VNC.

Stwrr katalog kompilacji i stwrr wewntrz szkielet folderw konfiguracji, wyczajc zalecane, aby utworzy minimalny system. A nastpnie utwrrz dwie pocztkowe listy pakietw: pierwsz wygenerowan ze skryptu dostarczonego przez live-build o nazwie Pakiety (patrz [Wygenerowane listy pakietw](#)), a drug uwzglndniajc pakiety xorg, gdm3, metacity i 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
```

Jak wyjaniono w [Podkrwanie APT, w celu zaoszczdzenia miejsca](#) moe trzeba ponownie doda niektre polecane pakiety do prawidowej pracy obrazu.

Najprostszym sposobem na wypisane listy rekomendowanych pakietw jest u ycie apt-cache. Na przykad:

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

W tym przykadzie okazao si, e musimy ponownie obj kilka pakietw zalecanych przez live-config i live-boot: user-setup do funkcji autologowania i sudo jako istotnego przy zamykaniu systemu programu. Poza tym, moe by przydatne, rwnie dodanie live-tools, aby mc skopiowa obraz systemu do pamici RAM i eject, aby ewentualnie wysun nonik live. Tak wic:

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

Po tym, stwrr katalog /etc/skel w config/includes.chroot i umie tam wasny .xsession dla domylnego uytownika, ktry bdzie uruchamia metacity i xvncviewer, podczajc si do portu 5901 na serwerze w 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
```

Zbuduj obraz:

```
# lb build
```

Korzystaj.

## 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
```

Aby uczyni by obraz pracowa prawidowo, musimy ponownie doda przy najmniej dwa pakiety, ktore zostay pominite przez opcj `-apt-recommends false`. Zobacz [Podkracanie APT w celu zaoszczdzenia miejsca](#)

```
$ 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
```

Teraz, zbuduj obraz w typowy sposb:

```
# 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 Pulpit GNOME w lokalnym jzyku oraz instalator

Przykad uycia: Stwrrz obraz z pulpitem GNOME i lokalizacj dla Szwajcarii wraz z instalatorem.

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.

wszystkie wykryte metapakiety zada do naszej listy pakietw w następujący sposób:

885 Naszym początkowym problemem jest odkrycie nazw odpowiednich zadań językowych. Obecnie, live-build nie może nam w tym pomóc. Chociaż możemy mieć szczęście i znaleźć to metodą prób i błędów, to jest narzędzie, `grep-dctrl`, które możemy użyć do ustalenia tego z opisów zadań w `tasksel-data` tak więc, aby przygotować się upewnij się, że masz obie te rzeczy:

892

886

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

887 Teraz możemy rozpocząć wyszukiwanie odpowiedniego zadania. Najpierw:

888

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

889 Dzięki temu poleceniu dowiadujemy się, że zadanie nazywa się po prostu niemiecki (ang. german). Teraz, aby znaleźć podobne zadania:

890

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

891 W czasie startu systemu będziemy generować lokalizację `de`CH.UTF-8` i wybierać układ klawiatury `ch`. Teraz poskładamy kawałki razem. Przypominamy sobie **Korzystanie z metapakietów** – metapakiety są poprzedzone przedrostkiem `task-`, po prostu określimy te parametry rozruchowe dotyczące języka, a następnie dodamy standardowe pakiety priorytetowe i

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

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



895	Przewodnik redakcyjny	906	Spodziewamy si, e rne odmiany jzyka mog miesza si bez tworzenia nieporozumie, ale oglnie naley stara si by spjnym i przed podjciem decyzji o uyciu brytyjskiego, amerykaskiego lub innego dialektu jzyka angielskiego wedle uznania, prosz przyjrze si, jak inni ludzie pisz i postara si ich naladowa.	
896	17. Przewodnik redakcyjny			
897	17.1 Wytyczne dla autorw		Bd zbalansowany	907
898	Ten rozdzia zajmuje si pewnymi oglnymi rozwaaniami, ktre naley wzi pod uwag podczas pisania dokumentacji technicznej dla live-instrukcji. S one podzielone na funkcje jzykowe oraz zalecane procedury.		Nie by stronniczy. Unikaj w tym odniesienia do ideologii zupenie niepowizanych z live-manual. Pismo techniczne powinny by moliwie jak najbardziej neutralne. Jak to jest w naturze pimiennictwa naukowego.	908
899	Uwaga: Autorzy powinni najpierw przeczyta <b>Wsptworzenie tego dokumentu</b>		Bd poprawny politycznie	909
900	17.1.1 Funkcje jzykowe		Staraj si unika seksistowskiego jzyka, jak to jest tylko moliwe. Jeli potrzebujesz, odwoania do trzeciej osoby liczby pojedynczej najlepiej uy jakiej formy bezosobowej lub wy zamiast on , ona lub niewygodnych wynalazkw, takie jak mgby/mogaby, byem/am i podobnych.	910
901	Uyj zwykego angielskiego		Bdz zwizy	911
902	Naley pamita, e wysoki procent czytelnikw nie s rodzimymi uytkownikami jzyka angielskiego. Wic jako generaln zasad sprbuj uywa krtkich, sensownych zda, zakoczonych kropk.		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.	912
903	To nie znaczy, e trzeba korzysta z uproszczonego, naiwnego stylu. Proponuje si unika, na ile to moliwe, zoonych zda podrzdnych, ktre czyni tekst trudny do zrozumienia dla nie rodzimych uytkownikw jzyka angielskiego.		Oszczd pracy tumaczycym	913
904	Rnorodno jzyka angielskiego		Naley pamita, e cokolwiek napiszesz bdzie musiao zosta przetumaczone na kilka innych jzykw. Oznacza to, e sporo osb, bdzie musia wykona dodatkow prac jeli dodasz bezuzyteczne lub nadmiarowe informacje.	914
905	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.		Bd zgodny	915
			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.	916

917	Be spjny	wyszukiwarki, aby sprawdzi, jak inni autorzy mog korzysta z niektrych	
918	Use as many text-forming devices as necessary to make your text cohesive and unambiguous. (Text-forming devices are linguistic markers such as connectors).	wyrae moe bardzo pomoc.	
		Faszywi przyjaciele, idiomy i inne wyraenia idiomatyczne	927
919	Bd opisowy	Uwaaj na faszywych przyjaci. Bez wzgldu na to, jak biegy jeste w obcym	928
920	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.	jzyku nie mona pomoc wpadajc od czasu do czasu w puapki tzw. faszy- wych przyjaci, sowa, ktre wygdaj podobnie w dwch jzykach, ale ktorych znaczenie lub zastosowanie moe by zupenie inne.	
		Staraj si unika idiomw w jak najwikszym stopniu. Idiomy to wyraenia,	929
921	Sownik	ktre mog mie znaczenie zupenie odmienne od tego, co ich poszczeglnie sowa wydaj si oznacza. Czasami, idiomy, mog by trudne do zrozumienia nawet dla rodzimych uytownikw jzyka angielskiego!	
922	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.	Unikaj slangu, skrtw, mowy potocznej...	930
		Nawet jeli popierasz korzystanie ze zwykego, codziennego jzyka angielskiego, pisanie techniczne naley do formalnej formy jzyka.	931
923	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.	Staraj si unika slangu, niepopularnych skrtw, ktre s trudne do zrozumi- nienia, a przede wszystkim skrtw, ktre prbuj naladowa jzyk mwiony. Nie wspominajc o typowych dla kanaw IRC i przyjaznych dla zamkni- tych gron wyrae.	932
924	Zgodnie z ogln zasad, jeli koncepcja moe by wyraony za pomoc rnych synonimw to dobr rad bdzie wybieranie pierwszego sowa zaproponowanego przez sownik. W razie wtpliwoci, czsto susznym jest wybieranie sowa pochodzenia germaskiego (zwykle jednosylabowe sowa). Ostrzegamy, e te dwie techniki, mog produkowa raczej mow nieformaln, ale przynajmniej wybr sw bdzie o szerokim zastosowaniu i oglnie przyjty.	17.1.2 Procedury	933
		Przetestuj przed zapisaniem	934
925	Korzystanie ze sownika kolokacji jest zalecane. S one bardzo pomocne, kiedy przychodzi do znajomoci sw najczciej wystpujcych razem.	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.	935
926	Ponownie; dobr praktyk jest, aby uczy si z pracy innych. Korzystanie z	Przykady	936



W przypadku dostarczania przykadu sprbuj by tak dokadny jak tylko moesz. Przykad jest, mimo wszystko, tylko przykadem.

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.

#### Linki zewntrzne

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.

Poza tym, ludzie, ktrzy czytaj instrukcj w trybie offline nie bd mogli ledzi tych linkw.

Unikaj nadawania marki i rzeczy, ktre naruszaj licencj zgodnie z ktr podrcznik ten zosta opublikowany

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 jest oparty na licencji GNU GPL. Ma to wiele skutkw, ktre odnosz si do redystrybucji materiau (dowolnego rodzaju, w tym grafiki chronionej prawami autorskimi lub logo), ktry jest opublikowany wraz nim.

Napisz pierwszy szkic, przeglndnij go, edytuj, popraw i cofnij zmiany

jeeli wymaga tego sytuacja

- Burza mzgwl. Najpierw musisz zorganizowa swoje pomysy w logicznej kolejnoci zdarze.

- Kiedy ju w jaki sposb masz zorganizowane te koncepcje w gowie napisz pierwszy szkic.

- Dokonaj przeglndu gramatyki, skadni i pisowni. Naley pamita, e waciwe nazwy wyda, takich jak trixie lub sid nie powinny by kapitalizowane, gdy odnosi si do nich jako nazw kodowych. Aby sprawdzi pisowni mona uruchomi cel spell. tj. polecenie make spell

- Udoskonalaj swoje wyraenia, a jeli to konieczne cofnij i przerb kad cz.

#### Rozdzia

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.

#### Znaczники

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
```

To s przykady znacznikw, ktre mog okaza si uyteczne:

- Dla pogrubienia uyj:

```
*-foo"* lub !-foo"!

```

powoduje: foo lub foo . Uyj tego by wyszczeglni okrelone sowa kluc-  
zowe.

- Dla kursywy uyj:

```
/-foo"/

```

powoduje: foo. Uyj tego dla np. nazw paczek Debiana.

- Dla czcionki o staej szerokoci uyj:

```
#-foo"#

```

powoduje: foo. Uyj tego np. dla nazw polece. A take aby uwidoczni  
poszczególne sowa kluczowe jak cieki dostpowe.

- Dla blokw z kodem uyj:

```
code-
    $ foo
    # bar
"code

```

powoduje:

```
$ foo
# bar

```

Uyj code- do otwarcia i #-"code# do zamknicia tagu. Wane jest, aby  
pamita, by zostawi miejsce na pocztku kadej linii kodu.

## 17.2 Wytyczne dla tumaczy

Ta sekcja zajmuje si pewnymi oglnymi rozwaaniami, ktore nalezy wzi pod  
uwag przy tumaczeniu zawartoci live-manual.

Jako ogne zalecenie, tumacze powinni przeczyta i zrozumie zasady tu-  
maczenia, ktore maj zastosowanie do ich specyficznych jzykw. Zazwyczaj,  
grupy i listy dyskusyjne tumacze dostarczaj informacji o tym, jak tworzy  
przetumaczone prac zgodne z normami jakoci Debiana.

Uwaga: Tumacze powinni rwnie przeczyta **Wsptworzenie tego doku-  
mentu**. W szczeglnoci rozdzia **Tumaczenie**

### 17.2.1 Wskazwki tumaczenia

#### Komentarze

The role of the translator is to convey as faithfully as possible the mean-  
ing 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.

#### UT, Uwagi Tumacza

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.

## Wyraenia w trybie bezosobowym

siężone w oryginalnych plikw. Znaki te czsto pojawiaj si, na przykad, w blokach z kodem.

Nie popeniaj bedw, to nie znaczy, e przetumaczony tekst musi mie tak sam dugo jak w wersji angielskiej. Jest to prawie niemoliwe.

Nieprzetumaczalne cigi znakw 993

Tumacze nigdy nie powinni tumaczy: 994

- Nazw kodowych wyda (ktre powinny by pisane majmi literami) 995

- Nazw programw 996

- Komend podanych jako przykad 997

- Metadanych (zazwyczaj pomidzy dwukropkami :metadata: ) 998

- Linkw 999

- cieek dostpnowych 1000

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.

## Faszywi przyjaciele

Puapka faszywych przyjaci wyjanionych wczniej szczeglnie dotyczy tumaczy. Dwukrotnie sprawdzi znaczenie podejrzanych faszywych przyjaci w razie wtpliwoci.

## Znaczniki

Tumacze pracujcy pocztkowo nad plikami pot , a pniej z plikami \*\_po" \* znajd wiele znacznikw w cigach. Mog oni przetumaczy tekst tak, jak to jest tylko moliwe do tumaczenia, ale niezwykle wanym jest, aby uywali oni dokadnie takich samych znacznikw jak w oryginalnej wersji angielskiej.

## Bloki kodowe

Mimo, e bloki z kodem s zazwyczaj nieprzetumaczalne, zawarcie ich w tumaczeniach jest jedynym sposobem, aby zdoby 100% kompletne tumaczenie. I mimo, e oznacza to wiecej pracy na pocztku, bo to moe wymaga interwencji od tumaczy jeli kod si zmieni, to jest to najlepszy sposb, w duszej perspektywie czasu na okrelenie, co ju zostao przetumaczone, a co nie podczas sprawdzania integralnoci plikw .po.

## Nowe linijki

Przetumaczone teksty musz mie te same znaki nowej linii jak teksty oryginalne. Naley uwaa, aby nacisn klawisz Enter lub wpisa jeli pojawi

## SiSU Metadata, document information

Ruby version: ruby 3.3.7 (2025-01-15 revision be31f993d7) [x86\_64-linux-gnu]

Title: Debian Live Manual

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

Rights: 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.

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

Date: 2025-02-26

### Version Information

Sourcefile: 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)=3ecd29a0acab4d07de310519f7a592dc-  
638f844ed15c18eac179d4fb9df0d5c0

### Generated

Document (ao) last generated: 2025-02-27 00:02:09 +0000

Generated by: SiSU 7.3.0 of 2023w44/1 (2023-10-30)