

# Podręcznik Systemów Live

Projekt Systemów Live<[debian-live@lists.debian.org](mailto:debian-live@lists.debian.org)>

2015-08-23

Copyright © Copyright (C) 2006-2015 Projekt Systemów Live

Ten program jest wolnym oprogramowaniem: moesz go rozprowadza dalej i / lub modyfikowa zgodnie z warunkami Powszechnej Licencji Publicznej GNU opublikowanej przez Free Software Foundation, wedug wersji 3 tej Licencji lub (wedug Twojego wyboru) dowolnej póniejszej wersji

Ten program jest rozpowszechniany w nadziei, e bdzie uyteczny, ale BEZ JAKIEJKOLWIEK GWARANCJI; bez nawet domylnej gwarancji PRZYDATNOCI HANDLOWEJ albo PRZYDATNOCI DO OKRELONYCH ZASTOSOWA. Patrz GNU General Public License aby uzyska wiecej szczegóów.

Powiniene otrzyma kopi Licencji GNU General Public License wraz z tym programem. Jeli nie, odwied <http://www.gnu.org/licenses/>.

Peny tekst licencji GNU General Public mona znale w pliku /usr/share/common-licenses/GPL-3.

---

# **Podręcznik Systemów Live**

## **Projekt Systemów Live<debian-live@lists.debian.org>**

1

---

<b>Podręcznik Systemów Live</b>	<b>a</b>
<b>O tym podręczniku</b>	<b>2</b>
<b>O tym podręczniku</b>	<b>3</b>
<b>O tym podręczniku</b>	<b>4</b>
Dla niecierpliwych	4
Definicje	4
Autorzy	6
Wnoszenie wkładu do tego dokumentu	6
Nanoszenie zmian	6
Tumaczenie	7
<b>O <code>{project}</code></b>	<b>9</b>
<b>O <code>{project}</code></b>	<b>10</b>
Motywacja	10
Co jest nie tak w moim dotychczasowym systemie live?	10
Czemu tworzy nasz własny system live?	10
Filozofia	10
Tylko niezmienione pakiety z działo Debian "main"	10
Bez konfiguracji pakietów systemu live	11
Kontakt	11
<b>Użytkownik</b>	<b>12</b>

---

<b>Instalacja</b>	<b>13</b>
<b>Instalacja</b>	<b>14</b>
Wymagania	14
Instalowanie live-build	14
Z repozytorium Debiana	14
Ze róda	14
Ze zrzutów deweloperskich	15
Instalowanie live-boot i live-config	15
Z repozytorium Debiana	15
Ze róda	15
Ze zrzutów deweloperskich	16
<b>Podstawy</b>	<b>17</b>
<b>Podstawy</b>	<b>18</b>
Co to jest system live?	18
Pobieranie prekompilowanych obrazów	19
Using the web live image builder	19
Uywanie i przestrogi dotyczce Web buildera	19
Pierwsze kroki: budowanie obrazu ISO-hybrydy	20
Korzystanie z hybrydowego obrazu ISO live	20
Wypalanie obrazu ISO na fizycznym noniku	20
Kopiowanie obrazu ISO-hybrydy na nonik USB	21
Wykorzystanie przestrzeni pozostaej na noniku USB	21
Uruchamianie nonika live	22
Uywanie wirtualnej maszyny do testowania	22
Testowanie obrazu ISO z uyciem QEMU	22

---

Testowanie obrazu ISO z uyciem VirtualBox'a	23
Budowanie i uywanie obrazu HDD	23
Budowanie obrazu netboot	24
Serwer DHCP	25
Serwer TFTP	25
Serwer NFS	26
Netboot testing HowTo	26
Qemu	26
Webbooting	27
Getting the webboot files	27
Uruchamianie obrazów webboot	27
<b>Przegld narzdzi</b>	<b>29</b>
<b>Przegld narzdzi</b>	<b>30</b>
Pakiet live-build	30
Polecenie lb config	30
Wcicia	31
Polecenie lb build	31
Polecenie lb clean	31
Pakiet live-boot	31
Pakiet live-config	32
<b>Zarzdzanie konfiguracj</b>	<b>33</b>
<b>Zarzdzanie konfiguracj</b>	<b>34</b>
Radzenie sobie ze zmianami konfiguracji	34
Czemu uywa automatycznych skryptów? Co one robi?	34

---

Uyj przykładowych automatycznych skryptów	34
Klonowanie konfiguracji opublikowanej przez Git	35
<b>Dostosowywanie zawartoci</b>	<b>37</b>
<b>Opis dostosowywania</b>	<b>38</b>
Konfiguracja podczas kompilacji vs. podczas uruchamiania systemu	38
Etapy kompilacji	38
Uzupenie lb config plikami	39
Zadania dostosowywania	39
<b>Dostosowywanie instalacji pakietów</b>	<b>40</b>
<b>Dostosowywanie instalacji pakietów</b>	<b>41</b>
rda pakietu	41
Dystrybucja, dziaay archiwum i tryb	41
Serwery lustrzane dystrybucji	42
Serwery lustrzane dystrybucji uywane podczas budowania obrazu	42
Serwery lustrzane dystrybucji uyte podczas uruchomienia	42
Dodatkowe repozytoria	42
Wybieranie pakietów do instalacji	43
Lista pakietów	43
Uywanie metapakietów	43
Lokalna lista pakietów	44
Lokalna lista pakietów binarnych	44
Wygenerowana lista pakietów	44
Uywanie instrukcji warunkowych w listach pakietów.	45
Usuwanie pakietu podczas instalacji	45

---

Pulpit i zadania jzykowe	46
Rodzaj jdra i wersja	46
Niestandardowe jdra	47
Instalowanie zmodyfikowanych pakietów lub pakietów innych firm	47
Uywanie packages.chrootdo instalacji niestandardowych pakietów	48
Uywanie repozytorium APT aby zainstalowa niestandardowe pakiety	48
Niestandardowe pakiety i APT	48
Konfigurowanie APT podczas kompilacji	48
Wybieranie apt lub aptitude	49
Uywanie serwera proxy z APT	49
Podkrwanie APT celu zaoszczdzenia miejsca	49
Przekazywanie opcji do apt lub aptitude	50
Pinning APT	50
<b>Dostosowywanie zawartoci</b>	<b>52</b>
<b>Dostosowywanie zawartoci</b>	<b>53</b>
Uwzglndnianie	53
Lokalnie uwzglndniane w chroot/live	53
Lokalnie uwzglndniane dane binarne	54
Haki	54
Lokalne haki chroot/live	54
Haki podczas uruchamiania	54
Lokalne haki binarne	55
Wstpne ustawienie pyta Debconfa (Preseeding)	55



---

<b>Dostosowywanie zdarze podczas uruchamiania systemu</b>	<b>56</b>
<b>Dostosowywanie zdarze podczas uruchamiania systemu</b>	<b>57</b>
Personalizacja uytkownika live	57
Ustawianie lokalizacji i jzyka	57
Persistence	59
Plik persistence.conf	60
Uywanie wiecej ni jednego magazynu persistence	61
Using persistence with encryption	62
<b>Dostosowywanie obrazu binarnego</b>	<b>64</b>
<b>Dostosowywanie obrazu binarnego</b>	<b>65</b>
Programy adujce (ang. Bootloadery)	65
Metadane ISO	65
<b>Dostosowywanie Instalatora Debiana</b>	<b>67</b>
<b>Dostosowywanie Instalatora Debiana</b>	<b>68</b>
Typy Instalatora Debiana	68
Dostosowywanie Instalatora Debiana przez preseeding	69
Dostosowywanie zawartoci Instalatora Debiana	69
<b>Projekt</b>	<b>70</b>
<b>Wnoszenie wkadu do tego projektu</b>	<b>71</b>
<b>Wnoszenie wkadu do tego projektu</b>	<b>72</b>
Wprowadzanie zmian	72

---

<b>Zgaszanie bđów</b>	<b>74</b>
<b>Zgaszanie bđów</b>	<b>75</b>
Znane problemy	75
Przebuduj od zera	75
Uywaj aktualnych pakietów	75
Zbierz potrzebne informacje	76
Wyizoluj prawdopodobn wad, jeli to moliwe	76
Wybierz odpowiedni pakiet dla którego zgaszasz bđ	77
W czasie budowania podczas adowania poczkowego (bootstrapping)	77
W czasie budowania podczas instalacji pakietów	77
W czasie uruchamiania	77
W czasie gdy system jest ju uruchomiony	78
Spróbuj wykona par kroków	78
Gdzie zgasa bdy	78
<b>Styl Kodowania</b>	<b>79</b>
<b>Styl Kodowania</b>	<b>80</b>
Kompatybilno	80
Wcicia	80
Zawijanie	80
Zmienne	81
Róne	82
<b>Procedury</b>	<b>83</b>
<b>Procedury</b>	<b>84</b>
Gówne wydanie	84

---

Wydanie Docelowe	84
Ostatnie Wydanie Docelowe Debiana	84
Szablon obwieszczenia dla wydania docelowego	84
<b>Repozytorium Git</b>	<b>86</b>
<b>Repozytorium Git</b>	<b>87</b>
Obsuga wielu repozytoriów	87
<b>Przykady</b>	<b>89</b>
<b>Przykady</b>	<b>90</b>
<b>Przykady</b>	<b>91</b>
Uywanie przykadów	91
Samouczek 1: Domylny obraz	91
Samouczek 2: Narzdzie przegldarka	92
Samouczek 3: Spersonalizowany obraz	92
Pierwsza zmiana	93
Druga zmiana	94
Kiosk-klient serwera VNC	94
Bazowy obraz dla nonika USB z 128MB pamici.	95
Pulpit GNOME w lokalnym jzyku oraz instalator	96
<b>Dodatek</b>	<b>98</b>
<b>Przewodnik redakcyjny</b>	<b>99</b>
<b>Przewodnik redakcyjny</b>	<b>100</b>
Wytyczne dla autorów	100
Funkcje jzykowe	100

---

Procedury	102
Wytyczne dla tłumaczy	104
Wskazówki tłumaczenia	104

# O tym podręczniku

2

---

# O tym podręczniku

3

## O tym podręczniku

This manual serves as a single access point to all documentation related to the `{project}` and in particular applies to the software produced by the project for the Debian 9.0 "`{stable}`" release. An up-to-date version can always be found at

`http://live-systems.org/`

Podczas, gdy podręcznik `live-manual` skupia się przede wszystkim na pomocy w budowaniu systemu `live`, a nie na tematach użytkownika `kocowego`. To użytkownik `kocowy` 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 `persistency`.

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 odróżnić te polecenia, które mogą być wykonywane przez nieuprzywilejowanego użytkownika i te wymagające praw administratora, polecenia zostają poprzedzone przez odpowiednio `$` or `#`. Symbol ten nie jest częścią polecenia.

## 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 o szybkich postępach 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 wicze 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`.

Zachęcamy do powrotu i bardziej dogłębnej analizy podręcznika, aby następnie razem czytać Podstawy, przejrzeć lub pominiąć Budowanie obrazu `netboot`, a skoczywszy czytaj Omówienie dostosowywania i rozdziałów, które po nim następują. W tym momencie, mamy nadzieję, że jesteś zainteresowany tym, co można zrobić z systemami `live` i zmotywowany, aby przeczytać resztę tego podręcznika, od deski do deski.

## Definicje

**System `live`:** System operacyjny, który można uruchomić bez instalacji na dysku twardym. Systemy `live` nie zmieniają lokalnego systemu(-ów) lub pliku(-ów) już

zainstalowany na dysku twardym komputera, chyba e zostao to specjalnie ustawione. Systemy live s zwykle uruchamiane z noników takich jak pyty CD, DVD lub pamici USB. Niektóre mog si take uruchamia z sieci (za porednictwem obrazów netboot, patrz Budowanie obrazu netboot) i przez Internet (za pomoc parametru startowego fetch=URL, patrz Webbooting). webbooting).

**Nonik live:** W odrónieniu od systemu live, nonik live odnosi si do dysku CD, DVD lub pamici USB, gdzie znajduj si pliki binarne stworzone przez live-build, które s uywane do uruchamiania systemu live. Szerzej, termin ten odnosi si równie do kadego miejsca, w którym znajduj si te pliki binarne, które s potrzebne do uruchomienia systemu live, równie takich miejsc jak miejsca dla plików startowych w sieci.

**`live`:** Projekt, który 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:** Zbiór skryptów wykorzystywanych do uruchamiania systemów live.

**live-build:** Zbiór skryptów wykorzystywanych do budowy niestandardowych systemów live.

**live-config:** Zbiór skryptów uywane do konfiguracji systemu live w czasie procesu bootowania.

**live-tools:** Zbiór dodatkowych skryptów 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, które mog by wprowadzone w wierszu bootloadera, aby wpyn na jdro lub live-config.

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

**Obraz binarny:** Plik zawierajcy system live, takie jak `live-image-i386.hybrid.iso` lub `live-image-i386.img`.

**Dystrybucja docelowa:** dystrybucja, na której opiera si bdzie system ywo. To moe si róni od dystrybucji systemu hosta.

**stable/testing/unstable:** Dystrybucja **stable**, obecna nazwa kodowa to `stable`, zawiera najnowsze oficjalnie wydanie dystrybucji Debian. Dystrybucja **testing**, tymczasowo nadana nazwa kodowa to `testing`, to obszar postoi przed nastpnym wydaniem **stable**. Gówn 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 rozwój Debian. Ogólnie rzecz biorc, to dystrybucja prowadzona jest przez deweloperów i tych, którzy lubi ycie na krawdzi. W tym podręczniku, mamy tendencj do korzystania z nazw kodowych wyda, takich jak `testing` lub `sid`, bo gównie w tych wydaniach jest zawarte to co aktualnie zawieraj narzdzia same w sobie.



<b>Autorzy</b>	29
Lista Autorów (w kolejności alfabetycznej):	30
Ben Armstrong	31
Brendan Sleight	32
Carlos Zuferrì	33
Chris Lamb	34
Daniel Baumann	35
Franklin Piat	36
Jonas Stein	37
Kai Hendry	38
Marco Amadori	39
Mathieu Geli	40
Matthias Kirschner	41
Richard Nelson	42
Trent W. Buck	43

## **Wnoszenie wkadu do tego dokumentu**

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

## **Nanoszenie zmian**

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

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

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

```
$ make build
```

Ponieważ zbudowanie podręcznika we wszystkich obsługiwanych językach zajmuje trochę czasu, autorzy mogą zdecydować, czy wygodniej będzie skorzystać z jednego z szybko korygujących skrótów podczas zamieszczania nowej dokumentacji, którą dodali do podręcznika angielskiego. Używanie `PROOF=1` tworzy live-manual w formacie HTML, ale bez posegmentowanych plików HTML, a przy użyciu `PROOF=2` tworzy się live-manual w formacie PDF, ale tylko sformatowane jako A4 i listowy pionowy. Dlatego korzystając z jednej z możliwości opcji `PROOF=` można zaoszczędzić sporo czasu, np.:

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

Gdy zatwierdzasz jedno z tłumaczeń, możliwe jest zbudowanie tylko dla jednego języka, przez wykonanie, np.:

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

Jest to możliwe, aby zbudować po typie dokumentu, np.:

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

Lub kombinacja obu, np.:

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

Po rewizji swojej pracy i upewnieniu się, że wszystko jest w porządku, nie należy używać `make commit` chyba aktualizujesz już istniejące tłumaczenia, i w tym przypadku, nie należy mieszać zmian do instrukcji angielskiej i tłumaczeń w tej samej wysłanej zmianie, ale należy użyć osobnych zgłoszeń zmian dla każdego tłumaczenia. Zobacz sekcję Tłumaczenie, aby uzyskać więcej szczegółów.

## Tłumaczenie

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 wyluj 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.

Po sprawdzeniu, aby wczytać nowy język w autobuild to wystarczy dodać początkowo przetłumaczone pliki do `manual/po/${LANGUAGE}/` i uruchomić `make commit`. A następnie, edytować `manual/_sisu/home/index.html` dodając nazwę języka i jego nazwy w języku angielskim w nawiasach.

- Kontynuuj pracę z już rozpoczętym tłumaczeniem 65
- Jeli Twój język docelowy został już dodany, można losowo kontynuować tłumaczenia pozostałych plików `.po` w `manual/po/{LANGUAGE}/` za pomocą dowolnego edytora (np. *poedit*). 66
- 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 wyjątkowo w Zatwierdzanie zmian 67
- 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 wysłać swój wkład w tłumaczenie. 68
- `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. 69
- 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 dokumentacją Debian o lokalizacjach (l10n) i tymi specyficznymi dla `live-manual`: Poradnik dla tłumaczy. 70
- 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. 71

---

# O `{project}`

## O `{project}` 73

### Motywacja 74

#### Co jest nie tak w moim dotychczasowym systemie live? 75

Gdy `{project}` został zainicjowany, było już dostępnych kilka systemów live opartych na Debianie i które teraz wietnie się spisują. Z punktu widzenia Debiana większość z nich ma jednak jedną lub więcej z następujących wad:

Nie są to projekty Debiana i dlatego nie posiadają takiego wsparcia ze strony Debiana. 77

Mieszają różne dystrybucje, np.: **testing** i **unstable**. 78

Wspierają tylko i386. 79

Modyfikują zachowanie i/lub wygląd pakietów przez obcinanie ich w celu zaoszczędzenia miejsca. 80

Zawierają pakiety z poza archiwum Debiana. 81

\* Są dostarczane z jądrem systemu zawierającym dodatkowe aty, które nie są czyste Debiana. 82

Są duże i powolne a ze względu na swój zwykły wielkość, a zatem nie nadają się do zagadnień odzyskiwania. 83

Nie są one dostępne na różnych nośnikach, np. CD, DVD, USB-stick czy obrazy netboot. 84

#### Czemu tworzy nasz własny system live? 85

Debian to Uniwersalny system operacyjny: Debian będzie posiadał system live do przetestowania i dokładnego zaprezentowania systemu Debiana z następującymi głównymi zaletami:

Będzie pod-projektem Debiana. 87

Będzie odzwierciedlał stan aktualnej dystrybucji. 88

Będzie wspierał tak wiele architektur jak to tylko możliwe. 89

Będzie składał się tylko z niezmiennych pakietów Debiana. 90

Nie będzie zawierał żadnych pakietów, które nie są w archiwum Debiana. 91

Będzie korzystał z niezmiennego jądra Debiana bez dodatkowych poprawek. 92

### Filozofia 93

#### Tylko niezmiennione pakiety z działu Debian "main" 94

Będziemy używać tylko pakietów z repozytorium Debiana w sekcji "main". Sekcja "non-free" nie jest czystym Debianem, a zatem nie może być używana do budowania oficjalnych obrazów systemu live. 95

Nie będziemy zmieniać żadnych pakietów. Zawsze, gdy będziemy musieli coś zmienić, zrobimy to w porozumieniu z opiekunem tego pakietu w Debianie. 96

W drodze wyjątku, nasze własne pakiety, takie jak live-boot, live-build lub live-config mogłyby być zastosowane tymczasowo z własnego repozytorium z przyczyn rozwojowych (np. do tworzenia zrzutów rozwojowych). Zostaną one przesłane do Debiana na bieżąco. 97

## Bez konfiguracji pakietów systemu live 98

Na tym etapie nie będziemy dostarczać lub testować instalacji przykładowych lub alternatywnych konfiguracji pakietów. Wszystkie pakiety będą użyte w ich podstawowych konfiguracjach takich jak po normalnej instalacji Debiana. 99

Za każdym razem, gdy potrzebujemy innej domyślnej konfiguracji, zrobimy to w porozumieniu z opiekunem pakietu w Debianie. 100

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 Debiana, aby lepiej pracowały w systemie live w porównaniu do dokonywania zmian w live toolchain lub konfiguracji obrazu prekompilowanego klonując konfigurację przez git. Aby uzyskać więcej informacji, zobacz Omówienie dostosowywania. 101

## Kontakt 102

**Lista Mailingowa:** Podstawowym kontaktem do projektu jest lista mailingowa na <https://lists.debian.org/debian-live/>. Możesz napisać do listy bezpośrednio przez zadrośnienie poczty do [debian-live@lists.debian.org](mailto:debian-live@lists.debian.org). Archiwa listy dostępne są na <https://lists.debian.org/debian-> 103

**IRC:** Wielu użytkowników i deweloperów jest obecnych na kanale #debian-live na [irc.debian.org](https://irc.debian.org) (OFTC). Kiedy zadajesz pytanie na IRC, prosimy o cierpliwie czekać na odpowiedź. W przypadku, gdy odpowiedź nie pojawi się, napisz na listę mailingową. 104

**BTS:** <https://www.debian.org/Bugs/> (BTS) zawiera informacje o błędach 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#bugs}. 105

---

# Uytkownik

106

---

# Instalacja

107



<b>Instalacja</b>	108
<b>Wymagania</b>	109
Budowanie obrazów systemu live ma bardzo niewielkie wymagania systemowe:	110
Dostęp do konta (root) super-użytkownika	111
Aktualna wersja live-build	112
Powłoka zgodna z POSIX, taka jak <i>bash</i> lub <i>dash</i>	113
<i>debootstrap</i>	114
Linux 2.6 lub nowszy.	115
Należy pamiętać, że użycie Debiana lub dystrybucji pochodzącej od Debiana nie jest wymagane - live-build będzie działał na prawie każdej dystrybucji spełniającej powyższe wymagania.	116
<b>Instalowanie live-build</b>	117
Mожesz zainstalować live-build na wiele różnych sposobów:	118
Z repozytorium Debiana	119
Ze źródła	120
Ze zrzutów deweloperskich	121
Jeli używasz Debiana, zalecanym sposobem jest zainstalowanie live-build poprzez repozytorium Debiana.	122
<b>Z repozytorium Debiana</b>	123
Zwyczajnie zainstaluj live-build jak każdą inną paczkę:	124
	125
<pre># apt-get install live-build</pre>	
<b>Ze źródła</b>	126
live-build jest opracowana z wykorzystaniem systemu kontroli wersji Git. W systemach opartych na Debianie, jest on dostarczany przez pakiet <i>git</i> . Aby sprawdzić najnowszy kod, wykonaj:	127
	128
<pre>\$ git clone git://live-systems.org/git/live-build.git</pre>	
Mожesz zbudować i zainstalować swoją paczkę Debiana wykonując:	129
	130

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

Teraz zainstaluj którąkolwiek wiec zbudowan paczk #`{.deb}`, wedle wyboru, np.

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

Moesz równie zainstalowa live-build bezporednio w swoim systemie wykonujc:

```
# make install
```

i odinstalowa go wykonujc:

```
# make uninstall
```

## Ze zrzutów deweloperskich

Jeli nie chcesz, kompilowa i zainstalowa live-build ze róda, moesz uy zrzutów deweloperskich. S to automatycznie zbudowane paczki z najnowszej wersji Git i s one dostpne na <http://live-systems.org/debian/>.

## Instalowanie live-boot i live-config

**Uwaga:** Nie musisz instalowa live-boot lub live-config w systemie do tworzenia nies-tandardowych systemów ywych. Jednak ten sposób nie zaszkodzi i jest przydatny do celów porównawczych. Jeli chcesz tylko przejrze dokumentacj, moesz zainstalowa pakiety *live-boot-doc* i *live-config-doc* oddzielnie.

## Z repozytorium Debiana

Zarówno live-boot oraz live-config s dostpne w repozytorium Debiana, tak jak w Instalacji live-build.

## Ze róda

Aby uywa najnowszych róde z repozytorium GIT, uyj poniszego polecenia. Prosz upewnij si, e zapoznae si z warunkami wymienionymi w Warunkach.

Sklonuj róda live-boot i live-config

```
$ git clone git://live-systems.org/git/live-boot.git
$ git clone git://live-systems.org/git/live-config.git
```

Porad si podręcznikiem man pakietó live-boot i live-config aby uzyska szczegółowe informacje na temat dostosowywania, jeeli to jest Twój powód do budowania tych pakietów ze róde. 147

Zbuduj pliki .deb live-boot i live-config 148

Musisz budowa obraz albo na dystrybucji docelowej lub w rodowisku chroot zawierającym platform docelowej: oznacza to, czy celem jest `{testing}` to obraz naley budowa na `{testing}`. 149

Uywaj osobistych konstruktorów takich jak *pbuilder* lub *sbuild*, jeeli istnieje potrzeba zbudowania live-boot na dystrybucji docelowej, która róni si od systemu budowania. Na przykad, dla obrazów `{testing}` live, zbuduj live-boot w rodowisku chroot `{testing}`. Jeli dystrybucja docelowa zgadza si z dystrybucj systemu kompilacji, mona wtedy zbudowa bezporednio na wbudowanym systemie, uywajc `dpkg-buildpackage` (dostarczanego przez pakiet *dpkg-dev*): 150

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

 151

Uyj majcych wygenerowanych plików .deb 152

Przez to, e live-boot i live-config s instalowane przez system live-build, instalacji pakietów w systemie gospodarza nie jest wystarczajca: naley traktowa wygenerowane pliki deb jak inne pakiety niestandardowe.. Poniewa z reguy celem budowania ze róda jest testowanie nowe rzeczy w krótkim okresie przed oficjaln premier, poinstruuji Instalowanie zmodyfikowanych paczek innych firm, aby tymczasowo umieci odpowiednie pliki w konfiguracji. W szczególności naley zauway, e oba pakiety s podzielone na rodzajowe czci, cz dokumentacji i jeden lub wicej 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 pakietów znajduj si w katalogu powyzej, te polecenia bash skopiuj wszystkie odpowiednie pakiety, w tym domylne dla nich back-endy: 153

```
$ cp ../live-boot{_,-initramfs-tools,-doc}*.deb config/packages.chroot/
$ cp ../live-config{_,-sysvinit,-doc}*.deb config/packages.chroot/
```

 154

## Ze zrzutów deweloperskich 155

Moesz pozwoli live-build automatycznie skorzysta z najnowszych zrzutów deweloperskich live-boot i live-live-config przez skonfigurowanie repozytorium pakietu live-systems.org jako repozytorium innych firm w katalogu konfiguracyjnym live-build. 156

---

# Podstawy

157

## Podstawy

158

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 a 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.

159

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.

160

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.

161

### Co to jest system live?

162

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).

163

With live systems, it's an operating system, built for one of the supported architectures (currently amd64 and i386). It is made from the following parts:

164

**Obraz jdra Linuxa**, zazwyczaj nazwany `mlinuz*`

165

**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.

166

**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).

167

**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.

168

You can use live-build to build the system image from your specifications, set up a

169

Linux kernel, its initrd, and a bootloader to run them, all in one medium-dependant format (ISO9660 image, disk image, etc.).

## Pobieranie prekompilowanych obrazów

170

While the focus of this manual is developing and building your own live images, you may simply wish to try one of our prebuilt images, either as an introduction to their use or instead of building your own. These images are built using our live-images git repository and official stable releases are published at <https://www.debian.org/CD/live/>. In addition, older and upcoming releases, and unofficial images containing non-free firmware and drivers are available at <http://live-systems.org/cdimage/release/>.

171

## Using the web live image builder

172

As a service to the community, we run a web-based live image builder service at <http://live-systems.org/build/>. This site is maintained on a best effort basis. That is, although we strive to keep it up-to-date and operational at all times, and do issue notices for significant operational outages, we cannot guarantee 100% availability or fast image building, and the service may occasionally have issues that take some time to resolve. If you have problems or questions about the service, please contact us, providing us with the link to your build.

173

## Uywanie i przestrogi dotyczące Web buildera

174

The web interface currently makes no provision to prevent the use of invalid combinations of options, and in particular, where changing an option would normally (i.e. using live-build directly) change defaults of other options listed in the web form, the web builder does not change these defaults. Most notably, if you change `-architectures` from the default `i386` to `amd64`, you must change the corresponding option `-linux-flavours` from the default `586` to `amd64`. See the `lb_config` man page for the version of live-build installed on the web builder for more details. The version number of live-build is listed at the bottom of the web builder page.

175

The time estimate given by the web builder is a crude estimate only and may not reflect how long your build actually takes. Nor is the estimate updated once it is displayed. Please be patient. Do not refresh the page you land on after submitting the build, as this will resubmit a new build with the same parameters. You should contact us if you don't receive notification of your build only once you are certain you've waited long enough and verified the notification e-mail did not get caught by your own e-mail spam filter.

176

The web builder is limited in the kinds of images it can build. This keeps it simple and efficient to use and maintain. If you would like to make customizations that are not provided for by the web interface, the rest of this manual explains how to build your own images using live-build.

177

## 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-i386.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.

## Korzystanie z hybrydowego obrazu ISO live

After either building or downloading an ISO hybrid image, which can be obtained at <https://www.debian.org/CD/live/>, 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.

## 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-i386.hybrid.iso
```

## 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-i386.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.

## Wykorzystanie przestrzeni pozostajej na noniku USB

After copying the `live-image-i386.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-i386.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.**



## Uruchamianie nonika live

206

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.

207

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.

208

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, such as a standard flavour prebuilt image, you should be automatically logged in on the console to the user account and see a shell prompt, ready to use.

209

## Uywanie wirtualnej maszyny do testowania

210

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:

211

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

212

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

213

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

214

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

215

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

216

## Testowanie obrazu ISO z uyciem QEMU

217

The most versatile VM in Debian is QEMU. If your processor has hardware support for virtualization, use the *qemu-kvm* package; the *qemu-kvm* package description briefly lists the requirements.

218

First, install *qemu-kvm* if your processor supports it. If not, install *qemu*, in which case the program name is *qemu* instead of *kvm* in the following examples. The *qemu-utils* package is also valuable for creating virtual disk images with *qemu-img*.

219

220

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

Uruchamianie obrazu ISO jest proste: 221

222

```
$ kvm -cdrom live-image-i386.hybrid.iso
```

Zobacz podręczniki man, aby uzyskać więcej szczegółów. 223

## Testowanie obrazu ISO z użyciem VirtualBox'a 224

W celu przetestowania ISO w *VirtualBox*'ie: 225

226

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

Create a new virtual machine, change the storage settings to use `live-image-i386.hybrid2.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. 228

229

```
$ 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. 230

231

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

## Budowanie i użycie obrazu HDD 232

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-i386.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. 233

**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): 234

235

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

Run the `lb config` command as before, except this time specifying the HDD image type: 236

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

A teraz zbuduj obraz używając polecenia `lb build`: 238

```
# lb build
```

When the build finishes, a `live-image-i386.img` file should be present in the current directory. 239

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-i386.img` instead of `live-image-i386.hybrid.iso`. 240

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-i386.img` as the first hard drive. 241

```
$ kvm -hda live-image-i386.img
```

## Budowanie obrazu netboot 242

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. 243

**Note:** if you performed any previous examples, you will need to clean up your working directory with the `lb clean` command: 244

```
# 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. 245

Run the `lb config` command as follows to configure your image for netbooting: 246

```
$ 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` 247

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-i386.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.

## Serwer 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;
}
```

## 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.

## 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 zainstalować pakiet `nfs-kernel-server`.

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

```
/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 <http://www.syslinux.org/wiki/index.php/PXELINUX> or the Debian Installer Manual's TFTP Net Booting section at <http://d-i.alioth.debian.org/manual/en.i386/ch04s05.html>. They might help, as their processes are very similar.

## 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 ułatwić sobie życie możemy użyć wirtualizacji.

## 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. 281

Uruchom qemu z "-net nic,vlan=0 -net tap,vlan=0,ifname=tun0" 282

## Webbooting 283

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. 284

## Getting the webboot files 285

As usual, you can build the images yourself or use the prebuilt files, which are available on the project's homepage at <http://live-systems.org/>. 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`. 286

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: 287

```
# 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 automatized. 288

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. 289

## Uruchamianie obrazów webboot 291

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. 292

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 293

way, it is possible to use the downloaded compressed filesystem as a regular live system. For example:

294

```
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.

295

---

# Przegląd narzędzi

296



## Przegląd narzędzi

297

Ten rozdział zawiera przegląd trzech głównych narzędzi stosowanych w budowie systemów live: live-build, live-boot i live-config. 298

### Pakiet live-build

299

live-build to zbiór skryptów do budowania systemów live. Skrypty te są również określane jako "polecenia". 300

Pomyślem stojącym za live-build jest bycie oprawy, która używa struktury katalogów jako konfiguracji, aby całkowicie zautomatyzować i dostosować wszystkie aspekty budowania obrazu live. 301

Wiele jest podobnych do tych używanych do budowania pakietów Debiana z użyciem *debhelper'a*: 302

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 ten sam sposób, live-build przechowuje swoją konfigurację w całości w podkatalogu *config/*. 303

Skrypty są niezależne - to znaczy, że zawsze jest bezpieczne uruchomienie poszczególnych poleceń. 304

W przeciwieństwie do *debhelper'a*, live-build zapewnia narzędzia do generowania szkieletów 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 pozostała część tego rozdziału omawia cztery najważniejsze polecenia. Należy zauważyć, że głównym wrapperem dla polecenia live-build jest *lb*. 305

**lb config:** Odpowiedzialny za inicjowanie katalogu konfiguracji systemu live. Zobacz Polecenie *lb config*, aby uzyskać więcej informacji. 306

**lb build:** Odpowiedzialny za rozpoczęcie kompilacji systemu live. Zobacz polecenie *lb build* aby uzyskać więcej informacji. 307

**lb clean:** Odpowiedzialny za czyszczenie kompilacji systemu live. Zobacz polecenie *lb clean* aby uzyskać więcej informacji. 308

### Polecenie *lb config*

309

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 może być czasochłonne i podatne na błąd, polecenie *lb config* może być używane do tworzenia początkowej konfiguracji drzewa katalogów. 310

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/*. 311

312

```
$ lb config
[2015-01-06 19:25:58] lb config
P: Creating config tree for a debian/stretch/i386 system
P: Symlinking hooks...
```

## Wcicia

313

Normalnie, pewnie będziesz chciał określić niektóre opcje. Na przykład, aby określić, jakiego menadżera pakietów chcesz użyć podczas budowania obrazu:

314

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

315

Jest możliwe ustalenie wielu opcji, takich jak:

316

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

317

Ponadto lista opcji dostępna jest w podręczniku man pakietu `lb_config`.

318

## Polecenie `lb build`

319

Polecenie `lb build` czyta konfigurację z katalogu `config/`. A następnie uruchamia polecenia niższego poziomu potrzebne do budowy Twojego systemu live.

320

## Polecenie `lb clean`

321

Zadaniem polecenia `lb clean`, jest to aby usunąć różne części kompilacji tak aby można było zacząć od czystego stanu. Domyślnie etapy `chroot`, `binary` and `source` są sprzątane, ale cache pozostaje nienaruszone. Ponadto, tylko poszczególne etapy mogą być oczyszczone. Na przykład, jeśli zostały wprowadzone zmiany, które wpływają tylko na etap binarny, należy użyć `lb clean --binary` przed budowaniem nowych plików binarnych. Jeśli zmiany unieważniają proces `bootstrap` i/lub zmieniają cache pakietów, np. po zmianie opcji `--mode`, `--architecture`, lub `--bootstrap`, trzeba użyć `lb clean --purge`. Zobacz podręcznik man pakietu `lb_clean` aby uzyskać listę wszystkich opcji.

322

## Pakiet `live-boot`

323

`live-boot` to zbiór skryptów zapewniających haki do `initramfs-tools`, wykorzystywane do wytwarzania plików `initramfs`, które są w stanie uruchomić system live, takich jak te stworzone przez `live-build`. Obejmuje to obrazy ISO systemów live, archiwa `netboot` i obrazy dysku USB.

324

W czasie rozruchu będzie szukał noników tylko do odczytu zawierających katalog `/live/`, gdzie jest przechowywany system plików (często skompresowany obraz systemu plików jak SquashFS). Jeśli znajdzie taki, stworzy zapisywalne środowisko, stosując `aufs`, dla systemów takich jak Debian, aby z niego wystartować.

325

Wiecej informacji na temat poczkowych plików ramfs w Debianie mona znale w Podręczniku Debiana Linux Kernel na <http://kernel-handbook.alioth.debian.org/> w rozdziale initramfs. 326

### **Pakiet live-config**

327

live-config zawiera skrypty, które 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. 328

---

# Zarządzanie konfiguracją

329

## Zarządzanie konfiguracją

330

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

331

### Radzenie sobie ze zmianami konfiguracji

332

Konfiguracje live rzadko są idealne na przy pierwszej próbie. Powinno się dodać opcję `lb config` z linii poleceń do wykonania pojedynczej kompilacji, ale bardziej typowe jest sprawdzenie tych opcji i kompilacja ponownie aż do uzyskania satysfakcji. Aby poradzić sobie z tymi zmianami, potrzeba automatycznych skryptów, które zapewnią, że konfiguracja przechowywana jest w stanie spójnym.

333

### Czemu używa się automatycznych skryptów? Co one robią?

334

Polecenie `lb config` zapisuje opcje, które są wprowadzane do plików w `#{config}/*#` oraz wielu innych opcjach 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 uruchomił `lb config` ponownie z nowymi wartościami dla `-binary-images`, wszelkie opcje zależne, które zostały przypisane jako domyślne dla poprzedniej dystrybucji mogą nie współgrać z nowym ustawieniem. Pliki te nie są przeznaczone do odczytu lub edycji. Przechowuj one wartości dla ponad stu opcji, więc nikt nie mówił 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 staremu, które nie są już aktualne.

335

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. Każdy 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.

336

### Użyj przykładowych automatycznych skryptów

337

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:

338

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

339

Edytuj `auto/config`, dodajc wszelkie opcje, jakie uwaasz. Przykadowo:

340

341

```
#!/bin/sh
lb config noauto \
  --architectures i386 \
  --linux-flavours 686-pae \
  --binary-images hdd \
  --mirror-bootstrap http://ftp.ch.debian.org/debian/ \
  --mirror-binary http://ftp.ch.debian.org/debian/ \
  "${@}"
```

Teraz, za kadym razem kiedy korzystasz z `lb config`, `auto/config` skasuje konfiguracj w oparciu o te opcje. Gdy chcesz wprowadzi zmiany do nich, naley edytowa opcje w tym pliku zamiast przekazywa je do `lb config`. Podczas korzystania z `lb clean`, `auto/clean` oczyci pliki w `config/*` wraz z innymi produktami kompilacji. I wreszcie, kiedy uywasz `lb build`, log kompilacji zostanie zapisany przez `auto/build` w `build.log`.

342

**Uwaga:** Specjalny parametr `noauto` jest uyty tutaj, aby powstrzyma kolejne za- 343  
pytania do `auto/config`, zapobiegajc w ten sposób nieskoczonej rekurencji. Upewnij  
si, e przypadkowo nie zosta usunity podczas dokonywania zmiany. Równie naley  
zadba o to aby podczas dzielenia polecenia `lb config` na wiele lini dla czytelności, jak  
pokazano w powyszym przykadzie, nie zapomnisz odwrotnego ukonika (   
) na kocu kadej linii, która kontynuuje polecenie w nastpnej linijce.

## Klonowanie konfiguracji opublikowanej przez Git

344

Uyj opcji `lb config --config` aby sklonowa repozytorium Git, który zawiera konfigu- 345  
racj systemu live. Jeli chcesz oprze swoj konfiguracj na jednej dostarczanej przez  
`${project}`, odwied <http://live-systems.org/gitweb/> i szukaj repozytorium o nazwie `live-images`  
(obrazy live) w kategorii Packages (pakiety). To repozytorium zawiera konfiguracje dla  
prekompilowanych obrazów systemów live.

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

346

347

```
$ mkdir live-images && cd live-images
$ lb config --config git://live-systems.org/git/live-images.git
$ cd images/standard
```

Edytuj `auto/config` i wszelkie inne rzeczy, których wymagasz do wasnych potrzeb 348  
w drzewie katalogów `config`. Na przykad, nie oficjalne prekompilowane obrazy  
tworzy si dodajc po prostu `--archive-areas "main contrib non-free"`.

Opcjonalnie mona zdefiniowa skrót w konfiguracji Git przez dodanie nastpujących 349  
opcji do ``${HOME}/.gitconfig`:

350

```
[url "git://live-systems.org/git/"]
  insteadOf = lso:
```

Dziki temu mona skorzysta z `lso`: wszdzie, gdzie trzeba poda adres repozytorium git 351

live-systems.org. Jeli również opuszczysz opcjonalny przyrostek `.git`, rozpoczynając nowy obraz przy użyciu tej konfiguracji jest to tak proste:

352

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

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

353

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

354

---

# Dostosowywanie zawartoci

355



## Opis dostosowywania

356

Ten rozdział zawiera przegląd różnych sposobów, w jaki można dostosować system live.

357

### Konfiguracja podczas kompilacji vs. podczas uruchamiania systemu

358

Opcje konfiguracji systemu live są podzielone na opcje w czasie budowania, które są stosowane w czasie kompilacji i na opcje w czasie rozruchu, które są stosowane podczas uruchamiania systemu. Opcje podczas uruchamiania są podzielone na te występujące wcześniej podczas uruchamiania, zastosowane przez live-boot, i na te, występujące później, zastosowane przez live-config. Każdy parametr rozruchu może zostać zmodyfikowany przez użytkownika poprzez ustalenie go podczas startu. Obraz może być również zbudowany z domylnymi parametrami startowymi, dzięki czemu użytkownicy mogą normalnie tylko uruchomić bezpośrednio system live bez podawania żadnych parametrów, gdy wszystkie opcje domyślne są odpowiednie. W szczególności argument `lb -bootappend-live` składa się z wszelkich opcji wiersza poleceń domyślnych dla jądra systemu live, takich jak `trwao` (ang. *persistence*), układ klawiatury, lub strefa czasowa. Zobacz Dostosowywanie lokalizacji i języka, dla przykładów.

359

Opcje konfiguracyjne w czasie budowania są opisane w podręczniku `man` na stronie `lb config`. Opcje konfiguracyjne w czasie rozruchu opisane są w podręczniku `man` na stronach `live-boot` i `live-config`. Choć pakiety startowe `live-boot` i `live-config` są zainstalowane w systemie live, który budujesz, zaleca się również zainstalować je w systemie budowania dla łatwego odniesienia podczas Twojej pracy przy konfiguracji. Jest to bezpieczne, ponieważ żadne z zawartych w nich skryptów nie będzie wykonywane, chyba że system zostanie skonfigurowany jako system live.

360

### Etapy kompilacji

361

Proces kompilacji jest podzielony na etapy, w każdym z nich z zastosowanymi w kolejności różnymi dostosowaniami. Pierwszym etapem do uruchomienia jest etap **bootstrap**. Jest to wstępna faza wypeniania katalogu `chroot` pakietami aby stworzyć katalog systemu Debian. Następnym etapem jest **chroot**, który kończy budowę katalogu `chroot`, wypeniania go wszystkimi pakietami wymienionymi w konfiguracji, wraz z innymi materiałami. Najwięcej dostosowywania zawartości odbywa się w tym etapie. Ostatnim etapem przygotowania obrazu live jest etap **binarny** (ang. *binary*), który tworzy moliwy do uruchomienia obraz, używając zawartości katalogu `chroot` do budowy głównego systemu plików w systemie live, a tym instalatora i wszelkich innych dodatkowych materiałów na noniku docelowym poza system plików na systemie live. Po skompilowaniu obrazu live, jeśli włączono, archiwum różdżki tarball jest budowane podczas etapu **source** (ang. *source*).

362

W każdym z tych etapów istnieje szczególna sekwencja, w której stosuje się polecenia. Są one używane w taki sposób, aby zapewnić modyfikacjom być ułożonym w rozstrzygnięty sposób. Na przykład, w etapie **chroot**, prekonfiguracja (ang. *preseeding*) jest stosowana, zanim zostaną zainstalowane jakiegokolwiek pakiety, pakiety są instalowane zanim jakiegokolwiek lokalnie zawarte pliki zostaną skopiowane, a haki są wprowadzane później, gdy wszystkie materiały są już na miejscu.

363

## Uzupenie lb config plikami

364

Mimo, że `lb config` tworzy konfigurację katalogów w `config/`, aby osiągnąć swoje cele, może być konieczne udostępnienie dodatkowych plików w podkatalogach `config/`. W zależności od tego, gdzie pliki są przechowywane w konfiguracji, mogą być skopiowane do systemu plików systemu live lub do binarnego obrazu systemu plików, lub mogą zostać zapewnione konfiguracje w czasie budowy systemu, które by były kopotliwie do przekazania jako opcje wiersza polecenia. Można zawrzeć rzeczy takie jak niestandardowe listy pakietów, niestandardowa grafika lub inny skrypt do uruchomienia zarówno w czasie kompilacji jak i w czasie startu systemu, zwiększając jego znaczną elastyczność `debian-live` swoim własnym kodem.

365

## Zadania dostosowywania

366

Kolejne rozdziały są podzielone na rodzaje zadań dostosowywania, które użytkownicy zazwyczaj wykonują: Dostosowywanie instalacji pakietu, Dostosowywanie zawartości i Dostosowywanie ustawień regionalnych i języka obejmują tylko niektóre z rzeczy, które możesz zrobić.

367

---

# Dostosowywanie instalacji pakietów

368

## Dostosowywanie instalacji pakietów

369

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.

370

### różnorodność pakietu

371

### Dystrybucja, działy archiwum i tryb

372

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` for the `testing` version of live-build. 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 behave as needed to build each supported distribution. For example, to build against the **unstable** release, `sid`, specify:

373

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

374

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.

375

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

376

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.

377

**Note:** The projects for whom these modes were added are primarily responsible for supporting users of these options. The `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.

378

## Serwery lustrzane dystrybucji

379

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.

380

## Serwery lustrzane dystrybucji używane podczas budowania obrazu

381

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.

382

383

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

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

## Serwery lustrzane dystrybucji użyte podczas uruchomienia

385

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 `httpredir.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.

386

387

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

## Dodatkowe repozytoria

388

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.

389

For example, `config/archives/live.list.chroot` allows you to install packages from

390

the debian-live snapshot repository at live system build time.

391

```
deb http://live-systems.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.

392

Jeeli takie pliki istnieją to zostaną wybrane automatycznie.

393

Należy również umieścić klucz GPG używany do podpisywania repozytorium w `config/archives/your-repository.gpg`.

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

395

## Wybieranie pakietów do instalacji

396

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.

397

### Lista pakietów

398

Listy pakietów są skutecznym sposobem wyrażenia, które pakiety powinny zostać zainstalowane. Składnia list obsługuje instrukcje warunkowe, które ułatwiają budowanie list i dostosowywanie ich do wykorzystania w wielu konfiguracjach. Nazwy pakietów mogą być także wstrzykiwane do listy za pomocą pomocników powłoki w czasie kompilacji.

399

**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.

400

### Używanie metapakietów

401

Najprostszym sposobem, aby wypisać swój list pakietów jest użycie metapakietu z dostarczanych przez dystrybucję. Na przykład:

402

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

403

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

404

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-
```

Oprócz tych, znajdziesz jeszcze inne metapakiety do różnych celów. Niektóre są podzbiórami 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
```

## 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.

**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.

## 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.

## 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
```

### Uywanie 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 przykład, aby zainstalować `ia32-libs` jeżeli wybrano `-architectures amd64`:

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

Można przetestować dowolny szereg wartości, na przykład zainstalować `memtest86+`, jeżeli 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 obsługiwane.

### 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.

## Pulpit i zadania językowe

430

Desktop and language tasks are special cases that need some extra planning 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.

431

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.

432

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:

433

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

434

## Rodzaj jdra i wersja

435

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.

436

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.

437

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:

438

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

## Niestandardowe jdra

440

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.

441

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.

442

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.

443

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:

444

- Uyj startowego dysku RAM.

445

- Include the union filesystem module (i.e. usually `aufs`).

446

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

447

## Instalowanie zmodyfikowanych pakietów lub pakietów innych firm

448

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.

449

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> may be of interest, however. The creation of bespoke packages is covered in the `Debian New Maintainers' Guide` at <https://www.debian.org/doc/maint-guide/> and elsewhere.

450

Istniej dwa sposoby instalacji niestandardowych pakietów:

451

- `packages.chroot`

452

- Uywanie niestandardowego repozytorium APT

453

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. 454

### Uywanie `packages.chroot` do instalacji niestandardowych pakietów 455

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. 456

Packages **must** be named in the prescribed way. One simple way to do this is to use `dpkg-name`. 457

Korzystanie z `packages.chroot` do instalacji niestandardowych pakietów ma wady: 458

- Nie jest możliwe użycie bezpiecznego APT. 459

- You must install all appropriate packages in the `config/packages.chroot/` directory. 460

- It does not lend itself to storing live system configurations in revision control. 461

### Uywanie repozytorium APT aby zainstalować niestandardowe pakiety 462

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. 463

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. 464

### Niestandardowe pakiety i APT 465

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. 466

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. 467

### Konfigurowanie APT podczas kompilacji 468

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. 469

## 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.

## Uywanie serwera proxy z APT

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

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

## 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 indeksów 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, such as `user-setup` which `live-config` recommends and is used to create the live user. 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" (Debian Policy Manual, section 7.2), some packages that users of your Live system actually need may be omitted. Therefore, we suggest you review the difference turning off recommends makes to your packages list (see the `binary.packages` file generated by `lb build`) and re-include in your list any missing packages that you still want installed. Alternatively, if you find you only want a small number of recommended packages left out, leave recommends enabled and set a negative APT pin priority on selected packages to prevent them from being installed, as explained in APT pinning.

### 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.

### 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 `testing` 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, `testing`. The following will accomplish this:

494

```
$ 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` but don't want the user prompted to store wifi passwords in the keyring. This meta-package 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`:

496

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

---

# Dostosowywanie zawartoci

497

## Dostosowywanie zawartoci

498

Ten rozdział omawia dokładną regulację dostosowywania zawartości systemu poza samym wybieraniem pakietów do wyboru, które będą zawarte. Uwzględnianie pozwala dodać lub wymienić dowolne pliki w obrazie systemu live, haki pozwalają na wykonanie dowolnego polecenia na różnych etapach produkcji oraz w czasie rozruchu a opcja pre-seeding pozwala skonfigurować pakiety, gdy są zainstalowane poprzez dostarczenie odpowiedzi na pytania debconfa.

499

## Uwzględnianie

500

Chociaż najlepiej byłoby gdyby system live zawierał się wyłącznie z plików dostarczonych przez całkowicie niemodyfikowane pakiety, to czasami wygodniejszym jest, dostarczenie lub zmodyfikowanie pewnych treści za pomocą plików. Korzystając z uwzględniania, można dodać (lub wymienić) dowolne pliki w systemie obrazu live. live-build dostarcza dwa takie mechanizmy wykorzystania:

501

Lokalne uwzględnianie chroot: Pozwoli Ci dodać lub zastąpić pliki w systemie plików chroot/live. Zobacz Lokalne uwzględnianie chroot/live, aby uzyskać więcej informacji.

502

Lokalne uwzględnianie danych binarnych: Pozwoli Ci dodać lub zastąpić pliki w obrazie binarnym. Zobacz Lokalne uwzględnianie danych binarnych, aby uzyskać więcej informacji.

503

Proszę zobaczyć Definicje aby uzyskać więcej informacji na temat różnicy między obrazami "binarnymi" a obrazami "live".

504

## Lokalnie uwzględniane w chroot/live

505

Lokalnie uwzględnione w chroot mogą być używane, aby dodać lub zastąpić pliki w systemie plików chroot/live, tak aby mogły być one użyte w systemie live. Typowym zastosowaniem jest dostarczenie szkieletowego katalogu użytkownika (/etc/skel) używanego przez system live do tworzenia katalogu domowego użytkownika live. Innym jest dostarczanie plików konfiguracyjnych, które mogą być po prostu dodane lub podmienione w obrazie bez przetwarzania; zobacz Lokalne haki chroot/live, czy potrzebne jest przetwarzanie.

506

Aby doczytać pliki, po prostu dodaj je do katalogu config/includes.chroot. Ten katalog odnosi się do katalogu root / systemu live. Na przykład, aby dodać plik /var/www/index.html do systemu live, użyj:

507

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

508

Konfiguracja będzie mieć następujący układ:

509

```
- config
  [...]
  |-- includes.chroot
  | `-- var
```

510



```
|          |-- www
|          |-- index.html
[...]
```

Lokalnie uwzględnione w chroot są instalowane po instalacji pakietów, tak jak pliki zainstalowane przez pakiety są zastępowane. 511

### Lokalnie uwzględniane dane binarne 512

Aby zawiera materiały takie jak filmy lub dokumentacja na systemie plików nonika tak, aby były one dostępne od razu po woeniu nonika bez uruchamiania systemu live, możesz użyć lokalnego uwzględniania danych binarnych. Działanie to w podobny sposób do lokalnego uwzględniania w chroot/live. Załóżmy na przykład, że pliki `/video_demo.*` to filmy demonstracyjne systemu live opisanego przez i dowiązane przez stron indeksu HTML. Wystarczy skopiować materiały z `config/includes.binary/` w następujący sposób: 513

```
$ cp ~/video_demo.* config/includes.binary/ 514
```

Pliki te pojawią się teraz w katalogu głównym nonika live. 515

### Haki 516

Haki pozwalają poleceniom aby były wykonywane w etapach chroot i binarnym kompilacji w celu dostosowania obrazu. 517

### Lokalne haki chroot/live 518

Aby uruchomić komendy na etapie chroot, należy utworzyć skrypt hak z przyrostkiem `.hook.chroot` zawierającym polecenia w katalogu `config/hooks/`. Hak będzie działał w chroot po resztę konfiguracji chroot zastosowano, więc pamiętaj, aby zapewnić, że konfiguracja zawiera wszystkie pakiety i pliki do Twoich potrzeb haka w celu uruchomienia. Zobacz przykładowe skrypty hak chroot dla różnych zadań dostosowywania wspólnej chroot przewidzianych w `# {/usr/share/doc/live-build/przyklady/haki} #`, który można skopiować lub linku do korzystania z nich w swoim własnym konfiguracji. 519

### Haki podczas uruchamiania 520

Aby wykonać polecenia przy starcie systemu, możesz dostarczyć haki live-config, jak wyjątkowo w jego podręczniku man w sekcji "Customization" (ang. dostosowywanie). Przeanalizuj własne haki live-config dostarczone w `/lib/live/config/`, zwracając uwagę na numery sekwencji. Następnie dodaj swój własny hak z odpowiednim prefiksem numeru sekwencji, albo jako lokalnie uwzględnione w chroot w `config/includes.chroot/lib/live/conf` lub w postaci niestandardowego pakietu omówione w Instalowanie zmodyfikowanych pakietów lub pakietów innych firm. 521

## Lokalne haki binarne

522

Aby uruchomić komendy na etapie binarnym, należy utworzyć skrypt hak z przyrostkiem `.hook.binary` zawierającym polecenia w katalogu `config/hooks/`. Hak będzie uruchomiony po wszystkich innych binarnych poleceniach, ale przed `binary_checksums`; ostatnim poleceniem. Polecenia haka nie działają w środowisku `chroot`, więc należy zadbać, aby nie modyfikować żadnych plików poza drzewem kompilacji albo może to uszkodzić system kompilacji! Zobacz przykładowe skrypty binarne dla różnych typowych binarnych zadań dostosowywania przewidzianych w `/usr/share/doc/live-build/examples/hooks`, które można skopiować lub utworzyć dowiązanie symboliczne, aby skorzystać z nich w swojej własnej konfiguracji.

523

## Wstępne ustawienie pyta Debconfa (Preseeding)

524

Pliki w katalogu `config/preseed/` z rozszerzeniem `.cfg` w następującym etapie (`.chroot` lub `.binary`) są brane pod uwagę jako pliki preseed debconfa i są instalowane za pomocą `live-build` przez `debconf-set-selections` podczas odpowiedniego etapu.

525

Aby uzyskać więcej informacji o debconf, proszę przeczytać `debconf(7)` z pakietu `debconf`.

526

---

# **Dostosowywanie zdarze podczas uruchamiania systemu**

527

## Dostosowywanie zdarze podczas uruchamiania systemu

528

Caa konfiguracja, która odbywa si w czasie pracy systemu jest wykonywana przez live-config. Oto niektóre z najbardziej popularnych opcji live-config, którymi mog by zainteresowani uytkownicy. Pen list wszystkich moliwoci mona znale w podręczniku man pakietu live-config.

529

### Personalizacja uytkownika live

530

Jednym wanym czynnikiem jest to, e uytkownik jest tworzony przez live-boot w czasie startu systemu, a nie live-build w czasie kompilacji. To wpywa nie tylko, na to gdzie materiaiy dotyczce uytkownika live s wprowadzone w kompilacji, jak to opisano w Uwzgldnianie lokalne Live/chroot, ale równie na wszelkie grupyi uprawnienia zwizane z uytkownikiem live.

531

Mona okreli dodatkowe grupy, do których uytkownik live bdzie nalee korzystajc z jednej z moliwoci, aby skonfigurowa live-config. Na przykad, aby doda uytkownika live do grupy fuse, mona doda nastpujcy plik w config/includes.chroot/etc/live/config/user-setup.conf

532

```
LIVE_USER_DEFAULT_GROUPS="audio cdrom dip floppy video plugdev netdev powerdev scanner  
bluetooth fuse"
```

533

lub uyj live-config.user-default-groups=audio,cdrom,dip,floppy,video,plugdev,netdev,powerdev,scanner jako parametru startowego.

Moliwe jest równie, aby zmieni domyln nazw uytkownika "user" i domylne haso "live". Jeli chcesz to zrobi, z jakiegokolwiek powodu, mona to atwo osign w nastpujcy sposób:

535

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

536

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

537

Jednym z moliwych sposobów 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/

538

### Ustawianie lokalizacji i jzyka

539

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

540

generowanie plików lokalizacji

541

ustawienie konfiguracji klawiatury

542

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

543

```
$ lb config --bootappend-live "boot=live components locales=de_CH.UTF-8"
```

Wiele lokalizacji może być określone w postaci listy rozdzielonej przecinkami.

Parametr ten, jak również parametr konfiguracyjny klawiatury jak wskazano poniżej, może być również używany w linii poleceń jądra. Można określić ustawienia regionalne poprzez `language_country` (w tym przypadku używane jest kodowanie domyślne) lub pełną nazwę z kodowaniem `language_country.encoding`. Lista obsługiwanych lokalizacji i kodowa może być znaleziona w `/usr/share/i18n/SUPPORTED`.

Zarówno konfiguracja konsoli i klawiatury X wykonywana jest przez `live-config` przy pomocy pakietu `console-setup`. Aby je skonfigurować, ustaw parametry startowe `keyboard-layouts`, `keyboard-variants`, `keyboard-options` i `keyboard-model` przez opcję `--bootappend-live`. Prawidłowe wartości opcji można znaleźć w `/usr/share/X11/xkb/rules/base.lst`. Aby znaleźć układy i warianty klawiatury dla danego języka, spróbuj wyszukać nazw w języku angielskim i/lub kraju, gdzie mówi się danym językiem, np.:

```
$ egrep -i '(^!|german.*switzerland)' /usr/share/X11/xkb/rules/base.lst
! model
! layout
  ch          German (Switzerland)
! variant
  legacy      ch: German (Switzerland, legacy)
  de_noadkeys ch: German (Switzerland, eliminate dead keys)
  de_sundeadkeys ch: German (Switzerland, Sun dead keys)
  de_mac      ch: German (Switzerland, Macintosh)
! option
```

Należy pamiętać, że każdy wariant wymienia układ, którego dotyczy w opisie.

Często tylko układ klawiatury musi być skonfigurowany. Na przykład, aby uzyskać listę plików lokalizacyjnych dla niemieckiego i szwajcarskiego niemieckiego układu klawiatury w systemie X użyj:

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

Jednak dla bardzo konkretnych przypadków użycia, można dodać inne parametry. Na przykład, aby ustawić francuski system z układem klawiatury `French-Dvorak` (zwanym `Bepo`) na klawiaturze typu `USB TypeMatrix EZ-Reach 2030`, użyj:

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

Wiele wartości oddzielonych przecinkami może być przypisane do każdego z parametrów `keyboard-*`, z wyjątkiem `keyboard-model`, który przyjmuje tylko jedną wartość. Proszę przeczytać podręcznik `man keyboard(5)` aby uzyskać więcej szczegółów i przykładów zmiennych `XKBMODEL`, `XKBLAYOUT`, `XKBVARIANT` i `XKBOPTIONS`. Jeśli podano wiele `keyboard-variants` (ang. warianty klawiatury), będą one dopasowane jeden do drugiego przez wartość `keyboard-layouts`

(patrz opcja `setxkbmap(1) -variant`). Puste wartości są dozwolone; na przykład aby zdefiniować dwa układy, domyślny US QWERTY oraz drugi US Dvorak, zastosuj:

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

555

## Persistence

556

Odmian Live CD jest preinstalowany system, który uruchamia się z nośników tylko do odczytu, takich jak CD-ROM, gdzie operacje zapisu i modyfikacje nie przetrwają restartów sprzętowych hosta, na którym jest uruchomiony.

557

System live jest uogólnieniem tego paradygmatu, a tym samym wspiera media inne prócz płyt CD; ale dalej jako jego domyślne zachowanie, należy uważać operacje tylko do odczytu a wszelkie zmiany, w czasie działania są tracone podczas zamykania systemu.

558

'Persistence' (ang. trwałość) to wspólna nazwa dla różnych rodzajów rozwiązań dla zapisania niektórych lub wszystkich danych między restartami podczas używania i wprowadzania zmian do systemu. Aby zrozumieć, jak to działa to przydatna byłaby wiedza, a nawet wtedy, gdy system jest uruchamiany i działa z nośnika tylko do odczytu, to modyfikacje plików i katalogów są zapisywane na zapisywalnych nośnikach, typowo dysk RAM (tmpfs) a dane w pamięci RAM nie przetrwają restartu.

559

Dane przechowywane na tym ram dysku powinny być przechowywane na zapisywalnym trwałym nośniku takim jak lokalny dysk, lokalny udział sieciowy lub nawet na sesji wielosesyjnego dysku CD-RW/DVD-RW. Wszystkie te nośniki są obsługiwane w systemach live na różne sposoby i wszystkie, oprócz ostatniej wymagają specjalnego parametru startowego określonego w czasie startu systemu: `persistence`.

560

Jeli ustawiony jest parametr startowy `persistence` (a `nopersistence` nie jest ustawiony), lokalne nośniki (np. dyski twarde, napędy USB) będą przeszukane w celu znalezienia woluminów trwałości podczas startu systemu. Możliwe jest ograniczenie, jakiego typu woluminy trwałości będą wykorzystane przez określenie pewnych parametrów startowych opisanych w podręczniku `man live-boot(7)`. Wolumin trwałości może być każdym z wymienionych:

561

partycja, identyfikowana po nazwie GPT.

562

system plików, identyfikowany po etykiecie.

563

plik obrazu zlokalizowany na każdym obsługiwanym systemie plików (nawet na partycji NTFS innego systemu), identyfikowany po nazwie pliku.

564

Etykieta dla woluminu musi być `persistence`, ale będzie ignorowana, dopóki nie będzie zawarty w katalogu głównym plik o nazwie `persistence.conf`, który jest używany by poinformować o woluminie `persistence`, to znaczy, że wskazuje się w nim katalogi, w których chcesz zapisać na woluminie zmiany przy restarcie. Zobacz Plik `persistence.conf`, aby uzyskać więcej szczegółów.

565

Oto kilka przykładów, jak przygotować wolumin, aby mógł być użyty z opcją `persistence`. Może to być, na przykład, partycja ext4 na dysku twardym lub na nośniku wymiennym stworzona przez, np.:

566

567

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

Zobacz również Wykorzystanie przestrzeni pozostałej na noniku USB.

568

Jeli masz już partycję na urządzeniu, można po prostu zmienić jej etykietę używając następującego polecenia:

569

570

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

Oto przykład, jak stworzyć plik obrazu opartego na ext4 do zastosowania z opcją persistence:

571

572

```
$ 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 przykład, aby sprawić by katalog /usr był przezroczysty, ale tylko zapisywałyby zmiany wprowadzone w tym katalogu, a nie całe zawartość /usr, można użyć opcji "union". Jeli plik obrazu znajduje się w katalogu domowym, należy skopiować go do katalogu głównego systemu plików na dysku twardym i zamontować go w /mnt w następujący sposób:

573

574

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

Następnie utwórz plik persistence.conf dodając zawartość i odmontuj plik obrazu.

575

576

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

Teraz uruchom ponownie i wybierz nonik live, a następnie uruchom dodając parametr startowy "persistence".

577

### Plik persistence.conf

578

Partycję z etykietą persistence należy skonfigurować za pomocą pliku persistence.conf, aby dowolne katalogi stały się trwałe. Ten plik, znajdujący się w głównym katalogu systemu plików partycji, kontroluje, które katalogi są trwałe i w jaki sposób.

579

To jak niestandardowe wierzchnie zamontowania są skonfigurowane jest opisane w szczegółach w podręczniku man persistence.conf(5), ale ten prosty przykład powinien być wystarczający dla większości zastosowań. Powiedzmy, że chcemy, aby nasz katalog domowy i cache APT były trwałe w pamięci podręcznej systemu plików ext4 na partycji /dev/sdb1:

580

581

```
# 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
```

Następnie uruchamiamy ponownie komputer. Podczas pierwszego uruchomienia zawartość `/home` i `/var/cache/apt` zostanie skopiowana do woluminu trwałości i od tej pory wszystkie zmiany w tych katalogach będą przechowywane na tym woluminie. Należy pamiętać, że wszelkie cieciki wymienione w pliku `persistence.conf` nie mogą zawierać spacji lub specjalnych komponentów cieciki: `.` i `..`. Ponadto, ani `/lib`, `/lib/live` (lub którykolwiek z jego podkatalogów) ani `/` nie może zostać utrwalony za pomocą wasnych punktów montowania. Jako obejście tego ograniczenia można dodać `/ union` do pliku `persistence.conf` w celu osiągnięcia pełnej trwałości. 582

### Używanie więcej niż jednego magazynu persistence 583

Istnieją różne sposoby korzystania z wielu magazynów trwałości (ang. persistence) dla różnych zastosowań. Na przykład, przy użyciu kilku magazynów w tym samym czasie lub wybranie tylko jednego, spośród różnych, do bardzo specyficznych zastosowań. 584

Kilka różnych niestandardowych woluminów-nakadek (z waszymi plikami `persistence.conf`) może być używane w tym samym czasie, ale jeżeli kilka woluminów tworzy ten sam katalog trwały, będzie używany tylko jeden z nich. Jeśli jakie dwa punkty montowania są "zagniedzone" (np. jeden jest podkatalogiem drugiego) to katalog rodzic (ang. rodzic) zostanie zamontowany przed katalogiem child (ang. dziecko) tak że jeden punkt montowania nie będzie ukryty przed innym. Zagniedzone niestandardowe woluminy są problematyczne, jeżeli są wymienione w tym samym pliku `persistence.conf`. Zobacz podręcznik `man persistence.conf(5)`, jak radzi sobie z tym przypadkiem, jeśli naprawd potrzebujesz (Wskazówka: zazwyczaj nie potrzebujesz). 585

Jednym z możliwych przypadków użycia: Jeśli chcesz przechowywać dane użytkownika np. `/home` i dane superużytkownika tj. `/root` na różnych partycjach, utwórz dwie partycje z etykiet `persistence` i dodaj plik `persistence.conf` na każdej z nich, tak jak `# echo "/home" > persistence.conf` na pierwszej partycji, przez co będą zapisywane pliki użytkownika i `# echo "/root" > persistence.conf` na drugiej partycji, która będzie przechowywać pliki superużytkownika. Wreszcie, należy użyć parametru startowego `persistence`. 586

Jeśli użytkownik będzie potrzebował wiele magazynów trwałości tego samego typu dla różnych miejsc lub dla celów testowych, takich jak magazyny `private` i `work` parametr startowy `persistence-label` użyty w połączeniu z parametrem `persistence` pozwoli na wiele unikatowych magazynów trwałości. Przykładem może być, jeśli użytkownik chciałby użyć partycji trwałości oznaczonej `private` dla prywatnych danych, takich jak zakładki w przeglądarce lub innych typów danych, to mógłby użyć parametrów startowych: `persistence persistence-label=private`. A do przechowywania danych związanych z pracą, takich jak dokumenty, projekty badawcze lub inne rodzaje, mógłby skorzystać z parametrów startowych: `persistence persistence-label=work`. 587

Ważne jest, aby pamiętać, że każda z tych partycji, `private` i `work`, także potrzebuje pliku 588



`persistence.conf`. Podręcznik `man` pakietu `live-boot` zawiera więcej informacji o tym, jak korzystać z tych etykiet z zapisanymi nazwami.

## Using persistence with encryption

589

Korzystanie z funkcji trwałości (ang. `persistence`) oznacza, że niektóre poufne dane mogą zostać narażone na ryzyko. Zwłaszcza jeśli trwałe dane są przechowywane na urządzeniu przenośnym, takim jak pamięć USB lub zewnętrzne dyski twarde. To jest miejsce, gdzie przydatne staje się szyfrowanie. Nawet jeśli cała procedura może wydawać się skomplikowana, ze względu na liczbę kroków, które należy podjąć, to jest bardzo łatwo obsłużyć szyfrowane partycje z `live-boot`. Aby móc korzystać z **luks**, który jest obsługiwany typem szyfrowania, musisz zainstalować `cryptsetup` zarówno na maszynie tworzenia zaszyfrowanych partycji, a także w systemie `live`, który będzie używał szyfrowanej trwałości partycji.

590

Aby zainstalować `cryptsetup` na twoim komputerze:

591

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

592

Aby zainstalować `cryptsetup` na twoim systemie `live` dodaj go do listy pakietów:

593

```
$ lb config
$ echo "cryptsetup" > config/package-lists/encryption.list.chroot
```

594

Gdy Twój system `live` posiada `cryptsetup`, to w zasadzie wystarczy, że tylko utworzysz nową partycję, zaszyfrujesz ją i uruchomisz z parametrami `persistence` i `persistence-encryption=luks`. Mogłobyś przewidzieć ten krok i dodać parametry startowe w czasie kompilacji przestrzegając następującej procedury:

595

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

596

Przejdźmy do szczegółów dla tych wszystkich, którzy nie są zaznajomieni z szyfrowaniem. W poniższym przykładzie mamy zamiar użyć partycji na dysku USB, która odpowiada `/dev/sdc2`. Należy zaznaczyć, że należy ustalić, która partycja jest jedną z tych, które masz zamiar użyć w tym konkretnym przypadku.

597

Pierwszym krokiem jest podłączenie dysku USB i określenie, którym jest urządzeniem. Zalecaną metodą tworzenia listy urządzeń w `live-manual` jest `ls -l /dev/disk/by-id`. Następnie utworzymy nową partycję, a następnie zaszyfrujemy ją hasłem w następujący sposób:

598

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

599

Następnie otwieramy partycję LUKS w wirtualnym elemencie odwzorującym urządzenie `/dev/mapper`. Można tu użyć dowolnej nazwy. Używamy **live** jako przykład:

600

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

601

Nastpnym krokiem jest wypenienie urzdzenia zerami przed utworzeniem systemu plików: 602

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

 603

Teraz jestemy gotowi do stworzenia systemu plików. Warto zauway, e dodajemy etykiet persistence tak, aby urzdzenie zotao zamontowane w jako persistence store (magazyn persistence) w czasie startu systemu. 604

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

 605

Aby kontynuowa nasz konfiguracj, musimy zamontowa urzdzenie, na przykad w /mnt. 606

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

 607

I stwórz plik persistence.conf w katalogu gównym partycji. To jest, jak wyjaniono wyej, absolutnie konieczne. Zobacz Plik persistence.conf. 608

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

 609

Potem odmontuj punkt montowania: 610

```
# umount /mnt
```

 611

I opcjonalnie, cho moe to by dobry sposób na zabezpieczenie danych, które wanie dodalimy do partycji, moemy zamkn urzdzenie: 612

```
# cryptsetup luksClose live
```

 613

Podsumujmy proces. Do tej pory stworzylimy system live z moliwoci szyfrowania, który mona skopiowa na nonik usb, jak wyjaniono w kopiowaniu hybrydowego obrazu ISO na nonik pamici USB. Stworzylimy równie zaszyfrowan partycj, która 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. 614

---

# Dostosowywanie obrazu binarnego

615

## Dostosowywanie obrazu binarnego

616

### Programy adujce (ang. Bootloadery)

617

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

618

W celu wykorzystania tego 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 adujcego (ang. bootloader), tylko zapewnienie lokalnego zmodyfikowanego kopii jednego z typu programów, np. `* {isolinux}` \* w `# {config / programy adujce / isolinux} #` wystarczy to, w zależności od przypadku użycia.

619

Podczas modyfikacji jednego z domylnych motywów, jeśli chcesz korzystać z indywidualnego obrazu ta, który będzie wyświetlany razem z menu startowym, dodaj obraz `splash.png` 640x480 pikseli. Następnie usu plik `splash.svg`.

620

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

621

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

622

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

623

### Metadane ISO

624

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

625

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

626

`LB_ISO_PREPARER/--iso-preparer NAZWA`: Pole to powinno opisywać przygotowującego obraz, zwykle z kilkoma danymi kontaktowymi. Domyślną wartością tej opcji jest używana wersja live-build, które może później pomóc przy debugowaniu. Maksymalna długość tego pola wynosi 128 znaków.

627

`LB_ISO_PUBLISHER/--iso-publisher NAZWA`: Pole to powinno opisywać wydawcę obrazu, zwykle z kilkoma danymi kontaktowymi. Maksymalna długość tego pola wynosi 128 znaków.

628

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. 629

---

# Dostosowywanie Instalatora Debiana

630

## Dostosowywanie Instalatora Debiana

631

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

632

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

633

### Typy Instalatora Debiana

634

Trzy główne rodzaje instalatora to:

635

**Instalator Debiana "Normal":** To jest normalny obraz systemu live z oddzielnym jądrem i initrd, który (gdy wybrany z odpowiedniego menu programu adjucego) 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).

636

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.

637

Cały 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.

638

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

639

Instalacja będzie przebiegała 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 specjalnemu pakietowi udeb zwanemu live-installer.

640

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

641

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

642

**Instalator Debiana "Desktop":** Niezależnie od wybranego rodzaju Instalatora Debiana, jeśli 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.

643

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

644

jdro systemu live musi by zgodne z jdrem wykorzystywanym przez d-i dla okrelonej architektury. Na przykad:

645

```
$ lb config --architectures i386 --linux-flavours 586 \  
--debian-installer live  
$ echo debian-installer-launcher >> config/package-lists/my.list.chroot
```

## Dostosowywanie Instalatora Debiana przez preseeding

646

Jak opisano w Podręczniku Instalatora Debiana, Zacznik B na <https://www.debian.org/releases/stable/i386/ap>, "Plik preseed dostarcza sposób, aby ustawi odpowiedzi na pytania zadawane w trakcie procesu instalacji, bez konieczności ręcznego wprowadzania odpowiedzi, podczas instalacji. Pozwala to w pełni zautomatyzować większość rodzajów instalacji, a nawet oferuje kilka funkcji niedostępnych w zwykłych instalacjach." Takie dostosowanie jest najlepiej osiągnięte używając live-build poprzez umieszczenie konfiguracji w pliku preseed.cfg zawartym w config/includes.installer/. Na przykład, aby ustawi lokalizację na en\_US:

648

```
$ echo "d-i debian-installer/locale string en_US" \  
>> config/includes.installer/preseed.cfg
```

## Dostosowywanie zawartości Instalatora Debiana

649

Do celów dowiadczalnych lub debugowania, możesz zechcieć doczy lokalnie zbudowany element di w paczce udeb. Należy umieścić je w config/packages.binary/, aby wczytać je do obrazu. Również dodatkowe lub zamiennie pliki i katalogi mogą być zawarte w inittrd instalatora, w sposób podobny do uwzględniania lokalnie live/chroot, poprzez umieszczenie materiałów w config/includes.installer/.

650



---

# Projekt

651

---

# Wnoszenie wkładu do tego projektu

652

## Wnoszenie wkładu do tego projektu

653

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.

654

Wkład do projektu, taki jak tłumaczenia i poprawki, są bardzo mile widziane. Każdy może bezpośrednio zaangażować się w repozytoria, jednak prosimy wysłać większe zmiany do listy mailingowej, aby omówić je w pierwszej kolejności. Patrz rozdział kontakt aby uzyskać więcej informacji.

655

Projekt używa Git jako systemu kontroli wersji i zarządzania kodem źródłowym. Jak wyjaśniono w repozytorium Git są dwie główne gałęzie rozwoju: **debian** i **debian-next**. Każdy może wprowadzić zmiany do gałęzi **debian-next** w repozytoriach live-boot, live-build, live-config, live-images, live-manual i live-tools.

656

Jednakże istnieją pewne restrykcje. Serwer odrzuci:

657

- Non fast-forward pushes.

658

- Zmiany wymagające scalenia.

659

- Dodawanie lub usuwanie tagów lub gałęzi rozwojowych.

660

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

661

Pisz komentarze do zmian, które składają się z pełnych, 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. ...".)

662

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

663

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

664

## Wprowadzanie zmian

665

W celu wysłania zmian do repozytoriów, należy wykonać następujące procedury. Tutaj używamy live-manual jako przykładu, więc zastąp go nazwą repozytorium, z którym chcesz pracować. Aby uzyskać szczegółowe informacje na temat edycji podręcznika live-manual zobacz: Współtworzenie tego dokumentu.

666

Pobierz publiczny klucz do wprowadzania zmian:

667

668

```
$ mkdir -p ~/.ssh/keys
$ wget http://live-systems.org/other/keys/git@live-systems.org -O ~/.ssh/keys/git@live-systems.org
```

```
$ wget http://live-systems.org/other/keys/git@live-systems.org.pub -O ~/.ssh/keys/git@live-systems.org.pub
$ chmod 0600 ~/.ssh/keys/git@live-systems.org*
```

Dodaj następującą sekcję do twojej konfiguracji klienta openssh:

669

670

```
$ cat >> ~/.ssh/config << EOF
Host live-systems.org
  Hostname live-systems.org
  User git
  IdentitiesOnly yes
  IdentityFile ~/.ssh/keys/git@live-systems.org
EOF
```

Sprawdź i sklonuj kopię live-manual przez ssh:

671

672

```
$ git clone git@live-systems.org:/live-manual.git
$ cd live-manual && git checkout debian-next
```

Upewnij się, że masz ustawionego autora i adres email w konfiguracji Git:

673

674

```
$ git config user.name "John Doe"
$ git config user.email john@example.org
```

**Ważne:** Pamiętaj, że powinno się wprowadzać wszelkie zmiany wyłącznie w gałęzi **debian-next**.

675

Wprowadź zmiany. W tym przykładzie najpierw napisz nowy działający dotyczący stosowania plastrów, a następnie przygotuj się do poproszenia o dodanie plików i napisanie poproszenia wiadomo tak:

676

677

```
$ git commit -a -m "Adding a section on applying patches."
```

Wyślij poprawki na serwer:

678

679

```
$ git push
```

---

# Zgaszanie błędów

680

## Zgłaszanie błędów

681

Systemy live są dalekie od doskonałości, ale chcemy, uczyni je jak najbliższe perfekcji - z Waszą pomocą. Nie wahać się zgłaszać błędów. Lepiej jest wypisać raport dwa razy niż wcale. Ten rozdział zawiera zalecenia dotyczące sposobu składania raportów o błędach.

682

Dla niecierpliwych:

683

Zawsze należy sprawdzić najpierw aktualizacje statusu obrazu na naszej stronie internetowej <http://live-systems.org/>, w poszukiwaniu znanych problemów.

684

Przed wysłaniem raportu o błędzie zawsze spróbuj odtworzyć problem z **najnowszą wersją** gązi live-build, live-boot, live-config i live-tools, których używasz (jak najnowszą wersją 4.x live-build, jeśli używasz live-build 4).

685

Spróbuj podać **jak najwięcej specyficznych informacji, jak to tylko możliwe** o błędzie. Obejmuje to (co najmniej) wersje używanych live-build, live-boot, live-config i live-tools oraz dystrybucję systemu live którą kompilujesz.

686

## Znane problemy

687

Ponieważ Debian **testing** \* i Debian **unstable** dystrybucjami ruchomymi, w przypadku określenia jednej z nich jako dystrybucji systemu docelowego, kompilowanie nie zawsze zakończy się sukcesem.

688

Jeśli budowanie systemu opartego na **testing** lub **unstable** powoduje u Ciebie zbyt wiele trudności, raczej wykorzystaj wydanie **stable**. live-build zawsze ustawia domyślnie wydanie **stable**.

689

Obecnie znane problemy wymienione są w ramach sekcji "status" na naszej stronie internetowej w <http://live-systems.org/>.

690

To jest poza zakresem tej instrukcji, aby szkolić się poprawnie identyfikować i naprawiać problemy pakietów z dystrybucji rozwojowych, jednak istnieją dwie rzeczy, których zawsze można spróbować: Jeśli kompilowanie nie powiedzie się, gdy docelową dystrybucją jest **testing**, to spróbuj z **unstable**. Jeśli **unstable** również nie działa, to przywróć do dystrybucji **testing** i przypnij nowszą wersję wadliwego pakietu z wersji **unstable** (patrz Przypinanie APT aby uzyskać szczegóły).

691

## Przebuduj od zera

692

Aby upewnić się, że dany błąd nie jest spowodowany nie w pełni wyczyszczonym budowanym systemem, prosz zawsze odbudować cały system live od podstaw, aby sprawdzić, czy błąd jest powtarzalny.

693

## Używaj aktualnych pakietów

694

Używanie przestarzałych pakietów może powodować powstanie problemów podczas próby odtworzenia (i ostatecznego rozwiązania) problemu. Upewnij się, że system na którym kompilujesz jest aktualny i wszelkie pakiety zawarte w obrazie również.

695

**Zbierz potrzebne informacje**

696

Prosz dostarczy wystarczając ilo informacji z raportem. Obejmujce co najmniej, dokadn 697  
wersj live-build, gdzie napotkano bd i kroki, jak go odtworzy. Prosz uy zdrowego rozs-  
zdku i przedstawi wszelkie inne istotne informacje, jeli uwaasz, e moe to pomóc w  
rozwizaniu problemu.

Aby w peni wykorzysta Twój raport o bdzie, bdziemy potrzebowa przynajmniej nast- 698  
pujących informacji:

Architektura systemu hosta 699

Dystrybucja systemu hosta 700

Wersja live-build na systemie hosta 701

Version of *debootstrap* on the host system 702

Architektura systemu live 703

Dystrybucja systemu live 704

Wersja live-boot na systemie live 705

Wersja live-config na systemie live 706

Wersja live-tools na systemie live 707

Mona wygenerowa log procesu kompilacji przy uyciu polecenia `tee`. Polecamy robi 708  
to automatycznie przy uyciu skryptu `auto/build` (patrz Zarzdzanie konfiguracj w celu  
uzyskania szczegóów).

709

```
# lb build 2>&1 | tee build.log
```

W czasie uruchamiania live-boot i live-config przetrzymuj swoje logi w `/var/log/live/`. 710  
Sprawd je w poszukiwaniu bdów.

Dodatkowo, aby wykluczy inne bdy, zawsze dobrym pomyslem jest, aby spakowa do 711  
archiwum tar swój katalog `config/` i przesa go gdzie (**nie** naley wysa go jako zacznik  
do listy), tak aby mona spróbowo odtworzy napotkane bdy. Jeli sprawia to trudno (np.  
ze wzgldu na rozmiar) mona uy wyjcia z polecenia `lb config -dump`, które tworzy pod-  
sumowanie drzewa konfiguracyjnego (czyli np. list plików w podkatalogach `config/`,  
ale bez zaczania ich).

Pamitaj, aby wysa wszystkie dzienniki i logi, które zostay wyprodukowane z ustawie 712  
regionalnych angielskich, np. uruchom polecenia live-build ze zmiennymi `LC_ALL=C`  
lub `LC_ALL=en_US`.

**Wyizoluj prawdopodobn wad, jeli to moliwe**

713

Jeli to moliwe, naley wyizolowa przypadek do najmniejszej zmiany, które generuje 714  
bd. Nie zawsze jest atwo to zrobi, wic jeli nie moesz doda takiej informacji do raportu,  
nie martw si. Jednak, jeli uytownik dobrze planuje swój cykl rozwoju, przy uyciu  
maych zestawów zmian na iteracj, mona by w stanie wyizolowa problem poprzez

budow prostszej konfiguracji "bazy", która blisko pasuje do rzeczywistej konfiguracji oraz tylko uszkodzony zestaw zmian do niej dodany. Jeli masz trudności z sortowaniem, które z Twoich zmian wygenerowały błąd, może to znaczyć, że uwzględniono za błąd w każdym zestawie zmian i powinno się raczej kształcić w mniejszych krokach.

### **Wybierz odpowiedni pakiet dla którego zgłaszasz błąd**

715

Jeli nie wiesz, który komponent jest odpowiedzialny za błąd, lub jeli błąd jest ogólnym błędem dotyczącym systemów live, można wypełnić raport błędów dotyczący pseudo-pakietu `debian-live`.

716

Jednak będziemy wdzięczni, jeli postarasz się zawziąć pole możliwości, w których może pojawiać się błąd.

717

### **W czasie budowania podczas adowania początkowego (bootstrapping)**

718

`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.

719

W obu przypadkach nie jest to błąd w systemie live, ale w samym Debianie i prawdopodobnie nie możemy naprawić go bezpośrednio. Proszę zgłosić taki błąd jako dotyczący narzędzia adowania początkowego (ang. bootstrapping tool) lub uszkodzonego pakietu.

720

### **W czasie budowania podczas instalacji pakietów**

721

`live-build` instaluje dodatkowe pakiety z archiwum Debiana i w zależności od używanej dystrybucji Debiana i codziennego stanu archiwum, może się to nie powieć. Jeli błąd pojawi się w tym miejscu, należy sprawdzić, czy błąd jest powtarzalny w normalnym systemie.

722

Jeli jest to przypadek, gdzie błąd nie występuje w systemie live, ale w Debianie - zgłoś go jako dotyczący wadliwego pakietu. Uruchomienie *debootstrap* niezależnie od samej kompilacji systemu live lub uruchomienie `live bootstrap - debug` daje więcej informacji.

723

Ponadto, w przypadku korzystania z lokalnego serwera lustrzanego i/lub jakichkolwiek serwerów proxy i dowiadczania problemów prosimy zawsze najpierw spróbować odtworzyć czynności z użyciem oficjalnego serwera lustrzanego.

724

### **W czasie uruchamiania**

725

Jeli obraz nie uruchamia się, należy zgłosić go do listy wraz z informacjami wymaganymi w Zbierze potrzebne informacje. Nie zapomnij wspomnieć, jak/kiedy dokładnie obraz nie zadziała, czy by użyć za pomocą wirtualizacji lub rzeczywistego sprzętu. Jeli korzystasz z technologii wirtualizacji w jakiegokolwiek formie, proszę zawsze uruchomić go na prawdziwym sprzęcie przed zgłoszeniem błędów. Zapewnienie zrzutu ekranu awarii jest również bardzo pomocne.

726



## W czasie gdy system jest już uruchomiony

727

Jeli pakiet został zainstalowany, ale nie działa, gdy system live faktycznie działa, jest to prawdopodobnie błąd w systemie live. Jednakże:

728

## Spróbuj wykonać parę kroków

729

Przed zgłoszeniem błędów, proszę poszukać w internecie danego komunikatu o błędzie lub objawu, jaki otrzymujesz. Ponieważ jest mało prawdopodobne, że jesteś jedyną osobą, która zmaga się ze szczególnym problemem. Zawsze jest szansa, że został on już omówiony w innym miejscu i zaproponowano już możliwe rozwiązanie, obejście lub patch.

730

Należy zwrócić szczególną uwagę na list mailingowy systemów live, jak również stronę główną, ponieważ zawierają one prawdopodobnie najbardziej aktualne informacje. Jeli taka informacja istnieje, zawsze należy zawrzeć odniesienie do niej w swoim raporcie o błędzie.

731

Ponadto, należy sprawdzić aktualne wykazy błędów dla live-build, live-boot, live-config i live-tools, aby zobaczyć, czy coś podobnego nie zostało już zgłoszone.

732

## Gdzie zgłaszać błędy

733

Wszystkie błędy w systemie zarządzania błędami (BTS). Aby uzyskać informacje na temat korzystania z tego systemu, zobacz <https://bugs.debian.org/>. Możesz też przesłać błędy używając polecenia `#reportbug` z pakietu o tej samej nazwie.

734

Ogólnie rzecz biorąc, należy zgłaszać błędy podczas kompilacji: jako dotyczące pakietu live-build, błędy podczas uruchamiania: live-boot oraz błędy w czasie działania systemu live: jako dotyczące live-config. Jeli nie jesteś pewien, który pakiet będzie odpowiedni lub potrzebujesz więcej pomocy, przed złożeniem zgłoszenia błędów, zgłoś raport dotyczący pseudo-pakietu debian-live. Zajmiemy się wtedy nim i przypiszemy do odpowiedniego pakietu.

735

Należy pamiętać, że błędy znalezione w dystrybucji pochodzących od Debiana (takich jak Ubuntu, i inne) **nie** powinny być zgłaszane do BTS Debiana, chyba że błędy te mogłyby zostać utworzone w Debianie przy użyciu oficjalnych pakietów Debiana.

736

---

# Styl Kodowania

737

## Styl Kodowania

738

Rozdział ten dokumentuje styl kodowania używany w systemach live.

739

### Kompatybilność

740

Nie wolno używać składni lub semantyki, które są unikalne dla basha. Na przykład, unikaj konstrukcji ukadu.

741

Nie używaj podzbiórów POSIX'a - na przykład, używaj `$(foo)` zamiast `'foo'`.

742

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

743

Upewnij się, że cały kod powokół działa z `'set-e'`.

744

### Wcięcie

745

Zawsze używaj tabulatorów zamiast spacji.

746

### Zawijanie

747

Generalnie linie mają maksymalnie 80 znaków.

748

Używaj zakończeń linii "typowych dla Linuxa":

749

le:

750

751

```
if foo; then
    bar
fi
```

Dobrze:

752

753

```
if foo
then
    bar
fi
```

To samo dotyczy funkcji:

754

le:

755

756

```
Foo () {
    bar
}
```

Dobrze:

757

758

```
Foo ()
{
    bar
}
```

## Zmienne

Zmienne występuj zawsze zapisane drukowanymi literami.

Zmienne wykorzystane w live-build zawsze zaczynaj si przedroskiem LB\_.

Wewntrzne tymczasowe zmienne w live-build naley rozpoc od przedrostka  
\_LB\_.

Lokalne zmienne zaczynaj si przedrostkiem live-build'a

\_LB\_.

Zmienne dotyczce parametrów startowych live-config zaczynaj si od LIVE\_.

Wszystkie inne zmienne w live-config zacznij przedroskiem \_.

Uywaj nawiasów wokó zmiennych; na przykad napisz \${F00} zamiast \$F00.

Zawsze chro zmienne znakami cytatu do zachowania potencjalnych biaych znaków:  
napisz "\${F00}", a nie \$F00}.

\_\* Dla zachowania spójności, naley zawsze uywa znaków cytatu podczas przypisy-  
wania wartoci do zmiennych:

le:

```
F00=bar
```

Dobrze:

```
F00="bar"
```

Jeli zastosowane jest wiele zmiennych, przytocz cae wyraenie:

le:

```
if [ -f "${F00}"/foo/"${BAR}"/bar ]
then
    foobar
fi
```

Dobrze:

```
if [ -f "${F00}/foo/${BAR}/bar" ]
then
    foobar
fi
```

## Róne

778

Uywaj "|" (bez otaczajcych wyraenie znaków cytatu) jako rozdzielacz w zapytania do sed'a, np. "sed -e 's|foo|bar|'" (bez ""). 779

Nie uywaj komendy test dla porówna i testów, uyj "[" "]" (bez ""); np. "if [ -x /bin/foo ]; ..." a nie "if test -x /bin/foo; ...". 780

\* Uyj case gdzie to jest moliwe zamiast test, jest to atwiejsze do odczytania i szybsze w wykonaniu. 781

Uyj nazw funkcji pisanych wielkimi literami, aby ograniczy niepodane dziaanie e rodowisku uytkownika. 782

---

# Procedure

783

## Procedury

784

Rozdział ten dokumentuje procedury dla `{project}` dla różnych zadań, które wymagają współpracy z innymi zespołami w Debianie.

785

## Główne wydanie

786

Wydawanie nowej stabilnej wersji głównej Debiana zawiera wiele różnych zespołów pracujących razem, aby tak się stało. W pewnym momencie, na zespół live dochodzi do pewnego momentu i buduje obrazy systemów live. Wymagania, aby to zrobić to:

787

Serwer lustrzany zawierający wydane wersje dla Debiana i archiwum debian-bezpieczeństwa, które mogą uzyskać dostęp buildd debian-live.

788

Nazwy obrazu muszą być znane (np. `debian-live-WERSJA-ARCH-FLAVOUR.iso`).

789

Dane z `debian-cd` muszą zostać zsynchronizowane (listy wykluczające `udeb`).

790

Obrazy są budowane i przechowywane na `cdimage.debian.org`.

791

## Wydanie Docelowe

792

Ponownie potrzebujemy zaktualizowanych serwerów lustrzanych Debiana i `debian-security`.

793

Obrazy są budowane i przechowywane na `cdimage.debian.org`.

794

Wylij wiadomość z obwieszczeniem.

795

## Ostatnie Wydanie Docelowe Debiana

796

Pamiętaj, aby dostosować zarówno `chroot` i serwery lustrzane z pakietami binarnymi przy budowie ostatniego zestawu obrazów dla wydania Debiana po to, co zostało przeniesione z `ftp.debian.org` do `archive.debian.org`. W ten sposób stare prekompilowane obrazy live będą wciąż użyteczne bez modyfikacji dokonanych przez użytkownika.

797

## Szablon obwieszczenia dla wydania docelowego

798

Email z obwieszczeniem dla wydania docelowego może być wygenerowany przy użyciu poniższego szablonu i następującego polecenia:

799

800

```
$ sed \
-e 's|@MAJOR@|9.0|g' \
-e 's|@MINOR@|9.0.1|g' \
-e 's|@CODENAME@|stretch|g' \
-e 's|@ANNOUNCE@|2017/msgXXXXX.html|g'
```

Proszę dokładnie sprawdzić wiadomość przed wysłaniem i przekazać ją innym, aby dokonali korekt.

801

Updated Live @MAJOR@: @MINOR@ released

The Live Systems Project is pleased to announce the @MINOR@ update of the Live images for the stable distribution Debian @MAJOR@ (codename "@CODENAME").

The images are available for download at:

<<http://live-systems.org/cdimage/release/current/>>

and later at:

<<http://cdimage.debian.org/cdimage/release/current-live/>>

This update includes the changes of the Debian @MINOR@ release:

<<https://lists.debian.org/debian-announce/@ANNOUNCE@>>

Additionally it includes the following Live-specific changes:

- \* [WPISZ TUTAJ SPECYFICZN DLA LIVE ZMIAN]
- \* [WPISZ TUTAJ SPECYFICZN DLA LIVE ZMIAN]
- \* [POWANIEJSZE PROBLEMY MOG WYMAGA SWOJEJ OSOBNEJ SEKCJI]

About Live Systems

-----

The Live Systems Project produces the tools used to build official live systems and the official live images themselves for Debian.

About Debian

-----

The Debian Project is an association of Free Software developers who volunteer their time and effort in order to produce the completely free operating system Debian.

Contact Information

-----

For further information, please visit the Live Systems web pages at <<http://live-systems.org/>>, or contact the Live Systems team at <[debian-live@lists.debian.org](mailto:debian-live@lists.debian.org)>.



---

# Repozytorium Git

803

## Repozytorium Git

804

Lista wszystkich dostępnych repozytoriów dla `${project}` można znaleźć na stronie <http://live-systems.org>. Adres URL projektu git ma postać: `protocol://live-systems.org/git/repository`. Tak więc, w celu sklonowania `live-manual`, uruchom:

```
$ git clone git://live-systems.org/git/live-manual.git
```

806

Lub

807

```
$ git clone https://live-systems.org/git/live-manual.git
```

808

Lub

809

```
$ git clone http://live-systems.org/git/live-manual.git
```

810

Adres do sklonowania z uprawnieniami zapisu ma postać: `git@live-systems.org:/repository`.

A zatem jeszcze raz, aby sklonować `live-manual` po ssh wpisz:

812

```
$ git clone git@live-systems.org:live-manual.git
```

813

Drzewo git składa się z wielu różnych gałęzi. Gałęzi, które szczególnie wymagają powściągliwości uwagi to **debian** i **debian-next**, ponieważ zawierają one rzeczywiste prace, które ostatecznie będą znajdować się w każdej nowej wersji.

814

Po sklonowaniu każdej z istniejących repozytoriów, będziesz w gałęzi **debian**. To jest właściwe, aby móc przyjrzeć się stanowi najnowszej wersji projektu, ale przed rozpoczęciem pracy ważne jest, aby przejść do gałęzi **debian-next**. Aby to zrobić:

815

```
$ git checkout debian-next
```

816

W gałęzi **debian-next**, która nie zawsze porusza się do przodu, gdzie wszystkie zmiany są najpierw wprowadzane przed poczeniem w gałęzi **debian**. Aby dokonać analogii, to jest jak poligon dowiadczalny. Jeśli pracujesz w tej branży i potrzebujesz wykonać polecenie pull (wyciągnięcie), będzie trzeba użyć `git pull -rebase`, tak aby lokalne modyfikacje zostały zachowane przy wyciągnięciu z serwera, a następnie Twoje zmiany zostaną wprowadzone na szczycie wszystkich innych.

817

## Obsługa wielu repozytoriów

818

Jeśli masz zamiar sklonować kilka repozytoriów systemów live i chcesz przejść do gałęzi **debian-next** od razu aby sprawdzić najnowszy kod, napisać poprawkę lub przyczynić się z tłumaczeniem, powinieneś wiedzieć, że serwer git zapewnia `mrconfig`. Plik, który ułatwia obsługę wielu repozytoriów. Aby z niego korzystać musisz zainstalować pakiet `mr` a po tym, uruchomić:

819

820

```
$ mr bootstrap http://live-systems.org/other/mr/mrconfig
```

Ta komenda automatycznie sklonuje i sprawdzi do gazi **debian-next** repozytorium rozwojowego pakietów Debiana wytworzonych w ramach projektu. Należą do nich, między innymi, repozytorium live-images, który zawiera konfiguracje, używane do gotowych obrazów, które projekt publikuje do ogólnego użytku. Aby uzyskać więcej informacji na temat korzystania z tego repozytorium, zobacz Klonowanie konfiguracji opublikowanej przez Git 821

---

# Przykłady

822

---

# Przykłady

823

## Przykłady

824

W tym rozdziale omówiono 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, które pomog Ci uywa i rozumie pozostae przykłady.

825

## Uywanie przykadów

826

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

827

Naley zauway, e, ze wzgldu na zwizo, w tych przykadach nie okreiono 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 domyln 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:

828

829

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

## Samouczek 1: Domylny obraz

830

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

831

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.

832

Nie mona tego zrobi atwiej ni tak:

833

834

```
$ 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.

835

A teraz jako super-uytkownik, zbuduj obraz zapisujc przy tym log podczas budowania uywajc `tee`.

836

837

```
# lb build 2>&1 | tee build.log
```

Zakładaj, że wszystko poszło dobrze, po jakim czasie, bieżący katalog będzie zawierał `live-image-i386.hybrid.iso`. Ten hybrydowy obraz ISO można uruchomić bezpośrednio na maszynie wirtualnej, jak opisano w [Testowanie obrazu ISO z Qemu](#) i [Testowanie obrazu ISO z VirtualBox](#), lub jeszcze odpowiednio nagrany obraz na nośnikach optycznych lub urządzeniu flash USB, w sposób opisany w [Nagrywanie obrazu ISO na nośnik fizycznym](#) i [Kopiowanie hybrydowego obrazu ISO na nośnik USB](#). 838

## Samouczek 2: Narzędzie przeglądarki 839

**Przykład użycia:** Stwórz obraz z przeglądarką internetową, ucz się jak wprowadzać modyfikacje. 840

W tym samouczku, stworzymy obraz odpowiedni do wykorzystania jako narzędzie przeglądarki internetowej, służy jako wstęp do dostosowywania obrazów systemu live. 841

842

```
$ mkdir tutorial2
$ cd tutorial2
$ lb config
$ echo "task-lxde-desktop iceweasel" >> config/package-lists/my.list.chroot
```

Nasz wybór LXDE na tym przykładzie odzwierciedla nasze pragnienie, aby zapewnić minimalne środowisko pulpitu, ponieważ celem obrazu jest jednorazowy użytek, który mamy na myśli, czyli przeglądarka internetowa. Możemy pójść dalej i zapewnić domyślną konfigurację dla przeglądarki w `config/includes.chroot/etc/iceweasel/profile/`, lub dodatkowe pakiety wsparcia dla wyświetlania różnego rodzaju treści internetowych, ale pozostawiamy to jako wiczenie dla czytelnika. 843

Zbuduj ponownie obraz jako super-użytkownik, zachowując log jak to opisano w [Samouczku 1](#): 844

845

```
# lb build 2>&1 | tee build.log
```

Jeszcze raz, zweryfikuj czy obraz jest OK i przetestuj go jak to opisano w [Samouczku 1](#). 846

## Samouczek 3: Spersonalizowany obraz 847

**Przykład użycia:** Stwórz projekt spersonalizowanego obrazu zawierającego twoje ulubione oprogramowanie tak aby mógł go zabrać ze sobą gdziekolwiek pójdziesz i zapisujący sukcesywnie zmiany, kiedy tego potrzebujesz oraz zmiany w konfiguracji. 848

Ponieważ będziemy zmieniać nasz indywidualny obraz wprowadzając wiele zmian, chcemy, aby prowadziły te zmiany, próbując rzeczy eksperymentalnych i ewentualnie przywracając je, jeśli coś nie wyjdzie, będziemy trzymali naszą konfigurację w popularnym systemie kontroli wersji `git`. Będziemy również wykorzystywać najlepsze praktyki autokonfiguracji poprzez skrypty `auto` jak opisano w [Zarządzanie konfiguracją](#). 849

## Pierwsza zmiana

850

851

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

Edtuj auto/config tak, aby zawiera:

852

853

```
#!/bin/sh

lb config noauto \
  --architectures i386 \
  --linux-flavours 686-pae \
  "${@}"
```

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

854

855

```
$ lb config
```

Teraz uzupełnij swój lokaln list pakietów:

856

857

```
$ echo "task-lxde-desktop iceweasel xchat" >> config/package-lists/my.list.chroot
```

Po pierwsze, `-architectures i386` zapewnia, e w naszym systemie kompilacji `amd64`, moemy zbudowa 32-bitow wersj odpowiedni do stosowania na wikszości maszyn. Po drugie, moemy uy `-linux-flavours 686-pae` bo nie przewidujemy uywania tego obrazu na duo starszych systemach. Po trzecie, wybralimy metapakiet zadania `lxde`, który daje nam minimalny pulpit. I w kocu, dodalimy dwa wstpne ulubione pakiety: `iceweasel` i `xchat`.

858

A teraz, zbuduj obraz:

859

860

```
# lb build
```

Naley zauway, e w przeciwieństwie do dwóch pierwszych samouczków, nie musimy ju wpisywa `2>&1 | tee build.log` bo jest to obecnie zawarte w `auto/build`.

861

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

862

863

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



## Druga zmiana

W tej zmianie, będziemy sprząta nasz pierwszy zbudowany obrazu, dodawa pakiet `vlc` do naszej konfiguracji, budowa ponownie, testowa i potwierdza zmiany.

Polecenie `lb clean` oczyszczy wszystkie wygenerowane pliki z poprzedniej kompilacji z wyjątkiem pamięci podręcznej (cache), co oszczędza konieczności ponownego pobierania pakietów. To gwarantuje, że kolejne polecenie `lb build` ponownie uruchomi wszystkie etapy regeneracji pliki z naszej nowej konfiguracji.

```
# lb clean
```

Teraz dodaj pakiet `vlc` do naszej lokalnej listy pakietów w `config/package-list/my.list.chroot`:

```
$ echo vlc >> config/package-lists/my.list.chroot
```

Zbuduj ponownie:

```
# lb build
```

Przetestuj i jeżeli jesteś zadowolony wprowadź następujące zmiany:

```
$ git commit -a -m "Adding vlc media player."
```

Oczywiście, możliwe są bardziej skomplikowane zmiany w konfiguracji, prawdopodobnie dodając pliki w podkatalogach `config/`. Kiedy wprowadzasz nowe zmiany, tylko uważaj, aby nie edytować ręcznie lub zmieniać plików najwyższego poziomu w `config` zawierających zmienne `LB_*`, ponieważ są to take efekty budowania i są zawsze sprzątane przez `lb clean` i tworzone ponownie przez `lb config` przez odpowiednie automatyczne skrypty.

Dotarliśmy do końca naszej serii samouczka. Chociaż możliwe jest o wiele więcej rodzajów personalizacji, nawet tylko za pomocą kilku funkcji pokazanych w tych prostych przykładach, może być stworzona niemal nieskończona różnorodność obrazów. Pozostałe przykłady w tym rozdziale obejmują kilka innych przypadków użycia zaczerpnięte z zebranych dowiadczek użytkowników systemów live.

## Kiosk-klient serwera VNC

**Przykład użycia:** Stwórz obraz za pomocą live-build, który podczas uruchamiania, czyści automatycznie z serwerem VNC.

Stwórz katalog kompilacji i stwórz wewnątrz szkielet folderów konfiguracji, wycząc zalecane, aby utworzy minimalny system. A następnie utwórz dwie początkowe listy pakietów: pierwszą wygenerowaną ze skryptu dostarczonego przez live-build o nazwie `Pakiety` (patrz Wygenerowane listy pakietów), a drugą uwzględniając pakiety `xorg`, `gdm3`, `metacity` i `xvnc4viewer`.

```
$ mkdir vnc-kiosk-client
$ cd vnc-kiosk-client
$ lb config -a i386 -k 686-pae --apt-recommends false
$ echo '! Packages Priority standard' > config/package-lists/standard.list.chroot
$ echo "xorg gdm3 metacity xvnc4viewer" > config/package-lists/my.list.chroot
```

Jak wyjaniono w Podkrkanie APT, w celu zaoszczdzenia miejsca moe trzeba ponownie 880  
doda niektóre polecane pakiety do prawidowej pracy obrazu.

Najprostszym sposobem na wypisane listy rekomendowanych pakietów jest u ycie 881  
*apt-cache*. Na przykad:

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

W tym przykadzie okazało si, e musimy ponownie obj kilka pakietów zalecanych 883  
przez live-config i live-boot: *user-setup* do funkcji autologowania i *sudo* jako istotnego  
przy zamykaniu systemu programu. Poza tym, moe by przydatne, równie dodanie  
*live-tools*, aby móc 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, stwórz katalog */etc/skel* w *config/includes.chroot* i umie tam wasny *.xsession* 885  
dla domylnego uytownika, który 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: 887

```
# lb build
```

Korzystaj. 889

## Bazowy obraz dla nonika USB z 128MB pamici. 890

**Przykad uycia:** Stwórz domylny obraz z usunitymi niektórymi komponentami tak, 891  
aby zmieci si on na noniku USB z 128MB pamici z pozostawieniem niewielkiej przestrzeni  
do wykorzystania wedug potrzeb.

Przy optymalizacji obrazu, aby dopasowa go do okrelonego rozmiaru nonika, musisz 892  
dokona pewnych kompromisów midzy rozmiarem a funkcjonalnoci. W tym przykadzie,

przytniemy tylko tyle, aby zrobić miejsce dla dodatkowego materiału medialnego w rozmiarze 128MB, ale robienia czegokolwiek, co mogoby zniszczyć integralność zawartych pakietów, np. czyszczenie danych ustawie regionalnych poprzez pakiet *localepurge*, lub inne tego typu *inwazyjne* optymalizacje. Szczególnie godne uwagi, jest użycie `--debootstrap-options` by stworzyć minimalny system od podstaw.

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

Aby uczynić obraz pracujący prawidłowo, musimy ponownie dodać przynajmniej dwa pakiety, które zostaną pominięte przez opcję `--apt-recommends false`. Zobacz Podkręcanie APT w celu zaoszczędzenia miejsca

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

Teraz, zbuduj obraz w typowy sposób:

```
# lb build 2>&1 | tee build.log
```

On the author's system at the time of writing this, the above configuration produced a 110MB image. This compares favourably with the 192MB image produced by the default configuration in Tutorial 1.

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 91MB. 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.

## Pulpit GNOME w lokalnym języku oraz instalator

**Przykład użycia:** Stwórz obraz z pulpitem GNOME i lokalizacją dla Szwajcarii wraz z instalatorem.

Chcemy stworzyć obraz ISO-hybrydy dla architektury i386 z naszym preferowanym pulpitem, w tym przypadku GNOME, zawierając wszystkie pakiety, które byłyby zainstalowane przez standardowy instalator Debiana dla GNOME.

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

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

Teraz możemy rozpocząć wyszukiwanie odpowiedniego zadania. Najpierw:

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

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

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

W czasie startu systemu będziemy generować lokalizację `de_CH.UTF-8` i wybierzemy układ klawiatury `ch`. Teraz poskładamy kawałki razem. Przypominamy sobie korzystanie z metapakietów i metapakiety są poprzedzone przedrostkiem `task-`, po prostu określimy te parametry rozruchowe dotyczące języka, a następnie dodamy standardowe pakiety priorytetowe i wszystkie wykryte metapakiety zadań do naszej listy pakietów w następujący sposób:

```
$ mkdir live-gnome-ch
$ cd live-gnome-ch
$ lb config \
  -a i386 \
  --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. The 586 kernel flavour, which is currently necessary for the launcher to work properly, will be included by default.

---

# Dodatek

913

---

# Przewodnik redakcyjny

914

**Przewodnik redakcyjny** 915**Wytyczne dla autorów** 916

Ten rozdział zajmuje się pewnymi ogólnymi rozważaniami, które należy wziąć pod uwagę podczas pisania dokumentacji technicznej dla live-instrukcji. Są one podzielone na funkcje językowe oraz zalecane procedury. 917

**Uwaga:** Autorzy powinni najpierw przeczytać Współtworzenie tego dokumentu 918

**Funkcje językowe** 919*Użyj zwykłego angielskiego* 920

Należy pamiętać, że wysoki procent czytelników nie są rodzimymi użytkownikami języka angielskiego. Wic jako generalną zasadę spróbuj używać krótkich, sensownych zdań, zakończonych kropką. 921

To nie znaczy, że trzeba korzystać z uproszczonego, naiwnego stylu. Proponuje się unikać, na ile to możliwe, złożonych zdań podrzędnych, które czyni tekst trudny do zrozumienia dla nie rodzimych użytkowników języka angielskiego. 922

*Różnorodność języka angielskiego* 923

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. 924

Spodziewamy się, że różne odmiany języka mogą mieszać się bez tworzenia nieporozumień, ale ogólnie należy starać się być spójnym i przed podjęciem decyzji o użyciu brytyjskiego, amerykańskiego lub innego dialektu języka angielskiego według uznania, proszę przyjrzeć się, jak inni ludzie piszą i postarać się ich naśladować. 925

*Bądź zbalansowany* 926

Nie bądź stronniczy. Unikaj w tym odniesienia do ideologii zupełnie niepowiązanych z live-manual. Pismo techniczne powinno być jak najbardziej neutralne. Jak to jest w naturze piśmiennictwa naukowego. 927

*Bądź poprawny politycznie* 928

Staraj się unikać seksistowskiego języka, jak to jest tylko możliwe. Jeśli potrzebujesz, odwołania do trzeciej osoby liczby pojedynczej najlepiej użyć jakiejś formy bezosobowej lub "wy" zamiast "on", "ona" lub niewygodnych wynalazków, takie jak "mógłby/mogaby", "byem/am" i podobnych. 929

*Bądź zwięzły* 930

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. 931

*Oszczędź pracę tłumaczycy* 932

Naley pamita, e cokolwiek napiszesz bdzie musiao zosta przetumaczone na kilka 933  
innych jzyków. Oznacza to, e sporo osób, bdzie musia wykona dodatkow prac jeli  
dodasz bezuzyteczne lub nadmiarowe informacje.

*Bd zgodny* 934

As suggested before, it is almost impossible to standardize a collaborative document 935  
into a perfectly unified whole. However, every effort on your side to write in a  
coherent way with the rest of the authors will be appreciated.

*Be spójny* 936

Use as many text-forming devices as necessary to make your text cohesive and un- 937  
ambiguous. (Text-forming devices are linguistic markers such as connectors).

*Bd opisowy* 938

It is preferable to describe the point in one or several paragraphs than merely using 939  
a number of sentences in a typical "changelog" style. Describe it! Your readers will  
appreciate it.

*Sownik* 940

Look up the meaning of words in a dictionary or encyclopedia if you do not know 941  
how to express certain concepts in English. But keep in mind that a dictionary can  
either be your best friend or can turn into your worst enemy if you do not know how  
to use it correctly.

English has the largest vocabulary that exists (with over one million words). Many 942  
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.

Zgodnie z ogóln zasad, jeli koncepcja moe by wyraony za pomoc rónych synonimów 943  
to dobr rad bdzie wybieranie pierwszego sowa zaproponowanego przez sownik. W  
razie wtpliwoi, czsto susznym jest wybieranie sowa pochodzenia germaskiego (zwykle  
jednosylabowe sowa). Ostrzegamy, e te dwie techniki, mog produkowa raczej mow  
nieformaln, ale przynajmniej wybór sów bdzie o szerokim zastosowaniu i ogólnie  
przyjty.

Korzystanie ze sownika kolokacji jest zalecane. S one bardzo pomocne, kiedy przy- 944  
chodzi do znajomoci sów najczciej wystpujących razem.

Ponownie; dobr praktyk jest, aby uczy si z pracy innych. Korzystanie z wyszuki- 945  
warki, aby sprawdzi, jak inni autorzy mog korzysta z niektórych wyrae moe bardzo  
pomóc.

*Faszywi przyjaciele, idiomy i inne wyraenia idiomatyczne* 946

Uwaaj na faszywych przyjació. Bez wzgldu na to, jak biegy jeste w obcym jzyku nie 947  
mona pomóc wpadajc od czasu do czasu w puapk tzw. "faszywych przyjació", sowa,  
które wygldej podobnie w dwóch jzykach, ale których znaczenie lub zastosowanie  
moe by zupenie inne.

Staraj si unika idiomów w jak najwikszym stopniu. "Idiomy" to wyraenia, które 948



mog mie znaczenie zupenie odmienne od tego, co ich poszczególne sowa wydaj si oznacza. Czasami, idiomy, mog by trudne do zrozumienia nawet dla rodzimych uytkowników jzyka angielskiego!

*Unikaj slangu, skrótów, mowy potocznej...* 949

Nawet jeli popierasz korzystanie ze zwykego, codziennego jzyka angielskiego, pisanie techniczne naley do formalnej formy jzyka. 950

Staraj si unika slangu, niepopularnych skrótów, które s trudne do zrozumienia, a przede wszystkim skrótów, które próbuj naladowa jzyk mówiony. Nie wspominajc o typowych dla kanaów IRC i przyjaznych dla zamknitych gron wyrae. 951

## **Procedury** 952

*Przetestuj przed zapisaniem* 953

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. 954

*Przykady* 955

W przypadku dostarczania przykadu spróbuj by tak dokadny jak tylko moesz. Przykad jest, mimo wszystko, tylko przykadem. 956

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. 957

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. 958

*Linki zewntrzne* 959

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. 960

Poza tym, ludzie, którzy czytają instrukcj w trybie offline nie bd mogli ledzi tych linków. 961

*Unikaj nadawania marki i rzeczy, które naruszają licencj zgodnie z którą podręcznik ten zosta opublikowany* 962

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. 963

live-manual jest oparty na licencji GNU GPL. Ma to wiele skutków, które odnosz si 964

do redystrybucji materiału (dowolnego rodzaju, w tym grafiki chronionej prawami autorskimi lub logo), który jest opublikowany wraz nim.

*Napisz pierwszy szkic, przeglądaj go, edytuj, popraw i cofnij zmiany jeżeli wymaga tego sytuacja* 965

- Burza mózgów!. Najpierw musisz zorganizować swoje pomysły w logicznej kolejności zdarzeń. 966

- Kiedy już w jakiś sposób masz zorganizowane te koncepcje w głowie napisz pierwszy szkic. 967

- Dokonaj przeglądu gramatyki, składni i pisowni. Należy pamiętać, że właściwe nazwy wydań, takich jak `{testing}` lub `sid` nie powinny być kapitalizowane, gdy odnosi się do nich jako nazw kodowych. Aby sprawdzić pisownię można uruchomić polecenie `spell`. tj. polecenie `make spell` 968

- Udoskonalaj swoje wyrażenia, a jeśli to konieczne cofnij i przerób każde. 969

*Rozdziały* 970

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. 971

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. 972

*Znaczniki* 973

And last but not least, live-manual uses 974

「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 są przykłady znaczników, które mogą okazać się użyteczne: 976

- Dla pogrubienia użyj: 977

```
*{foo}* lub !{foo}!
```

powoduje: **foo** lub **foo**. Użyj tego by wyszczególnić określone słowa kluczowe. 979

- Dla kursywy użyj: 980

```
/ {foo} /
```

powoduje: *foo*. Użyj tego dla np. nazw paczek Debiana. 982

- Dla czcionki o stałej szerokości użyj: 983

984

```
#{foo}#
```

powoduje: foo. Uyj tego np. dla nazw polece. A take aby uwidoczni poszczególne sowa kluczowe jak cieki dostpowe. 985

- Dla bloków z kodem uyj: 986

```
code{  
  $ foo  
  # bar  
}code
```

powoduje: 988

```
$ foo  
# bar
```

Uyj code{ do otwarcia i #{ }code# do zamknicia tagu. Wane jest, aby pamita, by zostawi miejsce na pocztku kadej linii kodu. 990

## Wytyczne dla tumaczy 991

Ta sekcja zajmuje si pewnymi ogólnymi rozwaaniami, które naley wzi pod uwag przy tumaczeniu zawartoci live-manual. 992

Jako ogólne zalecenie, tumacze powinni przeczyta i zrozumie zasady tumaczenia, które maj zastosowanie do ich specyficznych jzyków. Zazwyczaj, grupy i listy dyskusyjne tumacze dostarczaj informacji o tym, jak tworzy przetumaczone prac zgodne z normami jakoci Debiana. 993

**Uwaga:** Tumacze powinni równie przeczyta Wspótworzenie tego dokumentu. W szczególności rozdzia Tumaczenie 994

## Wskazówki tumaczenia 995

### *Komentarze* 996

The role of the translator is to convey as faithfully as possible the meaning of words, sentences, paragraphs and texts as written by the original authors into their target language. 997

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. 998

### *UT, Uwagi Tumacza* 999

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". 1000

*Wyraenia w trybie bezosobowym* 1001

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. 1002

*Faszywi przyjaciele* 1003

Puapka "faszywych przyjaciół" wyjanionych wczeniej szczególnie dotyczy tumaczy. Dwukrotnie sprawdzi znaczenie podejrzanych faszywych przyjaciół w razie wtpliwoi. 1004

*Znaczni* 1005

Tumacze pracujcy pocztkowo nad plikami **pot**, a póniej z plikami `*{po}` \* znajd wiele znaczników w cigach. Mog oni przetumaczy tekst tak, jak to jest tylko moliwe do tumaczenia, ale niezwykle wanym jest, aby uywali oni dokadnie takich samych znaczników jak w oryginalnej wersji angielskiej. 1006

*Bloki kodowe* 1007

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 sposób, w duszej perspektywie czasu na okrelenie, co ju zostao przetumaczone, a co nie podczas sprawdzania integralnoci plików .po. 1008

*Nowe linijki* 1009

Przetumaczone teksty musz mie te same znaki nowej linii jak teksty oryginalne. Naley uwaa, aby nacisn klawisz "Enter" lub wpisa **n** jeli pojawi si one w oryginalnych plików. Znaki te czsto pojawiaj si, na przykad, w blokach z kodem. 1010

Nie popeniaj bedów, to nie znaczy, e przetumaczony tekst musi mie tak sam dugo jak w wersji angielskiej. Jest to prawie niemoliwe. 1011

*Nieprzetumaczalne cigi znaków* 1012

Tumacze nigdy nie powinni tumaczy: 1013

- Nazw kodowych wyda (które powinny by pisane maymi literami) 1014

- Nazw programów 1015

- Komend podanych jako przykad 1016

- Metadanych (zazwyczaj pomidzy dwukropkami **:metadata:**) 1017

- Linków 1018

- cieek dostpnowych 1019