Podrcznik Systemów Live

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

2015-08-23

Podrcznik Systemów Live

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

O tym podrczniku

O tym podrczniku

O tym podrczniku

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 podrcznik live-manual skupia si przede wszystkim na pomocy w budowaniu systemu live, a nie na tematach uytkownika kocowego. To uytkownik kocowy równie moe znale przydatne informacje w nastpujcych sekcjach: Podstawy obejmuje pobieranie skompilowanych obrazów i przygotowanie obrazów tak aby byy uruchamiany z nonika przenonego lub z sieci, równoczenie uywanie web builder lub uruchamianie live-build bezporednio w systemie. Dostosowywanie zachowania w czasie dziaania systemu opisuje kilka opcji, które mog zosta okrelone podczas startu, na przykad wybieranie ukadu klawiatury i ustawienia regionalne, uywajc opcji persistence.

Niektóre z polece zawartych w tekcie musz by wykonywane z uprawnieniami superuytkownika, które mog by uzyskane przez stanie si uytkownikiem root poprzez su lub uywajc sudo. Aby odróni te polecenia, które mog by wykonywane przez nieuprzywilejowanego uytkownika i te wymagajce praw administratora, polecenia zostay poprzedzone przez odpowiednio by $ or # . Symbol ten nie jest czci polecenia.

Dla niecierpliwych

Wierzymy, e wszystko w tym podrczniku jest wane, przynajmniej dla niektórych z naszych uytkowników. Zdajemy sobie spraw równie, e zawiera on duo materiau do sprostania. I e moe chciaby dowiadczy szybkich postpów w uywaniu oprogramowania przed zagbianiem si w szczegóy. Dlatego sugerujemy czytanie w nastpujcej kolejnoci.

Najpierw przeczytaj rozdzia, O tym podrczniku, od pocztku, koczc na sekcji Warunki. Nastpnie przejd do trzech wicze na pocztku sekcji Przykady majcej na celu nauczy Ci podstaw budowania obrazu i dostosowywania. Najpierw przeczytaj Uywanie przykadów, nastpnie Tutorial 1: Domylny obraz, Tutorial 2: Narzdzie przegldarka i wreszcie Tutorial 3: Spersonalizowany obraz. Pod koniec tych tutoriali, bdziesz mie ogólny zarys tego, co mona zrobi z systemami live.

Zachcamy do powrotu i bardziej dogbnej analizy podrcznika, by moe nastpnym razem czytanie Podstaw, przejrzenie lub pominicie Budowanie obrazu netboot, a skoczywszy czytajc Omówienie dostosowywania i rozdziaów, które po nim nastpuj. W tym momencie, mamy nadziej, e s zostae zainteresowany tym, co mona zrobi z systemami live i zmotywowany, aby przeczyta reszt tego podrcznika, od deski do deski.

Definicje

System live: System operacyjny, który mona uruchomi bez instalacji na dysku twardym. Systemy live nie zmieniaj lokalnego systemu(-ów) lub pliku(-ów) ju

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.

${project}: 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 uruchomiania 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**, obecnia nazwa kodowa to ${stable}, zawiera najnowsze oficjalnie wydanie dystrybucji Debian. Dystrybucja **testing**, tymczasowo nadana nazwa kodowa to ${testing}, to obszar postoju 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 podrczniku, 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.

Autorzy

Lista Autorów (w kolejnoci alfabetycznej):

- Ben Armstrong
- Brendan Sleight
- Carlos Zuferri
- Chris Lamb
- Daniel Baumann
- Franklin Piat
- Jonas Stein
- Kai Hendry
- Marco Amadori
- Mathieu Geli
- Matthias Kirschner
- Richard Nelson
- Trent W. Buck

Wnoszenie wkadu do tego dokumentu

Intencj niniejszy podrcznik jest dzianie jako projekt spoeczny, a wic wszelkie propozycje usprawnie i zmian s bardzo mile widziane. Prosz przejrzyj 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 podrczniku 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
```

Poniewa zbudowanie podrcznika we wszystkich obsugiwanych jzykach zajmuje troch czasu, autorzy mog zdecydowa, e wygodniej bdzie skorzysta z jednego z szybko korygujcych skrótów podczas zamieszczania nowej dokumentacji, któr dodali do podrcznika angielskiego. Uywanie `PROOF=1` tworzy live-manual w formacie HTML, ale bez posegmentowanych plików HTML, a przy uyciu `PROOF=2` tworzy si live-manual w formacie PDF, ale tylko sformatowane jako A4 i listowy pionowy. Dlatego korzystajc z jednej z moliwoci opcji `PROOF=` mona zaoszczdzi sporo czasu, np.:

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

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

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

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

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

Lub kombinacja obu, np:

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

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

Tumaczenie

W celu przetumaczenia live-manual, wykonaj nastpujce kroki, w zalenoci od tego, czy rozpoczynasz tumaczenie od zera czy te kontynuujesz prac na ju istniejcym tumaczeniu:

Rozpocznij nowe tumaczenie od zera

Przetumacz pliki **about_manual.ssi.pot**, **about_project.ssi.pot** i **index.html.in.pot** w `manual/pot/` na swój jzyk uywajc swojego ulubionego edytora (np. *poedit*) i wylij przetumaczony plik `.po` do listy mailingowej, aby sprawdzi ich integralno. Sprawdzanie integralnoci live-manual sprawdza, nie tylko czy pliki `.po` s w 100% przetumaczone, ale równie wykrywa ewentualne bdy.

Po sprawdzeniu, aby wczy nowy jzyk w autobuild to wystarczy doda pocztkowo przetumaczone pliki do `manual/po/${LANGUAGE}/` i uruchomi `make commit`. A nastpnie, edytowa `manual/_sisu/home/index.html` dodajc nazw jzyka i jego nazwy w jzyku angielskim w nawiasach.

Kontynuuj prac z ju rozpocztym tumaczeniem

Jeli Twój jzyk docelowy zosta ju dodany, mona losowo kontynuowa tumaczenia pozostaych plików .po w `manual/po/${LANGUAGE}/` za pomoc dowolnego edytora (np. *poedit*).

Nie zapomnij, e musisz uruchomi `make commit` w celu zapewnienia, e przetumaczone podrczniki s aktualizowane z plików .po i e moesz przeglda swoje zmiany uruchomamiajc `make build` przed `git add .`, nastpnie `git commit -m "Translating..."` (Tumaczenie...) i `git push`. Pamitaj, e polecenie `make build` moe zaj duo czasu, moesz wybra indywidualnie korygowane jzyki jak wyjaniono w Zatwierdzanie zmian

Po uruchomieniu `make commit` zobaczysz jaki przewijajcy si tekst. S to przede wszystkim informacyjne komunikaty o statusie przetwarzania, a take niektóre wskazówki na temat tego, co mogoby by zrobione, aby poprawi live-manual. Chyba, e widzisz bd krytyczny, zazwyczaj w takim wypadku moesz kontynuowa i wysa swój wkad w tumaczenie.

live-manual skada si z dwóch narzdzi, które mog w znacznym stopniu pomóc tumaczom znale nieprzetumaczone i zmienione cigi znaków. Pierwszy z nich jest "make translate". Uruchamia skrypt, który mówi dokadnie ile nie przetumaczonych cigów jest w kadym pliku .po. Drugi, "make fixfuzzy", dziaa tylko na zmienionych cigach znaków, ale to pomaga znale je i edytowa jeden po drugim.

Naley pamita, e nawet jeli te narzdzia mog by bardzo pomocne do pracy z tumaczeniami w linii polece, to korzystanie z wyspecjalizowanego narzdzia jak *poedit* jest zalecanym sposobem, aby wykona zadanie. Jest to równie dobry pomys, aby zapozna si z dokumentacjami Debian o lokalizacjach (l10n) i tymi specyficznymi dla live-manual: Poradnik dla tumaczy.

Uwaga: Moesz uy `make clean` aby oczyci swoje drzewo git przed zamieszczeniem. Ten krok nie jest obowizkowy, dziki plikowi .gitignore, ale dobr praktyk jest unikna zatwierdzania plików odruchowo.

O ${project}

72

O ${project}

Motywacja

Co jest nie tak w moim dotychczasowym systemie live?

Gdy ${project} zosta zainicjowany, byo ju dostpnych kilka systemów live opartych na Debianie i które teraz wietnie si spisuj. Z punktu widzenia Debian wikszo z nich ma jednak jedn lub wicej z nastpujcych wad:

- Nie s to projekty Debian i dlatego nie posiadaj take wsparcia ze strony Debian.
- Mieszaj rózne dystrybucje, np.: **testing** i **unstable**.
- Wspieraj tylko i386.
- Modyfikuj zachowanie i/lub wygld pakietów przez obcinanie ich w celu zaoszczdzenia miejsca.
- Zawieraj pakiety z poza archiwum Debian.

_ * S dostarczane z jdrem systemu zawierajcym dodatkowe aty, które nie s czci Debian.

- S due i powolne a ze wzgldu na swoj zwyk wielko, a zatem nie nadaj si do zagadnie odzyskiwania.
- Nie s one dostpne na rónych nonikach, np. CD, DVD, USB-stick czy obrazy netboot.

Czemu tworzy nasz wasny system live?

Debian to Uniwersalny system operacyjny: Debian bdzie posiada system live do przetestowania i dokadnego zaprezentowania systemu Debian z nastpujcymi gównymi zaletami:

- Bdzie pod-projektem Debian.
- Bdzie odzwierciedla stan aktualnej dystrybucji.
- Bdzie wspiera tak wiele architektur jak to tylko moliwe.
- Bdzie si skada tylko z niezmienionych pakietów Debian.
- Nie bdzie zawiera adnych pakietów, które nie s w archiwum Debian.
- Bdzie korzysta z niezmienionego jdro Debian bez dodatkowych poprawek.

Filozofia

Tylko niezmienione pakiety z dziau Debian ”main”

Bdziemy uywa tylko pakietów z repozytorium Debian w sekcji ”main”. Sekcja ”non-free” nie jest czci Debian, a zatem nie moe by uywana do budowania oficjalnych obrazów systemu live.

Nie bdziemy zmienia adnych pakietów. Zawsze, gdy bdziemy musieli co zmieni, zrobimy to w porozumieniu z opiekunem tego pakietu w Debianie.

W drodze wyjtku, nasze wasne pakiety, takie jak live-boot, live-build lub live-config mog by zastosowane tymczasowo z wasnego repozytorium z przyczyn rozwojowych (np. do tworzenia zrzutów rozwojowych). Zostan one przesane do Debian na bieco.

Bez konfiguracji pakietów systemu live

Na tym etapie nie bdziemy dostarcza lub te instalowa przykadowych lub alternatywnych konfiguracji pakietów. Wszystkie pakiety bd uyte w u ich podstawowych konfiguracjach takich jakie s po normalnej instalacji Debian.

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

System do konfigurowania pakietów jest dostarczony przez uycie debconf'a; umoliwiajc instalacj niestandardowo skonfigurowanych pakietów w Twoim niestandardowo stworzonym obrazie systemu live. A dla prekompilowanych obrazów live zdecydowalimy, aby pozostawi pakiety w swojej domylnej konfiguracji, chyba e bdzie to absolutnie niezbdne do pracy w rodowisku live. Wszdzie tam, gdzie to moliwe, wolimy dostosowywa pakiety w archiwum Debian, aby lepiej pracoway w systemie live w porównaniu do dokonywania zmian w live toolchain lub konfiguracji obrazu prekompilownego klonuj-konfiguracja-przez-git. Aby uzyska wicej informacji, zobacz Omówienie dostosowywania.

Kontakt

Lista Mailingowa: Podstawowym kontaktem do projektu jest lista mailingowa na 「https://lists.debian.org/debian-live/」. Moesz napisa do listy bezporednio przez zadresowanie poczty do debian-live@lists.debian.org. Archiwa listy dostpne s na 「https://lists.debian.org/debian-l
.

IRC: Wielu uytkowników i deweloperów jest obecnych na kanale #debian-live na irc.debian.org (OFTC). Kiedy zadajesz pytanie na IRC, prosimy o cierpliwie czeka na odpowied. W przypadku, gdy odpowiedz nie pojawi si, napisz na list mailingow.

BTS:
「Debian Bug Tracking System」 https://www.debian.org/Bugs/ (BTS) zawiera informacje o bdach zgoszonych przez uytkowników i deweloperów. Kady bd ma przypisany swój numer i jest przechowywany, dopóki nie zostaje on oznaczony jako rozwizany. Aby uzyska wicej informacji, zobacz {Zgaszanie bdów#bugs.

Uytkownik

106

Instalacja

107

Instalacja

Wymagania

Budowanie obrazów systemu live ma bardzo niewielkie wymagania systemowe:

- Dostp do konta (root) super-uytkownika
- Aktualna wersja live-build
- Powoka zgodna z POSIX, taka jak *bash* lub *dash*
- *debootstrap*
- Linux 2.6 lub nowszy.

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

Instalowanie live-build

Moesz zainstalowa live-build na wiele rónych sposobów:

- Z repozytorium Debiana
- Ze róda
- Ze zrzutów deweloperskich

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

Z repozytorium Debiana

Zwyczajnie zainstaluj live-build jak kad inn paczk:

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

Ze róda

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

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

Moesz zbudowa i zainstalowa wasn paczk Debiana wykonujc:

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

Teraz zainstaluj którkolwiek wieo 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 niestandardowych 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 podrcznikiem man pakietó live-boot i live-config aby uzyska szczegóowe informacje na temat dostosowywania, jeeli to jest Twój powód do budowania tych pakietów ze róde.

Zbuduj pliki .deb live-boot i live-config

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

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*):

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

Uyj majcych wygenerowanych plików .deb

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, poinstruuj si Instalowanie zmodyfikowanych paczek innych firm, aby tymczasowo umieci odpowiednie pliki w konfiguracji. W szczególnoci 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 powyej, te polecenia bash skopiuj wszystkie odpowiednie pakiety, w tym domylne dla nich back-endy:

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

Ze zrzutów deweloperskich

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.

Podstawy

157

Podstawy

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

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

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

Co to jest system live?

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

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

Obraz jdra Linuxa, zazwyczaj nazwany `vmlinuz*`

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

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

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

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

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

Pobieranie prekompilowanych obrazów

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/」.

Using the web live image builder

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.

Uywanie i przestrogi dotyczce Web buildera

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.

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.

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.

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 noniku

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 pozostaej 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

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

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

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

Uywanie wirtualnej maszyny do testowania

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

- Running a VM requires enough RAM for both the guest OS and the host and a CPU with hardware support for virtualization is recommended.
- There are some inherent limitations to running on a VM, e.g. poor video performance, limited choice of emulated hardware.
- When developing for specific hardware, there is no substitute for running on the hardware itself.
- Occasionally there are bugs that relate only to running in a VM. When in doubt, test your image directly on the hardware.

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

Testowanie obrazu ISO z uyciem QEMU

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.

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`.

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

Uruchamianie obrazu ISO jest proste:

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

Zobacz podrczniki man, aby uzyska wicej szczegóów.

Testowanie obrazu ISO z uyciem VirtualBox'a

W celu przetestowania ISO w *VirtualBox*'ie:

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

Create a new virtual machine, change the storage settings to use `live-image-i386.hybrid.iso` as the CD/DVD device, and start the machine.

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

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

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

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

Budowanie i uywanie obrazu HDD

Building an HDD image is similar to an ISO hybrid one in all respects except you specify `-b hdd` and the resulting filename is `live-image-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.

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

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

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

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

A teraz zbuduj obraz uywajc polecenia `lb build`:

```
# lb build
```

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

The generated binary image contains a VFAT partition and the syslinux bootloader, ready to be directly written on a USB device. Once again, using an HDD image is just like using an ISO hybrid one on USB. Follow the instructions in Using an ISO hybrid live image, except use the filename `live-image-i386.img` instead of `live-image-i386.hybrid.iso`.

Likewise, to test an HDD image with Qemu, install *qemu* as described above in Testing an ISO image with QEMU. Then run `kvm` or `qemu`, depending on which version your host system needs, specifying `live-image-i386.img` as the first hard drive.

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

Budowanie obrazu netboot

The following sequence of commands will create a basic netboot image containing a default live system without X.org. It is suitable for booting over the network.

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

```
# lb clean
```

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

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

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

In contrast with the ISO and HDD images, netbooting does not, itself, serve the filesystem image to the client, so the files must be served via NFS. Different network filesystems can be chosen through lb config. The `–net-root-path` and `–net-root-server`

options specify the location and server, respectively, of the NFS server where the filesystem image will be located at boot time. Make sure these are set to suitable values for your network and server.

A teraz zbuduj obraz uywajc polecenia `lb build`:

```
# lb build
```

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

For example, if you unpack the generated `live-image-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 zainstalowa 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 uatwi sobie ycie moemy uy 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`.

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

Webbooting

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

Getting the webboot files

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`.

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

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

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

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.

Uruchamianie obrazów webboot

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

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

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

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

Przegld narzdzi

296

Przegld narzdzi

Ten rozdzia zawiera przegld trzech gównych narzdzi stosowanych w budowie systemów live: live-build, live-boot i live-config.

Pakiet live-build

live-build to zbiór skryptów do budowania systemów live. Skrypty te s równie okrelane jako ”polecenia”.

Pomysem stojcym za live-build jest bycie opraw, która uywa struktury katalogów jako konfiguracji, aby cakowicie zautomatyzowa i dostosowa wszystkie aspekty budowania obrazu live.

Wiele poj jest podobnych do tych uywanych do budowania pakietów Debiana z uyciem *debhelper*'a:

Skrypty posiada centraln lokalizacj dla konfiguracji ich dziaania. Dla *debhelper*'a jest to podkatalog drzewa pakietów `debian/`. Na przykad, dh_install bdzie szuka, sporód innych, pliku o nazwie `debian/install` do okrelenia, które pliki powinny zawiera si w okrelonym pakiecie binarnym. W taki sam sposób, live-build przechowuje swoj konfiguracj w caoci w podkatalogu `config/`.

Skrypty s niezalene - to znaczy, e zawsze jest bezpieczne uruchomienie poszczególnych polece.

W przeciwiestwie do *debhelpera*, live-build zapewnia narzdzia do generowania szkieletu katalogów konfiguracyjnych. Moe to by uznane za podobne do narzdzi takich jak *dh-make*. Aby uzyska wicej informacji na temat tych narzdzi, kontynuuj czytanie, poniewa pozostaa cz tego rozdziau omawia cztery najwaniejsze polecenia. Naley zauway, e gównym wrapperem dla polecenia live-build jest `lb`.

lb config: Odpowiedzialny za inicjowanie katalogu konfiguracji systemu live. Zobacz Polecenie lb config, aby uzyska wicej informacji.

lb build: Odpowiedzialny za rozpoczcie kompilacji systemu live. Zobacz polecenie lb build aby uzyska wicej informacji.

lb clean: Odpowiedzialny za czyszczenie kompilacji systemu live. Zobacz polecenie lb clean aby uzyska wicej informacji.

Polecenie `lb config`

Jak omówiono w live-build, skrypty, które skadaj si na live-build czytaj swoj konfiguracj przy uyciu polecenia `source` z katalogu o nazwie `config/`. Budowanie tego katalogu rcznie byoby czasochonne i podatne na bdy, polecenie `lb config` moe by uywane do tworzenia pocztkowej konfiguracji drzewa katalogów.

Wykonanie `lb config` bez adnych argumentów tworzy podkatalog `config`, w którym zapisane s niektóre domylne ustawienia, w plikach konfiguracyjnych, oraz dwa szkielety drzew o nazwach `auto/` i `local/`.

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

Wcicia

Normalnie, pewnie bdziesz chcia okreli niektóre opcje. Na przykad, aby okreli, jakiego menadera pakierów chcesz uy podczas budowania obrazu:

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

Jest moliwe ustalenie wielu opcji, takich jak:

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

Pena lista opcji dostpna jest w podrczniku man pakietu `lb_config`.

Polecenie `lb build`

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

Polecenie `lb clean`

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

Pakiet live-boot

live-boot to zbiór skryptów zapewniajcych haki do *initramfs-tools*, wykorzystywane do wytwarzania plików initramfs, które s w stanie uruchomi system live, takich jak te stworzone przez live-build. Obejmuje to obrazy ISO systemów live, archiwa netboot i obrazów dysku USB.

W czasie rozruchu bdzie szuka noników tylko do odczytu zawierajcych katalog `/live/`, gdzie jest przechowywany system plików (czsto skompresowany obraz systemu plików jak SquashFS). Jeli znajdzie takowy, stworzy zapisywalne rodowisko, stosujc aufs, dla systemów takich jak Debian, aby z niego wystartowa.

Wicej informacji na temat pocztkowych plików ramfs w Debianie mona znale w Podrczniku Debiana Linux Kernel na 「http://kernel-handbook.alioth.debian.org/」 w rozdziale initramfs.

Pakiet live-config

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.

Zarzdzanie konfiguracj

329

Zarzdzanie konfiguracj

Ten rozdzia wyjania, jak zarzdza konfiguracj live od pocztku jej tworzenia, przez kolejne zmiany i kolejne wersje oprogramowania live-build i obrazu live.

Radzenie sobie ze zmianami konfiguracji

Konfiguracje live rzadko s idealne na przy pierwszej próbie. Powinno si doda opcje `lb config` z linii polece do wykonania pojedynczej kompilacji, ale bardziej typowe jest sprawdza te opcje i kompilowa ponownie a do uzyskania satysfakcji. Aby poradzi sobie z tymi zmianami, potrzeba automatycznych skryptów, które zapewni, e konfiguracja przechowywana jest w stanie spójnym.

Czemu uywa automatycznych skryptów? Co one robi?

Polecenie `lb config` zapisuje opcje, które s wprowadzane do plików w #{config}/*# oraz wielu innym opcj przypisuje wartoci domylne. Jeli uruchomisz `lb config` ponownie, nie skasuje to adnych opcji, która zostaa zapisana na podstawie pocztkowych opcji. Tak wic, na przykad, jeli by uruchomi `lb config` ponownie z now wartoci dla `–binary-images`, wszelkie opcje zalene, które zostay przypisane jako domylne dla poprzedniej dystrybucji mog ju nie wspógra z nowym ustawieniem. Pliki te nie s przeznaczone do odczytu lub edycji. Przechowuj one wartoci dla ponad stu opcji, wic nikt, nie mówic ju o robieniu tego w pojedynk, nie bdzie móg zobaczy, które z tych opcji s faktycznie przypisane. I wreszcie, po uruchomieniu `lb config`, a nastpnie uaktualnieniu live-build a zdarza si, e zmieniaj si nazwy opcji, `config/*` nadal bdzie zawiera zmienne nazwane po staremu, które nie s ju aktualne.

Z tych wszystkich powodów, skrypty `auto/*` czyni Twoje ycie atwiejszym. S proste wrappery do polece `lb config`, `lb build` i `lb clean`, które s zaprojektowane, aby pomóc w zarzdzaniu konfiguracj. Skrypt `auto/config` przechowuje toje polecenie `lb config` ze wszystkimi podanymi opcjami, skrypt `auto/clean` usuwa pliki zawierajce wartoci zmiennych konfiguracyjnych a skrypt `auto/build` zachowuje log `build.log` kadej kompilacji. Kady z tych scenariuszy jest uruchamiany automatycznie przy kadym uruchomieniu odpowiedniego polecenia `lb`. Korzystajc z tych skryptów, konfiguracja jest bardziej czytelna i jest przechowywana w sposób wewntrznie spójny z jednej wersji do nastpnej. Ponadto, bdzie o wiele atwiej zidentyfikowa opcje, które naley zmieni po uaktualnieniu live-build i po przeczytaniu dokumentacji aktualizacji.

Uyj przykadowych automatycznych skryptów

Dla Twojej wygody, live-build jest dostarczany z przykadowymi skryptami powoki do automatycznego kopiowania i edycji. Rozpocznij now, domyln konfiguracj, a nastpnie skopiuj do niej przykady:

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

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

```
#!/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.

Uwaga: Specjalnny parametr noauto jest uyty tutaj, aby powstrzyma kolejne zapytania 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 czytelnoci, jak pokazano w powyszym przykadzie, nie zapomnisz odwrotnego ukonika (
) na kocu kadej linii, która kontynuuje polecenie w nastpnej linijce.

Klonowanie konfiguracji opublikowanej przez Git

Uyj opcji lb config –config aby sklonowa repozytorium Git, który zawiera konfiguracj 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:

```
$ 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 w drzewie katalogów config. Na przykad, nie nieoficjalne prekompilowane obrazy tworzy si dodajc po prostu –archive-areas ”main contrib non-free”.

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

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

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

`live-systems.org`. Jeli równie opucisz opcjonalny przyrostek `.git`, rozpoczynajc nowy obraz przy uyciu tej konfiguracji jest to tak proste:

352

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

Klonowanie caego repozytorium `live-images` (obrazów live) sciga konfiguracj uywan dla kilku rónych obrazów. Jeli masz ochot na budow innego obrazu po zakoczeniu pracy z pierwszym, przejd do innego katalogu i ponownie w miar potrzeb dokonaj zmian. 353

W kadym przypadku naley pamita, e za kadym razem trzeba bdzie budowa obraz jako super-uytkownik: `lb build` 354

Dostosowywanie zawartoci

355

Opis dostosowywania

Ten rozdzia zawiera przegld rónych sposobów, w jaki mona dostosowa system live.

Konfiguracja podczas kompilacji vs. podczas uruchamiania systemu

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

Opcje konfiguracyjne w czasie budowania s opisane w podrczniku man na stronie `lb config`. Opcje konfiguracyjne w czasie rozruchu opisane s w podrczniku man na stronach live-boot i live-config. Chocia pakiety startowe live-boot i live-config s zainstalowane w systemie live, który budujesz, zaleca si równie zainstalowa je w systemie budowania dla atwego odniesienia podczas Twojej pracy przy konfiguracji. Jest to bezpieczne, poniewa aden z zawartych w nich skryptów nie bdzie wykonywany, chyba e system zostanie skonfigurowany jako system live.

Etapy kompilacji

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

W kadym z tych etapów istnieje szczególna sekwencja, w której stosuje si polecenia. S one usytuowane w taki sposób, aby zapewni modyfikacjom bycie uoonym w rozsdny sposób. Na przykad, w etapie **chroot**, prekonfiguracja (ang. preseeding) jest stosowana, zanim zostan zainstalowane jakiekolwiek pakiety, pakiety s instalowane zanim jakiekolwiek lokalnie zawarte pliki zostan kopiowane, a haki s wprowadzane póniej, gdy wszystkie materiay s ju na miejscu.

Uzupenienie lb config plikami

Mimo, e `lb config` tworzy konfiguracj katalogów w `config/`, aby osign swoje cele, moe by konieczne udostpnienie dodatkowych plików w podkatalogach `config/`. W zalenoci od tego, gdzie pliki s przechowywane w konfiguracji, mog by skopiowane do systemu plików systemu live lub do binarnego obrazu systemu plików, lub mog zosta zapewnione konfiguracje w czasie budowy systemu, które byoby kopotliwie do przekazania jako opcje wiersza polecenia. Mona zawrze rzeczy takie jak niestandardowe listy pakietów, niestandardowa grafika lub inny skrypt do uruchomienia zarówno w czasie kompilacji jak i w czasie startu systemu, zwikszajc ju znaczn elastyczno debian-live swoim wasnych kodem.

Zadania dostosowywania

Kolejne rozdziay s podzielone na rodzaje zada dostosowywania, który uytkownicy zazwyczaj wykonuj: Dostosowywanie instalacji pakietu, Dostosowywanie zawartoci i Dostosowywanie ustawie regionalnych i jzyka obejmuj tylko niektóre z rzeczy, które moesz zrobi.

Dostosowywanie instalacji pakietów

368

Dostosowywanie instalacji pakietów

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

róda pakietu

Dystrybucja, dziay archiwum i tryb

The distribution you choose has the broadest impact on which packages are available to include in your live image. Specify the codename, which defaults to ${testing} 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:

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

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

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

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

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

Serwery lustrzane dystrybucji

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

Serwery lustrzane dystrybucji uywane podczas budowania obrazu

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

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

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

Serwery lustrzane dystrybucji uyte podczas uruchomienia

The `–mirror-binary*` options govern the distribution mirrors placed in the binary image. These may be used to install additional packages while running the live system. The defaults employ `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.

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

Dodatkowe repozytoria

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

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

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

```
deb http://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.

Jeeli takie pliki istniej to zostan wybrane automatycznie.

Naley równie umieci klucz GPG uywany do podpisywania repozytorium w `config/archives/twój-k`

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.

Wybieranie pakietów do instalacji

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

Lista pakietów

Listy pakietów s skutecznym sposobem wyraenia, które pakiety powinny zosta zainstalowane.Skadnia list obsuguje instrukcje warunkowe, które uatwiaj budowanie list i dostosowywanie ich do wykorzystania w wielu konfiguracjach. Nazwy pakietów mog by take wstrzykiwane do listy za pomoc pomocników powoki w czasie kompilacji.

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

Uywanie metapakietów

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

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

This supercedes the older predefined list method supported in `live-build` 2.x. Unlike predefined lists, task metapackages are not specific to the Live System project. Instead, they are maintained by specialist working groups within the distribution and therefore reflect the consensus of each group about which packages best serve the

needs of the intended users. They also cover a much broader range of use cases than the predefined lists they replace.

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

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

Oprócz tych, znajdziesz jeszcze inne metapakiety do rónych celów. Niektóre s podzbiorami szerszych pakietów zada, jak `gnome-core`, podczas gdy inne s indywidualne specjalistyczne czci czystego Debiana mieszanki, takie jak metapakiety `education-*`. Aby wywietli wszystkie metapakiety w archiwum, naley zainstalowa pakiet `debtags` i wywietli wszystkie pakiety z tagiem `role::metapackage` w nastpujcy 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 przykad, aby zainstalowa `ia32-libs` jeeli wybrano `–architectures amd64`:

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

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

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

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

```
#if ARCHIVE_AREAS contrib non-free
vrms
#endif
```

Zagniedanie instrukcji warunkowych jest nie obsugiwane.

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 jzykowe

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.

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

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

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

Rodzaj jdra i wersja

One or more kernel flavours will be included in your image by default, depending on the architecture. You can choose different flavours via the `–linux-flavours` option. Each flavour is suffixed to the default stub `linux-image` to form each metapackage name which in turn depends on an exact kernel package to be included in your image.

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

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

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

Niestandardowe jdra

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

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

If you opt to build the kernel packages without the matching metapackages, you need to specify an appropriate `-linux-packages` stub as discussed in Kernel flavour and version. As we explain in Installing modified or third-party packages, it is best if you include your custom kernel packages in your own repository, though the alternatives discussed in that section work as well.

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

Uyj startowego dysku RAM.

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

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

Instalowanie zmodyfikowanych pakietów lub pakietów innych firm

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

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

Istniej dwa sposoby instalacji niestandardowych pakietów:

`packages.chroot`

Uywanie niestandardowego repozytorium APT

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

Uywanie packages.chrootdo instalacji niestandardowych pakietów

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

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

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

Nie jest moliwe uycie bezpiecznego APT.

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

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

Uywanie repozytoriium APT aby zainstalowa niestandarkowe pakiety

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

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

Niestandardowe pakiety i APT

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

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

Konfigurowanie APT podczas kompilacji

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

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/
```

Podkrcanie 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.lis` 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`: 495

496

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

Dostosowywanie zawartoci

497

Dostosowywanie zawartoci

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

Uwzgldnianie

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

Lokalne uwzgldnianie chroot: Pozwoli Ci doda lub zastpi pliki w systemie plików chroot/live. Zobacz Lokalne uwzgldnianie chroot/live, aby uzyska wicej informacji.

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

Prosz zobaczy Definicje aby uzyska wicej informacji na temat rónicy midzy obrazami ”binarnymi” a obrazami ”live”.

Lokalnie uwzgldniane w chroot/live

Lokalnie uwzgldnione w chroot mog by uywane, aby doda lub zastpi pliki w systemie plików chroot/live, tak aby mogy by one uyte w systemie live. Typowym zastosowaniem jest dostarczenie szkieletowego katalogu uytkownika (`/etc/skel`) uywanego przez system live do tworzenia katalogu domowego uytkownika live. Innym jest dostarczanie plików konfiguracyjnych, które mog by po prostu dodane lub podmienione w obrazie bez przetwarzania; zobacz Lokalne haki chroot/live, czy potrzebne jest przetwarzanie.

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

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

Konfiguracja bdzie mie nastpujcy ukad:

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

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

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

Lokalnie uwzgldniane dane binarne

Aby zawiera materiay takie jak filmy lub dokumentacja na systemie plików nonika tak, aby byy one dostpne od razu po woeniu nonika bez uruchamiania systemu live, moesz uy lokalnego uwzgldniania danych binarnych. Dziaa to w podobny sposób do lokalnego uwzgldniania w chroot/live. Zaómy na przykad, e pliki `/video_demo.*` to filmy demonstacyjne systemu live opisanego przez i dowizane przez stron indeksu HTML. Wystarczy skopiowa materia z `config/includes.binary/` w nastpujcy sposób:

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

Pliki te pojawi si teraz w katalogu gównym nonika live.

Haki

Haki pozwalaj poleceniom aby byy wykonywane w etapach chroot i binarnym kompilacji w celu dostosowania obrazu.

Lokalne haki chroot/live

Aby uruchomi komendy na etapie chroot, naley utworzy skrypt hak z przyrostkiem `.hook.chroot` zawierajcym polecenia w katalogu `config/hooks/`. Hak bdzie dziaa w chroot po reszt konfiguracji chroot zastosowano, wic pamitaj, aby zapewni, e konfiguracja zawiera wszystkie pakiety i pliki do Twoich potrzeb haka w celu uruchomienia. Zobacz przykadowe skrypty hak chroot dla rónych zada dostosowywania wspólnej chroot przewidzianych w # {/ usr / share / doc / live-build / przykady / haki} #, który mona skopiowa lub linku do korzystania z nich w swoim wasnym konfiguracji.

Haki podczas uruchamiania

Aby wykona polecenia przy starcie systemu, moesz dostarczy haki live-config, jak wyjaniono w jego podrczniku man w sekcji ”Customization” (ang. dostosowywanie). Przeanalizuj wasne haki live-config dostarczone w `/lib/live/config/`, zwracajc uwag na numery sekwencji. Nastpnie dodaj swój wasne hak z odpowiednim prefiksem numeru sekwencji, albo jako lokalnie uwzgldnione w chroot w `config/includes.chroot/lib/live/conf` lub w postaci niestandardowego pakietu omówione w Instalowanie zmodyfikowanych pakietów lub pakietów innych firm.

Lokalne haki binarne

Aby uruchomi komendy na etapie binarnym, naley utworzy skrypt hak z przyrostkiem `.hook.binary` zawierajcym polecenia w katalogu `config/hooks/`. Hak bdzie uruchomiony po wszystkich innych binarnych poleceniach, ale przed binary_checksums; ostatnim poleceniem. Polecenia haka nie dziaaj w rodowisku chroot, wic naley zadba, aby nie modyfikowa adnych plików poza drzewem kompilacji albo moe to uszkodzi system kompilacji! Zobacz przykadowe skrypty binarne dla rónych typowych binarnych zada dostosowywania przewidzianych w `/usr/share/doc/live-build/examples/hooks`, które mona skopiowa lub utworzy dowizanie symboliczne aby skorzysta z nich w swojej wasnej konfiguracji.

Wstpne ustawienie pyta Debconfa (Preseeding)

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

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

Dostosowywanie zdarze podczas uruchamiania systemu

527

Dostosowywanie zdarze podczas uruchamiania systemu

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 podrczniku man pakietu live-config.

Personalizacja uytkownika live

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 materiay 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.

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.co`

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

lub uyj `live-config.user-default-groups=audio,cdrom,dip,floppy,video,plugdev,netdev,powerdev,sc` 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:

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

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

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/`

Ustawianie lokalizacji i jzyka

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

- generowanie plików lokalizacji
- ustawienie konfiguracji klawiatury

Domylne 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.

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

Wiele lokalizacji moe by okrelone w postaci listy rozdzielonej przecinkami.

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

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

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

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

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

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

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

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

Wiele wartoci oddzielonych przecinkami moe by przypisane do kadego z parametrów `keyboard-*`, z wyjtkiem `keyboard-model`, który przyjmuje tylko jedn warto. Prosz przejrze podrcznik man `keyboard(5)` aby uzyska wicej szczegóów i przykadów zmiennych `XKBMODEL`, `XKBLAYOUT`, `XKBVARIANT` i `XKBOPTIONS`. Jeli podano wiele `keyboard-variants` (ang warianty klawiatury), bd one dopasowane jeden do drugiego przez warto `keyboard-layouts`

(patrz opcja `setxkbmap(1) -variant`). Puste wartoci s dozwolone; na przykad aby zdefiniowa dwa ukady, domylny US QWERTY oraz drugi US Dvorak, zastosuj:

555

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

Persistence

556

Odmian Live CD jest preinstalowany system, który uruchamia si z noników tylko do odczytu, takich jak cdrom, gdzie operacje zapisu i modyfikacje nie przetrwaj restartów sprztowych hosta, na którym jest uruchomiony. 557

System live jest uogólnieniem tego paradygmatu, a tym samym wspiera media inne prócz pyt CD; ale dalej jako jego domylne zachowanie, naley uwaa operacje tylko do odczytu a wszekie zmiany, w czasie dziaania s tracone podczas zamykania systemu. 558

'Persistence' (ang. trwao) to wspólna nazwa dla rónych rodzajów rozwiza dla zapisania niektórych lub wszystkich danych midzy restartami podczas uywania i wprowadzania zmian do systemu. Aby zrozumie, jak to dziaa to przydatna bya by wiedza, e nawet wtedy, gdy system jest uruchamiany i dziaa z nonika tylko do odczytu, to modyfikacje plików i katalogów s zapisywane na zapisywalnych nonikach, typowo dysk RAM (tmpfs) a dane w pamici RAM nie przetrwaj restartu. 559

Dane przechowywane na tym ramdysku powinny by przechowywane na zapisywalnym trwaym noniku takim jak lokalny dysk, lokalny udzia sieciowy lub nawet na sesji wielosesyjnego dysku CD-RW/DVD-RW. Wszystkie te noniki s obsugiwane w systemach live na róne sposoby i wszystkie, oprócz ostatniej wymagaj specjalnego parametru startowego okrelonego w czasie startu systemu: `persistence`. 560

Jeli ustawiony jest parametr startowy `persistence` (a `nopersistence` nie jest ustawiony), lokalne noniki (np. dyski twarde, napdy USB) bd przeszukane w celu znalezienia woluminów trwaoci podczas startu systemu. Moliwe jest ograniczenie, jakiego typu woluminy trwaoci bd wykorzystane przez okrelenie pewnych parametrów startowych opisanych w podrczniku man live-boot(7). Wolumin trwaoci moe by kadym z wymienionych: 561

partycja, identyfikowana po nazwie GPT. 562

system plików, identyfikowany po etykiecie. 563

plik obrazu zlokalizowany na kadym obsugiwanym systemie plików (nawet na partycji NTFS innego systemu), identyfikowany po nazwie pliku. 564

Etykiet dla woluminu musi by `persistence`, ale bdzie ignorowane, dopóki nie bdzie zawarty w katalogu gównym plik o nazwie `persistence.conf`, który jest uywany by peni dostosowa wolumin persistence, to znaczy, e wskazuje si w nim katalogi, w których chcesz zapisa na woluminie zmiany przy restarcie. Zobacz Plik persistence.conf, aby uzyska wicej szczegóów. 565

Oto kilka przykadów, jak przygotowa wolumin, aby móg by on uyty z opcj persistence. Moe to by, na przykad, partycja ext4 na dysku twardym lub na noniku wymiennym stworzona przez, np.: 566

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

Zobacz równie Wykorzystanie przestrzeni pozostaej na noniku USB.

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

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

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

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

Po utworzeniu pliku obrazu, na przykad, aby sprawi by katalog `/usr` by prway, ale tylko zapisywa zmiany wprowadzone w tym katalogu, a nie ca zawarto `/usr`, mona uy opcji ”union”. Jeli plik obrazu znajduje si w katalogu domowym, naley skopiowa go do katalogu gównego systemu plików na dysku twardym i zamontowa go w `/mnt` w nastpujcy sposób:

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

Nastpnie utwórz plik `persistence.conf` dodajc zawarto i odmontowujc plik obrazu.

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

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

Plik persistence.conf

Partycj z etykiet `persistence` naley skonfigurowa za pomoc pliku `persistence.conf`, aby dowolne katalogi stay si trwae. Ten plik, znajdujcy si w gównym katalogu systemu plików partycji, kontroluje które katalogi s trwae i w jaki sposób.

To jak niestandardowe wierzchnie zamontowania s skonfigurowane jest opisane w szczegóach w podrczniku man persistence.conf(5), ale ten prosty przykad powinien by wystarczajcy dla wikszoci zastosowa. Powiedzmy, e chcemy, aby nasz katalog domowy i cache APT byo trwae w pamici podrcznej systemu plików ext4 na partycji /dev/sdb1:

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

Nastpnie uruchamiamy ponownie komputer. Podczas pierwszego uruchomienia za- 582
warto /home i /var/cache/apt zostanie skopiowana do woluminu trwaoci i od tej pory wszystkie zmiany w tych katalogach bd przechowywane na tym woluminie. Naley pamita, e wszelkie cieki wymienione w pliku persistence.conf nie mog zawiera spacji lub specjalnych komponentów cieki: . i .. . Ponadto, ani /lib, /lib/live (lub którykolwiek z jego podkatalogów) ani / nie moe zosta utrwalony za pomoc wasnych punktów montowania. Jako obejcie tego ograniczenia mona doda / union do pliku persistence.conf w celu osignicia penej trwaoci.

Uywanie wicej ni jednego magazynu persistence 583

Istniej róne sposoby korzystania z wielu magazynów trwaoci (ang. persistence) 584
dla rónych zastosowa. Na przykad, przy uywanie kilku magazynów w tym samym czasie lub wybranie tylko jednego, sporód rónych, do bardzo specyficznych zastosowa.

Kilka rónych niestandardowych woluminów-nakadek (z wasnymi plikami persistence.conf) 585
moe by uywane w tym samym czasie, ale jeeli kilka woluminów tworzy ten sam katalog trwaym, bdzie uywany tylko jeden z nich. Jeli jakie dwa punty montowania s "zagniedone" (np. jeden jest podkatalogiem drugiego) to katalog parent (ang. rodzic) zostanie zamontowany przed katalogiem child (ang. dziecko) tak e jeden punkt montowania nie bdzie ukryty przed innym. Zagniedone niestandardowe woluminy s problematyczne, jeeli s wymienione w tym samym pliku persistence.conf. Zobacz podrcznik man persistence.conf(5), jak radzi sobie z tym przypadkiem, jeli naprawd potrzebujesz (Wskazówka: zazwyczaj nie potrzebujesz).

Jednym z moliwych przypadków uycia: Jeli chcesz przechowywa dane uytkownika np. 586
/home i dane superuytkownika tj. /root na rónych partycjach, utwórz dwie partycje z etykiet persistence i dodaj plik persistence.conf} na kad z nich, tak #{# echo "/home" > persistence.conf na pierwszej partycji, przez co bdz zapisywane pliki uytkownika i # echo "/root" > persistence.conf na drugiej partycji, która bdzie przechowywa pliki superuytkownika. Wreszcie, naley uy parametru startowego persistence.

Jeli uytkownik bdzie potrzebowa wiele magazynów trwaoci tego samego typu dla 587
rónych miejsc lub dla celów testowych, takich jak magazyny private i work parametr startowy persistence-label uyty w poczeniu z parametrem persistence pozwoli na wiele unikatowych magazynów trwaoci. Przykadem moe by, jeli uytkownik chciaby uy partycji trwaoci oznaczonej private dla prywatnych danych, takich jak zakadki w przegldarce lub innych typów danych, to mógby uy parametrów startowych:persistence persistence-label=private. A do przechowywania danych zwizanych z prac, takich jak dokumenty, projekty badawcze lub inne rodzaje, mógby skorzysta z parametrów startowych: persistence persistence-label=work.

Wane jest, aby pamita, e kada z tych partycji, private i work, take potrzebuje pliku 588

persistence.conf. Podrcznik man pakietu live-boot zawiera wicej informacji o tym, jak korzysta z tych etykiet z zapisanymi nazwami.

Using persistence with encryption

Korzystanie z funkcji trwaoci (ang. persistence) oznacza, e niektóre poufne dane mog zosta naraone na ryzyko. Zwaszcza jeli trwae dane s przechowywane na urzdzeniu przenonym, takim jak pamici USB lub zewntrzne dyski twarde. To jest miejsce, gdzie przydatne staje si szyfrowanie. Nawet jeli caa procedura moe wydawa si skomplikowana, ze wzgldu na liczb kroków, które naley podj, to jest bardzo atwo obsugiwa szyfrowane partycje z live-boot. Aby móc korzysta z **luks**, który jest obsugiwanym typem szyfrowania, musisz zainstalowa *cryptsetup* zarówno na maszynie tworzenia zaszyfrowanych partycji, a take w systemie live, który bdzie uywa szyfrowanej trwaej partycji.

Aby zainstalowa *cryptsetup* na twoim komputerze:

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

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

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

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

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

przejdzmy do szczegóów dla tych wszystkich, którzy nie s zaznajomieni z szyfrowaniem. W poniszym przykadzie mamy zamiar uy partycji na dysku USB, która odpowiada /dev/sdc2. Naley zaznaczy, e naley ustali, która partycja jest jeden t, któr masz zamiar uywa w tym konkretnym przypadku.

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

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

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

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

Nastpnym krokiem jest wypenienie urzdzenia zerami przed utworzeniem systemu plików:

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

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.

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

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

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

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

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

Potem odmontuj punkt montowania:

```
# umount /mnt
```

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

```
# cryptsetup luksClose live
```

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.

Dostosowywanie obrazu binarnego

615

Dostosowywanie obrazu binarnego

Programy adujce (ang. Bootloadery)

live-build uywa *syslinux* i niektórych jego pochodnych (w zalenoci od typu obrazu) w domylnym programie adujcym (ang. bootloader). Mona je atwo dostosowa do wasnych potrzeb.

W celu wykorzystania penego motywu, skopiuj `/usr/share/live/build/bootloaders` do `config/bootloaders` i edytuj tam te pliki. Jeli nie chcesz si martwi modyfikacj wszystkich obsugiwanych konfiguracji programu adujcego (ang. bootloader), tylko zapewnienie lokalnego zmodyfikowan kopi jednego z typu programów, np. * {isolinux} * w # {config / programy adujce / isolinux} # wystarczy te, w zalenoci od przypadku uycia.

Podczas modyfikacji jednego z domylnych motywów, jeli chcesz korzysta z indywidualnego obrazu ta, który bdzie wywietlany razem z menu startowym, dodaj obraz splash.png 640x480 pikseli. Nastpnie usu plik splash.svg.

Istnieje wiele moliwoci, jeli chodzi o wprowadzanie zmian. Na przykad, pochodne syslinux maj domylnie skonfigurowany limit czasowy (ang. timeout) na 0 (zero), co oznacza, e wstrzymaj si one na czas nieokrelony na w ich ekranie powitalnym a do nacinicia klawisza.

Aby zmieni limit czasowy podczas rozruchu w domylnym obrazie `iso-hybrid` wystarczy zmieni domylny plik **isolinux.cfg** okrelajc limit czasu (ang. timeout) w jednostkach 1/10 sekundy. Zmodyfikowany **isolinux.cfg** uruchamiajcy rozruch po piciu sekundach byby podobny do tego:

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

Metadane ISO

Podczas tworzenia binarnego obrazu ISO9660, mona korzysta z nastpujcych opcji, aby doda róne metadane tekstowe do obrazu. To moe pomóc w identyfikacji wersji lub konfiguracji obrazu bez uruchamiania go.

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

`LB_ISO_PREPARER/--iso-preparer NAME`: Pole to powinno opisywa przygotowujcego obraz, zwykle z kilkoma danymi kontaktowymi. Domyln wartoci tej opcji jest uywana wersja live-build, które moe póniej pomóc przy debugowaniu. Maksymalna dugo tego pola wynosi 128 znaków.

`LB_ISO_PUBLISHER/--iso-publisher NAME`: Pole to powinno opisywa wydawc obrazu, zwykle z kilkoma danymi kontaktowymi. Maksymalna dugo tego pola wynosi 128 znaków.

`LB_ISO_VOLUME/-iso-volume NAME`: Pole to powinno opisywa ID woluminu obrazu. Jest uywane jako etykieta widoczna dla uytkownika na niektórych platformach, takich jak Windows i Apple Mac OS. Dugo maksymalna dla tego pola to 32 znaków.

Dostosowywanie Instalatora Debiana

630

Dostosowywanie Instalatora Debiana

Obrazy systemów live mog by zintegrowane z Instalatorem Debiana. Istnieje wiele rónych typów instalacji, rónicych si tym, co jest zaczone w obrazie i jak dziaa instalator.

Prosz zwróci uwag na uwane wykorzystanie wielkich liter, podczas odnoszenia si do ”Debian Installer” (ang. Instalatora Debiana) w tej sekcji - kiedy jest stosowany tak, e odnosi si wyranie do oficjalnego instalatora dla systemu Debian, a nie cokolwiek innego, to jest czsto postrzegana w skrócie jako ”d-i”.

Typy Instalatora Debiana

Trzy gówne rodzaje instalatora to:

Instalator Debiana ”Normal”: To jest normalny obraz systemu live z oddzielnym jdrem i initrd, który (gdy wybrany z odpowiedniego menu programu adujcego) uruchamia do standardowej instancji Instalatora Debiana, tak jakby pobrano obraz pyty Debiana i uruchomio si go. Obrazy zawierajce system typu live i inny niezaleny instalator s czsto okrelane jako ”poczone obrazy” (ang. combined images).

Na takich obrazach, Debian jest instalowany przez pobieranie i instalowanie pakietów .deb za pomoc *debootstrap*, z lokalnych mediów lub jakiego opartego na sieci, w wyniku czego domylny systemu Debian jest instalowany na dysku twardym.

Cay proces moe by zapisany w pliku preseed i dostosowany na wiele rónych sposób, przeczytaj odpowiednie strony w instrukcji Instalatora Debiana, aby uzyska wicej informacji. Kiedy utworzysz dziaajcy plik preseed, live-build moe automatycznie umieci go w Twoim obrazie i wczy go dla Ciebie.

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

Instalacja bdzie przebiega w identyczny sposób do instalacji ”normalnej” opisanej powyej, ale na tym samym etapie instalacji, zamiast uywa *debootstrap* aby pobra i zainstalowa pakiety, systemu plików obrazu live zostanie skopiowany do celu. Jest to moliwe dziki specjalnym pakiecie udeb zwanym live-installer.

Po tym etapie, Instalator Debiana kontynuuje normalnie, instalujc i konfigurujc elementy, takie jak programy adujce i lokalnych uytkowników, itp.

Uwaga: aby zapewni wsparcie dla wpisów zarówno normalnych i instalatora live w bootloaderze (ang. programie adujcym) na tym samym noniku live, naley wyczy instalatora live przez wpis w pliku preseed `live-installer/enable=false`.

Instalator Debiana ”Desktop”: Niezalenie od wybranego rodzaju Instalatora Debiana, `di` moe zosta uruchomiony z pulpitu, klikajc na jego ikon. Jest bardziej przyjazny uytkownikowi w niektórych sytuacjach. W celu skorzystania z tej opcji, pakiet debian-installer-launcher musi zosta uwzgldniony.

Naley pamita, e domylnie, live-build nie obejmuje obrazów Instalatora Debiana w obrazach, musi to by sprecyzowane przez `lb config`. Równie naley równie pamita, e aby dziaa ”Desktop Installer” (ang. instalator uruchamiany z poziomu pulpitu) to

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

```
$ 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

Jak opisano w Podrczniku 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 koniecznoci rcznego wprowadzania odpowiedzi, podczas instalacji. Pozwala to w peni zautomatyzowa wikszo rodzajów instalacji, a nawet oferuje kilka funkcji niedostpnych w zwykych instalacjach." Takie dostosowanie jest najlepiej osigane uywajc live-build poprzez umieszczenie konfiguracji w pliku `preseed.cfg` zawartym w `config/includes.installer/`. Na przykad, aby ustawi lokalizacj na `en_US`:

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

Dostosowywanie zawartoci Instalatora Debiana

Do celów dowiadczalnych lub debugowania, moesz zechcie doczy lokalnie zbudowany element `di` w paczce udeb. Naley umieci je w `config/packages.binary/`, aby wczy je do obrazu. Równie dodatkowe lub zamienne pliki i katalogi mog by zawarte w initrd instalatora, w sposób podobny do Uwzgldnianie lokalne live/chroot, poprzez umieszczenie materiaów w `config/includes.installer/`.

Projekt

651

Wnoszenie wkadu do tego projektu

652

Wnoszenie wkadu do tego projektu

Wnoszc swój wkad naley jasno okreli posiadacza jego praw autorskich i zawrze wszelkie stosowane owiadczenie licencjonowania. Naley pamita, e aby zmiany byy zaakceptowane, wkad musi by licencjonowany na tej samej licencji co reszta dokumentów, a mianowicie, GPL w wersji 3 lub nowszej.

Wkad do projektu, taki jak tumaczenia i poprawki, s bardzo mile widziane. Kady moe bezporednio zaangaowa si w repozytoriach, jednak prosimy wysa wiksze zmiany do listy mailingowej, aby omówi je w pierwszej kolejnoci. Patrz rozdzia kontakt aby uzyska wicej informacji.

${projekt} uywa Git jako systemu kontroli wersji i zarzdzania kodem ródowym. Jak wyjaniono w repozytorium Git s dwie gówne gazie rozwoju: **debian** i **debian-next**. Kady moe wprowadzi zmiany do gazi **debian-next** w repozytoriach ive-boot, live-build, live-config, live-images, live-manual i live-tools.

Jednake istniej pewne restrykcje. Serwer odrzuci:

Non fast-forward pushes.

Zmiany wymagajce scalenia.

Dodawanie lub usuwanie tagów lub gazi rozwojowych.

Nawet jeeli wszystkie zmiany mog by póniej zweryfikowane, prosimy aby uywa zdrowego rozsdku i tworzy dobre zmiany opisane dobrym komentarzem.

Pisz komentarze do zmian, które skadaj si z penych, sensownych zda w jzyku angielskim, poczwszy od duej litery, a koczcych si kropk. Zwykle te komentarze zaczynaj si od ”Fixing/Adding/Removing/Correcting/Translating/...”. (”Naprawianie/Dodawanie/Usuwanie/Korygowanie/Tumaczenie/ itd. ... ”.)

Pisz dobre komenarze do zmian. Pierwsza linia musi by dokadne podsumowanie treci popeni, która zostanie uwzgldniona w licie zmian (ang. changelog). Jeli musisz zrobi kilka wyjanie, napisz je niej pozostawiajc pusty wiersz po pierwszej lini, a nastpnie kolejny pusty wiersz po kadym akapicie. Linie kadego akapitu nie powinny przekracza 80 znaków.

_ * Wysyaj zmiany osobno. To znaczy; nie mieszaj niepowizanych ze sob rzeczy w tej samej zmianie. Doda osobn zmian dla kadej rzeczy, któr zmieniasz.

Wprowadzanie zmian

W celu wysania zmian do repozytoriów, naley wykona nastpujc procedur. Tutaj uywamy live-manual jako przykadu, wic zastp go nazw repozytorium, z którym chcesz pracowa. Aby uzyska szczegóowe informacje na temat edycji podrcznika live-manual zobacz: Współtworzenie tego dokumentu.

Pobierz publiczny klucz do wprowadzania zmian:

```
$ 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 nastpujc sekcj do twojej konfiguracji klienta openssh:

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

Sprawd i sklonuj kopi live-manual przez through ssh:

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

Upewnij si, e masz ustawionego autora i adres email w konfiguracji Git:

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

Wane: Pamitaj, e powinno si wprowadza wszelkie zmiany wycznie w gazi **debian-next**.

Wprowad zmiany. W tym przykadzie bdzie najpierw napisa nowy dzia dotyczcy stosowania plastrów, a nastpnie przygotowa si do popenienia dodawanie plików i pisanie popenienia wiadomo tak:

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

Wylij poprawki na serwer:

```
$ git push
```

Zgaszanie bdów

680

Zgaszanie bdów

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

Dla niecierpliwych:

Zawsze naley sprawdzi najpierw aktualizacje statusu obrazu na naszej stronie internetowej 「http://live-systems.org/」, w poszukiwaniu znanych problemów.

Przed wysaniem raportu o bdzie zawsze spróbuj odtworzy problem z **najnowsz wersj** gazi live-build, live-boot, live-config i live-tools, których uywasz (jak najnowsza wersja 4.x live-build, jeli uywasz live-build 4).

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

Znane problemy

Poniewa Debian **testing} * i Debian *{unstable** dystrybucjami ruchomymi, w przypadku okrelenia jednej z nich jako dystrybucji systemu docelowego, kompilowanie nie zawsze zakoczy si sukcesem.

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

Obecnie znane problemy wymienione s w ramach sekcji ”status” na naszej stronie internetowej w 「http://live-systems.org/」.

To jest poza zakresem tej instrukcji, aby szkoli si poprawnie identyfikowa i naprawia problemy pakietów z dystrybucji rozwojowych, jednak istniej dwie rzeczy, których zawsze mona spróbowa: Jeli kompilowanie nie powiedzie si, gdy docelow dystrybucj jest **testing**, to spróbuj z **unstable**. Jeli **unstable** równie nie dziaa, to przywró do dystrybucji **testing** i przypnij nowsz wersj wadliwego pakietu z wersji **unstable** (patrz Przypinanie APT aby uzyska szczegóy).

Przebuduj od zera

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

Uywaj aktualnych pakietów

Uywanie przestarzaych pakietów moe powodowa powane problemy podczas próby odtworzenia (i ostatecznego rozwizania) problemu. Upewnij si, e system na którym kompilujesz jest aktualny i wszelkie pakiety zawarte w obrazie równie.

Zbierz potrzebne informacje

Prosz dostarczy wystarczajc ilo informacji z raportem. Obejmujce co najmniej, dokadn wersj live-build, gdzie napotkano bd i kroki, jak go odtworzy. Prosz uy zdrowego rozsdku 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 nastpujcych informacji:

- Architektura systemu hosta
- Dystrybucja systemu hosta
- Wersja live-build na systemie hosta
- Version of *debootstrap* on the host system
- Architektura systemu live
- Dystrybucja systemu live
- Wersja live-boot na systemie live
- Wersja live-config na systemie live
- Wersja live-tools na systemie live

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

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

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

Dodatkowo, aby wykluczy inne bdy, zawsze dobrym pomysem jest, aby spakowa do archiwum tar swój katalog `config/` i przesa go gdzie (**nie** naley wysa go jako zacznik do listy), tak aby mona spróbowa odtworzy napotkane bdy. Jeli sprawia to trudno (np. ze wzgldu na rozmiar) mona uy wyjcia z polecenia `lb config –dump`, które tworzy podsumowanie 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 regionalnych angielskich, np. uruchom polecenia live-build ze zmiennymi `LC_ALL=C` lub `LC_ALL=en_US`.

Wyizoluj prawdopodobn wad, jeli to moliwe

Jeli to moliwe, naley wyizolowa przypadek do najmniejszej zmiany, które generuje bd. Nie zawsze jest atwo to zrobi, wic jeli nie moesz doda takiej informacji do raportu, nie martw si. Jednake, jeli uytkownik 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 trudnoci z sortowaniem, które z Twoich zmian wygeneroway bd, moe to znaczy, e uwzgldniono za duo w kadym zestawie zmian i powinno si raczej ksztaci w mniejszych krokach.

Wybierz odpowiedni pakiet dla którego zgaszasz bd

Jeli nie wiesz, który komponent jest odpowiedzialny za bd, lub jeli bd jest ogólnym bdem dotyczcym systemów live, mona wypeni raport bdu dotyczcy pseudo-pakietu debian-live.

Jednak bdziemy wdziczni, jeli postarasz si zawzi pole moliwoci, w których moe pojawia si bd.

W czasie budowania podczas adowania pocztkowego (bootstrapping)

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.

W obu przypadkach nie jest to bd w systemie live, ale w samym Debianie i prawdopodobnie nie moemy naprawi go bezporednio. Prosz zgosi taki bd jako dotyczcy narzdzia adowania pocztkowego (ang. bootstraping tool) lub uszkodzonego pakietu.

W czasie budowania podczas instalacji pakietów

live-build instaluje dodatkowe pakiety z archiwum Debiana i w zalenoci od uywanej dystrybucji Debian i codziennego stanu archiwum, moe si to nie powie. Jeli bd pojawi si w tym miejscu, naley sprawdzi, czy bd jest powtarzalny w normalnym systemie.

Jeli jest to przypadek, gdzie bd nie wystpuje w systemie live, ale w Debianie - zgo go jako dotyczcy wadliwego pakietu. Uruchomienie *debootstrap* niezalenie od samej kompilacji systemu live lub uruchomianie `lb bootstrap – debug` daje wicej informacji.

Ponadto, w przypadku korzystania z lokalnego serwera lustrzanego i/lub jakichkolwiek serwerów proxy i dowiadczania problemów prosimy zawsze najpierw spróbowa odtworzy czynoci z uyciem oficjalnego serwera lustrzanego.

W czasie uruchamiania

Jeli obraz nie uruchamia si, naley zgosi go do listy wraz z informacjami wymaganymi w Zbierz potrzebne informacje. Nie zapomnij wspomnie, jak/kiedy dokadnie obraz nie zadziaa, czy by uyty za pomoc wirtualizacji lub rzeczywistego sprztu. Jeli korzystasz z technologii wirtualizacji w jakiejkolwiek formie, prosz zawsze uruchomi go na prawdziwym sprzcie przed zgoszeniem bdu. Zapewnienie zrzutu ekranu awarii jest równie bardzo pomocne.

W czasie gdy system jest ju uruchomiony

Jeli pakiet zosta zainstalowany, ale nie dziaa gdy system live faktycznie dziaa, jest to prawdopodobnie bd w systemie live. Jednake:

Spróbuj wykona par kroków

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

Naley zwróci szczególn uwag na list mailingow systemów live, jak równie strona gówna, poniewa zawieraj one prawdopodobnie najbardziej aktualne informacje. Jeli takowa informacja istnieje, zawsze naley zawrze odniesienie do niej w swoim raporcie o bdzie.

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

Gdzie zgasza bdy

${project} ledzi wszystkie bdy w systemie ledzenia bdów (BTS). Aby uzyska informacje na temat korzystania z tego systemu, zobacz 「https://bugs.debian.org/」. Moesz te przesa bdy uywajc polecenia #{reportbug} z pakietu o tej samej nazwie.

Ogólnie rzecz biorc, naley zgasza bdy podczas kompilacji: jako dotyczce pakietu live-build, bdy podczas uruchamiania: live-boot oraz bdy w czasie dziaania systemu live: jako dotyczce live-config. Jeli nie jeste pewien, który pakiet bdzie odpowiedni lub potrzebujesz wicej pomocy, przed zoeniem zgoszenia bdu, zgo raport dotyczcy pseudo-pakietu debian-live. Zajmiemy si wtedy nim i przypiszemy do odpowiedniego pakietu.

Naley pamita, e bdy znalezione w dystrybucji pochodzcych od Debiana (takich jak Ubuntu, i inne) **nie** powinny by zgaszane do BTS Debiana, chyba e bdy te mog by odtworzone w Debianie przy uyciu oficjalnych pakietów Debiana.

Styl Kodowania

737

Styl Kodowania

Rozdzia ten dokumentuje styl kodowania uywany w systemach live.

Kompatybilno

Nie wolno uywa skadni lub semantyki, które jest unikalne dla basha. Na przykad, uycie ukadu konstrukcji.

Nie uywaj podzbiorów POSIX'a - na przykad, uywaj $(foo) zamiast'foo'.

Moesz sprawdzi swoje skrypty uywajc 'sh -n' i 'checkbashisms'.

Upewnij si, e cay kod powoki dziaa z 'set-e'.

Wcicia

Zawsze uywaj tabulatorów zamiast spacji.

Zawijanie

Generalnie linie maj maksymalnie 80 znaków.

Uywaj zakocze lini "typowych dla Linuxa":

le:

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

Dobrze:

```
if foo
then
        bar
fi
```

To samo dotyczy funkcji:

le:

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

Dobrze:

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

Zmienne

Zmienne wystpuj zawsze zapisane drukowanymi literami.

Zmienne wykorzystane w live-build zawsze zaczynaj si przedroskiem LB_.

Wewntrzne tymczasowe zmienne w live-build naley rozpocz 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 ${FOO} zamiast $FOO.

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

_ * Dla zachowania spójnoci, naley zawsze uywa znaków cytatu podczas przypisywania wartoci do zmiennych:

le:

```
FOO=bar
```

Dobrze:

```
FOO="bar"
```

Jeli zastosowane jest wiele zmiennych, przytocz cae wyraenie:

le:

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

Dobrze:

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

Róne

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

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

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

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

Procedury

783

Procedury

Rozdzia ten dokumentuje procedury dla ${project} dla rónych zada, które wymagaj wspópracy z innymi zespoami w Debianie.

Gówne wydanie

Wydawanie nowej stabilnej wersji gównej Debiana zawiera wiele rónych zespoów pracujcych razem, aby tak si stao. W pewnym momencie, na zespó live dochodzi do pewnego momentu i buduje obrazy systemów live. Wymagania, aby to zrobi to:

Serwer lustrzany zawierajcy wydane wersje dla Debiana i archiwum debian-bezpieczestwa, które mog uzyska dostp buildd debian-live.

Nazwy obrazu musz by znane (np. debian-live-WERSJA-ARCH-FLAVOUR.iso).

Dane z debian-cd musz zosta zsynchronizowane (listy wykluczajce udeb).

Obrazy s budowane i przechowywane na cdimage.debian.org.

Wydanie Docelowe

Ponownie potrzebujemy zaktualizowanych serwerów lustrzanych Debiana i debian-security.

Obrazy s budowane i przechowywane na cdimage.debian.org.

Wylij wiadomo z obwieszczeniem.

Ostatnie Wydanie Docelowe Debiana

Pamitaj, aby dostosowa zarówno chroot i serwery lustrzane z pakietami binarnymi przy budowie ostatniego zestawu obrazów dla wydania Debiana po to zosta przeniesione z ftp.debian.org do archive.debian.org. W ten sposób stare prekompilowane obrazy live bd wci uyteczne bez modyfikacji dokonanych przez uytkownika.

Szablon obwieszczenia dla wydania docelowego

Email z obwieszczeniem dla wydania docelowego moe by wygenerowany przy uyciu poniszego szablonu i nastpujcego polecenia:

```
$ 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'
```

Prosz dokadnie sprawdzi wiadomo przed wysaniem i przekaza j innym, aby dokonali korekt.

802

```
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>.
```

Repozytorium Git

803

Repozytorium Git

Lista wszystkich dostpnych repozytoriów dla ${project} mona znale na stronie 「http://live-systems.org/git/repositories」. Adres URL projektu git ma posta: `protocol://live-systems.org/git/repository`. Tak wic, w celu sklonowania live-manual, uruchom:

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

Lub

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

Lub

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

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

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

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

Drzewo git skada si z wielu rónych gazi. Gazie, które szczególnie wymagaj powicenia uwagi to **debian** i **debian-next**, poniewa zawieraj one rzeczywist pracy, które ostatecznie bdzie znajdowa si w kadej nowej wersji.

Po sklonowaniu kadego z istniejcych repozytoriów, bdziesz w gazi **debian**. To jest waciwe, aby móc przyjrze si stanowi najnowszej wersji projektu, ale przed rozpoczciem pracy wane jest, aby przej do gazi **debian-next**. Aby to zrobi:

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

Ga **debian-next**, która nie zawsze porusza si do przodu, gdzie wszystkie zmiany s najpierw wprowadzane przed poczeniem w gazi **debian**. Aby dokona analogii, to jest jak poligon dowiadczalny. Jeli pracujesz w tej brany i potrzebujesz wykona polecenie pull (wycign), bdzie trzeba uy `git pull –rebase`, tak aby lokalne modyfikacje zostay zachowane przy wyciganiu z serwera, a nastpnie Twoje zmiany zostan wprowadzone na szczycie wszystkich innych.

Obsuga wielu repozytoriów

Jeli masz zamiar sklonowa kilka repozytoriów systemów live i chcesz przej do gazi **debian-next** od razu aby sprawdzi najnowszy kod, napisa poprawk lub przyczyni si z tumaczeniem powiniene wiedzie, e serwer git zapewnia `mrconfig`. Plik,który uatwia obsug wielu repozytoriów. Aby z niego korzysta musisz zainstalowa pakiet *mr* a po tym, uruchomi:

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

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

Przykady

822

Przykady

823

Przykady

W tym rozdziale omówiono przykady 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 przykady.

Uywanie przykadów

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

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

```
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

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

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

Nie mona tego zrobi atwiej ni tak:

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

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

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

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

Zakadajc, e wszystko poszo dobrze, po jakim czasie, biecy katalog bdzie zawiera `live-image-i386.hybrid.iso`. Ten hybrydowy obraz ISO mona uruchomi bezporednio na maszynie wirtualnej, jak opisano w Testowanie obrazu ISO z Qemu i Testowanie obrazu ISO z VirtualBox, lub jeszcze odpowiednio nagrany obraz na nonikach optycznych lub urzdzenia flash USB, w sposób opisany w Nagrywanie obrazu ISO na noniku fizycznym i Kopiowanie hybrydowego obrazu ISO na nonik USB.

Samouczek 2: Narzdzie przegldarka

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

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

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

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

Zbuduj ponownie obraz jako super-uytkownik, zachowujc log jak to opisano w Samouczku 1:

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

Jeszcze raz, zweryfikuj czy obraz jest OK i przetestuj go jak to opisano w Samouczku 1.

Samouczek 3: Spersonalizowany obraz

Przykad uycia: Stwórz projekt spersonalizowanego obrazu zawierajcego twoje ulubione oprogramowanie tak aby móg go zabra ze sob gdziekolwiek pójdziesz i zapisujcy sukcesywnie zmiany, kiedy tego potrzebujesz oraz zmiany w konfiguracji.

Poniewa bdziemy zmienia nasz indywidualny obraz wprowadzajc wiele zmian, chcemy ledzi te zmiany, próbujc rzeczy eksperymentalnych i ewentualnie przywracajc je, jeli co nie wyjdzie, bdziemy trzyma nasz konfiguracj w popularnym systemie kontroli wersji `git`. Bdziemy równie wykorzystywa najlepsze praktyki autokonfiguracji poprzez skrypty `auto` jak opisano w Zarzdzanie konfiguracj.

Pierwsza zmiana

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

Edtuj `auto/config` tak, aby zawiera:

```
#!/bin/sh

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

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

```
$ lb config
```

Teraz uzupenij swoj lokaln list pakietów:

```
$ 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 wikszoci 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*.

A teraz, zbuduj obraz:

```
# lb build
```

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

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

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

Druga zmiana

W tej zmianie, bdziemy sprzta nasz pierwszy zbudowany obrazu, dodawa pakiet *vlc* do naszej konfiguracji, budowa ponownie, testowa i potwierdza zmiany.

Polecenie `lb clean` oczyci wszystkie wygenerowane pliki z poprzedniej kompilacji z wyjtkiem pamici podrcznej (cache), co oszczdza koniecznoci ponownego pobierania pakietów. To gwarantuje, e kolejne polecenie `lb build` ponownie uruchomi wszystkie etapy regeneracji pliki z naszej nowej konfiguracji.

```
# lb clean
```

Teraz doczpakiet *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 jeeli jeste usatysfakcjonowany wprowad nastpn zmian:

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

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

Doszlimy do koca naszej serii samouczka. Chocia moliwe jest o wiele wicej rodzajów personalizacji, nawet tylko za pomoc kilku funkcji pokazanych w tych prostych przykadach, moe by stworzona niemal nieskoczona rónorodno obrazów. Pozostae przykady w tym rozdziale obejmuje kilka innych przypadków uycia zaczerpnite z zebranych dowiadcze uytkowników systemów live.

Kiosk-klient serwera VNC

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

Stwórz katalog kompilacji i stwórz wewntrz szkielet folderów konfiguracji, wyczajc zalecane, aby utworzy minimalny system. A nastpnie utwórz dwie pocztkowe listy pakietów: pierwsz wygenerowan ze skryptu dostarczonego przez live-build o nazwie `Pakiety` (patrz Wygenerowane listy pakietów), a drug uwzgldniajc 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 Podkrcanie APT, w celu zaoszczdzenia miejsca moe trzeba ponownie doda niektóre polecane pakiety do prawidowej pracy obrazu.

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

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

W tym przykadzie okazao si, e musimy ponownie obj kilka pakietów zalecanych przez live-config i live-boot: `user-setup` do funkcji autologowania i `sudo` jako istotnego przy zamykaniu systemu programu. Poza tym, moe by przydatne, 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` dla domylnego uytkownika, 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:

```
# lb build
```

Korzystaj.

Bazowy obraz dla nonika USB z 128MB pamici.

Przykad uycia: Stwórz domylny obraz z usunitymi niektórymi komponentami tak, 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 dokona pewnych kompromisów midzy rozmiarem a funkcjonalnoci. W tym przykadzie,

przytniemy tylko tyle, aby zrobi miejsce dla dodatkowego materiau medialnego w rozmiarze 128MB, ale robienia czegokolwiek, co mogoby zniszczy integralno zawartych pakietów, np. czyszczenie danych ustawie regionalnych poprzez pakiet *localepurge*, lub inne tego typu *inwazyjne* optymalizacje. Szczególnie godne uwagi, jest uycie `–debootstrap-options` by stworzy minimalny system od podstaw.

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

Aby uczyni by obraz pracowa prawidowo, musimy ponownie doda przynajmniej dwa pakiety, które zostay pominite przez opcj `–apt-recommends false`. Zobacz Podkrcanie APT w celu zaoszczdzenia 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 jzyku oraz instalator

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

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

Naszym pocztkowym problem jest odkrycie nazw odpowiednich zada jzykowych. Obecnie, live-build nie moe nam w tym pomóc. Chocia moemy mie szczcie i znale to metod prób i bdów, to jest narzdzie, `grep-dctrl`, które moe by uyte do ustalenia to z opisów zada w tasksel-data tak wic, aby przygotowa si upewnij si, e masz obie te rzeczy:

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

Teraz moemy rozpocz wyszukiwanie odpowiedniego zadania. najpierw:

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

Dziki temu poleceniu dowiadujemy si, e zadanie nazywa si po prostu niemiecki (ang. german). Teraz, aby znale 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 bdziemy generowa lokalizacj **de_CH.UTF-8** i wybiera ukad klawiatury **ch**. Teraz poskadajmy kawaki razem. Przypominamy sobie Korzystanie z metapakietów e metapakiety s poprzedzone przedrostkiem `task-`, po prostu okrelimy te parametry rozruchowe dotyczce jzyka, a nastpnie dodamy standardowe pakiety priorytetowe i wszystkie wykryte metapakiety zada do naszej listy pakietów w nastpujcy 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

Wytyczne dla autorów

Ten rozdzia zajmuje si pewnymi ogólnymi rozwaaniami, które naley wzi pod uwag podczas pisania dokumentacji technicznej dla live-instrukcji. S one podzielone na funkcje jzykowe oraz zalecane procedury.

Uwaga: Autorzy powinni najpierw przeczyta Współtworzenie tego dokumentu

Funkcje jzykowe

Uyj zwykego angielskiego

Naley pamita, e wysoki procent czytelników nie s rodzimymi uytkownikami jzyka angielskiego. Wic jako generaln zasad spróbuj uywa krótkich, sensownych zda, zakoczonych kropk.

To nie znaczy, e trzeba korzysta z uproszczonego, naiwnego stylu. Proponuje si unika, na ile to moliwe, zoonych zda podrzdnych, które czyni tekst trudny do zrozumienia dla nie rodzimych uytkowników jzyka angielskiego.

Rónorodno jzyka angielskiego

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.

Spodziewamy si, e róne odmiany jzyka mog miesza si bez tworzenia nieporozumie, ale ogólnie naley stara si by spójnym i przed podjciem decyzji o uyciu brytyjskiego, amerykaskiego lub innego dialektu jzyka angielskiego wedle uznania, prosz przyjrze si, jak inni ludzie pisz i postara si ich naladowa.

Bd zbalansowany

Nie by stronniczy. Unikaj w tym odniesienia do ideologii zupenie niepowizanych z live-manual. Pismo techniczne powinny by moliwie jak najbardziej neutralne. Jak to jest w naturze pimiennictwa naukowego.

Bd poprawny politycznie

Staraj si unika seksistowskiego jzyka, jak to jest tylko moliwe. Jeli potrzebujesz, odwoania do trzeciej osoby liczby pojedynczej najlepiej uy jakiej formy bezosobowej lub ”wy” zamiast ”on” , ”ona” lub niewygodnych wynalazków, takie jak ”mógby/mogaby”, ”byem/am” i podobnych.

Bdz zwizy

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.

Oszczd pracy tumaczcym

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

Bd zgodny

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

Be spójny

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

Bd opisowy

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

Sownik

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

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

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

Korzystanie ze sownika kolokacji jest zalecane. S one bardzo pomocne, kiedy przychodzi do znajomoci sów najczciej wystpujcych razem.

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

Faszywi przyjaciele, idiomy i inne wyraenia idiomatyczne

Uwaaj na faszywych przyjació. Bez wzgldu na to, jak biegy jeste w obcym jzyku nie mona pomóc wpadajc od czasu do czasu w puapk tzw. ”faszywych przyjació”, sowa, które wygldaj 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

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...

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

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.

Procedury

Przetestuj przed zapisaniem

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.

Przykady

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

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

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

Linki zewntrzne

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

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

Unikaj nadawania marki i rzeczy, które naruszaj licencj zgodnie z któr podrcznik ten zosta opublikowany

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

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

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

Napisz pierwszy szkic, przegldnij go, edytuj, popraw i cofnij zmiany jeeli wymaga tego sytuacja

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

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

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

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

Rozdziay

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

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

Znaczniki

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

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

To s przykady znaczników, które mog okaza si uyteczne:

- Dla pogrubienia uyj:

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

powoduje: **foo** lub **foo**. Uyj tego by wyszczególni okrelone sowa kluczowe.

- Dla kursywy uyj:

```
/{foo}/
```

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

- Dla czcionki o staej szerokoci uyj:

```
#{foo}#
```

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

- Dla bloków z kodem uyj:

```
code{

 $ foo
 # bar

}code
```

powoduje:

```
$ foo
# bar
```

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

Wytyczne dla tumaczy

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

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.

Uwaga: Tumacze powinni równie przeczyta Wspótworzenie tego dokumentu. W szczególnoci rozdzia Tumaczenie

Wskazówki tumaczenia

Komentarze

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

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

UT, Uwagi Tumacza

It is perfectly acceptable however, to include a word or an expression in brackets in the translated text if, and only if, that makes the meaning of a difficult word or expression clearer to the reader. Inside the brackets the translator should make evident that the addition was theirs using the abbreviation ”TN” or ”Translator’s Note”.

Wyraenia w trybie bezosobowym

Documents written in English make an extensive use of the impersonal form ”you”. In some other languages that do not share this characteristic, this might give the false impression that the original texts are directly addressing the reader when they are actually not doing so. Translators must be aware of that fact and reflect it in their language as accurately as possible.

Faszywi przyjaciele

Puapka ”faszywych przyjació” wyjanionych wczeniej szczególnie dotyczy tumaczy. Dwukrotnie sprawdzi znaczenie podejrzanych faszywych przyjació w razie wtpliwoci.

Znaczniki

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.

Bloki kodowe

Mimo, e bloki z kodem s zazwyczaj nieprzetumaczalne, zawarcie ich w tumaczeniach jest jedynym sposobem, aby zdoby 100% kompletne tumaczenie. I mimo, e oznacza to wicej 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.

Nowe linijki

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.

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

Nieprzetumaczalne cigi znaków

Tumacze nigdy nie powinni tumaczy:

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

- Nazw programów

- Komend podanych jako przykad

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

- Linków

- cieek dostpnowych