第 24 回 関西 Debian 勉強会資料

関西 Debian 勉強会担当者 山下 尊也

2009 年 6 月 28 日

1 Introduction

山下 尊也

関西 Debian 勉強会は Debian GNU/Linux のさまざまなトピック (新しいパッケージ、Debian 特有の機能の仕組、Debian 界隈で起こった出来事、などなど）について話し合う会です。

目的として次の三つを考えています。

- ML や掲示板ではなく、直接顔を合わせる事での情報交換の促進
- 定期的に集まれる場所
- 資料の作成

それでは、楽しい一時をお楽しみ下さい。

関西デビアン勉強会

目次

2 ハッカーに一歩近づく Tips：Bash 編

山下康成@京都府向日市

表 1 Bash コマンド一覧

<<コマンドの再実行>>		
1	以前実行したコマンドラインの呼出し	上矢印キー、下矢印キー
2	以前実行したコマンドラインの呼出し	CTRL-P, CTRL-N
3	直前のコマンドの再実行	!!
4	以前実行したコマンドラインに前方一致	!（前方一致文字列）
5	以前実行したコマンドラインに部分一致	!?（部分一致文字列）
6	これまでに実行したコマンドラインの列挙	history
7	ヒストリ番号を用いた過去コマンド指定	!(数字)
8	指定した回数前に実行したコマンド	!（負の数字）
9	実行されるコマンドの確認	:p
<<補完>>		
10	パス名の補完	TAB
11	コマンドの補完	TAB
<<行編集>>		
12	カーソルの左右移動	CTRL-B/左矢印、CTRL-F/右矢印
13	カーソルの左の文字の消去	BackSpace/Delete
14	カーソルの行頭・行末への移動	CTRL-A, CTRL-E
15	カーソル位置の文字を消す	CTRL-D
16	行末まで消去	CTRL-K
17	語頭まで消去	CTRL-W
18	文字の入れ換え	CTRL-T
<<直前のコマンドの一部を再利用する>>		
19	直前のコマンドの最後の引数	!$
20	直前のコマンドの引数	!*
21	直前のコマンドラインの一部を変更	^A^B
22	直前のコマンドラインの一部を変更	（以前のコマンド指定）:s/A/B/
<<ワイルドカード>>		
23	（ドット）で始まる以外に一致	*
24	1 文字に一致	?
<<プログラムの実行結果を利用する>>		
25	パイプ	|
26	リダイレクト	> >> <
27	コマンドの実行結果を文字列として扱う	‘（逆シングルクォート）
<<その他>>		
28	ホームディレクトリを指す	~
29	指定したユーザのホームディレクトリを指す	~ユーザ名
30	（コマンドを中断する）	CTRL-C

2.1 参考資料

- ハッカーに一歩近づく Tips: Bash 編
  `http://www.yamasita.jp/tips/bash/`

3 【CD/DVD/USB メモリ】Debian JP 版 Debian Live を作るよ【netboot も】

のがたじゅん

3.1 はじめに

OS をハードディスクにインストールせずメディアから直接起動して使う「ライブシステム」が知られて久しいですが、Debian にも Debian Live というライブシステムがあります。今回は Debian Live 作成ツールの live-helper を使って、Debian JP 版 Debian Live や関西 Debian 勉強会の配布 DVD を作成する中で、自分好みの Debian ライブシステムを作成する方法やコツなどを解説します。

3.2 Debian Live Project とは

Debian Live の作成の前に、Debian Live Project について紹介します。

Debian Live Project は、ライブシステム作成のためのフレームワーク live-helper と live-initramfs などのライブシステムにまつわるユーティリティを開発するプロジェクトで、Debian の公式サブプロジェクトです。

既存のライブシステム・ディストリビューションとの違いは、ライブシステムを作ってリリースすることより、フレームワークとしてのツールの制作に重きを置いているところでしょうか。

live-helper を使って作られた Debian ライブシステムは Debian Live と呼ばれ、Lenny から同時にリリースされています。

3.3 Debian Live の特徴

Debian Live は、既存のライブシステム・ディストリビューションの欠点を解消するように作られており、以下の特徴を持っています。

- 一つのディストリビューションのみで完結。
- カーネルも含めてパッチを当てたりなど特別なパッケージを使わない。
- リマスタリングではない新しい環境からシステムを作ることができる。
- CD、DVD、USB メモリ起動のシステムだけでなく、ネットワークやインターネット越しに起動するシステムを作ることができる。
- 複数のアーキテキクチャをサポート。
- Debian Installer を含めることができる。

Debian Live は Debian のみで作成できるようにデザインされていますが、自作のパッケージや独自の Apt リポジトリのパッケージを追加して作ることも可能です。

アーキテキクチャについては、現在、正式に対応しているのは i386、amd64、powerpc だけですが、基本的に Debian がサポートしているアーキテキクチャには対応する予定です。サポートしているモードも Debian だけでなく、emdebian や ubuntu にも対応しています。(sid 以上で –mode ubuntu と指定すると不完全ながらも ubuntu live ができます。)

Debian Installer については通常の netinst や businesscard のインストーラに加え、Debian Live のパッケージをそっくりそのままインストールする live もサポートされています。また、計画段階ですが、Debian Live 上から直接 Debian をインストールするためのインストーラの制作も予定されています。

3.4 live-helper について

live-helper とは、Debian Live を作成するためのファイル名の頭に「lh_」とついたシェルスクリプト群です。群というだけあってスクリプトは多数ありますが、この中で Debian Live 作成に使用するコマンドはたった 3 つ。設定のための「lh_config」、作成のための「lh_build」、作業ディレクトリをクリーンナップする「lh_clean」だけです。

残りのスクリプトは、3 つのスクリプトの中から設定に応じて適宜呼ばれるので、通常、意識する必要はありません。

3.5 Debian Live の最初の一歩

それでは live-helper を使った Debian Live の作成の基本について解説します。

3.5.1 live-helper のインストール

live-helper は、すでに Debian のリポジトリに用意されているので、aptitude などを使ってインストールします。パッケージインストールについての注意ですが、live-helper パッケージに Suggests や Recommends されたパッケージを使用する場面が多いので、特に事情が無ければ、すべてインストールしておいてください。

live-helper のインストール

```
# aptitude --with-recommends install live-helper
```

3.5.2 作業の準備

Debian Live 作成のための作業ディレクトリを作成します。ここでは live-work と名づけて作りました。

作業ディレクトリの作成

```
$ mkdir live-work
```

Debian Live の設定ファイルや作成作業のファイルは、すべてカレントディレクトリに置かれます。よく使うディレクトリで Debian Live 作成作業をおこなうと、これらのファイルと通常のファイルが混ざって収集がつかなくなるので、作業の前には専用のディレクトリを用意しましょう。

3.5.3 まず Debian Live を作ってみる

設定には lh_config コマンドを使います。作業ディレクトリに降りて lh_config と入力します。

```
$ cd debian-live/
$ lh_config
```

config と scripts ディレクトリが作成されたはずです。ディレクトリの意味については後ほど説明するので、まずは Debian Live を作成してみましょう。「sudo lh_build」と入力します。(lh_config コマンド以外は管理者権限が必要になるので、コマンドの前に sudo をつけて実行します。)

```
$ sudo lh_build
```

マシンとネットワークの状況にもよりますが 15 分〜30 分ほどで binary.iso、binary.list、binary.package というファイルができているはずです。他には binary、cache、chroot というディレクトリができています。

できていなければ「sudo lh_clean」と入力し、作業途中のディレクトリを消去してから、もう一度試してみてください。

```
$ ls -la
drwxr-xr-x  6 jun  jun        296 2009-06-24 16:24 .
drwxr-xr-x 10 jun  jun        320 2009-06-24 16:08 ..
drwxr-xr-x  2 root root       640 2009-06-24 16:24 .stage
drwxr-xr-x  6 root root       176 2009-06-24 16:24 binary
-rw-r--r--  1 root root 132192256 2009-06-24 16:24 binary.iso
-rw-r--r--  1 root root      2171 2009-06-24 16:24 binary.list
-rw-r--r--  1 root root     11123 2009-06-24 16:23 binary.packages
drwxr-xr-x  6 root root       184 2009-06-24 16:08 cache
drwxr-xr-x 20 root root       600 2009-06-24 16:24 chroot
drwxr-xr-x 22 jun  jun        936 2009-06-24 16:08 config
```

3.5.4 イメージファイルを確認する

生成されたファイルの中に binary.iso というファイルがあります。これが Debian Live の CD イメージファイルです。それでは起動テストをおこないますが、CD-R などに書き込んでのテストはマシンを再起動しなければいけませんし、ライトワンスのメディアを使うのは環境にも良くないので、仮想マシンを使って確認します。

例では qemu を使いましたが、KVM や VirtualBox、VMware Player など好きな仮想化ソフトを使ってかまいません。qemu を使う場合、そのままの状態では遅いので、あらかじめ高速化カーネルモジュールの kqemu を組み込んでおいてください。

kqemu を組み込む

```
# m-a a-i kqemu
# modprobe kqemu
```

qemu 上で Debian Live を実行する

```
$ qemu -cdrom binary.iso -boot d -m 256
```

qemu 上で Debian Live が起動したでしょうか？画面は白黒のコンソール画面、キーボードは英語配列。想像していたものと、だいぶかけ離れた画面が現れたと思います。

これが最小の Debian Live です。ここから自分なりの設定を加えて、自分オリジナルの Debian Live を作り上げます。

3.6 Debian Live の設定

さて、これから本格的な Debian Live の設定に入りますが、Debian Live が作成される手順を知っておくと、より的確に設定することができるので説明します。

Debian Live の作成には lh_build コマンドを使いますが、lh_build は 4 つのコマンドを呼び出しそれぞれの作業をおこないます。その 4 つのコマンドは以下になります。

1. debootstrap によるベースシステムのインストール (lh_bootstrap)
2. ベースシステムに chroot して必要なソフトのインストールや設定をおこなう (lh_chroot)
3. 作成されたシステムを一つのファイルにまとめ、起動できるバイナリイメージを作成 (lh_binary)
4. 作成されたバイナリイメージにソースが必要ならば、ソースをまとめたイメージを作成 (lh_source)

このように bootstrap → chroot → binary → source の順に作業が進み、それぞれのステージにおいて独自の設定をすることができます。

設定ファイルは config ディレクトリ以下にありますが、common、bootstrap、chroot、binary、source の 5 つのファイルは lh_config から設定するので、直接編集しません。

それ以外のディレクトリでは、chroot ステージに関係したものは「chroot_」、binary ステージに関係したものは「binary_」と、それぞれの状態を表すプリフィクスがついているので、参考にしながら場面に応じた設定をしていきます。

3.6.1 設定と作成の基本

Debian Live の設定は lh_config コマンドを使って行います。lh_config –help と入力してみましょう。

```
$ lh_config --help
```

ものすごい数の設定が出てきましたが、すべてを設定する必要はありません。設定をしなければ基本の設定が使われるので必要な箇所のみ変更していきます。

```
$ lh_config \
      --binary-images iso \
      --distribution lenny \
      --language ja \
      --bootappend-live "quiet locale=ja_JP.UTF-8 keyb=jp kmodel=jp106" \
      --mirror-bootstrap "http://ftp.jp.debian.org/debian/" \
      --mirror-chroot "http://ftp.jp.debian.org/debian/" \
      --mirror-chroot-security "http://security.debian.org/" \
      --mirror-binary "http://ftp.jp.debian.org/debian/" \
      --mirror-binary-security "http://security.debian.org/"
```

上から順に説明します。

–binary-images は生成するバイナリイメージの種類を指定します。iso 以外にも usb-hdd など指定できます。–distribution にはディストリビューションを指定。lenny を指定しています。squeeze は現在カーネルパッケージの不整合があるので作れません。–language は言語です。Iceweasel や OpenOffice.org のように言語別パッケージがあるときの判断に利用されます。

–bootappend-live は Debian Live 起動時のブートパラメータを指定します。ここではカーネルのメッセージ出力を抑制する「quiet」と、ロケールとキーボードを日本語の設定にしています。quiet 以外は Debian Live 独自のパラメータです。パラメータ一覧は、live-initramfs パッケージの/usr/share/doc/live-initramfs/parameters.txt を参照してください。

–mirror は Apt の取得先を指定します。ミラーは、それぞれのステージで設定できますが、特別な事がない限り分けて指定する必要はないので、ftp.jp.debian.org と security.debian.org を設定しています。

とても長かったですが、これを毎回設定するのは大変です。毎回設定しないようにするには、どうしたらいいのでしょうか。

lh_config を最初に実行した際、設定を保存する config ディレクトリのほかに、scripts ディレクトリが生成されていたことを覚えているでしょうか。その scripts ディレクトリに設定のための config をスクリプトとして置きましょう。

scripts/config

```
#!/bin/sh

MIRROR_DEBIAN="http://ftp.jp.debian.org/debian/"
MIRROR_SECURITY="http://security.debian.org/"

BOOTOPTION_LIVE="quiet locale=ja_JP.UTF-8 keyb=jp kmodel=jp106"

lh_config noautoconfig \
       --binary-images iso \
       --distribution lenny \
       --language ja \
       --bootappend-live "${BOOTOPTION_LIVE}" \
       --mirror-bootstrap ${MIRROR_DEBIAN} \
       --mirror-chroot ${MIRROR_DEBIAN} \
       --mirror-chroot-security ${MIRROR_SECURITY} \
       --mirror-binary ${MIRROR_DEBIAN} \
       --mirror-binary-security ${MIRROR_SECURITY}
       ${@}
```

この scripts/config スクリプトは、lh_config コマンドを実行したとき、scripts/config スクリプトが存在すれば、再帰的に呼び出され実行される仕組みになっています。

これを見て「わざわざ手間のかかることをしているのはなぜ?」と思った方もいると思います。それには「クリーンな環境からのビルド」と「ビルドの自動化」の二つの理由があります。

「クリーンな環境からのビルド」ですが、live-helper で生成した設定ファイルは今は何も変わりませんが、この先バージョンが上がった場合どうでしょう? オプションが廃止されたり、新しいオプションが追加されるかもしれません。古いバージョンの設定ファイルを使いつづけていると、それらに気づかないまま対応できない以外にも、無用なトラブルを呼び込むかもしれません。

それらを避けるためにも毎回 config ディレクトリを消去して新たに設定を生成する必要があります。

もう一つ「ビルドの自動化」ですが、ライブシステムでは、同一内容で設定が少し違うものが欲しいことがあります。たとえば CD イメージ版と USB メモリ版などです。

そのとき、毎回オプションを指定して作ることは効率が悪いので、共通する設定はスクリプト化しておき、変更点のみ指定して 2 回呼び出せば、クリーンかつ必要なイメージを作成することができます。

scripts/config スクリプトに続いて、lh_build コマンドから呼び出される scripts/build スクリプトと、lh_clean コマンドから呼び出される scripts/clean スクリプトも作っておきましょう。

scripts/build スクリプトでは、ビルド作業のログと生成されるイメージをわかりやすくするためファイル名に作業時間を含めるようにし、scripts/clean スクリプトは、config ディレクトリ内で空のディレクトリも消去するようにしています。

scripts/build

```
#!/bin/sh

IMAGE_PREFIX=debian_live-binary
BUILDDATE=‘date +%Y%m%d%H%M%S‘

lh_build noautoconfig 2>&1 | tee ${IMAGE_PREFIX}-${BUILDDATE}.buildlog

# rename files
if [ -f binary.iso ]; then
   mv binary.iso ${IMAGE_PREFIX}-${BUILDDATE}.iso
elif [ -f binary.img ]; then
   mv binary.img ${IMAGE_PREFIX}-${BUILDDATE}.img
fi
[ -f binary.list ] && mv binary.list ${IMAGE_PREFIX}-${BUILDDATE}.list
[ -f binary.packages ] && mv binary.packages ${IMAGE_PREFIX}-${BUILDDATE}.packages
```

scripts/clean

```
#!/bin/sh

lh_clean noautoconfig --all ${@}

# Remove generated files
rm -f config/binary config/bootstrap config/chroot config/common config/source

# Remove empty directories in config tree
if ls config/*/ > /dev/null 2>&1
then
	rmdir --ignore-fail-on-non-empty config/*/
fi

if [ -d config ]
then
	rmdir --ignore-fail-on-non-empty config
fi
```

これで lh_config と入力すると、いつでも自分の基本設定の状態から作成することができます。sudo lh_build や sudo lh_clean と入力するとログをとりながらビルドしたり、クリーンナップもできます。

さて、自動化といえば Make。ということで簡単ですが Makefile も用意してみました。sudo lh_build なんて長く打たなくても make と入力するだけでビルドできますし、ほんの少し設定を変えるときも Makefile の中で処理することができるので、とても楽になりました。

Makefile

```
all: config build

config: clean
	lh_config

build:
	sudo lh_build

clean:
	sudo lh_clean

distclean: clean
	sudo lh_clean --purge
	sudo rm -f *.iso *.img *.list *.packages *.buildlog *.md5sum
```

もう一つついでに、git で live-helper のレシピを管理するととても楽ですが、その場合、以下のような「.gitignore」を作成しておくとよいでしょう。

```
*.buildlog
*.img
*.iso
*.list
*.md5sum
*.packages
*~
.*~
.stage/
binary/
cache/
chroot/
config/binary
config/bootstrap
config/chroot
config/common
config/source
```

3.7 Debian Live のカスタマイズ

作成環境ができあがったので、具体的なカスタマイズについて述べます。

3.8 パッケージの追加

自分好みの Debian Live にするために最初に始めることはパッケージの追加でしょう。パッケージを追加するにはいくつか方法はありますが順番に説明します。

3.8.1 パッケージを追加する

APT リポジトリにあるパッケージを追加するには–packages オプションを使います。パッケージの指定方法は、パッケージ名をスペースで区切って並べていきます。

```
lh_config --packages "PACKAGE PACKAGE2 ···"
```

自作パッケージを追加するには、config/chroot_local-packages/ディレクトリに deb パッケージを置き、–packages オプションでパッケージ名を指定します。

3.8.2 パッケージリストで追加する

–packages オプションでは、まとまった数のパッケージを追加するには、いくつもパッケージ名を並べないといけないので不便です。よく使われるデスクトップ環境などのパッケージリストはデフォルトで用意されているので、–packages-lists オプションを使って指定します。

パッケージリストの一覧は/usr/share/live-helper/lists/をご覧ください

```
lh_config --packages-lists "gnome"
```

自分で追加したいパッケージ名を並べたパッケージリストを config/chroot_local-packageslists/ディレクトリに用意し、–packages-lists オプションで指定することもできます。

「FILE」という名前のパッケージリストを作成

```
config/chroot_local-packageslists/FILE

$ cat  config/chroot_local-packageslists/FILE
lv manpages-ja nkf
iceweasel-l10n-ja
openoffice.org-help-ja openoffice.org-l10n-ja
ttf-kochi-gothic ttf-kochi-mincho ttf-sazanami-gothic ttf-sazanami-mincho ttf-vlgothic
uim uim-applet-gnome uim-prime uim-qt uim-qt3
```

–packages-lists オプションで「FILE」を指定します。

```
lh_config --packages-lists "FILE"
```

パッケージリスト書式

別のパッケージリストを含める

```
#include <gnome>
iceweasel
```

アーキテクチャで分岐する

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

セクションで分岐する

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

3.8.3 APT リポジトリを追加する

APT リポジトリを追加するには、config/chroot_sources/ディレクトリに Apt-line を書いたファイルと、パッケージの署名を検証する GPG 鍵ファイルを置きます。

Apt-line を書いたファイル名は chroot ステージと binary ステージで別々に指定することができ、chroot ステージの場合には拡張子を「.chroot」、binary ステージでは拡張子を「.binary」にします。GPG 鍵ファイルは拡張子を「.gpg」にしておきます。

```
restricted-debian_multimedia.chroot
restricted-debian_multimedia.chroot.gpg
restricted-debian_multimedia.binary
restricted-debian_multimedia.binary.gpg
```

キーリングパッケージを指定する

GPG 鍵がキーリングパッケージとして配布されている場合は、–keyring-packages オプションにキーリングパッケージを指定します。

```
lh_config --keyring-packages "debian-archive-keyring debian-multimedia-keyring"
```

3.8.4 ファイルを追加する

パッケージではなく、ファイルそのものを追加する場合は config/chroot_local-includes/ディレクトリにファイルを置きます。ファイルを置く場合の注意としては、ディレクトリ構造がそのままコピーされるので、/usr/local/bin/hoge というファイルを置く場合は、config/chroot_local-includes/ディレクトリをトップに見立て usr/local/bin/ディレクトリを作り、その中に hoge を置きます。

/usr/local/bin/hoge を置く場合のディレクトリ構造

```
$ ls chroot_local-includes/usr/local/bin/
hoge
```

3.8.5 カーネルパッケージを追加する

カーネルパッケージを指定してインストールするには、–linux-packages オプションと–linux-flavours オプションを設定します。

```
lh_config --linux-packages "linux-image-2.6 aufs-modules-2.6 squashfs-modules-2.6" --linux-flavours "686"
```

カスタムカーネルパッケージを追加する

カスタムカーネルパッケージを追加するには、カーネルのパッケージ以外にも、ファイルシステムを透過的に重ねる aufs/unionfs パッケージ、Kernel 2.6.28 以前のカーネルで squashfs を使うなら squashfs パッケージなどを用意する必要があります。

1. 用意したパッケージを config/chroot_local-packages/ディレクトリに置く。
2. –linux-packages オプションに”none”、–linux-packages オプションに用意したパッケージそれぞれを書く。–linux-flavours オプションにはパッケージ名のカーネルバージョン以降を書く。

```
 lh_config --linux-packages "none" --linux-packages \
> "linux-image-2.6.26.6-rt11 aufs-modules-2.6.26.6-rt11 squashfs-modules-2.6.26.6-rt11" --linux-flavours "rt11"
```

3.9 設定のカスタマイズ

パッケージを追加したなら、次は設定を追加しましょう。設定の追加にもさまざまなパターンがあるので、順に紹介します。

3.9.1 シェルスクリプトを実行する

Debian Live で設定変更のため一番使われる手法は、シェルスクリプトを実行することです。パッケージのインストールから設定の書き換え、不要ファイルの除去など、ほぼなんでも行うことができます。

シェルスクリプトを実行するには、config/chroot_local-hooks/ディレクトリに実行させたいシェルスクリプトを置きます。

スクリプト例 (Iceweasel の初期設定を変更する)

```
#!/bin/bash
set -e

ICEWEASEL_PREFS=/etc/iceweasel/profile/prefs.js

cat << _EOL_ >>${ICEWEASEL_PREFS}

/* Debian Live tune */
user_pref("browser.cache.disk.parent_directory","/tmp");
user_pref("browser.cache.disk.capacity", 5000);
user_pref("browser.startup.homepage", "http://www.debian.or.jp/");
_EOL_
```

3.9.2 パッチを当てる

設定の変更にはシェルスクリプトを実行して sed で書き換えてもいいですが、パッチを当てたほうが早い場合もあります。パッチを当てるには chroot のトップから patch -p1 で当てられるように作ったパッチファイルを config/chroot_local-patches/ディレクトリに置きます。

パッチファイル例 (/etc/dhcp3/dhclient.conf を変更する)

```
*** chroot/etc/dhcp3/dhclient.conf.orig 2009-04-12 23:03:36.000000000 +0900
--- chroot/etc/dhcp3/dhclient.conf      2009-04-12 23:05:05.000000000 +0900
***************
*** 23,30 ****
       netbios-name-servers, netbios-scope, interface-mtu,
       rfc3442-classless-static-routes;
 #require subnet-mask, domain-name-servers;
! #timeout 60;
! #retry 60;
 #reboot 10;
 #select-timeout 5;
 #initial-interval 2;
--- 23,30 ----
       netbios-name-servers, netbios-scope, interface-mtu,
       rfc3442-classless-static-routes;
 #require subnet-mask, domain-name-servers;
! timeout 0;
! retry 0;
 #reboot 10;
 #select-timeout 5;
 #initial-interval 2;
```

3.9.3 debconf の設定をする

/etc にある設定などはシェルスクリプトやパッチで書き換えることができますが、debconf の設定は db に納められているので直接書き換えることはできません。その場合は config/chroot_local-preseed/ディレクトリに debconf の設定を置いて設定します。debconf の設定は、debconf-utils パッケージに納められている deboconf-get-selections を使って読み出します。

FUSE(Filesystem in Userspace) を使うためにユーザーを fuse グループに登録するには

1. debconf の設定を取り出して config/chroot_local-preseed/user-default-groups.preseed に書き出す

```
$ sudo debconf-get-selections | grep user-default-groups > config/chroot_local-preseed/user-default-groups.preseed
```

2. fuse グループを追加

```
user-setup      passwd/user-default-groups      string  audio cdrom dialout floppy video plugdev netdev powerdev scanner fuse
```

3.9.4 ブートローダーについて

ブートローダーについては i386/amd64 では syslinux と grub が選択できます。

ブートローダーのスプラッシュ画面を変更するには、syslinux の場合は config/binary_syslinux/ディレクトリに、grub の場合は config/binary_grub/ディレクトリに設定を置きます。

grub についてですが、usb-hdd では grub のインストールがまだサポートされていないので、syslinux しか使うことができません。誰か書いてみませんか？

3.9.5 スプラッシュスクリーンについて

最近のディストリビューションは起動時にスプラッシュスクリーンを表示していますが、Debian Live でも可能です。splashy か usplash パッケージをインストールして、–bootappend-live オプションに”vga=788 splash”と加えてください。vga の部分はフレームバッファの解像度なので適宜変更してください。

3.9.6 作成されるファイルとディレクトリについて

lh_config や lh_build が作成するファイルとディレクトリは以下のようになります。

表 2　Debian Live のディレクトリ

.stage/	進行状況を示すフラグが納められる
config/	設定が納められる
scripts/	作成のための config、build、clean スクリプトを置く
cache/	パッケージなどをキャッシュされる
chroot/	chroot イメージの作業ディレクトリ
binary/	binary イメージの作業ディレクトリ

3.10 その他、積み残したことなど

3.10.1 Debian Live を使っていてデータを保存するには

出来上がった Debian Live を使っていて、データを保存しておきたいことがあります。

その場合は、全体を保存するには live-rw、ホームディレクトリのみを保存するなら home-rw というラベル名で、ext2/3 でフォーマットしたパーティションをあらかじめ作成しておき、起動時、ブートパラメータに persistent をつけて起動すると自動的にそれぞれのパーティションをマウントします。

専用パーティションを用意しない場合は、live-snapshot コマンドを使うとデータが保存できるようなのですが、persistent をつけてもなぜか読み込んでくれないので、誰か live-snapshot の使い方を知っていたら教えてください。

3.10.2 live-helper と Debian Live 関連パッケージ

Debian Live 関連するパッケージについては以下のようになります。

Debian 公式パッケージ

- live-helper: Debian Live を作成するためのスクリプト。
- live-initramfs: Debian Live が起動する際に使われる initramfs の中で動くスクリプト。
- live-magic: GUI で Debian Live を作成するプログラム。決まったものしかできないので使うことないと思います。

Debian Live Project のみで配布 (Debian 非公式パッケージ)

- live-manual: マニュアル
- live-initscripts: 起動時にスクリプトを実行するオプションを追加する
- live-helper, live-initramfs, live-magic のスナップショット

Debian Live の開発はとても早いので、興味があるなら、Debian Live Project で配布されているスナップショット版か git のものを使うとよいでしょう。スナップショットの在り処などは、Debian Live Project のリンクページ[*1]に掲載されています。

live-helper は Debian に依存しないように作られています。Debian 以外のシステムでも以下の要件を満たしていれば、live-helper を使って Debian Live を作成することができます。

- Linux 2.6.x
- POSIX 準拠のシェル (bash や dash など)
- 管理者 (root) 権限
- debootstrap もしくは cdebootstrap
- live-helper

3.11 まとめ

Debian Live を使っていて、日本語のまとまった資料があまりないので頑張って書いてみましたが、いかがでしたでしょうか? 今回は CD/DVD/USB メモリ起動に絞って書きましたが、ネットワーク越しのブートや、live-helper の構造など、まだまだ書ききれない部分が残っているので、それはまた日を改めてということで、皆様のお役に立てればさいわいです。

[*1] DebianLiveProject:http://debian-live.alioth.debian.org/links.html

参考文献

[1] Debian Live Project: http://debian-live.alioth.debian.org/

[2] DebianLive - Debian Wiki: http://wiki.debian.org/DebianLive

[3] DebianLive/FAQ - Debian Wiki: http://wiki.debian.org/DebianLive/FAQ

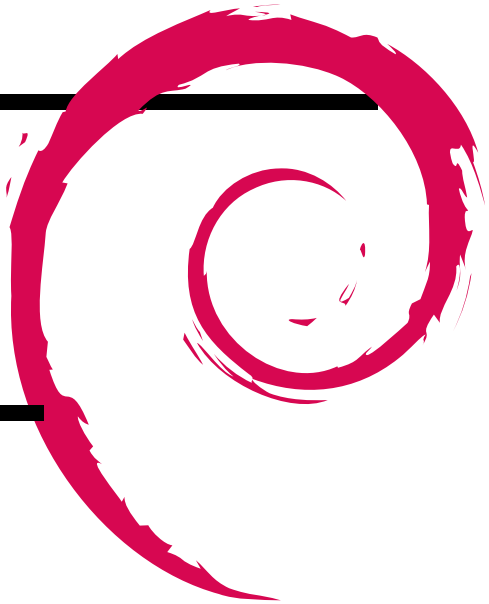
[4] Debian Live Manual: http://live.debian.net/manual/html/index.html

[5] Daniel Baumann - Blog: Debian Live Initiative
http://blog.daniel-baumann.ch/2006/02/14#20060214_debian-live-initiative

[6] 第 34 回東京エリア Debian 勉強会，2007 年 11 月勉強会:
http://tokyodebian.alioth.debian.org/2007-11.html

[7] Software Design 2008 年 8 月号
live-helper で構築するオリジナル Live CD 岩松信洋

[8] Software Design 2009 年 5 月号
インストール，ネットブック活用から独自バージョンの作成まで - Ubuntu の魅力を体感！
6 章：Inside Ubuntu 吉田史
7 章：カスタム CD イメージの作成 小林準



4　今後の予定

山下 尊也

オープンソースカンファレンス 2009 Kansai

7 月 10 日（金）と 11 日（土）に京都コンピュータ学院 京都駅前校にて、オープンソースカンファレンス 2009 Kansai[*2] が開催されます。

関西 Debian 勉強会では、有志が作成した T シャツや冊子の販売、実機などの展示を予定しています。

[*2] http://www.ospn.jp/osc2009-kansai/

5 メモ

関西Debian勉強会

Debian 勉強会資料

2009 年 6 月 28 日　　初版第 1 刷発行
関西 Debian 勉強会　（編集・印刷・発行）