

mkimage-profiles

Michael Shigorin

mkimage-profiles

Michael Shigorin

Содержание

Введение	iv
I. Основы	1
1. Welcome to m-p!	2
2. Переменные make	4
3. Фичи	8
4. Списки пакетов	9
II. Каталоги	11
5. conf.d	12
6. image.in	14
7. features.in	15
8. sub.in	16
8.1. sub.in/main	16
8.2. sub.in/stage1	17
8.3. sub.in/stage2	17
9. pkg.in	18
9.1. pkg.in/lists	18
9.2. pkg.in/lists/tagged	18
10. lib	19
III. Приложения	20
11. Предположения	21
12. Ловля плюх	23
13. Оформление кода	24
14. Сборка образов VM	25
15. QEMU	26

Введение

mkimage-profiles, или *m-p* — результат осмысления и обобщения опыта создания семейств дистрибутивов свободного программного обеспечения на базе ALT Linux.

Цели

- пригодность к применению как очень крутыми хакерами, так и новичками
- относительно низкая трудоёмкость использования
- поощрение совместной работы

Средства

Двухуровневость: метапрофиль более объёмен и сложен, но выгоден для долгосрочной разработки сгенерированный дистрибутивный профиль более легко обозрим и модифицируем как одноразовый форк наследственность на уровне индивидуальных особенностей и образов в целом прозрачность и диагностируемость формирования конфигурации документированность

Примеры использования

Выполняем начальные инструкции по документации

```
git clone git://git.altlinux.org/people/mike/packages/mkimage-profiles.git
cd mkimage-profiles
make icewm.iso
```

Часть I. Основы

Глава 1. Welcome to m-p!

Brief summary

Configurables: `~/mkimage/profiles.mk`;
see `doc/params.txt` and `conf.d/README`

License: GPLv2+, see COPYING

Most docs are in Russian, welcome to learn it or ask for English.

Задача:

- конфигурирование и создание образов на базе ALT Linux

Концепция:

- конфигурация, как и образ — объект постадийной сборки
- метапрофиль служит репозиторием для построения индивидуального профиля, по которому создаётся итоговый образ

Особенности:

- метапрофиль может быть полностью read-only при сборке
- для сборки подыскивается предпочтительно `tmpfs`
- в профиль копируются только нужные объекты; он автономен относительно метапрофиля

Стадии работы:

- инициализация сборочного профиля
- сборка конфигурации образа
- наполнение сборочного профиля
- сборка образа

Объекты:

- дистрибутивы и виртуальные среды/машины:
 - описываются в `conf.d/*.mk` или соответственно `lib/{distro,ve,vm}.mk`
 - могут основываться на предшественниках, расширяя их
 - дистрибутивы также:
 - включают один или более субпрофилей по надобности
 - желательно избегать множественного наследования, см. тж. фичи

- субпрофили:
 - список собирается в `$(SUBPROFILES)`
 - базовые комплекты помещены в подкаталогах под `sub.in/`; их наборы скриптов могут расширяться фичами
- фичи:
 - законченные блоки функциональности (или наборы таковых)
 - описываются в индивидуальных `features.in/*/config.mk`
 - могут требовать другие фичи, а также субпрофили
 - накопительный список собирается в `$(FEATURES)`
 - при сборке `$(BUILDDIR)` содержимое фич добавляется в профиль
- списки пакетов (`*_LISTS`):
 - просьба по возможности избегать дублирования (см. `bin/pkgdups`)
- индивидуальные пакеты (`*_PACKAGES`): см. тж. `conf.d/README`

Результат:

- при успешном завершении сборки образ называется по имени цели и укладывается в `$(IMAGEDIR)`:
 - указанный явно,
 - либо `~/out/` (если возможно),
 - или `$(BUILDDIR)/out/` иначе
- формируются отчёты, если запрошены (`REPORT`)

См. тж.:

- <http://altlinux.org/m-p>
- `doc/`:
 - `params.txt`: переменные, указываемые при запуске сборки
 - `pkglists.txt`: формирование состава образа
 - `features.txt`: обзор подключаемых особенностей

Удачи; что не так — пишите.

Michael Shigorin <mike@altlinux.org [mailto:mike@altlinux.org]>

Глава 2. Переменные make

При запуске на сборку принимается ряд переменных (см. тж. `profiles.mk.sample`):

- **APTCNF**
 - задаёт путь к требуемому `apt.conf`
 - значение: пусто (по умолчанию системный) либо строка
 - см. `../image.in/Makefile`, `apt.conf.sample`
- **ARCH**
 - задаёт целевую архитектуру образов
 - значение: пусто (по умолчанию авто), `i586`, `x86_64`, `arm`, `ppc`
 - см. `../lib/build.mk`
- **ARCHES**
 - задаёт набор целевых архитектур при параметрическом задании `APTCNF`
 - значение: пусто (по умолчанию авто) либо список через пробел
 - см. `../Makefile`, `profiles.mk.sample`
- **BELL**
 - подаёт сигнал после завершения сборки
 - значение: пусто (по умолчанию нет) либо любая строка
 - см. `../lib/build.mk`
- **BUILDDIR**
 - задаёт каталог генерируемого профиля и сборки
 - значение: пусто (по умолчанию авто) либо строка
 - см. `../lib/profile.mk`
- **BUILDDIR_PREFIX**
 - задаёт префикс каталога генерируемого профиля и сборки
 - значение: строка; по умолчанию выбирается алгоритмически
 - см. `../main.mk`

- BUILDLOG
 - задаёт путь к файлу журнала сборки/очистки
 - значение: по умолчанию \$(BUILDDIR)/build.log либо строка
 - см. ../lib/log.mk
- CHECK
 - включает режим проверки сборки конфигурации (без сборки образа)
 - значение: пусто (по умолчанию) либо любая строка
 - см. ../lib/build.mk, ../lib/clean.mk
- CLEAN
 - экономия RAM+swap при сборке в tmpfs, иначе места на диске
 - очистка рабочего каталога после успешной сборки очередной стадии
 - может помешать использовать некоторые варианты отладки
 - значение: любая строка; по умолчанию пусто при DEBUG, иначе 1
 - см. ../lib/clean.mk
- DEBUG
 - включает средства отладки, может отключить зачистку после сборки
 - значение: пусто (по умолчанию), 1 или 2
 - см. ../lib/log.mk, ../lib/clean.mk
- HOMEPAGE, HOMENAME, HOMEWAIT
 - указывают адрес, название и таймаут перехода для домашней страницы
 - значение: корректный URL, строка, целое неотрицательное число
 - см. ../features.in/live/generate.mk (тж. по умолчанию)
- ISOHYBRID
 - включает создание гибридного ISO-образа
 - значение: пусто (по умолчанию) либо любая строка
 - см. ../features.in/pack/config.mk
- NICE
 - понижает нагрузку системы сборочной задачей

- значение: пусто (по умолчанию) либо любая строка
- см. ../lib/build.mk
- REPORT
 - запрашивает создание отчётов о собранном образе
 - значение: пусто (по умолчанию) либо любая строка
 - NB: в силу специфики обработки передаётся только явно
 - см. ../Makefile, ../report.mk, ../lib/report.mk
- ROOTPW
 - устанавливает пароль root по умолчанию для образов виртуальных машин
 - значение: пусто (по умолчанию root) либо строка
 - см. ../features.in/build-vm/image-scripts.d/00root
- SAVE_PROFILE
 - сохраняет архив сгенерированного профиля в .disk/
 - значение: пусто (по умолчанию) либо любая строка
 - см. ../image.in/Makefile
- SORTDIR
 - дополнительно структурирует каталог собранных образов
 - значение: пусто (по умолчанию) либо строка
 - например, `$(IMAGE_NAME)/$(DATE)`
 - см. ../image.in/Makefile
- SQUASHFS
 - определяет характер сжатия squashfs для stage2
 - значение:
 - пусто (по умолчанию) либо normal: xz
 - tight: xz с -Xbcj по платформе (лучше, но дольше — подбор в два прохода)

- fast: gzip/lzo (быстрее запаковывается и распаковывается, меньше степень)
- см. ../features.in/stage2/stage1/scripts.d/03-test-kernel
- STATUS
 - добавляет в имя образа указанный префикс
 - значение:
 - пусто (по умолчанию) либо строка (например, "alpha", "beta")
 - см. ../image.in/Makefile
- VM_SIZE
 - задаёт размер несжатого образа виртуальной машины в байтах
 - значение: пусто (по умолчанию двойной размер чрута) или целое
 - см. ../features.in/build-vm/lib/90-build-vm.mk, ../bin/tar2vm

пример

```
make DEBUG=1 CLEAN=1 syslinux.iso
```

Глава 3. Фичи

Особенности дистрибутива, не учитываемые в пакетной базе или зависящие от переменных времени сборки/установки образа; по необходимости влияют на конфигурацию, приносят с собой или запрашивают скрипты, которые могут быть оформлены как:

- `scripts.d/` или `image-scripts.d/` различных стадий;
- пакеты `installer-feature-*` (тж. <http://www.altlinux.org/Installer/beans>).

В большинстве случаев можно рекомендовать создание `feature` средствами метапрофиля, поскольку при этом дерево кода более удобно для анализа и обновления (и в отличие от *m-p-d* — нет вынужденной необходимости либо контролировать включение нужных фич "вручную" в скриптах по косвенным признакам, либо выносить их в пакеты `installer-feature-*`).

Создание и упаковку `installer-feature-*` можно рекомендовать, если:

- необходимы пакетные зависимости (в т.ч. версии/конфликты);
- требуется компилируемый платформозависимый код (для чего бы...);
- код фичи достаточно специфичен, нетривиален и объёмен, чтобы загромождать метапрофиль было не очень осмысленно;
- фича представляет из себя отдельный мини-продукт, над которым ведётся активная работа (возможно, несколькими людьми).

Стоит избегать изменения пакетных умолчаний в случае, когда их представляется осмысленным и возможным скорректировать в пакете: таким образом они станут более дистрибутивными.

Обратите внимание, что фичи включаются в комплект инкрементально: что добавили, то уже не убрать; поэтому при необходимости следует выделять промежуточные цели сборки, собирающие необходимые фичи и оставляющие те, по которым есть расхождения, на включение ближе к конечной дистрибутивной цели.

Соглашение по именованию таково, что цели `use/ФИЧА` и `use/ФИЧА/...` определяются в файле `features.in/ФИЧА/config.mk` и только в нём.

Глава 4. Списки пакетов

Состав пакетной базы субпрофилей определяется значениями следующих переменных профиля (см. тж. ../conf.d/README; некоторые "*" ниже заэкранированы ради парсера asciidoc):

- main: пакетная база для установки
 - sub.in/main/Makefile, features.in/*/main/lib/*.mk
 - THE_LISTS, BASE_LISTS, MAIN_LISTS
 - THE_GROUPS, MAIN_GROUPS
 - THE_PACKAGES, BASE_PACKAGES, MAIN_PACKAGES, SYSTEM_PACKAGES, COMMON_PACKAGES
 - THE_PACKAGES_REGEX, BASE_PACKAGES_REGEX, MAIN_PACKAGES_REGEX
 - THE_KMODULES, BASE_KMODULES, MAIN_KMODULES, BASE_KMODULES_REGEX
 - KFLAVOURS
- stage2: общая часть installer, live, rescue
 - sub.in/stage2/Makefile, features.in/*/stage2/lib/*.mk
 - SYSTEM_PACKAGES, STAGE2_PACKAGES
 - STAGE1_KMODULES, STAGE1_KMODULES_REGEX, STAGE2_KMODULES, STAGE2_KMODULES_REGEX
 - STAGE1_KFLAVOUR или последний из KFLAVOURS
- installer: компактная "живая" система, содержащая только инсталлятор
 - см. stage2
 - features.in/install2/install2/stage2cfg.mk, features.in/*/install2/lib/*.mk
 - INSTALL2_PACKAGES
- live: пользовательский LiveCD (может содержать также инсталлятор)
 - см. stage2
 - features.in/live/live/stage2cfg.mk, features.in/*/live/lib/*.mk
 - THE_LISTS, LIVE_LISTS
 - THE_GROUPS, LIVE_GROUPS

- THE_PACKAGES, LIVE_PACKAGES, COMMON_PACKAGES
- THE_PACKAGES_REGEX, LIVE_PACKAGES_REGEX
- THE_KMODULES, LIVE_KMODULES
- rescue: спасательный LiveCD
 - см. stage2
 - features.in/rescue/rescue/stage2cfg.mk
 - RESCUE_PACKAGES, COMMON_PACKAGES
 - RESCUE_LISTS
- stage1: ядро и загрузчик второй стадии
 - sub.in/stage1/Makefile, features.in/*/stage1/lib/*.mk
 - STAGE1_PACKAGES, SYSTEM_PACKAGES
 - STAGE1_PACKAGES_REGEX
 - STAGE1_KMODULES_REGEX
 - STAGE1_KFLAVOUR или последний из KFLAVOURS

Часть II. Каталоги

Глава 5. conf.d

Этот каталог содержит включаемые фрагменты конфигурации образов с тем, чтобы было удобнее параллельно разрабатывать специфические образы без излишних merge conflict'ов.

Следует понимать, что основная цель появления mkimage-profiles на свет — это уменьшение "форков" внутри семейства дистрибутивных профилей. Поэтому при возможности следует всё-таки работать над общей базовой частью, включая скриптовые хуки и списки пакетов, а также оптимизировать граф зависимостей между конфигурациями образов.

Просту говоря, сору-paste — тревожный признак.

По переменным (см. тж. ../doc/pkglists.txt):

- для пользовательского окружения (live, main) предназначены THE_PACKAGES, THE_LISTS, THE_GROUPS, THE_PACKAGES_REGEX
- для "обычного общего" (live, main, rescue) есть COMMON_PACKAGES (NB: тоже попадают в базовую установку)
- SYSTEM_PACKAGES стоит применять крайне осторожно — эти пакеты попадут во все стадии, в том числе в образ чувствительной к объёму install2 (в stage1 — только в инструментальный чрут); применяйте для того, что обязано быть и в инсталляторе, и в готовой системе
- для направленного действия служат:
 - STAGE1_PACKAGES, STAGE1_PACKAGES_REGEX (первая стадия загрузки)
 - STAGE2_PACKAGES (инсталлятор и спасательная/"живая" система)
 - INSTALL2_PACKAGES (инсталлятор)
 - BASE_PACKAGES, BASE_LISTS, BASE_PACKAGES_REGEX (базовая система)
 - MAIN_PACKAGES, MAIN_LISTS, MAIN_PACKAGES_REGEX (дополнительные пакеты)
 - LIVE_PACKAGES, LIVE_LISTS, LIVE_PACKAGES_REGEX ("живая" система)
- аналогично по модулям ядра:
 - THE_KMODULES попадут в "пользовательскую" среду (live, main)
 - STAGE1_KMODULES доступны в производных от stage2 (install2, live, rescue)

- `BASE_KMODULES` попадут в установку по умолчанию
- `MAIN_KMODULES` будут доступны для установки с носителя
- `LIVE_KMODULES` предназначены для LiveCD/LiveFlash

Не стоит бояться такого разнообразия, для большинства задач достаточно `THE_*`.

По подстановкам:

- `$(VAR)` подставляются перед их записью в `$(CONFIG)`, который `distcfg.mk`
- `$$ (VAR)` раскрываются позже, при включении `$(CONFIG)` и востребовании значений; в этом случае их значения могут изменяться до окончания конфигурации, а также зависеть от значений других переменных

По спискам пакетов:

- на этапе экспериментирования можно забивать прямо в описание образа
- при фиксации состояния стоит воспользоваться существующими списками, а дополнительные оформить как можно более чётко обособленными по тем задачам, для решения которых они и подобраны
- повторяющиеся логически связанные группы списков может иметь смысл выделить в фичу (см., например, `power` или `x11`)
- если явной фичи не наблюдается, но у группы дистрибутивов намечается заметная общая часть — её можно выделить в промежуточную цель вида `distro/.name`, не являющуюся самостоятельно собираемой

Глава 6. image.in

Этот каталог копируется из метапрофиля в профиль "как есть" и формирует "заготовку" финальной стадии, собирающей собственно образ из результатов работы индивидуальных субпрофилей (для distro) либо непосредственно "на месте" (для ve, vm).

Содержимое files/ копируется в корень образа.

Соответственно для сборки также потребуется одна из фич ../features.in/build-*.

Пакетная база рабочего чрута минимальна (может чуть расширяться фичами — см. ../features.in/repo/lib/50-genbasedir.mk в качестве примера).

Если требуется какая-либо иная обработка чрута, следует предпочитать scripts.d/ — для универсальной обработки скрипт можно добавить здесь, для специфичной — в фичу.

Результат — готовый образ в \$(IMAGEDIR)/.

Глава 7. features.in

Этот каталог содержит т.н. фичи (features, особенности).

Фича — отдельно подключаемая сущность, которая содержит повторно используемые конфигурацию/код и определяет одну из особенностей создаваемого образа. Может зависеть от других фич либо субпрофилей.

Каждая фича должна содержать файл `config.mk`, включаемый в `../main.mk` при построении конфигурации будущего профиля; он может описывать одну или более целей вида `use/*`, дополняющих конфигурацию, и обязан добавить имя фичи в `$(FEATURES)`, для чего создана функция `add_feature`.

На этапе генерации сборочного профиля фичи рассматриваются после инициализации профиля (см. `../image.in/`) и копирования субпрофилей (см. `../sub.in/`). Для каждой фичи, указанной в `$(FEATURES)`, копируются подкаталоги сообразно включенным субпрофилям, а также `lib/` и `{image-,}scripts.d/`; затем выполняются `generate.sh` и `generate.mk` при их наличии.

Если фича дополняет хуками семейство целевых субпрофилей, построенных на одном базовом, можно воспользоваться подкаталогом с именем исходного базового субпрофиля (см. `$src`, `$dst` в `Makefile`).

Рекомендуется давать несколько различающиеся имена скриптам, которые одна и та же фича может добавлять в различные стадии, чтобы они не выглядели одинаково в логе сборки.

Наиболее востребованные цели можно снабжать "ярлычками" вроде `"+icewm"` с тем, чтобы сделать более краткими и выразительными использующие их правила. Просьба не злоупотреблять количеством, такие имена предполагается показывать в интерфейсе к профилю.

Каталог `lib/` является специфическим для фич, определяющих построение конкретного вида образа — см. `build-*/`.

Несложный пример содержится в `00example/`, более близкий к жизни и нынешним пределам возможностей метапрофиля — в `syslinux/`.

См. тж. файлы `README` в каталогах фич (отсутствие — баг!).

Глава 8. sub.in

Этот каталог содержит субпрофили; содержимое затребуемых (названия которых содержатся в значении переменной SUBPROFILES, которую заполняют цели sub/* — см. ../lib/distro.mk) будет скопировано в корневой каталог формируемого профиля.

Просьба ответственно относиться к изменению существующих субпрофилей и вдумчиво — к созданию новых; возможно, достаточно всего лишь оформить нужное новой фичей (см. ../features.in/).

Обратите внимание: поскольку сборка частей дистрибутивного образа и происходит в каталогах субпрофилей, то повторное использование одного простого субпрофиля в рамках сгенерированного профиля штатным образом невозможно. Если требуется создать несколько близких по реализации субпрофилей, изучите stage2 и задействующие его фичи.

Краткое описание существующих вариантов:

- stage1: propagator и загрузчик (совместно с фичей syslinux); типично требуется для инсталляторов, live- и rescue-образов, но может использоваться без добавления таковых в образ, обеспечивая сетевую загрузку второй стадии
- stage2: наиболее сложный технологически субпрофиль, поскольку он является только базовым для получения ряда итоговых частей дистрибутива (install2, live, rescue); задействуется для этого только опосредованно через use/stage2/* и модифицирует stage1 в силу наличия связи между ними (в stage1 попадает образ ядра и firmware, в stage2 — соответствующие модули)
- main: пакетная база, укладываемая на образ (NB: поскольку рабочий чрут в этом случае не содержит ничего, кроме пакетов, добавлять image-scripts.d/* смысла нет, только scripts.d/*)

Шаблонное правило сейчас определено в ../lib/distro.mk, поскольку субпрофили востребованы только дистрибутивами.

8.1. sub.in/main

Этот каталог содержит субпрофиль main, собирающий пакетную базу для локальной инсталляции дистрибутива из полученного образа, включая необязательные пакеты; в distro/live-builder применяется как локальный репозиторий для сборки.

Подбирает:

- SYSTEM_PACKAGES, COMMON_PACKAGES, BASE_PACKAGES, BASE_LISTS: в установку по умолчанию;

- `MAIN_PACKAGES`, `MAIN_LISTS`: дополнительные пакеты.

В `image-scripts.d` смысла нет, только `scripts.d`, т.к. рабочий чрут не содержит исполняемых файлов.

Не следует использовать этот субпрофиль напрямую, для добавления пакетного репозитория в образ предназначена фича `use/repo/main`.

Результат — каталог `ALTLinux/RPMS.main` для копирования в образ.

8.2. sub.in/stage1

Этот каталог содержит субпрофиль первой стадии загрузки; здесь место `syslinux` (загрузчик) и `propagator` (ориентировка на местности, вытягивание второй стадии с `CD/FTP/...`).

Скрипты запускаются извне формируемого образа (`scripts.d/`); следует крайне бережно относиться к объёму этой стадии.

Обратите внимание: если не указать явно требуемый вариант ядра посредством `STAGE1_KFLAVOUR`, будет взят последний из перечисленных в `KFLAVOURS`; если не указать явно регэкс, описывающий требуемые в инсталляторе модули, посредством `STAGE1_KMODULES_REGEX` — будет подмножество модулей из `kernel-image` (упаковываются в `syslinux/alt0/full.cz`).

Требуется для инсталляционных, `live`- и `rescue`-образов, соответствующими фичами подключается автоматически (в силу зависимости `stage2` от `stage1`).

Результат — каталог `syslinux/` для копирования в образ.

8.3. sub.in/stage2

Этот каталог содержит общий базовый субпрофиль "живой" второй стадии, используемый для сборки образов `install2`, `live`, `rescue` (возможно, нескольких одновременно в составе одного дистрибутива).

Зависимость на него стоит прописывать в таких фичах; сама по себе (без нужного `stage2cfg.mk`) смысла не имеет.

Обратите внимание, что набор потенциально доступных в `stage1` модулей ядра для `stage2` может быть расширен (`STAGE2_KMODULES`).

Результат — соответственно названный файл со `squashfs`, подлежащий копированию в итоговый образ.

NB: смонтированный образ доступен в такой системе как `/image/`.

Глава 9. pkg.in

Этот каталог содержит все возможные списки пакетов и описания групп, которые по мере необходимости копируются из метапрофиля в формируемый профиль.

9.1. pkg.in/lists

Этот каталог содержит списки пакетов, копируемые из метапрофиля в создаваемый профиль по необходимости (определяется по наличию имён списков в переменных *_LISTS, см. реализацию в Makefile).

Список .base является особенным (формирует базовую систему, см. <http://www.altlinux.org/Alterator-pkg>); он создаётся из содержимого ряда переменных (см. Makefile).

Подкаталог tagged/ содержит тегированные списки, имена которых удобно получать функцией tags() (см. ../../lib/functions.mk).

Для выявления дубликатов в составе списков служит 'make pkgdups'; пытаться избежать дублей на 100% скорее бесполезно, но бродячие устойчивые группы пакетов могут заслуживать выделения отдельным списком.

9.2. pkg.in/lists/tagged

Этот каталог содержит тегированные списки; на данный момент реализация (../../bin/tags2lists) требует, чтобы каждый из тегов был отдельным словом, состоящим из символов из набора [a-zA-Z0-9_] (внимание: не используйте в слове "-"); рекомендуется разделять слова "+".

Применение: дополнение жёстко статически заданной функциональности более "плавающим" в долгосрочном плане результатом раскрытия списка тегов.

Реализация является экспериментальной и требует утряски с ../groups/; комментарии и помощь всячески приветствуются.

pkg.in/groups

Этот каталог содержит описания групп, копируемые из метапрофиля в создаваемый профиль по необходимости (только фигурирующие в списке, которым является значение переменной MAIN_GROUPS).

В данный момент перенесено почти 1:1 из mkimage-profiles-desktop, требует увязки с ../lists/tagged/.

Глава 10. lib

Этот каталог содержит вспомогательные makefiles, обеспечивающие основную функциональность создания конфигурации образа и генерации соответствующего профиля для сборки; см. тж. ../conf.d/.

Следует помнить, что будучи включаемыми в ../main.mk, они работают в каталоге верхнего уровня.

Часть III. Приложения

Глава 11. Предположения

Некоторые фрагменты кода закладываются на определённое поведение других частей `mkimage-profiles` либо содержание переменных.

NB: пути приводятся от верхнего уровня; проект в целом предполагает ALT Linux 6.0+ и GNU make 3.81+ (на которых и разрабатывается), но может быть портирован вместе с `mkimage`. Если что-либо не работает или не собирается, стоит проверить на Sisyphus (`mkimage`, `make`, `hasher`, собственно пакетная база), поскольку именно на нём происходит основная разработка `mkimage-profiles`. Сломанная сборка на текущем стабильном бранче считается ошибкой и подлежит исправлению, если оно технически возможно на базе этого бранча.

- `lib/report.mk`
 - ожидает, что каждая подлежащая трассированию цель каждого `makefile` при сборке конфигурации образа содержит непустой `recipe` — хотя бы `"@:"` — т.к. зависит от запуска `$(SHELL)`
 - характерный признак пропуска — разрыв графа (`report-targets.png`)
- `pkg.in/lists/Makefile`
 - ожидает, что названия пакаджлистов указываются в переменных вида `*_LISTS`, и копирует в генерируемый профиль только их
 - если задать имя файла пакаджлиста непосредственно в `Makefile` субпрофиля, он не будет скопирован
 - характерное сообщение об ошибке:
`E: Couldn't find package`
- `features.in/stage2/stage1/scripts.d/02-kdefault`
- `features.in/stage2/stage1/scripts.d/03-test-kernel`
- `sub.in/stage1/Makefile`
 - ожидают, что в `stage1` попадёт строго одно ядро сообразно явному указанию в `STAGE1_KFLAVOUR` (либо последнее указанное в `KFLAVOURS`)
 - если добавить какой-либо `kernel-image` в `STAGE1_PACKAGES*`, результат может быть неожиданным
 - вероятная ошибка: незагрузка полученного `squashfs`
- `features.in/install2/install2/stage2cfg.mk`
- `features.in/live/live/stage2cfg.mk`

- `features.in/rescue/rescue/stage2cfg.mk`
- `features.in/syslinux/cfg.in/15live.cfg`
- `features.in/syslinux/cfg.in/20install2.cfg`
- `features.in/syslinux/cfg.in/80rescue.cfg`
- `features.in/syslinux/scripts.d/20-propagator-ramdisk`
 - ожидают, что названия squashfs-образов второй стадии инсталлятора, `livedcd` и спасательной системы соответственно `altinst`, `live` и `rescue`
- `image.in/Makefile`
 - ожидает, что конфигурация будет в `distcfg.mk` (см. тж. `lib/profile.mk`), а лог сборки — в `build.log` (см. тж. `lib/log.mk`); альтернативой было бы пробрасывание переменных с полным путём ради единственного места

Глава 12. Ловля плюх

При отладке сборки конфигурации или самого дистрибутива могут оказаться полезными следующие средства:

- `build/distcfg.mk`
 - формируется автоматически в процессе построения конфигурации;
 - содержит трассировочную информацию (откуда что взялось);
 - этот файл применяется как авторитетный конфигурационный
- `build/build.log`
 - подробность зависит от значения переменной `DEBUG`, которую можно передать при запуске `make` (см. `params.txt`);
 - содержит коммит, из которого происходит сборка, и признак "грязности" рабочего каталога при наличии модификаций после этого коммита;
 - содержит список конфигурационных переменных и их конечных значений, созданный на основании `distcfg.mk` (см. тж. `build/vars.mk`)
- `REPORT=1` включает генерацию дополнительного вывода:
 - `build/reports/targets.png` — граф зависимостей между целями
 - `build/reports/scripts.log` — порядок запуска скриптовых хуков
 - `build/reports/cleanlog.log` — более пригодный для `diff(1)` журнал сборки

Общая информация по отладке сборки профилей `mkimage` доступна на вики: <http://www.altlinux.org/Mkimage/debug>

Глава 13. Оформление кода

- постарайтесь не вносить без обсуждения разноречивых стилей, если есть предметные пожелания по коррекции текущего — пишите в `devel-distro@` или мне (`mike@`), обсудим;
- перед тем, как делать существенные переработки уже имеющегося кода — опять же опишите проблему, идею и предполагаемый результат, порой могут выясниться непредвиденные последствия;
- документируйте на русском (README) или английском (README.en) языке то, что написали или изменили.

рекомендации

- трезво относитесь ко входным данным и не пренебрегайте кавычками: название дистрибутива с пробелом или получение текста ошибки вместо ожидаемого вывода команды могут привести к сложнодиагностируемым ошибкам; вместе с тем в ряде случаев, где требуется пословная обработка значений переменных или раскрытие шаблонов shell (`*?[]`) — кавычки могут оказаться лишними;
- пользуйтесь `if [...]; then ...; fi` вместо `[...] && ...`: это кажется более громоздким, но текст оказывается более читаемым и в т.ч. расширяемым, поскольку между `then/else/fi` можно спокойно добавлять строки; с `&&` проще забыть `{ ... }` или оставить ненулевой код возврата при неудаче теста, когда он считается несущественным;
- предпочтительно применение `$()` вместо ``` (особенно при вложенности);
- постарайтесь не вылазить за 80 колонок.

ССЫЛКИ

- <https://lists.altlinux.org/mailman/listinfo/devel-distro> (подписка по приглашению)

Глава 14. Сборка образов VM

ВНИМАНИЕ: заключительная операция создания образа жёсткого диска из архива с содержимым корневой файловой системы требует доступа к `sudo` и разрешения на выполнение скрипта `bin/tar2vm` в корневом каталоге метапрофиля при установке `mkimage-profiles` из пакета (это в планах исправить, но подход к `libguestfs` пока успехом не увенчался).

Соответствующий фрагмент конфигурации `sudo(8)` может выглядеть как:

```
mike ALL=NOPASSWD: /usr/share/mkimage-profiles/bin/tar2vm
```

При работе с локальной копией `mkimage-profiles.git` следует иметь в виду, что предоставлять недоверенному пользователю право выполнять от имени `root` доступный ему по записи скрипт равнозначно предоставлению полных привилегий `root`.

Для работы с более специфичными форматами, чем `raw` ("буквальный" образ диска), потребуется утилита `qemu-img` из одноименного пакета; см. тж. вывод команды `make help/vm`

Также потребуется пакет `multipath-tools (/sbin/kpartx)`.

Пример сборки и запуска VM:

```
$ make R00TPW=reallysecret1 vm/bare.img && kvm -hda ~/out/bare.img
```

Глава 15. QEMU

Для сборки на "неродной" архитектуре с применением трансляции посредством QEMU установите пакет `livesd-qemu-arch` и выполните команду `register-qemu-arm` от имени `root` (также предоставляется `register-qemu-ppc`, но как минимум при сборке под `ppc32` на `x86_64` известны проблемы эмуляции).

Пример запуска:

```
make ARCH=arm APTCONF=/etc/apt/apt.conf.sisyphus.arm ve/bare.tar
```