mkimage-profiles

Michael Shigorin <mike@altlinux.org>,
Anton Midyukov <antohami@altlinux.org>

mkimage-profiles

Michael Shigorin <mike@altlinux.org>, Anton Midyukov <antohami@altlinux.org>

Содержание

Введение vi
I. Основы 1
1. Welcome to m-p! 2
2. Переменные make 5
2.1. Переменная BRANCH 11
3. Фичи 12
4. Списки пакетов 13
4.1. Суффикс @KFLAVOUR в списках пакетов 14
II. Каталоги 15
5. conf.d 16
6. image.in 18
7. features.in 19
7.1. features.in/00example 19
7.2. features.in/alternatives 20
7.3. features.in/apt-conf 21
7.4. features.in/arm-rpi4 21
7.5. features.in/blacklist-pkgs 21
7.6. features.in/bootloader 21
7.7. features.in/branding 22
7.8. features.in/browser 22
7.9. features.in/build-distro 23
7.10. features.in/build-ve 23
7.11. features.in/build-vm 23
7.12. features.in/cleanup 23
7.13. features.in/control 24
7.14. features.in/datetime 24
7.15. features.in/deflogin 24
7.16. features.in/dev 25
7.17. features.in/docs 26
7.18. features.in/dos 26
7.19. features.in/drm 26
7.20. features.in/e2k 26
7.21. features.in/efi 27
7.22. features.in/firmware 27
7.23. features.in/fonts 27
7.24. features.in/gitlab-runner 27
7.25. features.in/grub 28
7.26. features.in/hdt 29
7.27. features.in/homeros 29
7.28. features.in/im 29
7.29. features.in/initrd-bootchain 29
7.30. features.in/initrd-propagator 30
7.31. features.in/init 30
7.32. features.in/install2 30
7.33. features.in/isohybrid 31
7.34. features.in/kernel 31
7.35. features.in/l10n 31

7.36. features.in/ldm 31
7.37. features.in/live-install 31
7.38. features.in/live 32
7.39. features.in/lowmem 33
7.40. features.in/ltsp 33
7.41. features.in/luks 33
7.42. features.in/lxc-guest 33
7.43. features.in/lxc 33
7.44. features.in/mediacheck 33
7.45. features.in/memtest 34
7.46. features.in/metadata 34
7.47. features.in/net-dns 34
7.48. features.in/net-eth 34
7.49. features.in/net 34
7.50. features.in/net-ssh 35
7.51. features.in/net-usershares 35
7.52. features.in/no-sleep 35
7.53. features.in/ntp 35
7.54. features.in/oem 35
7.55. features.in/office 35
7.56. features.in/pack 36
7.57. features.in/pid1 36
7.58. features.in/pkgpriorities 36
7.59. features.in/plymouth 37
7.60. features.in/power 37
7.61. features.in/relname 37
7.62. features.in/repo 37
7.63. features.in/rescue 38
7.64. features.in/server 38
7.65. features.in/services 38
7.66. features.in/sound 39
7.67. features.in/speech 39
7.68. features.in/stage2 39
7.69. features.in/syslinux 39
7.70. features.in/tty 40
7.71. features.in/uboot 40
7.72. features.in/uuid-iso 41
7.73. features.in/vagrant 41
7.74. features.in/vmguest 41
7.75. features.in/volumes 41
7.76. features.in/wireless 41
7.77. features.in/x11-autologin 41
7.78. features.in/x11-autostart 42
7.79. features.in/x11 42
7.80. features.in/x11-vnc 43
7.81. features.in/xdg-user-dirs 43
8. sub.in 44
8.1. sub.in/main 44
8.2. sub.in/stage1 45

8.3. sub.in/stage2 45
9. pkg.in 47
9.1. pkg.in/lists 47
9.2. pkg.in/lists/tagged 47
10. lib 49
III. Приложения 50
11. Предположения 51
12. Ловля блох 53
13. Оформление кода 54
14. Сборка образов VM 55
15. QEMU 56
16. Архитектурно-зависимые фрагменты 57
16.1. Makefile 57
16.2. скрипты 57
16.3. списки пакетов, профили групп 57
16.4. загрузчики 58
17. Метапакеты 59
17.1. списки пакетов, профили групп 59

Введение

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

Цели

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

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/gears/m/mkimage-profiles.git
cd mkimage-profiles
make rescue.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 при сборке может быть доступен только на чтение
- 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
  - 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):
  - *внимание*: не надо создавать фичу, если достаточно списка пакетов!
  - 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.:

- https://altlinux.org/mkimage-profiles: обзорная документация, в т.ч. howto
- doc/:
  - params.txt: переменные, указываемые при запуске сборки
  - pkglists.txt: формирование состава образа
  - features.txt: обзор подключаемых особенностей

Примечание: пути в документации задаются от каталога верхнего уровня, если не указаны как относительные в явном виде (./) или по смыслу.

Удачи; что не так — пишите.

Michael Shigorin <mike@altlinux.org [mailto:mike@altlinux.org]>, Anton Midyukov <antohami@altlinux.org [mailto:antohami@altlinux.org]>

Глава 2. Переменные make

Переменные могут быть заданы, как в команде сборки в качестве аргументов, так и в файле настроек $HOME/.mkimage/profiles.mk. При запуске на сборку принимается ряд переменных (см. тж. profiles.mk.sample):

- APTCONF
  - задаёт путь к требуемому apt.conf
  - значение: пусто (по умолчанию системный) либо строка
  - см. image.in/Makefile, doc/apt.conf.sample
- ARCH
  - задаёт целевую архитектуру образов
  - значение: пусто (по умолчанию авто) либо строка
  - см. lib/build.mk
- ARCHES
  - задаёт набор целевых архитектур при параметрическом задании APTCONF
  - значение: пусто (по умолчанию авто) либо список через пробел
  - см. Makefile, doc/profiles.mk.sample
- AUTOCLEAN
  - включает уборку (distclean) после успешной сборки образа
  - значение: пусто (по умолчанию нет) либо любая строка
  - см. lib/build.mk
- BELL
  - подаёт сигнал после завершения сборки
  - значение: пусто (по умолчанию нет) либо любая строка
  - см. lib/build.mk
- BRANCH
  - указывает для какого бранча производится сборка
  - собирает вместо регулярок стартеркиты

  - значение:
    - не определено - пытается определиться автоматически
    - пусто - присваивается значение sisyphus
    - имя бранча (sisyphus, p10, p9)
  - см. doc/branch.txt main.mk, conf.d/mixin.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: $(BUILDDIR)/build.log (по умолчанию) либо строка
  - см. lib/log.mk
- CHECK
  - включает режим проверки сборки конфигурации (без сборки образа)
  - значение:
  - пусто (по умолчанию) - проверка не осуществляется;
  - 0 - прверяется только конфигурация, списки пакетов не проверяются;
  - другое значение - полная проверка.
  - 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: пусто, 0, 1, 2; 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
- DISTRO_VERSION
  - задаёт версию дистрибутива, если применимо
  - значение: пусто (по умолчанию) либо любая строка
  - см. image.in/Makefile
- GPG_KEY
  - указывает путь до экспортированного секретного ключа gnugpg,
  - чтобы скопировать его в чрут в /.in/ под именем gpgkey
  - см. image.in/Makefile
- 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 умолчаниям)
- IMAGEDIR
  - указывает путь для сохранения собранного образа
  - значение: равно $HOME/out, если существует, иначе $(BUILDDIR)/out
  - (по умолчанию), либо другой путь
  - см. lib/profile.mk
- ISOHYBRID
  - включает создание гибриднго ISO-образа
  - значение: пусто (по умолчанию) либо любая строка

  - см. features.in/pack/config.mk, features.in/isohybrid/config.mk
- ISO_LEVEL
  - задаёт опцию -iso-level для xorriso (mkisofs) в mkimage
  - значение: 2 (по умолчанию) или 1, 3, 4
  - см. man mkisofs
- LOGDIR
  - указывает путь для сохранения логов сборки
  - значение: равно $(IMAGEDIR) (по умолчанию), либо другой путь
  - NB: в силу специфики обработки передаётся только явно
  - см. lib/build.mk, lib/profile.mk
- MKIMAGE_PREFIX
  - указывает путь до mkimage. Если параметр не указан, то используется системный mkimage
- NICE
  - понижает нагрузку системы сборочной задачей
  - значение: пусто (по умолчанию) либо любая строка
  - см. lib/build.mk
- NO_SYMLINK
  - не создавать символические ссылки на собранный образ
  - значение: пусто (по умолчанию) либо любая строка
  - см. image.in/Makefile
- NOCHECK
  - отключает проверку списков пакетов перед сборкой
  - значение: пусто (по умолчанию) либо любая строка
  - см. pkg.in/lists/Makefile
- QUIET
  - отключает поясняющие сообщения при сборке (например, под cron)
  - значение: пусто (по умолчанию) либо любая строка

  - см. Makefile, lib/build.mk, lib/clean.mk, lib/profile.mk
- PACK_SQUASHFS_PROCESSORS
  - задаёт количество процессоров (ядер или потоков), которые будут задействованы для упаковки сквоша
  - значение:
    - пусто (по умолчанию) - будут задействованы все доступные процессоры
    - любое число, не превышающее число доступных процессоров
  - см. mkimage/config-squash.mk
- REPO
  - задаёт источник репозитория для включения внутри образа
  - значение:
    - префикс имени файла из /etc/apt/sources.list.d/
    - протокол ("http" либо "rsync"), "/" и префикс имени файла
  - см. features.in/repo/image-scripts.d/90-enable-repo
- REPORT
  - запрашивает создание отчётов о собранном образе
  - требует включения DEBUG и отключения CHECK
  - значение:
  - пусто (по умолчанию) - создание отчёта выключено
  - 2 - создать архив из каталога отчёта
  - любая другое непустое значение - создать отчёт в виде каталога
  - 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/deflogin/rootfs/image-scripts.d/50-root

- 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
- STDOUT
  - выводить сообщения при включенном DEBUG одновременно в лог и на экран
  - значение:
    - 1 - включить вывод на экран, если включен DEBUG
  - см. lib/log.mk

- USE_QEMU
  - использовать qemu, если архитектура не совпадает
  - значение:
    - 1 (по умолчанию), для отключения используйте любое другое значение
  - см. lib/build.mk
- VM_SAVE_TARBALL
  - указывает, что нужно сохранить промежуточный тарбол, из которого
  - создаётся образ виртуальной машины, в заданном формате
  - значения: tar tar.gz tar.xz
  - см. features.in/build-vm/lib/90-build-vm.mk
- 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/tar2fs

пример

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

2.1. Переменная BRANCH

Переменная make, указывающая для какого бранча производится сборка. Если не задана, определяется автоматически. Если переменная имеет пустое значение, назначается sisyphus. Для того, чтобы при указании этой переменной сборка осуществлялась для целевого бранча, требуется:

- прописать в ~/.mkimage/profiles.mk:

```
APTCONF = ~/apt/apt.conf.$(BRANCH).$(ARCH)
```

- создать целевые конфиги apt по указанным выше путям.

Помимо этого переменная BRANCH, если определена, заменяет в имени собираемой цели слово "regular" на "alt-$BRANCH". Таким образом достигается сборка стартеркитов из профиля регулярок под заданный бранч.

Также эту переменную можно использовать в профилях других целей для обеспечения поддержки целевого бранча.

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;
- в виде патчей в image-patches.d/ различных стадий;
- пакеты installer-feature-* (тж. https://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-*); также возможно добиться большей степени интеграции по данным (например, язык и LiveCD).

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):

- main: пакетная база для установки
  - sub.in/main/Makefile, features.in/*/main/lib/
  - THE_LISTS, BASE_LISTS, MAIN_LISTS, COMMON_LISTS
  - THE_GROUPS, MAIN_GROUPS
  - THE_PACKAGES, BASE_PACKAGES, MAIN_PACKAGES, SYSTEM_PACKAGES, COMMON_PACKAGES
  - THE_PACKAGES_REGEXP, BASE_PACKAGES_REGEXP, MAIN_PACKAGES_REGEXP
  - THE_KMODULES, BASE_KMODULES, MAIN_KMODULES, BASE_KMODULES_REGEXP
    - KFLAVOURS
- stage2: общая часть install2, live, rescue
  - sub.in/stage2/Makefile, features.in/*/stage2/lib/
  - SYSTEM_PACKAGES, STAGE2_PACKAGES
  - STAGE1_KMODULES, STAGE1_KMODULES_REGEXP, STAGE2_KMODULES, STAGE2_KMODULES_REGEXP
    - STAGE1_KFLAVOURS или KFLAVOURS для grub, для других загрузчиков последний из STAGE1_KFLAVOURS или KFLAVOURS
- install2: компактная "живая" система, содержащая только инсталятор
  - см. stage2
    - features.in/install2/install2/stage2cfg.mk, features.in/*/install2/lib/
    - INSTALL2_PACKAGES
- live: пользовательский LiveCD (может содержать также инсталятор)
  - см. stage2
  - features.in/live/live/stage2cfg.mk, features.in/*/live/lib/
  - THE_LISTS, LIVE_LISTS, COMMON_LISTS
  - THE_GROUPS, LIVE_GROUPS

    - THE_PACKAGES, LIVE_PACKAGES, COMMON_PACKAGES
    - THE_PACKAGES_REGEXP, LIVE_PACKAGES_REGEXP
    - THE_KMODULES, LIVE_KMODULES
- rescue: спасательный LiveCD
    - см. stage2
    - features.in/rescue/rescue/stage2cfg.mk
    - RESCUE_PACKAGES, COMMON_PACKAGES
    - RESCUE_LISTS, COMMON_LISTS
- stage1: ядро и загрузчик второй стадии
    - sub.in/stage1/Makefile, features.in/*/stage1/lib/
    - STAGE1_PACKAGES, SYSTEM_PACKAGES
    - STAGE1_PACKAGES_REGEXP
    - STAGE1_KMODULES_REGEXP
        - STAGE1_KFLAVOURS или KFLAVOURS для grub, для других загрузчиков последний из STAGE1_KFLAVOURS или KFLAVOURS

4.1. Суффикс @KFLAVOUR в списках пакетов

В списках пакетов можно указать суффикс @KFLAVOUR. Вместо этого суффикса будет осуществлена подстановка значений из переменной KFLAVOURS. С каждой новой подстановкой будет добавляться ещё одна строка. Если значением два, то и строки будет две.

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.

Попросту говоря, copy-paste — тревожный признак.

Вместо него нередко может помочь выделение кусочков конфигурации в пределах включаемого файла в цели mixin/*, которые не являются самостоятельными или даже промежуточными, но включают полезные группы настроек, нужных в различных образах, не наследующих друг другу — посмотрите существующие примеры использования.

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

- для пользовательского окружения (live, main) предназначены THE_PACKAGES, THE_LISTS, THE_GROUPS, THE_PACKAGES_REGEXP
- для "обычного общего" (live, main, rescue) есть COMMON_PACKAGES и COMMON_LISTS (NB: тоже попадают в базовую установку, ve/ и vm/ сборки)
- 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 инсталяторе, и в готовой системе (но не в rescue)
- для направленного действия служат:
  - STAGE1_PACKAGES, STAGE1_PACKAGES_REGEXP (первая стадия загрузки)
  - STAGE2_PACKAGES (инсталятор и спасательная/"живая" система)
  - INSTALL2_PACKAGES (инсталятор)
  - BASE_PACKAGES, BASE_LISTS, BASE_PACKAGES_REGEXP (базовая система)
  - MAIN_PACKAGES, MAIN_LISTS, MAIN_PACKAGES_REGEXP (дополнительные пакеты)
  - LIVE_PACKAGES, LIVE_LISTS, LIVE_PACKAGES_REGEXP ("живая" система)

- аналогично по kernel-modules-*:
  - 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 image.in/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!).

7.1. features.in/00example

Этот каталог содержит "заготовку" фичи в качестве примера и должен дать представление о том, какой код *может* быть включён в настоящую фичу: статические файлы, два makefile для создания конфигурации и генерирования части профиля, а также шелл-скрипт для такого генерирования.

Вовсе не требуется втягивать всё в свою фичу: лучше постараться сделать её минимальной, самодостаточной и полезной в качестве "кирпичика".

Единственной обязательной частью фичи является файл config.mk, который включается в distro.mk верхнего уровня и предоставляет цель вида use/*, специфичную для данной фичи, а также добавляет её в переменную FEATURES. Если название фичи не упоминается в списке, содержащемся в этой переменной, то она не задействуется при построении профиля, а только при сборке конфигурации.

Для наиболее ходовых целей use/**, особенно если их много, можно создавать цели-алиасы +** (например, +power). Просьба относиться вдумчиво, т.к. в дальнейшем предполагается визуализировать такие цели в UI конфигурирования образа.

Остальное содержимое является дополнительным и используется в таком порядке (см. features.in/Makefile):

- сперва в $(BUILDDIR)/image/ копируются все подкаталоги, соответствующие итоговым именам субпрофилей, запрошенных для профиля образа; при этом они сливаются с деревом, которое уже сформировано субпрофилями (sub.in/**) и уже скопированными фичами; если какие-либо файлы перекрылись по именам, rsync должен оставить резервные копии (**~), которые должны просигнализировать о беспорядке;

- запускается generate.sh, если существует и исполнимый;

- применяется generate.mk, если существует и непустой.

Например, если используются субпрофили stage1, stage2/install2 и main, можно решить собрать специфические для фичи скрипты инсталятора в install2/image-scripts.d/ (или необходимые для операций с пакетной базой — в main/scripts.d/, смотря чего надо добиться; загляните также в документацию mkimage).

А если требуются нетривиальные действия по конфигурированию (как при сборке syslinux.cfg из кусочков, в зависимости от того, что из запрошенных модулей оказалось в наличии) — то их можно произвести из generate.sh и generate.mk.

Пожалуйста, присылайте отзывы о (бес)полезности кода в этом каталоге mike@altlinux.org [mailto:mike@altlinux.org].

7.2. features.in/alternatives

Данная фича позволяет задавать альтернативы [1]. Например, так:

```
@$(call add,ALTERNATIVES,/usr/bin/xvt:/usr/bin/xterm)
```

Также в самом конфиге могут быть уже преопределены цели для установки альтернатив. Например, use/alternatives/xvt/% позволяет задавать произвольные альтернативы для xvt: use/alternatives/xvt/xterm, use/

alternatives/xvt/mate-terminal и т.д. При этом альтернатива должна быть доступна.

1. https://www.altlinux.org/Alternatives

7.3. features.in/apt-conf

Данная фича определяет то, какая конфигурация apt попадёт в образ.

7.4. features.in/arm-rpi4

Настраивает систему для Raspberry Pi 4.

Данная фича добавляет initrd фичу kickstart с конфигом для расширения корневого раздела до максимума.

7.5. features.in/blacklist-pkgs

Чёрный список пакетов BLACKLIST_PKGS, которых не должно быть ни в одной из стадий сборки. Проверка осуществляется в самом конце сборки iso образа, проверяются списки в .disk/pkglist iso образа.

Переменная BLACKLIST_PKGS содержит имена или части имён пакетов без поддержки масок.

7.6. features.in/bootloader

Добавление установки загрузчика основной системы, затребованного посредством указания "grub" или "uboot" в BASE_BOOTLOADER.

Модуль alterator-grub добавляется в устанавливаемую систему (он НЕ должен требоваться пакету installer-distro-**) и требует пакет выбранного загрузчика. Так как для uboot такого модуля нет и в тоже время uboot не используется в установочных дистрибутивах, то установка модуля alterator была ограничена целями distro/**, формирующими ISO-образы.

Обратите внимание: в процессе конфигурирования дистрибутива "переключение" загрузчика может происходить только в одну сторону — если выставлен grub, произведено переключение на lilo и затем произведена ещё одна попытка переключения на grub, то в конфигурации останется lilo как последняя "новая" цель с точки зрения make.

При необходимости всё-таки "пересилить" последнее изменение можно

```
@$(call set,BASE_BOOTLOADER,grub)
```

Реализация экспериментальная (нужно модуляризовать installer-steps).

7.7. features.in/branding

Эта фича врезается в makefile субпрофилей и обеспечивает добавление задающих внешний вид и сообщения дистрибутива пакетов; см. тж. https://www.altlinux.org/Branding

Реализация "двумерная" — отдельно задаётся BRANDING (см. пакеты branding-*-%version-%release.src.rpm), затем отдельно указывается, какие и куда помещать компоненты заданного брендинга.

Назначение и возможные значения (если требуются):

- STAGE1_BRANDING
  - относится к загрузке со сгенерированного образа (например, ISO)
  - bootloader bootsplash (при старте)
- STAGE2_BRANDING
  - общая часть для всех вариантов stage2
  - bootsplash (при выключении)
- INSTALL2_BRANDING
  - специфические пакеты брендирования инсталятора
  - notes slideshow
- THE_BRANDING
  - общий список для использования в установленной системе и LiveCD
  - alterator bootsplash graphics indexhtml notes slideshow

7.7.1. Подстановка в списках пакетов

В списках пакетов вместо прописывания конкретного брендинга нужно прописывать @BRANDING@, который при генерации профиля будет заменён на заданный брендинг.

7.8. features.in/browser

Эта фича обеспечивает наличие и конкретизацию выбора браузера. Разумеется, дополнительные варианты могут быть установлены явным или косвенным затребованием.

Следует понимать, что каждая из целей может быть использована лишь один раз, повторное упоминание будет проигнорировано make.

7.9. features.in/build-distro

Эта фича конфигурирует создание образа дистрибутива, включая работу с субпрофилями — которая сейчас нужна только дистрибутивным целям.

Дополняет финальную стадию сборки (lib/, scripts.d/) и тесно с ней связана.

При желании более полно воспользоваться доступными средствами фиксации метаданных обратите внимание на следующие переменные: META_SYSTEM_ID, META_PUBLISHER, META_PREPARER, META_APP_ID, META_VOL_ID, META_VOL_SET, META_BIBLIO, META_ABSTRACT; см. тж. genisoimagerc(5) из пакета genisoimage.

Для управления наименованием дистрибутива, отображаемым во время загрузки образа, можно воспользоваться переменной ISO_BOOT_DISTRO_NAME. Это позволяет указывать длинные имена, не ограниченные 32 символами. Если не задано, используется META_VOL_ID / META_VOL_SET / RELNAME.

При необходимости задать своё содержимое файла .disk/info, который также используется propagator, см. META_DISK_INFO.

7.10. features.in/build-ve

Эта фича конфигурирует создание образа виртуального окружения (VE), что используется для сборки шаблонов OpenVZ и ARM-чрутов для TWRP.

Дополняет финальную стадию сборки (lib/, image-scripts.d/) и тесно с ней связана.

7.11. features.in/build-vm

Эта фича конфигурирует создание образа виртуальной машины (VM) или тарбола rootfs для использования его на реальном компьютере. Дополняет финальную стадию сборки (lib/, image-scripts.d/). Для создания образа виртуальной машины требуется sudo(8) Для создания тарбола sudo не требуется. — см. тж. doc/vm.txt

7.12. features.in/cleanup

Эта фича вместо созидания занимается выкидыванием лишнего (например, части модулей инсталятора из установленной системы).

По возможности стоит работать над дизайном инфраструктуры и пакетной базой так, чтобы ставить-удалять приходилось как можно меньше. В идеале такой антифичи не должно быть вовсе :)

Для пакетов, которые следует удалять из установленной классическим инсталятором системы, но не из livecd, применяйте переменную CLEANUP_BASE_PACKAGES.

Для удаления пакетов только из livecd используйте переменную CLEANUP_LIVE_PACKAGES.

Внимание

также удаляет rpm, apt и базу по пакетам из livecd, если в него не был добавлен инсталятор!

Генрирует контрольными суммы всех бинарных файлов во всех RPM-пакетах. Необходимо передать переменную GPG_KEY.

Фича добавлет в корень iso-образа каталог containers. Каталог наполнить нужно самому.

7.13. features.in/control

Эта фича предоставляет интерфейс для конфигурирования дистрибутивных значений по умолчанию control(8).

См. тж.:

https://www.altlinux.org/Control

7.14. features.in/datetime

Данная фича предназначена для настройки часового пояса и переключения хранения времени в BIOS между UTC (по Гринвичу) и местным временем.

- TIME_UTC
  - Переключает хранение времени в BIOS с UTC (по Гринвичу) на местное время
  - значение: 0 - местное; 1 - UTC
- TIME_ZONE
  - Задаёт часовой пояс
  - значение: формат регион/город из каталога /usr/share/zoneinfo/регион/город

7.15. features.in/deflogin

Эта фича конфигурирует root login и пользователей по умолчанию.

Если ROOTPW не задан, то подходящий пароль не существует. При необходимости задать пустой пароль root (например, на LiveCD) выставьте переменную ROOTPW_EMPTY в значение "1".

Внимание

применяйте разумно, т.к. крайне легко создать и оставить дыру в безопасности!

Пользователи добавляются через переменную USERS через пробел в формате:

```
login:passwd:admin:sudo
```

Например:

```
@$(call add,USERS,fedya:123:1:1 masha:321::)
```

Будут созданы два пользователя: fedya с паролем 123', с правами администратора и настроенным sudo, и пользователь masha с паролем *321*, без прав администратора и без sudo.

Также можно определить группы, в которые будут добавляться пользователи через переменную GROUPS.

В версии mkimage-profiles 1.4.4 появилась возможность создать пользователя с произвольными uid, gid, домашним каталогом, интерпретатором shell и т.д. Используйте для этого следующую конструкцию:

```
@$(call set,SPEC_USER,имя_пользователя:группа:uid:gid:home_dir:shell)
```

Например:

```
@$(call set,SPEC_USER,user:user:500:500:/home/user:/bin/bash)
```

При этом нужно иметь в виду, что будет создана соответствующая группа с соответствующим gid (нужно быть уверенным, что одноимённая группа не существует), а пользователь будет добавлен в неё. Пользователь будет создан без пароля. Для установки пароля при первом запуске, смотрите фичу oem.

7.16. features.in/dev

Эта фича служит для создания образов, предназначающихся для разработки. В первую очередь обеспечивается развёртывание hasher и mkimage.

Реализованы поддержка LiveCD, VM, VE и добавление группы в инсталятор.

Обратите внимание: use/dev/repo достаточно серьёзно изменяет поведение субпрофиля main, оставляя из всего обычного множества обрабатываемых

переменных только MAIN_PACKAGES, MAIN_PACKAGES_REGEXP и MAIN_LISTS во избежание дублирования не требующихся для сборки минимальных образов пакетов.

7.17. features.in/docs

Эта фича добавляет в образ распакованную документацию дистрибутива, а именно вводную страничку (входит в пакет branding-**-indexhtml), и/или специфичное для данного продукта руководство из пакета docs-** (вариант задаётся отдельно выставлением переменной DOCS), и/или тексты лицензионных соглашений.

Обратите внимание, что для indexhtml создаётся переброска с учётом языка (при наличии index-LL.html), поэтому ожидается задействование фичи l10n с соответствующим указанием языка по умолчанию.

Примечание

предполагается применение при формировании ISO-образов, другие случаи наверняка потребуют доработки.

7.18. features.in/dos

Фича добавляет создание FreeDOS "live floppy" в stage1.

Текущее состояние — загружается минимальная система. Для практического применения (в первую очередь это перешивка firmware) требуется сделать подключение CD и/или USB Flash с тем, чтобы класть туда прошивки.

Спасибо за идею и местами реализацию: Александру Бандуре, Сергею Горенко, Николаю Гречуху и Алексею Фролову.

7.19. features.in/drm

Фича drm решает задачу создания общей точки входа для добавления drm-модулей ядра для разных списков пакетов. Потребность выделения в отдельную фичу возникла с одной стороны в связи с необходимостью сделать переключатель между свободным и проприетарным драйвером NVIDIA, с другой из-за необходимости добавлять только drm-модули ядра в таких целях, как use/stage2/kms и use/plymouth.

7.20. features.in/e2k

Эта фича содержит необходимое для поддержки систем архитектуры "Эльбрус".

7.21. features.in/efi

Фича добавляет в образы необходимое для поддержки EFI/UEFI.

Конфигурируется заданием загрузчика (EFI_BOOTLOADER) и файла сертификата (EFI_CERT) при помощи целей; пример использования доступен в conf.d/regular.mk

См. тж.:

- https://www.altlinux.org/UEFI
- https://www.rodsbooks.com/efi-bootloaders/
- https://bugzilla.altlinux.org/showdependencytree.cgi?id=27804

7.22. features.in/firmware

Эта фича добавляет комплекты различного firmware в инсталятор, устанавливаемую систему и т.п.

Следует учитывать, что объём добавленного может быть довольно существенным — в десктопе вряд ли нужен микрокод для FC HBA или SCSI RAID, который может быть критичен для полезного серверного или спасательного дистрибутива (см. #18047).

7.23. features.in/fonts

Эта фича позволяет системно конфигурировать файлы конфигурации подсистемы конфигурирования шрифтов fontconfig (sic!), заодно предоставляя прошедшие обкатку в дистрибутивах варианты предварительно заданной конфигурации для удобства.

7.24. features.in/gitlab-runner

This feature installs gitlab-runner according official guide [1]

The following envs can be altered:

GL_USER - define default gitlab-runner username (*gitlab-runner* by default)
GL_SSH_KEY - ssh pubkey added to authorized_keys of GL_USER

Примечание

this feature depends on network enablement in hasher (see [2] for details) and mkimage [3]

1. https://docs.gitlab.com/runner/install/linux-manually.html

2. https://bugzilla.altlinux.org/34596

3. https://git.altlinux.org/gears/m/mkimage.git?p=mkimage.git;a=commit;h=242549e3a01306e4539757e2129ea39f5a199b90

7.25. features.in/grub

Добавление поддержки grub; требуется для инсталяторов, live/rescue; реализуется в рамках stage1.

Самостоятельное творческое использование на данный момент подразумевает изучение кусочков конфигурации, которые уже существуют.

Цели config.mk:

- use/grub/ui/% — конфигурирование интерфейса (см. cfg.in/01gfxterm.cfg); при использовании автоматически добавляют grub в FEATURES;
- use/grub/timeout/% — задание таймаута автозагрузки (в секундах);
- use/grub/%.cfg — подключение кусочков конфигурации.

Переменные generate.mk:

- BOOTARGS — дополнительные аргументы загрузчику;
- BOOTLOADER — isolinux (реализовано с оглядкой на grub/grub4);
- GRUB_GFXMODE — разрешения экрана графического интерфейса (например, 1920x1080,auto). По дефолту auto.
- GRUB_UI — тип интерфейса (если указан gfxboot, то графический, иначе текстовый);
- GRUB_CFG — дополнительные кусочки конфигурации (например, live_rw);
- GRUBTHEME — имя темы grub, если не задана, то получает значение BRANDING за вычетом традиционной приставки alt-);
- DISABLE_LANG_MENU — отключает меню выбора языка в grub, если задана.

Здесь производится первичная обработка конфигурационных данных, окончательно проверяемых и используемых уже в инструментальном чруте.

Обратите внимание: фрагменты, соответствующие именам субпрофилей, добавляются автоматически; это поведение при необходимости отключается выставлением переменной grub_DIRECT и тогда вместо use/grub/*.cfg следует применять прямое указание вида @$(call set,grub_CFG, …).

Установить дефолтный пункт: Для того, чтобы установить конкретный дефолтный пункт (пример для LiveCD без поддержки сессии):

```
@$(call set,GRUB_DEFAULT,live)
```

Именем дефолтного пункта является --id.

Запуск iso образа с неправильно работающей в grub графике (только EFI): На ESP-разделе образа можно отредактировать конфиг EFI/BOOT/grub.cfg, добавив в его начало:

```
GRUB_TERMINAL='console'
```

Если нужно включить последовательную консоль, пропишите в нём:

```
GRUB_TERMINAL='console serial'
```

```
GRUB_SERIAL_COMMAND='serial --unit=0 --speed=115200'
```

7.26. features.in/hdt

Добавление модуля hdt (Hardware Detection Tool) к syslinux; может быть востребовано для инсталяторов, live/rescue.

Фича не только требует фичу syslinux (и соответствующий пакет, а также pciids), но и тесно с ней интегрирована; в частности, фрагмент конфигурационного файла располагается не "здесь", а "там", поскольку сам hdt.c32 также входит в пакет syslinux.

7.27. features.in/homeros

Каталог содержит основную feature для создания адаптированного дистрибутива Homeros. Это промежуточный вариант, при помощи которого можно получить минимальный разговаривающий образ, но, возможно, помимо его дальнейшего естественного развития требуется ещё осмысление с точки зрения идей mkimage-profiles.

7.28. features.in/im

Эта фича добавляет средства настройки методов ввода (Input Methods).

На данный момент является экспериментальной, приветствуется помощь в тестировании/доработке/отладке.

7.29. features.in/initrd-bootchain

Осуществляется сборка initrd.img при помощи make-initrd с включенными фичами bootchain. Это альтернатива фичи initrd-propagator. Используется тот же список модулей ядра, что и при сборке с propagator, но в отличие от последнего модули добавляются в initrd.img самим make-initrd.

Документацию по использованию bootchain следует смотреть в пакетах make-initrd-bootchain-*:

```
/usr/share/make-initrd/features/bootchain-*/README.md
```

Конфиг находится в stage1/files/bootchain. Почти все переменные можно переопределить. Переменные в m-p по сравнению с конфигом bootchain имеют приставку *BOOTCHAIN_*. Дефолты заданы в config.mk

Переопределить можно так:

```
$(call set,BOOTCHAIN_OEM_WELCOME_TEXT,Welcome to my OS!)
```

7.30. features.in/initrd-propagator

Добавляется поддержка propagator. propagator обеспечивает первую стадию загрузчика. Ранее был неотъемлемой частью субпрофиля stage1. Был вынесен в фичу для обеспечения возожности собирать образы с использованием специально собранного initrd вместо него.

7.31. features.in/init

Эта фича определяет систему инициализации, которая будет использована в пользовательской среде (livecd, установленный дистрибутив, vm). Она не влияет на состав инсталятора и rescue-образа.

Обратите внимание: как и с use/bootloader/%, в силу особенностей make переключение в каждую позицию возможно лишь один раз, далее эта цель считается достигнутой и при последующих вызовах не отрабатывает.

См. тж.:

https://www.altlinux.org/Sysvinit

https://www.altlinux.org/Systemd

7.32. features.in/install2

Эта фича дополняет базовый "живой" образ второй стадии специфическими для инсталяционного образа настройками и скриптовыми хуками.

Рекомендуется подключать при помощи +installer, чтобы обеспечить включение типового набора связанных с инсталятором функций.

При добавлении скриптов в image-scripts.d/ следует позаботиться, чтобы в компактном livecd, которым является инсталятор, оказались нужные им утилиты (INSTALL2_PACKAGES). Перегружать его не следует, поскольку это прямо влияет на требования по минимальному размеру оперативной памяти для установки (если не задействован параметр загрузки ядра lowmem, обрабатываемый propagator).

При необходимости принудительно удалить что-либо из попавшего в образ инсталятора (вместе с "оптовым" пакетом либо по зависимостям, когда точно известно, что для данного применения они избыточны) можно воспользоваться переменной INSTALL2_CLEANUP_PACKAGES для указания списка пакетов на удаление без учёта зависимостей перед формированием squashfs и INSTALL2_CLEANUP_KDRIVERS для удаления излишних модулей ядра.

7.33. features.in/isohybrid

Эта фича обеспечивает формирование ISO-образа с добавлением липовой таблицы разделов с целью обеспечения возможности его загрузки как с CD/DVD, так и с USB-флэшки.

Можно указать в цепочке зависимостей дистрибутива явно с тем, чтобы гарантировать гибридный вид образа, либо запросить включение этой фичи при сборке конфигурации произвольного дистрибутива (ISOHYBRID=1, см. features.in/pack/config.mk).

7.34. features.in/kernel

Эта фича привносит код, имеющий смысл при добавлении в образ ядра, и задаёт начальный вариант такового.

Также занимается складированием наборов имён пакетов kernel-modules-* с тем, чтобы избавить релиз-менеджеров от необходимости учитывать полные списки и точные имена дополнительных модулей для поддержки, скажем, Ethernet.

7.35. features.in/l10n

Эта фича занимается поддержкой локализации (l10n).

7.36. features.in/ldm

Simple hook to run Linux Driver Management tools to configure hybrid graphics (aka Optimus/PRIME) for different DM’s.

Currently supported: + LightDM + GDM + SDDM

See https://github.com/solus-project/linux-driver-management

7.37. features.in/live-install

Эта фича дополняет live образ второй стадии специфическими для инсталяционного образа настройками и скриптовыми хуками.

В отличие от фичи install2 не собирается отдельный образ второй стадии altinst, а дополняется образ live пакетами инсталятора с целью уменьшить общий объём iso-образа.

Есть два варианта инсталятора:

1. установка из live при выборе цели use/live-install

2. установка из пакетов, как в altinst при выборе цели use/live-install/pkg

Первый вариант выглядит так:

1. Распаковывается образ live, как в livecd-install

2. Устанавливаются дополнительные пакеты BASE_PACKAGES и группы пакетов THE_GROUPS, как в install2.

В отличии от install2 в репозиторий main помещаются только те пакеты, которых нет в live образе. Этим и достигается уменьшение размера iso-образа.

Второй вариант не отличается от altinst.

По дефолту пакеты kernel-image и kernel-modules в режиме распаковки образа live удаляются. Чтобы это предотвратить задайте переменную MAIN_KERNEL_SAVE:

```
@$(call set,MAIN_KERNEL_SAVE,yes)
```

7.38. features.in/live

Эта фича дополняет базовый "живой" образ второй стадии специфическими для полноценного LiveCD настройками и скриптовыми хуками, а также создаёт файл index.html с домашней страницей (редиректором) в корне образа.

Графический вариант безусловно требует x11-autologin, при появлении необходимости обойтись без него можно временно продублировать содержимое цели и сообщить о таком случае.

Дополнительно обрабатываемые переменные:

- LIVE_REPO
  - позволяет выбрать и включить зеркало репозитория
  - значение: http/alt (по умолчанию) либо протокол/зеркало
  - см. тж. /etc/apt/sources.list.d/, проверьте наличие нужного!
- LIVE_CLEANUP_KDRIVERS
  - перечисляет префиксы каталогов драйвров ядра для удаления

  - значение: пусто (по умолчанию) или список через пробел
  - см. config.mk
- LIVE_RUNAPP_BINARY
  - указывает имя программы для запуска в режиме киоска
  - значение: пусто (по умолчанию) или путь (из $PATH или полный)
  - см. тж. пакеты livecd-0ad, livecd-fgfs и сам livecd-runapp

7.39. features.in/lowmem

Эта фича дополняет зачистку "живой" стадии инсталятора с тем, чтобы уменьшить её размер и требования к памяти.

7.40. features.in/ltsp

Эта фича обеспечивает добавление функций терминального сервера:

- загрузку бездисковых тонких клиентов по сети;
- предоставление им доступа к серверу приложений.

На данный момент является экспериментальной.

7.41. features.in/luks

Эта секретная фича добавляет в инсталяторы поддержку шифрования файловых систем с помощью LUKS при их создании.

7.42. features.in/lxc-guest

Adds systemd generators needed to run lxd container.

7.43. features.in/lxc

Эта фича предназначена для создания контейнеров LXC и LXD.

7.44. features.in/mediacheck

Эта фича конфигурирует внедрение контрольной суммы в образ инсталятора после его сборки с целью проверки целостности на ранней стадии установки.

NB: прототип, для реального использования надо сделать микрообраз на основе stage2.

7.45. features.in/memtest

Добавление memtest86+ в загрузку с образа и в устанавливаемую пакетную базу; востребовано для инсталяторов, live/rescue. Интегрируется с syslinux.

7.46. features.in/metadata

Эта фича занимается метаданными в составе образов — в первую очередь инсталяционных и пригодных к установке "живых".

Обязательные к установке по умолчанию пакеты задаются переменными SYSTEM_PACKAGES, COMMON_PACKAGES, BASE_PACKAGES, BASE_LISTS, THE_PACKAGES, THE_LISTS и передаются инсталятору посредством Metadata/pkg-groups.tar (файл .base).

См. тж. фичу main.

7.47. features.in/net-dns

Эта фича позволяет сконфигурировать публично доступный рекурсивный DNS-сервер для условий, когда локальный неизвестен заранее или попросту отсутствует; следует понимать, что это в некотором роде утечка данных, т.е. риск безопасности.

Также возможно указать свои NAMESERVERS через пробел у себя в фиче или конфигурации дистрибутива, которая задействует use/net-dns.

7.48. features.in/net-eth

Эта фича позволяет задать конфигурацию Ethernet-интерфейсов.

udev-rule-generator-net штатно добавляется для обеспечения предсказуемых имён вида eth0/eth1/... в силу того, что их сломали в апстримном systemd-udevd; для генерируемых "на лету" следует применять конфигурирование времени загрузки (NM, connman либо пакет livecd-net-eth).

7.49. features.in/net

Эта фича конфигурирует базовую поддержку сети, включая нужную подсистему (etcnet, NetworkManager поверх etcnet или connman).

Используйте TARGET_HOSTNAME для определения имени узла (файлы /etc/sysconfig/network и /etc/hostname).

7.50. features.in/net-ssh

Эта фича предназначена для добавления в образ поддержки SSH: добавляется клиент и конфигурируется сервер (требуется задание пути к существующему публичному ключу посредством переменной SSH_KEY).

7.51. features.in/net-usershares

Эта фича выполняет предварительное конфигурирование системы для работы плагинов файл-менеджеров, реализующих взаимодействие с Samba-сервером для динамического создания разделяемых файловых ресурсов ("пользовательских шар"). Без добавления соответствующего файл-менеджера и нужного плагина смысла не имеет.

7.52. features.in/no-sleep

Эта фича отключает спящий и ждущий режимы, а также гибернацию. Нужна для одноплатных компьютеров вроде Raspberry Pi, не поддерживающих их.

7.53. features.in/ntp

Эта фича конфигурирует службу NTP в качестве клиента с целью предоставления точного времени в составе LiveCD; для установленных систем рекомендуется применение модуля alterator-datetime.

7.54. features.in/oem

Эта фича обеспечивает автоматический запуск предварительной настройки, характерный для OEM-образов.

Дефолтные шаги определяются в файле /etc/alterator-setup/steps. Его дефолтное содержание: sysconfig notes-license datetime root users setup-finish

Для переопределения списка шагов используйте переменную OEM_STEPS. Пример: цель: use/oem @$(call set,OEM_STEPS,sysconfig notes-license datetime setup-finish)

Список доступных шагов для alterator-setup находится в /usr/share/alterator/steps/

7.55. features.in/office

Эта фича обеспечивает наличие и конкретизацию выбора офисного пакета по аналогии с выбором браузера. Разумеется, дополнительные варианты могут быть установлены явным или косвенным затребованием.

Следует понимать, что каждая из целей может быть использована лишь один раз, повторное упоминание будет проигнорировано make.

7.56. features.in/pack

Эта фича определяет формат упаковки создаваемого образа.

На данный момент поддерживаются iso (загрузочный ISO9660) и tar для дистрибутивов, tar/cpio с возможностью сжатия gz/xz (виртуальные окружения), а также различные варианты для образов виртуальных машин, поддерживаемые qemu-img.

7.57. features.in/pid1

Эта экспериментальная фича предназначена для обеспечения запуска заданного приложения в моно^Wкачестве единственного, т.е. PID 1.

Особенности результата:

- предельно быстрый запуск;
- работа под root;
- отсутствие какой бы то ни было инициализации окружения.

Возможна настройка сетевых интерфейсов средствами ядра, условия:

- статически собранный модуль для используемого интерфейса;
- доступность DHCP/BOOTP/RARP в сегменте локальной сети.

Пакет следует добавить в STAGE1_PACKAGES; путь к бинарнику задаётся в PID1_BIN; PID1_PANIC позволяет указать время до перезагрузки ядра при завершении работы приложения.

7.58. features.in/pkgpriorities

Эта фича обеспечивает добавление записей в файл `$(PKGBOX)/aptbox/etc/apt/pkgpriorities` после инициализации чрута, но перед установкой пакетов.

Содержимое файла `pkgpriorities` формируется на основе списка `PINNED_PACKAGES`. Значение приоритета по умолчанию определяется переменной `PIN_PRIORITY`, в которую при инициализации фичи записывается `"Important"`. Список приоритетов:

```
Essential, Important, Required, Standard, Optional, Extra
```

Переопределить значение приоритета можно отдельно для каждого пакета в списке, указав желаемый приоритет через двоеточие после имени пакета; например:

```
$(call add,PINNED_PACKAGES,my-package:Essential)
```

Используя PINNED_PACKAGES, можно заранее определить выбор того или иного пакета для удовлетворения виртуальной зависимости. Если виртуальный пакет присутствует в основном списке пакетов для установки, а пакет, его предоставляющий — в этом списке, то вероятность его установки повышается согласно приоритету. Однако если виртуальный пакет не выбран для установки или приоритетный пакет отсутствует в репозитории, то сборка образа продолжится без изменений. Следовательно, с помощью списка PINNED_PACKAGES можно влиять на состав дистрибутива, но его содержание, в отличие от обыкновенных списков пакетов, имеет *рекомендательный*, а не обязательный, характер.

7.59. features.in/plymouth

Эта фича предназначена для добавления поддержки plymouth — современной реализации bootsplash. Плотно взаимодействует с фичей branding по объективным причинам, но оформлена отдельно для возможности собирать образы с частичным брендированием либо "без излишеств".

Аргументы kernel cmdline *quiet splash* можно переопределить в переменной SPLASH_ARGS, например так:

```
@$(call set,SPLASH_ARGS,splash)
```

7.60. features.in/power

Эта фича конфигурирует поддержку управления питанием — выключение и регулировку частоты CPU для ACPI, засыпание для APM (не проверялось).

7.61. features.in/relname

Эта фича меняет содержимое файла /etc/altlinux-release в соответствии с установленной переменной RELNAME, что изменяет пункты загрузки GRUB.

Применяется при необходимости перекрыть внесенный брендингом текст.

7.62. features.in/repo

Эта фича предназначена для конфигурирования репозиториев в образе, включая генерацию хэшей и подключение к LiveCD.

По умолчанию таким репозиторием является RPMS.main (создаваемый sub/main), но возможно добавление addons, updates или иных по мере необходимости.

Результат — каталог ALTLinux/base/ для копирования в образ.

Дополнительно обрабатываемые переменные:

- REPO
  - позволяет выбрать и включить зеркало репозитория
  - не влияет ни на что, если вызывается цель use/repo/main
  - значение: http/alt (по умолчанию) либо протокол/зеркало
  - см. тж. /etc/apt/sources.list.d/, проверьте наличие нужного!
- REPO_MAIN
  - включает создание репозитория main внутри rootfs
  - влияет только на vm/ цели
  - значение: 1 для включения, любое другое значение для отключения

7.63. features.in/rescue

Эта фича дополняет базовый "живой" образ второй стадии специфическими для спасательного образа настройками и скриптовыми хуками.

Цель use/rescue/rw добавляет предварительно настроенный пункт загрузки, который в случае "просто гибридного" (не GPT) ISO, записанного на USB Flash, обеспечит создание и монтирование дополнительного раздела для сохранения данных между сессиями.

7.64. features.in/server

Эта фича предоставляет типичные для серверных образов наборы списков пакетов и модулей ядра.

7.65. features.in/services

Данная фича конфигурирует автоматический запуск сервисов при загрузке системы.

Поскольку в конкретном образе может быть желательно перекрыть умолчания предыдущей конфигурации, рекомендуется в фичах работать с переменными DEFAULT_SERVICES_* и оставить переменные SERVICES_* для релиз-менеджеров.

Выключение сервиса в каждой из этих пар имеет приоритет перед включением.

Предприняты особые меры в виде скрипта для install2, чтобы передать указание настроить службы инсталятору.

Для включения systemd-специфичных сервисов рекомендуется использовать DEFAULT_SYSTEMD_SERVICES_* и SYSTEMD_SERVICES_*.

Для включения служб systemd-logind нужно использовать DEFAULT_SYSTEMD_USER_SERVICES_* и SYSTEMD_USER_SERVICES_*.

Для того, чтобы замаскировать или размаскировать юнит systemd используйте SYSTEMD_SERVICES_MASK и SYSTEMD_SERVICES_UNMASK.

7.66. features.in/sound

Эта фича добавляет поддержку аудиоподсистемы (как ядерную, если не включена в kernel-image, так и утилиты).

7.67. features.in/speech

Эта фича полностью подготавливает русскоязычный или англоязычный вывод речи на базе сервера VoiceMan.

7.68. features.in/stage2

Эта фича служит для добавления в первую стадию хуков, необходимых при наличии в stage1 ядра (что типично, но не обязательно).

Передача информации о конфигурации ядра между stage1 и stage2 также требуется для оптимального сжатия squashfs-образа второй стадии.

Возможно пополнение списка опций конфигурации ядра (CONFIG_*), необходимых для загрузки целевого дистрибутива, посредством переменной STAGE1_KCONFIG (см. фичу efi в качестве примера).

7.69. features.in/syslinux

Добавление поддержки syslinux; требуется для инсталяторов, live/rescue; реализуется в рамках stage1.

Самостоятельное творческое использование на данный момент подразумевает знакомство с /usr/share/doc/syslinux-*/syslinux.txt и изучение кусочков конфигурации, которые уже существуют.

Цели config.mk:

- use/syslinux/ui/% — конфигурирование интерфейса (см. cfg.in/00*.cfg); при использовании автоматически добавляют syslinux в FEATURES;
- use/syslinux/timeout/% — задание таймаута автозагрузки (в десятых секунды);

- use/syslinux/%.com, use/syslinux/%.c32 — подключение одноименных модулей (копирование бинарников и включение кусочков конфигурации; экспериментальное);
- use/syslinux/%.cfg — подключение кусочков конфигурации.

Переменные generate.mk:

- BOOTARGS — дополнительные аргументы загрузчику;
- BOOTLOADER — isolinux (реализовано с оглядкой на syslinux/syslinux4);
- BOOTVGA — видеорежим, запрашиваемый у ядра (параметр vga=);
- SYSLINUX_UI — модуль интерфейса (если не указан, то внутренний prompt);
- SYSLINUX_MODULES — модули .com или .c32 (перечисляются без расширения);
- SYSLINUX_CFG — дополнительные кусочки конфигурации (например, localboot).

Здесь производится первичная обработка конфигурационных данных, окончательно проверяемых и используемых уже в инструментальном чруте.

Обратите внимание: фрагменты, соответствующие именам субпрофилей, добавляются автоматически; это поведение при необходимости отключается выставлением переменной SYSLINUX_DIRECT и тогда вместо use/syslinux/*.cfg следует применять прямое указание вида @$(call set,SYSLINUX_CFG,...).

Установить дефолтный пункт: Для того, чтобы установить конкретный дефолтный пункт (пример для LiveCD с поддержкой сессии):

```
@$(call set,SYSLINUX_DEFAULT,session)
```

Именем дефолтного пункта является LABEL.

7.70. features.in/tty

Эта фича занимается терминалами ввода-вывода, в первую очередь COM-портами (serial console).

Следует заметить, что systemd занимается развешиванием agetty самостоятельно.

7.71. features.in/uboot

Настраивает систему для использования загрузчика uboot.

7.72. features.in/uuid-iso

По умолчанию при сборке образа xorriso генерирует UUID образа вида YYYY-MM-DD-hh-mm-ss-cc из текущего времени. Если в командной строке xorriso есть пареметр -volume_date uuid YYYYMMDDhhmmsscc то UUID образа генерируется из него. Данная фича читает текущее время и создаёт переменные: UUID_ISO, содержащую YYYY-MM-DD-hh-mm-ss-cc UUID_ISO_SHRT, содержащую YYYYMMDDhhmmsscc Также фича добавляет в initrd файл /lib/udev/rules.d/60-cdrom_id.rules Это позволяет идентифицировать CD/DVD по UUID и использовать для загрузки инсталлятора method:disk,uuid:YYYY-MM-DD-hh-mm-ss-cc

7.73. features.in/vagrant

Эта фича обеспечивает специфичную для vagrant предварительную настройку образа файловой системы виртуальной машины.

Обратите внимание, что специфика включает широко известные: - пароли root и пользователя vagrant с беспарольным sudo; - "секретный" ключ от публичной части у пользователя vagrant.

См. тж.:

https://bugzilla.altlinux.org/28553

https://docs.vagrantup.com/v2/boxes/base.html

7.74. features.in/vmguest

Эта фича предназначена для конфигурирования поддержки выполнения дистрибутивов в качестве гостей в среде виртуальных машин.

7.75. features.in/volumes

Эта фича обеспечивает выставление нужного профиля разбивки дисков при установке с помощью installer или livecd-install.

7.76. features.in/wireless

Эта фича занимается добавлением поддержки беспроводных соединений.

7.77. features.in/x11-autologin

Эта фича добавляет в формируемый пользовательский корень (как правило, live) функцию автоматического входа путём конфигурирования отдельно

запрошенного для установки display manager (например, lightdm) либо специального средства (пакеты nodm или autologin).

Обратите внимание: с autologin могут быть проблемы под systemd, а при использовании работающего в таком окружении nodm на сегодня отмечено наличие /sbin:/usr/sbin в пользовательском PATH перед /bin:/usr/bin, что приводит к неработоспособности consolehelper и livecd-install, который им пользуется.

7.78. features.in/x11-autostart

Эта фича добавляет в формируемый пользовательский корень (как правило, live) функцию автоматического запуска графической сессии; обратите внимание, что автоматическим входом после запуска графики занимается соседняя фича x11-autologin.

7.79. features.in/x11

Эта фича добавляет базовую поддержку графической системы X11, а также комплектует типовые десктопные окружения и средства графического входа в систему.

Для добавления X-сервера и драйверов используйте цели: - use/x11/xorg — свободные драйверы, может недоставать акселерации, особенно 3D, и функций энергосбережения, но поддерживают наиболее широкий спектр оборудования для типичных десктопных задач; - use/x11/3d — по возможности подключаются проприетарные драйверы NVIDIA, обычно обладающие более высоким уровнем ускорения графики, но также имеющие и больше проблем совместимости со свежими ядрами/xorg-server, а заодно обычно рано теряющие поддержку "устаревших" видеокарт.

Возможно предоставлять в образе одновременно свободные и закрытые драйверы, но в этом случае следует понимать, что автоопределение в X.org предпочитает свободный драйвер и nvidia при наличии nouveau не будет автоматически выбран, т.е. потребуется дополнительное конфигурирование (вручную или при помощи alterator-x11) — для live-систем это может быть лишено практического смысла.

Обратите внимание: как и в фиче bootloader, переключение на какой-либо дисплейный менеджер срабатывает только один раз;

```
use/x11/xdm use/x11/lxdm use/x11/xdm
```

приведёт к выставлению lxdm, а не xdm, поскольку это будет последняя "новая" цель с точки зрения make.

При необходимости перекрыть последнее изменение добавьте:

```
@$(call set,THE_DISPLAY_MANAGER,нужный)
```

7.80. features.in/x11-vnc

This feature allows to use X11 through VNC server. It adds x11vnc package and sets default password to alt. Another thing is that this feature adds dummy video adapter configuration to the /etc/X11/xorg.conf.d/. x11vnc becomes default service.

7.81. features.in/xdg-user-dirs

Эта фича обеспечивает наличие "ручки" для конфигурирования типовых пользовательских каталогов для нескольких типов данных и предоставляет возможность задавать предпочитаемые умолчания, которые могут различаться по дистрибутивам.

См. тж.:

https://freedesktop.org/wiki/Software/xdg-user-dirs

Глава 8. sub.in

Этот каталог содержит субпрофили; содержимое затребованных (названия которых содержатся в значении переменной SUBPROFILES, которую заполняют цели sub/* — см. lib/sugar.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 (см. соотв. README):

- rootfs является особым случаем, который используется при формировании файловых систем, предназначенных для пользователя (т.е. корень LiveCD, образа VM, …)
- stage1: propagator и загрузчик (совместно с фичей syslinux); типично требуется для инсталяторов, 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 что-либо в image-scripts.d смысла нет, только в scripts.d)

8.1. sub.in/main

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

Рекомендуется использовать BASE_PACKAGES и BASE_LISTS для того, что должно быть установлено по умолчанию, и MAIN_PACKAGES, MAIN_LISTS — для того, что должно быть доступно на носителе; подробнее см. в документации фичи metadata.

Если что-либо требуется как в main, так и в live, применяйте THE_PACKAGES и THE_LISTS вместо дублирования вручную.

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 (если не указан иной суффикс посредством переменной MAIN_SUFFIX).

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 ядер посредством STAGE1_KFLAVOURS, то будет взят из KFLAVOURS; если используется загрузчик отличный от grub, то будет взят последний указанный в STAGE1_KFLAVOURS или KFLAVOURS; если не указать явно регэкс, описывающий требуемые в инсталяторе kernel-modules-**, посредством STAGE1_KMODULES_REGEXP — будут доступны модули из kernel-image (упаковываются в boot/full**.cz).

Сам список модулей, попадающих в full.cz, определяется в файле modules (наиболее базовые!) и дополняется указанным в переменной STAGE1_MODLISTS набором списков модулей, см. features.in/stage2/stage1/modules.d/ в качестве примера.

Требуется для инсталяционных, 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. реализацию).

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.

При копировании списков в BUILDROOT происходит их обработка с применением архитектурнозависимых макросов, см. doc/archdep.txt

NB: списки пакетов есть смысл выделять в файлы при повторном использовании либо при заметном объёме, когда перечисление прямо в конфигурации сильно снижает её читаемость.

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 из символов в наборе [a-zA-Z0-9_]

Не используйте в слове "-"); рекомендуется разделять слова "+".

Применение: дополнение жёстко статически заданной функциональности (как правило, обязательной в данном образе) более "плавающим" в долгосрочном плане результатом раскрытия списка тегов (который может покрывать второстепенные вещи способом, обычно требующим меньше внимания).

Реализация никак не сопряжена с pkg.in/groups/

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 pkg.in/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 наличие ALT 8.0+ и GNU make 3.82+ (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)
  - трассировка выполняется при REPORT=1 для формирования графа зависимостей между промежуточными целями сборки конечного образа
  - характерный признак пропуска — разрыв графа (report-targets.png)
- pkg.in/lists/Makefile
  - ожидает, что названия списков пакетов указываются в переменных вида *_LISTS, и копирует в генерируемый профиль только их
  - если задать имя файла списка пакетов непосредственно в Makefile субпрофиля, он не будет скопирован
  - характерное сообщение об ошибке:

    ```
    E: Couldn't find package
    ```
- features.in/kernel/stage1/scripts.d/80-make-initrd
- features.in/stage2/stage1/scripts.d/03-test-kernel
- sub.in/stage1/Makefile
  - если используется загрузчик отличный от grub, то в stage1 попадёт последнее ядро, указанное в STAGE1_KFLAVOURS или KFLAVOURS
  - если добавить какой-либо kernel-image в STAGE1_PACKAGES*, результат может быть неожиданным
  - обратите внимание: bin/tar2fs умеет несколько ядер
  - 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 инсталятора, livecd 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 блох

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: https://www.altlinux.org/Mkimage/debug

Глава 13. Оформление кода

- постарайтесь не вносить без обсуждения разнобой 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, если бы сами хотели прочесть описание сделанного на месте другого человека; в любом случае старайтесь внятно описывать коммиты, при необходимости также спрашивайте совета: документация кода порой не менее важна, чем сам код, и призвана не повторять его, но пояснять намерения и неочевидности.

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 вылезать за 80 колонок;

- избегайте merge-коммитов в коде, который предлагаете для включения в основную ветку, предлагайте pull-реквесты без конфликтов при мерже, что облегчит работу с промежуточными состояниями.

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/tar2fs 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/tar2fs
```

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 (поэтому фича build-vm сперва проверяет наличие системно установленного пакета и по возможности старается запустить под sudo скрипт из него, доступный по записи только 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 ROOTPW=reallysecret1 vm/bare.img && kvm -hda ~/out/bare.img
```

Если при сборке образа файловой системы произойдёт сбой, может оказаться нужным вручную освободить используемые loop-устройства, например, так:

```
# losetup -a
# kpartx -d /dev/loop0
# losetup -d /dev/loop0
```

Глава 15. QEMU

Для сборки на "неродной" архитектуре с применением трансляции посредством QEMU установите пакет livecd-qemu-arch и выполните команду register-qemu-armh 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=armh APTCONF=/etc/apt/apt.conf.sisyphus.arm ve/bare.tar
```

Обратите также внимание на https://bugzilla.altlinux.org/34638

Глава 16. Архитектурно-зависимые фрагменты

16.1. Makefile

Достаточно воспользоваться ifeq/ifneq, сравнивая $(ARCH) с нужным:

```
ifeq (x86_64,$(ARCH))
EFI_LISTS := $(call tags,base efi)
endif
```

При необходимости сравнить со списком ("любой x86") можно сделать так:

```
ifeq (,$(filter-out i586 x86_64,$(ARCH)))
use/x11/xorg: use/x11 use/x11/intel use/firmware
else
use/x11/xorg: use/x11
endif
```

В рецептах (shell-часть Makefile) используйте $(ARCH) или $$ARCH.

16.2. скрипты

В скриптовых хуках ({image-,}scripts.d/*) проверяйте $GLOBAL_ARCH.

16.3. списки пакетов, профили групп

Бывает так, что в списке пакетов есть смысл упоминать какой-либо из них только для определённой архитектуры (например, wine или steam); в таких случаях можно воспользоваться механизмом подстановки, который пословно обрабатывает списки и в случае наличия суффикса @ARCH оставляет только слова, в которых этот суффикс соответствует заданной архитектуре сборки.

Например, для Simply Linux в mkimage-profiles-desktop есть строчки:

```
@I586_ONLY@haspd
@X86_64_ONLY@i586-haspd
```

В случае mkimage-profiles они должны выглядеть так:

```
haspd@i586
i586-haspd@x86_64
```

или упрощённо (с версии 1.2.12):

```
haspd@IA32
```

С версии 1.3.15 поддерживается макрос E2K ("любое поколение e2k*") и ARM (armh или aarch64), а также выборка "для любой архитектуры, кроме" (например, @!E2K, или "@!ARM).

С версии 1.4.21 поддерживается перечисление архитектур через запятую после "@" или "@!":

```
LibreOffice-still@X86,aarch64
java-11-openjdk@!E2K,riscv64
```

Для преобразования можно воспользоваться следующей командой:

```
sed -r -e 's/@I586_ONLY@([^\t ]+)/\1@i586/g' \
       -e 's/@X86_64_ONLY@([^\t ]+)/\1@x86_64/g'
```

При необходимости добавить пакет только на x86-архитектурах (неважно, i586 или x86_64) можно воспользоваться макросом X86 (с версии 1.2.12):

```
xorg-drv-intel@X86
```

Аналогичная функциональность реализована для профилей установки.

16.4. загрузчики

Как правило, сперва понадобится доработка mkimage — см. скрипты tools/mki-pack-*boot — и лишь затем профиля; см. тж. lib/boot.mk и фичу pack.

Глава 17. Метапакеты

17.1. списки пакетов, профили групп

Для раскрытия метапакета в список используется суффикс @META:

```
engineering-2D-CAD@META
```

apt запрашивает зависимости такого пакета и добавляет их в список пакетов после этого метапакета.

Обратите внимание, что игнорируются нечёткие зависимости, т.е. предоставляемые несколькими пакетами.

Возможности совмещать с суффиксом @ARCH нет. Так что имейте в виду, что метапакет должен быть доступен для всех целевых архитектур.

Для лучшего понимания работы механизма раскрытия списка нужно смотреть bin/metadep-expander. metadep-expander выполняется до archdep-filter.