От пропагатора к altboot

Леонид Кривошеин <klark@altlinux.org>
Антон Мидюков <antohami@altlinux.org>

Version 4.0, 01.09.2024

Оглавление

Об этом документе

Перед вами полное справочное руководство к **bootchain** + **altboot**.

Краткое руководство доступно на ВиКи: https://www.altlinux.org/Installer/common/altboot

На момент написания (версия **0.1.5-alt24**) весь проект представляет собой 12 пакетов noarch, один SRPM, 170 исходных файлов на bash и make или 8650 строк кода, включая README в подпакетах.

История изменений

1.0 / 02.07.2021

Первоначальная редакция в формате ODT (revision 141), созданная сразу после публикации в репозитории make-initrd-bootchain 0.1.0-alt1.

2.0 / 22.07.2024

Вторая редакция в формате ODT (revision 183), в которой выполнена актуализация документации до версии make-initrd-bootchain 0.1.5-alt23.

3.0 / 28.07.2024

Третья редакция в формате ODT (revision 186) от Антона Мидюкова <antohami@altlinux.org>.

2.1 / 08.08.2024

Вторая редакция в формате Asciidoc. Выполнено конвертирование документа редакции 2.0 из ODT в формат Asciidoc без изменений по тексту, добавлены стилевые файлы, документация перенесена в дерево проекта.

3.1 / 11.08.2024

Третья редакция в формате Asciidoc без изменений по тексту, но с учётом правок от Антона Мидюкова <antohami@altlinux.org> к редакции 2.0.

4.0 / 01.09.2024

Четвёртая редакция в формате Asciidoc. Выполнена финальная вёрстка и синхронизация с кратким руководством на ВиКи, расставлены перекрёстные ссылки, добавлены описания шагов ping и wait-resume, добавлена история изменений.

Вводные определения

`propagator`

Компактная программа, выполняющая поиск второй стадии инсталлятора (altinst), LiveCD (live) или rescue. Написана на C в конце 90-х. Использовалась в Linux Mandrake и далее во всех дистрибутивах ОС Альт, вплоть до «десятой платформы». Давно морально устарела, имеет много нерешённых проблем.

`make-initrd`

Дистронезависимый инструмент для сборки образов initramfs, обеспечивающих начальную загрузку. make-initrd со всеми своими фичами также предоставляет некий run-time (набор скриптов), попадающий в initramfs и выполняющийся на первой стадии загрузки (stage1). Задача этого run-time — как можно скорее обнаружить корень со второй стадией загрузки (stage2), подмонтировать этот корень и передать управление в INIT второй стадии загрузки.

`make-initrd-propagator`

Фича make-initrd, обеспечивающая попадание в образ initramfs универсального загрузочного носителя программы propagator и дополнительных скриптов. Загрузка таких носителей устроена иначе, нежели чем на обычной установленной rootfs. Вместо более гибкого run-time make-initrd в них работает propagator, а следом за ним отрабатывает скрипт **init-bottom** из описываемой фичи. Он обеспечивает, в том числе, взаимодействие с **alterator-netinst**.

`pipeline`

Фича make-initrd, предлагающая «пошаговый» принцип загрузки, написанная автором make-initrd Алексеем Гладковым (a.k.a. legion@). В исходном варианте не обеспечивает слоя совместимости с инсталлятором, LiveCD или Rescue в ОС Альт, конфигурируется только через параметры в /proc/cmdline, но может быть использована для создания простых сценариев загрузки с фактором синхронизации событий и позволяет гибко комбинировать написанные «шаги»-скрипты.

`bootchain`

Форк и дальнейшее развитие pipeline. На момент написания предлагает более десятка модулей и ещё не попала в апстрим make-initrd.

`altboot`

«Альтернативная загрузка» — дистронезависимая система загрузки, построенная поверх bootchain, работающая как часть run-time make-initrd, альтернатива propagator с make-initrd-propagator в дистрибутивах ОС Альт на «одиннадцатой платформе», обеспечивающая дополнительный функционал, слой совместимости с установщиком, alterator-netinst и более гибкие возможности дальнейшего расширения.

См. также:

- https://www.altlinux.org/Propagator
- https://www.altlinux.org/Make-initrd
- https://www.altlinux.org/Alterator-netinst

1. Достоинства bootchain + altboot

- интеграция с run-time **make-initrd** и его фичами;
- возможность работы и отладки на обычной установленной Linux rootfs;
- модульная, легко расширяемая архитектура;
- возможность загрузки на всех поддерживаемых платформах;
- независимость от вендора: больше нет жёсткой привязки к ОС Альт;
- возможность более тонко настраивать некоторые параметры загрузки;
- поддержка read-only слоёв LiveCD всеми «монтируемыми» методами загрузки, а не только NFS;
- расширены исходные возможности пропагатора по загрузке с сервера SAMBA;
- появилась возможность создания постоянного хранилища сеансов LiveCD на NVME дисках и высокоскоростных накопителях MMC;
- возрождён и улучшен утраченный функционал для добавления «забытых» модулей ядра или правил udev при загрузке;
- тихая загрузка: интерактивная часть **chaind** работает на терминале tty2, переключение на который происходит спустя 8 секунд бездействия;
- в отличии от **pipelined**, демон **chaind** может перейти на «передний план» и работать на конкретном терминале;
- в отличии от **pipelined**, демон **chaind** может менять логику «на лету», поддерживает циклы и условные переходы;
- вместе **bootchain+interactive** обеспечивают основу для построения «шагов» текстового инсталлятора в stage1;
- полная обратная совместимость с ранее написанными «шагами» для демона **pipelined**;
- обратная совместимость с текущим инструментарием создания дистрибутивов ОС Альт: mkimage-profiles, alterator-netinst, различными installer-фичами;
- штатная поддержка закрытия на запись обычной rootfs «шагами» **liveboot** и **overlayroot**.

См. также:

- https://bugzilla.altlinux.org/30472
- https://bugzilla.altlinux.org/30315
- https://www.altlinux.org/Propagator
- https://www.altlinux.org/Make-initrd
- https://www.altlinux.org/Make-initrd-propagator
- https://www.altlinux.org/Alterator-netinst
- https://www.altlinux.org/Mkimage-profiles

2. Недостатки bootchain + altboot

- в **altboot** пока не реализована более тесная интеграция с фичей make-initrd «resume», возможно стоит отключать эту фичу при использовании на обычной Linux rootfs во избежании «гонок»;
- в **altboot** пока не реализована более тесная интеграция с фичей make-initrd «network», в отличии от пропагатора сеть сейчас конфигурируется только через /proc/cmdline, нет собственных диалогов для её настройки;
- скрипты и диалоги **altboot**, как и установщика ОС Альт, пока не рассчитаны на работу с IPv6, хотя фича make-initrd «network» понимает IPv6;
- **propagator** умел конфигурировать ADSL-соединение, фича make-initrd «network» не предназначена для настройки ADSL/PPPoE;
- не все фичи bootchain+altboot снабжены полноценным **README.md**;
- **review** кода bootchain+altboot был выполнен примерно наполовину и в этом коде пока остаются неисправленные замечания, вследствие чего он ещё не стал частью проекта make-initrd, а ведение проекта отдельно от make-initrd требует периодической синхронизации кода с апстримным проектом.

Примечания:

1. Ручная настройка **ADSL/PPPoE**-соединения для реализации начальной загрузки на сегодняшний день неактуальна. Можно сказать, произошёл отказ от этого **legacy**.
2. Современная техника для сетевой загрузки так или иначе использует стандартную прошивку iPXE. Для унификации сетевой загрузки старое оборудование тоже может выполнить переход в **прошивку iPXE** на самом раннем этапе. Стандартная **прошивка iPXE** позволяет во встроенных диалогах вручную сконфигурировать любую сетевую карту. Таким образом, встроенных в **прошивку iPXE** диалогов для какого-то особого ручного конфигурирования сети вполне достаточно.
3. Сам **altboot** не занимается настройкой сети, он ждёт, когда её настроит другая фича make-initrd. Она, в свою очередь, может быть сконфигурирована через параметры загрузки, в том числе, на получение данных от **прошивки iPXE**.

3. Модули bootchain + altboot

Разделение на модули (подпакеты) позволяет добиться оптимального наполнения initramfs только необходимым для загрузки содержимым. Так, для загрузки по протоколу NFS достаточно установить пакет **make-initrd-bootchain-nfs** и задействовать фичу **bootchain-nfs**, всё остальное попадёт в initramfs по зависимостям. Для создания универсального образа stage1 достаточно установки одного мета-пакета **make-initrd-bootchain**.

- **make-initrd-bootchain** — мета-пакет, собирающий воедино все существующие модули bootchain, он же исходный SRPM;
- **make-initrd-bootchain-core** — базовый функционал, включая «шаги» debug, fg, mountfs, noop, noretry, overlayfs, retry, rootfs и wait-resume, т.е. это расширение исходной фичи pipeline;
- **make-initrd-bootchain-getimage** — обеспечивает метод загрузки образов по сети утилитой wget по протоколам HTTP и FTP, отделён в подпакет из исходной фичи pipeline;
- **make-initrd-bootchain-waitdev** — обеспечивает метод ожидания локального носителя по заданной спецификации, немного расширен и отделён в подпакет из исходной фичи pipeline;
- **make-initrd-bootchain-interactive** — обеспечивает *интерактивное взаимодействие* и предоставляет *диалоговые виджеты*;
- **make-initrd-bootchain-waitnet** — обеспечивает ожидание сети и экспорт сетевых настроек в stage2, используется всеми сетевыми *методами загрузки* altboot (ftp, http, nfs, cifs) и является временным кодом;
- **make-initrd-bootchain-altboot** — весь общий функционал пропагатора и «шаг», транслирующий его аргументы;
- **make-initrd-bootchain-localdev** — обеспечивает *методы загрузки* disk/cdrom, попадание в stage1 на этапе загрузки дополнительных модулей ядра и правил udev;
- **make-initrd-bootchain-liverw** — обеспечивает дополнительный функционал для работы с *постоянным хранилищем сеансов LiveCD*;
- **make-initrd-bootchain-nfs** — обеспечивает *метод загрузки* с сервера NFS и поддержку *read-only слоёв LiveCD* (сквошей);
- **make-initrd-bootchain-cifs** — обеспечивает *метод загрузки* с сервера SAMBA и поддержку *read-only слоёв LiveCD* (сквошей);
- **make-initrd-bootchain-doc** — документация и набор тестов.

См. также:

- https://packages.altlinux.org/ru/sisyphus/srpms/make-initrd-bootchain/
- https://git.altlinux.org/gears/m/make-initrd-bootchain.git
- https://github.com/klark973/make-initrd-bootchain

4. Быстрое знакомство с bootchain + altboot

4.1. Propagator + init-bottom ► bootchain + altboot

Тем, кто уже пользовался пропагатором, достаточно подключить *необходимые модули* **bootchain** вместо **propagator** + **make-initrd-propagator** и добавить к привычным всего пару опций в /proc/cmdline:

```
root=bootchain bootchain=fg,altboot automatic=... [ip=...] \
  [ramdisk_size=...] [lowmem] [stagename={altinst|live|rescue}] ...
```

Для работы сетевой загрузки необходимо добавлять параметр **ip=...**, он нужен make-initrd для настройки сети: **altboot** полагается на то, что сеть будет настроена кем-то и ожидает появления сети при использовании сетевых *методов загрузки*, таких, как **nfs** или **http**. Например, **ip=dhcp** заставит make-initrd поднять сеть и автоматически её настроить на всех имеющихся сетевых интерфейсах по протоколам DHCPv4 и DHCPv6.

Для дистрибутивов ОС Альт лучше задавать через /proc/cmdline параметр **ip=dhcp4**, поскольку установщик не работает с IPv6, а отсутствие в сети двух разных типов DHCP-серверов увеличивает время настройки секунд на 20 для каждого интерфейса.

В конфигурациях с несколькими сетевыми интерфейсами можно задавать сразу два параметра: **ifname=bootif0:<macaddr>** **ip=bootif0:dhcp4** — это заставит фичу make-initrd «network» переименовать интерфейс с указанным MAC-адресом в «bootif0» и поднять сеть только на нём и только по протоколу DHCPv4. В сценариях сетевой загрузки по iPXE с несколькими сетевыми картами прошивка уже выбрала ту, на которой есть несущая, и которая настроилась по DHCP, именно её MAC-адрес передаётся такой записью в iPXE-скрипте: **ifname=bootif0:${net/mac}**.

Конфигурирование **bootchain** и **altboot** выполняется через файл **/etc/sysconfig/bootchain**, используемые фичи перечисляются в **/etc/initrd.mk** перед созданием образа stage1 командой **make-initrd**. Например:

```
apt-get install make-initrd-bootchain-nfs
echo "FEATURES += bootchain-nfs" >>/etc/initrd.mk
make-initrd
```

Некоторые детали есть также в подразделе «*Создание своего initrd.img с bootchain*».

Параметр **automatic** обрёл несколько новых аргументов, утратив при этом аргументы, связанные с настройкой сети, типа «**network:...**» (они игнорируются, т.к. сеть теперь настраивает сам make-initrd). Например, можно задавать пользовательский таймаут или дополнительные опции монтирования. См. детали в разделе «*Аргументы параметра automatic*». Немного поменялась и логика работы со спецификациями дисков: теперь спецификация загрузочного диска должна однозначно указывать только на один диск. Если спецификации соответствует больше или меньше носителей, **altboot** выведет *диалог выбора накопителя*, тогда как **propagator** мог загрузиться с первого попавшегося.

Ядерный параметр **ramdisk_size=...** стал необязательным: начиная с версии 0.1.5-alt17 образы загружаются в TMPFS, если параметр не указан. Новая логика работы некоторых параметров для методов **http** и **ftp**:

- Если указан **ramdisk_size=...**, на сервере ожидается каталог, из которого сквош скачивается в RAM-диск;
- При наличии **automatic=type:iso,...**, нераспакованный ISO-образ скачивается в TMPFS;
- Иначе на сервере ожидается каталог, из которого сквош скачивается в TMPFS.

При этом параметры **lowmem** и **live** не оказывают влияния на использование памяти. Для всех остальных методов:

- Если указаны **lowmem** или **live**, в ОЗУ ничего не загружается, просто монтируется сквош;
- Если указан **ramdisk_size=...**, сквош загружается в RAM-диск;
- Иначе сквош загружается в TMPFS.

В последних трёх случаях, если указан путь не к каталогу, а к нераспакованному ISO-образу, то сначала будет смонтирован этот ISO-образ. **altboot** ориентируется не столько на наличие парметра **ramdisk_size=...**, сколько на наличие RAM-диска, созданного ядром по этому параметру, при этом сравнивая его фактический размер с необходимым для загрузки сквоша. Даже при наличии данного параметра, ядро не всегда создаёт RAM-диски. Например, некоторые Realtime-ядра не создают.

4.1.1. Создание своего initrd.img с bootchain

Для сборки универсального **initrd.img** требуется создать в пользовательском каталоге два конфига: **bootchain** и **initrd.mk**, после чего выполнить команду:

```
make-initrd --no-checks AUTODETECT= \
  -c <путь/к/каталогу_с_конфигами>/initrd.mk \
  BOOTCHAIN_PATH=<путь/к/каталогу_с_конфигами> \
  -v -k <версия_ядра>
```

В конфиге **initrd.mk** подключаются необходимые фичи и добавляются модули ядра либо каталоги с ними. В качестве примера смотрите дистрибутивный скрипт сборки универсального образа initramfs. Внутри каждого ISO-образа, собранного при помощи mkimage-profiles, в каталоге **/.disk** находятся конфиги **bootchain** и **initrd.mk**, а также скрипт **mkinitrd** с командами для сборки initrd*.img из этого ISO-образа.

4.2. Pipeline ► bootchain

Тем, кто уже пользовался фичей make-initrd «**pipeline**», достаточно подключить *модуль* **make-initrd-bootchain-waitdev** для локальной загрузки или **make-initrd-bootchain-getimage** для сетевой загрузки или оба, если это необходимо. *Можно сделать фичу «**pipeline**», вытягивающую оба модуля, тогда и этого не потребуется!* В /proc/cmdline можно использовать схожий синтаксис с **bootchain**, а можно оставить прежний вариант с **pipeline**:

```
root=pipeline pipeline=waitdev,mountfs,mountfs,overlayfs,rootfs ...
```

При втором варианте работы демон **chaind** ведёт себя полностью как **pipelined** с одним небольшим отличием: если «шаг» зафейлится, **chaind** повторит его запуск ещё четыре раза с паузами в две секунды, тогда как оригинальный **pipelined** повторял бы запуск «шага» бесконечно с паузой в одну секунду. Примечание: *в последних версиях make-initrd такое поведение можно изменить, а сам pipeline перестал быть демоном.*

chaind допускает завершение работы скриптов с кодом 2 для «шагов», когда работает в режиме **pipelined**, и воспринимает этот код от них, как необходимость разорвать цикл и перейти в stage2. Но в своём «родном» режиме демон всегда ожидает от скриптов код возврата 0, рассматривая остальные ситуации, как сбой. См. детали в разделе «*Совместимость с pipeline*».

4.3. bootchain + altboot «с нуля»

Тем, кто незнаком ни с **propagator**, ни с **pipeline**…

Демон **chaind** запускает «шаги» (небольшие скрипты) один за другим, обеспечивая «пошаговую» загрузку. Каждый «шаг» выполняет какую-то работу: скачивает образ, монтирует, накладывает оверлей, и т.д., при этом может использовать результаты любых предыдущих «шагов». Список «шагов» первоначально определяется параметром **bootchain=…**, но в процессе работы список оставшихся «шагов» может быть изменён. Для каждого «шага» отдельно определяются его входные параметры, что может загромождать /proc/cmdline, однако есть «шаги», такие как **altboot** или **overlayroot**, убирающие синтаксическое нагромождение во внутренний конфигурационный файл. Из сказанного следует несколько важных выводов:

- «Шаги» **altboot** можно выполнять и конфигурировать каждый по отдельности, не используя одноимённый «шаг» **altboot**.
- «Шаги» **altboot** работают в рамках «пошаговой» концепции **bootchain**, первоначально спроектированной для **pipeline**.
- Возможно комбинировать «шаги» **pipeline**, **bootchain** и **altboot**, когда необходимо добиться чего-то «особого».

Режим полной совместимости с пропагатором обеспечивается «шагом» с именем **altboot**. В этом режиме **altboot**'овским «шагам» не требуется дополнительное конфигурирование через /proc/cmdline и каждый «шаг» может быть пройден лишь единожды. Если «шага» **altboot** нет в цепочке, другие **altboot**'овские «шаги» указываются непосредственно в параметре **bootchain=…**, работают «сами по себе», т.е. могут повторяться и должны отдельно конфигурироваться через /proc/cmdline.

Демон **chaind** может не только менять «на лету» логику работы благодаря перегрузке списка «шагов», он также может в любой момент времени перейти на передний план и продолжить выполнение в **интерактивном режиме** на конкретном терминале. См. детали в *соответствующем разделе* и в *описании «шага» **fg***. Вместе эти возможности создают основу для построения текстовых инсталляторов в stage1, **propagator** в ОС Альт как раз является первой частью инсталлятора с текстовым интерфейсом.

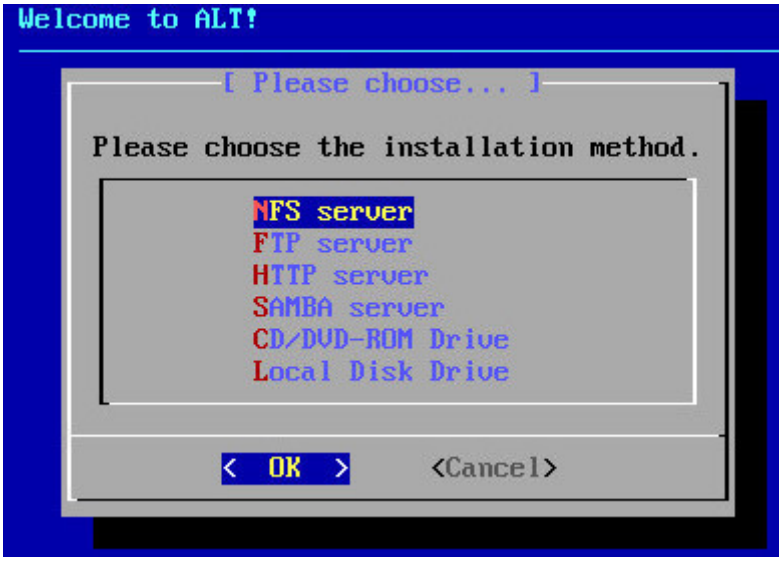
См. также:

- http://0x1.tv/20210418F
- https://youtu.be/C-NsPKvsCAw?t=6855
- https://bugzilla.altlinux.org/42967#c4
- https://bugzilla.altlinux.org/50888#c15
- https://bugzilla.altlinux.org/42966
- https://www.altlinux.org/Propagator
- https://www.altlinux.org/Realtime

5. Методы загрузки altboot

Метод загрузки «живой» системы (или **метод установки** ОС) определяет способ нахождения сквоша второй стадии загрузки (stage2), такой, как **live**, **rescue** или **altinst**. По сути, это самый первый «шаг» установщика (или загрузчика) дистрибутива в stage1, если не считать «шага» **выбора метода** загрузки, который обычно скрыт от пользователя. Данный подход изначально был заведён в пропагаторе, все **методы загрузки** (установки) ОС были заимствованы из него вместе с диалогами и соответствующими *параметрами загрузки*.

В **altboot** практически все параметры любого **метода загрузки** имеют свои значения по умолчанию, т.е. для загрузки достаточно указать в /proc/cmdline (**automatic=method:...**) или выбрать средствами диалога только нужный **метод загрузки**. При загрузке с обычной Linux rootfs, а не с универсального загрузочного носителя, напротив, метод не указывается (method='').



- **nfs** — указанный ISO-образ (либо каталог с распакованным ISO-образом) монтируется с сервера NFS;
- **ftp** — сквош stage2 (или ISO-образ) скачивается целиком с сервера по протоколу FTP;
- **http** — сквош stage2 (или ISO-образ) скачивается целиком с сервера по протоколу HTTP;
- **cifs** — указанный ISO-образ (либо каталог с распакованным ISO-образом) монтируется с сервера SAMBA;
- **cdrom** — загрузка с локального носителя CD/DVD-ROM Drive, в том числе, с по-байтно скопированного на перезаписываемый диск образа ISO-9660 Hybrid, и опциональной возможностью работы с *постоянным хранилищем сеансов LiveCD*;
- **disk** — загрузка с локального носителя, в том числе, записанного в формате ISO-9660 Hybrid на перезаписываемый диск, с раздела диска либо лежащего на нём ISO-образа, и опциональной возможностью работы с *постоянным хранилищем сеансов LiveCD*.

Методы загрузки **http**/**ftp** пока что не предназначены для работы с *read-only слоями LiveCD*, но если кому-то это будет очень нужно, добавить их поддержку не очень сложно. То же касается и указания номера порта. Пропагатор не поддерживал добавление номера порта и явно подразумевал использование стандартых портов для протокола (80, 21). В этой части **altboot** унаследовал данное поведение.

Методы **http/ftp** обеспечиваются «шагом» **download**. Существует два режима работы с этими **методами загрузки** — старый вариант, когда путь на сервере рассматривается как каталог, из которого загружается файл со сквошом второй стадии, и новый вариант, когда путь на сервере рассматривается как ISO-образ, загружаемый целиком, вместо сквоша. В пропагаторе поддержка нового варианта тоже появилась, но позже, так что по параметрам могут быть отличия. В случае **altboot**, для работы с новым режимом достаточно убрать параметр загрузки **ramdisk_size=...** и добавить параметр **automatic=...,type:iso,...**

Внутренний метод **url** аналогичен **http** (или **ftp** с анонимным доступом), в диалоговое меню **выбора методов** он не выводится, но его можно задействовать через /proc/cmdline, а также он используется для указания файла в каталоге, смонтированного на предыдущем шаге, и предназначенного для дальнейшего использования в качестве ISO-образа или сквоша. Для всех «сетевых» **методов загрузки** отдельно через /proc/cmdline должна быть сконфигурирована сеть. См. параметр **ip=...** и другие в фиче **make-initrd** «network».

При использовании «локальных» **методов загрузки** (**disk** или **cdrom**), становится доступной возможность обновлять «на лету» образ initramfs с дополнительного внешнего носителя, что в частности позволяет догружать недостающие модули ядра или правила udev. См. детали в *описании «шага» **oemsetup***. Совместно с *модулем* **make-initrd-bootchain-liverw** оба метода поддерживают возможность работы с *постоянным хранилищем сеансов LiveCD*.

Оба метода имеют небольшие отличия от исходной реализации в пропагаторе. Если гибридный ISO-образ записан на перезаписываемый носитель (USB-флеш, MMC-карту, и т.д.), то метод загрузки **cdrom** выполняет загрузку не с диска, а с первого раздела, что позволяет, при первой необходимости, создать *дополнительный раздел для работы с сеансами LiveCD*. Метод **disk** позволяет организовать загрузку не только с диска или раздела, но и с лежащего на нём ISO-образа, а это можно использовать для загрузки разных дистрибутивов, размещённых на одном носителе.

См. также:

- https://bugzilla.altlinux.org/37080
- https://bugzilla.altlinux.org/40554
- https://www.altlinux.org/Propagator
- https://www.altlinux.org/Make-initrd
- https://www.altlinux.org/Alterator-netinst
- https://github.com/osboot/make-initrd/tree/master/features/network
- https://ru.wikipedia.org/wiki/ISO_9660
- https://en.wikipedia.org/wiki/Hybrid_disc

6. Базовые «шаги» altboot

В результате декомпозиции пропагатора различные типовые действия всех *методов загрузки* превратились в отдельные «шаги» **altboot**, выполняющие конкретные задачи в рамках концепции pipeline/bootchain. Базовые «шаги» — это не все возможные «шаги» загрузочной цепочки **bootchain**, а только те, что используются практически с каждым из *методов загрузки* **altboot**. Код «шагов» для большинства загрузочных методов вынесен в отдельные подпакеты.

Подпакет **make-initrd-bootchain-altboot** определяет следующие базовые «шаги»:

- **altboot** — определяет *метод загрузки* и следующие «шаги» bootchain, транслирует аргументы пропагатора в аргументы следующих «шагов» загрузочной цепочки;
- **download** — многофункциональный «шаг», использующий curl, обеспечивает методы загрузки url, http, ftp, загружает в память squash, может загрузить образ чего угодно прямо в указанное блочное устройство;
- **copyfile** — сильно упрощённый вариант «шага» download, не использующий сеть, и предназначенный для копирования уже доступного файла в указанное устройство или на TMPFS;
- **checksum** — подсчитывает контрольную сумму ранее загруженного или указанного образа, это может быть не только SHA-256, но и любой другой хэш;
- **iso9660** — монтирует устройство как CD/DVD-привод ISO-9660;
- **squashfs** — монтирует устройство или файл из каталога как сквош второй стадии загрузки (файл, записанный в формате файловой системы squashfs);
- **liveboot** — многофункциональный шаг финальной стадии загрузки, создающий оверлей LiveCD над read-only rootfs;
- **overlayroot** — обеспечивает простой синтаксис для закрытия на запись обычной rootfs.

Все «шаги» **bootchain** и **altboot** внутри initramfs являются скриптами в /lib/bootchain.

7. Параметры загрузки altboot

altboot получает параметры загрузки через /proc/cmdline так же, как это делал пропагатор и поддерживает их аналогичным образом. В **altboot** нет аналога параметра **propagator-debug**, поскольку в самом make-initrd есть параметры **rdshell** и **stop=runinit** для получения аварийной консоли до перехода в stage2. В **altboot** нет аналога для параметра **testing**, но есть возможность перенаправлять отладочный вывод, в том числе, на **/dev/console** или **/dev/ttyprintk**. Все остальные параметры так или иначе поддерживаются:

- **automatic** — основной накопитель *аргументов пропагатора и altboot*;
- **hash** — хэш-сумма сквоша второй стадии загрузки, посчитанная по алгоритму SHA-256 (в altboot можно использовать и другие варианты хэша с некоторыми синтаксическими добавками);
- **live** — (флаг) помечает ISO-образ как LiveCD, что предписывает всегда использовать режим экономии памяти;
- **live_rw** — (флаг) предписывает использовать *раздел для хранения сеансов LiveCD/Rescue*;
- **lowmem** — (флаг) включает режим экономии памяти, монтируя образ непосредственно с носителя без предварительной загрузки в ОЗУ;
- **profile** — профиль сетевой загрузки с поддержкой *read-only слоёв LiveCD* (в altboot по умолчанию profile=default и может работать не только с NFS);
- `propagator-debug` — в пропагаторе давал промежуточный shell перед переходом из stage1 в stage2, в make-initd есть свои аналоги, реализующие данный функционал, поэтому в altboot нереализован;
- **ramdisk_size** — (целое, ядерный) число килобайт, выделяемых ядром для устройств /dev/ram<N>, в один из которых будет загружаться сквош второй стадии (в altboot этот параметр стал необязательным);
- **rescue** — (флаг) помечает ISO-образ как Rescue LiveCD, что влияет только на вывод финальных сообщений;
- **stagename** — название файла сквоша с rootfs второй стадии загрузки, по факту только: «altinst», «rescue» и «live»;
- `testing` — «мёртвый» параметр, реально ни на что не влиял, поэтому в altboot нереализован, в последних версиях пропагатора перенаправлял диагностику на /dev/ttyprintk;
- **updatemodules** — в пропагаторе ранее использовался совместно с дискетами, пока они не перестали поддерживаться, в altboot используется для обновления «на лету» файловой системы в stage1, что в частности позволяет подгрузить недостающие модули ядра и правила udev.

7.1. Новые параметры bootchain и altboot

Помимо заимствованных «пропагаторных» параметров введено несколько новых. Подробнее нижеперечисленные параметры описаны в соответствующих разделах.

- **bc_debug** — (флаг) включает режим расширенной диагностики и копирует журнал в stage2;
- **bc_test** — название выполняемого тест-кейса, копирует его вместе с журналом в stage2;
- **bootchain** — список «шагов» bootchain через запятую (для совместимости с пропагатором д.б. «fg,altboot»);
- **cifsopts** — дополнительные опции монтирования SMB-ресурсов, используемые всеми «шагами» cifs;
- **console** — (ядерный) определяет консоль для вывода сообщений, в altboot влияет на настройку диалогов и вывод журнала;
- **curlopts** — (внешний) дополнительные опции curl, используемые всеми «шагами» download;
- **noaskuser** — (флаг) отключает диалоги ввода, как и некоторые значения параметра console;
- **nolines** — (флаг) запрещает вывод символов псевдографики в диалогах ввода/вывода;
- **nfsopts** — дополнительные опции монтирования NFS-каталогов, используемые всеми «шагами» nfs;
- **overlayroot** — обеспечивает простой синтаксис для закрытия на запись имеющейся rootfs.

См. также:

- https://www.altlinux.org/Propagator
- https://www.altlinux.org/Make-initrd
- https://www.altlinux.org/Make-initrd-propagator

8. Аргументы параметра automatic

Аргументы параметра **automatic=...** используются различными *методами загрузки*, они перечисляются в формате «параметр1:значение1,параметр2:значение2,...». Запись **automatic=0** заставит **altboot** вывести *диалог выбора метода загрузки* и эквивалентна *«automatic=method:auto»*. Приведённый ниже список изначально не исчерпывающий, и он может расширяться по мере создания новых *модулей*:

- **directory** — обычно монтируемый каталог или путь к ISO-образу;
- **disk** — имя диска с образом, например, nvme0n1;
- `domain` — название SMB-домена (*новое в altboot*);
- `fuid` — название проверочного файла в корне загрузочного раздела (*новое в altboot и в propagator, но в altboot используется ещё и для других целей внутренне*);
- `imgsize` — (целое) размер загружаемого образа (*новое в altboot*);
- **label** — метка тома файловой системы раздела с образом;
- **method** — название *метода загрузки* altboot (см. выше);
- `options` — дополнительные опции монтирования (*новое в altboot*);
- **overlays** — разное, в зависимости от выбранного *метода загрузки*;
- **partition** — раздел диска с образом, например, nvme0n1p2;
- **pass** — пароль авторизованного пользователя (FTP, SAMBA);
- **server** — имя или IP-адрес сервера (SAMBA, FTP, HTTP, NFS);
- `timeout` — (целое) таймаут в секундах для поиска носителя или ожидания отклика сервера (*новое в altboot*);
- `type` — тип загружаемого образа (*новое в altboot*);
- `url` — полный сетевой или локальный путь (*новое в altboot*);
- **user** — имя авторизованного пользователя (FTP, SAMBA);
- **uuid** — UUID файловой системы раздела с образом.

Исчерпывающий список поддерживаемых аргументов для каждого *метода загрузки* приведён в соответствующих разделах. См. также файлы в **/lib/altboot/automatic.d**, попадающие в initramfs.

altboot «понимает» все аргументы, которые передавались пропагатору, и позволяет через дополнительные модули бесконечно расширять данный список, при этом игнорируя аргументы, связанные с настройкой сети (такие, как *network*, *ip*, *dns*, *gateway*, *netmask*, *hostname* и *interface*), поскольку в make-initrd сеть конфигурируется через отдельную фичу «network».

Например, в пропагаторе таймауты в каждом методе загрузки задавались жёстко прямо в коде, но в **altboot** через /proc/cmdline дефолтный таймаут для любого метода загрузки можно переопределить:

```
root=bootchain bootchain=fg,altboot ip=dhcp4 \
    automatic=method:nfs,server:192.168.15.1,timeout:120,...
```

См. также:

- https://www.altlinux.org/Propagator
- https://github.com/osboot/make-initrd/tree/master/features/network
- https://www.altlinux.org/Bluescreen
- https://bugzilla.altlinux.org/48448

9. Конфигурирование bootchain и altboot

В дистрибутивах ОС Альт необходимое конфигурирование для создания универсальных загрузочных носителей уже *выполнено* в mkimage-profiles. Здесь представлена информация для разработчиков других дистрибутивов и пользователей **altboot**, желающих работать с ним в установленной Linux-системе на обычной rootfs, либо использующих *оверлеи LiveCD* для локальной и сетевой загрузки.

Можно поместить в образ initramfs свой файл с альтернативной цветовой палитрой и опциями диалогов: **/etc/dialogrc**, программа **dialog** использует этот файл по умолчанию. Для вывода сообщений об ошибках по умолчанию используется другой файл: **/etc/dialogrc.error**, при его наличии. Первый файл можно получить, запустив:

```
dialog --create-rc
```

И **altboot**, и **bootchain** настраиваются через один файл конфигурации, который также должен попасть в образ initramfs: **/etc/sysconfig/bootchain** (при запуске make-initrd переменной **$BOOTCHAIN_PATH** можно переопределить каталог, в котором находится конфигурация). В этом конфигурационном файле можно переопределить дефолты и задать параметры, которые изначально не определены. См. детали в примере bootchain-doc/samples/bootchain-alt.conf. Для добавления произвольных файлов используется директива в **/etc/initrd.mk**:

```
PUT_FILES += <файлы>
```

Про кофигурирование в mkimage-profiles смотрите features.in/initrd-bootchain/README. Далее будет приведено описание известных конфигурационных переменных.

См. также:

- Создание своего initrd.img с bootchain
- https://www.altlinux.org/Mkimage-profiles
- Пример конфигурации в пакете с документацией
- Дистрибутивный скрипт для создания initrd.img

9.1. Модуль make-initrd-bootchain-core

- **BC_DEBUG** — непустое значение, если включена расширенная отладка;
- **BC_LOG_VT** — если непустое значение, то порядковый номер TTY для вывода журнала;
- **BC_LOGFILE** — путь к файлу журнала или устройство для вывода в него сообщений отладки;
- **BC_DEVICE_TIMEOUT** — глобальный дефолтный таймаут для любых «шагов» altboot;
- **BC_FGVT_ACTIVATE** — если непустое значение, то через сколько секунд активировать интерактивный терминал.

9.2. Модуль make-initrd-bootchain-waitdev

- **WAITDEV_TIMEOUT** — глобальный дефолтный суммарный таймаут для всех «шагов» waitdev.

9.3. Модуль make-initrd-bootchain-altboot

- **OEM_WELCOME_TEXT** — используется в качестве заголовка верхнего уровня в диалогах altboot;
- **OEM_DISTRIBUTION** — используется в качестве названия дистрибутива в диалогах altboot;
- **OEM_CDROOT** — необязательный путь к корню ISO-образа внутри initramfs, в ОС Альт это /image;
- **OEM_LIVE_STORAGE** — метка тома *live_rw раздела*, если указана;
- **OEM_BAD_STORAGE** — метка тома, чтобы не использовать *live_rw раздел* на плохом или слишком медленном устройстве;
- **OEM_SETUP_STORAGE** — метка тома раздела для обновления initramfs «на лету»;
- **OEM_IMAGES_BASE** — куда в stage2 монтировать каталог с образами *слоёв LiveCD*;
- **OEM_OVERLAYS_DIR** — куда в stage2 монтировать сами *слои LiveCD*;
- **OEM_URL_NETINST** — значение по умолчанию компоненты directory для загрузки методами http и ftp;
- **OEM_NFS_NETINST** — значение по умолчанию компоненты directory для загрузки методом nfs;
- **OEM_CIFS_NETINST** — значение по умолчанию компоненты directory для загрузки методом cifs.

9.4. Модуль make-initrd-bootchain-waitnet

- **OEM_SRV_NETINST** — значение по умолчанию компоненты server для загрузки методами http и ftp, определяет IP-адрес или имя сервера сетевой загрузки.

10. Диагностика

10.1. Журналирование

Демон **chaind** ведёт журнал **/var/log/chaind.log**, доступный в stage1 на tty3 (по нажатию **Alt-F3**), что определяется дефолтными значениями **BC_LOG_VT** и **BC_LOGFILE** в *файле конфигурации*, а также перебивается параметром make-initrd **rdlog=console** или **rdlog=printk** в рантайме, что предписывает выводить журнал на **/dev/console** или на **/dev/ttyprintk**. Если демон запущен в *режиме совместимости с pipeline*, то журнал будет назван иначе: **/var/log/pipelined.log**.

Расширенная диагностика включается через /proc/cmdline параметром **bc_debug**. В этом случае в журнал попадает значительно больше информации и он копируется по тому же пути в stage2 после отработки последнего «шага», а если в новой rootfs нет **/var/log**, тогда копируется в **/dev/bootchain/**.

10.2. Отладка

Аналогом **propagator-debug** в самом make-initrd является параметр **rdshell** либо **stop=runinit**, для их использования образ initramfs должен быть собран с соответствующей фичей «rdshell».

В параметре **stop** нужно указывать сервис, перед запуском которого загрузка должна остановиться. При этом будет предложено запустить сервис и продолжить загрузку (**Y**), не запускать сервис и продолжить загрузку (**N**) и запустить rdshell (**S**). Необходимо запустить rdshell, нажав **S**. Для выхода из shell нужно нажать **Ctrl+D** или выполнить команду **exit**. Список сервисов находится внутри initramfs в каталоге **/etc/rc.d/init.d/**.

При отладке будет мешать интерактивная заставка **plymouth**, поэтому нужно также указывать параметр **nosplash**.

10.3. Тестирование

Пакет **make-initrd-bootchain-doc** содержит набор для тестирования с документацией, демон **chaind** поддерживает параметр **bc_test**=<NAME>, предписывающий скопировать журнал и название успешно пройденного теста <NAME> в каталог **/var/log** rootfs второй стадии загрузки, а если в новой rootfs нет **/var/log**, тогда эта информация копируется в **/dev/bootchain/**.

11. Совместимость с pipeline

Демон **chaind** может работать в одном из двух режимов: «родном» и в «режиме совместимости с **pipeline**». Для «родного» режима предусмотрен синтаксис:

```
root=bootchain bootchain=...
```

«Режим совместимости с **pipeline**» активируется при записи в параметрах загрузки:

```
root=pipeline pipeline=...
```

Table 1. Сходства и различия между двумя режимами перечислены в таблице:

Действие или значение	режим «pipeline»	режим «bootchain»
Завершение и переход в stage2	шаг может выйти с кодом 2	шаг должен вызвать break_bc_loop()
Шаг вызвал break_bc_loop()	демон завершается	демон завершается
Шаг завершился с кодом 0	переход к следующему шагу	переход к следующему шагу
Шаг завершился с кодом 2	демон завершается	обработка ошибки
Шаг завершился с кодом >0	обработка ошибки	обработка ошибки
Значение $mntdir	/dev/pipeline	/dev/bootchain
Журнал отладки	/var/log/pipelined.log	/var/log/chaind.log

Обработка ошибки демоном **chaind** выполняется следующим образом. По умолчанию разрешено 4 раза повторять зафейлившийся шаг с паузой в 2 секунды. Данный режим отключается шагом «**noretry**» и включается обратно шагом «**retry**». При отключении повторов демон выходит немедленно с соответствующей записью в журнале отладки. **chaind** не допускает возвращения скриптами «шагов» ненулевых кодов, но не знает, какие «шаги» теоретически могут фейлиться, а какие нет.

В «режиме совместимости с **pipeline**» демон делает исключение для обработки возвращаемого кода 2, рассматривая его как команду немедленного прекращения работы с соответствующей записью в журнале отладки. В любом режиме прекращение работы демона выполняется сразу после выхода из скрипта «шага», вызвавшего функцию **break_bc_loop()**. Значения по умолчанию для $mntdir и пути к журналу отладки можно переопределить в конфиге **/etc/sysconfig/bootchain**.

В последних версиях make-initrd оригинальная фича «**pipeline**» перестала быть демоном, поскольку в самом make-initrd произошли архитектурные изменения с момента форка «bootchain». Также в код make-initrd и в код фичи «pipeline» перетекли некоторые вещи, в частности, описанная выше обработка ошибок теперь и в «pipeline» стала более управляемой. Но современный «bootchain» всё больше отличается от современного «pipeline», здесь требуется периодическая синхронизация кода либо возвращение «bootchain» в апстрим make-initrd.

12. Совместимость с propagator и init-bottom

«Шаги» **altboot** могут работать в одном из двух режимов: в «обычном» режиме, как «шаги» **bootchain** (но не **pipeline**), и в «режиме совместимости с пропагатором». Внутри исходников «шагов» это управляется переменной **$ALBOOT_OLDROOT**. В «обычном» режиме «шаги» могут повторяться в цепочке, они конфигурируются по отдельности через /proc/cmdline, ничего не экспортируют в stage2, монтируют каталоги в стиле **pipeline**, по умолчанию через /dev/bootchain/dst/step<N>, например:

```
root=bootchain bootchain=fg,localdev,copyfile,checksum,squashfs,liveboot,rootfs \
  altboot_localdev=method=cdrom;uuid=2021-05-30-12-55-00-00;directory=/rescue \
  altboot_copyfile=dst=RD;src=/rescue ramdisk_size=497773 \
  altboot_checksum=613bf1a12803b448682d0c6d628400d2a52f38b61e6815af34f364cf71ad65ec \
  altboot_liveboot=stagename=rescue;method=cdrom;flags=live_ro,rescue,live_rw
```

Режим «совместимости с пропагатором» активируется «шагом» с именем «**altboot**». Он сам определяет и конфигурирует через внутренний конфиг все последующие «шаги», ориентируясь на параметры, предназначенные пропагатору и скрипту **init-bootom**. В этом режиме **$ALBOOT_OLDROOT=1**, каждый «шаг» **altboot** может быть пройден только единожды, в stage2 экспортируются определённые данные, в зависимости от выбранного *метода загрузки*, каталог **/dev/bootchain**, по возможности, очищается до перехода в stage2, монтирование выполняется «внахлёст» через каталоги **/root** (**$rootmnt**) и **/image** (**$OEM_CDROOT**):

```
root=bootchain bootchain=fg,altboot automatic=... \
  [lowmem] [ramdisk_size=...] [stagename={altinst|live|rescue}]
```

Для полной совместимости с пропагатором в *конфигурации* следует прописать **OEM_DEFAULT_STAGE2=altinst**, этот дефолт был в пропагаторе, в коде **altboot** его нет. Иначе образ, собранный без параметра **stagename=...** загрузиться не сможет. В mkimage-profiles такой дефолт уже определён.

13. Интерактивный режим работы

Некоторым шагам **altboot** требуется интерактивное взаимодействие с пользователем. Перед такими шагами следует включать шаг **fg**, даже если никаких диалогов в результате работы не будет выведено. По умолчанию интерактивная консоль находится на втором терминале (**tty2**). В **bootchain** авто-переключение на интерактивную консоль отложено на (BC_FGVT_ACTIVATE=7)+1, т.е. по умолчанию на 8 секунд без **bc_debug** или на 2 секунды с включённым **bc_debug**. Переключение выполняется немедленно по нажатию **Alt-F2** или автоматически: а) при возникновении диалога с ошибкой; б) при появлении диалога ввода; в) по истечению заданного таймаута. При автоматической активации интерактивной консоли **rootdelay** и прогресс бар **plymouth** временно отключаются.

Есть два параметра загрузки для управления интерактивным взаимодействием:

- **nolines** — Запрещает в диалогах вывод линий символами псевдографики. Полезно, если в системе нет шрифтов для их поддержки. В этом случае линии выводятся символами из обычного набора ASCII.
- **noaskuser** — Запрещает диалоги ввода. Обычные сообщения и сообщения об ошибках также являются диалогами ввода, поскольку требуют реакции пользователя. Если за консолью нет пользователя, то не стоит рассчитывать на его реакцию. Использование данного параметра приведёт к фатальной ошибке в коде, который попытается открыть диалог ввода. Поэтому **altboot** написан таким образом, чтобы учитывать данный параметр и если ввод запрещён, то ограничиваться, по возможности, выводом сообщений в журнал.

Запретить вывод таких виджетов, как **gauge** (прогресс бар) или основанного на нём **ponder** (бесконечный процесс), нельзя, но если загрузка происходит очень быстро, то по умолчанию этих диалогов пользователь даже не заметит, так как переключение на интерактивную консоль отложено обычно на 8 секунд, в которые укладывается даже самая медленная загрузка в stage1.

См. также:

- https://bugzilla.altlinux.org/30472
- https://bugzilla.altlinux.org/41521
- https://bugzilla.altlinux.org/41097
- https://bugzilla.altlinux.org/41096
- https://bugzilla.altlinux.org/48448

14. Постоянное хранилище сеансов LiveCD

Поддержка данного режима требует «локального» метода загрузки **disk** или **cdrom**, а также пакета **make-initrd-bootchain-liverw**. К параметрам загрузки должен быть добавлен пропагаторный параметр «**live_rw**», если загрузка выполняется в *режиме совместимости с пропагатором*. Гибридный ISO-образ дистрибутива записывается на USB-флеш, MMC-карту или иной носитель, на котором должно оставаться не менее 1Гб свободного места.

При первой загрузке с параметром «**live_rw**» методом **cdrom** весь оставшийся диск будет использован под раздел с меткой тома, определяемый конфигурационной переменной **$OEM_LIVE_STORAGE**, в случае ОС Альт это «**alt-live-storage**». Все изменения будут записываться на созданный раздел и сохраняться между перезагрузками. При последующих загрузках данный раздел будет подключаться как верхний слой R/W автоматически. Для метода **disk** поведение аналогично, с той лишь разницей, что используется свободное место на том диске, где находится загрузочный раздел (в пропагаторе этого не было).

Следует иметь ввиду, что постоянное хранилище сеансов с технической точки зрения является компромиссом между скоростью и надёжностью. Опции форматирования и монтирования носителя ориентированы на средне-скоростные USB-носители. Данные могут потеряться при сбоях питания или «зависании». Постоянная запись на относительно медленные USB-носители не является эффективной, ей следует предпочесть другой вариант работы, описанный в *следующем разделе*.

Пример использования:

```
root=bootchain bootchain=fg,altboot live stagename=live \
  automatic=method:cdrom,uuid:2021-05-30-18-28-59-00 \
  ramdisk_size=1081725 showopts live_rw quiet splash
```

См. также:

- https://bugzilla.altlinux.org/32562
- https://www.altlinux.org/Make-initrd-propagator
- https://www.altlinux.org/Remount_rw

15. Read-only слои LiveCD

При сетевой загрузке с сервера **NFS** или с сервера **SAMBA**, а также при локальной загрузке методами **disk** или **cdrom** с универсальных образов **LiveCD** или **Rescue** допускается работа со слоями LiveCD, доступными только на чтение. При этом самым нижним слоем для чтения становится сквош второй стадии (корневой системы) на исходном ISO-образе, поверх него накладываются образы, создаваемые системным администратором. Самый верхний R/W-слой создаётся для записи на tmpfs или выделенный раздел (только при локальной загрузке при использовании параметра **live_rw**, см. *предыдущий раздел*).

Из полученного «слоёного пирога» формируется rootfs, в которую происходит загрузка, что даёт возможность администраторам вносить необходимые изменения в настройки исходного диска и предоставлять это пользователям сетевых бездисковых классов с доступом только на чтение, причём такой способ централизованного управления настройками и загрузки с использованием read-only сквошей значительно эффективнее механизма работы с *постоянным хранилищем сеансов*, тем более, если говорить о локальной загрузке с USB-флеш и подобных относительно медленных накопителей.

Table 2. Ниже представлена таблица с принятыми умолчаниями в ОС Альт:

Метод	Имя переменной	Путь к профилям с образами	Относительно чего
cdrom	-	/	См. параметр overlays
disk	-	/	См. параметр overlays
cifs	$OEM_CIFS_NETINST	/netinst/overlays-live	На том же сервере SAMBA
nfs	$OEM_NFS_NETINST	/srv/public/netinst/overlays-live	На том же сервере NFS

Имя профиля — это относительный путь к каталогу, задаваемый пропагаторным параметром **profile**, например, «rescue/overlays». При загрузке с использованием слоёв (оверлеев) LiveCD, все файлы с окончаниями **.squashfs** и **.iso** в каталоге профиля рассматриваются как оверлеи и монтируются слоями друг над другом в алфавитном порядке. Так что в каталоге на сервере (и в соответствующем локальном каталоге) можно разместить не один, а несколько разных профилей загрузки, а в каждом из них — собственный набор LiveCD-оверлеев. Путь к каталогу с профилями для сетевых *методов загрузки* определяется параметром **overlays**, но его можно опустить, и тогда используются дефолты из таблицы. Для локальных *методов загрузки* **overlays** указывает на раздел с оверлеями.

Особенности реализации поиска профиля зависят от выбранного *метода загрузки*:

- **disk** или **cdrom**: если аргумент **overlays=local_profile**, профиль будет искаться на том же разделе, с которого выполняется загрузка системы, иначе аргумент **overlays** рассматривается как метка тома устройства (LABEL), на котором будет искаться профиль с LiveCD-оверлеями;
- **cifs**: используется путь, определённый аргументом **overlays**, если его нет, то последняя часть в пути к ISO-образу или каталогу, указанного при загрузке в **automatic=...** или на шаге **cifs**, заменяется на «overlays-live», а если не удаётся определить путь таким способом, то используется формула: «$OEM_CIFS_NETINST/overlays-live», при этом, первая часть в полученном пути определяет название ресурса SAMBA;

- **nfs**: используется путь, определённый аргументом **overlays**, если его нет, то последняя часть в пути к ISO-образу или каталогу, указанного при загрузке в **automatic=...** или на шаге **nfs**, заменяется на «overlays-live», в том числе, на вычисляемое значение может влиять явно передаваемый шагу **liveboot** путь через аргумент **directory**, а если не удаётся определить путь таким способом, то используется формула: «$OEM_NFS_NETINST/overlays-live».

Проще говоря, на всё есть умолчания, перебиваемые руками через /proc/cmdline.

Все *методы загрузки* **altboot** используют общие фрагменты кода для работы с read-only слоями LiveCD, небольшие отличия обеспечиваются скриптовыми «хуками» *методов загрузки*. Вплоть до выхода продуктов на «одинадцатой платформе» в ОС Альт возможность работы с read-only слоями LiveCD обеспечивалась только для *метода загрузки* **nfs**. При загрузке средствами **propagator** с **init-bottom**, если имя профиля не указывалось, то соответствующий уровень каталога не использовался. При переходе на **altboot** следует учитывать, что если **profile** не указан в /proc/cmdline, его значение по умолчанию — «default», соответствующий подкаталог должен быть создан на сервере.

При локальной загрузке с параметром **live_rw** (см. *предыдущий раздел*) можно вытащить содержимое самого верхнего R/W-слоя из каталога /**.rw**, отфильтровать ненужное и упаковать в сквош командой **mksquashfs**, который потом использовать в качестве read-only оверлея. Кроме того, при локальной загрузке допускается одновременная работа и с несколькими read-only оверлеями, и с *постоянным хранилищем сеансов LiveCD*.

Примеры использования:

```
root=bootchain bootchain=fg,altboot rescue \
  automatic=method:cdrom,uuid:2021-05-31-01-09-58-00,overlays:SLICES \
  stagename=rescue ramdisk_size=497773 profile=rescue/overlays
```

Здесь локальная загрузка с ALT Rescue методом **cdrom** в режиме совместимости с пропагатором. Файлы с оверлеями находятся на отдельном разделе с меткой тома «SLICES» в подкаталоге /rescue/overlays.

```
root=bootchain bootchain=fg,altboot live lowmem ip=dhcp4 \
  automatic=method:nfs,network:dhcp,server:192.168.15.10 \
  stagename=live ramdisk_size=1107233 profile=alt-ws11
```

Здесь сетевая загрузка с LiveCD методом **nfs** в режиме совместимости с пропагатором. Файлы с оверлеями находятся на том же NFS сервере в подкаталоге /srv/public/netinst/overlays-live/alt-ws11.

См. также:

- https://altlinux.org/Netboot
- https://www.altlinux.org/NetInstall
- https://www.altlinux.org/Make-initrd-propagator
- https://www.altlinux.org/Remount_rw

16. Шаги bootchain

В основном, «шаги» **bootchain** — это относительно небольшие скрипты, попадающие в образ initramfs в каталог **/lib/bootchain/**. Но также среди них есть и внутренние псевдо-шаги, реализуемые непосредственно демоном **chaind**.

16.1. debug

Входит в подпакет **make-initrd-bootchain-core**, является расширением **pipeline**, в дополнительных параметрах не нуждается. Указывать данный «шаг» в синтаксической цепочке загрузки не следует. Он автоматически запускается при использовании параметра **bc_debug** в /proc/cmdline перед каждым «шагом» **bootchain** и перед переходом в stage2, оказывая влияние только на содержимое журнала **/var/log/chaind.log** (или **/var/log/pipelined.log**).

16.2. fg

Входит в подпакет **make-initrd-bootchain-core**, является расширением **pipeline**, в дополнительных параметрах не нуждается. Это внутренний псевдо-шаг главного цикла демона **chaind**, обеспечивающий его перевод в интерактивный режим при наличии в initramfs фичи **bootchain-interactive**. Самому **bootchain** интерактивность не требуется, но в ней могут нуждаться некоторые «шаги», такие как **altboot**. См. детали в разделе «*Интерактивный режим работы*».

Пример использования:

```
root=bootchain bootchain=fg,altboot
```

16.3. getimage

При форке **bootchain** данный «шаг» перешёл из **pipeline** без изменений, но был выделен в отдельный подпакет **make-initrd-bootchain-getimage**. Скачивает по сети и монтирует указанный образ. Можно использовать на быстрых интернет-каналах или в сценариях автоматического тестирования или с небольшими образами для загрузки по протоколам HTTP/FTP утилитой wget, поскольку визуализации процесса нет, а процесс загрузки ограничен 180 секундами. В остальном это скорее «proof of concept».

Параметры загрузки:

- **getimage** — URL скачиваемого образа.

Пример использования:

```
root=bootchain bootchain=getimage,mountfs,overlayfs,rootfs ip=dhcp4 \
  getimage=http://ftp.altlinux.org/pub/people/mike/iso/misc/vi-20140918-i586.iso \
  mountfs=rescue
```

16.4. mountfs

Входит в подпакет **make-initrd-bootchain-core**, перешёл из **pipeline** без изменений при форке **bootchain**. Позволяет смонтировать то, что было получено в результате работы предыдущих «шагов».

Параметры загрузки:

- **mountfs** — имя файла, либо **dev**, либо **DEVNAME**.

16.5. noop

Входит в подпакет **make-initrd-bootchain-core**, является расширением **pipeline**, в дополнительных параметрах не нуждается. Это внутренний псевдо-шаг главного цикла демона **chaind**, не выполняющий никаких действий, и предназначенный для отрыва результата предыдущего «шага» от входа следующего «шага», что может быть полезно, например, когда мы не хотим, чтобы результаты **waitdev** были использованы в следующем «шаге» **localdev**.

16.6. noretry

Входит в подпакет **make-initrd-bootchain-core**, является расширением **pipeline**, в дополнительных параметрах не нуждается. Это внутренний псевдо-шаг главного цикла демона **chaind**, запрещающий следующим «шагам» завершаться с ненулевым кодом возврата, что приведёт к немедленному завершению работы демона в случае сбоя в скрипте любого следующего «шага». По умолчанию «шагам» разрешено фейлиться, демон будет перезапускать их повторно несколько раз.

16.7. overlayfs

Входит в подпакет **make-initrd-bootchain-core**, перешёл из **pipeline** без изменений при форке **bootchain**. Создаёт оверлей над read-only rootfs при LiveCD загрузке.

Параметры загрузки:

- **overlayfs** — необязательный параметр, может содержать перечисленные через запятую названия пройденных «шагов» в формате *step<N>*, в каталоги которых смонтированы отдельные read-only слои, если на предыдущем «шаге» не был смонтирован каталог с read-only rootfs.

16.8. ping

После форка **bootchain** данный «шаг» был скопирован из **pipeline** без изменений, но был выделен в отдельный подпакет **make-initrd-bootchain-getimage**. В самом make-initrd данный «шаг» появился сравнительно недавно (конец 2023 года). Позволяет проверить доступность удалённого хоста. Имеет смысл использовать его перед **getimage**. По умолчанию данный «шаг» ждёт неограниченное время, пока хост не станет доступен. Если параметр не указан, проверяется дефолтный шлюз.

Параметры загрузки:

- **ping** — необязательные опции работы «шага», перечисляемые через двоеточие.

Возможные значения опций:

- **v4** — использовать 4-ю версию протокола IP;
- **v6** — использовать 6-ю версию протокола IP;
- **waitfor** — предписывает пинговать без ограничений по числу попыток;
- **iter**=<N> — предписывает пинговать не более указанного числа попыток;
- **%gateway** — предписывает пинговать шлюз по умолчанию, который будет определён автоматически по таблице маршрутизации;
- *другие значения* — рассматриваются как непосредственный адрес или имя пингуемого хоста.

Пример использования:

```
root=bootchain bootchain=noretry,ping,getimage,mountfs,overlayfs,rootfs ip=dhcp4 \
  ping=v4:iter=30:ftp.altlinux.org \
  getimage=http://ftp.altlinux.org/pub/people/mike/iso/misc/vi-20140918-i586.iso \
  mountfs=rescue
```

16.9. retry

Входит в подпакет **make-initrd-bootchain-core**, является расширением **pipeline**, в дополнительных параметрах не нуждается. Это внутренний псевдо-шаг главного цикла демона **chaind**, разрешающий всем последующим «шагам» завершаться с ненулевым кодом возврата, что приведёт к их пятикратному запуску. Такой режим работы демона действует по умолчанию.

16.10. rootfs

Входит в подпакет **make-initrd-bootchain-core**, перешёл из **pipeline** с незначительными изменениями при форке **bootchain**. Переносит в rootmnt=/root результаты предыдущего «шага» и сообщает демону **chaind** о том, что это последний «шаг» и можно переходить в stage2. В дополнительных параметрах не нуждается.

16.11. wait-resume

Входит в подпакет **make-initrd-bootchain-core**, был скопирован из **pipeline** без изменений после форка **bootchain**. В самом make-initrd данный «шаг» появился уже после основного форка (осень 2022 года). Иногда загрузочная цепочка успевает отработать быстрее, чем происходит возобновление работы после «спящего режима», в результате чего выполняется новая загрузка. Данный «шаг» позволяет приостановить нормальный процесс загрузки до того, как возобновление завершится неудачей. Имеет смысл использовать на обычной rootfs, а не с универсальными «живыми» носителями.

16.12. waitdev

Входит в подпакет **make-initrd-bootchain-waitdev**, является расширением одноимённого «шага» **pipeline**. Изначально каждый «шаг» **waitdev** предписывает дожидаться устройства по заданной в одноимённом параметре спецификации, чей формат аналогичен параметру **root=...** make-initrd. Расширение исходного кода **pipeline** здесь сводится к трём изменениям:

- Допускается префикс «CDROM:» в спецификации либо использование этого префикса вместо спецификации. В первом случае waitdev будет дожидаться первого попавшегося устройства типа CD/DVD-ROM или имеющего формат файловой системы ISO-9660. Во втором случае префикс дополняет спецификацию, требуя, чтобы устройство было CD/DVD-ROM'ом или имело формат ISO-9660.
- Имя найденного устройства записывается ещё и в файл DEVNAME в каталоге результата текущего «шага», что позволяет видеть более понятное администратору имя устройства в /proc/mounts при его монтировании последующими «шагами».
- В /proc/cmdline можно прописать параметр **waitdev_timeout**. Он определяет суммарный таймаут для всех «шагов» waitdev, что позволит использовать другие «шаги» в качестве fallback, если указанные устройства не будут обнаружены в заданное время. Так, **altboot**'овский «шаг» **localdev** сначала пытается использовать результаты предыдущего «шага», но если их нет, выводится диалог выбора устройства, т.е. данное расширение обеспечивает переход от ожидания к сканированию или выбору устройств.

Параметры загрузки:

- **waitdev** — определяет спецификацию устройства для каждого «шага» waitdev в стиле root=... make-initrd;
- **waitdev_timeout** — определяет общий таймаут для всех «шагов» waitdev чтобы использовать fallback;

Примеры использования:

```
root=bootchain bootchain=waitdev,mountfs,mountfs,overlayfs,rootfs \
  waitdev=CDROM:LABEL=ALT_regular-rescue/x86_64 \
  mountfs=dev \
  mountfs=rescue

root=bootchain bootchain=waitdev,waitdev,fg,altboot \
  waitdev=LABEL=alt-live-storage \
  waitdev=CDROM: \
  automatic=method:cdrom

root=bootchain bootchain=waitdev,fg,download,... ip=dhcp4 \
  waitdev=MODEL=SAMSUNG_SSD_960_EVO_250GB \
  altboot_download=method=url;url=http://192.168.15.1/images/alt-ws11.img
```

На момент написания настоящей документации спецификация определения дисков в стиле **MODEL=...** не поддерживается make-initrd, так что третий пример здесь скорее гипотетический.

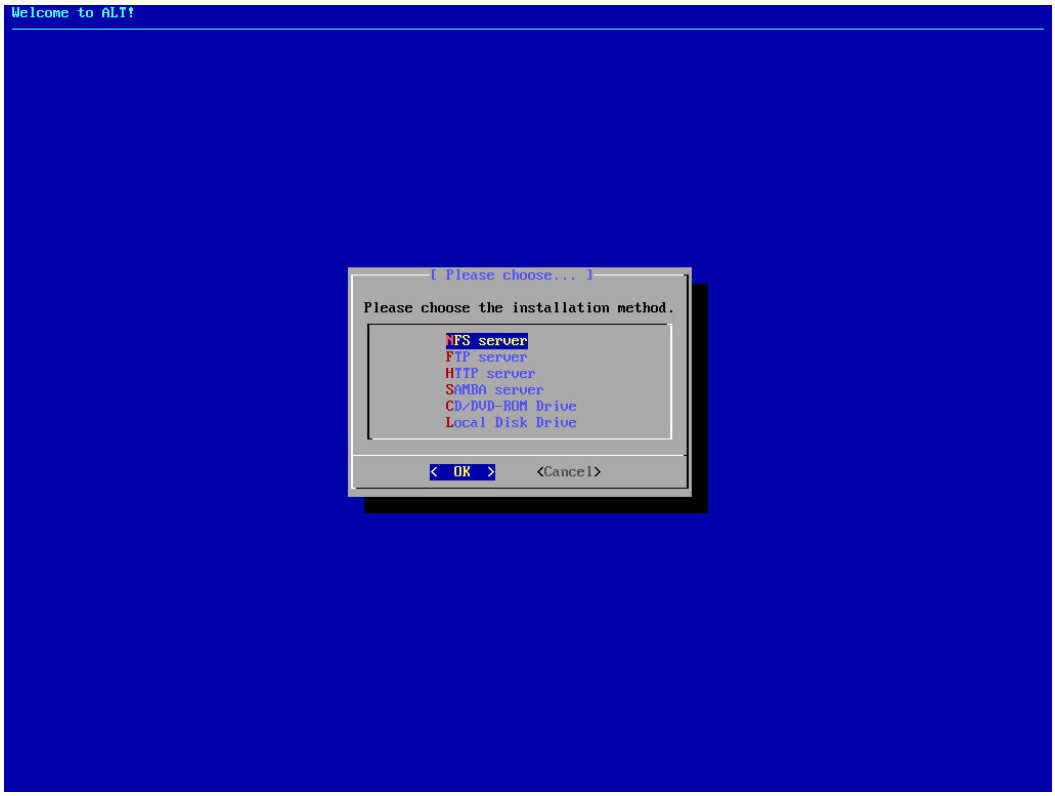
17. Шаги altboot

«Шаги» **altboot** являются отельными действиями, выполнявшимися когда-то пропагатором, а теперь скриптами, попадающими в образ initramfs в каталог **/lib/bootchain/**. Большинство «шагов» **altboot** требуют предварительного переключения демона **chaind** в *Интерактивный режим работы*, т.е. в /proc/cmdline слева от этих шагов должен указываться шаг **fg** (см. *выше*).

17.1. altboot

Данный «шаг» входит в подпакет **make-initrd-bootchain-altboot**, определяет *метод загрузки* и следующие «шаги», транслирует *аргументы* **automatic=...** и другие известные *параметры пропагатора* в аргументы следующих «шагов», в собственных параметрах не нуждается. Часть его кода вынесена в виде «хуков» в другие *подпакеты*.

Без параметра **automatic** ничего работать не будет, **altboot** на него ориентируется. В процессе работы может вывести диалог выбора метода загрузки с использованием виджета **choice** и создать файл **/.initrd/bootchain/altboot.conf** с уже разобранной конфигурацией, используемой следующими «шагами».



Пример использования:

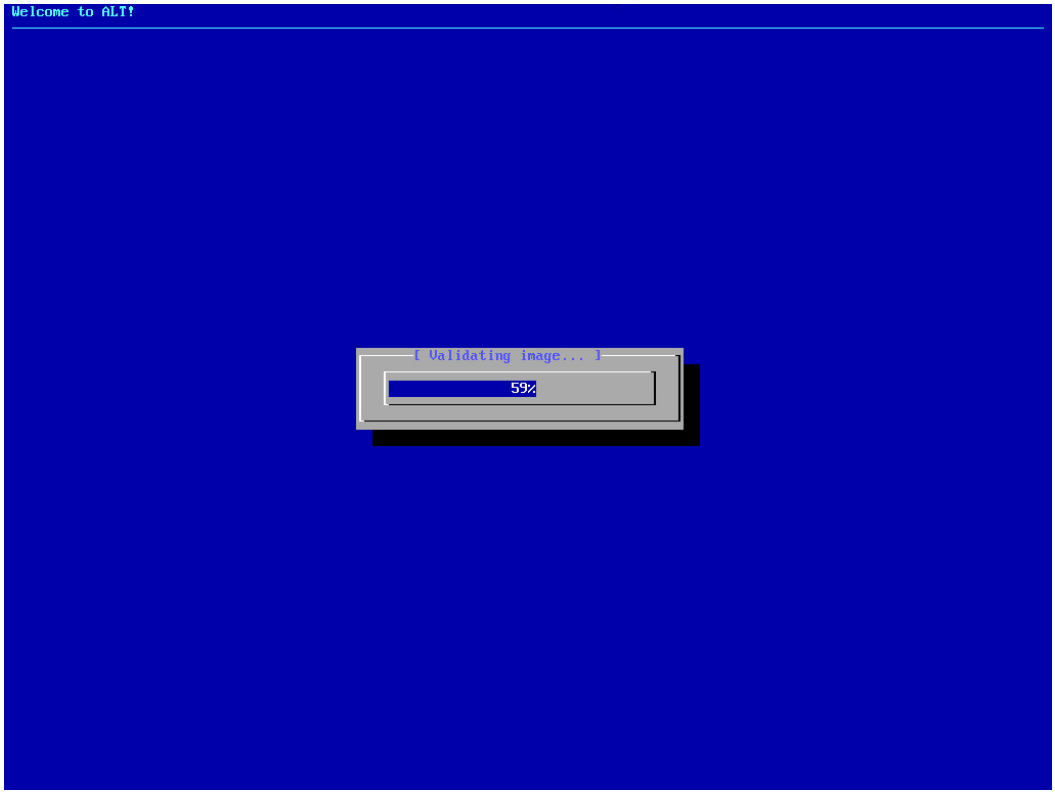
```
root=bootchain bootchain=fg,altboot lowmem stagename=rescue \
  automatic=method:cdrom,label:ALT_regular-rescue/x86_64
```

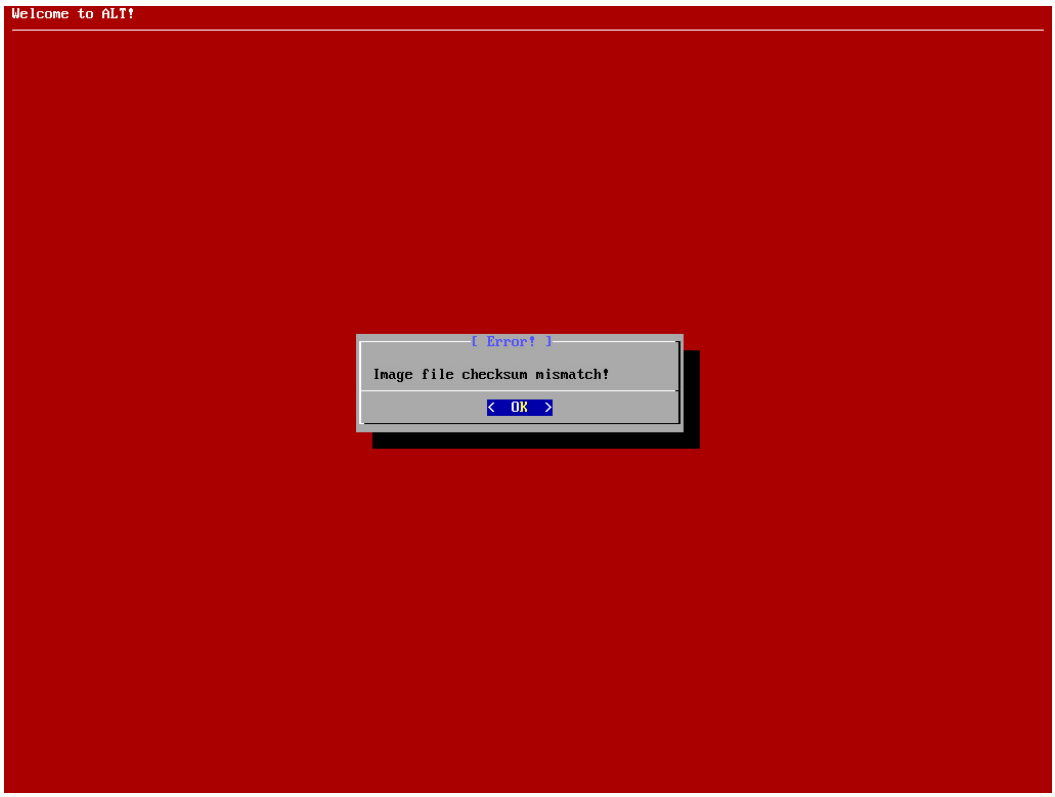
См. также:

- https://bugzilla.altlinux.org/40267

17.2. checksum

«Шаг» входит в подпакет **make-initrd-bootchain-altboot**, подсчитывает контрольную сумму ранее загруженного или указанного, это может быть не только SHA-256, но и любой другой хэш. По умолчанию используется программа **sha256sum**. Процесс подсчёта контрольной суммы визуализируется виджетом **gauge**.





Параметры загрузки:

altboot_checksum — единственный обязательный параметр, определяющий контрольную сумму и (опционально) хэш-программу для подсчёта этой суммы, а также имя файла в смонтированном на предыдущем «шаге» каталоге. Формат этого параметра таков:

- altboot_checksum=<**HASH**> -либо-
- altboot_checksum=[<hashprog>]:<**HASH**> -либо-
- altboot_checksum=[<hashprog>]:<**HASH**>:<**filename**>

Следует иметь ввиду, что если имя файла не передано через единственный параметр, то на обязательном предыдущем «шаге» должно быть создано устройство, определяемое файлом **DEVNAME** либо **dev**, а также размер образа в байтах, загруженного в это устройство, определяемое файлом **FILESIZE**. Данному поведению удовлетворяют шаги **download** и **copyfile**, которые в случае успеха монтируют загруженный сквош или ISO-образ через lomount().

Пропагатор подсчитывал контрольную сумму при скачивании файла. В **altboot** фактически существует два сценария подсчёта контрольной суммы: загрузить большой образ в память и посчитать ЛИБО посчитать сначала «на месте», например, на ранее смонтированном каталоге, а потом делать с ним что-либо дальше, что приводит к двойному чтению образа. Первый способ эффективнее, но при включенном lowmem или live, **altboot** в *режиме совместимости с пропагатором* использует второй сценарий.

Примеры использования:

```
root=bootchain bootchain=fg,download,checksum,... \
  altboot_checksum=01ba4719c80b6fe911b091a7c05124b64eeece964e09c058ef8f9805daca546b

root=bootchain bootchain=fg,download,checksum,... \
  altboot_checksum=md5sum:68b329da9893e34099c7d8ad5cb9c940:/live
```

См. также:

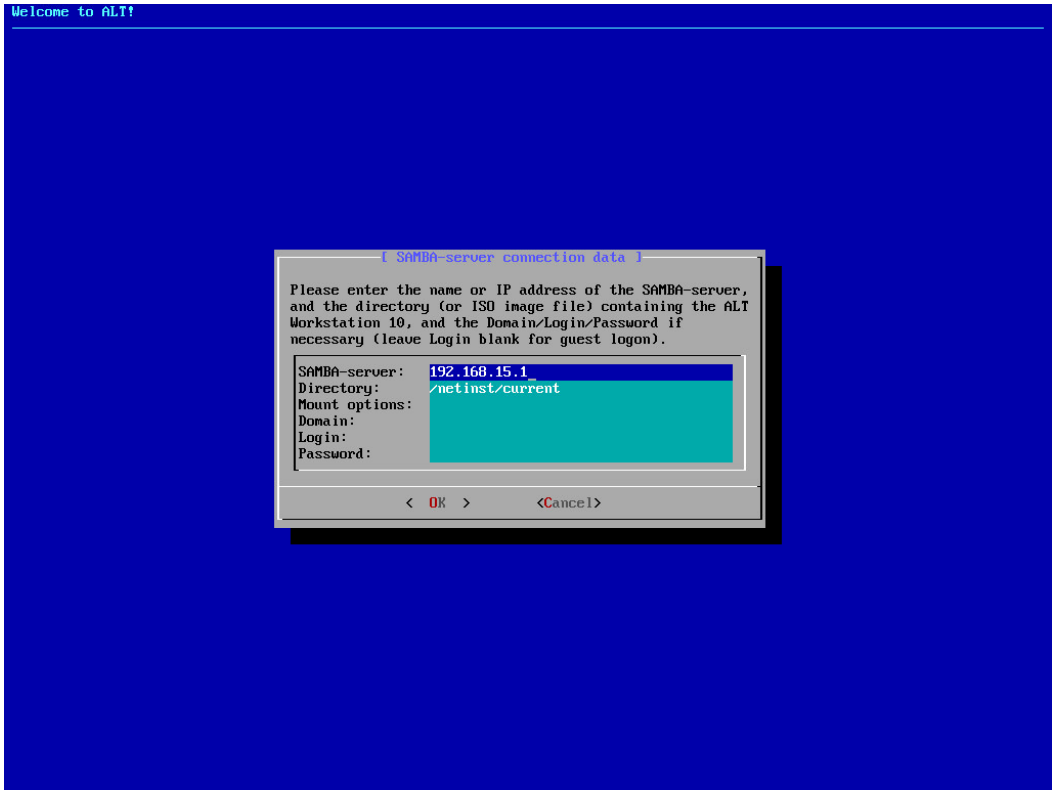
- https://bugzilla.altlinux.org/30745

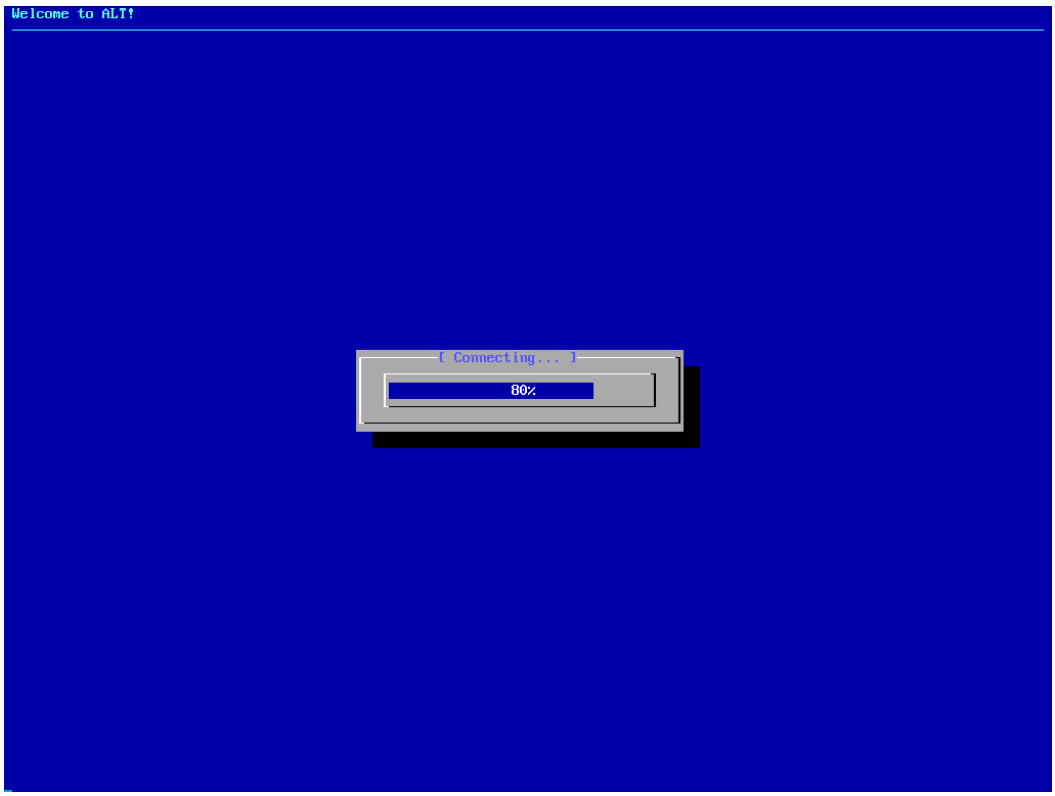
17.3. cifs

Входит в подпакет **make-initrd-bootchain-cifs**, обеспечивает *метод загрузки* с сервера SAMBA по протоколу CIFS и поддержку *read-only слоёв LiveCD* (сквошей), размещаемых также на сервере SAMBA. В процессе работы может вывести диалог ввода данных для соединения с сервером SAMBA (виджет **form**). При поиске сервера и ожидании установки соединения используется виджет **ponder**.

Параметры загрузки:

- **altboot_cifs** — набор поддерживаемых аргументов в виде «ключ=значение», перечисленных через «;» (см. ниже);
- `cifsopts` — дополнительные опции монтирования **mount.cifs** для всех «шагов» cifs.





Поддерживаемые аргументы, в том числе, через **automatic=...**:

- **server** — имя или IP-адрес SAMBA сервера, по умолчанию определяется DHCP-сервером как первый WINS или DNS-сервер;
- **directory** — путь к каталогу дистрибутива или ISO-образу, по умолчанию «/netinst/current»;
- `domain` — название домена SAMBA;
- `user` — имя авторизованного пользователя;

- `pass` — пароль авторизованного пользователя;
- `timeout` — предел временного ожидания доступности сервера, по умолчанию 60 секунд;
- `options` — дополнительные опции монтирования, индивидуальные для каждого шага cifs;
- `overlays` — путь к каталогу с профилями оверлеев LiveCD, по умолчанию «/netinst/overlays-live».

Для каждого «шага» **cifs** задаётся свой набор параметров в **altboot_cifs**. Параметр **cifsopts**, напротив, имеет глобальное значение для всех «шагов» **cifs**. Если набор дополнительных опций монтирования разделяется пробелами, можно прописать их в **CIFSOPTS=...** в конфигурационном файле **/etc/sysconfig/bootchain**. Имя пользователя, пароль и домен необязательны, при их отсутствии метод **cifs** повторяет поведение исходной реализации данного метода в пропагаторе, т.е. монтирует с **-o guest**. Если не указать **server**, то на его поиск выделяется дополнительное время, равное половине **timeout**.

В случае успеха загрузки и только в *режиме совместимости с пропагатором* «шаг» **cifs** экспортирует в stage2 переменные окружения:

- **METHOD**=cifs — название выбранного *метода загрузки* altboot;
- **HOST** — имя или IP-адрес SAMBA сервера;
- **PREFIX** — путь к каталогу дистрибутива или ISO-образу;
- `SMBOPTS` — все опции монтирования, включая данные аутентификации (*новое в altboot*);
- `DOMAIN` — название домена SAMBA, если был указан domain;
- `LOGIN` — имя авторизованного пользователя, если был указан user;
- `PASSWORD` — пароль авторизованного пользователя, если был указан user;
- **PIGGYBACK**=1 — сообщает о двойном монтировании, когда приходится монтировать не только ресурс samba.

Пример использования:

```
root=bootchain bootchain=fg,altboot ip=dhcp4 \
  automatic=method:cifs,server:192.168.15.1,directory:/netinst/current
```

См. также:

- https://bugzilla.altlinux.org/40554

17.4. copyfile

Базовый интерактивный «шаг», входящий в подпакет **make-initrd-bootchain-altboot** (начиная с версии 0.1.5-alt4), это сильно упрощённая версия «шага» download, не использующая curl и сеть, она была создана чтобы в некоторых сценариях загрузки пропагатора не проходить «шаг» download дважды. Загружает уже доступный файл на смонтированной файловой системе в указанное устройство, в указанный каталог или на TMPFS.

Параметры загрузки:

- **altboot_copyfile** — набор аргументов в виде «ключ=значение», перечисленных через «;» (см. ниже).

Поддерживаемые аргументы, в том числе, через **automatic=...**:

- **src** — название файла, полученного на предыдущем «шаге»;
- **dst** — определяет место, куда загружать образ (см. ниже);
- **size** — размер копируемых данных, в байтах.

Возможные места для загрузки образов (значения аргумента **dst**):

- **RD** — использовать первый свободный RAM-диск /dev/ram<N>;
- **/dev/...** — использовать непосредственно указанное блочное устройство;
- **pipe**<N> или `step<N>` — имя устройства берётся из файла **DEVNAME** в каталоге результата указанного «шага»;
- *другое* — использовать обычный файл в памяти (на TMPFS) в стиле pipeline.

Пример использования:

```
root=bootchain bootchain=fg,localdev,copyfile,checksum,squashfs,liveboot,rootfs \
  altboot_localdev=method=cdrom;uuid=2021-05-31-01-09-58-00;directory=/rescue \
  altboot_copyfile=dst=RD;src=rescue ramdisk_size=497773 ...
```

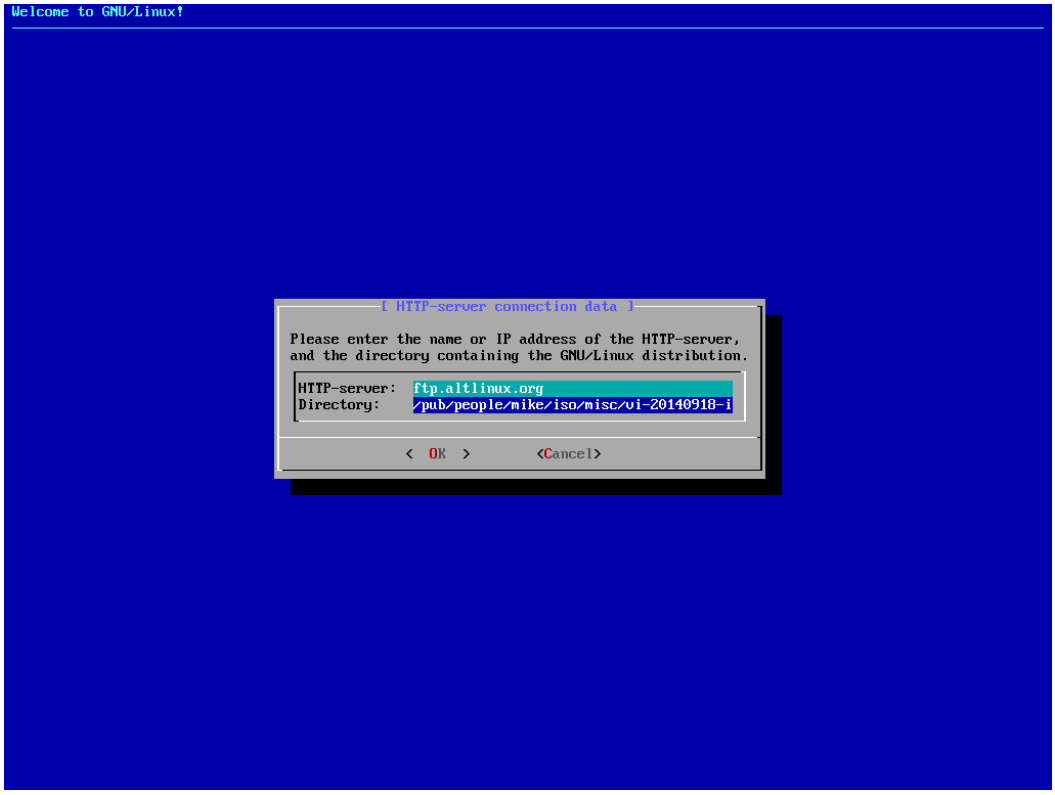
17.5. download

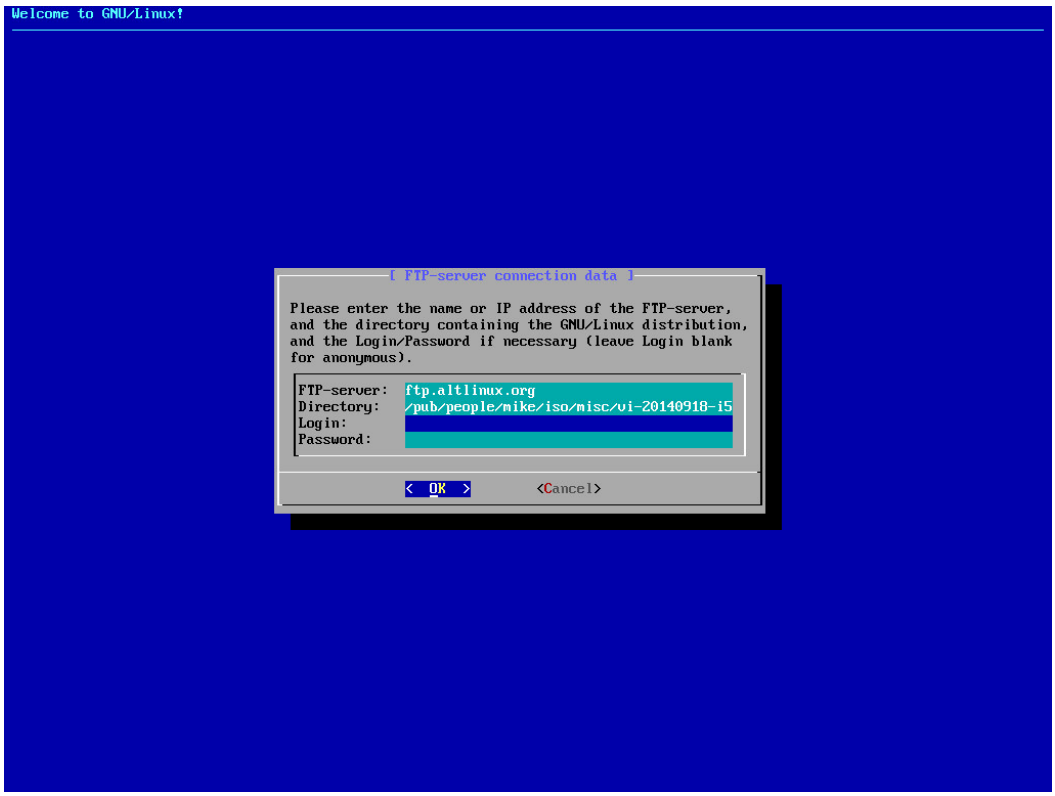
Многофункциональный интерактивный базовый «шаг», использующий curl. Входит в подпакет **make-initrd-bootchain-altboot**. Обеспечивает *методы загрузки* **url**, **http**, **ftp**, также может использоваться для загрузки сквоша в память, может загрузить образ чего угодно в любое указанное блочное устройство или как файл в каталог.

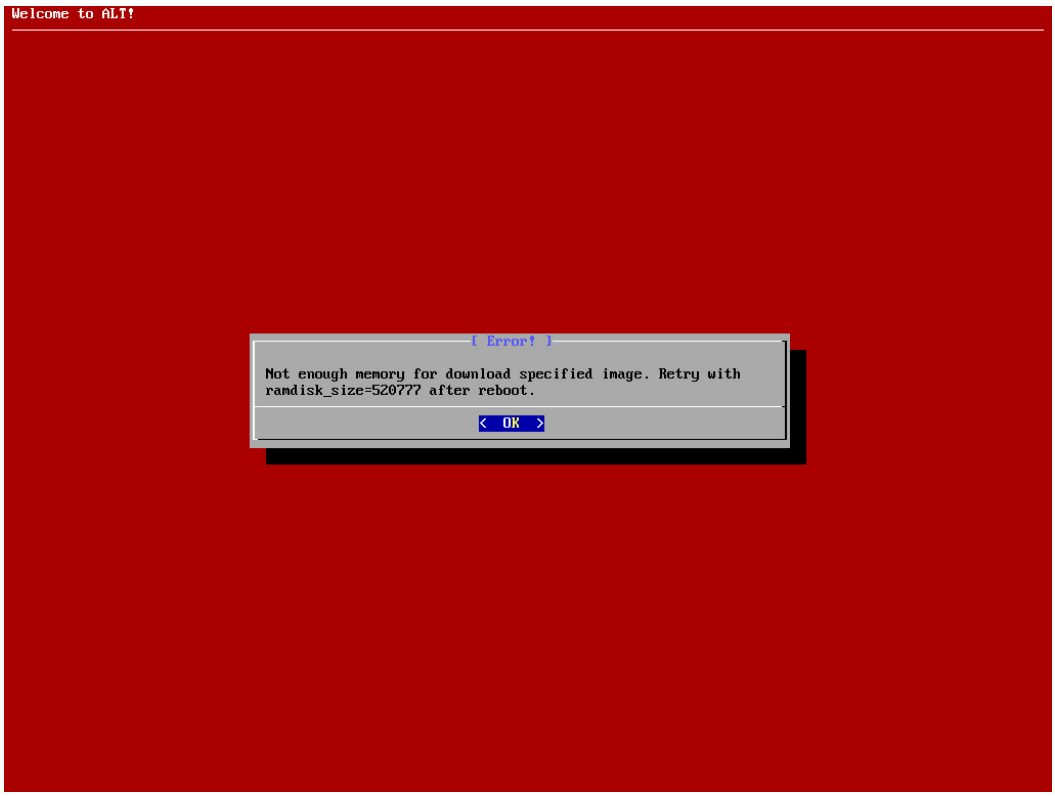
Для сетевой загрузки дополнительно требуется фича **make-initrd** «network». В процессе работы может выводить диалоги ввода данных (виджет **form**). При поиске сервера и ожидании установки соединения используется виджет **ponder**. При скачивании образа используется виджет **gauge**.

Параметры загрузки:

- **altboot_download** — набор аргументов в виде «ключ=значение», перечисленных через «;» (см. ниже);
- `curlopts` — дополнительные опции curl для всех «шагов» download.


Welcome to GNU/Linux!
[HTTP-server connection data]
Please enter the name or IP address of the HTTP-server,
and the directory containing the GNU/Linux distribution.
HTTP-server: ftp.altlinux.org
Directory: /pub/people/mike/iso/misc/vi-20140918-i
< OK >
<Cancel>


Welcome to GNU/Linux!
[FTP-server connection data]
Please enter the name or IP address of the FTP-server,
and the directory containing the GNU/Linux distribution,
and the Login/Password if necessary (leave Login blank
for anonymous).
FTP-server: ftp.altlinux.org
Directory: /pub/people/mike/iso/misc/vi-20140918-i5
Login:
Password:
< OK >
<Cancel>



Возможные места для загрузки образов (значения аргумента **to**):

- **RD** — использовать первый свободный RAM-диск /dev/ram<N>;
- **/dev/...** — использовать непосредственно указанное блочное устройство;
- **pipe**<N> или `step<N>` — имя устройства берётся из файла DEVNAME в каталоге результата указанного «шага»;
- *другое* — использовать обычный файл в памяти (на TMPFS) в стиле pipeline.

Поддерживаемые аргументы, в том числе, через **automatic=...**:

- **method** — *метод загрузки* altboot (см. выше);
- `to` — определяет место, куда загружать образ (см. выше);
- `url` — путь к файлу либо полный URL при использовании метода url;
- **server** — имя или IP-адрес сервера при использовании методов http/ftp;
- **directory** — полный путь к ISO-образу на сервере для методов http/ftp;
- **user** — имя авторизованного пользователя при использовании метода ftp;
- **pass** — пароль авторизованного пользователя при использовании метода ftp;
- `imgsize` — размер ISO-образа или сквоша в байтах, чтобы не запрашивать этот размер у сервера (*новое в altboot*);
- `timeout` — предел временного ожидания на соединение с HTTP/FTP-сервером (*новое в altboot*).

Параметр **curlopts** имеет глобальное значение для всех «шагов» **download**. Если набор дополнительных опций curl разделяется пробелами, можно прописать их в **CURLOPTS=...** в конфигурационном файле **/etc/sysconfig/bootchain**. Имя пользователя и пароль необязательны, при их отсутствии метод **download** повторяет поведение исходной реализации метода FTP в пропагаторе, т.е. скачивает файл анонимно.

Если не указать **server**, то на его поиск выделяется дополнительное время, равное половине **timeout**, по умолчанию используется опция **next-server** DHCP-сервера или адрес шлюза. Чтобы не запрашивать на сервере размер файла образа, на что тратится дополнительное время, и что может привести к ошибке, если сервер не возвращает размер файла по выбранному протоколу или возвращает неверный размер файла, следует использовать аргумент **imgsize**. Если не указан путь к ISO-образу (**directory**), по умолчанию предлагается значение из конфигурационной переменной **$OEM_URL_NETINST**, для ОС Альт это начальный путь всех официальных образов.

В случае успеха загрузки и только в *режиме совместимости с пропагатором* «шаг» **download** экспортирует в stage2 переменные окружения:

- **METHOD** — название выбранного *метода загрузки* altboot (http, ftp, url);
- **HOST** — имя или IP-адрес HTTP/FTP сервера;
- **PREFIX** — путь к ISO-образу на сервере;
- `URL` — полный URL, если был выбран одноимённый *метод загрузки*;
- **LOGIN** — имя авторизованного пользователя FTP, если был указан user;
- **PASSWORD** — пароль авторизованного пользователя FTP, если был указан user.

На момент написания настоящей документации спецификация определения дисков в стиле **MODEL=...** не поддерживается make-initrd, так что второй пример (ниже) скорее гипотетический.

Примеры использования:

```
root=bootchain bootchain=fg,altboot ip=dhcp4 \
  automatic=method:http,network:dhcp,server:ftp.altlinux.org,directory:\
  /pub/distributions/ALTLinux/p9/images/workstation/x86_64/alt-workstation-9.1-x86_64.iso

root=bootchain bootchain=waitdev,fg,download,... ip=dhcp4 \
  waitdev=MODEL=SAMSUNG_SSD_960_EVO_250GB \
  altboot_download=method=url;url=http://192.168.15.1/alt-ws11.img

root=bootchain bootchain=fg,localdev,download,squashfs,liveboot,rootfs \
  altboot_localdev=method=cdrom;uuid=2021-05-31-01-09-58-00;directory=/rescue \
  altboot_download=to=RD;method=url;url=file:///rescue ramdisk_size=497773 \
  altboot_liveboot=stagename=rescue;method=cdrom;overlays=SLICES;flags=live_ro,rescue
```

См. также:

- https://bugzilla.altlinux.org/40267
- https://bugzilla.altlinux.org/34546
- https://bugzilla.altlinux.org/36751

17.6. iso9660

Базовый неинтерактивный «шаг», входящий в подпакет **make-initrd-bootchain-altboot**. Монтирует устройство, полученное на предыдущем «шаге», как CD- или DVD-привод стандарта ISO-9660. Никаких параметров не требуется. Обычно используется после «шага» **download** при сетевой загрузке.

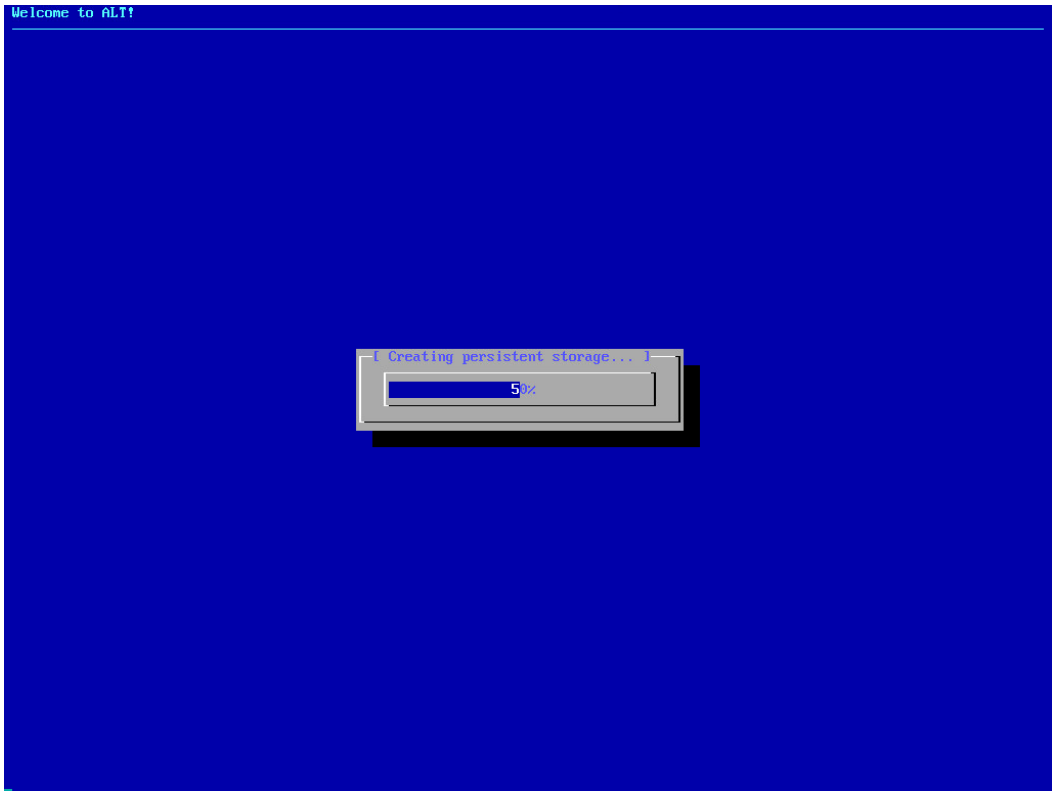
Примеры использования:

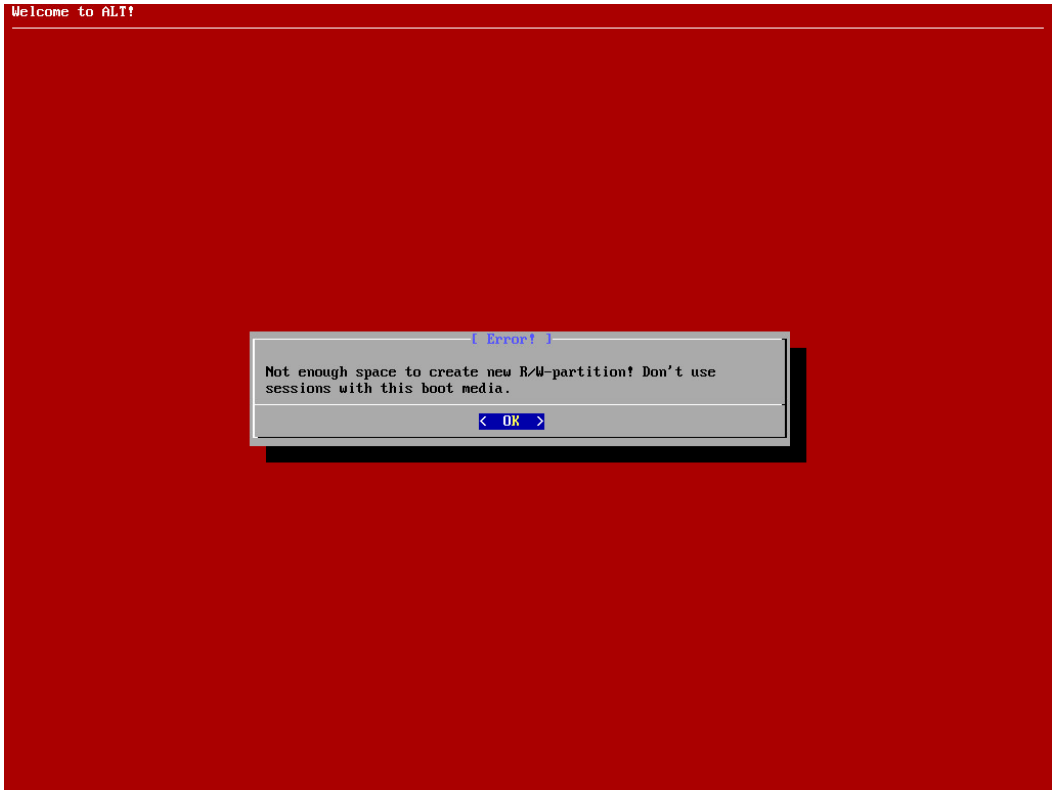
```
root=bootchain bootchain=fg,altboot ip=dhcp4 \
  automatic=method:ftp,server:192.168.15.1,directory:/srv/public/netinst/current

root=bootchain bootchain=fg,download,iso9660,squashfs,liveboot,rootfs ip=dhcp4 \
  altboot_download=method=http;server=192.168.15.1;directory=/srv/public/netinst/current \
  altboot_squashfs=/rescue \
  altboot_liveboot=stagename=rescue;method=http;flags=live_ro,live_rw
```

17.7. liveboot

Многофункциональный «шаг» финальной стадии загрузки, создающий оверлей LiveCD над read-only rootfs. Входит в подпакет **make-initrd-bootchain-altboot**. По сути это сильно переписанный скрипт **init-bottom** из пакета make-initrd-propagator. Часть его кода вынесена в виде «хуков» в другие подпакеты. **liveboot** можно использовать на обычной rootfs без шага **altboot** для закрытия корневой файловой системы на запись и организации оверлея над ней в виде TMPFS или указанного раздела, куда будут сохраняться изменения.

liveboot, в основном, неинтерактивный «шаг», но диалоги могут возникнуть при использовании в /proc/cmdline параметра **live_rw** и первом запуске LiveCD либо Rescue в *режиме сохранения сеансов LiveCD* с «локальными» *методами загрузки* (**disk** или **cdrom**). При этом будут задействованы виджеты **ponder** и **errmsg**:





Параметры загрузки:

- **altboot_liveboot** — набор аргументов в виде «ключ=значение», перечисленных через «;» (см. ниже).

Поддерживаемые аргументы:

- **method** — *метод загрузки* altboot (см. выше);
- **stagename** — название файла второй стадии загрузки (stage2);
- **overlayroot** — используется для закрытия обычной Linux rootfs на запись:
  - **disabled** — позволяет временно отключить overlayroot и загрузиться как с обычной rootfs;
  - **tmpfs** — предписывает использовать tmpfs в качестве оверлея, изменения потеряются при перезагрузке;
  - **UUID**=... — предписывает использовать раздел с заданным UUID в качестве оверлея;
  - **LABEL**=... — предписывает использовать раздел с заданной меткой тома в качестве оверлея;
  - **/dev/...** — предписывает использовать указанное блочное устройство в качестве оверлея;
- **directory** — монтируемый каталог или путь к ISO-образу, только для метода загрузки nfs;
- **profile** — название профиля (подкаталога) со сквошами или ISO-образами *read-only слоёв LiveCD*;
- **overlays** — определяет раздел, на котором находятся *read-only слои LiveCD* при «локальной» загрузке:
  - *метка тома* (LABEL) — профили и слои будут искаться на устройстве с файловой системой, содержащей указанную метку тома;
  - **local_profile** — будет использовано загрузочное устройство, на котором находится rootfs;
- **flags** — различные флаги создания оверлея LiveCD, передаваемые данному «шагу»:
  - **live_ro** — предписывает создавать оверлей над read-only LiveCD;
  - **live_rw** — предписывает использовать *хранилище сеансов LiveCD*;
  - **rescue** — влияет только на сообщения в журнале при загрузке с Rescue;
- **timeout** — предел временного ожидания для поиска *хранилища сеансов LiveCD*;

В случае шага **liveboot**, не все аргументы можно передать через пропагаторный параметр **automatic**.

Примеры использования:

```
root=bootchain bootchain=waitdev,waitdev,mountfs,liveboot,rootfs \
  waitdev=LABEL=OVERLAY \
  waitdev=UUID=e199d396-eb13-4575-b4fa-d2d160c67b6c \
  mountfs=DEVNAME \
  altboot_liveboot=overlayroot=LABEL=OVERLAY

root=bootchain bootchain=waitdev,mountfs,liveboot,rootfs \
  waitdev=UUID=e199d396-eb13-4575-b4fa-d2d160c67b6c \
  mountfs=DEVNAME \
  altboot_liveboot=method=
```

Ещё один пример использования:

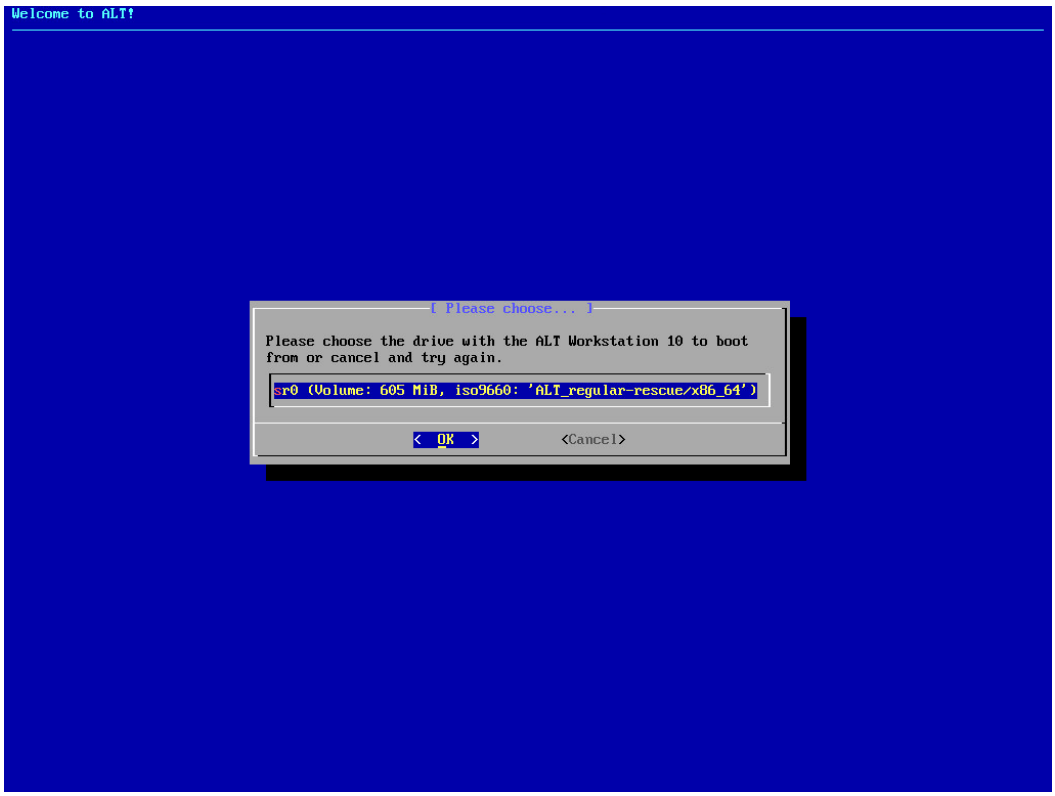
```
root=bootchain bootchain=fg,localdev,download,squashfs,liveboot,rootfs \
  altboot_localdev=method=cdrom;uuid=2021-05-31-01-09-58-00;directory=/rescue \
  altboot_download=to=RD;method=url;url=file:///rescue ramdisk_size=497773 \
  altboot_liveboot=stagename=rescue;method=cdrom;overlays=SLICES;flags=live_ro,rescue
```

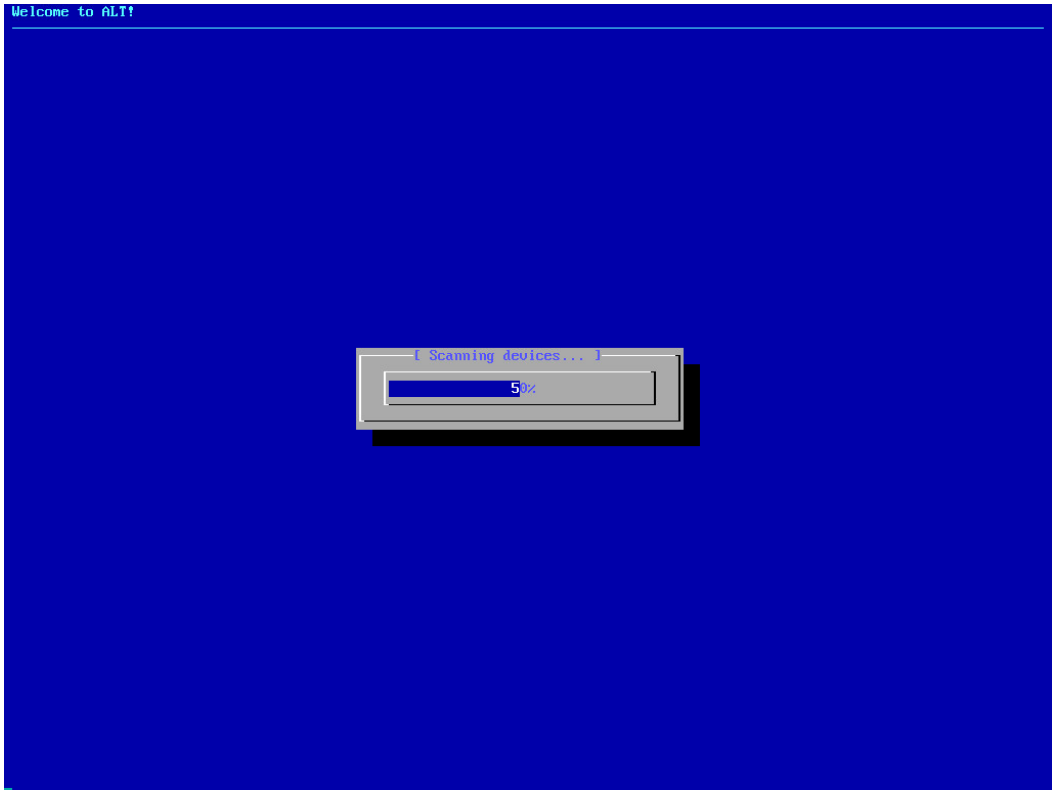
См. также:

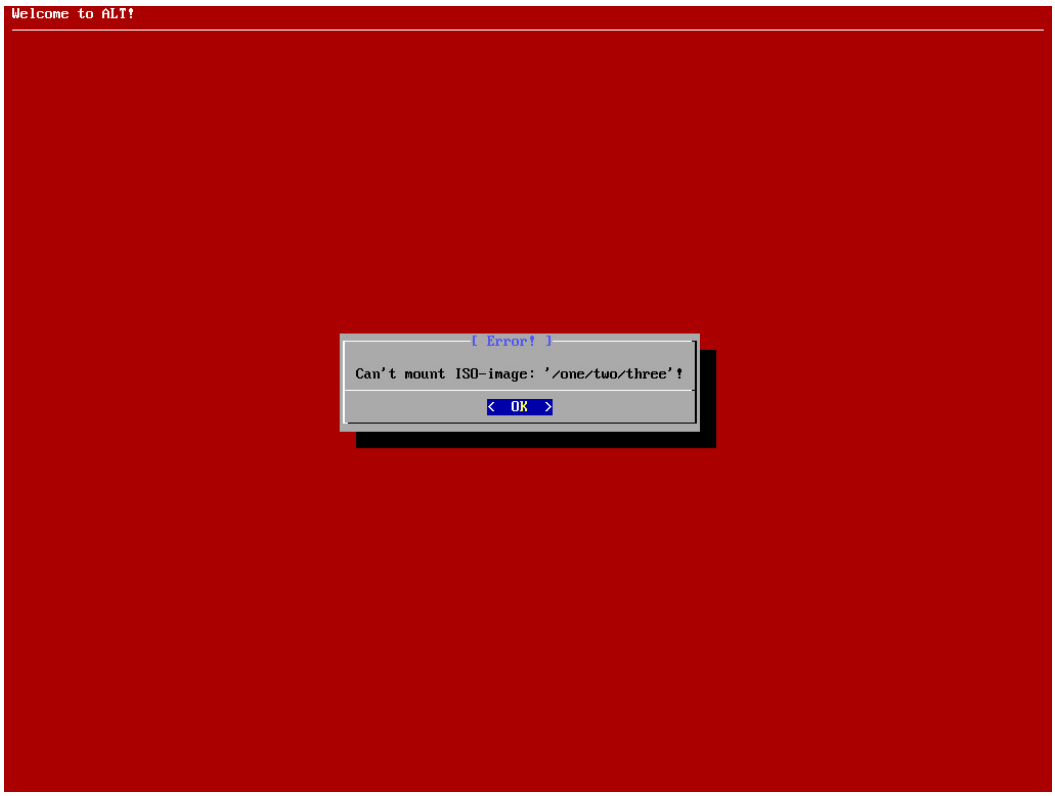
- https://bugzilla.altlinux.org/30513
- https://bugzilla.altlinux.org/30564
- https://bugzilla.altlinux.org/27852
- https://bugzilla.altlinux.org/32562

17.8. localdev

Интерактивный «шаг», входящий в подпакет **make-initrd-bootchain-localdev**, и обеспечивающий «локальные» *методы загрузки* **disk** и **cdrom**. В процессе работы может выводить диалоги выбора накопителя и ввода дополнительных данных для метода **disk** (виджеты **choice** и **form**). При поиске локального устройства используется виджет **ponder**. Если на предыдущем «шаге» (например, был «шаг» *waitdev*) устройство было найдено, то не выполняет сканирование устройств и не выводит диалогов, а использует найденное устройство, что позволяет использовать более богатый выбор спецификации устройств **waitdev** и далее **localdev** в качестве fallback, а также комбинировать «шаги», не внося изменений в код **altboot**. Если нужно явно игнорировать результаты предыдущего «шага», слева от **localdev** следует поставить «шаг» *noop*.







Параметры загрузки:

- **altboot_localdev** — набор поддерживаемых аргументов в виде «ключ=значение», перечисленных через «;» (см. ниже).

Поддерживаемые аргументы, в том числе, через **automatic=...**:

- **method** — метод загрузки altboot: disk или cdrom (см. выше);
- **directory** — путь к ISO-образу для метода disk, путь к сквошу для метода cdrom;

- **disk** — название блочного устройства, ссылающегося на целый диск, например: sda, md0, dm-0, nvme0n1;
- **part** — название блочного устройства, ссылающегося на раздел, например: sdb1, c0d0p3, md0p1, nvme0n1p2;
- **uuid** — UUID файловой системы раздела для поиска по UUID;
- **label** — метка тома файловой системы раздела для поиска по LABEL;
- **overlays** — определяет раздел, на котором находятся read-only слои LiveCD при локальной загрузке:
  - *метка тома* (LABEL) — профили и слои будут искаться на устройстве с файловой системой, содержащей указанную метку тома;
  - `local_profile` — будет использовано загрузочное устройство, на котором находится rootfs;
- `timeout` — предел временного ожидания для поиска носителя;
- `options` — дополнительные опции монтирования.

В случае успеха загрузки и только в *режиме совместимости с пропагатором* «шаг» **localdev** экспортирует в stage2 следующие переменные окружения:

- **METHOD** — название выбранного *метода загрузки* altboot (disk или cdrom);
- **DEVICE** — название устройства загрузочного диска или раздела, например: sda, sdb1, md0p1, nvme0n1p2;
- **PREFIX** — полный путь к ISO-образу для метода disk, «/» — в остальных случаях;
- **PIGGYBACK**=1 — сообщает о двойном монтировании, когда приходится монтировать ещё и ISO-образ с диска;

Примеры использования:

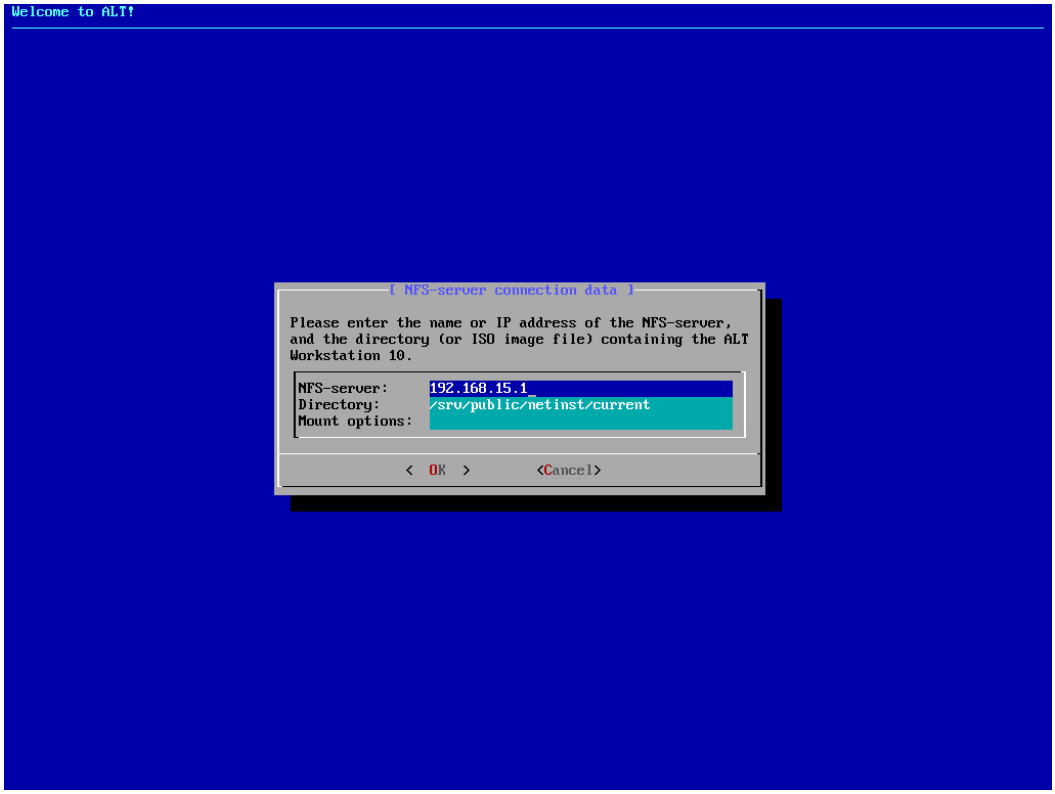
```
root=bootchain bootchain=fg,altboot \
  automatic=method:cdrom,uuid:2021-05-30-13-35-30-00 \
  stagename=altinst ramdisk_size=160905

root=bootchain bootchain=fg,altboot \
  automatic=method:cdrom,uuid:2021-05-30-18-28-59-00,overlays:SLICES \
  stagename=live ramdisk_size=1081725

root=bootchain bootchain=fg,localdev,copyfile,checksum,squashfs,liveboot,rootfs \
  altboot_localdev=method=cdrom;uuid=2021-05-30-12-55-00-00;directory=/rescue \
  altboot_copyfile=dst=RD;src=rescue ramdisk_size=497773 \
  altboot_checksum=613bf1a12803b448682d0c6d628400d2a52f38b61e6815af34f364cf71ad65ec \
  altboot_liveboot=stagename=rescue;method=cdrom;flags=live_ro,rescue,live_rw
```

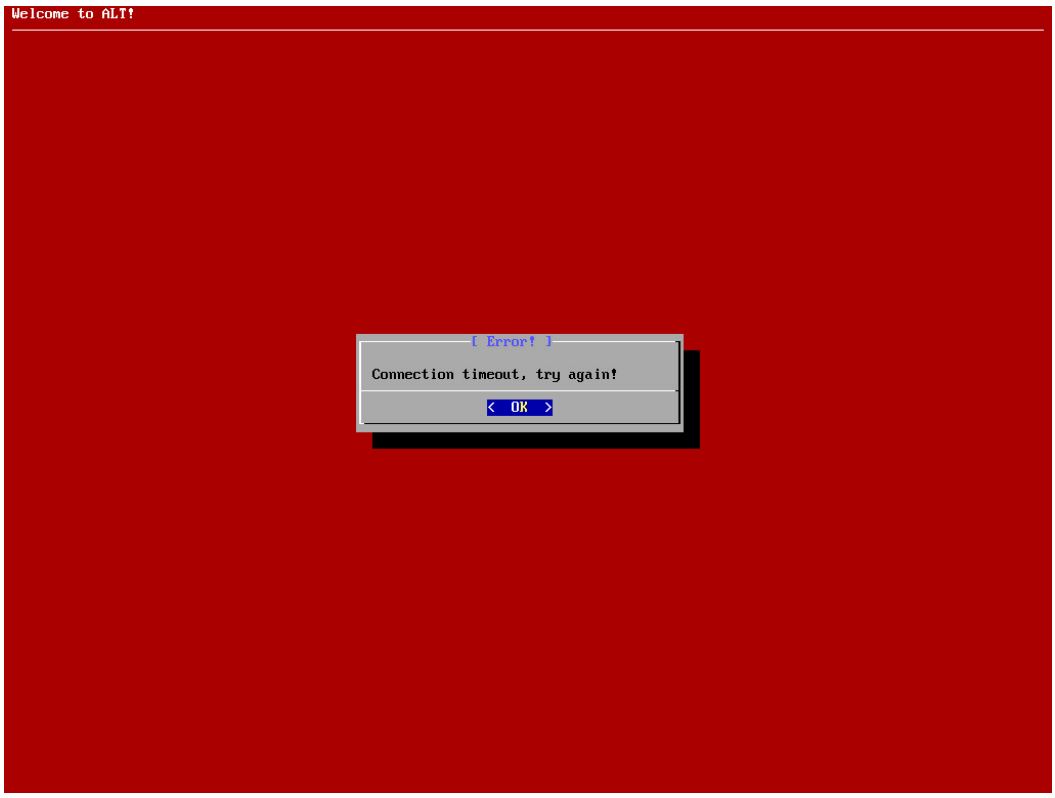
См. также:

- https://bugzilla.altlinux.org/33954
- https://bugzilla.altlinux.org/31068

17.9. nfs

Входит в подпакет **make-initrd-bootchain-nfs**, обеспечивает сетевой *метод загрузки* с сервера NFS и поддержку *read-only слоёв LiveCD* (сквошей), размещаемых также на сервере NFS. В процессе работы может вывести диалог ввода данных для соединения с сервером NFS (виджет **form**). При поиске сервера и ожидании установки соединения используется виджет **ponder**.





Параметры загрузки:

- **altboot_nfs** — набор поддерживаемых аргументов в виде «ключ=значение», перечисленных через «;» (см. ниже).
- `nfsopts` — дополнительные опции монтирования **mount.nfs** для всех «шагов» nfs.

Поддерживаемые аргументы, в том числе, через **automatic=...**:

- **server** — имя или IP-адрес NFS сервера, по умолчанию используется DHCP-опция next-server либо адрес дефолтного шлюза;
- **directory** — путь к каталогу дистрибутива со сквошом stage2 или ISO-образу, по умолчанию «/srv/public/netinst/current»;
- `timeout` — предел временного ожидания доступности сервера, по умолчанию 60 секунд;
- `options` — дополнительные опции монтирования, индивидуальные для каждого «шага» nfs;
- **overlays** — путь к каталогу с профилями оверлеев LiveCD, по умолчанию «/srv/public/netinst/overlays-live».

Для каждого «шага» **nfs** задаётся свой набор параметров в **altboot_nfs**. Параметр **nfsopts**, напротив, имеет глобальное значение для всех «шагов» **nfs**. Если набор дополнительных опций монтирования разделяется пробелами, можно прописать их в **NFSOPTS=...** в конфигурационном файле **/etc/sysconfig/bootchain**. Если не указать **server**, то на его поиск выделяется дополнительное время, равное половине **timeout**.

В случае успеха загрузки и только в *режиме совместимости с пропагатором* «шаг» **nfs** экспортирует в stage2 переменные окружения:

- **METHOD**=nfs — название выбранного *метода загрузки* altboot;
- **HOST** — имя или IP-адрес NFS сервера;
- **PREFIX** — путь к каталогу дистрибутива или ISO-образу;
- **PIGGYBACK**=1 — сообщает о двойном монтировании, когда приходится монтировать ещё и ISO-образ с сервера.

Пример использования:

```
root=bootchain bootchain=fg,altboot ip=dhcp4 \
    automatic=method:nfs,server:192.168.15.1,directory:/srv/public/netinst/current
```

17.10. oemsetup

Интерактивный «шаг», входящий в подпакет **make-initrd-bootchain-localdev**, предназначенный для догрузки в initrams «на лету» недостающего (модулей ядра, правил udev, и т.д.). В процессе работы может выводить диалог выбора накопителя (виджет **choice**). При поиске локального устройства используется виджет **ponder**. Данный «шаг» можно использовать сам по себе, но в случае **altboot** его не следует указывать, т.к. при наличии непустого параметра **updatemodules** в /proc/cmdline, «шаг» **oemsetup** станет первым, что выполнит сам **altboot**.

По умолчанию «шаг» ищет накопитель с меткой тома, определяемой конфигурационной переменной **$OEM_SETUP_STORAGE**, однако пользователь может выбрать в диалоге любое другое устройство. Для дистрибутивов Альт здесь ожидается метка тома «**alt-drivers-update**», для других дистрибутивов дефолтом является «**OEMSETUP**». Для носителя с обновлениями от вендора допускается любая поддерживаемая файловая система (рекомендуется ext2/3/4 или vfat).

В корне устройства должен находиться каталог «**OEM-setup**» с профилями оборудования. Внутри подкаталога профиля ожидается файл «**initrd-update.img**», представляющий собой сжатый gzip'ом cpio-образ с файлами обновления initramfs. Название профиля оборудования (подкаталога) определяется параметром **updatemodules**. После распаковки архива в корень initramfs запускается скрипт «**/root/update.sh**», при его наличии, иначе выполняется команда «**depmod -a**».

Полный путь к архиву обновления на носителе: «**/OEM-setup/**<профиль>**/initrd-update.img**».

Параметры загрузки:

- **updatemodules** — определяет профиль оборудования и место, где будут искаться обновления на носителе.

Возможные значения updatemodules:

- **1** — подразумевается использование профиля по умолчанию («default»);
- **ID** — имя профиля вычисляется командой «**machine-info --drivers**»;
- *другое* — имя профиля указывается непосредственно в качестве значения.

Использование **updatemodules=ID** в /proc/cmdline позволяет администратору большой сети с различным проблемным оборудованием иметь единый носитель обновления для различающихся по конфигурации физических машин.

Поставщику содержимого архива «**initrd-update.img**» следует помнить, что модули ядра в нём должны быть совместимыми с версией ядра в целевом образе initramfs, другое содержимое архива также должно соответствовать целевому образу initramfs. Скрипт **/root/update.sh** может вызывать вспомогательные функции make-initrd, включая набор из **initeractive-sh-functions**, **bootchain-sh-functions** и **altboot-sh-functions**. См. детали в *разделе для разработчика*.

17.11. overlayroot

Неинтерактивный базовый «шаг», входящий в подпакет **make-initrd-bootchain-altboot**. Обеспечивает простой синтаксис для закрытия на запись имеющейся rootfs, т.е. на обычной установленной системе. Основные действия выполняются шагами *waitdev*, *mountfs* и *liveboot*, данный «шаг» лишь упрощает синтаксис, делая его похожим на аналогичную фичу в дистрибутивах Ubuntu. На текущий момент не рекомендуется использовать одновременно с фичей **make-initrd** «resume» во избежании «гонок».

Параметры загрузки:

- **overlayroot** — обязательный, имеет формат: «<rootdev>[;<overlay>]», т.е. определяет обычный корень, закрываемый на запись, и (необязательно) создаваемый над ним оверлей. Оба значения описывают ожидаемые устройства, подобно waitdev. Значения разделяются «;». Оверлей также может быть «tmpfs» (по умолчанию) или «disabled», что предписывает временно вернуть обычный режим работы rootfs.

Пример использования:

```
# Устанавливаем необходимые пакеты
apt-get install make-initrd-bootchain-altboot make-initrd-bootchain-waitdev

# Добавляем требуемые фичи в конфиг make-initrd
echo 'FEATURES += bootchain-altboot bootchain-waitdev' >>/etc/initrd.mk

# Добавляем параметры загрузки в конфиг grub:
uuid="$(grep -E '^UUID=.[[:space:]]/[[:space:]]' /etc/fstab |awk '{print $1}')"
cmdline="root=bootchain bootchain=overlayroot overlayroot=UUID=$uuid panic=30 splash"
echo "GRUB_CMDLINE_LINUX_DEFAULT='$cmdline'" >>/etc/sysconfig/grub2

# Выполняем ещё несколько команд
make-initrd
update-grub
reboot
```

После перезагрузки все изменения будут записываться на tmpfs и не сохраняться между перезагрузками. Чтобы временно отменить данный режим закрытия rootfs на запись, в самом начале загрузки нажимаем «**E**» в меню grub и дописываем к параметру загрузки **overlayroot**=... строку «**;disabled**». Должно получиться что-то вроде:

```
overlayroot=UUID=83dc6758-dc9c-48b0-8a13-683cc3d91d97;disabled
```

Чтобы изменения сохранялись не на tmpfs, а в отдельный раздел, например, с меткой тома «**OVERLAY**», дописываем строку «**;LABEL=OVERLAY**» к параметру **overlayroot**=... в **/etc/sysconfig/grub2**, запускаем **update-grub** и снова перезагружаемся. В результате в /proc/cmdine при каждой загрузке будет попадать что-то вроде:

```
root=bootchain bootchain=overlayroot \
  overlayroot=UUID=83dc6758-dc9c-48b0-8a13-683cc3d91d97;LABEL=OVERLAY
```

17.12. squashfs

Базовый неинтерактивный «шаг», входящий в подпакет **make-initrd-bootchain-altboot**. Монтирует через lomount() указанный файл из каталога или устройство, полученное на предыдущем «шаге», как сквош.

Параметры загрузки:

- **altboot_squashfs** — обязательный, если результат предыдущего «шага» — смонтированный каталог, то указывает путь к файлу со сквошом в этом каталоге. В терминах пропагатора — rootfs второй стадии загрузки (/$stagename). Если результатом предыдущего «шага» является устройство, передаваемое через **dev** или **DEVNAME**, тогда этот параметр не используется, но он всё равно должен содержать непустую строку (такая особенность регистрации массивов в make-initrd).

Примеры использования:

```
root=bootchain bootchain=fg,altboot ip=dhcp4 stagename=rescue \
  automatic=method:ftp,network:dhcp,server:192.168.15.1,directory:/netinst/alt.iso

root=bootchain bootchain=fg,download,iso9660,squashfs,liveboot,rootfs ip=dhcp4 \
  altboot_download=to=RD;method=http;server=192.168.15.1;directory=/netinst/alt.iso \
  altboot_squashfs=/live ramdisk_size=1081725 \
  altboot_liveboot=stagename=live;method=http;flags=live_ro,live_rw
```

18. Сопряжение входов/выходов

Демон **chaind** не накладывает каких-либо ограничений на то, чем апеллируют «шаги»-скрипты, но при составлении цепочки **bootchain=...** в /proc/cmdline необходимо учитывать, что каждый «шаг» ожидает на «входе» и что формирует на «выходе». Например, такая цепочка не будет работать:

```
bootchain=fg,download,liveboot,rootfs ...
```

потому что на «выходе» шаг «**download**» формирует имя смонтированного устройства в файле **DEVNAME**, а шаг «**liveboot**» ожидает на «входе» смонтированный каталог.

Среди обозначенных в документации шагов есть «**noop**» специально выполняющий функцию «разрыва» цепочки. Есть наоборот, «транзитные» шаги: «**altboot**», «**checksum**», «**oemsetup**», «**overlayroot**», связывающие «выход» предыдущего шага со «входом» следующего шага. Есть служебные и внутренние псевдо-шаги, на которые даже не тратится каталогов, они никакого влияния на сопряжение других шагов не оказывают и их не следует учитывать в нумерации шагов: «**debug**», «**fg**», «**noretry**» и «**retry**».

Table 3. Таблица маркировки входов/выходов:

Ожидает на «входе»	Название «шага»	Формирует на «выходе»
<bypass>	altboot	<bypass>
<multi-1>	checksum	<bypass>
—	cifs	<mp>
<mp>	copyfile	DEVNAME+dev+FILESIZE
<bypass>	debug	<bypass>
[DEVNAME]	download	DEVNAME+dev+FILESIZE
<bypass>	fg	<bypass>
—	getimage	<mp>
DEVNAME/dev	iso9660	<mp>
<mp>	liveboot	<bypass>
[DEVNAME/dev]	localdev	<mp>
[<resolve>]	mountfs	<mp>
—	nfs	<mp>
—	noop	<nothing>
<bypass>	noretry	<bypass>
<bypass>	oemsetup	<bypass>
[<mp>]	overlayfs	<mp>
<bypass>	overlayroot	<bypass>

Ожидает на «входе»	Название «шага»	Формирует на «выходе»
—	ping	<nothing>
<bypass>	retry	<bypass>
<mp>	rootfs	<nothing>
<mp>/DEVNAME/dev	squashfs	<mp>
—	wait-resume	<nothing>
—	waitdev	DEVNAME+dev

Обозначения в таблице:

- **<bypass>** — «транзитный» шаг: не имеет значения, что на «входе», то же будет и на «выходе»;
- **<mp>** — точка монтирования (каталог с файлами);
- **<nothing>** — на «выходе» НИЧЕГО, пустой результат;
- «—» (прочерк) — не имеет значения, что на «входе»;
- **[...]** (в квадратных скобках) — необязательно, т.е. может ничего не быть;
- «/» (знак деления) — на «входе»: ИЛИ то, ИЛИ другое;
- «+» (плюс) — на «выходе»: И то, И другое;
- **FILESIZE** — файл с таким именем, содержащий размер данных;
- **DEVNAME** — файл с таким именем, содержащий название устройства;
- **dev** — непосредственно сам файл устройства в стиле pipeline;
- **<resolve>** — зависит от результата resolve_target(), обычно файл или устройство;
- **<multi-1>** — слишком большой набор вариантов для размещения в одной ячейке, так что см. check_prevstep_results() в исходнике.

Стоит заметить, что в таблице не уточняется, о каких устройствах идёт речь при использовании **dev** или **DEVNAME**. В каких-то случаях подразумеваются любые устройства, включая символьные, в других случаях явно подразумеваются только блочные устройства. Такие детали можно исследовать в исходном коде.

19. Разработчику bootchain + altboot

19.1. Общий вспомогательный код

19.1.1. interactive-sh-functions

Общий код для обеспечения диалогов и *интерактивного взаимодействия* с пользователем. Входит в подпакет **make-initrd-bootchain-interactive**. *Библиотека виджетов* вынесена в отдельные скрипты, загружаемые динамически, по мере необходимости. Данный модуль не зависит от **bootchain** и **altboot**, есть смысл сделать его общей фичей в самом **make-initrd**, но автору make-initrd не нравится идея работать с TTY.

Синтаксис:

```
. interactive-sh-functions
```

Глобальные переменные:

- **$IM_BACKTITLE** — содержит заголовок верхнего уровня, выводимый всеми виджетами;
- **$IM_WIDGET_ARGS** — можно передавать дополнительные аргументы команде dialog перед использованием виджета;
- **$NOASKUSER** — непустое значение, если запрещены диалоги ввода;
- **$NOLINES** — непустое значение, если нет поддержи символов псевдографики.

Функции:

- **IM_is_active()** — возвращает 0, если интерактивный режим уже активен;
- **IM_exec()** — переводит указанный процесс на передний план, на конкретный терминал TTY;
- **IM_activate()** — запрашивает немедленную или отложенную активацию интерактивного режима;
- **IM_load_widgets()** — динамически подгружает запрошенные виджеты из библиотеки;
- **IM_load_all()** — подгружает из библиотеки все доступные виджеты;
- **IM_start_output()** — сообщает модулю о начале вывода;
- **IM_start_input()** — сообщает модулю о начале ввода;
- **IM_show_bootsplash()** — разрешает видеть на переднем плане заставку plymouth, если доступна;
- **IM_hide_bootsplash()** — скрывает заставку plymoth, если она есть, и приостанавливает прогресс бар;
- **IM_update_bootsplash()** — сообщает демону plymouthd, если есть, о прохождении очередной стадии загрузки.

19.1.2. bootchain-sh-functions

Общий код, используемый всеми «шагами» **bootchain** и **altboot**. Также пригоден и для «шагов» **pipeline**. Входит в подпакет **make-initrd-bootchain-core**.

Синтаксис:

```
. bootchain-sh-functions
```

Конфигурационный файл:

```
/etc/sysconfig/bootchain
```

Глобальные переменные:

- **$mntdir** — рабочий каталог демона для организации «входов» и «выходов» каждого «шага»;
- `$pipeline_mode` — непустое значение, если демон chaind работает в режиме «pipeline», а не в «родном» режиме;
- `$BC_DEBUG` — непустое значение, если включена расширенная диагностика;
- `$BC_LOG_VT` — если непустое значение, то порядковый номер TTY для вывода журнала;
- `$BC_LOGFILE` — путь к файлу журнала или устройство для вывода в него сообщений отладки;
- `$BC_DEVICE_TIMEOUT` — дефолтный таймаут для любых «шагов» bootchain;
- `$BC_FGVT_ACTIVATE` — если непустое значение, то через сколько секунд активировать интерактивный терминал.

Функции:

- **check_parameter()** — проверяет, чтобы обязательный параметр был не пуст или завершается через fatal();
- **get_parameter()** — вывод значения параметра текущего «шага» по индексу $callnum;
- **resolve_target()** — вывод пути к файлу, каталогу или устройству, в зависимости от параметра;
- `resolve_devname()` — вывод пути к специальному файлу устройства по указанному каталогу; обычно каталог «шага» содержит файл DEVNAME или dev, если устройство было результатом «шага», тогда функция вернёт читаемый /dev/УЗЕЛ;
- `debug()` — вывод текстового сообщения при расширенной диагностике;
- `enter()` — трассировка при расширенной диагностике: вход в указанную функцию;
- `leave()` — трассировка при расширенной диагностике: выход из указанной функции;
- `run()` — запуск внешней команды, при расширенной диагностике команда попадёт в журнал;
- `fdump()` — вывод содержимого указанного файла при расширенной диагностике;
- `assign()` — присвоение переменной указанного значения, попадающее в журнал;
- `next_bootchain()` — команда демону на смену последовательности следующих «шагов»;

- `is_step_passed()` — возвращает 0, если текущий «шаг» был пройден хотя бы один раз;
- `launch_step_once()` — если текущий «шаг» уже был пройден, завершает работу через вызов fatal();
- `break_bc_loop()` — сообщает демону о том, что текущий «шаг» последний и можно переходить в stage2;
- `bc_reboot()` — выполняет журналируемый перезапуск компьютера;
- `bypass_results()` — просит демон связать «выход» предыдущего «шага» со «входом» следующего «шага»;
- `initrd_version()` — вывод текущей версии make-initrd.

19.1.3. altboot-sh-functions

Общий код модуля **make-initrd-bootchain-altboot**, используемый всеми «шагами» **altboot**.

Синтаксис:

```
. altboot-sh-functions
```

Конфигурационные файлы:

```
/etc/sysconfig/bootchain: изначально используется общая конфигурация с bootchain;
/.initrd/bootchain/altboot.conf: конфигурация последующих «шагов», создаваемая altboot;
/.initrd/bootchain/altboot.env: окружение, экспортируемое «шагами» altboot в stage2;
/.initrd/bootchain/altboot.auto: сюда сохраняется метод загрузки, выбранный вручную.
```

Глобальные переменные:

- **$ALTBOOT_OLDROOT** — непустое значение, если «шаг» должен работать в режиме совместимости с пропагатором;
- **$OEM_WELCOME_TEXT** — используется в качестве заголовка верхнего уровня в диалогах altboot;
- **$OEM_DISTRIBUTION** — используется в качестве названия дистрибутива в диалогах altboot;
- **$OEM_CDROOT** — необязательный путь к корню ISO-образа внутри initramfs, в ОС Альт это /image;
- **$OEM_LIVE_STORAGE** — метка тома live_rw раздела, если указана;
- **$OEM_BAD_STORAGE** — метка тома, чтобы не использовать live_rw раздел на плохом или слишком медленном устройстве;
- **$OEM_SETUP_STORAGE** — метка тома раздела для обновления initramfs «на лету»;
- **$OEM_IMAGES_BASE** — куда в stage2 монтировать каталог с образами слоёв LiveCD;
- **$OEM_OVERLAYS_DIR** — куда в stage2 монтировать сами слои LiveCD;
- **$OEM_URL_NETINST** — значение по умолчанию компоненты directory для загрузки методами url, http и ftp;

- **$OEM_NFS_NETINST** — значение по умолчанию компоненты directory для загрузки методом nfs;
- **$OEM_CIFS_NETINST** — значение по умолчанию компоненты directory для загрузки методом cifs.

Функции:

- **get_bootarg()** — получает значение целого параметра загрузки текущего «шага» или указанного аргумента;
- **lomount()** — выполняет монтирование файла/устройства после выполнения losetup в стиле пропагатора;
- **stage2_setenv()** — устанавливает переменную окружения, экспортируемую в stage2;
- **stage2_getenv()** — выводит значение переменной окружения, ранее экспортированной в stage2;
- **get_free_ramdisk()** — присваивает указанной переменной имя устройства первого свободного RAM-диска;
- **mark_used_ramdisk()** — помечает указанный RAM-диск, как используемый;
- **mark_free_ramdisk()** — помечает указанный RAM-диск, как неиспользуемый;
- **use_hooks()** — запускает на выполнение указанный набор «хуков»;
- **altboot_restart()** — перезапускает все шаги altboot с самого начала.

19.1.4. altboot-net-functions

Общий код модуля **make-initrd-bootchain-waitnet**, используемый всеми «шагами» **altboot** для сетевой загрузки, начиная с версии 0.1.5-alt1.

Синтаксис:

```
. altboot-net-functions
```

Конфигурационный файл:

```
/etc/sysconfig/bootchain
```

Глобальные переменные:

- **$OEM_SRV_NETINST** — значение по умолчанию компоненты server для загрузки методами url, http и ftp, определяет IP-адрес или имя сервера сетевой загрузки.

Функции:

- **bits_to_mask4()** — конвертирует число бит сетевого адреса в сетевую маску, например, $(bits_to_mak4 24) = 255.255.255.0;
- **ip4_to_network()** — конвертирует IPv4-адрес хоста и сетевую маску в IPv4-адрес сети;

- **get_dhcp_next_server()** — извлекает IPv4-адрес next-server из опций DHCPv4, сохраняемых фичей «network» make-initrd;
- **get_dhcp_root_path()** — извлекает rootpath из опций DHCPv4, сохраняемых фичей «network» make-initrd;
- **get_dhcp_wins()** — извлекает IPv4-адрес первого сервера WINS из опций DHCPv4, сохраняемых фичей «network» make-initrd;
- **get_default_gateway()** — определяет шлюз по умолчанию;
- **get_first_dns()** — извлекает IPv4-адрес первого сервера DNS из файла /etc/resolv.conf.

19.1.5. scandev-sh-functions

Весьма объёмный общий код модуля **make-initrd-bootchain-localdev** между «шагами» **localdev** и **oemsetup**, который также может использоваться в иных разрабатываемых методах загрузки с локальных носителей.

Синтаксис:

```
. scandev-sh-functions
```

Глобальные переменные:

- **$disk** — название целого диска (sda, md0, dm-0, nvme0n1) для поиска соответствия;
- **$part** — название раздела (sda1, nvme0n1p2, c0d0p3) для поиска соответствия;
- **$uuid** — UUID раздела для поиска соответствия;
- **$label** — метка тома (LABEL) раздела для поиска соответствия;
- **$target** — используется для сохранения пути к найденному устройству;
- **$method** — один из методов сканирования: oem, disk или cdrom;
- **$devices** — массив названий обнаруженных устройств.

Функции:

- **scan_devices()** — сканирует устройства, заполняя переменные $devices и $target, с учётом указанных соответствий заданной спецификации;
- **device_choice()** — выводит меню выбора устройств, в зависимости от указанного метода загрузки.

19.2. machine-info

Утилитарный скрипт из подпакета **make-initrd-bootchain-core**, предназначенный для сбора уникальной информации о «железе» и создании уникального хэша по этой информации.

Синтаксис:

```
machine-info [<option>]
```

Ключи запуска:

- **-d**, **--drivers**: вывести уникальный хэш для привязки к типу машины;
- **-i**, **--instance**: вывести уникальный хэш для привязки к конкретному экземпляру машины;
- **без ключей**: вывести основную информацию о типе и экземпляре в виде текста.

19.3. Библиотека виджетов

Входит в подпакет **make-initrd-bootchain-interactive**. Скрипты виджетов попадают в каталог **/lib/IM-widgets** внутри формируемого образа initramfs. Библиотеку можно расширять, но следует иметь ввиду, что генератор **make-initrd** не допускает перезапись уже попавших в образ файлов.

В начале каждой функции виджета следует использовать вызов **IM_start_input**, если это виджет ввода, а в остальных случаях использовать **IM_start_output**. Помимо динамической загрузки необходимого кода, такой подход позволяет сообщить модулю о начале ввода, что является одним из триггеров переключения в интерактивный терминал (по умолчанию **/dev/tty2**). Если виджету требуется код другого виджета, то его следует указать в списке требуемых, например:

```
IM_start_output gauge errmsg
```

Все нижеперечисленные диалоги выводят заголовок верхнего уровня из переменной **$IM_BACKTITLE**, которая в случае **altboot** устанавливается из конфигурационной переменной **$OEM_DISTRIBUTION** (в основе параметр **dialog --backtitle**). Перед вызовом любой функции диалога можно экспортировать переменную **DIALOGRC**=/путь/к/кофигу_dialog для определения альтернативной палитры цветов. Все функции диалогов используют глобальную переменную **$IM_WIDGET_ARGS** для передачи программе **dialog** дополнительных аргументов.

19.3.1. choice

Диалог ввода. Выводит меню из одного или более пунктов. Метки перед пунктами не выводятся, а записываются в указанную переменную, когда пользователь выберет соответствующий пункт в меню. Если выбор будет сделан, возвращает 0. Основан на **dialog --menu**.

Синтаксис:

```
IM_choice <переменная> <текст> <метка1> <пункт1> [<метка2> <пункт2>...]
```

Пример использования:

```
text="Please choose the installation method."

while ! IM_choice method "$text" \
    nfs   "NFS server"      \
    ftp   "FTP server"      \
    http  "HTTP server"     \
    cifs  "SAMBA server"    \
    cdrom "CD-ROM Drive"    \
    disk  "Hard Disk Drive" \
    #
do
    sleep 0.5
done
```

19.3.2. dlgmsg

Диалог ввода. Выводит текстовое сообщение. Всегда возвращает 0. Основан на **dialog --msgbox**.

Синтаксис:

```
IM_dlgmsg <заголовок> <текст>
```

Пример использования:

```
IM_dlgmsg "Live is success!" "$text"
```

19.3.3. errmsg

Диалог ввода. Выводит сообщение об ошибке. Всегда возвращает 0. Основан на **dialog --msgbox**.

Синтаксис:

```
IM_errmsg <текст>
```

Пример использования:

```
IM_errmsg "Disk read error, try again!"
```

19.3.4. form

Диалог ввода. Выводит смешанную форму ввода данных. Позволяет ввести одно или несколько текстовых полей данных в указанные переменные. Некоторые имена переменных ассоциирует с паролем, для них вводимые символы выводятся «*» (звёздочкой). Если данные были введены и нажата ENTER или кнопка «OK», возвращает 0. Если нажата Esc или кнопка «Cancel», возвращает не 0. Основан на **dialog --mixedform**.

Синтаксис:

```
IM_form <заголовок> <текст> <высота_текста> \
  <переменная1> <макс_размер_строки1> <подпись_поля1> \
  [<переменная2> <макс_размер_строки2> <подпись_поля2>...]
```

Пример использования:

```
IM_form "$title" "$text" 5    \
  server     64 "HTTP-server" \
  directory 128 "Directory"   \
  ||
  continue
```

19.3.5. gauge

Диалог вывода. Выводит «градусник», он же «ползунок» или «прогресс бар». Целое число от 0 до 100 должно быть подано на вход через stdout для определения процента проделанной работы. Хорошо согласуется с программой **pv**. Всегда возвращает 0. Основан на **dialog --gauge**.

Синтаксис:

```
echo <целое> | IM_gauge <заголовок> [<текст>]
```

Пример использования:

```
for i in $(seq 1 10); do
  echo "${i}0" | IM_gauge "[ Loading... ]"
  sleep 1
done
```

19.3.6. ponder

Диалог вывода. Выводит виджет «ожидание», изображающий неопределённое время происходящего процесса, работает независимо от основного кода программы. Параметры <пауза> и <шаг> при запуске определяют на сколько процентов будет автоматически продвигаться градусник спустя заданное время, т.е. задают частоту и скорость обновления виджета. Всегда возвращает 0. Основан на виджете **ponder**.

Синтаксис:

```
IM_ponder_start <заголовок> [[[<текст>] <пауза>] <шаг>]
...
IM_ponder_stop
```

Пример использования:

```
IM_ponder_start "[ Scanning disk... ]" \
    "Searching random bits on the disk for fill CRNG entropy..."
( find / -type f -print0 |
  xargs -0 grep "Linus Torvalds"
) >/tmp/Linus.txt 2>/dev/null
IM_ponder_stop
```

19.4. Расширение altboot

Можно расширять **altboot**, создавая новые модули. Первый шаг «*altboot*» и последний шаг «*liveboot*» используют «хуки» (см. следующий подраздел). Для определения контекста их использования можно в каталоге проекта **make-initrd-bootchain** дать команду:

```
git grep use_hooks
```

Чтобы понять, между чем и чем приткнуть свой новый код, можно запустить скрипт:

```
./mix-altboot
```

из корня исходного проекта и посмотреть в полученном подкаталоге **altboot-mixed/hooks/**, в каком порядке будут склеиваться «хуки».

19.4.1. Назначение «хуков»

Внутри initramfs «хуки» складываются в каталог **/lib/altboot/**:

- **add-methods.d** — регистрация методов загрузки на шаге «altboot»;
- **automatic.d** — все возможные названия аргументов в параметре automatic=... и их обработчики, если требуется, для каждого аргумента — свой файл с опциональным обработчиком;
- **forget-args.d** — вывод переменных initramfs, про которые можно забыть после их трансляции во внутренние параметры «шагов» на шаге «altboot»;
- **global-args.d** — пустые файлы с названиями параметров «шагов» altboot;
- **liveboot-init.d** — выполняется на этапе инициализации переменных шага «liveboot»;
- **liveboot-post.d** — выполняется после создания «живой» rootfs и всех оверлеев LiveCD;
- **liveboot-pre.d** — выполняется перед наложением оверлеев LiveCD над rootfs;
- **livecd-slice.d** — монтирование read-only слоёв LiveCD на шаге «liveboot»;
- **rw-overlay.d** — монтирование верхнего R/W-слоя на шаге «liveboot»;
- **translate.d** — транслируют аргументы из параметра automatic=... во внутренние параметры каждого отдельного шага.

20. Полезные ссылки

- https://github.com/osboot/make-initrd
- https://www.altlinux.org/Make-initrd
- https://www.altlinux.org/Propagator
- https://www.altlinux.org/Make-initrd-propagator
- https://lists.altlinux.org/pipermail/make-initrd/
- https://bugzilla.altlinux.org/30315#c29
- https://lists.altlinux.org/pipermail/devel/2018-April/204192.html
- https://packages.altlinux.org/ru/sisyphus/srpms/make-initrd-bootchain/
- https://github.com/klark973/make-initrd-bootchain
- https://youtu.be/C-NsPKvsCAw?t=6855
- http://0x1.tv/20210418F