

# От пропатора к altboot

Редакция 2 от 22.07.2024 (0.1.5-alt23)

## Содержание

Термины и определения.....	3
Достоинства bootchain + altboot.....	4
Недостатки bootchain + altboot.....	5
Модули bootchain.....	6
Быстрое знакомство с bootchain + altboot.....	7
Propagator + init-bottom ► bootchain + altboot.....	7
Pipeline ► bootchain.....	8
bootchain + altboot «с нуля».....	8
Методы загрузки altboot, пришедшие из пропатора.....	10
Базовые шаги модуля make-initrd-bootchain-altboot.....	12
Параметры пропатора и скрипта init-bottom.....	13
Общие аргументы в automatic=.....	14
Конфигурирование bootchain и altboot.....	15
Журналирование, отладка и тестирование.....	15
Совместимость с pipeline.....	16
Совместимость с propagator и init-bottom.....	17
Интерактивный режим работы.....	18
Постоянное хранилище сеансов LiveCD.....	19
Read-only слои LiveCD.....	20
Шаги bootchain.....	23
debug.....	23
fg.....	23
getimage.....	23
mountfs.....	24
noop.....	24
noretry.....	24
overlayfs.....	24
retry.....	25
rootfs.....	25
waitdev.....	25
Шаги altboot.....	27
altboot.....	27
checksum.....	28
cifs.....	29
copyfile.....	31
download.....	32
iso9660.....	36
liveboot.....	36

localdev.....	39
nfs.....	41
oemsetup.....	43
overlayroot.....	44
squashfs.....	45
Сопряжение входов/выходов.....	46
Разработчику bootchain/altboot.....	48
Общий вспомогательный код.....	48
interactive-sh-functions (модуль bootchain-interactive).....	48
bootchain-sh-functions (модуль bootchain-core).....	49
altboot-sh-functions (модуль bootchain-altboot).....	50
altboot-net-functions (модуль bootchain-waitnet).....	52
scandev-sh-functions (модуль bootchain-localdev).....	52
machine-info (модуль bootchain-core).....	53
Библиотека виджетов (модуль bootchain-interactive).....	54
choice.....	54
dlgmsg.....	55
errmsg.....	55
form.....	55
gauge.....	56
ponder.....	56
Расширение altboot.....	57
Назначение «хуков».....	57
Полезные ссылки.....	58

## Термины и определения

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

**make-initrd** — дистронезависимый инструмент для сборки образов initramfs, обеспечивающих начальную загрузку. make-initrd со всеми своими фидами 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 initramfs-образ универсального носителя (altinst, live, rescue) программы propagator и дополнительных скриптов. Загрузка таких дисков устроена иначе, нежели чем на обычной установленной rootfs. Вместо более гибкого run-time make-initrd в них работает propagator, а следом за ним отработывает скрипт **init-bottom** из описываемой фичи, в том числе, он обеспечивает 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 использованием фактора синхронизации событий и позволяет гибко комбинировать написанные «шаги»-скрипты.

**bootchain** — форк и дальнейшее развитие pipeline. На момент написания предлагает с десятков модулей и ещё не попал 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 Альт на «одиннадцатой платформе», обеспечивающая дополнительный функционал, слой совместимости с установщиком, alterator-netinst и более гибкие возможности дальнейшего расширения.

*Настоящий документ можно считать официальной документацией к **bootchain + altboot**. На момент написания (0.1.5-alt23) весь проект представляет собой 12 пакетов poarch, один SRPM, 170 исходных файлов на bash и make или 8650 строк кода, включая документацию в пакете.*

## Достоинства bootchain + altboot

- возможность интеграции 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 пропатора по загрузке с сервера SAMBA;
- появилась возможность создания R/W-хранилища сеансов LiveCD на быстрых дисках MMC/NVME;
- возрождён и улучшен утраченный функционал для добавления «забытых» модулей ядра при загрузке;
- более «разумное» взаимодействие с текстовой консолью: нет мелькания при быстрой загрузке;
- в отличие от **pipelined**, демон **chaind** может перейти на «передний план» и работать на конкретном терминале;
- в отличие от **pipelined**, демон **chiand** может менять логику «на лету», поддерживает циклы и условные переходы;
- **bootchain+interactive** вместе 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 Альт: m-p, alterator-netinst и другими 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/Installer/common/propagator>
- <https://www.altlinux.org/Make-initrd-propagator>

## 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 пропатора сеть сейчас конфигурируется только через /proc/cmdline, нет диалогов для её настройки;
- скрипты и диалоги **altboot** пока не рассчитаны на работу с IPv6, как и установщика Альт, 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 на половину 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.

## Модули bootchain

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 и rootfs;
- **make-initrd-bootchain-getimage** — обеспечивает метод загрузки образов по сети утилитой wget по протоколам HTTP и FTP;
- **make-initrd-bootchain-waitdev** — обеспечивает метод ожидания локального носителя по заданной спецификации;
- **make-initrd-bootchain-interactive** — обеспечивает интерактивное взаимодействие, предоставляет диалоговые виджеты;
- **make-initrd-bootchain-waitnet** — обеспечивает ожидание сети и экспорт сетевых настроек в stage2, используется всеми сетевыми методами загрузки (ftp, http, nfs, cifs) и является временным кодом;
- **make-initrd-bootchain-altboot** — весь общий функционал пропегатора и шаг, транслирующий его аргументы;
- **make-initrd-bootchain-localdev** — обеспечивает методы загрузки disk/cdrom, установку дополнительных модулей ядра;
- **make-initrd-bootchain-liverw** — обеспечивает дополнительный функционал для работы с R/W-сеансами LiveCD и Rescue;
- **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>

# Быстрое знакомство с bootchain + altboot

## Propagator + init-bottom ► bootchain + altboot

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

```
root=bootchain bootchain=fg,altboot automatic=... [ramdisk_size=...] 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` сразу две опции: **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}**.

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=...**, сквош грузится в RAM-диск;
- При наличии **automatic=type:iso,...**, ISO-образ грузится в TMPFS;
- Если **ramdisk\_size=...** не указан, сквош грузится в TMPFS.

При этом параметр **lowmem** не имеет значения. Для **других методов**:

- Если указан **lowmem**, ничего не грузится, монтируется сквош;
- Если указан **ramdisk\_size=...**, сквош грузится в RAM-диск;
- Если **ramdisk\_size=...** не указан, сквош грузится в TMPFS.

Также можно говорить, что параметр **automatic** обрёл несколько новых аргументов, утратив при этом аргументы, связанные с настройкой сети, типа «**network:...**» (они игнорируются). Например, можно задавать пользовательский таймаут: «**timeout:120**» или указывать тип загружаемого диска для методов http/ftp: «**type:iso**». Вместо параметра **propagator-debug** следует использовать **bc\_debug** и/или **rdshell**.

## Pipeline ► bootchain

Тем, кто уже пользовался фичей `make-initrd` «**pipeline**», достаточно подключить модуль **bootchain-waitdev** для локальной загрузки или **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 про совместимость с **pipeline**.

## bootchain + altboot «с нуля»

Тем, кто не знаком 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, итд, 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 пропатором 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.

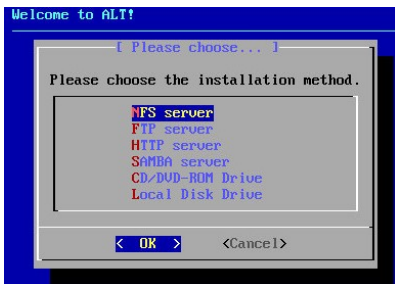
Демон **chaind** может не только менять «на лету» логику работы благодаря перегрузке списка «шагов», он также может в любой момент времени перейти на передний план и продолжить выполнение в **интерактивном режиме** на конкретном терминале. См. детали в соответствующем разделе и описании шага **fg**. 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/Installer/common/propagator>

## Методы загрузки altboot, пришедшие из пропатора

В **altboot** практически все параметры любого метода загрузки имеют свои значения по умолчанию, т.е. для загрузки достаточно указать в `/proc/cmdline` 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 монтируется с сервера NFS;
- **ftp** — образ скачивается целиком с сервера по протоколу FTP;
- **http** — образ скачивается целиком с сервера по протоколу HTTP;
- **cifs** — указанный ISO-образ либо каталог монтируется с сервера SAMBA;
- **cdrom** — загрузка непосредственно с локального носителя в формате ISO-9660 Hybrid с возможностью работы с R/W-слоем;
- **disk** — загрузка с локального диска, раздела либо лежащего на нём ISO-образа.

Методы загрузки **http/ftp** пока что не предназначены для работы с read-only слоями LiveCD, но если кому-то это нужно, добавить их поддержку не очень сложно. Оба метода обеспечиваются шагом **download**. Существует два режима работы с этими методами — старый вариант, когда путь на сервере рассматривается как каталог, из которого загружается файл со сквошем второй стадии, и новый вариант, когда путь на сервере рассматривается как ISO-образ, загружаемый целиком вместо сквоша. В пропаторе поддержка нового варианта тоже появилась, но позже, так что по параметрам могут быть отличия. В случае **altboot** для работы с новыми методами необходимо убрать из `/proc/cmdline` параметр загрузки **ramdisk\_size=...** и добавить к параметру **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 и правила `udev`. См. детали в описании «шага» **oemsetup**. Совместно с модулем **bootchain-liverw** оба метода поддерживают постоянное хранилище для R/W-сеансов LiveCD.

Оба метода имеют небольшие отличия от исходной реализации в пропегаторе. Если гибридный ISO-образ записан на R/W-носитель (USB-флеш, MMC-карту, итд), то метод загрузки **cdrom** выполняет загрузку не с диска, а с первого раздела, что позволяет, при необходимости, создать нужный R/W-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/Installer/common/propagator>
- <https://github.com/osboot/make-initrd/tree/master/features/network>

## Базовые шаги модуля `make-initrd-bootchain-altboot`

В результате декомпозиции пропегатора различные типовые действия, выполнявшиеся каждым его методом загрузки, превратились в отдельные «шаги» **altboot**, выполняющие какие-то действия 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.

- **altboot** — определяет метод загрузки и следующие «шаги» `bootchain`, транслирует аргументы пропегатора 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 (`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` внутри `initramfs` являются скриптами в `/lib/bootchain`.

## Параметры пропатора и скрипта `init-bottom`

Здесь перечислены все параметры, известные автору по исходникам `propagator` и скрипта `init-bottom`. В `altboot` для отладки вместо параметра `propagator-debug` используется параметр `bc_debug`. Также 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 пропатора;
- **hash** — хэш-сумма сквоша второй стадии загрузки, посчитанная по алгоритму SHA-256;
- **live** — практически «мёртвый» параметр, ни на что особо не влияет;
- **live\_rw** — предписывает использовать R/W-раздел для сеансов LiveCD/Rescue;
- **lowmem** — включает режим экономии памяти, монтируя образ непосредственно с носителя без предварительной загрузки в ОЗУ;
- **profile** — профиль сетевой загрузки NFS с поддержкой слоёв LiveCD;
- **propagator-debug** — даёт промежуточный shell перед переходом из `stage1` в `stage2`;
- **rescue** — практически «мёртвый» параметр, ни на что особо не влияет;
- **stagename** — название файла сквоша с `rootfs` второй стадии загрузки, по факту только: `altinst`, `rescue`, `live`;
- **testing** — «мёртвый» параметр, реально ни на что не влияет;
- **updatemodules** — «мёртвый» параметр, реально ни на что не влияет, хотя ранее использовался совместно с дискетами.

См. также:

- <https://www.altlinux.org/Installer/common/propagator>
- <https://www.altlinux.org/Make-initrd-propagator>

## Общие аргументы в `automatic=...`

Аргументы параметра `automatic=...` «общие» потому, что используются различными методами загрузки. Аргументы перечисляются в формате `arg1:value1,arg2:value2,...`. 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`);
- **fluid** — название проверочного файла в корне загрузочного раздела (новое в `altboot` и в `propagator`, но в `altboot` используется ещё и для других целей внутренне);
- **imgsize** — 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** — предел временного 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.

См. 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 (`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=dhcp \  
automatic=method:nfs,server:192.168.15.1,timeout:120,...
```

См. также:

- <https://www.altlinux.org/Installer/common/propagator>
- <https://github.com/osboot/make-initrd/tree/master/features/network>
- <https://bugzilla.altlinux.org/48448>

## Конфигурирование bootchain и altboot

В дистрибутивах ОС Альт необходимое конфигурирование для создания универсальных загрузочных носителей уже выполнено в т-р. Здесь представлена информация для разработчиков других дистрибутивов и пользователей **altboot**, желающих работать с ним в установленной Linux-системе на обычной rootfs, либо использующих LiveCD-оверлеи 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**. В нём можно переопределить дефолты и задать параметры, которые изначально не определены. Детали см. в примере **bootchain-doc/samples/bootchain-alt.conf**. Для добавления перечисленных файлов используется директива в **/etc/initrd.mk**:

```
PUT_FILES += <файлы>
```

## Журналирование, отладка и тестирование

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.

Расширенная отладка включается через **/proc/cmdline** параметром **bc\_debug**. В этом случае журнал копируется по тому же пути в stage2 после отработки последнего «шага», а если в новой rootfs нет **/var/log**, то копируется в **/dev/bootchain/**. Аналогом **propagator-debug** в самом **make-initrd** является параметр **rdshell** либо **stop=runit**, для его использования образ initramfs должен быть собран с соответствующей фичей «rdshell».

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.

## 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=...
```

Сходства и различия между двумя режимами перечислены в таблице:

Действие или значение	режим « <b>pipeline</b> »	режим « <b>bootchain</b> »
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 фиша «**pipeline**» перестала быть демоном, поскольку в самом make-initrd произошли архитектурные изменения с момента форка «bootchain». Также в код make-initrd и в код фиши «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 (это в планах).



## 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, 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 \
  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-bottom**. 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 «внахлёт» через каталоги `/root ($rootmnt)` и `/image ($OEM_CDROOT)`:

```
root=bootchain bootchain=fg,altboot automatic=... [lowmem] [ramdisk_size=...] [stagename=...]
```

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 т-р такой дефолт уже определён.

## 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>

## 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 **altboot**. 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 в созданный R/W-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](https://www.altlinux.org/Remount_rw)

## 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 R/W-сеансами, описанного в предыдущем разделе, тем более, если говорить о локальной загрузке с USB-флеш и подобных относительно медленных накопителей.

Ниже представлена таблица с принятыми умолчаниями в ОС Альт:

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

Переменные конфигурации определяются в **/etc/sysconfig/bootchain**, который должен попасть в образ initramfs. Имя профиля — это относительный путь к каталогу, задаваемый пропегаторным 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 тогда используются дефолты из таблицы. Для локальных методов загрузки **overlays** указывает на раздел с оверлеями.

Особенности реализации поиска профиля:

- **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 шаг **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.

Для всех методов загрузки в **altboot** используются 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 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 (см. предыдущий раздел).

Примеры использования:

```
root=bootchain bootchain=fg,altboot live \  
  automatic=method:cdrom,uuid:2021-05-31-01-09-58-00,overlays:SLICES \  
  stagenam=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=dhcp \  
  automatic=method:nfs, network:dhcp, server:192.168.15.1 \  
  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](https://www.altlinux.org/Remount_rw)

## 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 **chaid**.

### 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/chaid.log`.

### fg

Входит в подпакет **make-initrd-bootchain-core**, является расширением **pipeline**, в дополнительных параметрах не нуждается. Это внутренний псевдо-шаг главного цикла демона **chaid**, 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
```

### 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=dhcp \  
getimage=http://ftp.altlinux.org/pub/people/mike/iso/misc/vi-20140918-i586.iso \  
mountfs=rescue
```

## 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**.

## 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**.

## 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.

## 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.



## 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.

## 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.

## 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=dhcp \  
waitdev=MODEL=SAMSUNG_SSD_960_EVO_250GB \  
altboot_download=method=url;url=http://192.168.15.1/images/alt-ws11.img
```

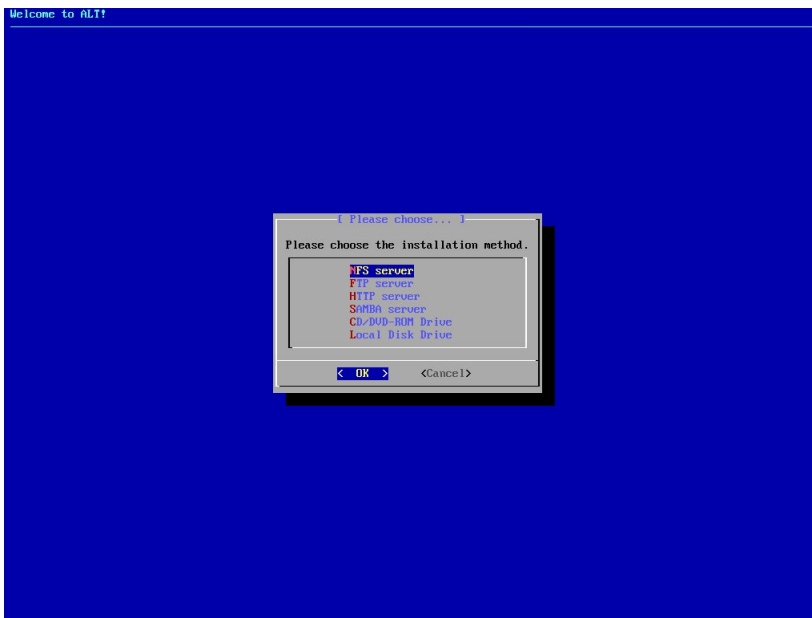
## Шаги altboot

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

## altboot

Этот «шаг» ходит в подпакет **make-initrd-bootchain-altboot**, определяет метод загрузки и следующие «шаги» **bootchain**, 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:

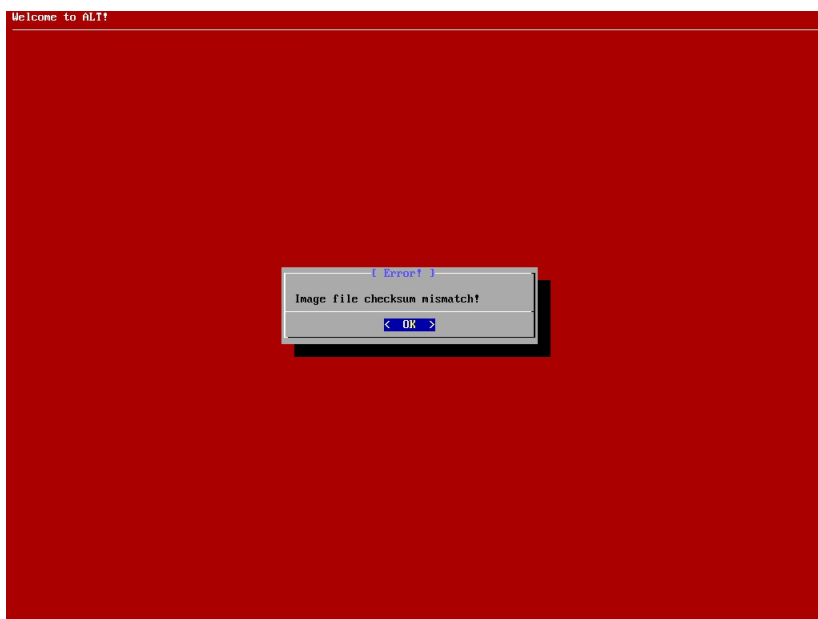
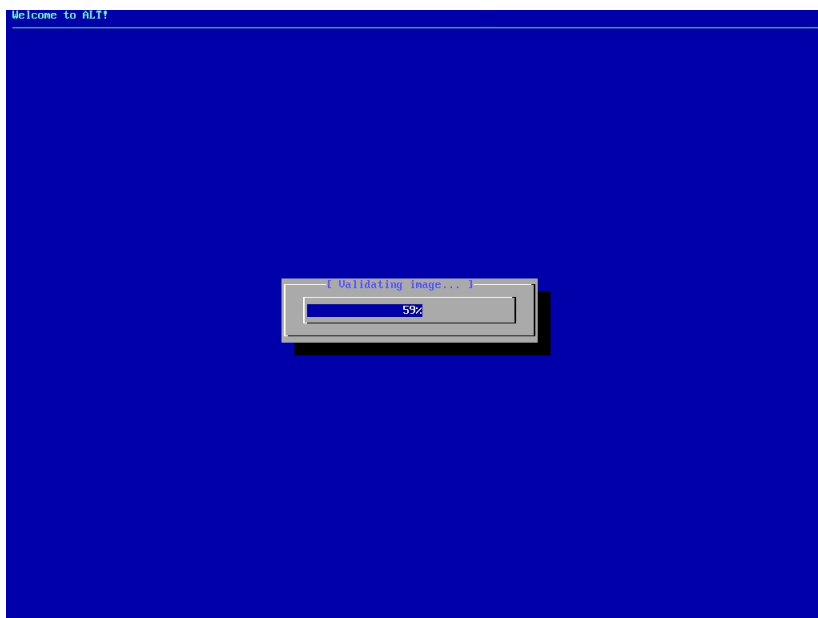
```
root=bootchain bootchain=fg,altboot lowmem stagename=rescue \  
automatic=method:cdrom,label:ALT_regular-rescue/x86_64
```

См. также:

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

## 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** — 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 считал 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`, **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:

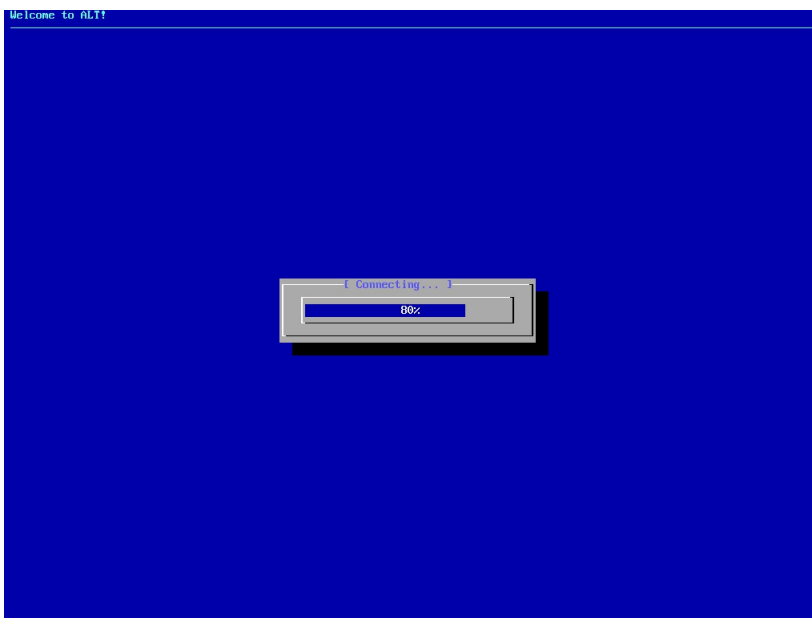
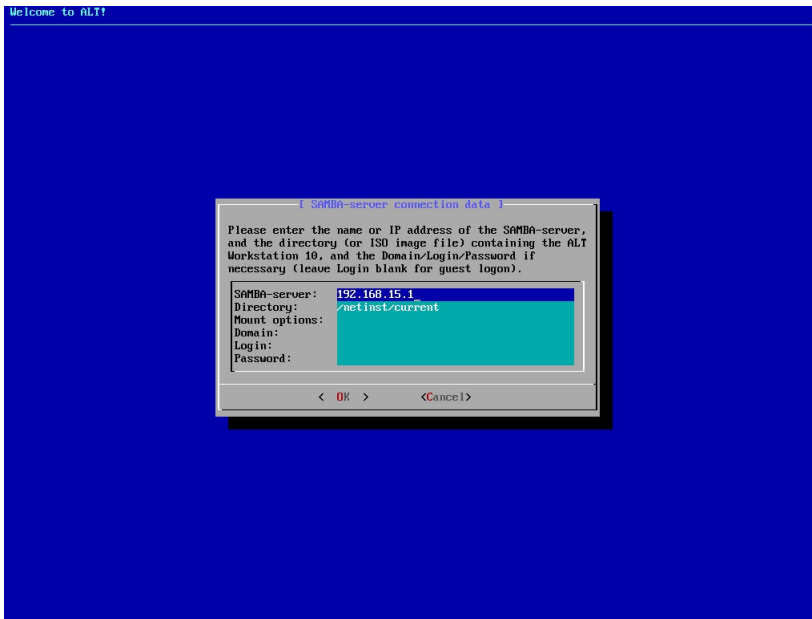
- <https://bugzilla.altlinux.org/30745>

## 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 пропегаторе, т.е. монтирует с **-o guest**. Если не указать **server**, то на его поиск выделяется дополнительное время, равное половине **timeout**.

В случае успеха загрузки в режиме совместимости с пропегатором 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,network:dhcp,server:192.168.15.1,directory:/netinst/current
```

См. также:

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

## 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 пропегатора не проходить «шаг» **download** дважды. Загружает уже доступный файл на смонтированной ФС 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/ramN`;
- **/dev/...** — использовать указанное блочное устройство;
- **step<N>** или **pipe<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 ...
```

## 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.atllinux.org  
Directory: /pub/people/mike/iso/misc/ui-20140918-i

< OK >      <Cancel>

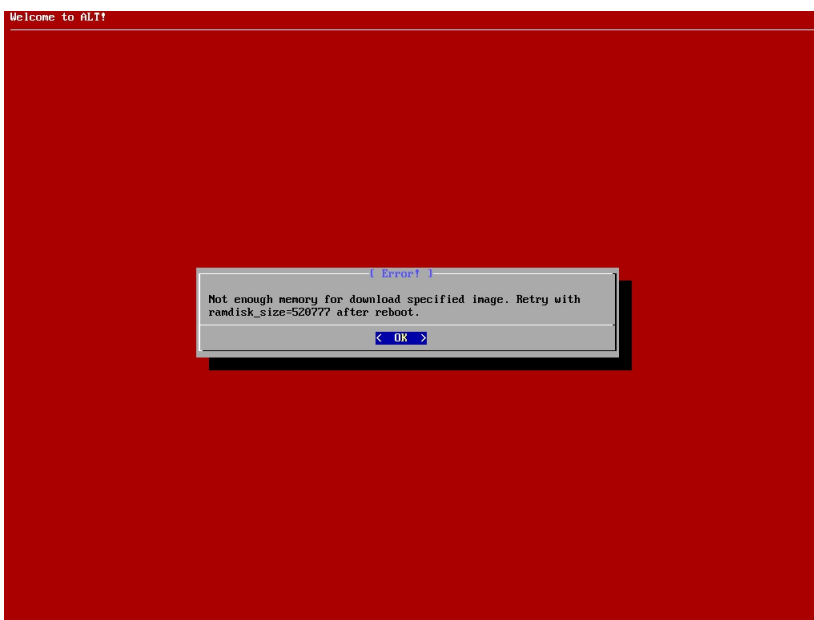
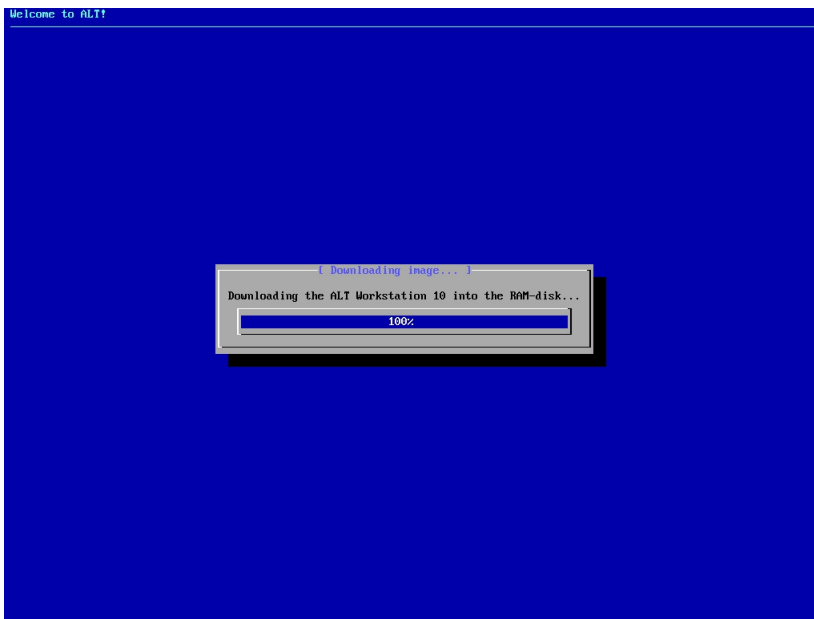
Welcome to GNU/Linux!

[ 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.atllinux.org  
Directory: /pub/people/mike/iso/misc/ui-20140918-i5  
Login:  
Password:

< OK >      <Cancel>



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

- **method** — метод загрузки altboot (см. выше);
- **to** — определяет место, куда загружать образ (см. ниже);
- **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 у сервера (новое в altboot);
- **timeout** — предел временного ожидания на соединение с HTTP/FTP-сервером (новое в altboot).

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

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

Параметр **curlopts** имеет глобальное значение для всех шагов **download**. Если набор дополнительных опций curl разделяется пробелами, можно прописать их в **CURL\_OPTS=...** в конфигурационном файле **/etc/sysconfig/bootchain**. Имя пользователя и пароль необязательны, при их отсутствии метод **download** повторяет поведение исходной реализации метода FTP в пропегаторе, т.е. монтирует 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 в режиме совместимости с пропегатором 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;

## 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 \  
  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>

## iso9660

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

## Примеры использования:

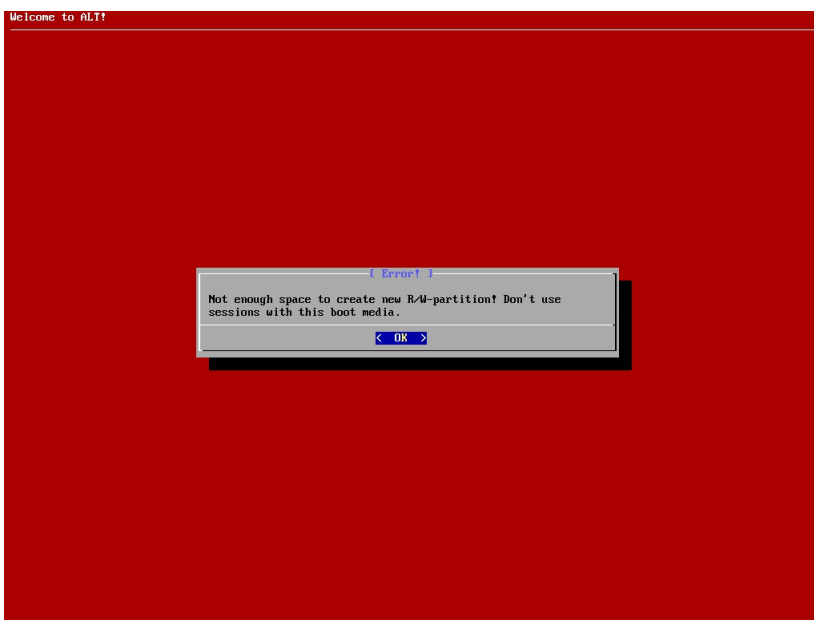
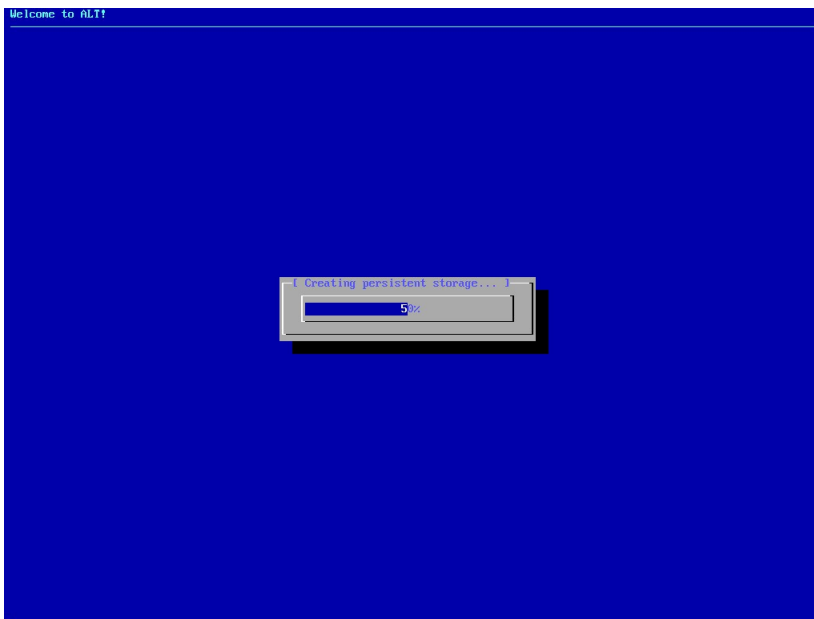
```
root=bootchain bootchain=fg,altboot ip=dhcp4 \  
  automatic=method:ftp,network:dhcp,server:192.168.15.1,directory:/srv/public/netinst/current
```

```
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
```

## 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 R/W-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 Live либо Rescue в режиме сохранения R/W-сеансов LiveCD с «локальными» методами загрузки. 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 R/W-оверлея, изменения потеряются;
- **UUID=...** — предписывает использовать раздел с заданным UUID в качестве R/W-оверлея;
- **LABEL=...** — предписывает использовать раздел с заданной меткой тома в качестве R/W-оверлея;
- **/dev/...** — предписывает использовать указанное блочное устройство в качестве R/W-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 искажаться на устройстве с указанной меткой тома;
  - **local\_profile** — будет использовано загрузочное устройство, на котором находится rootfs;
- **flags** — различные флаги создания LiveCD-оверлея, передаваемые данному «шагу»:
  - **live\_ro** — предписывает создавать оверлей над read-only LiveCD;
  - **live\_rw** — предписывает использовать R/W-хранилище сеансов LiveCD;
  - **rescue** — влияет только на сообщения в журнале при загрузке с Rescue;
- **timeout** — предел временного ожидания для поиска R/W-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=
```

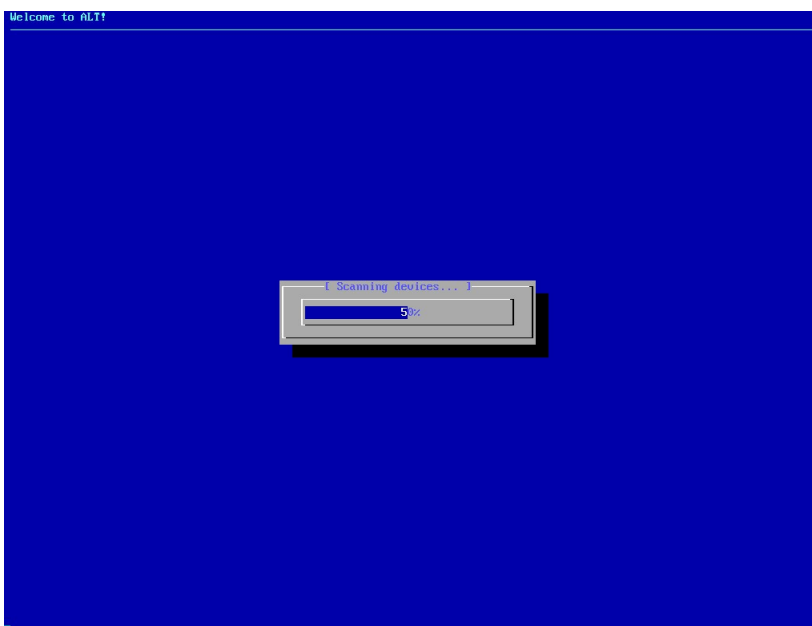
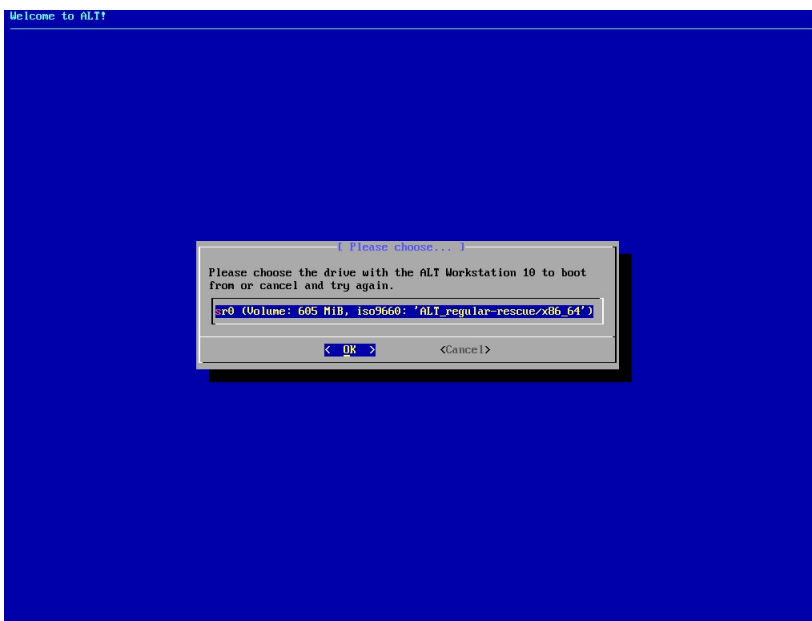
```
root=bootchain bootchain=fg,localdev,download,squashfs,liveboot,rootfs \
  altboot_localdev=method=cdrom;uuid=2021-05-31-01-09-58-00;directory=/rescue \
  altboot_download=to=RD;method=url;url=file:///rescue \
  altboot_liveboot=stagenamerescue;method=cdrom;overlays=SLICES;flags=live_ro,rescue
```

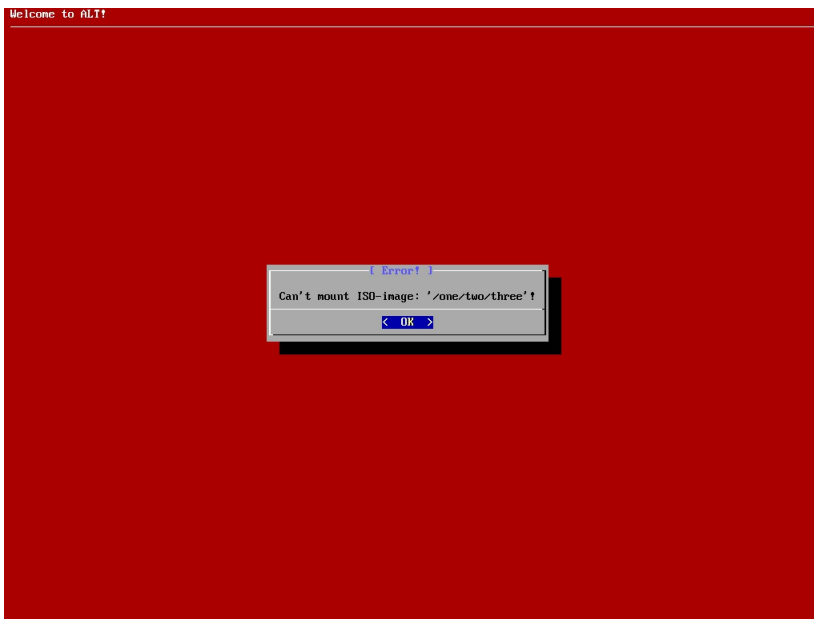
См. также:

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

## localdev

Интерактивный «шаг», входящий в подпакет **make-initrd-bootchain-localdev**, и обеспечивающий 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);
- **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;
- **directory** — путь к ISO-образу для метода disk, путь к сквошу для метода cdrom;
- **overlays** — определяет раздел, на котором находятся read-only слои LiveCD при локальной загрузке:
  - *метка тома* (LABEL) — профили и слои будут искать на устройстве с указанной меткой 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 в режиме совместимости с пропегатором «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 lowmem
```

```
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 \
  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>

## nfs

Входит в подпакет **make-initrd-bootchain-nfs**, обеспечивает 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**.

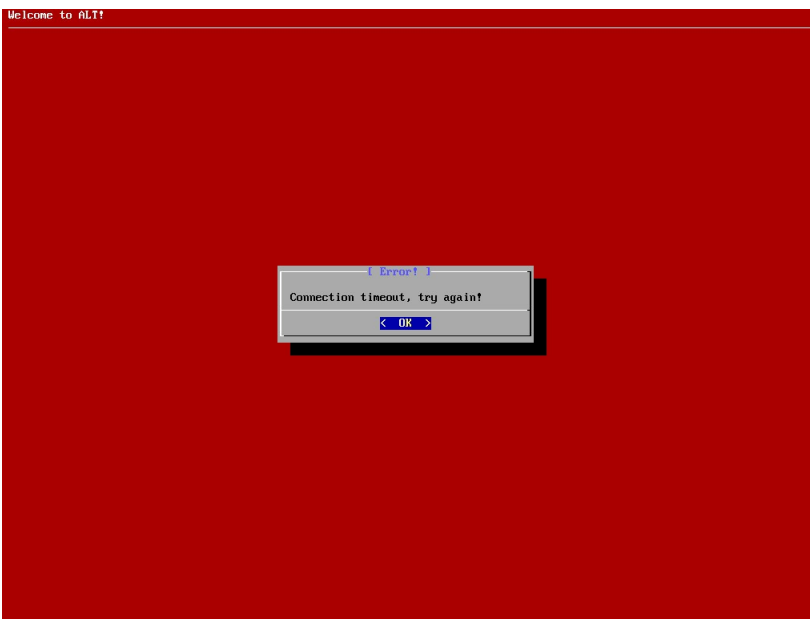
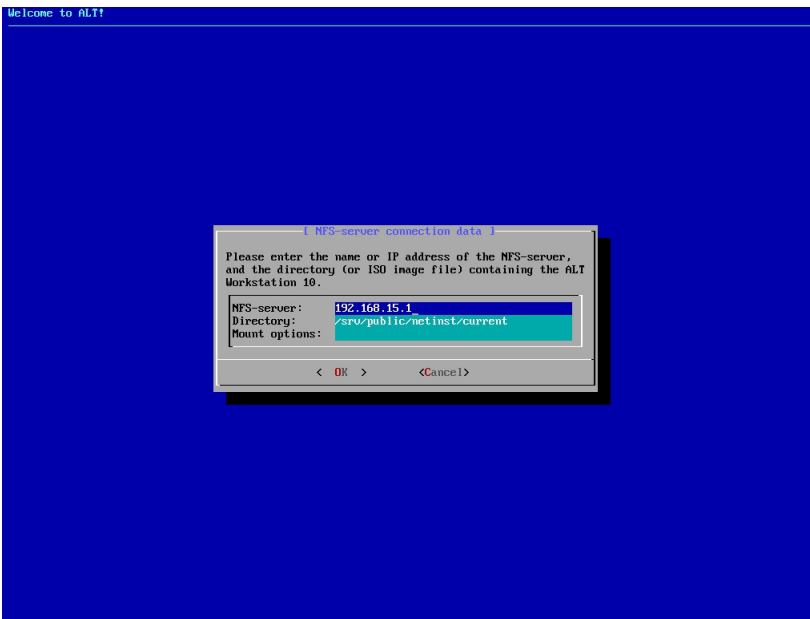
Параметры загрузки:

- **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 адрес шлюза;
- **directory** — путь к каталогу дистрибутива 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 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,network:dhcp,server:192.168.15.1,directory:/srv/public/netinst/current
```

## oemsetup

Интерактивный «шаг», входящий в подпакет **make-initrd-bootchain-localdev**, предназначенный для догрузки в initramfs «на лету» недостающего (модулей ядра, правил udev, итд). 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, данный шаг 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, а внутри каталога профиля ожидается файл «**initrd-update.img**», представляющий собой сжатый **gzip**'ом **cpio**-образ с файлами обновления initramfs. Название профиля оборудования 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 искаяться 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 **interactive-sh-functions**, **bootchain-sh-functions** и **altboot-sh-functions**. См. детали в разделе для разработчика.

## 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.

Примеры 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/cmdline** при каждой загрузке будет попадать что-то вроде:

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

## 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 сквошем в этом каталоге. В терминах пропегатора — 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 (особенность регистрации массивов в **make-initrd**).

Примеры использования:

```
root=bootchain bootchain=fg,altboot ip=dhcp 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=dhcp \
  altboot_download=to=RD;method=http;server=192.168.15.1;directory=/netinst/alt.iso \
  altboot_squashfs=/live \
  altboot_liveboot=stagename=live;method=http;flags=live_ro,live_rw
```

## 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 можно даже не учитывать в нумерации шагов: «**debug**», «**fg**», «**noretry**» и «**retry**».

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>
<bypass>	retry	<bypass>
<mp>	rootfs	<nothing>
<mp>/DEVNAME/dev	squashfs	<mp>
—	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.

# Разработчику bootchain/altboot

## Общий вспомогательный код

### interactive-sh-functions (модуль bootchain-interactive)

Общий код для обеспечения диалогов и интерактивного взаимодействия с пользователем. 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 поддержки символов псевдографики.

Функции:

- **IM\_is\_active()** — возвращает 0, если интерактивный режим доступен;
- **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.

## bootchain-sh-functions (модуль bootchain-core)

Общий код, используемый всеми шагами **bootchain** и **altboot**. Также пригоден и для шагов **pipeline**.

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 работает в режиме «pipeline», а не в «родном»;
- **\$BC\_DEBUG** — непустое значение, если включена расширенная отладка;
- **\$BC\_LOG\_VT** — если непустое значение, то порядковый номер ТТУ для вывода журнала;
- **\$BC\_LOGFILE** — путь к файлу журнала или устройство для вывода в него сообщений отладки;
- **\$BC\_DEVICE\_TIMEOUT** — дефолтный таймаут для всех «шагов»;
- **\$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 отладке;
- **enter()** — трассировка при расширенной отладке: вход в указанную функцию;
- **leave()** — трассировка при расширенной отладке: выход из указанной функции;
- **run()** — запуск внешней команды, при расширенной отладке команда попадёт в журнал;
- **fdump()** — вывод содержимого указанного файла при расширенной отладке;
- **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.

## altboot-sh-functions (модуль bootchain-altboot)

Общий код, 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 пропегатором;
- **\$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 пропегатора;
- **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.

## altboot-net-functions (модуль bootchain-waitnet)

Общий код, 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.

## scandev-sh-functions (модуль bootchain-localdev)

Весьма объёмный общий код 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: oem, disk или cdrom;
- **\$devices** — массив названий обнаруженных устройств.

Функции:

- **scan\_devices()** — сканирует устройства, заполняя переменную \$devices, также ищет соответствия;
- **device\_choice()** — выводит меню выбора устройств, в зависимости от указанного метода загрузки.

## machine-info (модуль bootchain-core)

Предназначена 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.

## Библиотека виджетов (модуль bootchain-interactive)

Скрипты виджетов попадают в каталог **/lib/IM-widgets** внутри формируемого образа initramfs. Библиотеку можно расширять, но следует иметь в виду, 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**. 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.

### 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
```

## dlgmsg

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

Синтаксис:

```
IM_dlgmsg <заголовок> <текст>
```

Пример использования:

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

## 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!"
```

## 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
```

## 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
```

## 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
```



## Расширение altboot

Можно расширять **altboot**, создавая новые модули. Первый шаг «altboot» и последний шаг «liveboot» используют «хуки» (см. следующий раздел). Для определения контекста их использования можно дать команду: **git grep use\_hooks**. Чтобы понять, между чем и чем приткнуть свой код, можно запустить скрипт **./mix-altboot** из корня исходного проекта и посмотреть в полученном подкаталоге **altboot-mixed/hooks/**, в каком порядке будут склеиваться «хуки».

### 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.

## 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>