Альт Платформа

ДОКУМЕНТАЦИЯ ALT LINUX TEAM

Альт Платформа 11.0

Документация

Руководство пользователя

Редакция апрель, 2026

Аннотация

Данное руководство соответствует текущему состоянию сведений, но какие-либо окончательные правки могли не попасть в него. В случае обнаружения ошибок и неточностей в руководство вносятся изменения.

Глава 1. Что такое Альт Платформа?

В этой главе рассматривается что такое Linux и Альт Платформа.

1.1. Что такое Альт Платформа

Альт Платформа (ALT Platform) — технологический комплекс для сборки прикладного программного обеспечения и выпуска продуктов ООО «Базальт СПО».

Альт Платформа основана на стабильной ветке репозитория Sisyphus, предназначена для разработки, тестирования, распространения, обновления и поддержки комплексных решений всех уровней — от встроенных устройств до серверов предприятий и дата-центров. Платформа создается и поддерживается ООО «Базальт СПО».

Альт Платформа обеспечивает интеграцию и поддержку программных продуктов различных разработчиков в среде операционных систем «Альт».

Состав Альт Платформа:

- дистрибутив ALT Platform Builder, содержащий следующие компоненты:
  - hasher — средство воспроизводимой сборки пакетов в изолированном окружении;
  - gear — инструмент для хранения исходных текстов в git и извлечения заданной версии;
  - mkimage — набор утилит для создания образов (в основном ISO);
  - mkimage-profiles — метапрофиль со множеством готовых компонентов и конфигураций образов;
  - alternator-mirror — модуль центра управления системой для зеркалирования репозиториев;
- инструменты для работы с контейнерами:
  - Podman — инструмент для запуска и управления контейнерами;
  - Buildah — инструмент для сборки OCI-совместимых контейнерных образов;
  - zot — инструмент для сборки OCI-совместимых контейнерных образов
  - regclient — набор утилит для работы с контейнерными реестрами (копирование, инспекция, управление образами);
  - crane — инструмент для взаимодействия с контейнерными реестрами (pull, push, копирование образов);

- репозиторий «Альт Платформа».

1.2. Что такое системы Альт

1.2.1. ALT Linux Team

Команда ALT Linux (https://www.altlinux.org/ALT_Linux_Team) — это интернациональное сообщество, насчитывающее более 300 разработчиков свободного программного обеспечения.

1.2.2. Сизиф

Sisyphus (https://packages.altlinux.org) — наш ежедневно обновляемый банк программ (часто называемый репозиторием). Поддерживаемая ALT Linux Team целостность Sisyphus, оригинальная технология сборки программ и утилита **`apt-get`** позволяют пользователям легко обновлять свои системы и быть в курсе актуальных новинок мира свободных программ.

Ежедневно изменяющийся репозиторий содержит самое новое программное обеспечение со всеми его преимуществами и недостатками (иногда ещё неизвестными). Поэтому, перед обновлением вашей системы из Sisyphus, мы советуем взвесить преимущества новых возможностей, реализованных в последних версиях программ, и вероятность возникновения неожиданностей в работе с ними (https://www.altlinux.org/Sisyphus_changes).

Разработка Sisyphus полностью открыта. У нас нет секретных изменений кода и закрытого тестирования с подписками о неразглашении. Всё, что мы сделали сегодня, завтра вы найдёте в сети. По сравнению с другими аналогичными банками программ (Debian unstable, Mandriva Cooker, PLD, Fedora), в Sisyphus есть немало самобытного. Особое внимание уделяется защите системы, локализации на русский язык, полноте и корректности зависимостей.

Название Sisyphus (Сизиф) заимствовано из греческой мифологии. С кропотливым Сизифом, непрерывно закатывающим в гору камни, команду ALT Linux Team объединяет постоянная работа над усовершенствованием технологий, заложенных в репозиторий.

Sisyphus, в первую очередь, — открытая лаборатория решений. Если вам это интересно, если вы хотите дополнить Sisyphus новыми решениями, если вы считаете, что можете собрать какую-то программу лучше — присоединяйтесь к проекту ALT Linux Team (https://www.altlinux.org/Join).

1.2.3. Что такое одиннадцатая платформа

Как уже говорилось ранее, Sisyphus является часто обновляемым репозиторием, скорее предназначенным для разработчиков. Решением для тех пользователей, которым стабильность и предсказуемость работы системы важнее расширенной функциональности (а это в первую очередь начинающие и корпоративные пользователи), являются дистрибутивы Альт. Такие дистрибутивы базируются на стабильном срезе репозитория Sisyphus. Эти срезы называются платформами.

Одиннадцатая платформа (p11) была создана в июне 2024 года и её поддержка продлится до июля 2027 года.

1.2.3.1. Основные новшества одиннадцатой платформы

- Одиннадцатая платформа основана на ядре Linux 6.12 (LTS) с расширенной поддержкой современного оборудования: процессорных архитектур Intel, включая Intel Meteor Lake, Intel Xeon Sapphire Rapids, AMD Ryzen 7000 (Zen 4) и EPYC Genoa; аппаратных интерфейсов — PCI Express Gen5, USB4, Thunderbolt 4, Wi-Fi 6/6E, NVMe 1.4/2.0; улучшенной поддержкой виртуализации;
- Программное обеспечение на одиннадцатой платформе использует обновленный OpenSSL 3.1. Платформа сохраняет поддержку OpenSSL 1.1 для совместимости с устаревшим ПО;
- Произошел переход на Python 3.12;
- Добавлены PHP 8.3 и 8.4;
- Системный интерпретатор сценариев /bin/sh теперь основан на Bash 5.2;
- Обновлены основные системные библиотеки и компиляторы: glibc 2.38, компилятор GCC 13 и LLVM/Clang 19;
- Пакет systemd обновлён до версии 255;
- Подсистема начальной загрузки установщика propagator заменена на altboot;
- Основным фрэймворком приложений графической подсистемы стал Qt6, с поддержкой Qt5 для обратной совместимости приложений. Qt6 в качестве основного стека также использует системный установщик;
- Включена поддержка нового Kerberos 1.21, полностью совместимого с Samba 4.20+. В дистрибутивах 11 платформы также доступны и ключевые изменения Samba 4.20+;
- Существенно обновлён Альтератор в качестве Центра Управления Системой — новый интерфейс, взаимодействие с D-Bus, модульная архитектура. Новый Альтератор поддерживает модули предыдущих версий;
- ALT Diagnostic Tool — графическая утилита диагностики ОС. ADT использует заранее подготовленный набор проверок, предоставляет возможность пользователю выполнить тесты без дополнительных привилегий и единый вид отчета по проверкам;
- Копидел — средство тиражирования установленной системы (*alterator-kopidel*);
- Платформа доступна для архитектур x86_64 и ARM64.

Глава 2. Установка дистрибутива

В этой главе рассматривается процесс установки дистрибутива ALT Platform Builder.

2.1. Создание загрузочного flash-диска

Предупреждение

Запись образа дистрибутива на flash-диск приведёт к изменению таблицы разделов на носителе, таким образом, если flash-диск выполнил функцию загрузочного\установочного устройства и требуется вернуть ему функцию переносного накопителя данных, то необходимо удалить все имеющиеся разделы на flash-диске и создать нужное их количество заново.

Для восстановления совместимости flash-диска с операционными системами семейства Windows может понадобиться также пересоздание таблицы разделов (например, при помощи parted). Нужно удалить таблицу GPT и создать таблицу типа msdos. Кроме того, должен быть только один раздел с FAT или NTFS.

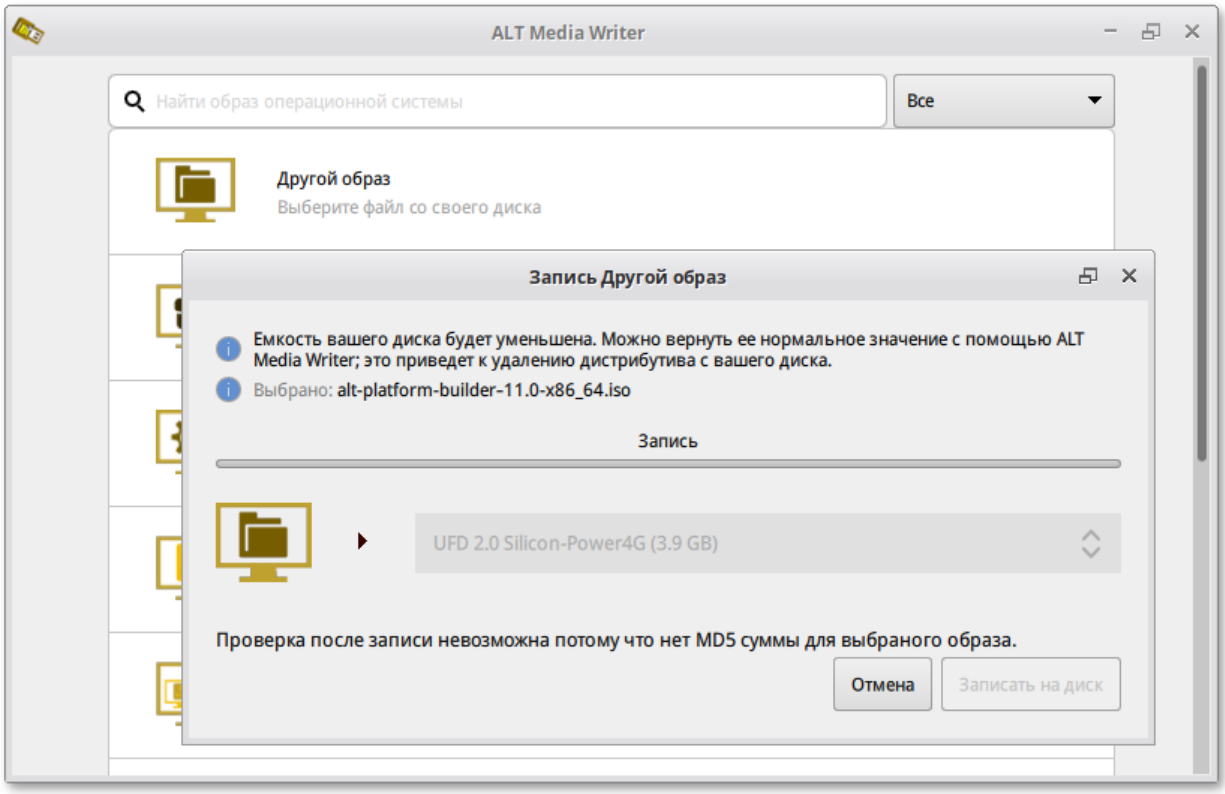
Для создания загрузочного flash-диска понадобится файл ISO-образа установочного диска с дистрибутивом. ISO-образы установочных дисков являются гибридными (Hybrid ISO/IMG), что позволяет производить установку, записав такой образ на flash-накопитель.

2.1.1. В операционной системе Windows

Для создания загрузочного flash-диска под операционной системой MS Windows используйте специальные программы: ALT Media Writer, Win32 Disk Imager, HDD Raw Copy Tool и другие.

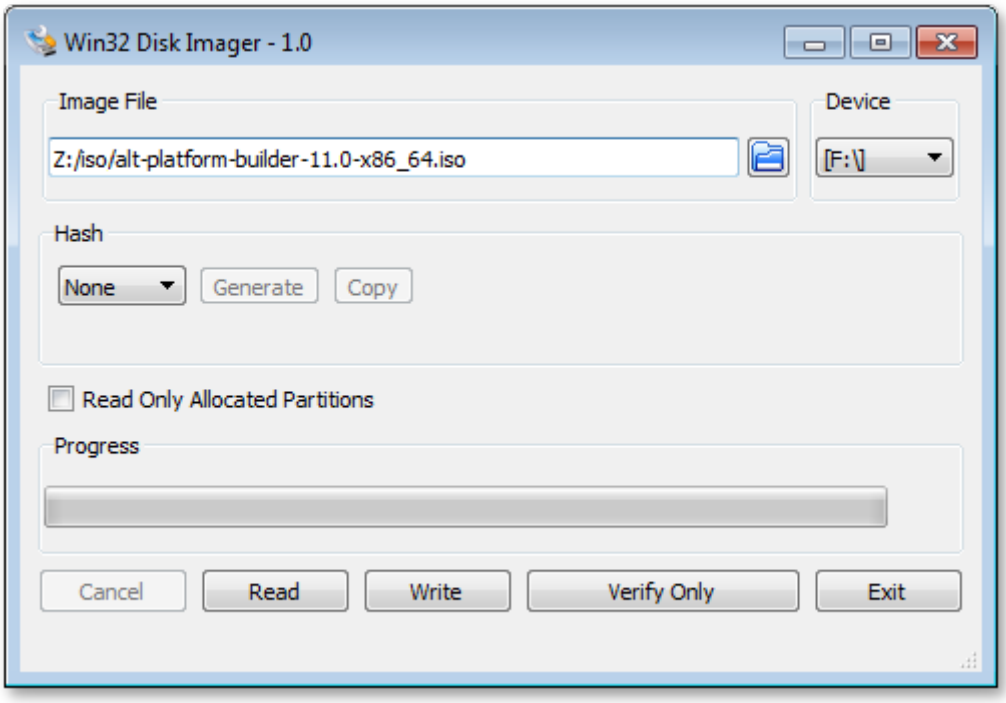
ALT Media Writer — инструмент, позволяющий записывать образы ALT на портативные накопители, такие как flash-диски. Он может автоматически загружать образы из интернета и записывать их. Для записи образа на flash-диск необходимо:

- скачать и установить **ALT Media Writer**;
- скачать образ дистрибутива;
- вставить flash-диск в USB-разъем (размер flash-диска должен быть не меньше размера скачанного образа диска);
- запустить **ALT Media Writer**;
- выбрать пункт **Другой образ** и в появившемся окне выбрать ISO-образ дистрибутива;
- выбрать устройство (flash-диск);
- нажать кнопку **`Записать на диск`**:



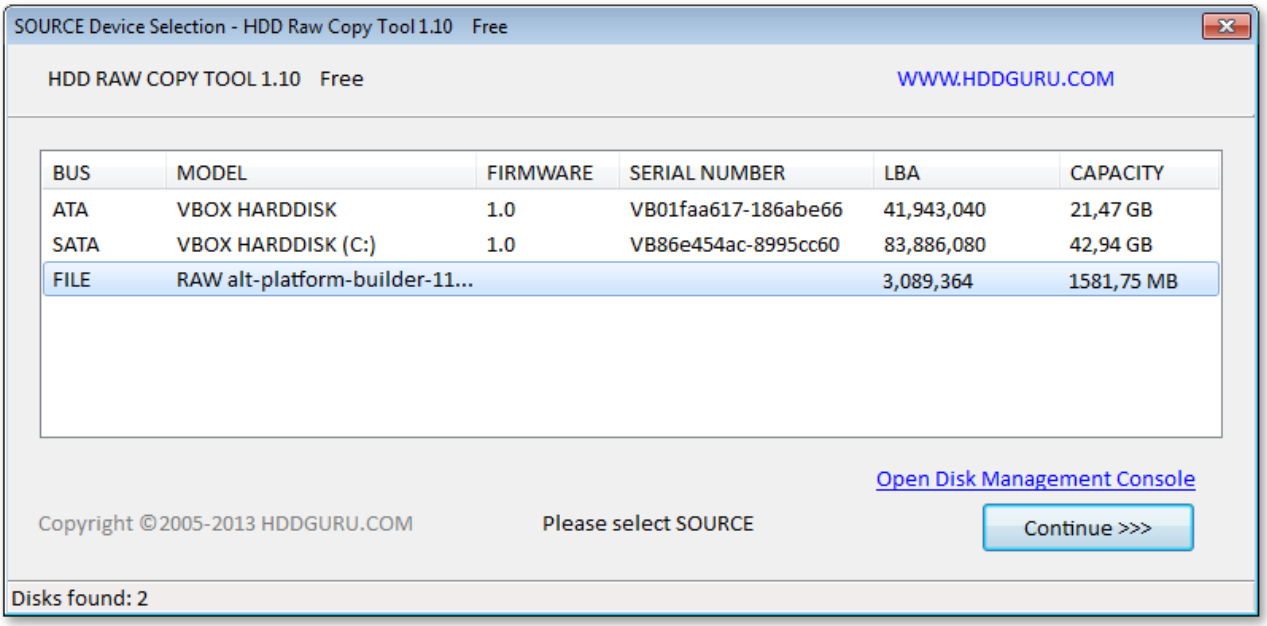
Инструкция для записи образа в программе **Win32 Disk Imager**:

- скачать и установить программу Win32 Disk Imager;
- скачать образ дистрибутива;
- вставить flash-диск в USB-разъем (размер flash-диска должен быть не меньше размера скачанного образа диска);
- запустить **Win32 Disk Imager**;
- в появившемся окне выбрать ISO-образ дистрибутива, выбрать устройство (flash-диск):

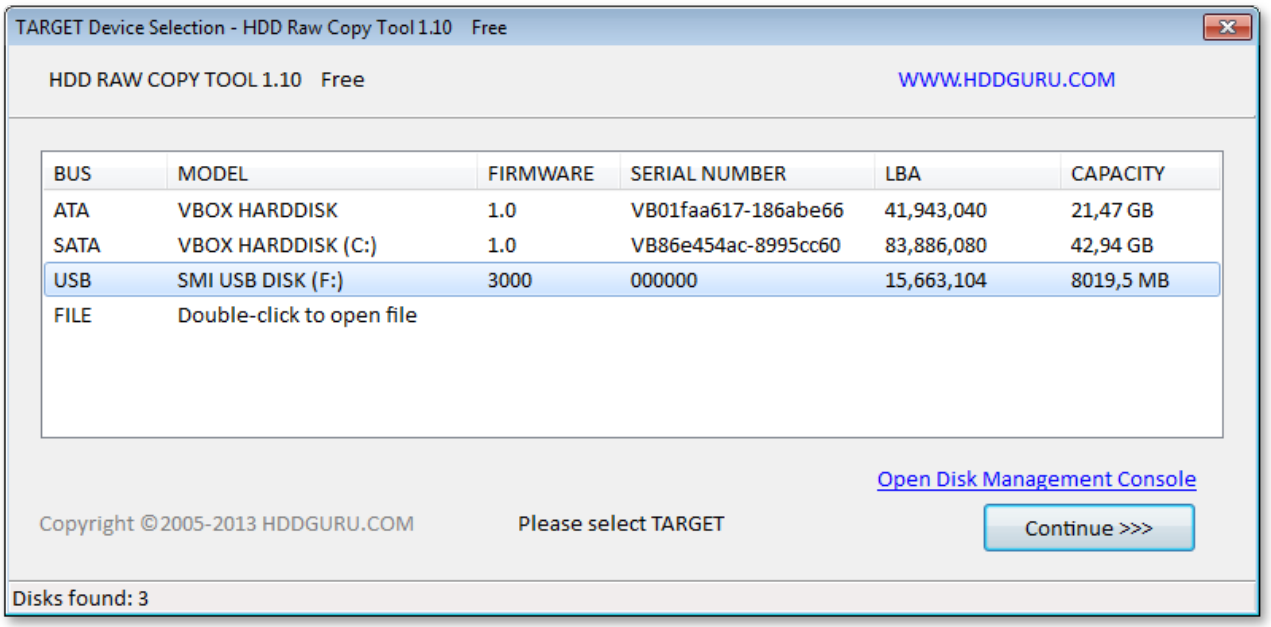


- нажать кнопку **`Write`** для записи образа на flash-диск.

Для записи образа на flash-диск подойдёт и утилита HDD Raw Copy Tool. На первом шаге нужно выбрать файл с образом диска:



На втором шаге нужно выбрать flash-диск, на который будет записан образ:



Предупреждение

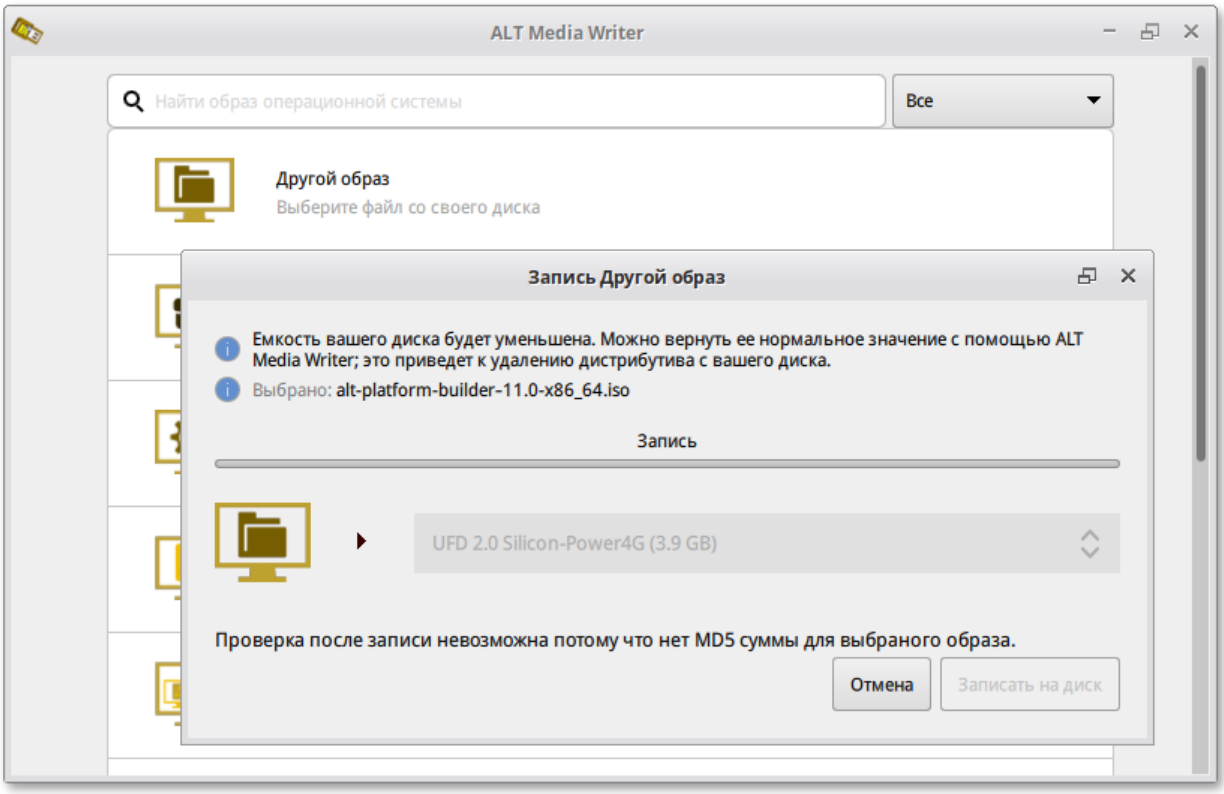
Будьте внимательны при указании имени USB-устройства — запись образа по ошибке на свой жёсткий диск приведёт к почти гарантированной потере данных на нём!

После проверки правильности выбранных параметров и нажатия кнопки **`Continue`** можно приступать к записи, нажав кнопку **`START`**. По успешному завершению записи окно с индикацией процесса записи закроется, после чего можно закрыть и окно самой программы.

2.1.2. В операционной системе Linux

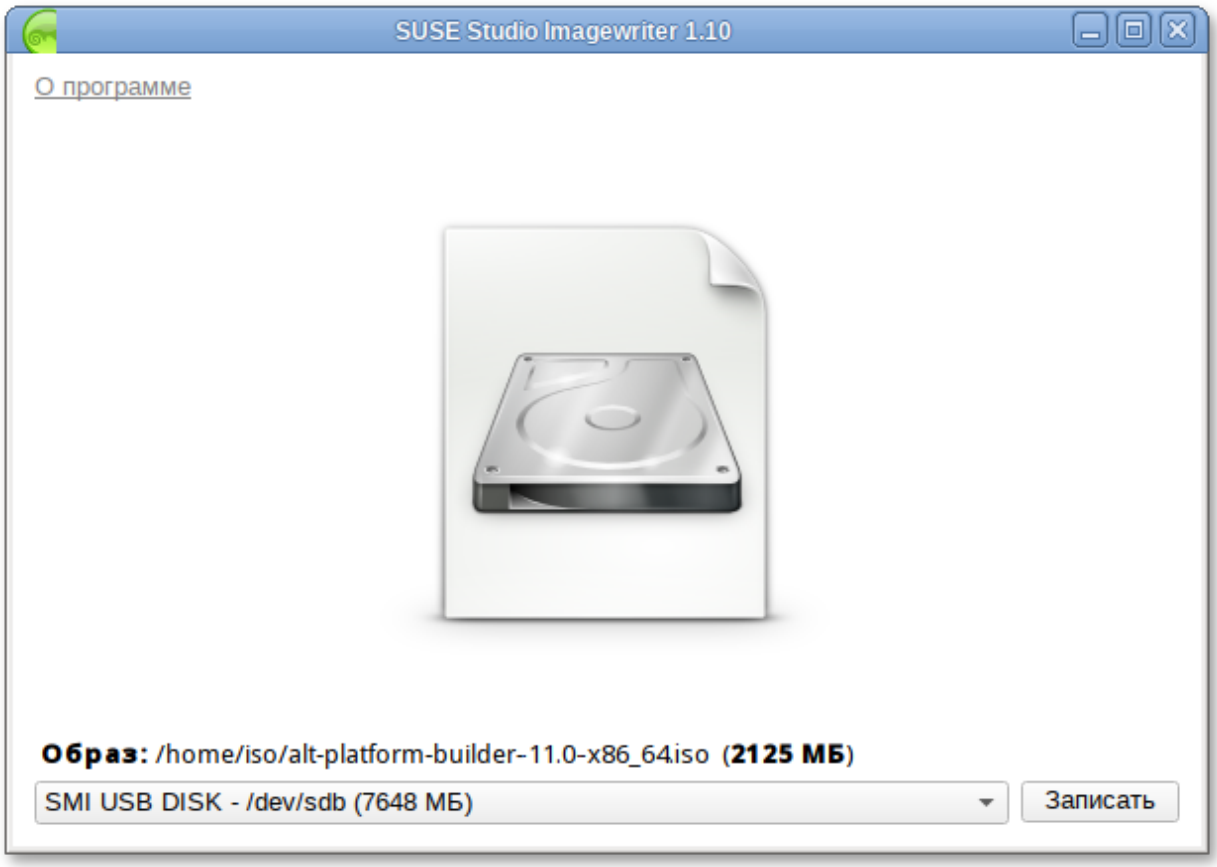
Для записи образа на flash-диск можно воспользоваться одной из программ с графическим интерфейсом:

- ALT Media Writer (*altmediawriter*):



ALT Media Writer может автоматически загружать образы из интернета и записывать их, при необходимости извлекая сжатые образы (img.xz).

SUSE Studio Imagewriter (*imagewriter*):



Предупреждение

Будьте внимательны при указании имени USB-устройства — запись образа по ошибке на свой жёсткий диск приведёт к почти гарантированной потере данных на нём!

Предупреждение

Не добавляйте номер раздела, образ пишется на flash-диск с самого начала!

Для записи установочного образа можно воспользоваться утилитой командной строки dd:

```
# dd oflag=direct if=<файл-образа.iso> of=/dev/sdX bs=1M status=progress;sync
```

где <файл-образа.iso> — образ диска ISO, а **/dev/sdX** — устройство, соответствующее flash-диску.

Для удобства показа прогресса записи можно установить пакет *pv* и использовать команду:

```
# pv <файл-образа.iso> | dd oflag=direct of=/dev/sdX bs=1M;sync
```

где <файл-образа.iso> — образ диска ISO, а **/dev/sdX** — устройство, соответствующее flash-диску.

Просмотреть список доступных устройств можно командой **lsblk** или (если такой команды нет) **blkid**.

Например, так можно определить имя flash-диска:

```
$ lsblk | grep disk
sda      8:0    0 931,5G  0 disk
sdb      8:16   0 931,5G  0 disk
sdc      8:32   1   7,4G  0 disk
```

flash-диск имеет имя устройства sdc.

Затем записать:

```
# dd oflag=direct if=/home/iso/alt-platform-builder-11.0-x86_64.iso of=/dev/sdc
bs=1M status=progress; sync
```

или, например, так:

```
#
pv /home/iso/alt-platform-builder-11.0-x86_64.iso | dd oflag=direct of=/dev/sdc
bs=1M;sync
dd: warning: partial read (524288 bytes); suggest iflag=fullblock
3GiB 0:10:28 [4,61MiB/s] [=================================>  ] 72% ETA 0:04:07
```

Предупреждение

Не извлекайте flash-диск, пока образ не запишется до конца! Определить финал процесса можно по прекращению моргания индикатора flash-диска либо посредством виджета **Безопасное извлечение съемных устройств**.

2.1.3. В операционной системе OS X

В операционной системе OS X для создания загрузочного flash-диска можно использовать команду:

```
sudo dd if=alt-platform-builder-11.0-x86_64.iso of=/dev/rdiskX bs=10M
sync
```

где **alt-platform-builder-11.0-x86_64.iso** — образ диска ISO, а /dev/rdiskX — flash-диск.

Просмотреть список доступных устройств можно командой:

```
diskutil list
```

Предупреждение

Будьте внимательны при указании имени USB-устройства — запись образа по ошибке на свой жёсткий диск приведёт к почти гарантированной потере данных на нём!

2.1.4. Проверка целостности записанного образа

Для проверки целостности записанного образа необходимо выполнить следующие шаги:

- определить длину образа в байтах:

```
$ du -b alt-platform-builder-1f1.0-x86_64.iso | cut -f1
2228344832
```

- посчитать контрольную сумму образа (или просмотреть контрольную сумму образа из файла MD5SUM на сервере FTP):

```
$ md5sum alt-platform-builder-11.0-x86_64.iso
1044e5aa9ecf6ee5a87fb990c8123af8  alt-platform-builder-11.0-x86_64.iso
```

- подсчитать контрольную сумму записанного образа на DVD или USB Flash (выполняется под правами пользователя root):

```
# head -c 2228344832 /dev/sdd | md5sum
1044e5aa9ecf6ee5a87fb990c8123af8
```

где размер после -c — вывод в п.1, а /dev/sdd — устройство DVD или USB Flash, на которое производилась запись.

2.2. Сохранение данных и меры предосторожности

Если необходимо установить ОС Альт Платформа и при этом сохранить уже установленную на компьютере операционную систему (например, другую версию GNU/Linux или Microsoft Windows), то нужно обязательно позаботиться о подготовке компьютера к установке второй системы и о сохранении ценных для вас данных.

Если у вас нет загрузочного диска для уже установленной системы, создайте его. В случае прерванной установки ОС Альт Платформа или неправильной настройки загрузчика, вы можете потерять возможность загрузиться в вашу предыдущую ОС.

Если на диске, выбранном для установки ОС Альт Платформа, не осталось свободного раздела, то программа установки должна будет изменить размер существующего раздела. От этой операции могут пострадать ваши данные, поэтому предварительно надо сделать следующие действия:

- Выполнить проверку раздела, который вы собираетесь уменьшать. Для этого воспользуйтесь соответствующим программным обеспечением (далее — ПО), входящим в состав уже установленной ОС. Программа установки Альт Платформа может обнаружить некоторые очевидные ошибки при изменении размера раздела, но специализированное ПО предустановленной ОС справится с этой задачей лучше.

- Выполнить дефрагментацию уменьшаемого раздела в целях повышения уровня безопасности данных. Это действие не является обязательным, но мы настоятельно рекомендуем его произвести: изменение размера раздела пройдёт легче и быстрее.

Предупреждение

Полной гарантией от проблем, связанных с потерей данных, является резервное копирование!

2.3. Начало установки: загрузка системы

Примечание

В данной инструкции рассмотрена установка системы в режиме UEFI. Особенности установки в режиме legacy отображены в примечаниях.

2.3.1. Загрузка с установочного носителя

Для того чтобы начать установку ОС Альт Платформа, достаточно загрузиться с носителя, на котором записан дистрибутив.

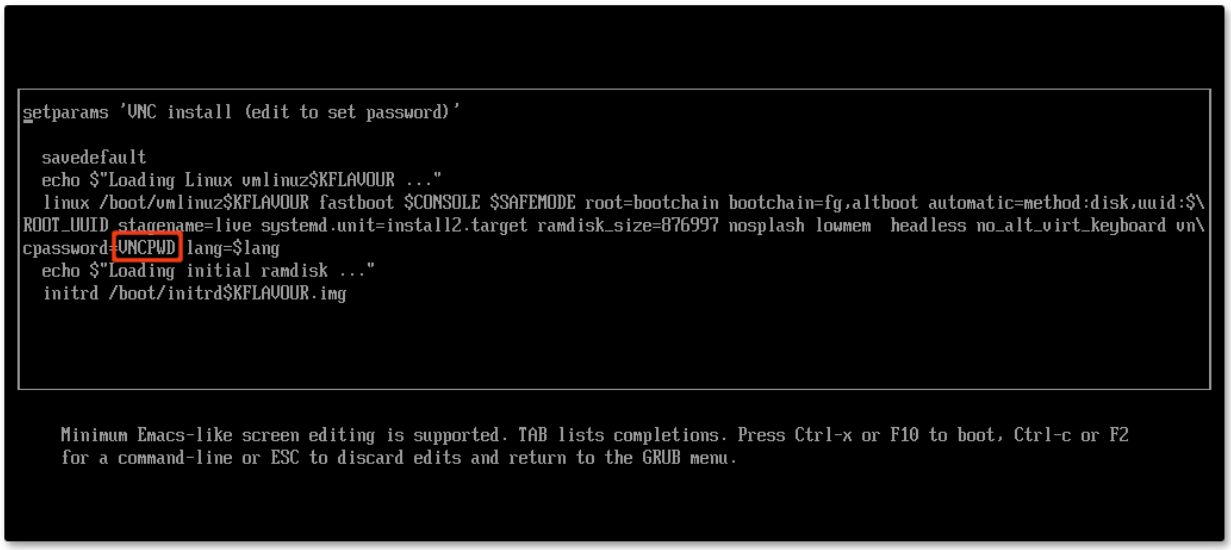
Примечание

Предварительно следует включить в BIOS опцию загрузки с оптического привода или с USB-устройства.

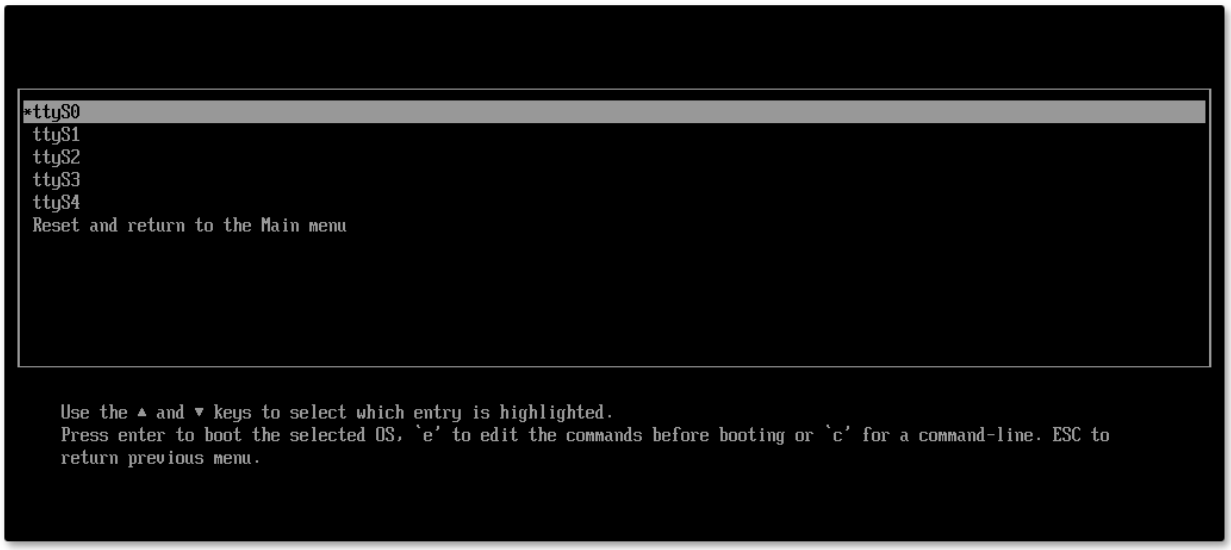
В большинстве случаев указание способа входа в BIOS отображается на вашем мониторе непосредственно после включения компьютера. Способ входа в меню BIOS и информация о расположении настроек определяется производителем используемого оборудования. За информацией можно обратиться к документации на ваше оборудование.

Загрузка с установочного диска или специально подготовленного USB-flash-накопителя начинается с меню, в котором перечислено несколько вариантов загрузки:

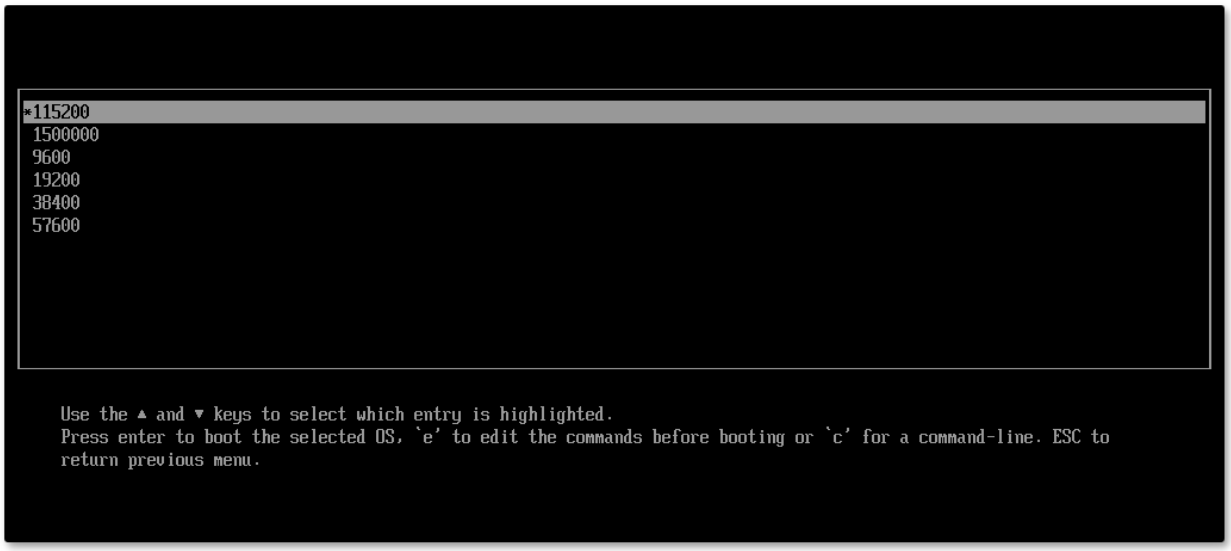
- **Install ALT Platform Builder 11.0** — установка операционной системы;
- **VNC install ALT Platform Builder 11.0 (edit to set password and connect here)** — установка по VNC с соединением в сторону устанавливаемой машины. Параметры установки по VNC передаются как параметры ядра. Нажатие клавиши **E** позволяет задать пароль (по умолчанию — VNCPWD):



- **Change serial console** — позволяет выбрать последовательный порт для консольного подключения (например, COM1/ttyS0):



и задать скорость передачи данных:

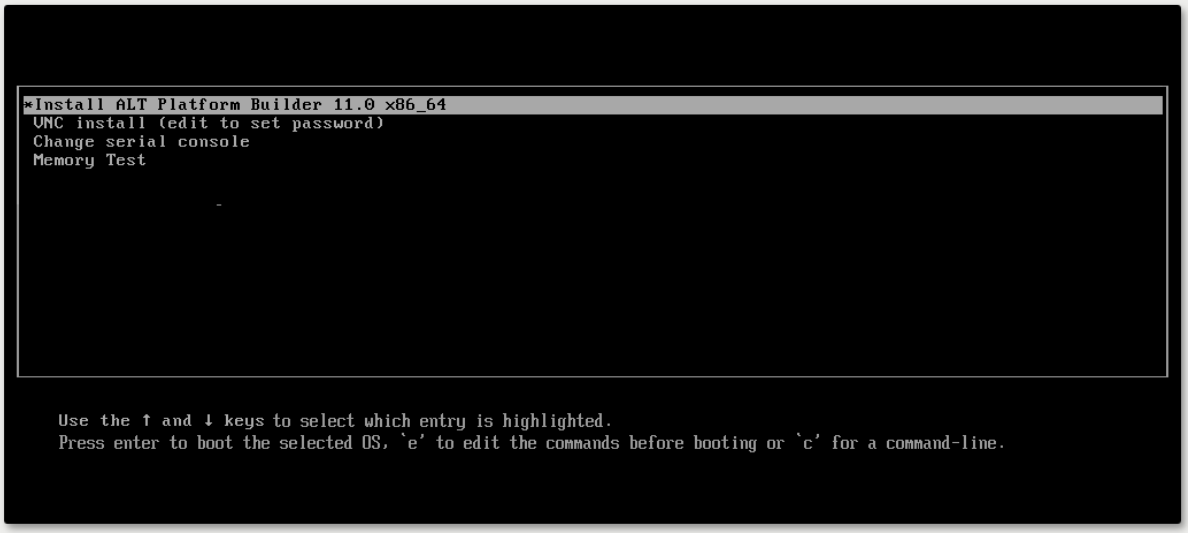


- **Memory Test (may not work with Secure Boot)** — проверка целостности оперативной памяти. Процесс диагностики заключается в проведении нескольких этапов тестирования каждого отдельного модуля ОЗУ (данный процесс будет выполняться бесконечно, пока его не остановят, необходимо дождаться окончания хотя бы одного цикла проверки).

- **UEFI Firmware Settings** — позволяет получить доступ к настройкам UEFI.

Примечание

Начальный загрузчик в режиме Legacy:



Примечание

Мышь на этом этапе установки не поддерживается. Для выбора опций установки и различных вариантов необходимо использовать клавиатуру.

Нажатием клавиши **E** можно вызвать редактор параметров текущего пункта загрузки. Если система настроена правильно, то редактировать их нет необходимости.

Чтобы начать процесс установки, нужно клавишами перемещения курсора **вверх** и **вниз** выбрать пункт меню **Install ALT Platform Builder** и нажать **Enter**.

Начальный этап установки не требует вмешательства пользователя: происходит автоматическое определение оборудования и запуск компонентов программы установки. Сообщения о происходящем на данном этапе можно просмотреть, нажав клавишу **ESC**.

Примечание

В начальном загрузчике установлено небольшое время ожидания: если в этот момент не предпринимать никаких действий, то будет загружена та система, которая уже установлена на жестком диске. Если вы пропустили нужный момент, перезагрузите компьютер и вовремя выберите пункт **Install ALT Platform Builder**.

2.4. Последовательность установки

До того как будет произведена установка базовой системы на жёсткий диск, программа установки работает с образом системы, загруженным в оперативную память компьютера.

Если инициализация оборудования завершилась успешно, будет запущен графический интерфейс программы-установщика. Процесс установки разделён на шаги. Каждый шаг посвящён настройке или установке определённого свойства системы. Шаги нужно проходить последовательно. Переход к следующему шагу происходит по нажатию кнопки **Далее**. При помощи кнопки **Назад**, при необходимости, можно вернуться к уже пройденному шагу и изменить настройки. Однако возможность перехода к предыдущему шагу ограничена теми шагами, в которых нет зависимости от данных, введённых ранее.

Если по каким-то причинам возникла необходимость прекратить установку, необходимо нажать кнопку <Reset> на корпусе системного блока компьютера.

Примечание

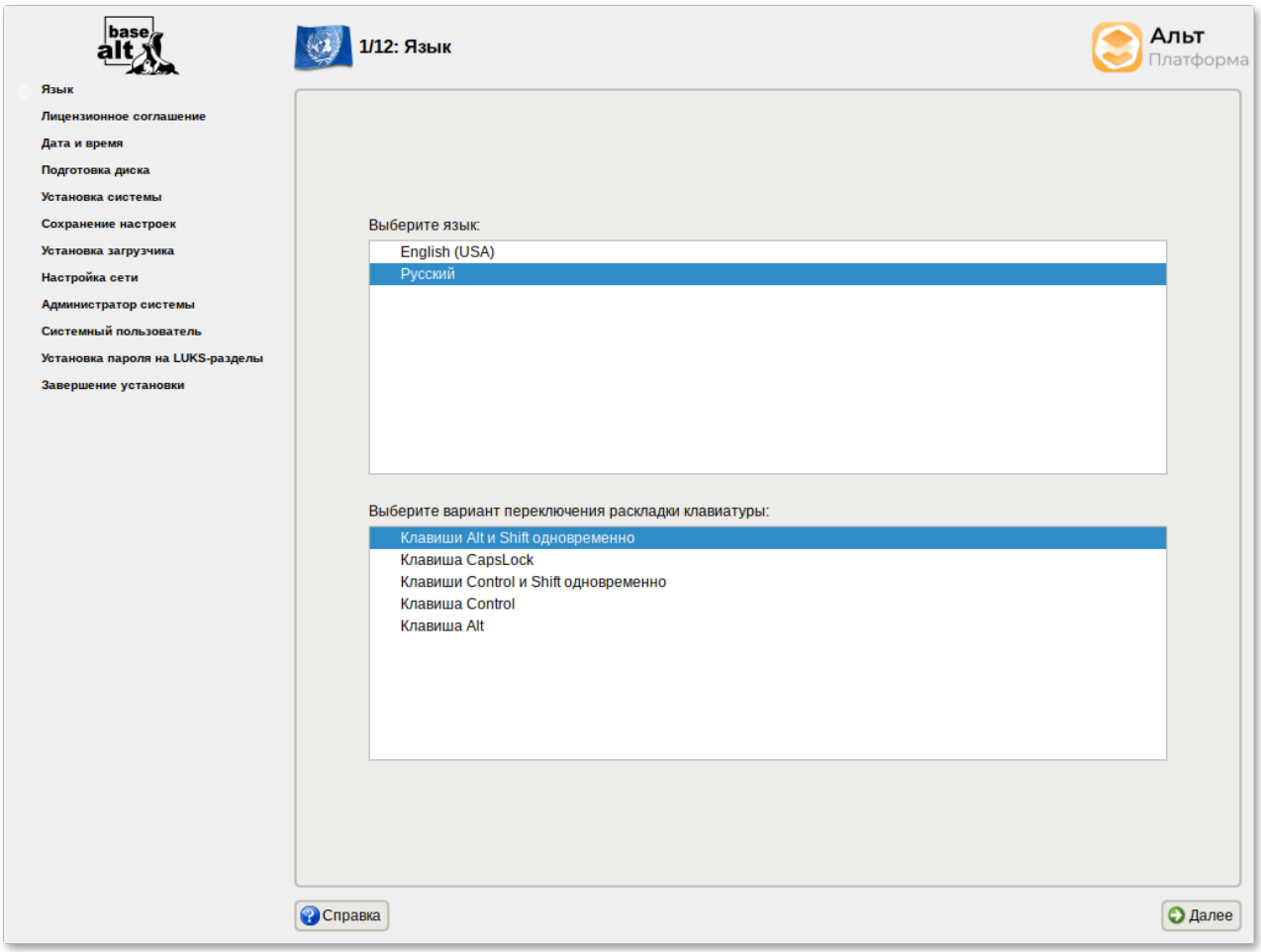
Совершенно безопасно выполнить отмену установки только до шага Подготовка диска, поскольку до этого момента не производится никаких изменений на жёстком диске. Если прервать установку между шагами Подготовка диска и Установка загрузчика, существует вероятность, что после этого с жёсткого диска не сможет загрузиться ни одна из установленных систем (если такие имеются).

Технические сведения о ходе установки можно посмотреть, нажав **Ctrl**+**Alt**+**F1**, вернуться к программе установки — **Ctrl**+**Alt**+**F7**. По нажатию **Ctrl**+**Alt**+**F2** откроется отладочная виртуальная консоль.

Каждый шаг сопровождается краткой справкой, которую можно вызвать, щёлкнув кнопку **Справка** или нажав клавишу **F1**.

Нажатие на кнопку

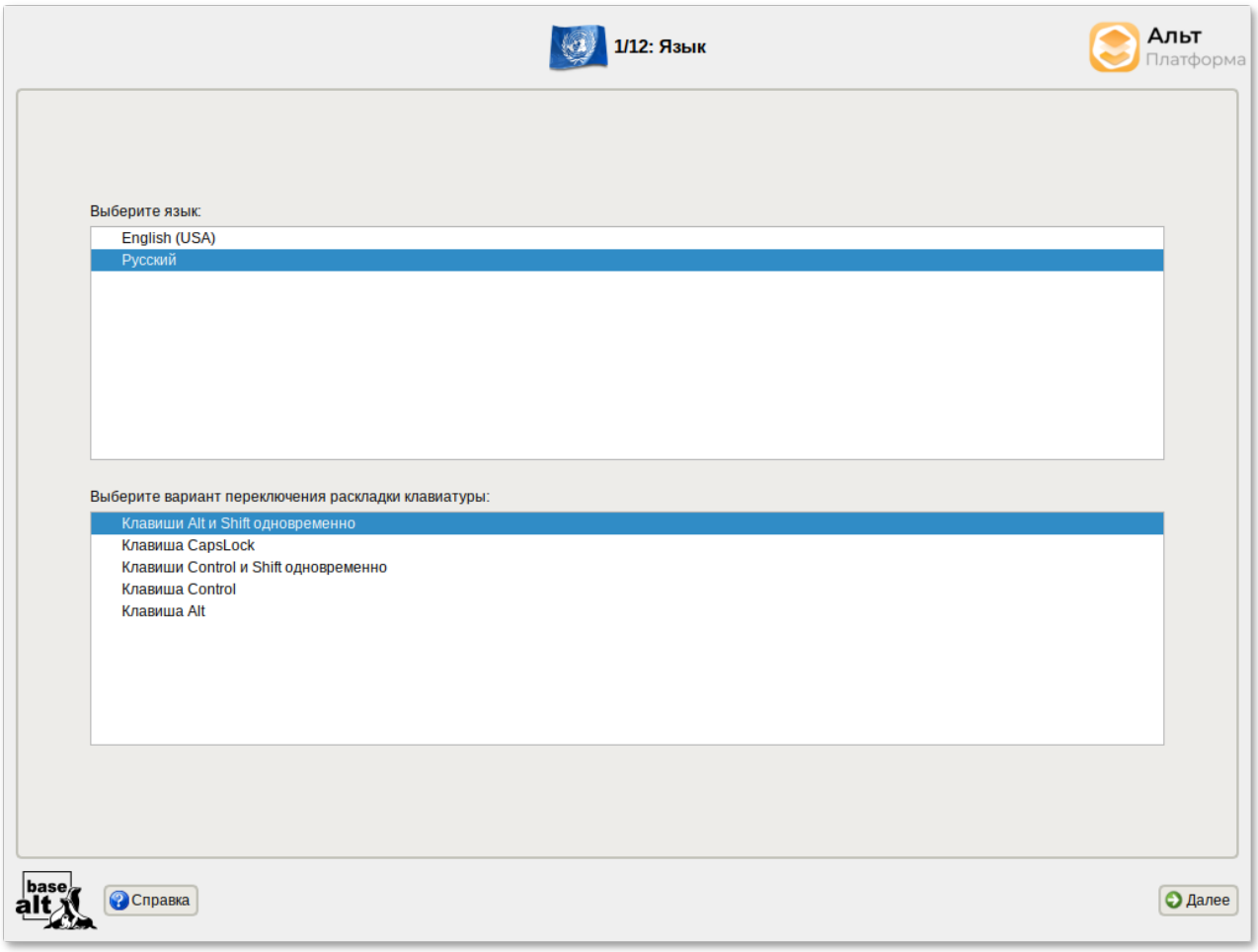
позволяет показать/скрыть панель со списком шагов установки:



Во время установки системы выполняются следующие шаги:

- Язык;
- Лицензионное соглашение;
- Дата и время;
- Подготовка диска;
- Установка системы;
- Сохранение настроек;
- Установка загрузчика;
- Настройка сети;
- Администратор системы;
- Системный пользователь;
- Установка пароля на шифрованные разделы;
- Завершение установки.

2.5. Язык



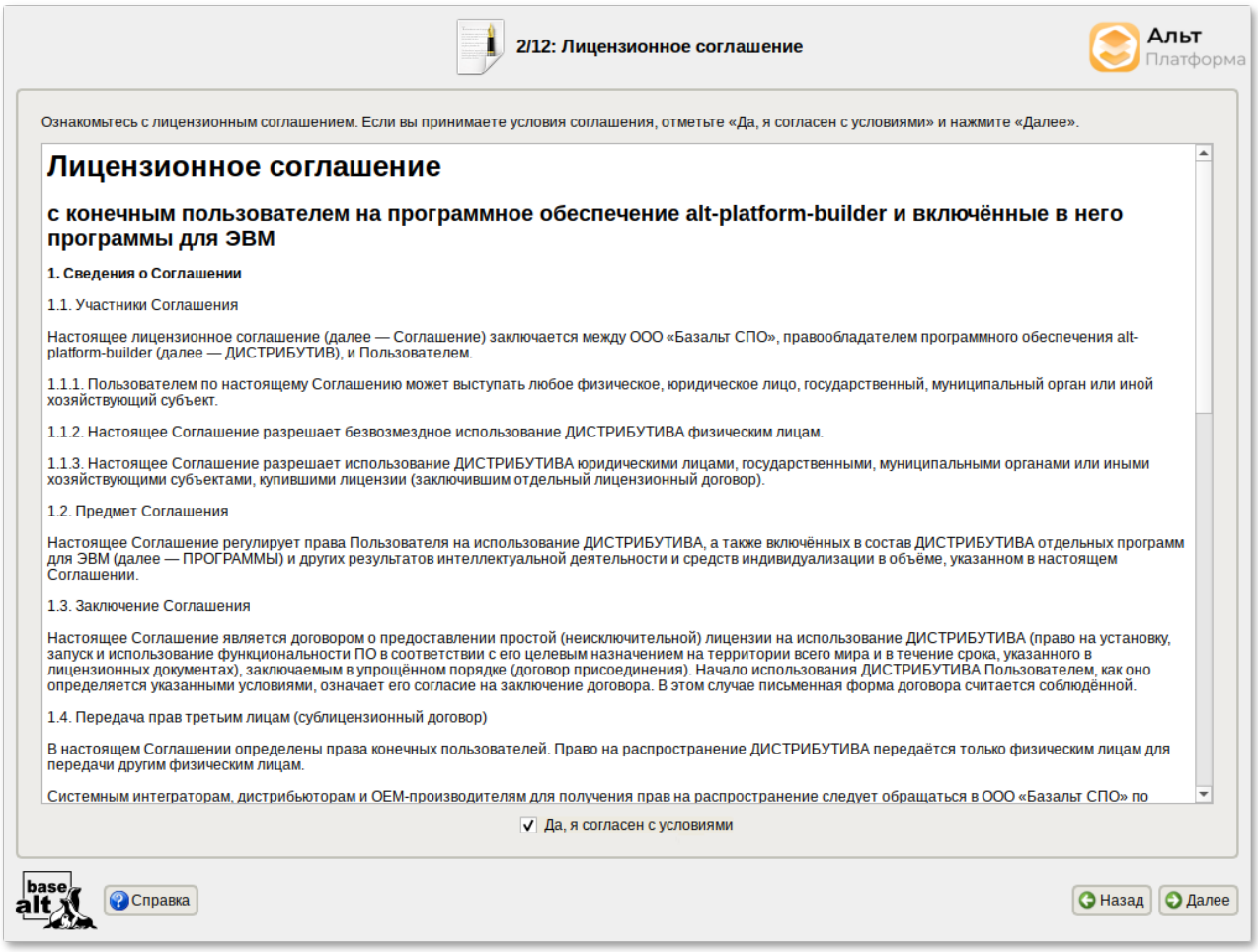
Установка ALT Platform Builder начинается с выбора основного языка — языка интерфейса программы установки и устанавливаемой системы. В списке, помимо доступных языков региона (выбранного на этапе начальной загрузки), указан и английский язык.

На этом же этапе выбирается вариант переключения раскладки клавиатуры. Раскладка клавиатуры — это привязка букв, цифр и специальных символов к клавишам на клавиатуре. Помимо ввода символов на основном языке, в любой системе Linux необходимо иметь возможность вводить латинские символы (имена команд, файлов и т.п.). Для этого обычно используется стандартная английская раскладка клавиатуры. Переключение между раскладками осуществляется при помощи специально зарезервированных для этого клавиш. Для русского языка доступны следующие варианты переключения раскладки:

- клавиши **`Alt`** и **`Shift`** одновременно;
- клавиша **`CapsLock`**;
- клавиши **`Control`** и **`Shift`** одновременно;
- клавиша **`Control`**;
- клавиша **`Alt`**.

Если выбранный основной язык имеет всего одну раскладку (например, при выборе английского языка в качестве основного), эта единственная раскладка будет принята автоматически.

2.6. Лицензионное соглашение



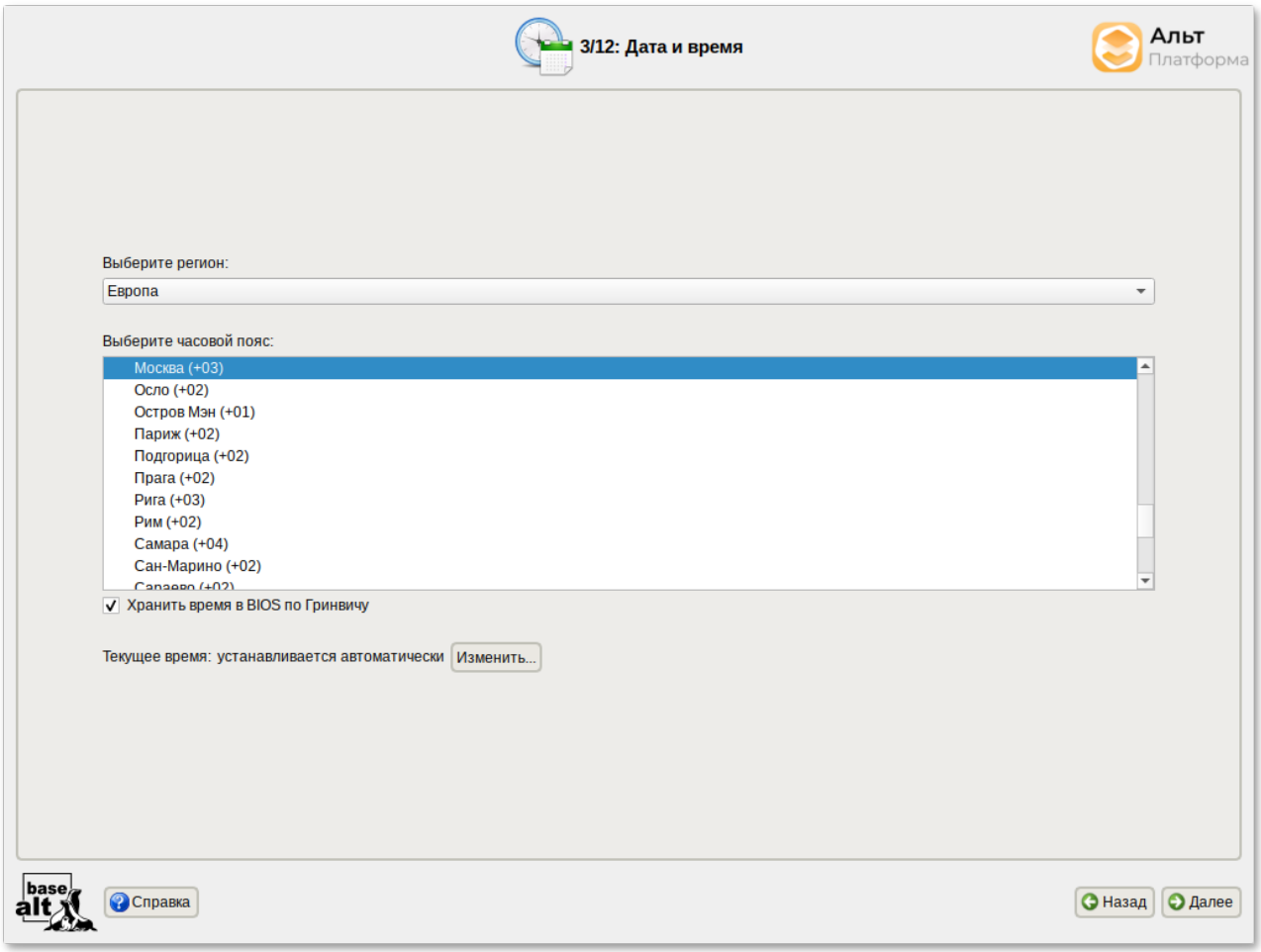
Перед продолжением установки следует внимательно прочитать условия лицензии. В лицензии говорится о ваших правах. В частности, за вами закрепляются права на:

- эксплуатацию программ на любом количестве компьютеров и в любых целях;
- распространение программ (сопровождая их копией авторского договора);
- получение исходных текстов программ.

Если вы приобрели дистрибутив, то данное лицензионное соглашение прилагается в печатном виде к вашей копии дистрибутива. Лицензия относится ко всему дистрибутиву Альт Платформа. Если вы согласны с условиями лицензии, отметьте пункт **`Да, я согласен с условиями`** и нажмите кнопку **Далее**.

2.7. Дата и время

На данном этапе выполняется выбор региона и города, по которым будет определен часовой пояс и установлены системные часы.



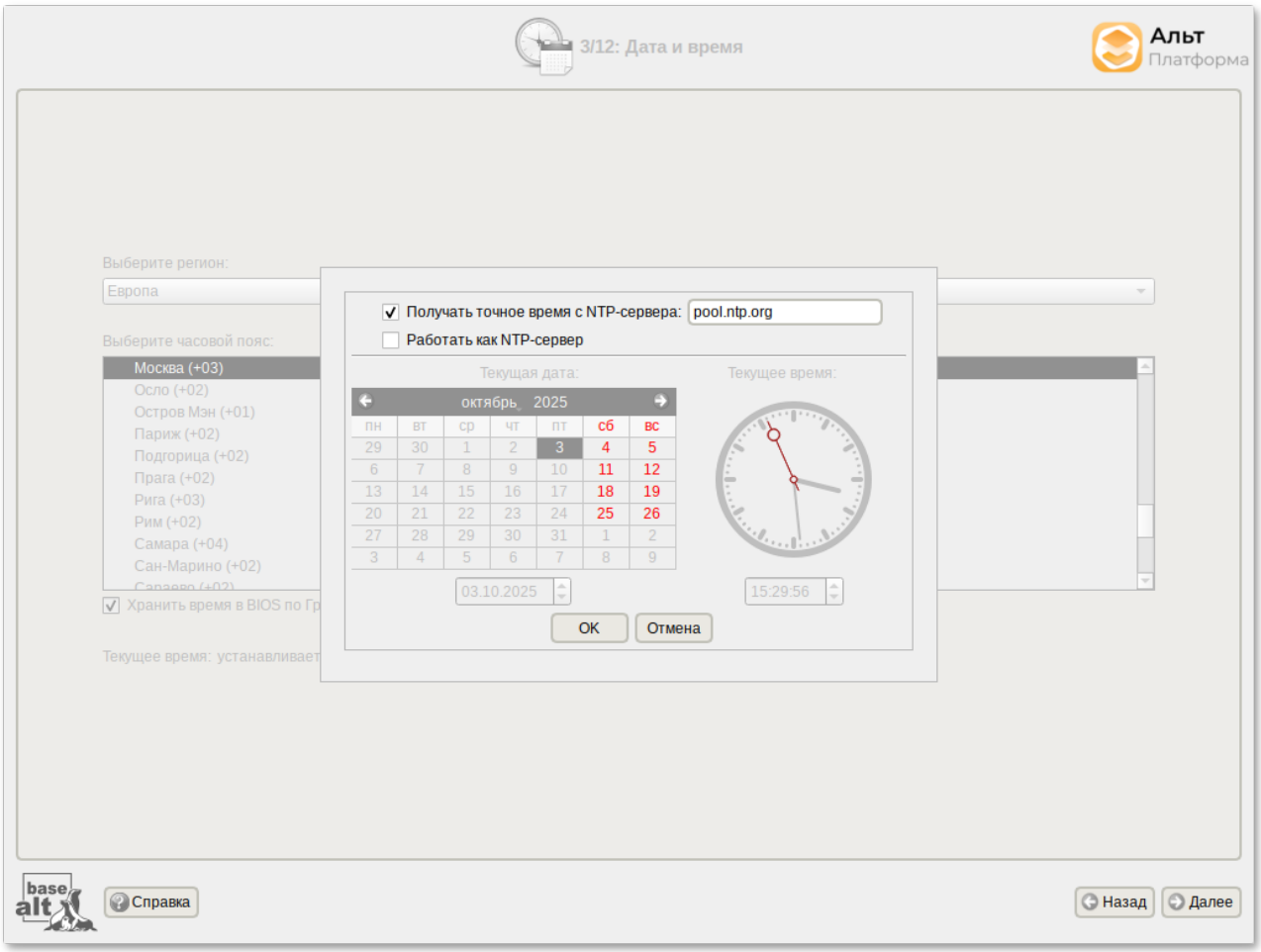
Для корректной установки даты и времени достаточно правильно указать часовой пояс и выставить желаемые значения для даты и времени.

Для указания часового пояса в соответствующих списках выберите регион, а затем город. Поиск по списку можно ускорить, набирая на клавиатуре первые буквы искомого слова.

Пункт **Хранить время в BIOS по Гринвичу** выставляет настройки даты и времени в соответствии с часовыми поясами, установленными по Гринвичу, и добавляет к местному времени часовую поправку для выбранного региона.

После выбора часового пояса будут предложены системные дата и время по умолчанию.

Для ручной установки текущих даты и времени нужно нажать кнопку **Изменить…**. Откроется окно ручной настройки системных параметров даты и времени.



Для сохранения настроек и продолжения установки системы в окне ручной установки даты и времени необходимо нажать кнопку **OK** и затем в окне **Дата и время** нажать кнопку **Далее**.

Примечание

В случае если ОС Альт Платформа устанавливается как вторая ОС, необходимо снять отметку с пункта **Хранить время в BIOS по Гринвичу**, иначе время в уже установленной ОС может отображаться некорректно.

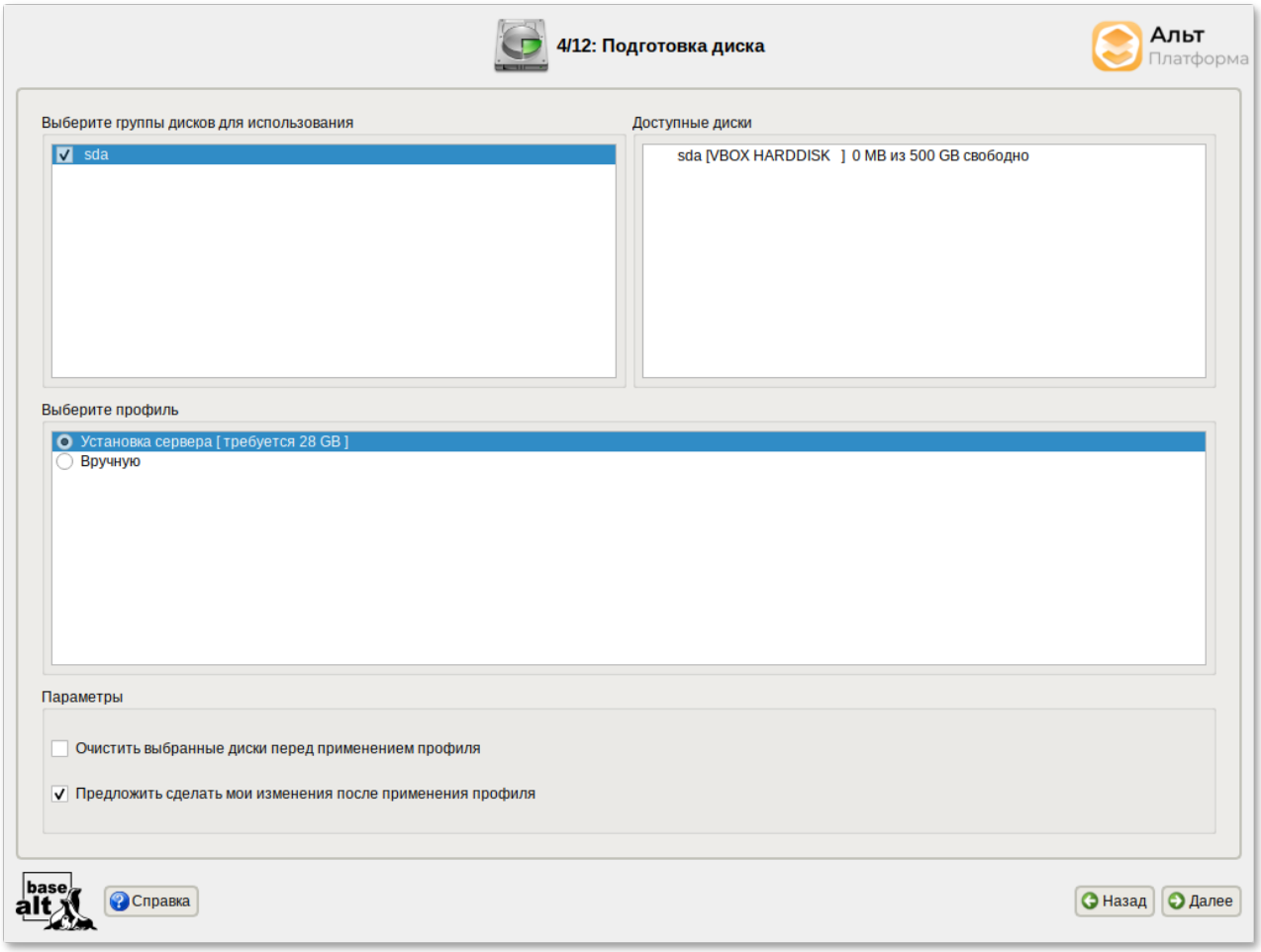
2.8. Подготовка диска

На этом этапе подготавливается площадка для установки ALT Platform Builder, в первую очередь — выделяется свободное место на диске.

Переход к этому шагу может занять некоторое время. Время ожидания зависит от производительности компьютера, объёма жёсткого диска, количества разделов на нём и других параметров.

2.8.1. Выбор профиля разбиения диска

После завершения первичной конфигурации загрузочного носителя откроется окно **Подготовка диска**. В списке разделов перечислены уже существующие на жёстких дисках разделы (в том числе здесь могут оказаться съёмные flash-диски, подключённые к компьютеру в момент установки).

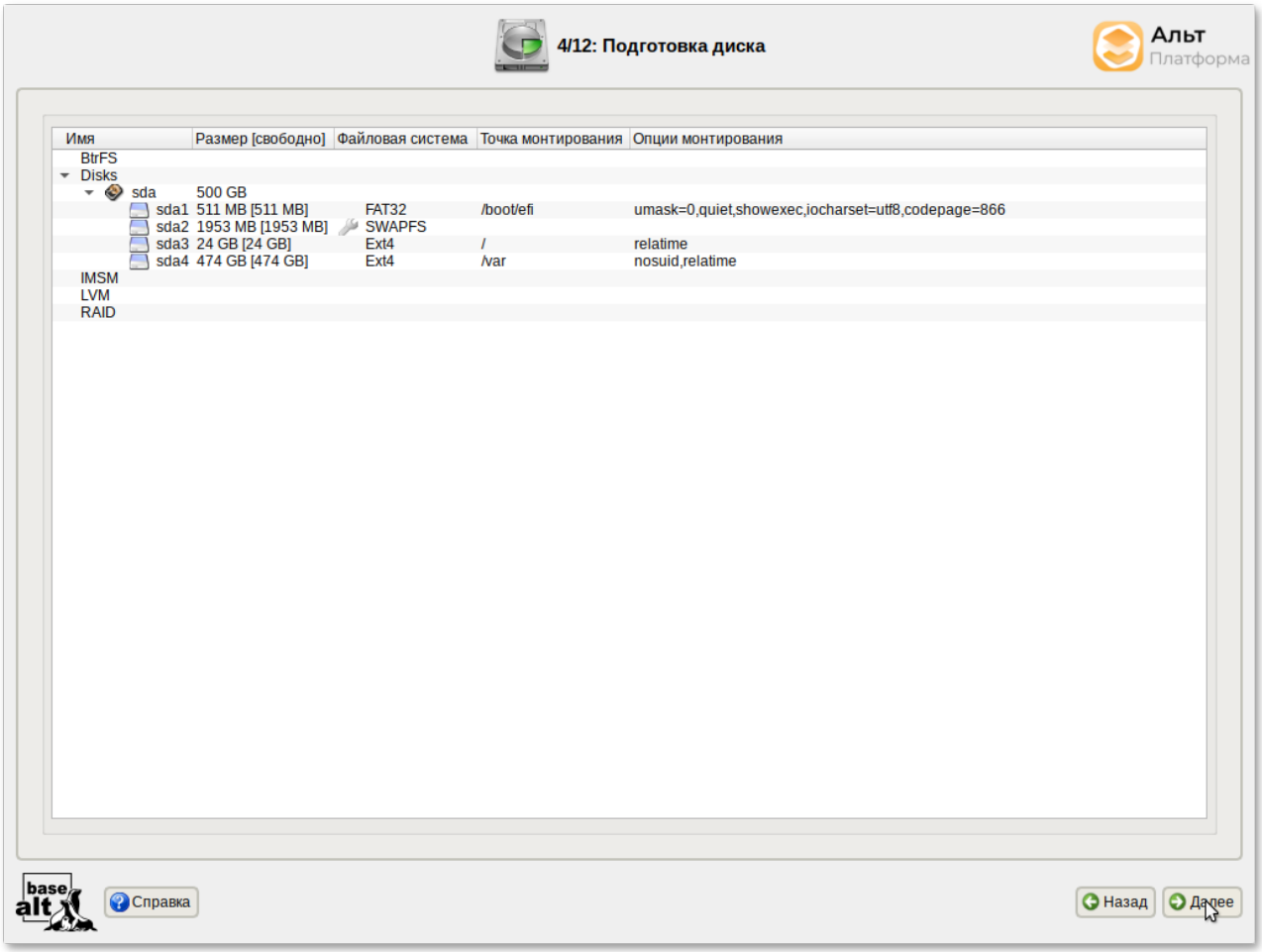


В списке **Выберите профиль** перечислены доступные профили разбиения диска. Профиль — это шаблон распределения места на диске для установки ОС. Можно выбрать один из профилей:

- **Установка сервера**;
- **Вручную**.

2.8.2. Автоматический профиль разбиения диска

Профиль **Установка сервера** предполагает автоматическое разбиение диска. При выборе этого профиля будут выделены отдельные разделы для подкачки, для EFI и для корневой файловой системы. Если размер диска больше 130 ГБ, будет также создан раздел **/var**.



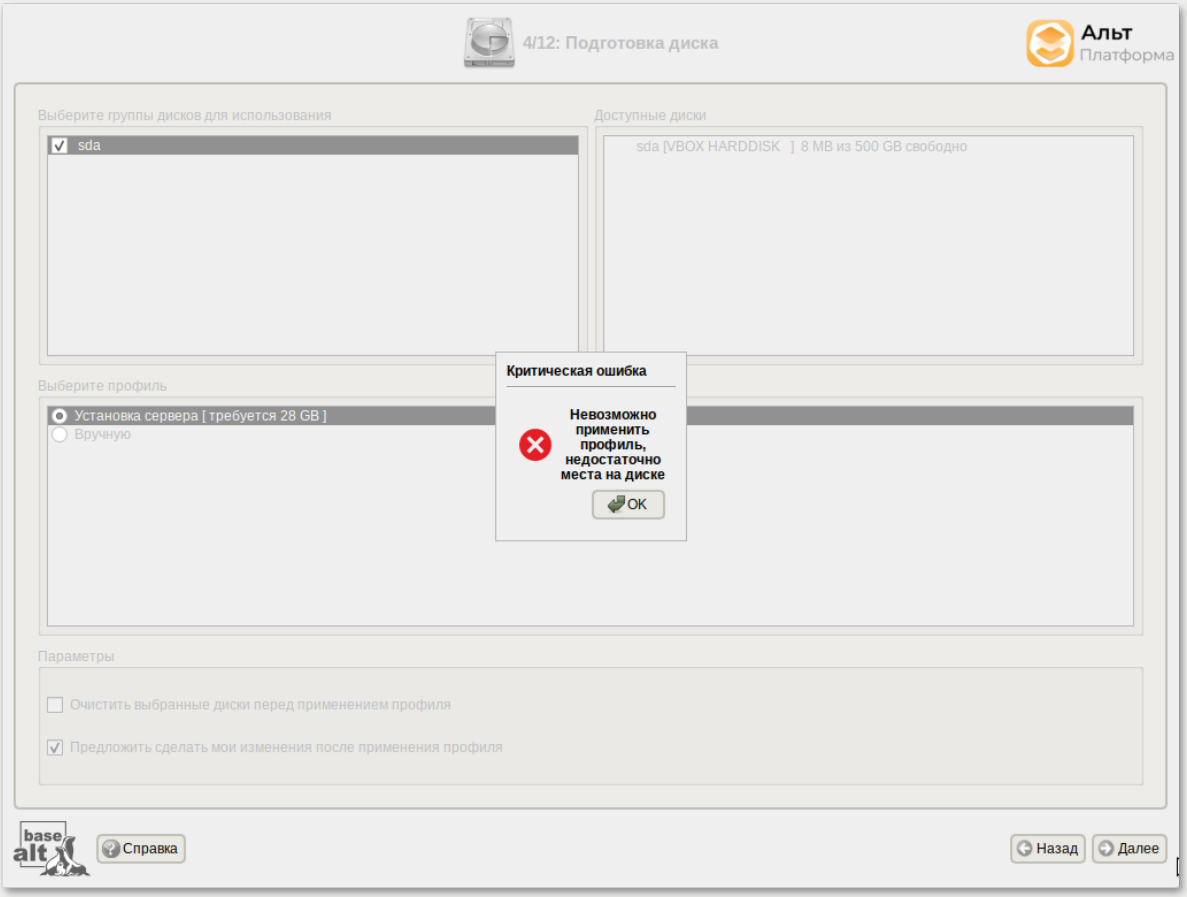
Если результат вас по каким-то причинам не устраивает, прямо сейчас можно его отредактировать.

От возможности редактировать результат разбиения можно отказаться, сняв выделение с пункта **Предложить сделать мои изменения после применения профиля**. В этом случае никакой информации о распределении дискового пространства на экране отображаться не будет. После осуществления физических изменений на жестком диске начнется установка базовой системы. Этот вариант подойдет для установки на чистый диск.

Рядом с названием профиля указан минимальный объём свободного места на диске, требуемый для установки в соответствии с данным профилем.

Примечание

Если при применении профиля автоматического разбиения диска доступного места на диске окажется недостаточно, то на монитор будет выведено сообщение об ошибке: **Невозможно применить профиль, недостаточно места на диске**.



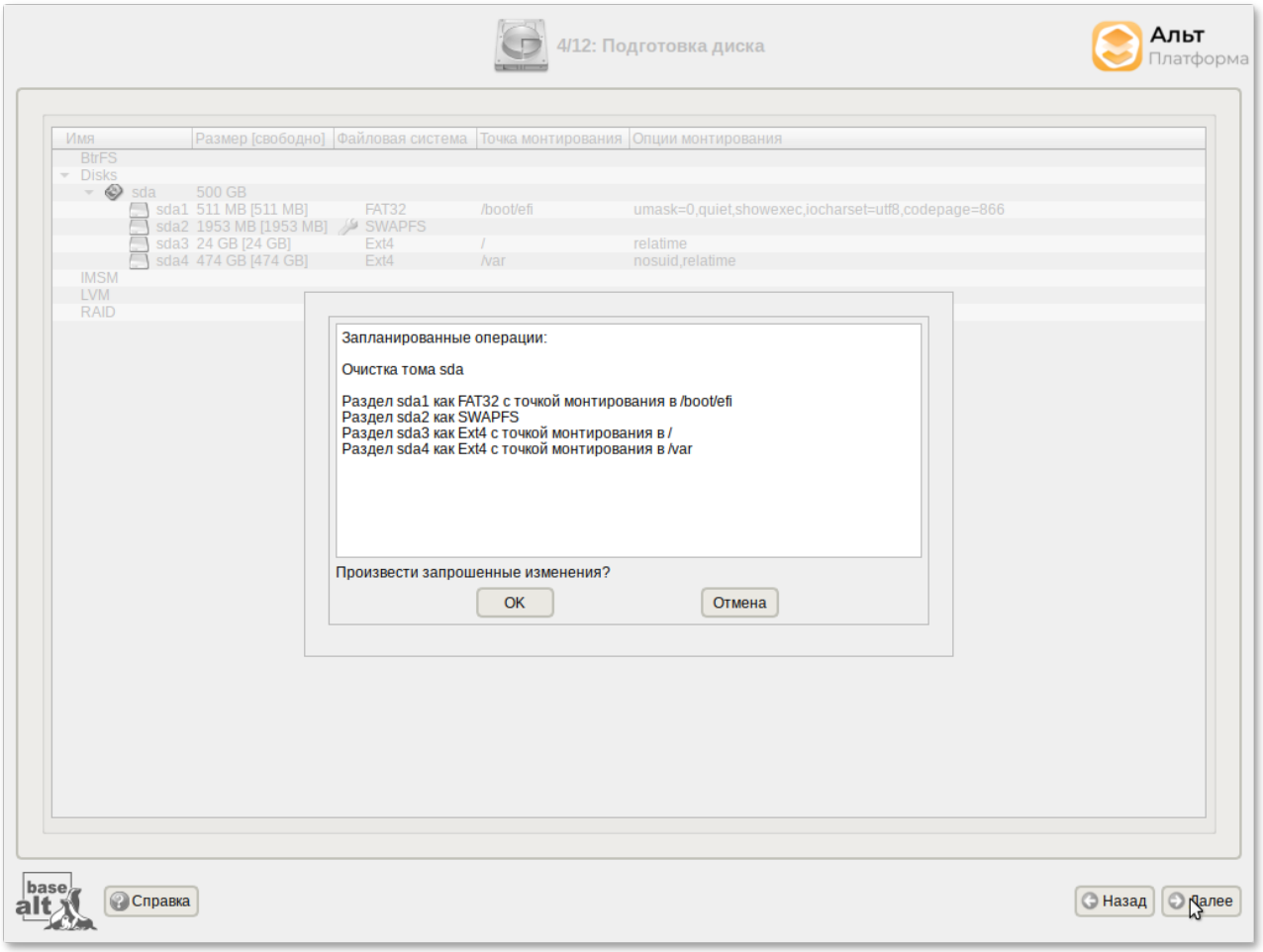
Для решения этой проблемы можно полностью очистить место на диске, отметив пункт **Очистить выбранные диски перед применением профиля** и применить профиль повторно.

Если сообщение о недостатке места на диске появляется и при отмеченном пункте **Очистить выбранные диски перед применением профиля**, то это связано с недостаточным для использования автоматических методов разметки объёмом выбранных дисков. В этом случае вы можете воспользоваться методом ручной разметки: профиль **Вручную**.

Предупреждение

При отмеченном пункте **Очистить выбранные диски перед применением профиля** будут удалены все данные с выбранных дисков (включая внешние USB-носители) без возможности восстановления. Рекомендуется использовать эту возможность при полной уверенности в том, что диски не содержат никаких ценных данных.

Для продолжения установки следует нажать кнопку **Далее**. Появится окно со списком настроенных разделов и их точек монтирования. Если вы уверены в том, что подготовка диска завершена, подтвердите переход к следующему шагу нажатием кнопки **OK**.



2.8.3. Ручной профиль разбиения диска

При необходимости освободить часть дискового пространства следует воспользоваться профилем разбиения **Вручную**. В этом случае можно удалить некоторые из существующих разделов или содержащиеся в них файловые системы. После этого можно создать необходимые разделы самостоятельно или вернуться к шагу выбора профиля и применить автоматический профиль. Выбор этой возможности требует знаний об устройстве диска и технологиях его разметки.

По нажатию кнопки **Далее** будет произведена запись новой таблицы разделов на диск и форматирование разделов. Только что созданные на диске программой установки разделы пока не содержат данных и поэтому форматируются без предупреждения. Уже существовавшие, но изменённые разделы, которые будут отформатированы, помечаются специальным значком в колонке **Файловая система** слева от названия. При уверенности в том, что подготовка диска завершена, подтвердите переход к следующему шагу нажатием кнопки **Далее**.

Не следует форматировать разделы с теми данными, которые вы хотите сохранить, например, со старыми пользовательскими данными (**/home**) или с другими операционными системами. С другой стороны, отформатировать можно любой раздел, который вы хотите «очистить» (удалить все данные).

Предупреждение

Не уменьшайте NTFS-раздел с установленной Microsoft Windows Vista/Windows 7 средствами программы установки. В противном случае вы не сможете загрузить Microsoft Windows Vista/Windows 7 после установки Альт Платформа. Для выделения места под установку Альт Платформа воспользуйтесь средствами, предоставляемыми самой Microsoft Windows Vista/Windows 7: **Управление дисками** → **Сжать**.

Для того чтобы система правильно работала (в частности могла загрузиться) с UEFI, при ручном разбиении диска надо обязательно сделать точку монтирования **/boot/efi**, в которую нужно смонтировать vfat раздел с загрузочными записями. Если такого раздела нет, то его надо создать вручную. При разбивке жёсткого диска в автоматическом режиме такой раздел создаёт сам установщик. Особенности разбиения диска в UEFI-режиме:

- требуется создать новый или подключить существующий FAT32-раздел с GPT-типом ESP (**efi system partition**) размером ~100—500 Мб (будет смонтирован в **/boot/efi**);
- может понадобиться раздел типа **bios boot partition** минимального размера, никуда не подключенный и предназначенный для встраивания grub2-efi;
- остальные разделы — и файловая система, и swap — имеют GPT-тип **basic data**; актуальный тип раздела задаётся отдельно.

Для сохранения всех внесенных настроек и продолжения установки в окне **Подготовка диска** нужно нажать кнопку **Далее**. Появится окно со списком настроенных разделов и их точек монтирования. Если вы уверены в том, что подготовка диска завершена, подтвердите переход к следующему шагу нажатием кнопки **OK**.

2.8.4. Дополнительные возможности разбиения диска

Ручной профиль разбиения диска позволяет установить ОС на программный RAID-массив, разместить разделы в томах LVM и использовать шифрование на разделах. Данные возможности требуют от пользователя понимания принципов функционирования указанных технологий.

2.8.4.1. Создание программного RAID-массива

Избыточный массив независимых дисков RAID (redundant array of independent disks) — технология виртуализации данных, которая объединяет несколько жёстких дисков в логический элемент для избыточности и повышения производительности.

Примечание

Для создания программного RAID-массива потребуется минимум два жёстких диска.

Программа установки поддерживает создание программных RAID-массивов следующих типов:

- RAID 1;
- RAID 0;
- RAID 4/5/6;

- RAID 10.

Процесс подготовки к установке на RAID условно можно разбить на следующие шаги:

- создание разделов на жёстких дисках;
- создание RAID-массивов на разделах жёсткого диска;
- создание файловых систем на RAID-массиве.

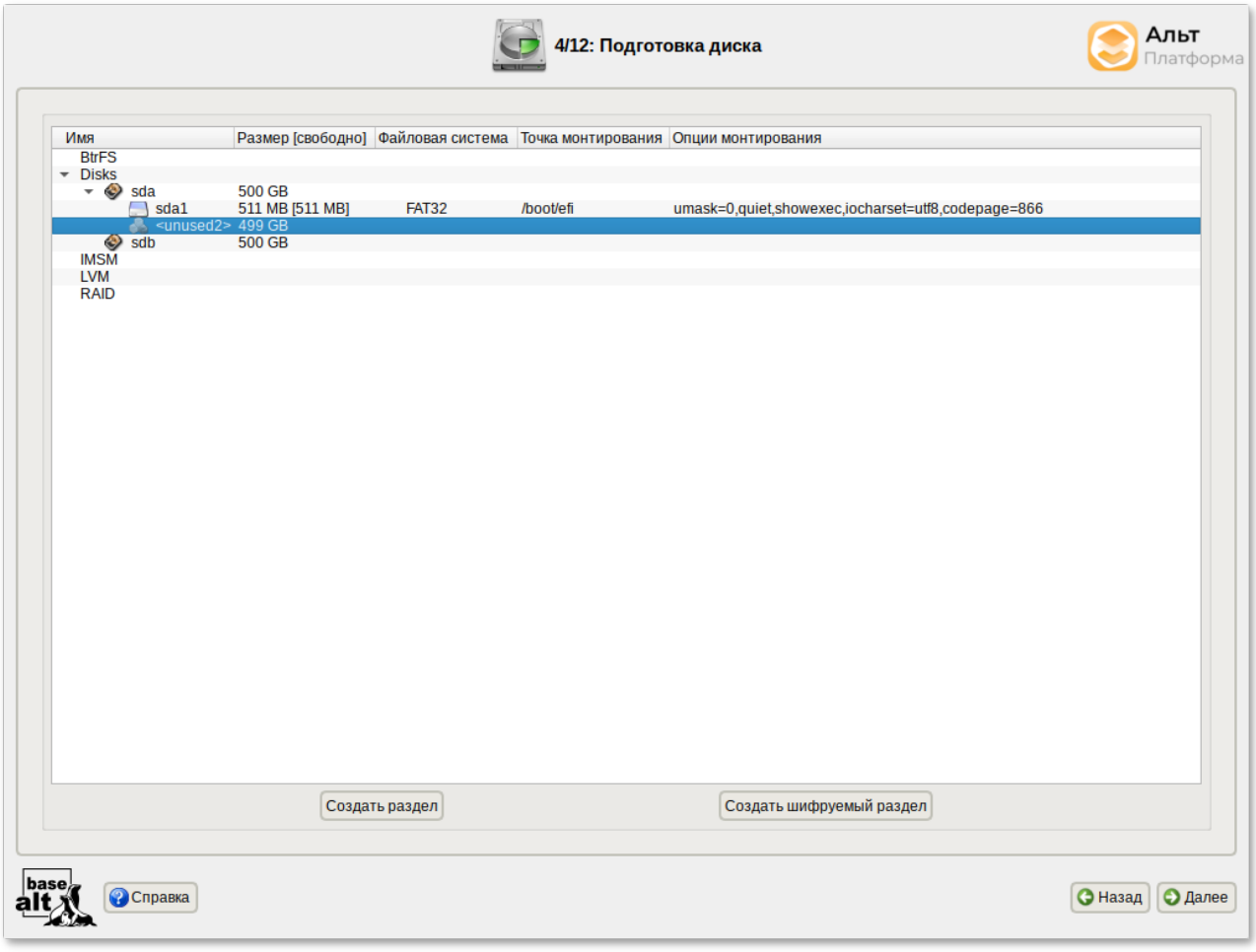
Важно

Для создания программного RAID-массива может потребоваться предварительно удалить существующую таблицу разделов с жёсткого диска.

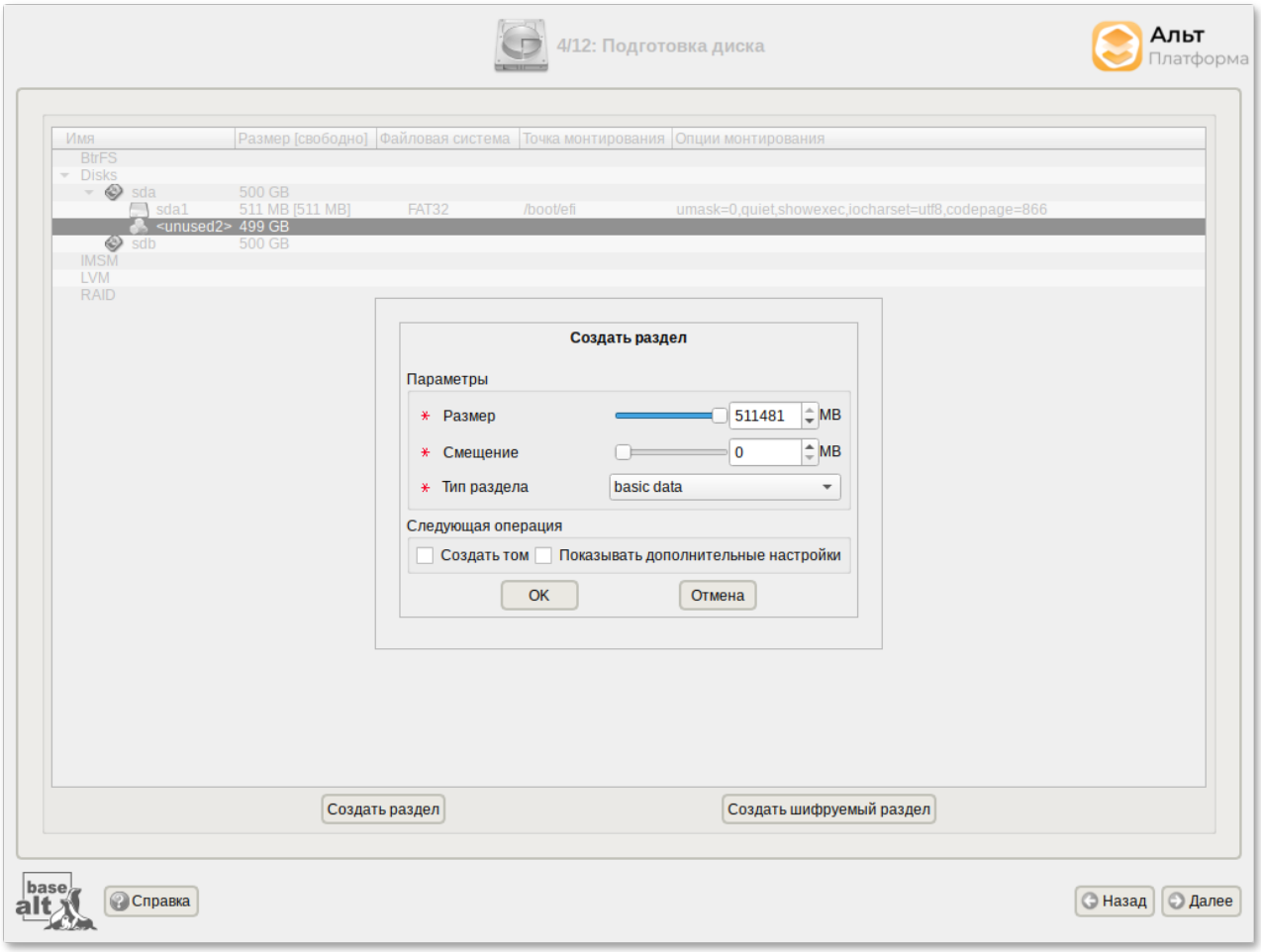
Важно

Системный раздел EFI должен быть физическим разделом в основной таблице разделов диска.

Для настройки параметров нового раздела из состава RAID-массива необходимо выбрать неразмеченный диск в окне профиля разбивки пространства **Вручную** и нажать кнопку **Создать раздел**.



Для создания программного массива на GPT-разделах следует сначала создать разделы типа **`basic data`** и не создавать на них том (снять отметку с пункта **`Создать том`**):

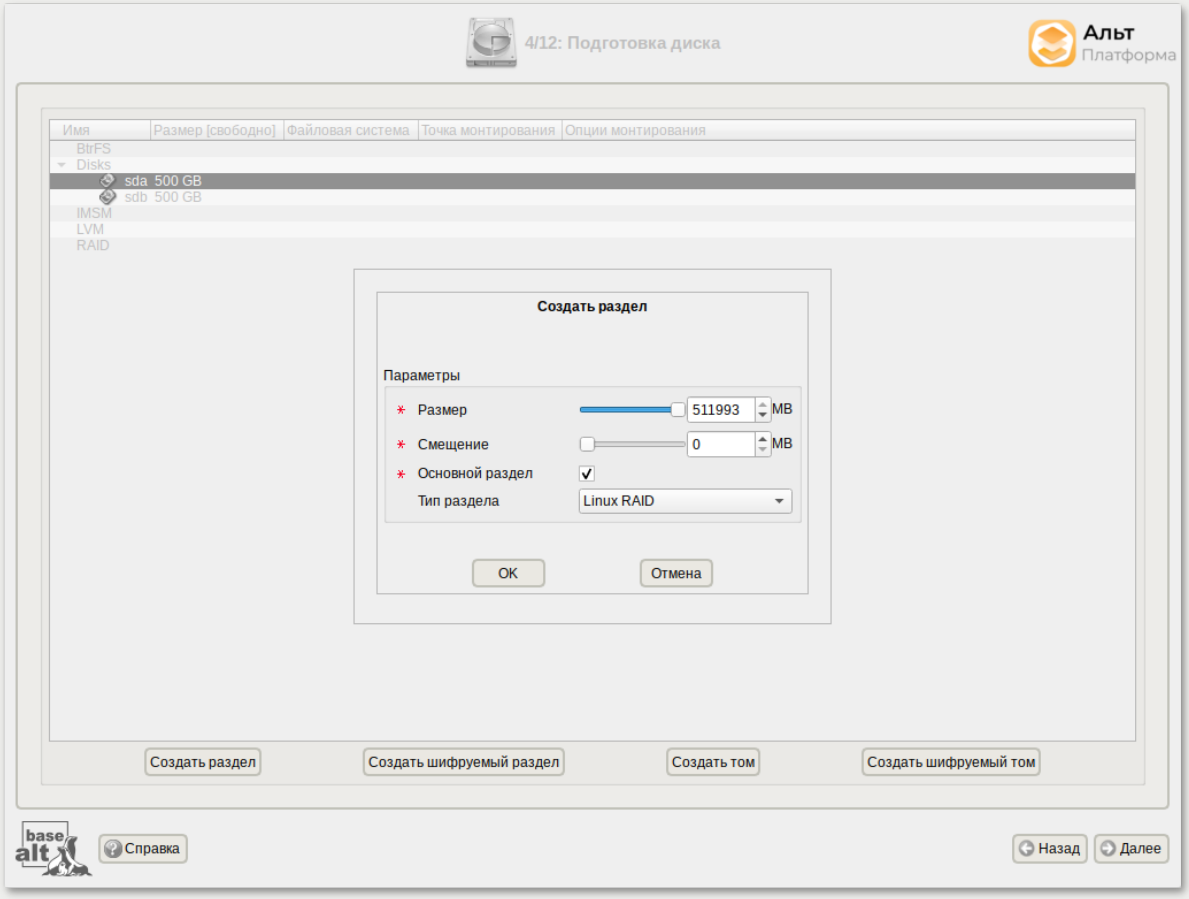


В этом окне необходимо настроить следующие параметры:

- **`Размер`** — в поле необходимо указать размер будущего раздела в Мбайт;
- **`Смещение`** — в поле необходимо указать смещение начала данных на диске в Мбайт;
- **`Тип раздела`** — в выпадающем поле нужно выбрать значение **basic data** для последующего включения раздела в RAID-массив.

Примечание

В режиме Legacy при создании разделов на жёстких дисках для последующего включения их в RAID-массивы следует указать **Тип раздела** для них равным **Linux RAID**:

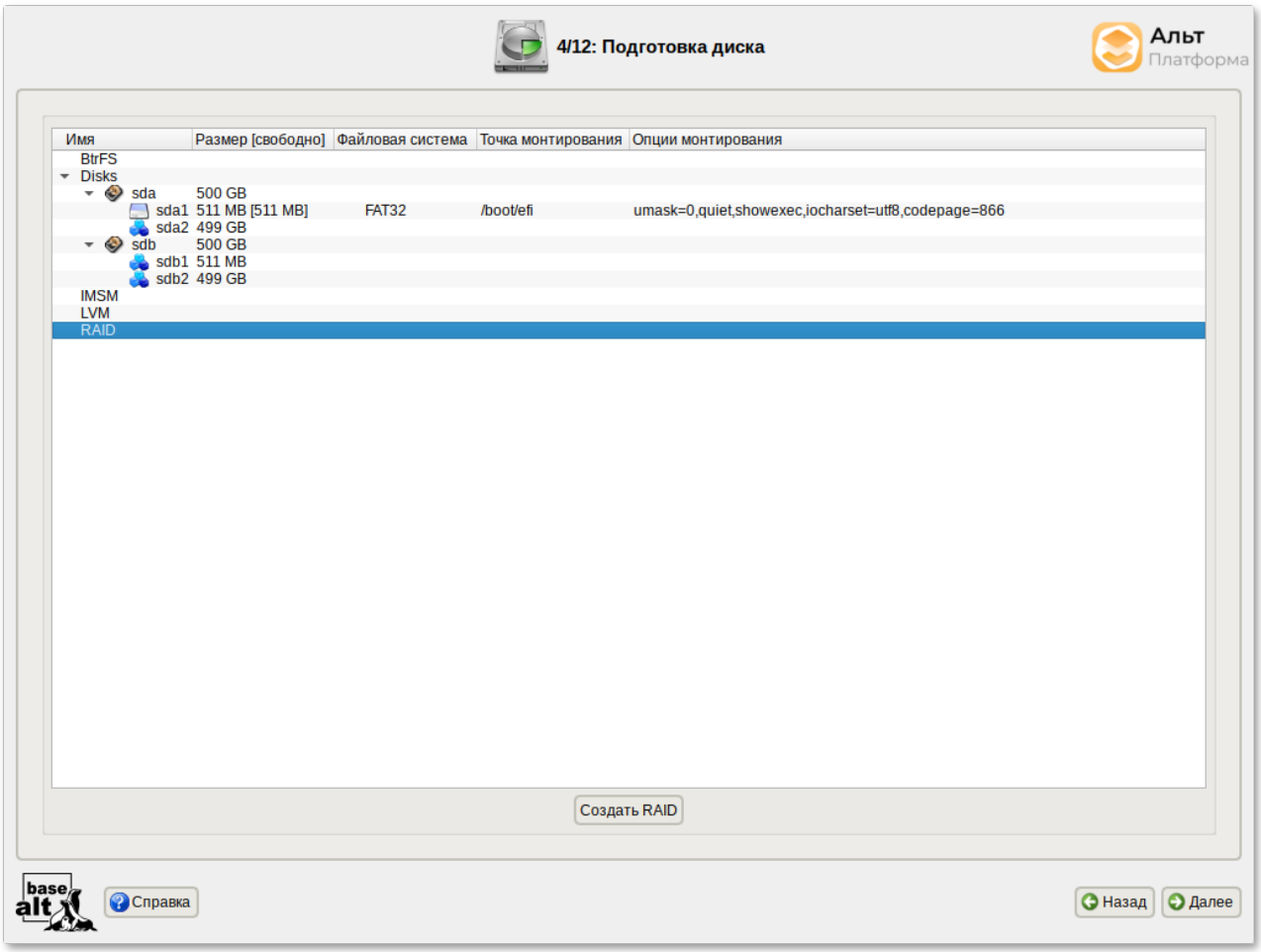


На втором диске создать два раздела с типом **basic data** без создания на них томов. При этом разделы на разных дисках должны совпадать по размеру.

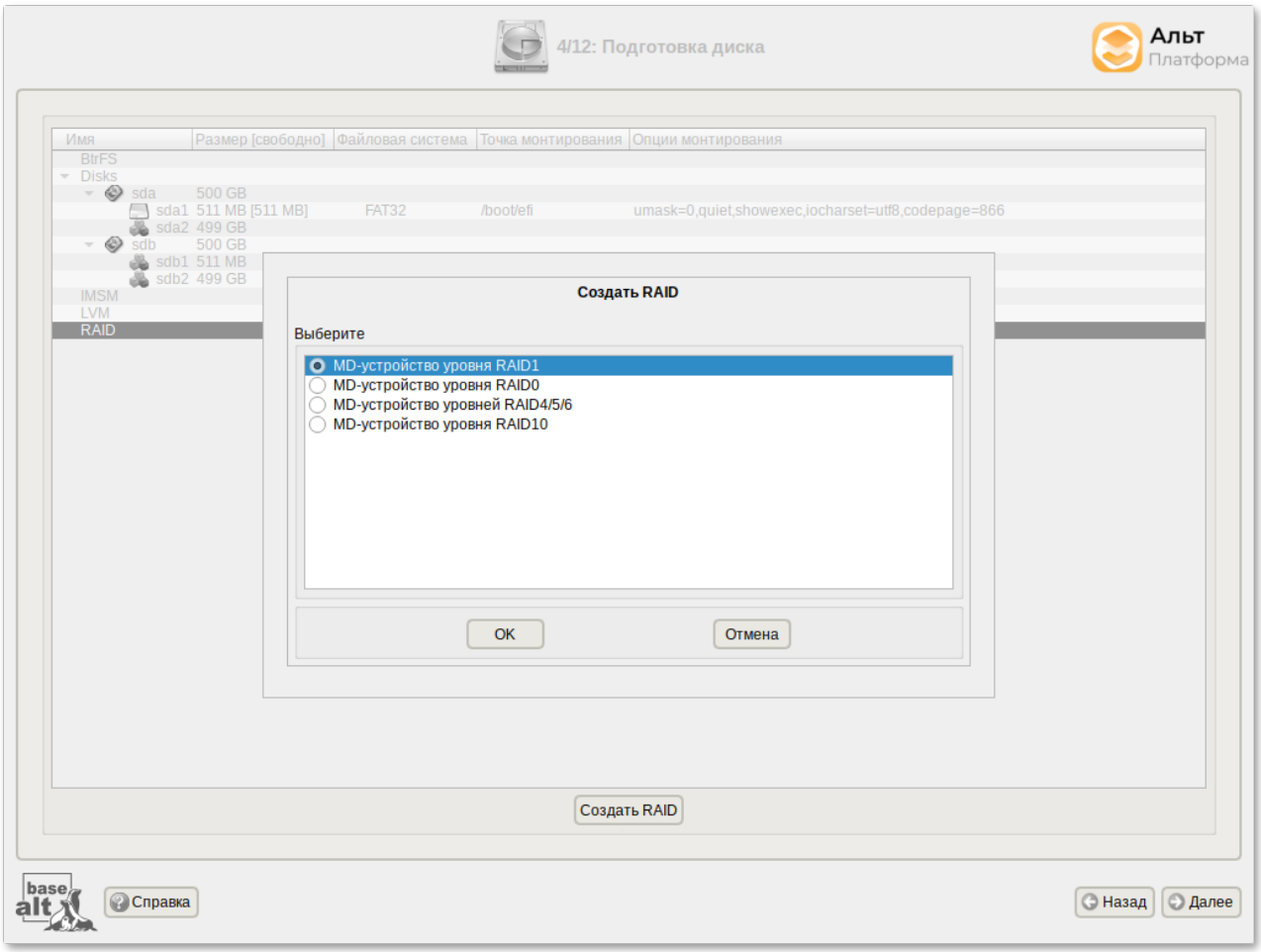
Примечание

При создании разделов следует учесть, что объём результирующего массива может зависеть от размера, включённых в него разделов жёсткого диска. Например, при создании RAID 1 результирующий размер массива будет равен размеру минимального участника.

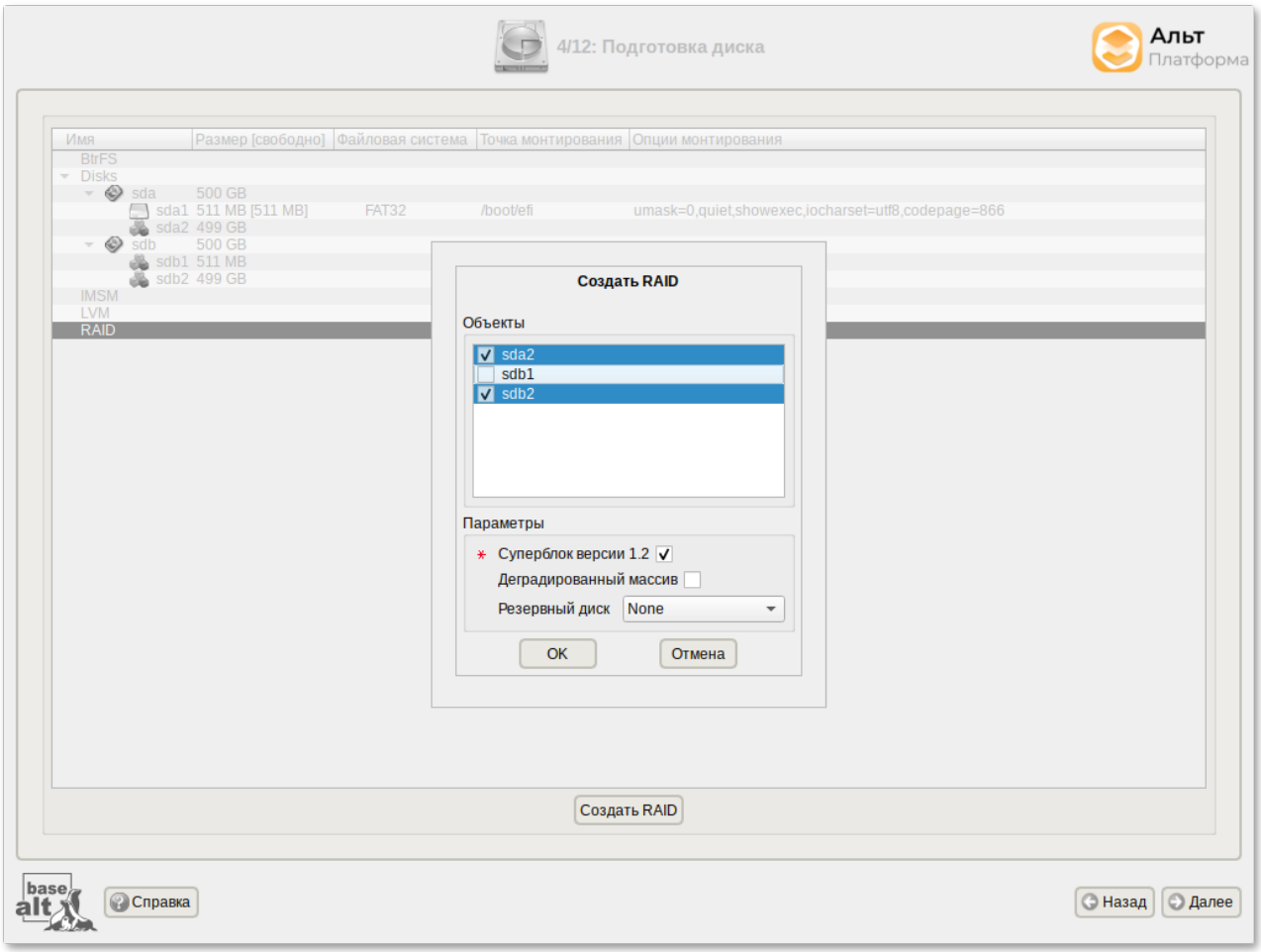
После создания разделов на дисках можно переходить к организации самих RAID-массивов. Для этого в списке следует выбрать пункт **RAID**, после чего нажать кнопку **Создать RAID**:



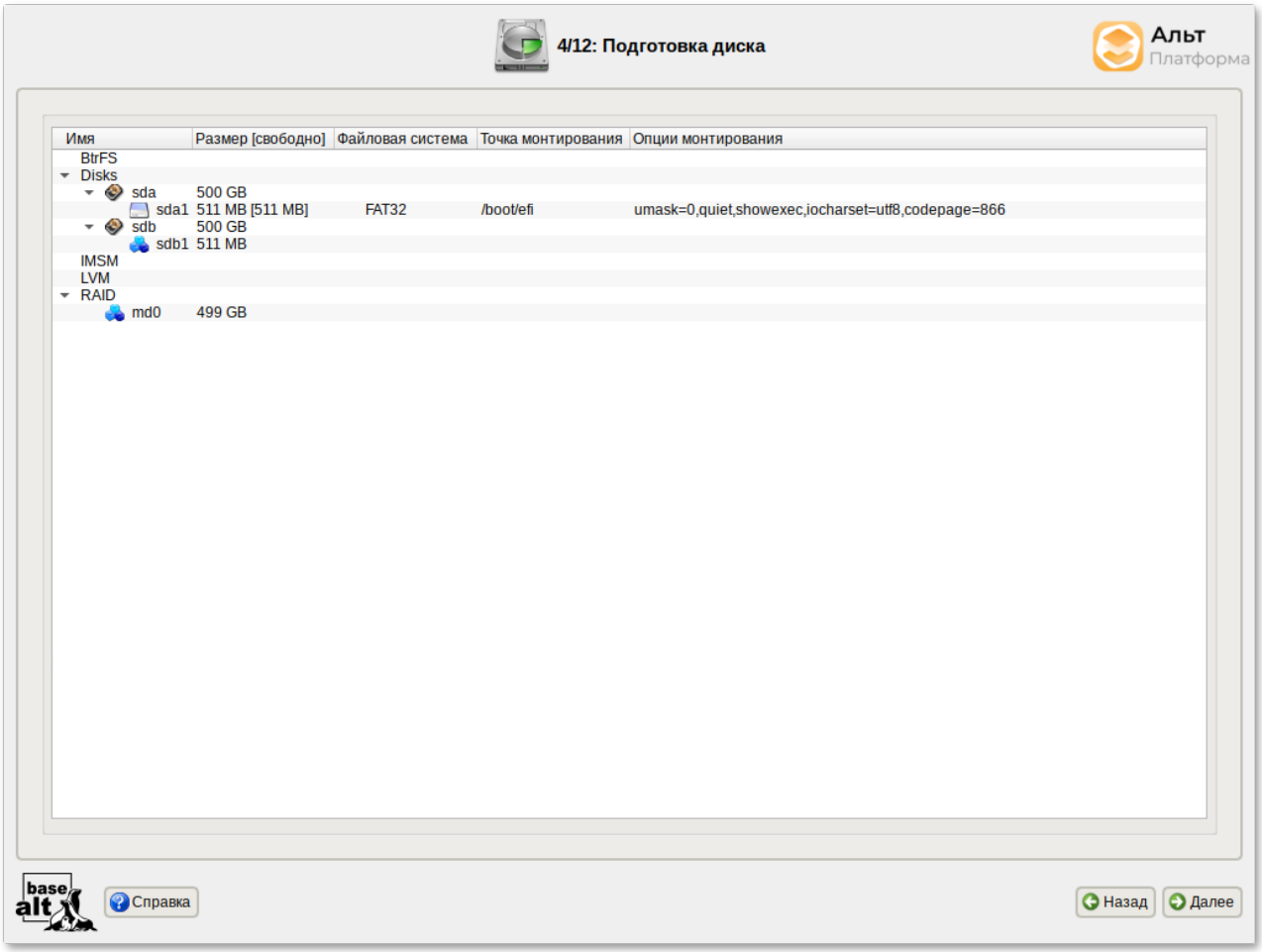
Далее мастер предложит выбрать тип массива:



И указать участников RAID-массива (по умолчанию выбираются все разделы, поэтому необходимо снять отметку с раздела **sdb1**):



Результат создания RAID-массива:



После того, как RAID-массив создан, его можно использовать как обычный раздел на жёстких дисках, то есть на нём можно создавать файловые системы или же, например, включать в LVM-тома.

Примечание

После установки системы можно будет создать ещё один RAID-массив и добавить в него загрузочный раздел (**`/boot/efi`**).

2.8.4.2. Создание LVM-томов

Менеджер логических дисков LVM (Logical Volume Manager) — средство гибкого управления дисковым пространством, позволяющее создавать поверх физических разделов (либо неразбитых дисков) логические тома, которые в самой системе будут видны как обычные блочные устройства с данными (обычные разделы).

Процесс подготовки к установке на LVM условно можно разбить на следующие шаги:

- создание разделов на жёстких дисках;
- создание группы томов LVM;
- создание томов LVM;
- создание файловых систем на томах LVM.

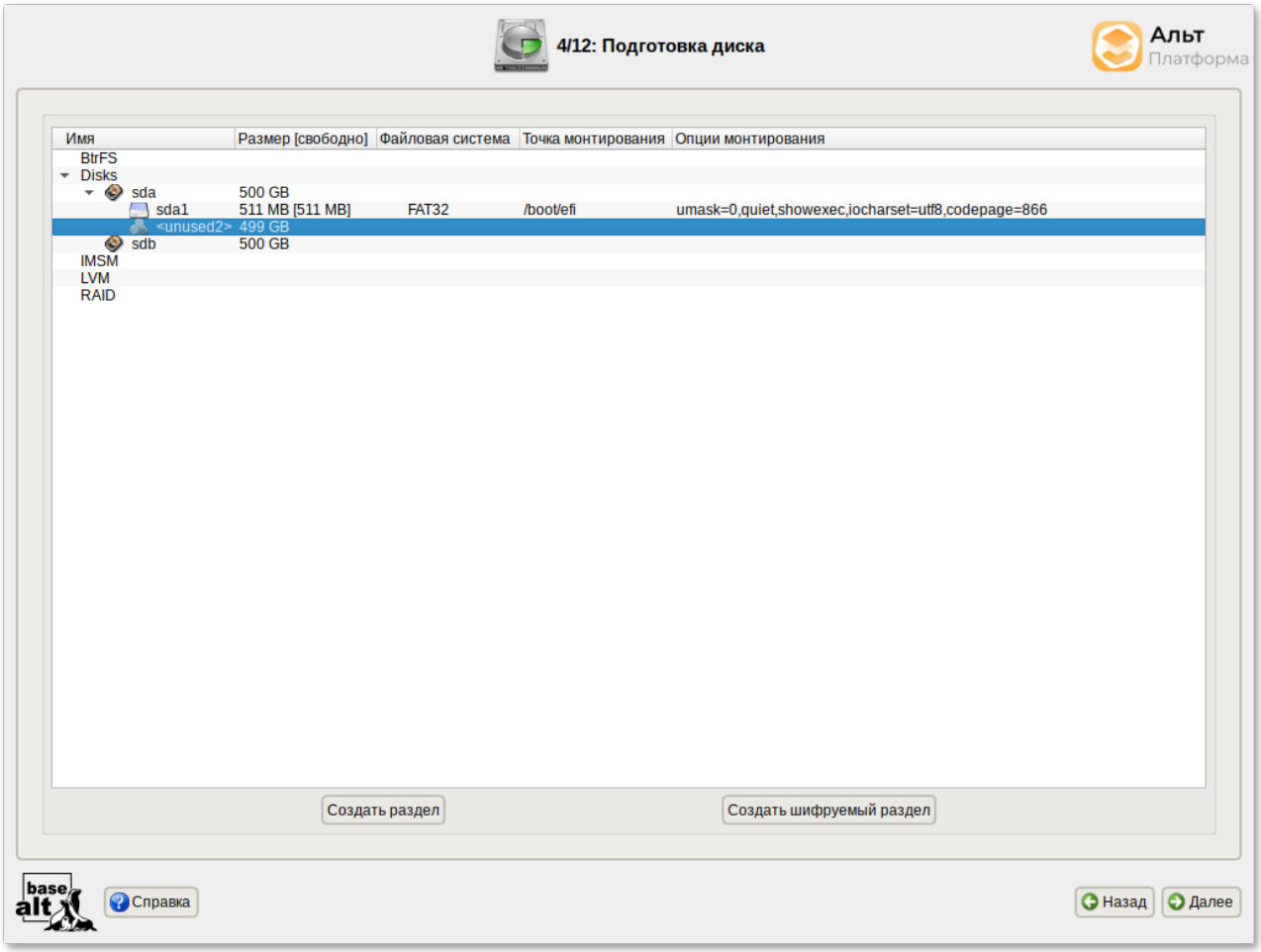
Важно

Для создания группы томов LVM может потребоваться предварительно удалить существующую таблицу разделов с жёсткого диска.

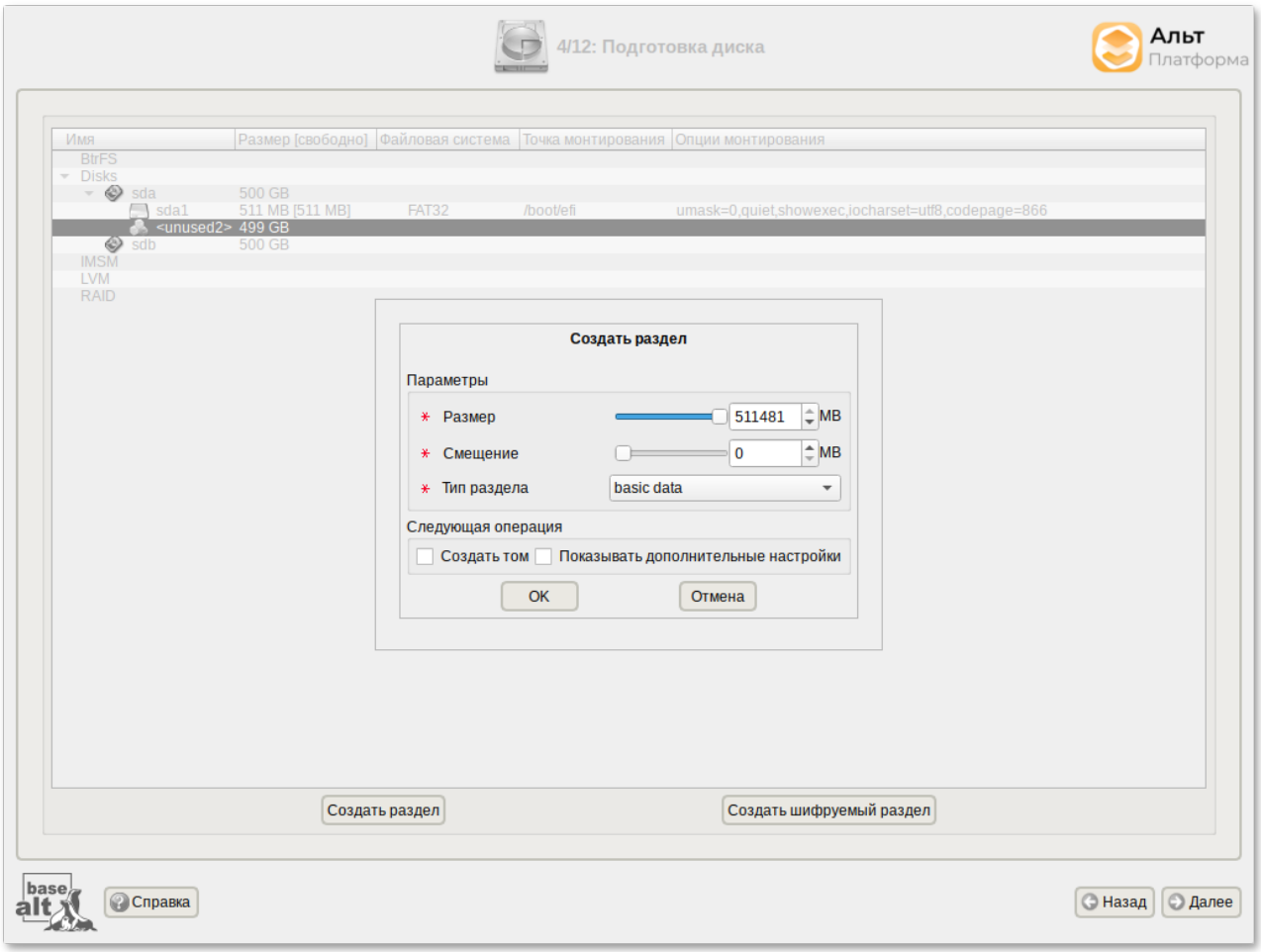
Важно

Системный раздел EFI должен быть физическим разделом в основной таблице разделов диска, не под LVM.

Для настройки параметров нового раздела необходимо выбрать неразмеченный диск в окне профиля разбивки пространства **Вручную** и нажать кнопку **Создать раздел**:

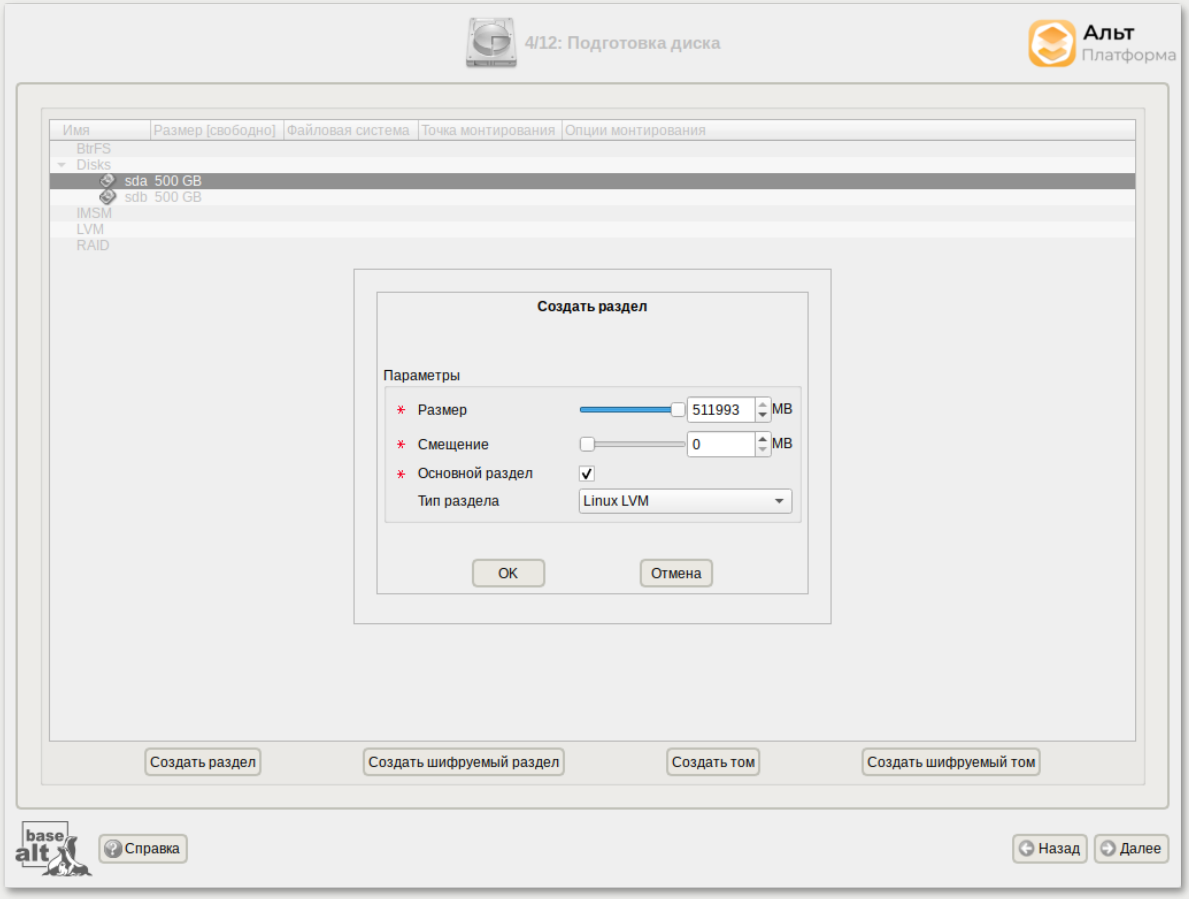


При создании разделов на жёстких дисках для последующего включения их в LVM-тома следует указать **Тип раздела** для них равным **basic data** и не создавать на них том (снять отметку с пункта **Создать том**):

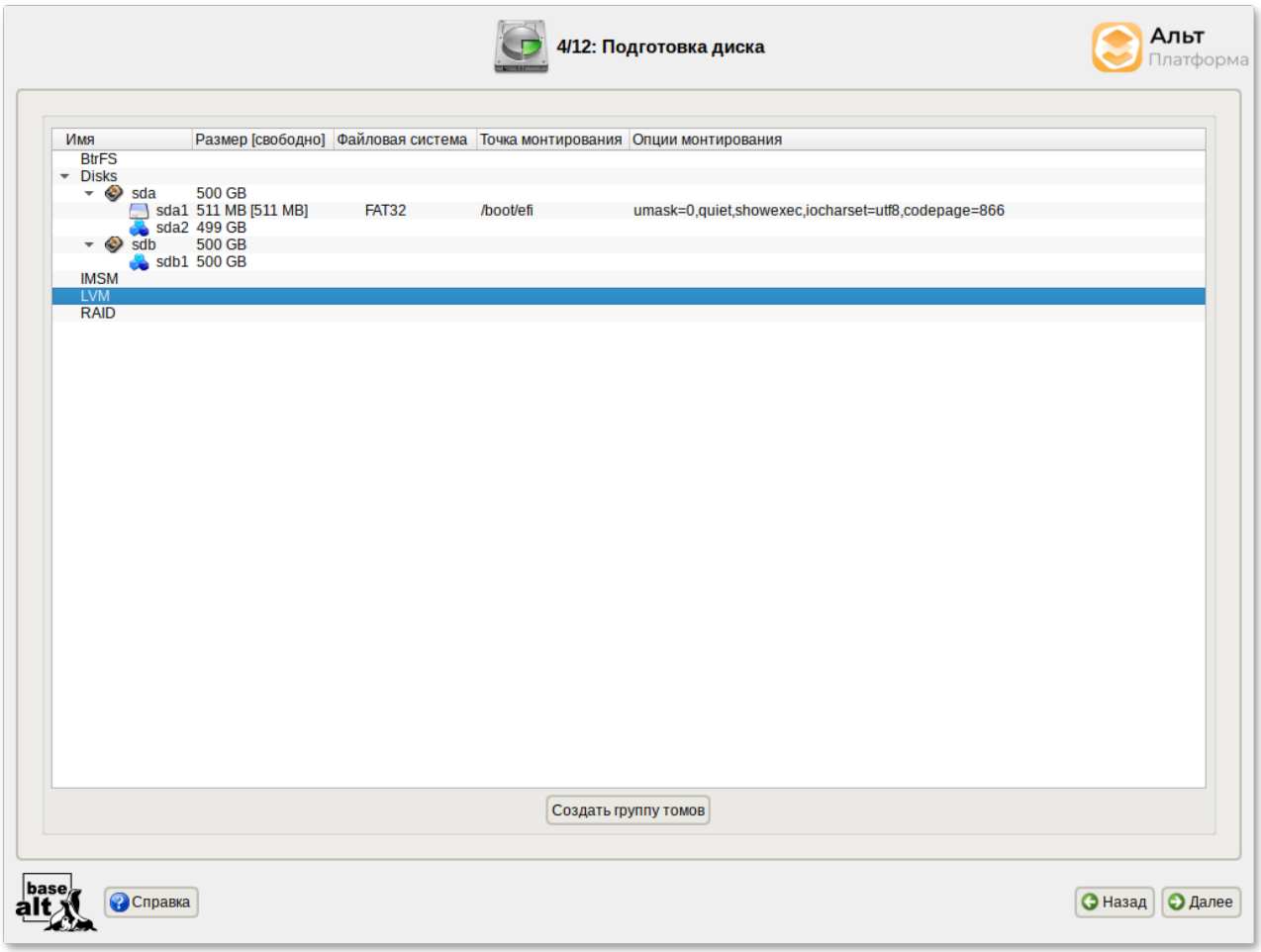

4/12: Подготовка диска
Альт
Платформа
Имя
Размер [свободно]
Файловая система
Точка монтирования
Опции монтирования
BtrFS
Disks
sda 500 GB
sda1 511 MB [511 MB] FAT32 /boot/efi umask=0,quiet,showexec,iocharset=utf8,codepage=866
<unused2> 499 GB
sdb 500 GB
IMSM
LVM
RAID
Создать раздел
Параметры
Размер 511481 MB
Смещение 0 MB
Тип раздела basic data
Следующая операция
Создать том
Показывать дополнительные настройки
OK
Отмена
Создать раздел
Создать шифруемый раздел
base alt
Справка
Назад
Далее

Примечание

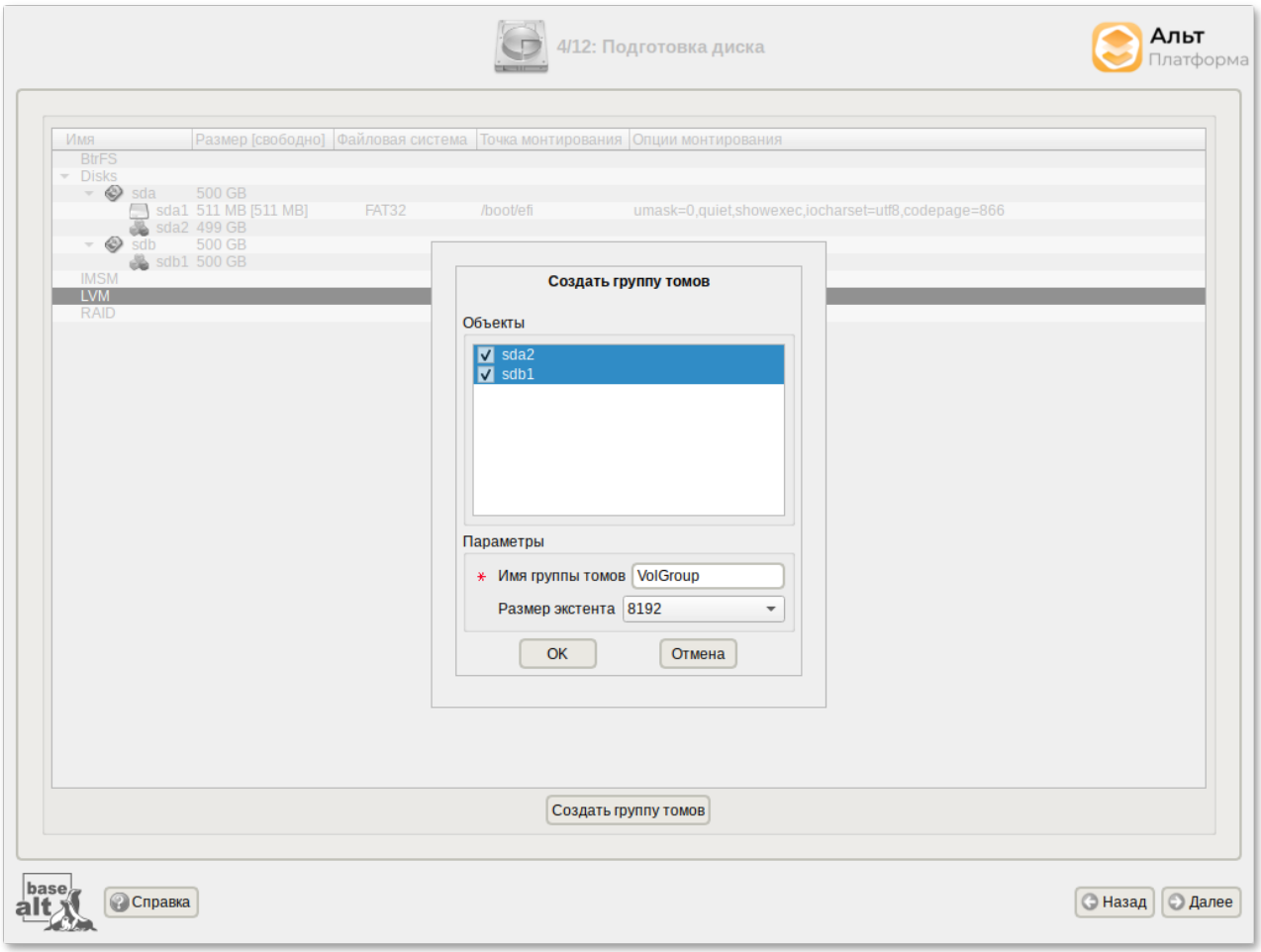
В режиме Legacy при создании разделов на жёстких дисках для последующего включения их в LVM-тома следует указать **Тип раздела** для них равным **Linux LVM**:



После создания разделов на дисках можно переходить к созданию группы томов LVM. Для этого в списке следует выбрать пункт **LVM**, после чего нажать кнопку **Создать группу томов**:



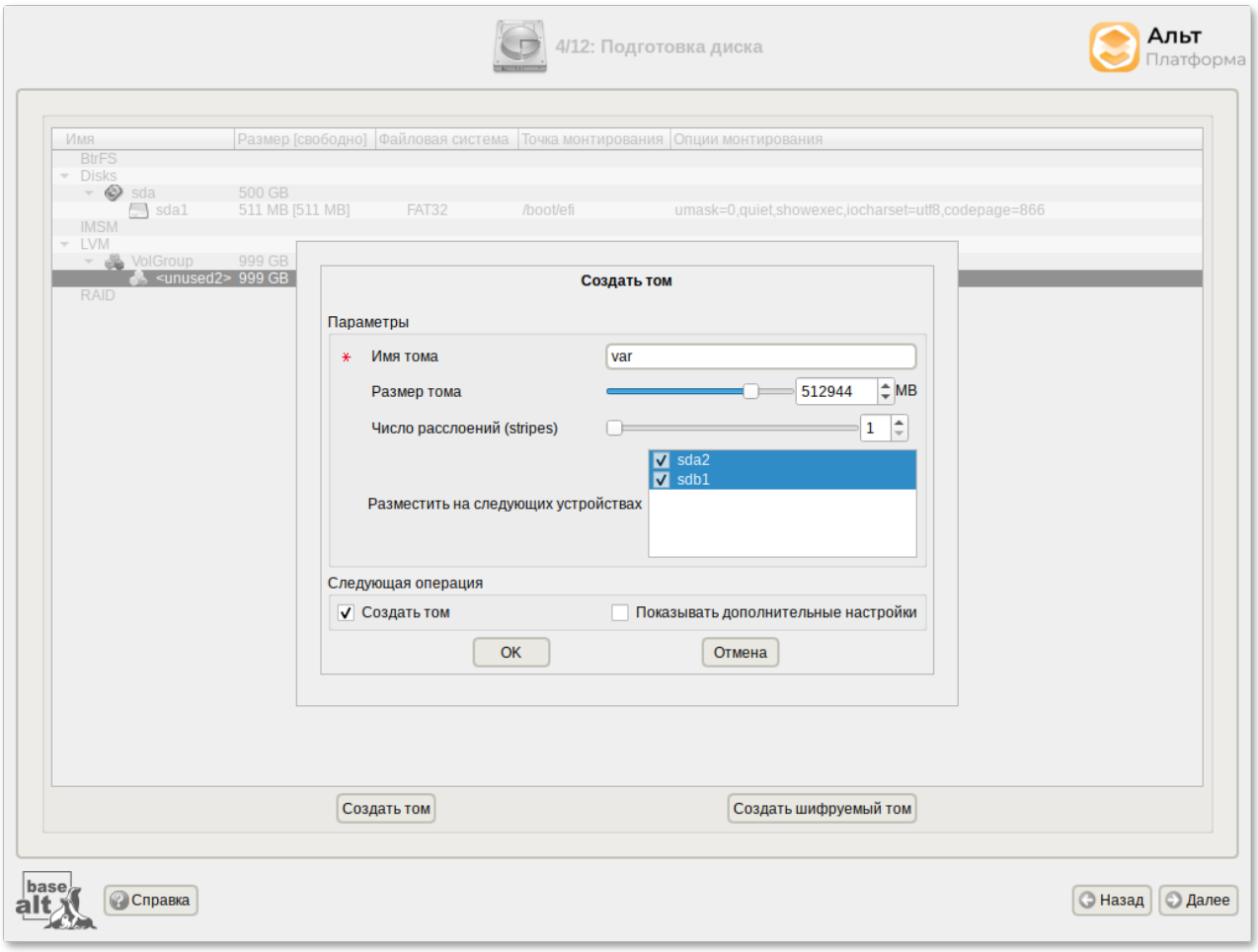
В открывшемся окне необходимо выбрать разделы, которые будут входить в группу томов, указать название группы томов и выбрать размер экстента:



Примечание

Размер экстента представляет собой наименьший объем пространства, который может быть выделен тому. Размер экстента по умолчанию 65536 (65536*512 байт = 32 Мб, где 512 байт — размер сектора).

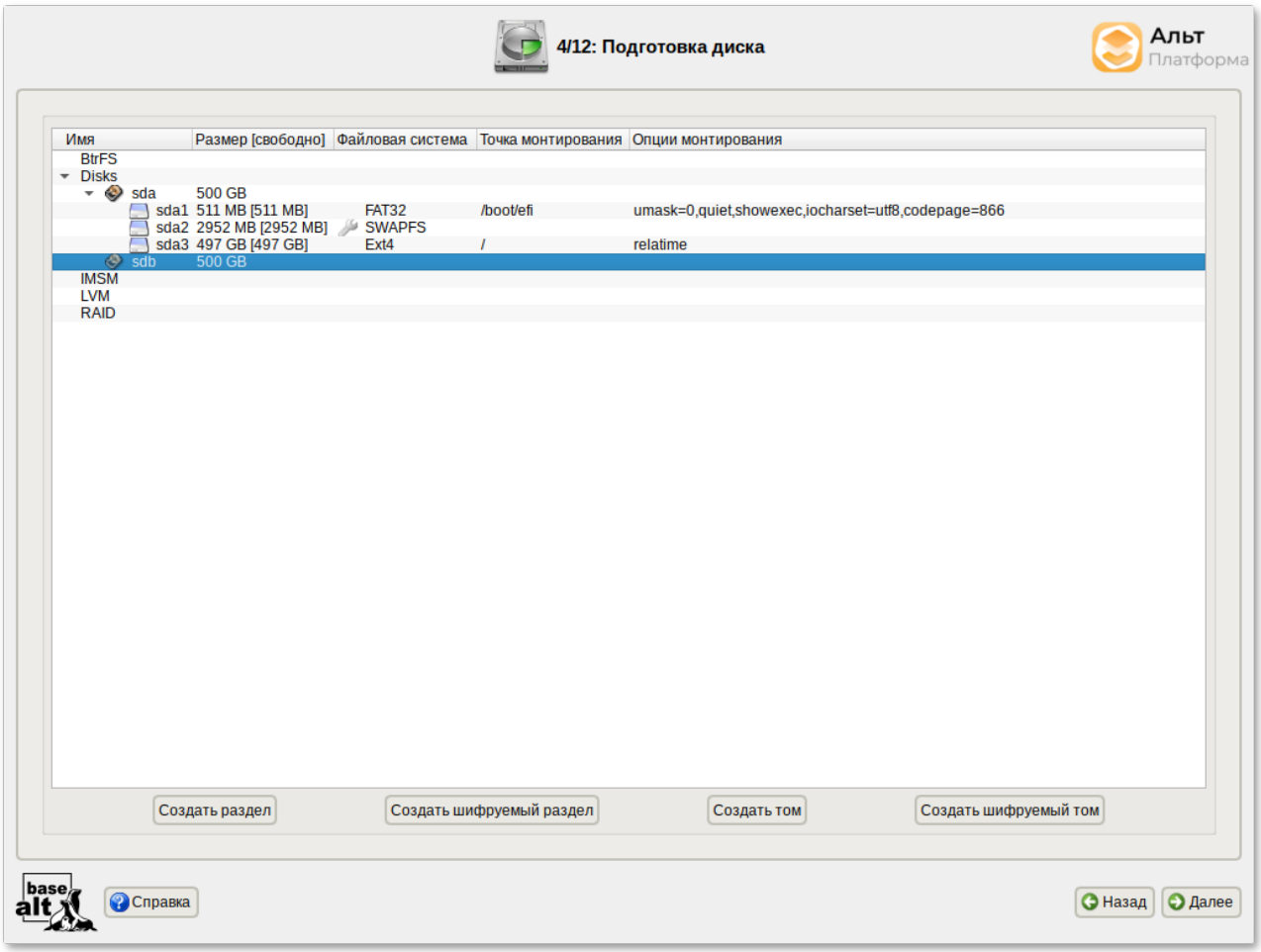
После того, как группа томов LVM создана, её можно использовать как обычный жёсткий диск, то есть внутри группы томов можно создавать тома (аналог раздела на физическом жёстком диске) и файловые системы внутри томов.



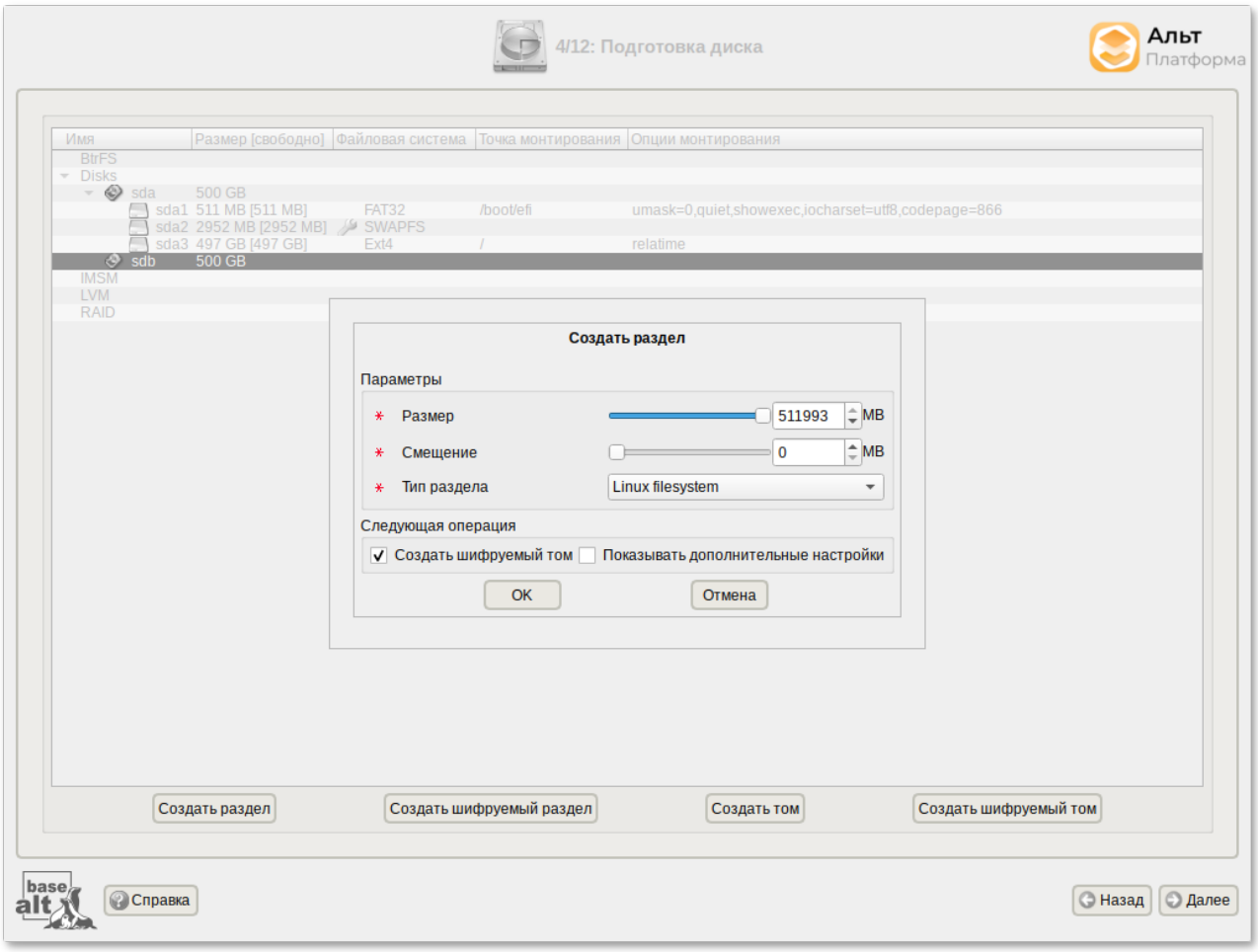
2.8.4.3. Создание шифрованных разделов

Программа установки Альт Платформа позволяет создавать шифрованные разделы.

Процесс создания шифрованного раздела ничем не отличается от процесса создания обычного раздела и инициируется нажатием на кнопку **`Создать шифруемый раздел`**:



После создания шифрованного раздела мастер, как и при создании обычного раздела, предложит создать на нём файловую систему и при необходимости потребует указать точку монтирования:

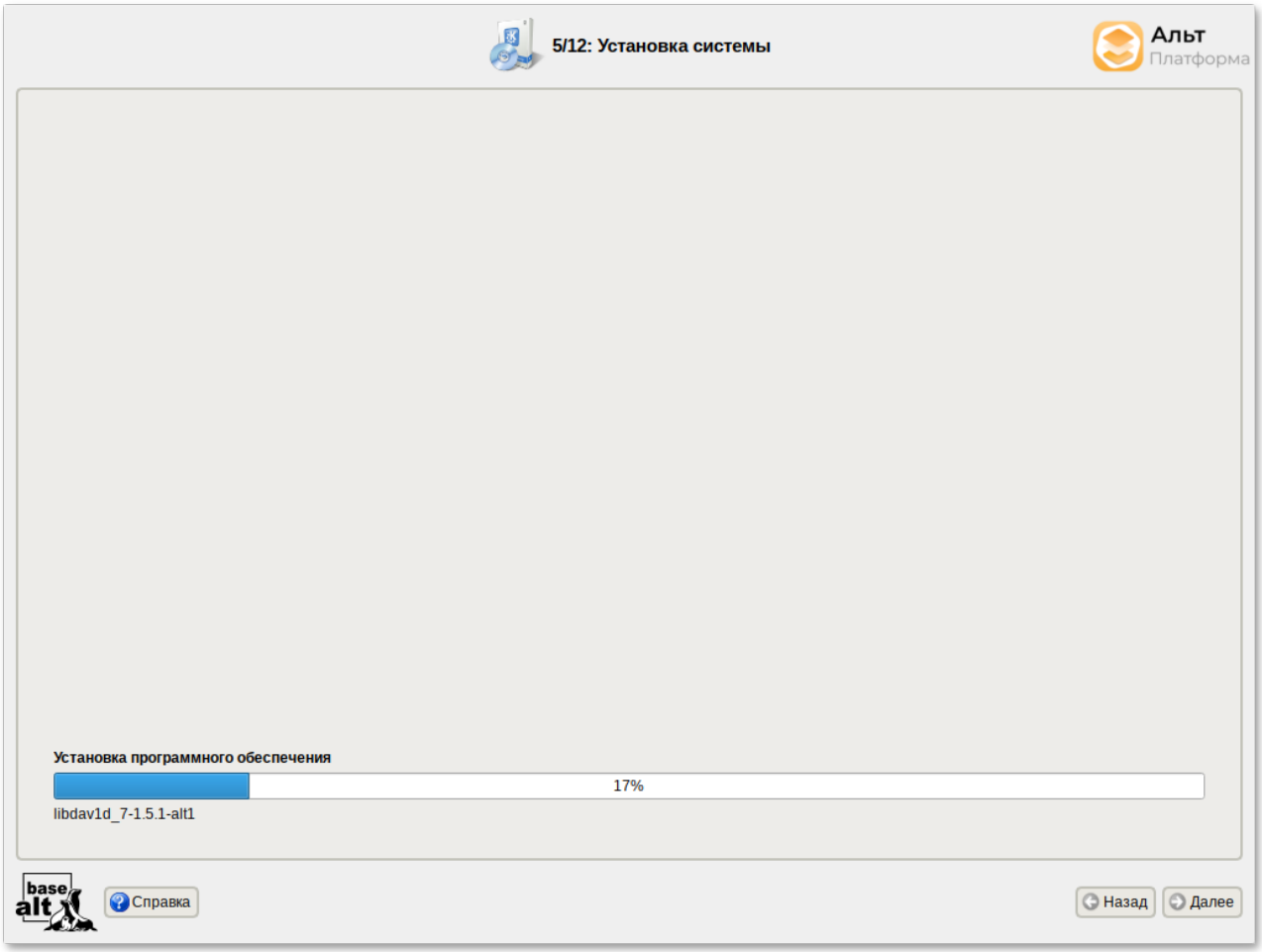


Предупреждение

Установка загрузчика на шифрованный раздел не поддерживается.

2.9. Установка системы

На данном этапе происходит распаковка ядра и установка набора программ, необходимых для работы дистрибутива ALT Platform Builder.



Установка происходит автоматически в два этапа:

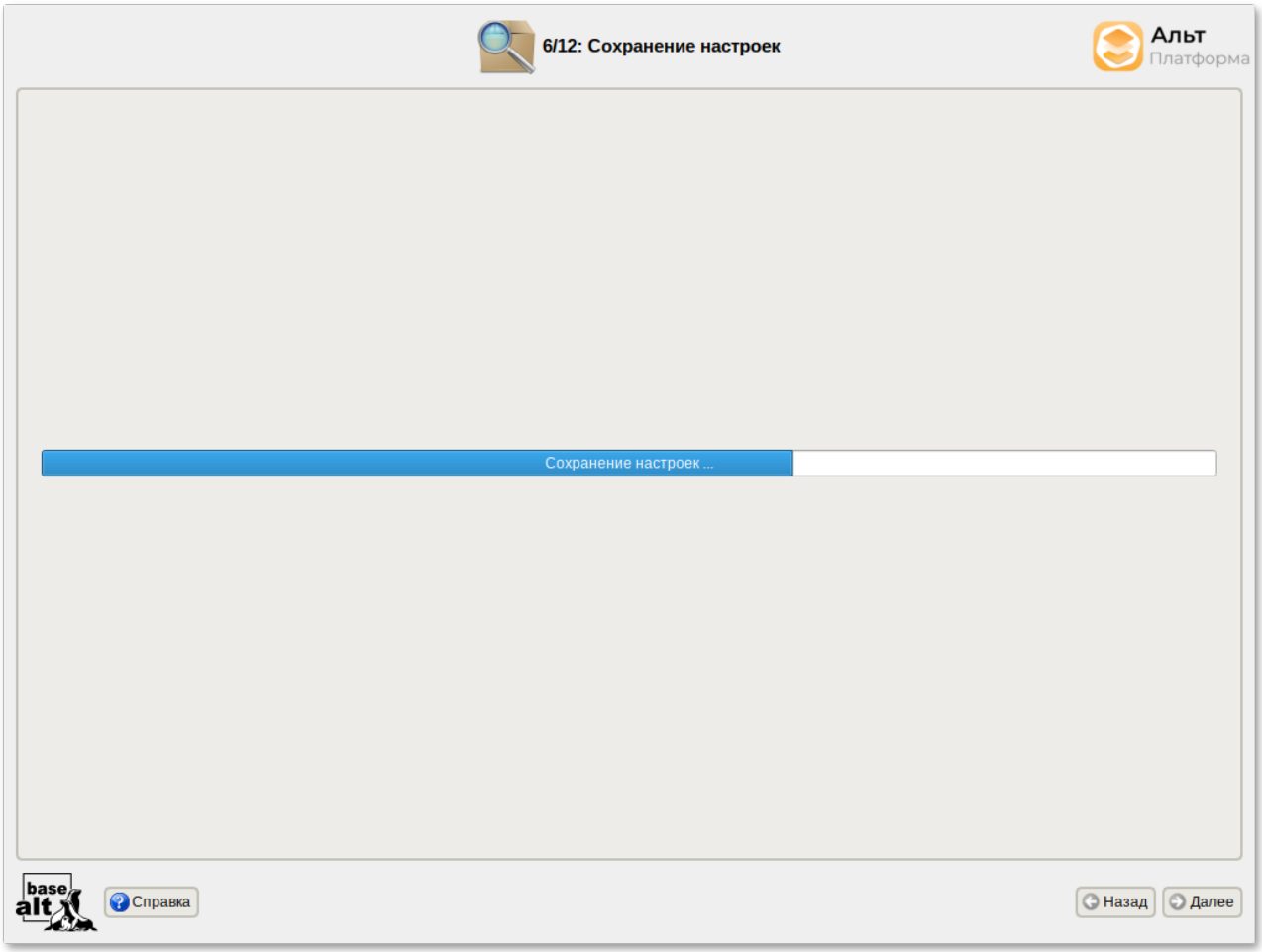
- получение пакетов;
- установка пакетов.

Получение пакетов осуществляется из источника, выбранного на этапе начальной загрузки. При сетевой установке (по протоколу FTP или HTTP) время выполнения этого шага будет зависеть от скорости соединения и может быть значительно большим в сравнении с локальной установкой.

2.10. Сохранение настроек

Начиная с данного этапа, программа установки работает с файлами только что установленной базовой системы. Все последующие изменения можно будет совершить после завершения установки посредством редактирования соответствующих конфигурационных файлов или при помощи модулей управления, включенных в дистрибутив.

По завершении установки базовой системы начинается шаг сохранения настроек. Он проходит автоматически и не требует вмешательства пользователя. На экране отображается индикатор выполнения.



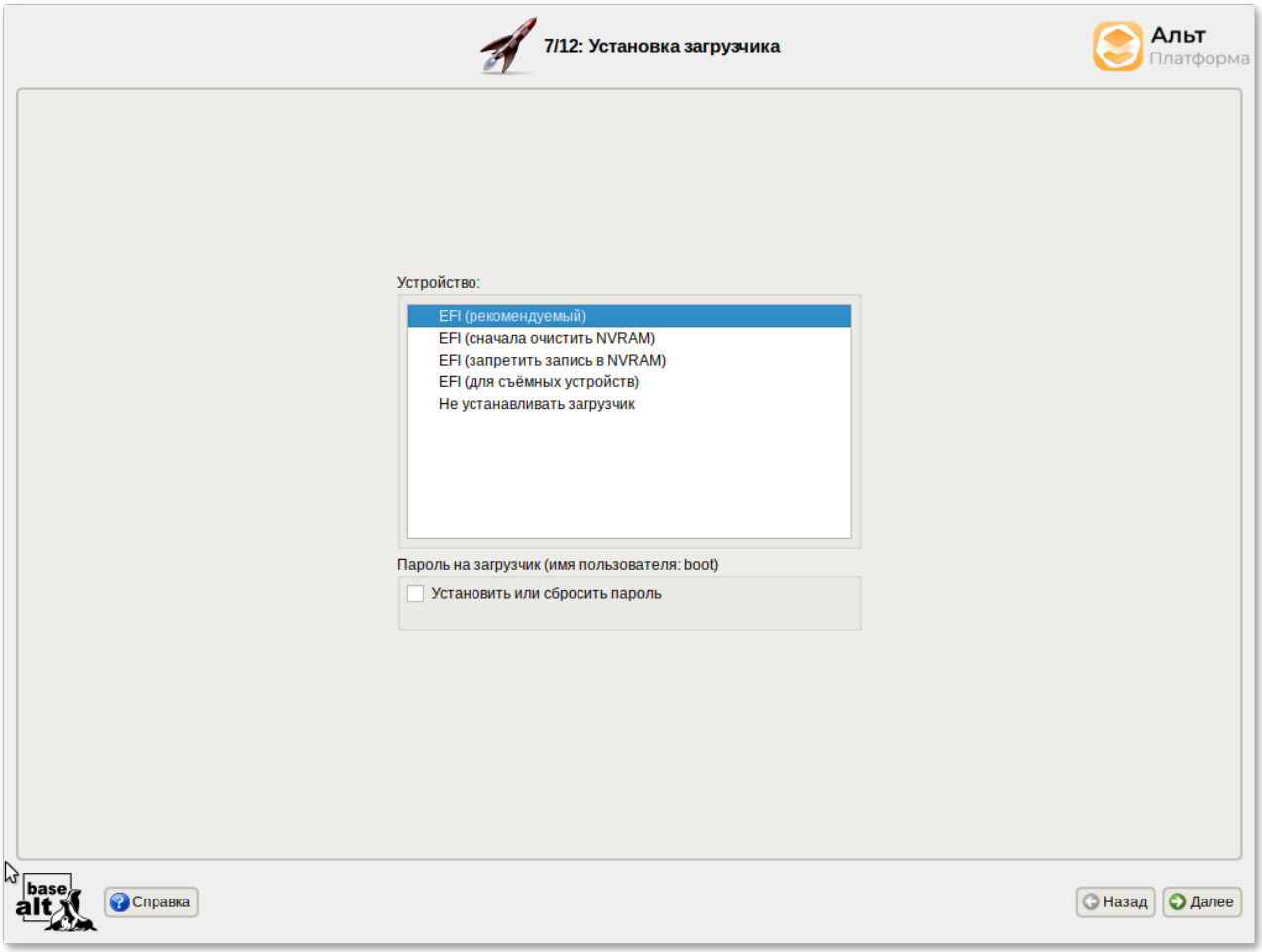
На этом шаге производится перенос настроек, выполненных на первых шагах установки, в только что установленную базовую систему. Производится также запись информации о соответствии разделов жесткого диска смонтированным на них файловым системам (заполняется конфигурационный файл **`/etc/fstab`**).

После сохранения настроек осуществляется автоматический переход к следующему шагу.

2.11. Установка загрузчика

Загрузчик ОС — это программа, которая позволяет загружать ALT Platform Builder и другие ОС, если они установлены на данной машине.

При установке на EFI модуль установки загрузчика предложит вариант установить загрузчик в специальный раздел **EFI** (рекомендуется выбрать автоматическое разбиение на этапе разметки диска для создания необходимых разделов для загрузки с EFI):



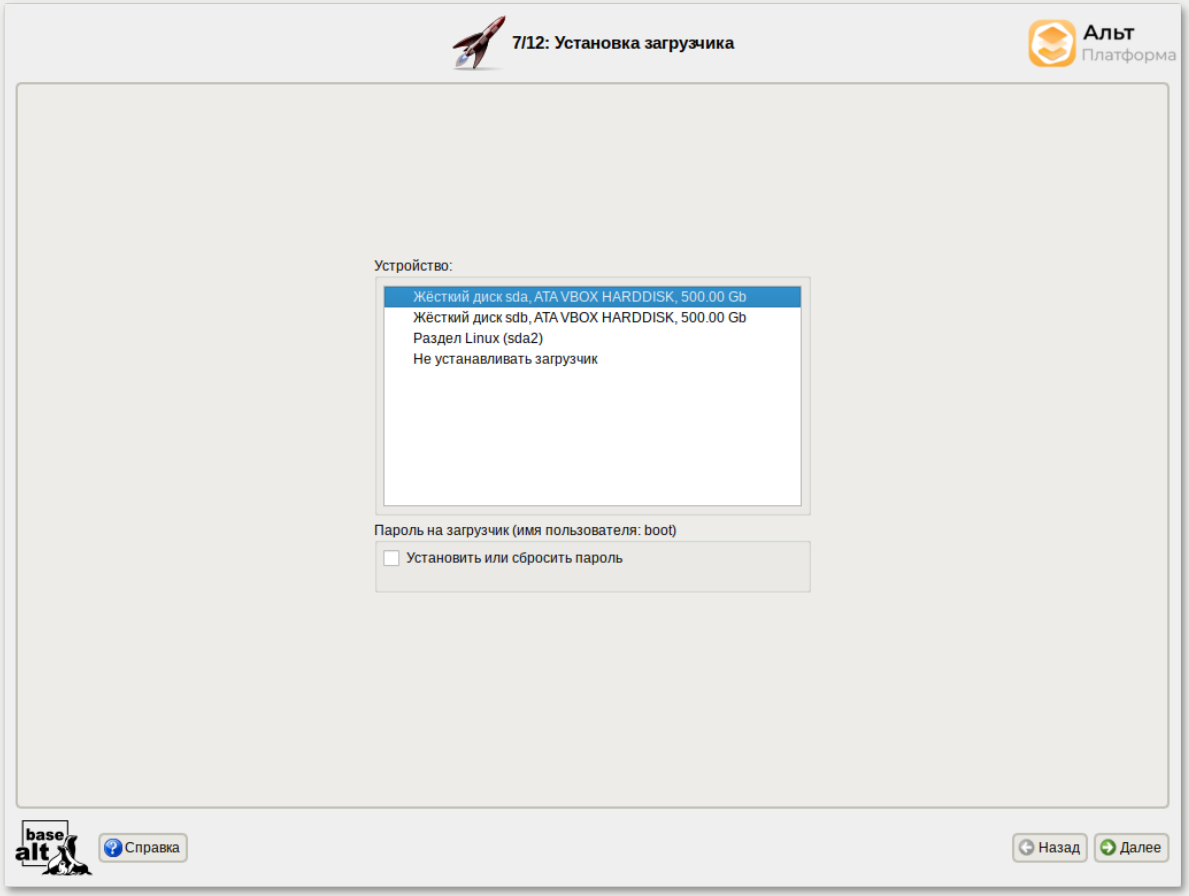
Варианты установки загрузчика при установке в режиме EFI:

- **EFI (рекомендуемый)** — при установке загрузчика в NVRAM добавляется запись, без которой большинство компьютеров не смогут загрузиться во вновь установленную ОС;
- **EFI (сначала очистить NVRAM)** — перед добавлением записи всё содержимое NVRAM сохраняется в **`/root/.install-log`**, затем из неё удаляются все загрузочные записи. Это освобождает NVRAM и гарантирует, что система загрузится в новую ОС;
- **EFI (запретить запись в NVRAM)** — этот вариант следует выбрать, только если инсталлятор не может создать запись в NVRAM или если заведомо известно, что запись в NVRAM может вывести компьютер из строя (вероятно, запись в NVRAM придётся создать после установки ОС средствами BIOS Setup);
- **EFI (для съёмных устройств)** — этот вариант следует выбрать, только если ОС устанавливается на съёмный накопитель. Этот вариант также можно использовать вместо варианта **EFI (запретить запись в NVRAM)** при условии, что это будет единственная ОС на данном накопителе. Создавать запись в NVRAM не потребуется.

Выбор варианта установки загрузчика зависит от вашего оборудования. Если не работает один вариант, попробуйте другие.

Примечание

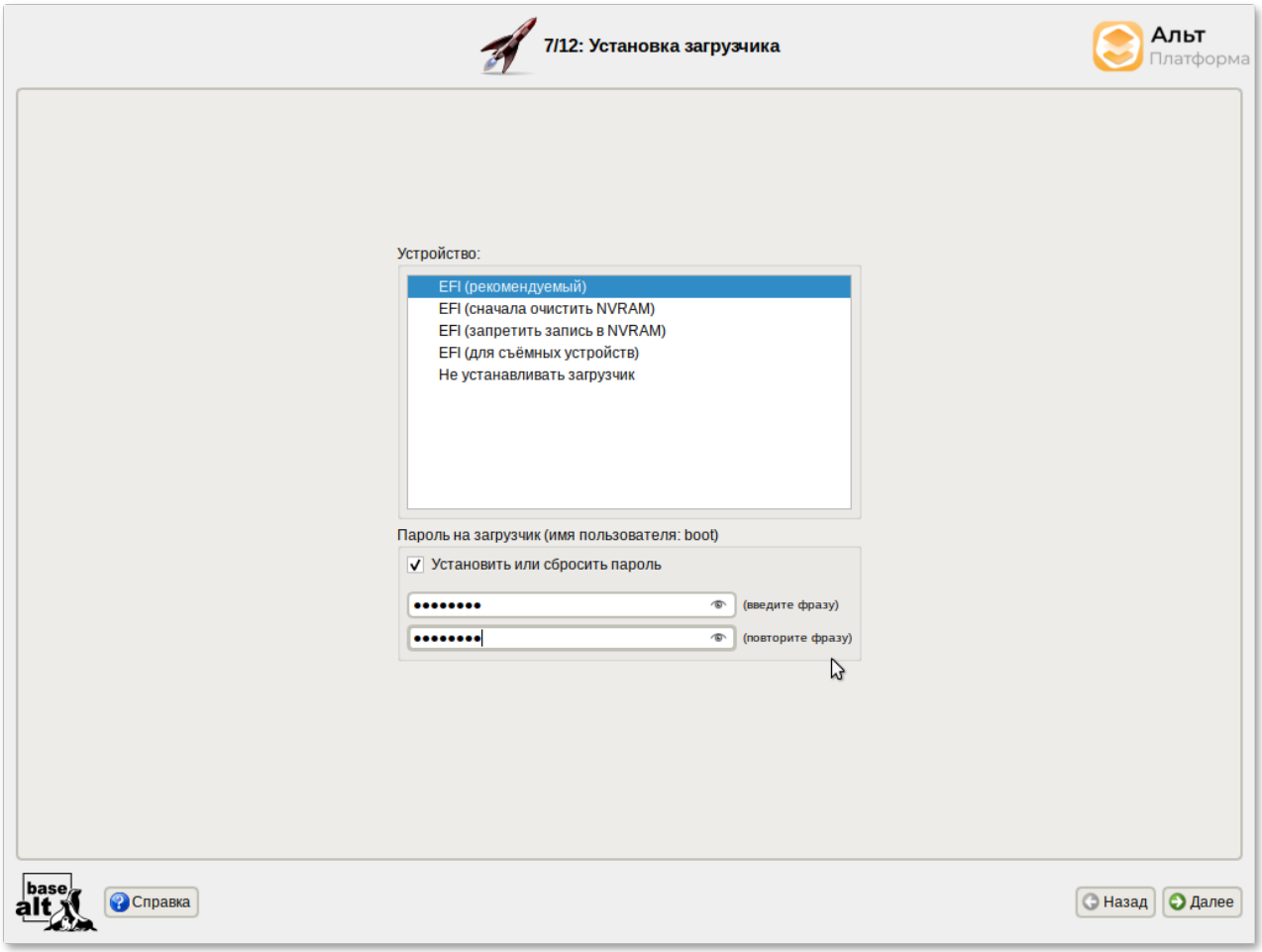
Установка загрузчика при установке в режиме Legacy:



Программа установки автоматически определяет, в каком разделе жёсткого диска следует располагать загрузчик для возможности корректного запуска ОС ALT Platform Builder. Положение загрузчика, в случае необходимости, можно изменить в списке **Устройство**, выбрав другой раздел.

Если же вы планируете использовать и другие ОС, уже установленные на этом компьютере, тогда имеет значение, на каком жёстком диске или в каком разделе будет расположен загрузчик.

Для ограничения доступа к опциям загрузки можно установить пароль на загрузчик. Для этого необходимо отметить пункт **Установить или сбросить пароль** и задать пароль в появившихся полях для ввода.



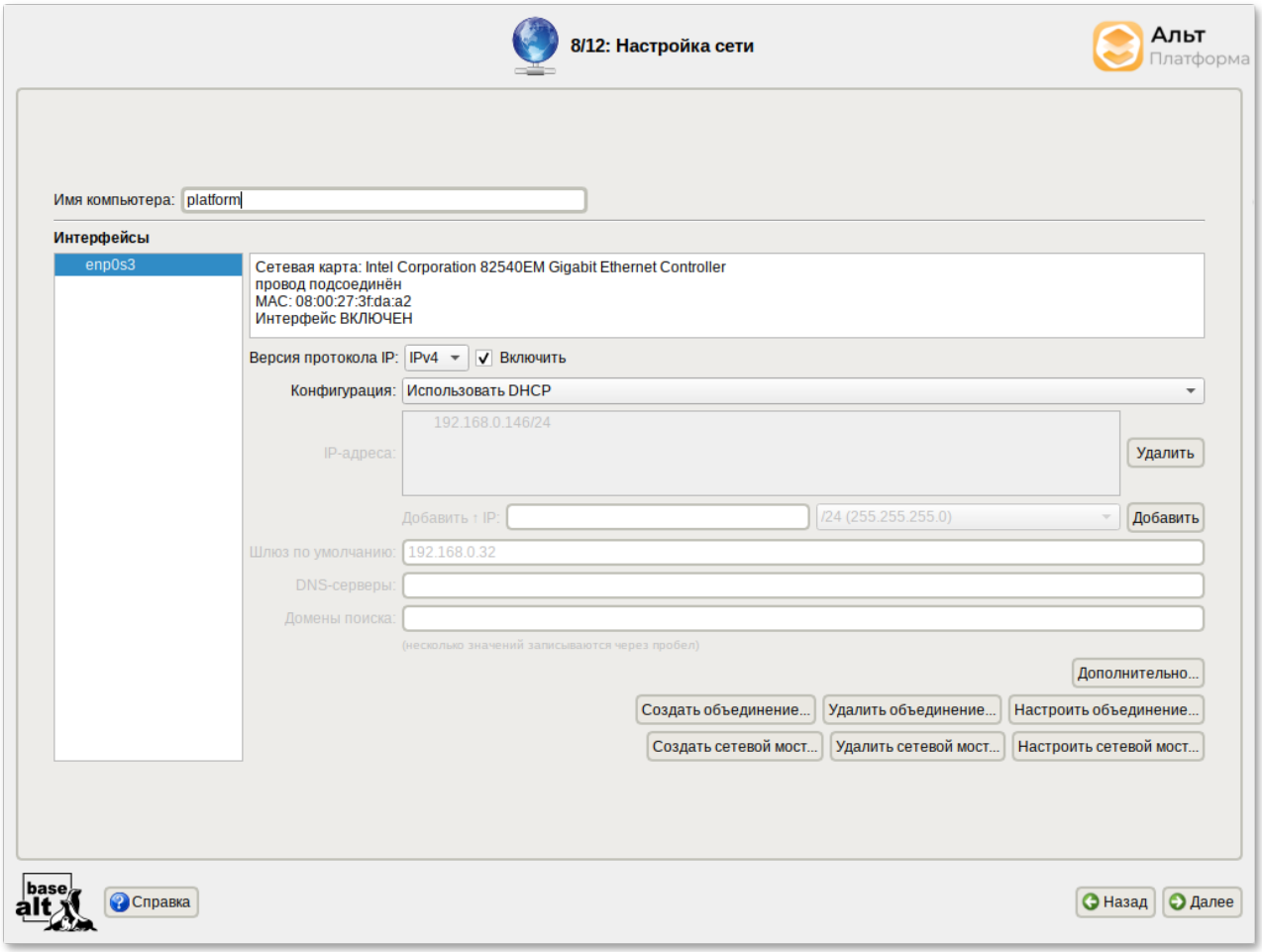
Примечание

При необходимости изменения опций загрузки при старте компьютера потребуется ввести имя пользователя «boot» и заданный на этом шаге пароль.

Для подтверждения выбора и продолжения работы программы установки необходимо нажать кнопку **Далее**.

2.12. Настройка сети

На этом этапе необходимо задать параметры работы сетевой карты и сетевые настройки: IP-адреса сетевых интерфейсов, DNS-сервер, шлюз и т.п. Конкретные значения будут зависеть от используемого вами сетевого окружения. Ручного введения настроек можно избежать при наличии в сети настроенного DHCP-сервера. В этом случае все необходимые сетевые параметры будут получены автоматически.



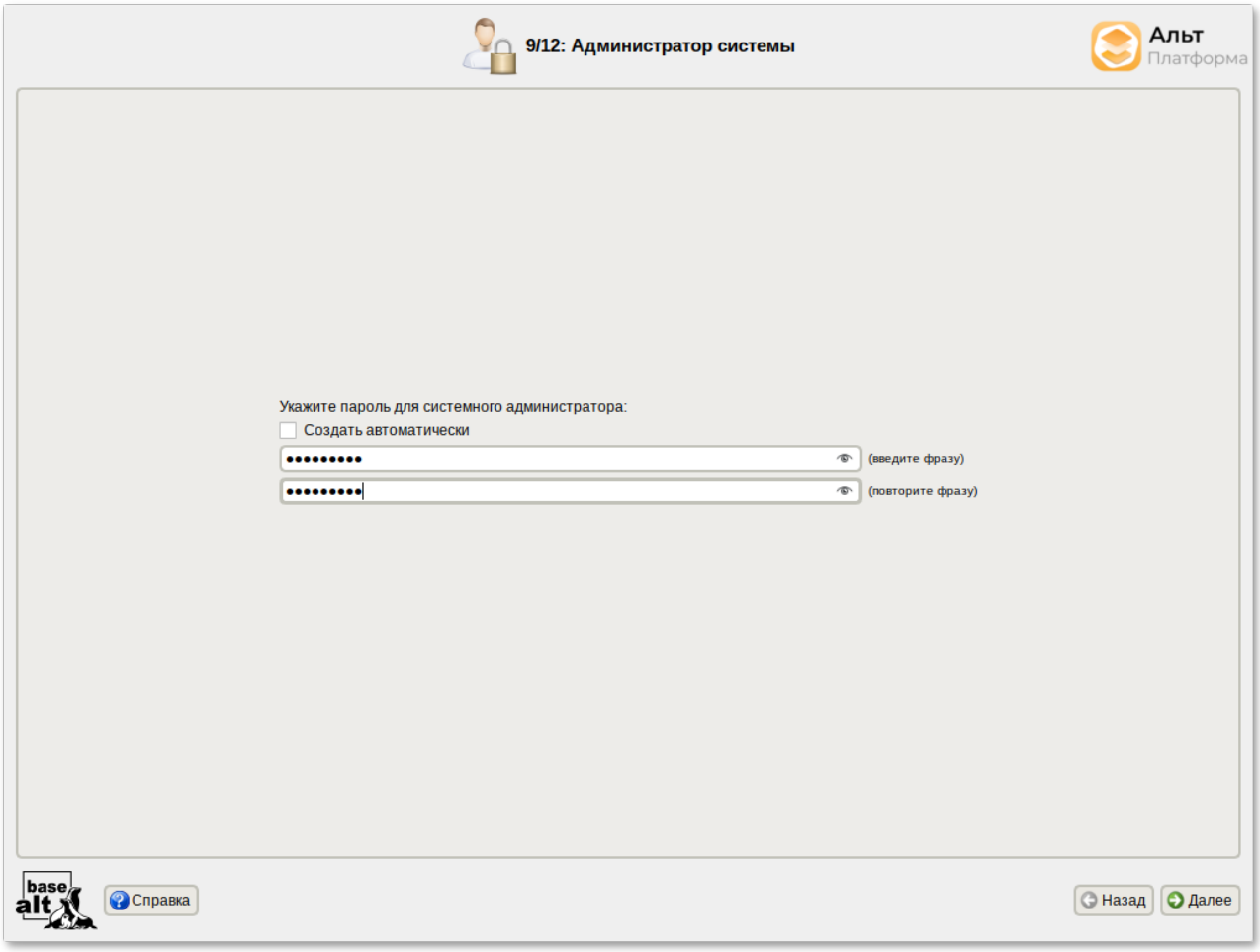
В окне **Настройка сети** доступны следующие поля:

- **Имя компьютера** — сетевое имя компьютера (это общий сетевой параметр, не привязанный к какому-либо конкретному интерфейсу);
- **Интерфейсы** — список доступных сетевых интерфейсов;
- **Версия протокола IP** — используемая версия IP-протокола (IPv4, IPv6);
- **Конфигурация** — способ назначения IP-адресов (**Использовать DHCP**, **Использовать Zeroconf**, **Вручную**);
- **IP-адреса** — пул назначенных IP-адресов (формируется из введённых в поле **Добавить ↑ IP**, для удаления адреса используется кнопка **Удалить**);
- **Добавить ↑ IP** — позволяет вручную ввести IP-адрес и выбрать маску сети из выпадающего списка. Для добавления адреса в пул IP-адресов нужно нажать кнопку **Добавить**;
- **Шлюз по умолчанию** — адрес маршрутизатора (шлюза), используемого по умолчанию;
- **DNS-серверы** — список DNS-серверов, используемых для разрешения доменных имён;
- **Домены поиска** — список доменов, по которым будет выполняться поиск (используется, например, при неполных DNS-запросах).

Для сохранения настроек сети и продолжения работы программы установки необходимо нажать кнопку **Далее**.

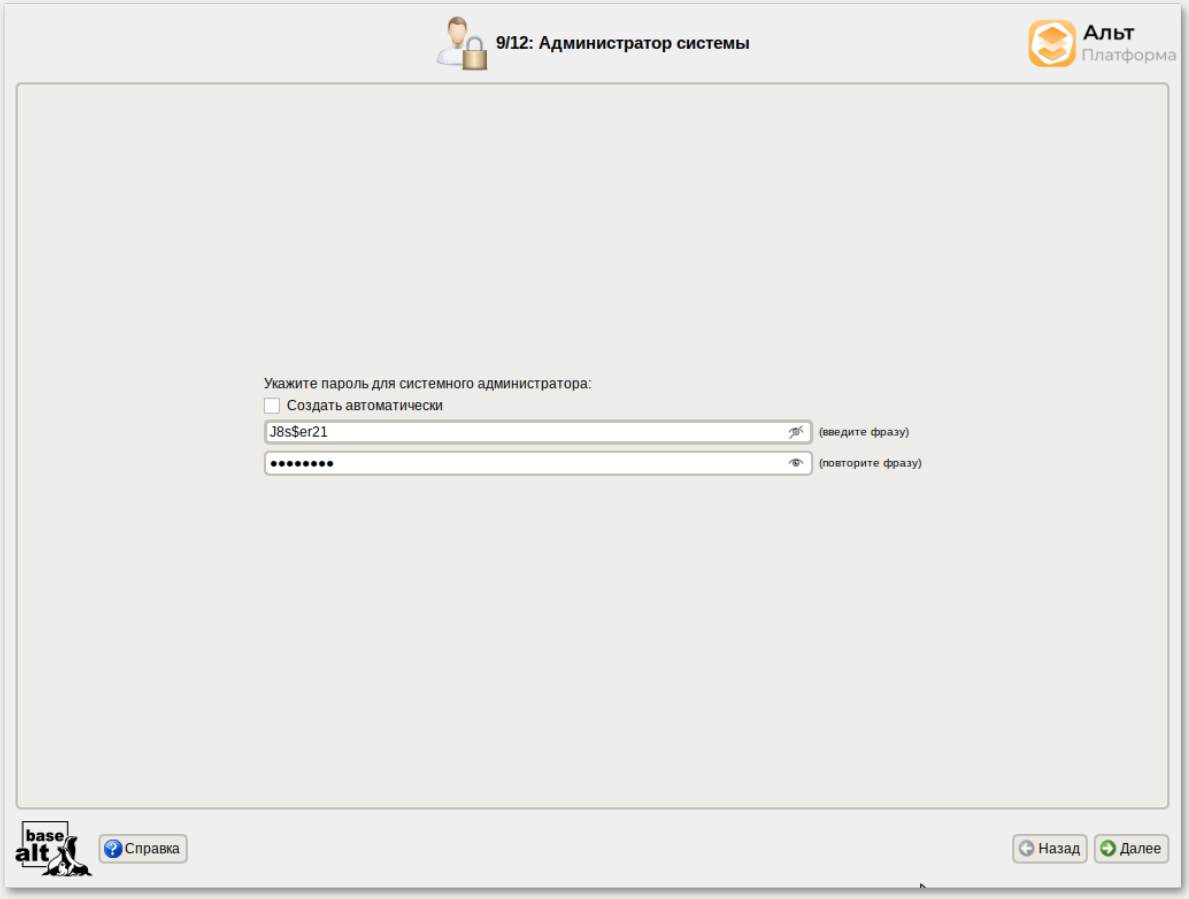
2.13. Администратор системы

На данном этапе загрузчик создает учетную запись администратора. В открывшемся окне необходимо ввести пароль учетной записи администратора (root). Чтобы исключить опечатки при вводе пароля, пароль учетной записи вводится дважды.



Примечание

Чтобы избежать последствий неверной раскладки клавиатуры можно просмотреть пароль, который будет сохранен. Для этого нажмите на значок глаза в поле ввода:



Для автоматической генерации пароля необходимо отметить пункт **`Создать автоматически`**. Система предложит пароль, сгенерированный автоматическим образом в соответствии с требованиями по стойкости паролей.

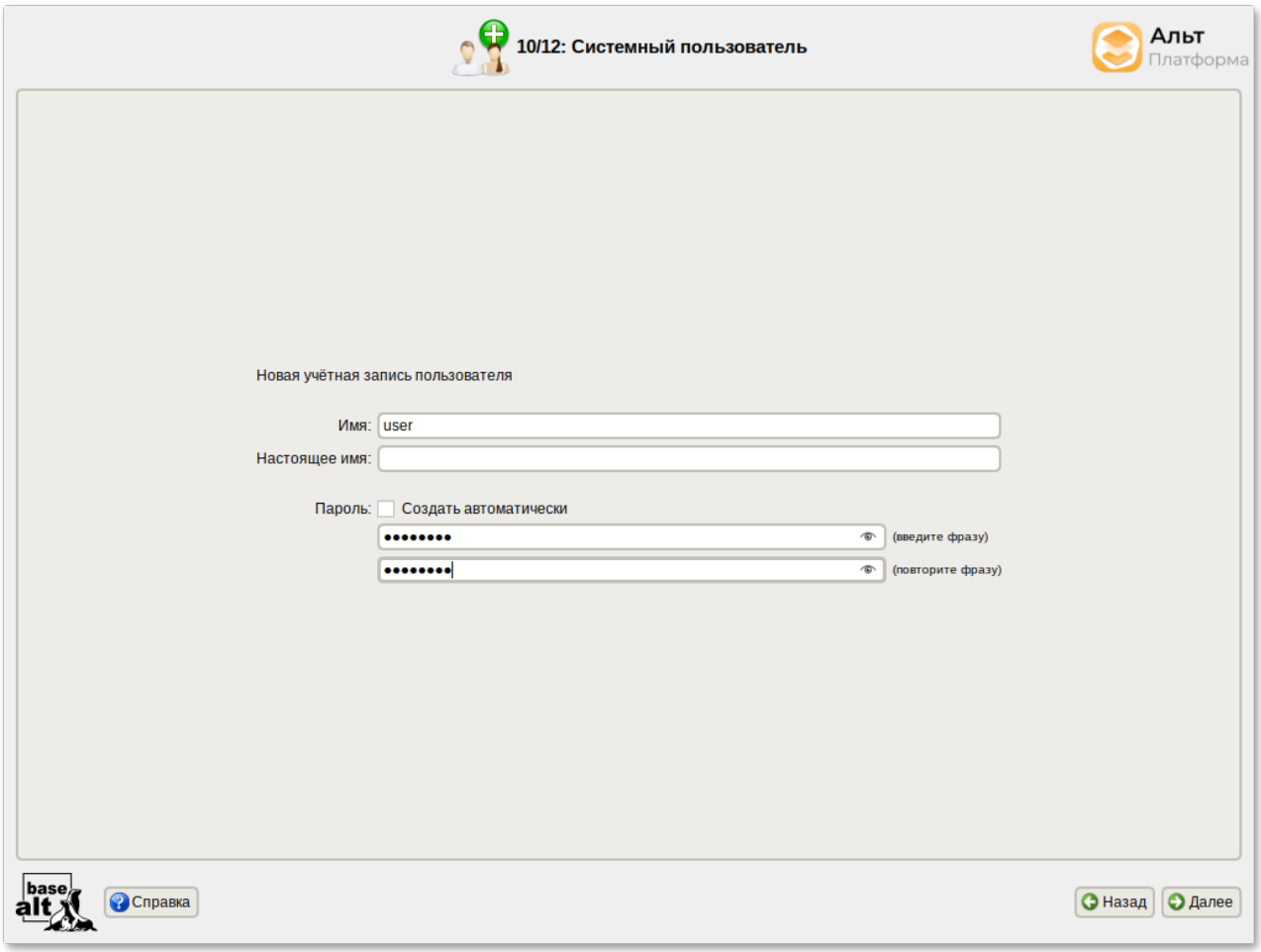
В любой системе Linux всегда присутствует один специальный пользователь — *администратор системы*, он же *суперпользователь*. Для него зарезервировано стандартное системное имя — root.

Администратор системы отличается от всех прочих пользователей тем, что ему позволено производить *любые*, в том числе самые разрушительные изменения в системе. Поэтому выбор пароля администратора системы — очень важный момент для *безопасности*. Любой, кто сможет ввести его правильно (узнать или подобрать), получит неограниченный доступ к системе. Даже ваши собственные неосторожные действия от имени root могут иметь катастрофические последствия для всей системы.

Подтверждение введенного (или сгенерированного) пароля учетной записи администратора (root) и продолжение работы программы установки выполняется нажатием кнопки **Далее**.

2.14. Системный пользователь

На данном этапе программа установки создает учетную запись системного пользователя (пользователя) Альт Платформа.



Помимо администратора (root) в систему необходимо добавить, по меньшей мере, одного обычного *системного пользователя*. Работа от имени администратора системы считается опасной, поэтому повседневную работу в Linux следует выполнять от имени ограниченного в полномочиях системного пользователя.

При добавлении системного пользователя предлагается ввести имя учётной записи пользователя. Имя учётной записи всегда представляет собой одно слово, состоящее только из строчных латинских букв (заглавные запрещены), цифр и символа подчёркивания «_» (причём цифра и символ «_» не могут стоять в начале слова).

Для того чтобы исключить опечатки, пароль пользователя вводится дважды. Пароль пользователя можно создать автоматически, по аналогии с автоматическим созданием пароля суперпользователя.

Для автоматической генерации пароля необходимо отметить пункт **`Создать автоматически`**. Система предложит пароль, сгенерированный автоматическим образом в соответствии с требованиями по стойкости паролей.

В процессе установки предлагается создать только одну учётную запись системного пользователя — от его имени можно выполнять задачи, не требующие привилегий суперпользователя. Учётные записи для всех прочих пользователей системы можно будет создать в любой момент после установки операционной системы.

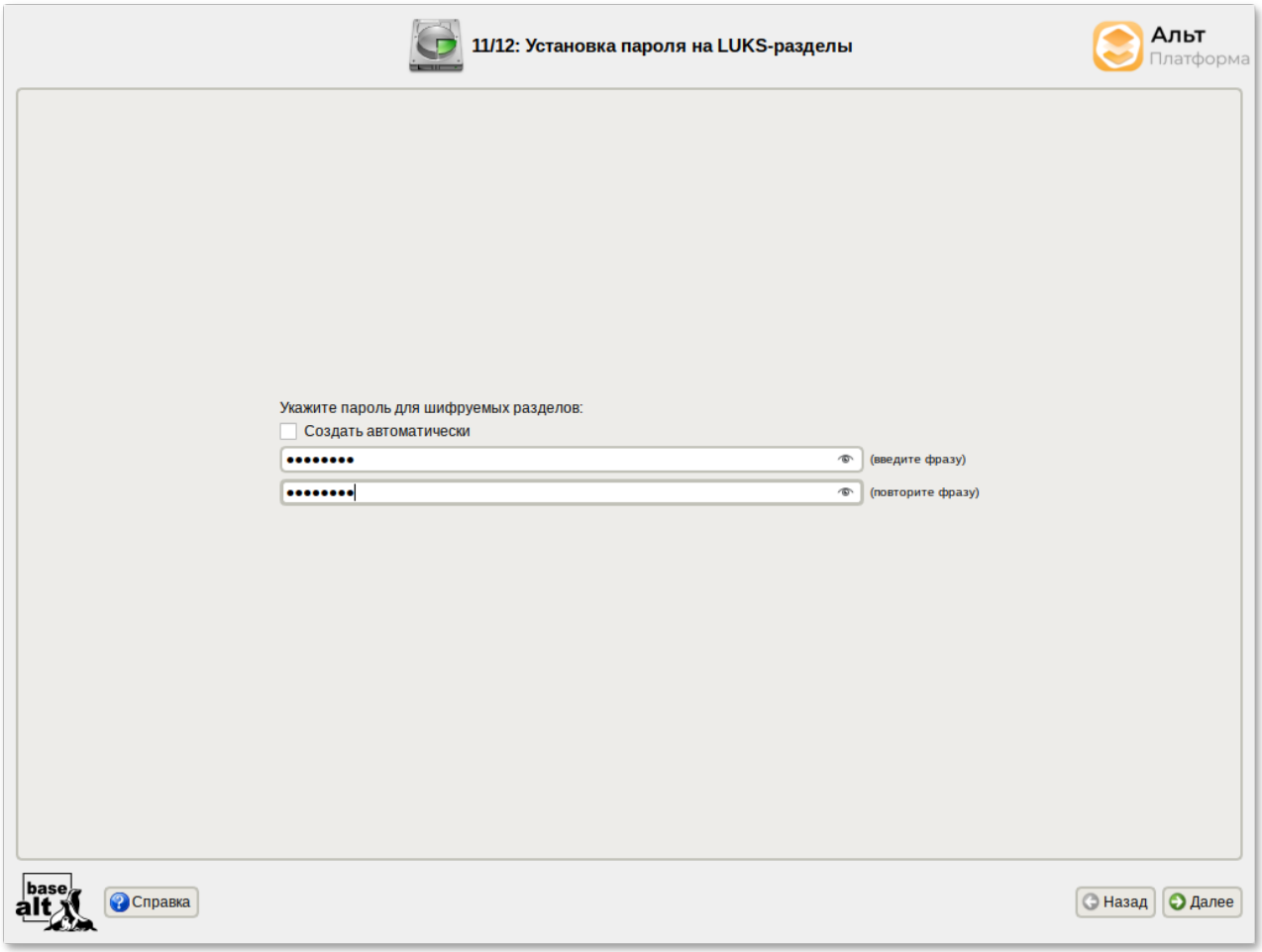
Подтверждение введенного (или сгенерированного) пароля учетной записи системного пользователя и продолжение работы программы установки выполняется нажатием кнопки **Далее**.

2.15. Установка пароля на шифрованные разделы

Примечание

Если вы не создавали шифруемые разделы, то этот шаг пропускается автоматически. В этом случае сразу переходите к главе Завершение установки.

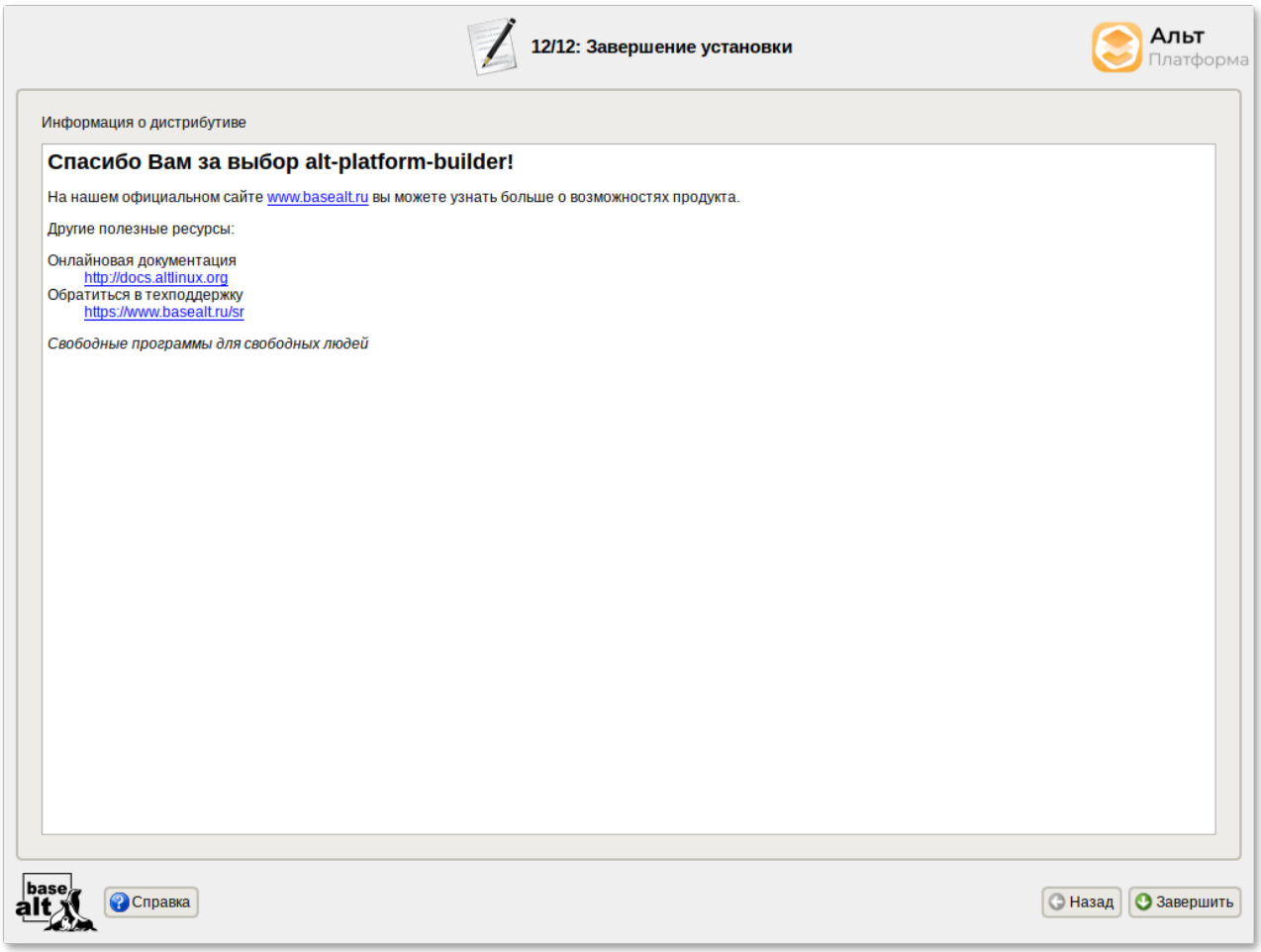
На этом этапе требуется ввести пароль для шифруемых разделов. Этот пароль потребуется вводить для того, чтобы получать доступ к информации на данных разделах.



Например, если вы зашифровали `/home`, то во время загрузки системы будет необходимо ввести пароль для этого раздела, иначе вы не сможете получить доступ в систему под своим именем пользователя.

2.16. Завершение установки

На экране последнего шага установки отображается информация о завершении установки ALT Platform Builder.



После нажатия кнопки **Завершить** автоматически начнется перезагрузка системы.

Не забудьте извлечь установочный диск (если это не происходит автоматически). Далее можно загружать установленную систему в обычном режиме.

2.17. Обновление системы до актуального состояния

После установки системы, её лучше сразу обновить до актуального состояния. Можно не обновлять систему и сразу приступать к работе только в том случае, если вы не планируете подключаться к сети или Интернету, не собираетесь устанавливать дополнительных программ.

Для обновления системы необходимо выполнить команды (с правами администратора):

```
# apt-get update
# apt-get dist-upgrade
# update-kernel
# apt-get clean
# reboot
```

Примечание

Получить права администратора можно, выполнив в терминале команду:

```
$ su -
```

или зарегистрировавшись в системе под именем **root**.

2.18. Первая помощь

Важно

В случае возникновения каких-либо неприятностей не паникуйте, а спокойно разберитесь в сложившейся ситуации. Linux не так уж просто довести до полной неработоспособности и утраты ценных данных. Поспешные действия отчаявшегося пользователя могут привести к плачевным результатам. Помните, что решение есть, и оно обязательно найдётся!

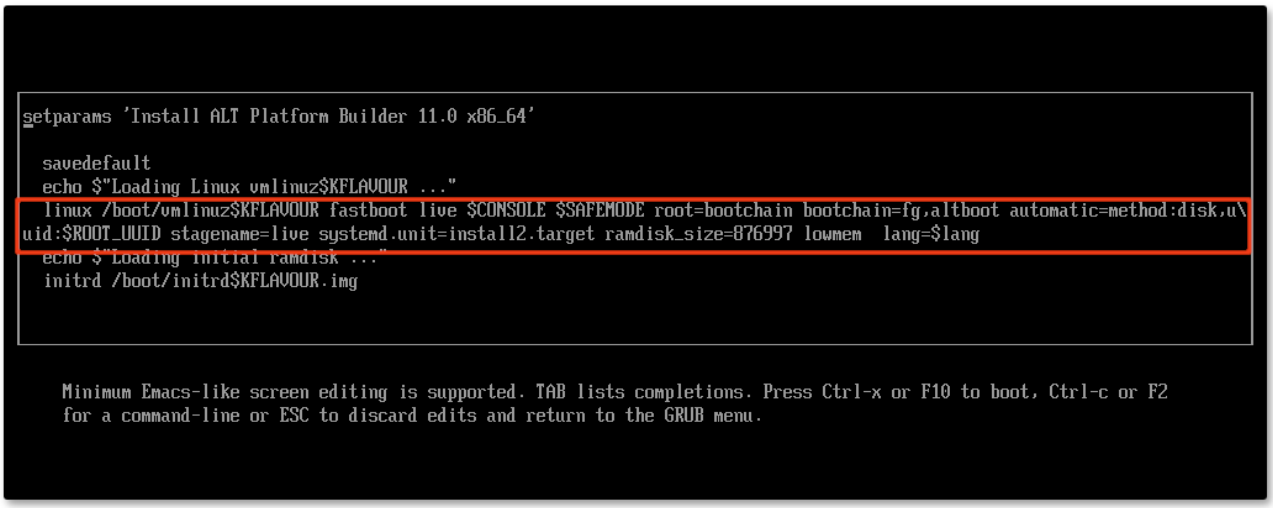
2.18.1. Проблемы при установке системы

Важно

При возникновении проблем с UEFI или Legacy/CSM рекомендуется изменить выбор используемого вида прошивки на другой. Не следует выбирать режим смешанной загрузки Legacy/UEFI! Рекомендуется отключить всевозможные оптимизации и ускорение UEFI-загрузки, а также отключить на время установки SecureBoot.

Если в системе не произошла настройка какого-либо компонента после стадии установки пакетов, не отчаивайтесь, доведите установку до конца, загрузитесь в систему и попытайтесь в спокойной обстановке повторить настройку.

Нажатием клавиши **E** можно вызвать редактор параметров текущего пункта загрузки. В открывшемся редакторе следует найти строку, начинающуюся с **linux /boot/vmlinuz**, в её конец дописать требуемые параметры, отделив пробелом и нажать **F10**.



Примеры параметров пункта загрузки:

- ***nomodeset*** — не использовать modeset-драйверы для видеокарты;
- ***vga=normal*** — отключить графический экран загрузки установщика;
- ***xdriver=vesa*** — явно использовать видеодрайвер vesa. Данным параметром можно явно указать нужный вариант драйвера;
- ***acpi=off noapic*** — отключение ACPI (управление питанием), если система не поддерживает ACPI полностью.

Если вы вообще не смогли установить систему (не произошла или не завершилась стадия установки пакетов), то сначала попробуйте повторить попытку в безопасном режиме (***apm=off acpi=off mce=off barrier=off vga=normal***). В безопасном режиме отключаются все параметры ядра, которые могут вызвать проблемы при загрузке. В этом режиме установка будет произведена без поддержки APIC. Возможно, у вас какое-то новое или нестандартное оборудование, но может оказаться, что оно отлично настраивается со старыми драйверами.

Если вы хотите получить точный ответ, то сообщите, пожалуйста, подробный состав вашего оборудования и подробное описание возникшей проблемы в нашей системе отслеживания ошибок.

2.18.2. Проблемы с загрузкой системы

Если не загружается ни одна из установленных операционных систем, то значит, есть проблема в начальном загрузчике. Такие проблемы могут возникнуть после установки системы, в случае если загрузчик все-таки не установлен или установлен с ошибкой. При установке или переустановке Windows на вашем компьютере загрузчик Linux будет перезаписан в принудительном порядке, и станет невозможно запускать Linux.

Повреждение или перезапись загрузчика никак не затрагивает остальные данные на жёстком диске, поэтому в такой ситуации очень легко вернуть работоспособность: для этого достаточно восстановить загрузчик.

Если у вас исчез загрузчик другой операционной системы или другого производителя, то внимательно почитайте соответствующее официальное руководство на предмет его восстановления. Но в большинстве случаев вам это не потребуется, так как загрузчик, входящий в состав Альт Платформа, поддерживает загрузку большинства известных операционных систем.

Для восстановления загрузчика достаточно любым доступным способом загрузить Linux и получить доступ к тому жёсткому диску, на котором находится повреждённый загрузчик. Для этого проще всего воспользоваться *восстановительным режимом*, который предусмотрен на установочном диске дистрибутива (пункт **Восстановление системы**).

Загрузка восстановительного режима заканчивается приглашением командной строки: `[root@localhost /]#` . Начиная с этого момента, система готова к вводу команд.

В большинстве случаев для восстановления загрузчика можно просто воспользоваться командой **`fixmbr`** без параметров. Программа попытается переустановить загрузчик в автоматическом режиме.

2.18.3. Полезные ссылки

Если у вас что-то не получается, вы всегда можете поискать решение на ресурсах, указанных в разделе Техническая поддержка продуктов «Базальт СПО».

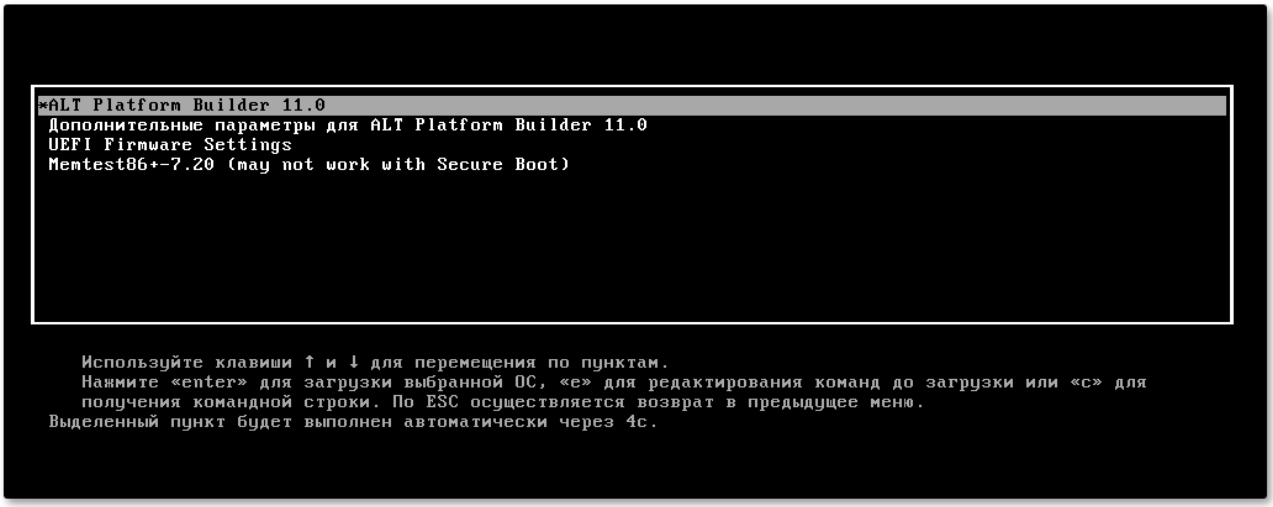
Глава 3. Начало использования Альт Платформа

В этой главе рассматривается загрузка установленной операционной системы.

3.1. Загрузка системы

Запуск ALT Platform Builder выполняется автоматически после запуска компьютера и отработки набора программ BIOS.

На экране появляется меню, в котором перечислены возможные варианты загрузки операционной системы.



Важно

При первом старте, в условиях установки нескольких ОС на один компьютер, возможно отсутствие в загрузочном меню пункта/пунктов с другой/другими операционными системами, они будут добавлены в список при последующей перезагрузке. Все перечисленные в меню после перезагрузки варианты могут быть загружены загрузчиком Linux.

Стрелками клавиатуры **Вверх** и **Вниз** выберите нужную операционную систему. Дополнительно к основным вариантам запуска ОС из этого меню можно загрузить Linux в безопасном режиме или запустить проверку памяти.

Загрузка операционной системы по умолчанию (первая в списке) начинается автоматически после небольшого времени ожидания (обычно несколько секунд). Нажав клавишу **Enter**, можно начать загрузку немедленно.

Нажатием клавиши **E** можно вызвать редактор параметров текущего пункта загрузки. Если система настроена правильно, то редактировать их нет необходимости.

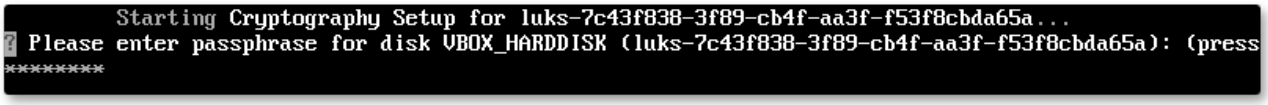
Примечание

Если при установке системы на этапе был установлен пароль на загрузчик, потребуется ввести имя пользователя «boot» и заданный на шаге Установка загрузчика пароль.

Загрузка операционной системы может занять некоторое время, в зависимости от производительности компьютера. Основные этапы загрузки операционной системы — загрузка ядра, подключение (монтирование) файловых систем, запуск системных служб — периодически могут дополняться проверкой файловых систем на наличие ошибок. В этом случае время ожидания может быть занять больше времени, чем обычно. Подробную информацию о шагах загрузки можно получить, нажав клавишу **Esc**.

3.2. Получение доступа к зашифрованным разделам

В случае, если вы создали шифрованный раздел, вам потребуется вводить пароль при обращении к этому разделу.



Например, если был зашифрован домашний раздел **/home**, то для того, чтобы войти в систему под своим именем пользователя, вам потребуется ввести пароль этого раздела и затем нажать **Enter**.

Важно

Если не ввести пароль за отведенный промежуток времени, то загрузка системы завершится ошибкой. В этом случае вам следует перезагрузить систему, нажав для этого два раза **Enter**, а затем клавиши **Ctrl**+**Alt**+**Delete**.

3.3. Вход в систему

3.3.1. Вход и работа в консольном режиме

Стандартная установка ALT Platform Builder включает базовую систему, работающую в консольном режиме.

При загрузке в консольном режиме работа загрузчика Альт Платформа завершается запросом на ввод логина и пароля учетной записи. В случае необходимости на другую консоль можно перейти, нажав **Ctrl**+**Alt**+**F2**.

Для дальнейшего входа в систему необходимо ввести логин и пароль учетной записи пользователя.

В случае успешного прохождения процедуры аутентификации и идентификации будет выполнен вход в систему. ОС ALT Platform Builder перейдет к штатному режиму работы и предоставит дальнейший доступ к консоли.

Примечание

Сразу после загрузки в консоли будут показаны имя и IP-адрес компьютера. К узлу можно подключиться по SSH, например (потребуется ввести пароль пользователя):

```
$ ssh user@192.168.0.160
```

3.3.2. Виртуальная консоль

В процессе работы ОС ALT Platform Builder активно несколько виртуальных консолей. Каждая виртуальная консоль доступна по одновременному нажатию клавиш **Ctrl**, **Alt** и функциональной клавиши с номером этой консоли от **F1** до **F6**.

При установке системы в профиле по умолчанию на первой виртуальной консоли пользователь может зарегистрироваться и работать в графическом режиме. При нажатии **Ctrl**+**Alt**+**F1** осуществляется переход на первую виртуальную консоль в графический режим.

Двенадцатая виртуальная консоль (**Ctrl**+**Alt**+**F12**) выполняет функцию системной консоли — на неё выводятся сообщения о происходящих в системе событиях.

Глава 4. Настройка модуля Сервер обновлений

4.1. Центр управления системой

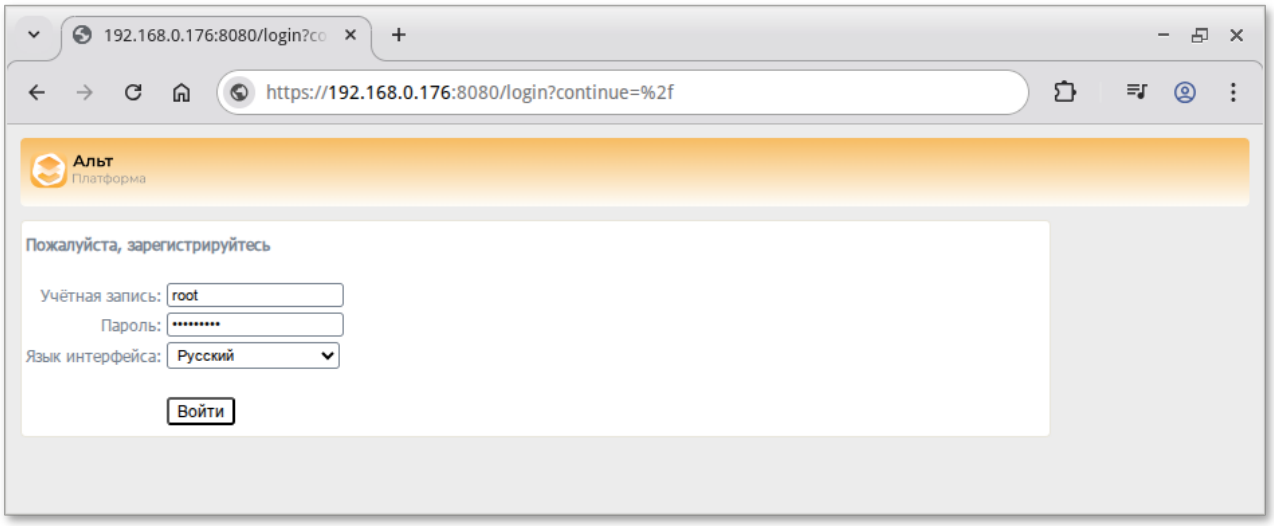
Центр управления системой (ЦУС) представляет собой удобный интерфейс для выполнения наиболее востребованных административных задач: добавление и удаление пользователей, настройка сетевых подключений, просмотр информации о состоянии системы и т.п.

Центр управления системой состоит из нескольких независимых диалогов-модулей. Каждый модуль отвечает за настройку определённой функции или свойства системы. Модули ЦУС имеют справочную информацию.

ЦУС имеет веб-ориентированный интерфейс, позволяющий управлять данным компьютером с любого другого компьютера сети.

Работа с ЦУС может происходить из любого веб-браузера. Для начала работы необходимо перейти по адресу **`https://ip-адрес:8080/`**.

При запуске центра управления системой необходимо ввести в соответствующие поля имя пользователя (**root**) и пароль пользователя:



После этого будут доступны все возможности ЦУС на той машине, к которой было произведено подключение через веб-интерфейс.

4.2. Модуль ЦУС Сервер обновлений

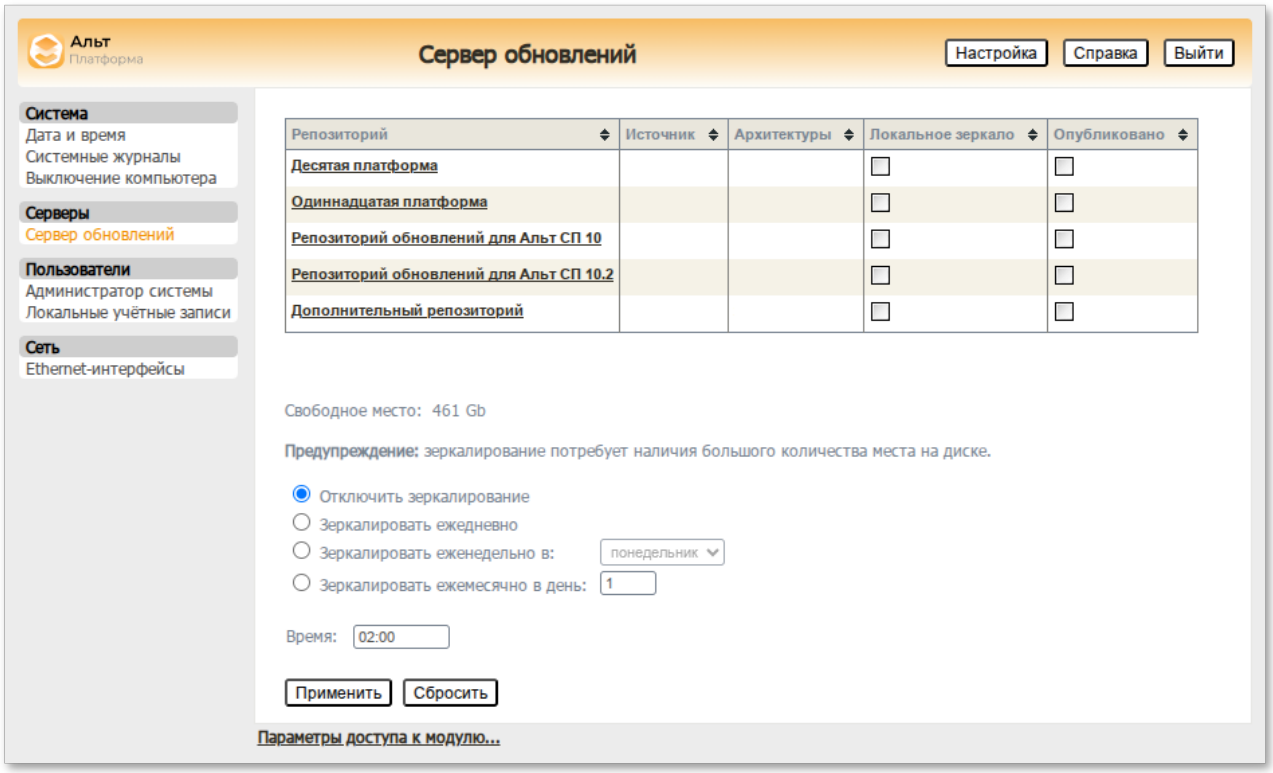
Сервер обновлений — технология, позволяющая настроить автоматическое обновление программного обеспечения, установленного на клиентских машинах (рабочих местах), работающих под управлением ОС Альт.

Кроме того, сервер обновлений предоставляет локальный доступ ко всем пакетам репозитория Альт Платформа, используемым для разработки и/или сборки ПО.

Модуль ЦУС **Сервер обновлений** (пакет *alterator-mirror*, раздел **Серверы**) предназначен для зеркалирования репозиториев и их публикации.

Модуль позволяет:

- просмотреть информацию о зеркалируемых репозиториях;
- выбрать репозитории для зеркалирования из предложенного списка;
- настроить периодичность зеркалирования;
- задать параметры каждого зеркалируемого репозитория: источник, архитектуру, параметры публикации;
- создать собственный дополнительный репозиторий.



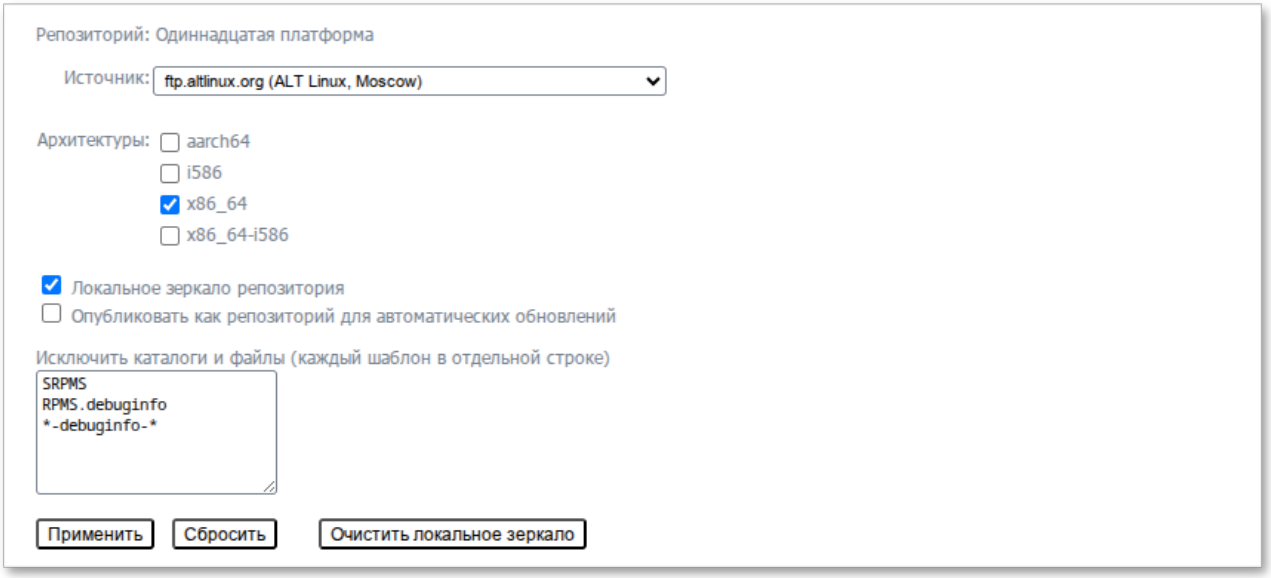
4.2.1. Настройки репозитория

При выборе репозитория открываются его настройки. При настройке репозитория необходимо указать:

- источник (сервер, с которого будет выполняться загрузка);
- архитектуры для зеркалирования (при наличии нескольких следует выбрать нужные).

Примечание

При выборе любой архитектуры автоматически добавляется источник с архитектурой noarch.



Сервер обновлений предоставляет возможность автоматически настроить обновление клиентских машин в нужном режиме.

Локальное зеркало репозитория

В этом режиме на сервере создаётся копия удалённого репозитория. Клиентские машины могут загружать пакеты с локального сервера по протоколам HTTP, HTTPS, FTP или rsync (для каждого протокола требуется отдельная настройка соответствующей службы).

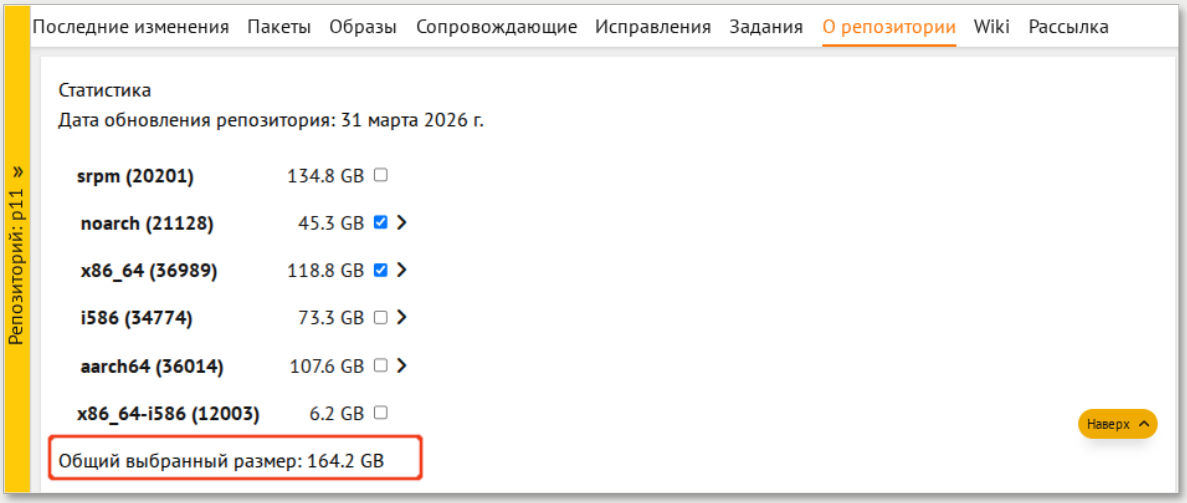
Важно

Зеркалирование требует значительного объёма дискового пространства. Полный размер зеркала зависит от выбранной ветки, архитектур и включённых компонентов.

Оценить объем зеркала можно на сайте packages.altlinux.org:

1. Выбрать ветку репозитория.
2. Перейти в раздел **О репозитории**.
3. Снять отметки с ненужных компонентов (архитектур).

После этого будет отображён общий размер выбранных компонентов — размер репозитория.:



Для уменьшения объёма загружаемых данных можно исключить из синхронизации отдельные каталоги и файлы, например, не скачивать пакеты с исходным кодом и пакеты с отладочной информацией:

```
SRPMS
*-debuginfo-*
```

Шаблоны указываются по одному в строке. Символ «*» обозначает произвольную последовательность символов.

Опубликовать как репозиторий для автоматических обновлений

В этом режиме публикуется:

- либо URL внешнего репозитория;
- либо (при включённом локальном зеркале) адрес сервера обновлений.

Это позволяет клиентским машинам автоматически настроить менеджер пакетов для работы с выбранным источником.

Примечание

При включении режима **Опубликовать как репозиторий для автоматических обновлений** Avahi:

- публикует сервис _apt._tcp в сети;
- сообщает клиентам:
  - имя сервера (например, server.local);
  - URL (например, http://server.local/mirror);
  - дополнительные параметры (архитектура, ветка и т.д.).

Для автоматических обновлений на клиентских машинах необходимо настроить модуль **Обновление системы**, включив режим **Обновление системы, управляемое сервером**.

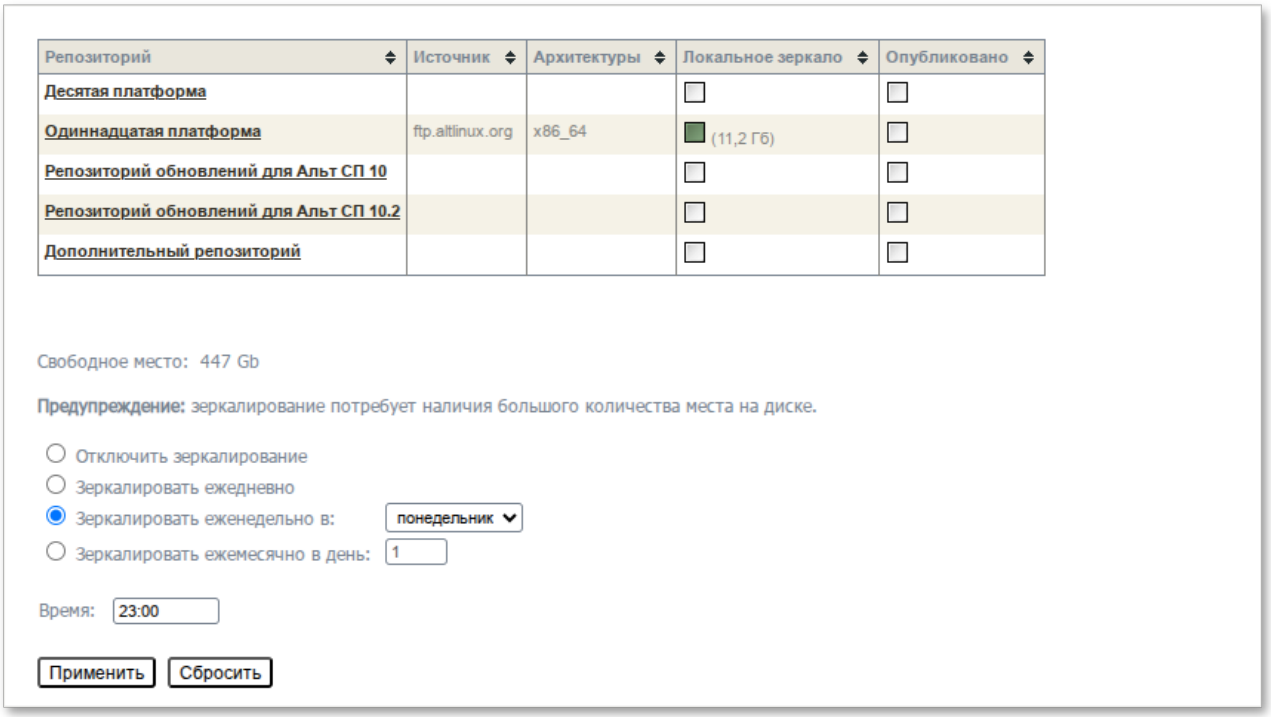
Настройка локального репозитория завершается нажатием кнопки **Применить**.

Примечание

По умолчанию локальное зеркало размещается в каталоге **/srv/public/mirror**. Чтобы использовать другой каталог, его необходимо примонтировать в **/srv/public/mirror**, добавив строку в файл **/etc/fstab**:

```
/media/disk/localrepo /srv/public/mirror none rw,bind,auto 0 0
```

где **/media/disk/localrepo** — каталог для хранения локального репозитория.



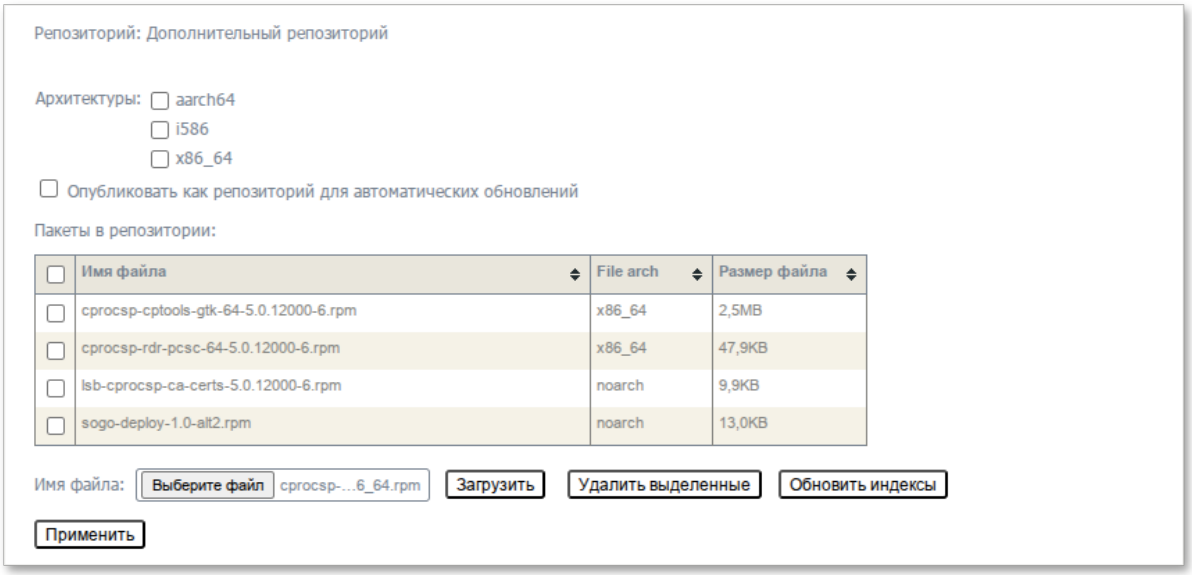
Примечание

Если в каталогах **/srv/public/mirror/<репозиторий>/branch/<архитектура>/base/** отсутствуют файлы **pkglist.***, это означает, что зеркалирование ещё не завершено (не все файлы загружены на сервер обновлений).

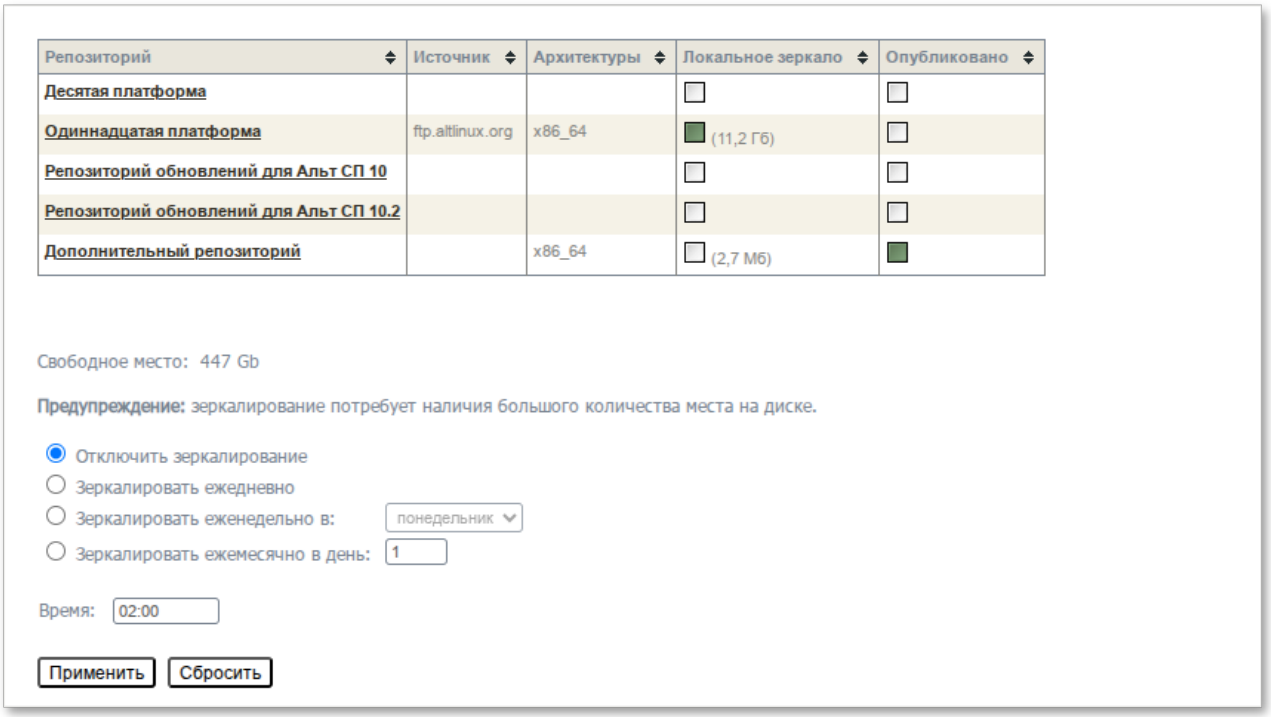
4.2.2. Использование дополнительного репозитория

Создание собственного репозитория:

1. На странице модуля нажать ссылку **Дополнительный репозиторий**.
2. В окне настройки дополнительного репозитория нажать кнопку **Выберите файл**, указать путь к RPM-пакету и нажать **Загрузить**:



3. Повторить шаг 2 для всех пакетов, которые нужно добавить в репозиторий.
4. При необходимости включить публикацию (режим **Опубликовать как репозиторий для автоматических обновлений**) и выбрать публикуемые архитектуры.
5. Нажать кнопку **Обновить индексы**, чтобы перестроить метаданные репозитория.
6. Нажать кнопку **Применить**.



4.3. Настройка списка репозиториев APT

По окончании первой синхронизации репозиторий готов к использованию.

Непосредственно после установки дистрибутива ALT Platform Builder в файлах **`/etc/apt/sources.list.d/*.list`** указан интернет-репозиторий:

```
$ apt-repo
rpm [p11] http://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/x86_64
classic
rpm [p11] http://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/x86_64-
i586 classic
rpm [p11] http://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/noarch
classic
```

Для того чтобы указать локальный репозиторий вместо интернет-репозитория, необходимо выполнить следующие действия:

- удалить все текущие источники:

```
# apt-repo rm all
```

- добавить источник типа file для зеркалированного репозитория p11:

```
# apt-repo add file:/srv/public/mirror/p11/branch
```

- добавить источник типа file для дополнительного репозитория:

```
# apt-repo add file:/srv/public/mirror/additional
```

- вывести список доступных репозиториев:

```
# apt-repo
rpm file:/srv/public mirror/additional/x86_64 classic
rpm file:/srv/public mirror/additional/noarch classic
rpm file:/srv/public/mirror p11/branch/x86_64 classic
rpm file:/srv/public/mirror p11/branch/noarch classic
```

Примечание

В команде **apt-repo add** необходимо указать URL с обязательным указанием протокола и необязательным указанием архитектуры и одним или несколькими компонентами. Если при добавлении источника в конце строки отсутствуют архитектура и компонент, будут добавлены две строки (текущая архитектура системы и «noarch») с компонентом «classic».

Глава 5. Работа с пакетами

5.1. Пакетный менеджер RPM

Все пакеты в Альт Платформа собираются в формате RPM.

RPM (RPM Package Manager) — это семейство пакетных менеджеров, применяемых в различных дистрибутивах GNU/Linux, в том числе в проекте Sisyphus (Сизиф) и в дистрибутивах «Альт». Практически каждый крупный проект, использующий RPM, имеет свою реализацию пакетного менеджера, отличающуюся от других.

Между представителями семейства RPM могут существовать следующие различия:

- наборы макросов, используемых в spec-файлах;
- различное поведение при сборке «по умолчанию» — при отсутствии явных указаний в spec-файлах;
- формат строк зависимостей;
- отличия в семантике операций (например, при сравнении версий пакетов);
- отличия в формате файлов.

Для пользователя эти различия чаще всего проявляются в невозможности установить «неродной» пакет — из-за несовместимости формата или проблем с зависимостями.

RPM в проекте Сизиф также не является исключением. Основные отличия RPM в «Альт» и Сизифе от RPM в других проектах:

- обширный набор макросов для сборки различных типов пакетов;
- отличающееся поведение «по умолчанию», уменьшающее объём шаблонного кода в spec-файлах;

- наличие механизмов автоматического определения межпакетных зависимостей;
- поддержка так называемых set-version зависимостей (начиная с версии 4.0.4-alt98.46), обеспечивающих дополнительный контроль изменений ABI-библиотек;
- до p8 (включительно) — использование устаревшей версии «базового» RPM (4.0.4); в Sisyphus и p9 реализован частичный переход на rpm 4.13.

5.2. Утилита командной строки RPM

RPM — это низкоуровневая утилита командной строки, используемая для установки, удаления, обновления, выполнения запросов и проверки целостности пакетов.

Все остальные утилиты управления пакетами в конечном итоге используют RPM. При этом RPM не работает с репозиториями напрямую, а оперирует файлами пакетов и их зависимостями, используя собственную базу данных.

В дистрибутивах «Альт» база данных RPM хранится в каталоге **`/var/lib/rpm`**. В один момент времени к ней может обращаться только один процесс — остальные запросы блокируются.

Пакеты могут:

- предоставлять (provide) функциональность;
- требовать (require) зависимости;
- конфликтовать (conflict) с другими пакетами.

Таким образом формируется система межпакетных зависимостей (dependencies).

В дистрибутивах «Альт» пакет может быть установлен только при удовлетворении всех зависимостей и отсутствии конфликтов. Это означает, что не следует изменять системные файлы вручную или устанавливать ПО в обход пакетного менеджера.

Пакеты содержат метаинформацию, используемую утилитой **`rpm`**, а также файлы, каталоги и символические ссылки. Так называемые метапакеты не содержат файлов, а лишь описывают зависимости от других пакетов.

Основные режимы работы утилиты **`rpm`**:

- Install — установка пакетов;
- Remove — удаление пакетов;
- Upgrade: обновление пакетов;
- Query — выполнение запросов;
- Verify — проверка целостности пакетов.

Примечание

Справку по ключам команды **rpm** можно получить командой:

```
$ rpm --help
```

В качестве примера в данной главе используется пакет *Yodl-docs* (файл **yodl-docs-4.03.00-alt2.noarch.rpm**).

5.2.1. Вывод информации о пакете

Для вывода информации о пакете, который еще не установлен в систему, используется ключ **-qip** (Query|Install|Package):

```
$ rpm -qip package.rpm
```

где package.rpm — файл пакета.

Например:

```
$ rpm -qip yodl-docs-4.03.00-alt2.noarch.rpm
Name        : yodl-docs
Epoch       : 1
Version     : 4.03.00
Release     : alt2
DistTag     : sisyphus+348019.100.1.1
Architecture: noarch
Install Date: (not installed)
Group       : Documentation
Size        : 3716344
License     : GPL
Signature   : RSA/SHA512, Пн 13 мая 2024 20:26:54, Key ID ff979dedda2773bb
Source RPM  : yodl-4.03.00-alt2.src.rpm
Build Date  : Пн 13 мая 2024 20:26:47
Build Host  : iv-sisyphus.hasher.altlinux.org
Relocations : (not relocatable)
Packager    : Aleksei Nikiforov <darktemplar@altlinux.org>
Vendor      : ALT Linux Team
URL         : https://gitlab.com/fbb-git/yodl
Summary     : Documentation for Yodl
Description :
Yodl is a package that implements a pre-document language and tools to
process it.  The idea of Yodl is that you write up a document in a
pre-language, then use the tools (eg. yodl2html) to convert it to some
final document language.  Current converters are for HTML, ms, man, LaTeX
SGML and texinfo, plus a poor-man's text converter.  Main document types
are "article", "report", "book" and "manpage".  The Yodl document
language is designed to be easy to use and extensible.

This package contais documentation for Yodl.
```

Примечание

Ключ **-p** (Package) с файлом пакета, а не с базой данных RPM.

Для вывода информации об установленном пакете используется команда:

```
$ rpm -qi package
```

где package — установленный пакет.

Например:

```
$ rpm -qi bash
Name        : bash
Version     : 5.2.15
Release     : alt1
DistTag     : p11+348780.500.1.1
Architecture: noarch
Install Date: Пн 30 мар 2026 10:54:15
Group       : Shells
Size        : 0
License     : None
Signature   : RSA/SHA512, Пн 27 мая 2024 20:47:07, Key ID e1130f0e925e1ff4
Source RPM  : bash-defaults-5.2.15-alt1.src.rpm
Build Date  : Пн 27 мая 2024 20:47:05
Build Host  : arseny-p11.hasher.altlinux.org
Relocations : (not relocatable)
Packager    : Gleb Fotengauer-Malinovskiy <glebfm@altlinux.org>
Vendor      : ALT Linux Team
Summary     : The GNU Bourne Again SHell (/bin/bash)
Description :
This package provides default setup for the GNU Bourne Again SHell (/bin/bash).
```

5.2.2. Установка пакета из файла

Примечание

В команде должен быть указан файл пакета или полный путь к нему.

Для установки RPM-пакета используется ключ **-i** (Install):

```
# rpm -i package.rpm
```

где package.rpm — файл пакета.

Пример выполнения команды:

```
# rpm -i yodl-docs-4.03.00-alt2.noarch.rpm
```

В команде можно указать дополнительные ключи **-vh** (Verbose|Hash):

- **-v** — подробный вывод;

- **-h** — отображение прогресса символами «#».

Пример выполнения команды:

```
# rpm -ivh yodl-docs-4.03.00-alt2.noarch.rpm
Подготовка...                ################################# [100%]
Обновление / установка...
1: yodl-docs-1:4.03.00-alt2  ################################# [100%]
Running /usr/lib/rpm/posttrans-filetriggers
```

5.2.3. Обновление пакета

Для обновления пакета используется ключ **-U** (если пакет не установлен, он будет установлен):

```
$ rpm -Uvh yodl-docs-4.03.00-alt2.noarch.rpm
Подготовка...
############################################################ [100%]
    пакет yodl-docs-1:4.03.00-alt2.noarch уже установлен
```

5.2.4. Просмотр файлов пакета

Чтобы получить список файлов в пакете, который не установлен в систему, используются ключи **-qpl** (Query|Package|List):

```
$ rpm -qpl package.rpm
```

где package.rpm — файл пакета.

Например:

```
$ rpm -qpl yodl-docs-4.03.00-alt2.noarch.rpm
/usr/share/doc/yodl
/usr/share/doc/yodl-doc
/usr/share/doc/yodl-doc/AUTHORS.txt
/usr/share/doc/yodl-doc/CHANGES
/usr/share/doc/yodl-doc/changelog
/usr/share/doc/yodl-doc/yodl.dvi
/usr/share/doc/yodl-doc/yodl.html
/usr/share/doc/yodl-doc/yodl.latex
/usr/share/doc/yodl-doc/yodl.pdf
/usr/share/doc/yodl-doc/yodl.ps
/usr/share/doc/yodl-doc/yodl.txt
/usr/share/doc/yodl-doc/yodl01.html
/usr/share/doc/yodl-doc/yodl02.html
/usr/share/doc/yodl-doc/yodl03.html
/usr/share/doc/yodl-doc/yodl04.html
/usr/share/doc/yodl-doc/yodl05.html
/usr/share/doc/yodl-doc/yodl06.html
/usr/share/doc/yodl/AUTHORS.txt
/usr/share/doc/yodl/CHANGES
```

Для просмотра файлов пакета, установленного в систему, используется команда:

```
$ rpm -ql package
```

Например:

```
$ rpm -qpl yodl-docs
```

5.2.5. Поиск пакета в системе

Проверить, установлен ли пакет в системе можно, выполнив команду:

```
$ rpm -q пакет
```

Например:

```
$ rpm -q yodl-docs
```

Вывод:

```
yodl-docs-4.03.00-alt2.noarch
```

или:

```
пакет yodl-docs не установлен
```

Вывести список всех установленных пакетов:

```
$ rpm -qa
```

Поиск пакета через grep:

```
$ rpm -qa | grep yodl-docs
yodl-docs-4.03.00-alt2.noarch
```

5.2.6. Список недавно установленных пакетов

Для получения списка пакетов, которые были недавно установлены в систему, используется команда:

```
$ rpm -qa -last
```

Пример вывода:

```
apt-repo-1.5.1-alt1.noarch                    Пн 06 апр 2026 17:46:25
altlinux-repos-additional-1.1-alt1.noarch     Пн 06 апр 2026 17:46:25
alterator-users-10.31-alt1.x86_64             Пн 06 апр 2026 17:46:25
alterator-sslkey-0.2.6-alt1.noarch            Пн 06 апр 2026 17:46:25
alterator-mirror-allowed-0.7.2-alt1.x86_64    Пн 06 апр 2026 17:46:25
…
```

Чтобы список прокручивался, можно указать параметр ***less***:

```
# rpm -qa -last | less
```

5.2.7. Узнать пакет по файлу

Узнать к какому пакету относится файл можно, выполнив команду:

```
$ rpm -qf /путь/к/файлу
```

Например:

```
$ rpm -qf /usr/share/doc/yodl-docs
yodl-docs-4.03.00-alt2.noarch
```

5.2.8. Зависимости пакетов

Чтобы узнать от каких пакетов зависит указанный пакет, используется конструкция:

```
$ rpm -q --requires пакет
```

Например:

```
$ rpm -q --requires yodl-docs
rpmlib(PayloadIsLzma)
```

Узнать, какие установленные пакеты зависят от указанного можно, выполнив команду:

```
$ rpm -q --whatrequires пакет
```

Пример:

```
$ rpm -q --whatrequires yodl-docs
ни один из пакетов не требует yodl-docs
```

Узнать, какие зависимости предоставляет указанный пакет можно, выполнив команду:

```
$ rpm -q --whatprovides пакет
```

Пример:

```
$ rpm -q --whatprovides yodl-docs
yodl-docs-4.03.00-alt2.noarch
```

5.3. Система управления пакетами APT

Утилита RPM делает операции с отдельными пакетами атомарными (одношаговыми): вместо копирования множества файлов и запуска нескольких сценариев пользователь выполняет одну команду — «установить» или «удалить» пакет. Однако такая атомарная с точки зрения пользователя операция (например, добавление в систему одного нового компонента) может включать несколько, а иногда множество операций над пакетами. Чтобы сделать процедуры установки, удаления и обновления компонентов системы атомарными, были разработаны системы управления пакетами.

Используемая в «Альт» усовершенствованная система управления программными пакетами APT представляет собой удобный инструмент с простым пользовательским интерфейсом. Она позволяет выполнять установку, обновление и повседневные операции с программами без необходимости изучения тонкостей низкоуровневого менеджера пакетов.

APT использует две базы данных: одна содержит информацию об установленных в системе пакетах, вторая — о пакетах во внешних репозиториях. APT отслеживает целостность системы и при обнаружении конфликтов зависимостей использует данные из внешних репозиториев для их разрешения.

Система APT состоит из нескольких утилит. Чаще всего используется утилита управления пакетами **apt-get**, которая автоматически определяет зависимости между пакетами и строго контролирует их соблюдение при выполнении операций установки, удаления или обновления пакетов.

APT хранит кеш загруженных пакетов в каталоге **/var/cache/apt/archives**, а кеш индексов в — **/var/lib/apt/lists**. Конфигурация APT расположена в каталоге **/etc/apt**: основной конфигурационный файл — **/etc/apt/apt.conf**, а списки подключенных репозиториев — в **/etc/apt/sources.list** и **/etc/apt/sources.list.d/***.

Посмотреть текущую конфигурацию APT можно командой:

```
$ apt-config dump
```

5.3.1. Репозитории

Репозитории, с которыми работает APT, отличаются от простого набора пакетов наличием метаинформации — индексов пакетов, содержащихся в репозитории, и сведений о них. Поэтому для получения полной информации о репозитории APT достаточно загрузить его индексы.

APT может работать одновременно с несколькими репозиториями, формируя единую базу данных обо всех доступных пакетах. При установке пакетов учитываются их имя, версия и зависимости, а конкретное расположение (в каком репозитории они находятся) значения не имеет. В рамках одной операции APT может использовать сразу несколько репозиториев.

Примечание

При использовании нескольких репозиториев необходимо следить за их совместимостью (они должны относиться к одной ветке или этапу разработки). Например, совместимы основной репозиторий дистрибутива и репозиторий обновлений безопасности. В то же время смешение стабильного репозитория и нестабильной ветки разработки (Sisyphus), либо репозиториев разных дистрибутивов, может привести к проблемам при обновлении системы.

APT поддерживает работу с репозиториями по различным протоколам. Наиболее распространённые — HTTP и FTP, но доступны и другие методы.

Для того чтобы APT мог использовать тот или иной репозиторий, информацию о нём необходимо поместить в файл **/etc/apt/sources.list**, либо в любой файл с расширением ***.list** (например, **mysources.list**) в каталоге **/etc/apt/sources.list.d**. Формат записи:

```
rpm [подпись] метод: путь база название
rpm-src [подпись] метод: путь база название
```

где:

- rpm или rpm-src — тип репозитория (скомпилированные программы или исходные тексты);

- [подпись] — необязательная строка-указатель на электронную подпись разработчиков. Наличие этого поля подразумевает, что каждый пакет из данного репозитория должен быть подписан соответствующей электронной подписью. Подписи описываются в файле **/etc/apt/vendor.list**;
- метод — способ доступа к репозиторию (ftp, http, file, cdrom, copy);
- путь — путь к репозиторию в терминах выбранного метода;
- база — относительный путь к базе данных репозитория;
- название — название репозитория.

Непосредственно после установки ОС «Альт» в файлах **/etc/apt/sources.list.d/*.list** обычно указывается интернет-репозиторий, соответствующий установленному дистрибутиву.

После изменения списка репозиториев необходимо обновить локальную базу данных пакетов APT. Это делается командой **apt-get update**.

Если в **sources.list** присутствует репозиторий, содержимое которого может изменяться (как происходит с любым постоянно разрабатываемым репозиторием, в частности, обновлений по безопасности), то прежде чем работать с APT, необходимо синхронизировать локальную базу данных с удаленным сервером командой **apt-get update**. Локальная база данных создается заново каждый раз, когда в репозитории происходит изменение: добавление, удаление или переименование пакета.

Для репозиториев на съёмных носителях (например, CD/DVD), добавленных с помощью **apt-cdrom add**, синхронизация выполняется однократно — в момент подключения.

При выборе пакетов APT учитывает все подключённые источники. Если в одном из них доступна более новая версия пакета (например, в интернет-репозитории по сравнению с локальным носителем), будет использована именно она.

Поэтому при отсутствии или ограничении доступа в Интернет (низкая скорость или высокая стоимость трафика) рекомендуется закомментировать строки в **/etc/apt/sources.list**, указывающие на удалённые репозитории.

5.3.1.1. Утилита apt-repo для работы с репозиториями

Для редактирования репозиториев можно воспользоваться утилитой **apt-repo**:

Основные команды:

- просмотреть список активных репозиториев:

```
$ apt-repo
```

- показать все доступные репозитории (неактивные будут закомментированы символом «#»):

```
$ apt-repo -a
```

- добавить репозиторий в список активных:

```
# apt-repo add репозиторий
```

- удалить или отключить репозиторий:

```
# apt-repo rm репозиторий
```

- удалить все источники и добавить новый репозиторий:

```
# apt-repo set репозиторий
```

- удалить все источники типа cdrom и все хранилища задач (task):

```
# apt-repo clean
```

- обновить информацию о репозиториях (выполнить **apt-get update**):

```
# apt-repo update
```

- вывести справку:

```
$ man apt-repo
```

или

```
$ apt-repo --help
```

Примечание

Для выполнения большинства команд требуются права администратора.

Типичный пример использования команды **apt-repo**: удалить все источники и добавить стандартный репозиторий P11 (архитектура выбирается автоматически):

```
# apt-repo rm all
# apt-repo add p11
```

Или то же самое одной командой:

```
# apt-repo set p11
```

Источник может быть указан в формате sources.list(5):

```
# apt-repo add "rpm http://git.altlinux.org/repo/414014/ x86_64 task"
```

Поддерживаются следующие типы репозиториев: rpm, rpm-dir и rpm-src. APT работает с протоколами: file://, copy://, http://, ftp://, rsync:// и cdrom://.

URL с обязательным указанием протокола может содержать также архитектуру и один или несколько компонентов. Если архитектура и компонент не указаны, автоматически добавляются записи для текущей архитектуры системы и noarch с компонентом classic.

Пример добавления локального репозитория:

```
# apt-repo add file:/srv/public/mirror/p11/branch
```

5.3.1.2. Добавление репозитория на CD/DVD-носителе

Для добавления в репозитория с компакт-диска в APT используется утилита **apt-cdrom**.

Чтобы добавить запись о репозитории на сменном диске необходимо:

1. Создать каталог для монтирования. Точка монтирования указывается в параметре **Acquire::CDROM::mount** в файле конфигурации **APT** (**/etc/apt/apt.conf**), по умолчанию это **/media/ALTLinux**:

   ```
   # mkdir /media/ALTLinux
   ```

2. Примонтировать носитель в указанную точку:

   ```
   # mount /dev/sdXN /media/ALTLinux
   ```

   где **/dev/sd*XN*** — соответствующее блочное устройство (например, **/dev/dvd** — для CD/DVD-диска).

3. Добавить носитель, выполнив команду:

   ```
   # apt-cdrom -m add
   ```

После этого в **sources.list** появится запись о подключённом диске.

Примечание

Команду **mount /dev/носитель /media/ALTLinux** необходимо выполнять перед каждой командой **apt-get install имя_пакета**.

5.3.1.3. Добавление репозиториев вручную

Для изменения списка репозиториев можно отредактировать в любом текстовом редакторе файлы из каталога **/etc/apt/sources.list.d/**.

Примечание

Для изменения этих файлов необходимы права администратора.

В файле **alt.list** может содержаться такая информация:

```
# ftp.altlinux.org (ALT Linux, Moscow)

# ALT Platform 11
#rpm [p11] ftp://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/x86_64
classic
#rpm [p11] ftp://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/x86_64-
i586 classic
#rpm [p11] ftp://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/noarch
classic

rpm [p11] http://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/x86_64
```

```
classic
rpm [p11] http://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/x86_64-
i586 classic
rpm [p11] http://ftp.altlinux.org/pub/distributions/ALTLinux p11/branch/noarch
classic

#rpm [p11] rsync://ftp.altlinux.org/ALTLinux p11/branch/x86_64 classic
#rpm [p11] rsync://ftp.altlinux.org/ALTLinux p11/branch/x86_64-i586 classic
#rpm [p11] rsync://ftp.altlinux.org/ALTLinux p11/branch/noarch classic
```

По сути, каждая строчка соответствует некому репозиторию. Неактивные репозитории — строки, начинающиеся с символа «#».

Для отключения репозитория достаточно закомментировать соответствующую строку. Для добавления нового репозитория необходимо дописать его в этот или другой файл.

После изменения списка репозиториев необходимо обновить информацию о пакетах:

```
# apt-get update
```

или:

```
apt-repo update
```

5.3.2. Поиск пакетов

Если точное название пакета неизвестно, для его поиска можно воспользоваться утилитой **apt-cache**, которая позволяет искать не только по имени пакета, но и по его описанию.

Команда **apt-cache search** позволяет найти все пакеты, в именах или описаниях которых присутствует указанная подстрока. Например:

```
$ apt-cache search ^stardict
stardict-wn - GCIDE - The Collaborative International Dictionary of English
stardict-mueller7 - V.K. Mueller English-Russian Dictionary, 7 Edition: stardict
format
stardict-slovnyk_be-en - Dictionary: Slovnyk Belarusian-English
stardict-slovnyk_be-ru - Dictionary: Slovnyk Belarusian-Russian
stardict-slovnyk_be-uk - Dictionary: Slovnyk Belarusian-Ukrainian
stardict-slovnyk_cs-ru - Dictionary: Slovnyk Czech-Russian
stardict-slovnyk_en-be - Dictionary: Slovnyk English-Belarusian
stardict-slovnyk_en-ru - Dictionary: Slovnyk English-Russian
stardict-slovnyk_en-uk - Dictionary: Slovnyk English-Ukrainian
…
```

Следует обратить внимание, что в данном примере в поисковом выражении используется символ «^», указывающий на то, что необходимо найти совпадения только в начале строки (в данном случае — в начале имени пакета).

Чтобы подробнее узнать о каждом из найденных пакетов и прочитать его описание, можно воспользоваться командой **apt-cache show**, которая выводит информацию о пакете из репозитория:

```
$ apt-cache show stardict-mueller7
Package: stardict-mueller7
Section: Text tools
Installed Size: 3094848
Maintainer: Anton V. Boyarshinov <boyarsh@altlinux.ru>
Version: 1.0-alt8@1338342590
Pre-Depends: rpmlib(PayloadIsLzma)
Depends: stardict (>= 2.4.2)
Provides: stardict-mueller7 (= 1.0-alt8)
Architecture: noarch
Size: 3134862
MD5Sum: 54f9e085c1fc67084253b3ba72a0c482
Filename: stardict-mueller7-1.0-alt8.noarch.rpm
Description: V.K. Mueller English-Russian Dictionary, 7th Edition, for stardict
 Electronic version of V.K. Mueller English-Russian Dictionary,
 7th Edition, in stardict format, for use with a stardict client.
```

При поиске с помощью **apt-cache** можно использовать русскую подстроку. В этом случае будут найдены пакеты, имеющие описание на русском языке.

5.3.3. Установка или обновление пакета

Примечание

Для установки пакетов требуются привилегии администратора.

Установка пакета с помощью APT выполняется командой:

```
# apt-get install <имя_пакета>
```

Важно

Перед установкой и обновлением пакетов необходимо выполнить команду обновления индексов:

```
# apt-get update
```

Утилита **apt-get** позволяет устанавливать в систему пакеты, которые требуют для работы другие, ещё не установленные пакеты. В этом случае она автоматически определяет необходимые зависимости и устанавливает их, используя все доступные репозитории.

Установка пакета *stardict-mueller7* командой **apt-get install stardict-mueller7** приведет к следующему диалогу с APT:

```
# apt-get install stardict-mueller7
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие дополнительные пакеты будут установлены:
  icon-theme-hicolor  libgtk+2  libgtk+2-locales  stardict
Следующие НОВЫЕ пакеты будут установлены:
  icon-theme-hicolor  libgtk+2  libgtk+2-locales  stardict  stardict-mueller7
0 будет обновлено, 5 новых установлено, 0 пакетов будет удалено и 25 не будет
```

```
обновлено.
Необходимо получить 9691kB архивов.
После распаковки потребуется дополнительно 36,2MB дискового пространства.
Продолжить? [Y/n] y
Получено: 1 https://ftp.altlinux.org p11/branch/noarch/classic icon-theme-hicolor
0.18-alt1:p11+349758.500.2.1@1717139306 [33,2kB]
Получено: 2 https://ftp.altlinux.org p11/branch/noarch/classic libgtk+2-locales
2.24.33-alt2:p11+363895.5700.2.1@1733170978 [2569kB]
Получено: 3 https://ftp.altlinux.org p11/branch/x86_64/classic libgtk+2 2.24.33-
alt2:p11+363895.5700.2.1@1733170978 [1975kB]
Получено: 4 https://ftp.altlinux.org p11/branch/x86_64/classic stardict 3.0.6-
alt2:p11+403210.100.1.1@1766080058 [1979kB]
Получено: 5 https://ftp.altlinux.org p11/branch/noarch/classic stardict-mueller7
1.0-alt8@1338342590 [3135kB]
Получено 9691kB за 0s (17,4MB/s).
Совершаем изменения...
Подготовка...                          ################################## [100%]
Обновление / установка...
1: libgtk+2-locales-2.24.33-alt2       ################################## [ 20%]
2: icon-theme-hicolor-0.18-alt1        ################################## [ 40%]
3: libgtk+2-2.24.33-alt2               ################################## [ 60%]
4: stardict-3.0.6-alt2                 ################################## [ 80%]
5: stardict-mueller7-1.0-alt8          ################################## [100%]
Завершено.
```

Команда **apt-get install имя_пакета** также используется для обновления уже установленного пакета или группы пакетов. В этом случае **apt-get** дополнительно проверяет, не появилась ли более новая версия пакета в репозитории.

Если пакет *stardict-mueller7* установлен и в репозитории нет обновлённой версии этого пакета, то вывод команды **apt-get install stardict-mueller7** будет таким:

```
# apt-get install stardict-mueller7
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Последняя версия stardict-mueller7 уже установлена.
0 будет обновлено, 0 новых установлено, 0 пакетов будет удалено и 25 не будет
обновлено.
```

С помощью **APT** можно установить и отдельный RPM-пакет, не входящий в состав репозиториев (например, загруженный из Интернета). Для этого достаточно выполнить команду

```
#  apt-get install /путь/к/файлу.rpm
```

При этом **APT** выполнит стандартную проверку зависимостей и конфликтов с уже установленными пакетами.

Иногда в результате операций с пакетами без использования APT целостность системы может нарушиться, и **apt-get** откажется выполнять операции установки, удаления или обновления. В этом случае необходимо повторить операцию с опцией **-f**, которая заставляет **apt-get** исправить зависимости, удалить или заменить конфликтующие пакеты.

При использовании этого режима необходимо внимательно следить за сообщениями **apt-get**, так как любые действия требуют подтверждения пользователя.

При установке пакетов происходит запись в системный журнал вида:

```
apt-get: имя-пакета installed
```

5.3.4. Удаление установленного пакета

Примечание

Для удаления пакетов требуются привилегии администратора.

Для удаления пакета используется команда:

```
# apt-get remove <имя_пакета>
```

Чтобы не нарушать целостность системы, будут удалены также все пакеты, зависящие от удаляемого.

При попытке удалить пакет, относящийся к базовым компонентам системы, **apt-get** потребует дополнительного подтверждения операции, чтобы предотвратить возможную ошибку. Пример предупреждения:

```
# apt-get remove filesystem
Обработка файловых зависимостей... Завершено
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие пакеты будут УДАЛЕНЫ:
…
ВНИМАНИЕ: Будут удалены важные для работы системы пакеты
Обычно этого делать не следует. Вы должны точно понимать возможные последствия!
…
0 будет обновлено, 0 новых установлено, 853 пакетов будет удалено и 4 не будет
обновлено.
Необходимо получить 0B архивов.
После распаковки будет освобождено 2211MB дискового пространства.
Вы делаете нечто потенциально опасное!
Введите фразу 'Yes, do as I say!' чтобы продолжить..
```

Каждую ситуацию, в которой APT выдаёт подобное сообщение, необходимо анализировать отдельно. Вероятность того, что система станет неработоспособной, в таких случаях очень высока.

5.3.5. Обновление всех установленных пакетов

Для обновления всех установленных пакетов необходимо выполнить команды:

```
# apt-get update
# apt-get dist-upgrade
```

Первая команда (**apt-get update**) обновит индексы пакетов. Вторая команда (**apt-get dist-upgrade**) позволяет обновить только те установленные пакеты, для которых в репозиториях, перечисленных в **/etc/apt/sources.list**, имеются новые версии.

В случае обновления всего дистрибутива APT проведет сравнение системы с репозиторием и удалит устаревшие пакеты, установит новые версии присутствующих в системе пакетов, а также отследит ситуации с переименованиями пакетов или изменения зависимостей между старыми и новыми версиями программ. Все, что потребуется поставить (или удалить) дополнительно к уже имеющемуся в системе, будет указано в отчете **apt-get**, которым APT предварит само обновление.

Примечание

Команда **apt-get dist-upgrade** не обновляет ядро ОС.

5.3.6. Обновление ядра

APT в дистрибутивах «Альт» по умолчанию не обновляет ядро вместе с системой (см. настройки hold в **apt.conf**), поскольку обновление такого критичного компонента системы может привести к нежелательным последствиям. Вместо этого в систему могут быть поставлены пакеты нескольких ядер и модулей к разным ядрам одновременно.

Для обновления ядра ОС необходимо выполнить команду:

```
# update-kernel
```

Примечание

Если индексы пакетов ещё не обновлялись, перед выполнением команды следует выполнить:

```
# apt-get update
```

Команда **update-kernel** обновляет также модули ядра, если они были изменены.

Новое ядро будет использоваться только после перезагрузки системы.

Если с новым ядром возникнут проблемы, можно выбрать предыдущую версию в меню загрузчика.

После успешной загрузки можно удалить старые версии ядра:

```
# remove-old-kernels
```

Глава 6. Основы сборки RPM-пакетов

6.1. RPM-пакет

RPM-пакет состоит из архива в формате cpio, содержащего файлы (скомпилированные исполняемые файлы, библиотеки, данные), и заголовка с метаданными (название, версия, группа и т.п.). Эти метаданные используются системой управления пакетами (например, apt) для разрешения зависимостей, определения путей установки и получения другой служебной информации.

Различают два вида RPM-пакетов:

- пакет с исходным кодом — SRPM-пакет (имеет расширение вида **`.src.rpm`**). Такой пакет содержит архив (один или несколько) с исходным кодом, spec-файл и, возможно, различные патчи и дополнения. Пакет src.rpm используется только для сборки двоичных пакетов, но не для установки. Сборка осуществляется командой:

```
$ rpmbuild --rebuild package.src.rpm
```

- собранный двоичный пакет — RPM-пакет (имеет расширение вида **`<архитектура>.rpm`**). Такие пакеты можно устанавливать командой:

```
# rpm -Uvh package.rpm
```

Однако при ручной сборке через **`rpmbuild`** возникают очевидные сложности:

- необходимо вручную удовлетворять сборочные зависимости (устанавливать компилятор, заголовочные файлы, библиотеки). При большом количестве собираемых пакетов система засоряется;
- для сборки пакета необходимо сформировать **`.src.rpm`** из файлов, расположенных в разных каталогах (по умолчанию это подкаталоги **`SOURCE`**, **`SPECS`** и каталоги сборки в **`~/RPM`**);
- исходные файлы должны быть упакованы, что затрудняет создание патчей;
- на рабочей системе легко пропустить необходимые зависимости.

Для решения этих проблем в Альт Платформа используется две технологии:

- **Hasher** — сборка в изолированном chroot-окружении. В chroot устанавливается базовый набор пакетов и пакеты, необходимые для сборки (поле ***`BuildRequires`*** в spec-файле). Если какой-либо пакет не указан в spec-файле, сборка завершится ошибкой. Это обеспечивает воспроизводимость и чистоту сборки. Обратной стороной является необходимость доступа к репозиторию, так как пакеты устанавливаются при каждой сборке;
- **Gear** — работа с распакованными исходниками в git с автоматической упаковкой в **`.src.rpm`**. В этом случае все файлы хранятся в распакованном виде и упаковываются в **`.src.rpm`** по правилам, определённым в **`.gear/rules`**. Это позволяет напрямую работать с содержимым, быстро создавать патчи, вести историю изменений и эффективно взаимодействовать в рамках командной разработки.

6.2. SPEC-файл

Spec-файл можно рассматривать как «инструкцию», которую утилита **`rpmbuild`** использует для сборки RPM-пакета. Он определяет все действия, выполняемые при сборке, а также операции, необходимые при установке и удалении пакета. Каждый .src.rpm-пакет содержит spec-файл.

Spec-файл представляет собой текстовый файл. Согласно соглашению об именовании, он должен называться по шаблону: **`имя_пакета.spec`**.

Spec-файл сообщает системе сборки, что необходимо выполнить, задавая инструкции в виде набора разделов. Эти разделы условно делятся на две части:

- **Hasher** — содержит метаданные пакета;
- **Hasher** — содержит инструкции по сборке и установке.

Преамбула содержит элементы метаданных, которые используются в основной части Основная часть включает команды и сценарии, выполняемые на различных этапах сборки.

Текст внутри spec-файла имеет специальный синтаксис. Его конструкции определяют порядок сборки, версию пакета, зависимости и другую информацию, которая впоследствии может быть получена из базы данных RPM.

6.2.1. Пример spec-файла

Пример spec-файла:

```
Name: sampleprog
Version: 1.0
Release: alt1

Summary: Sample program specfile
Summary(ru_RU.UTF-8): Пример spec-файла для программы

License: GPLv2+
Group: Development/Other
Url: http://www.altlinux.org/SampleSpecs/program

Packager: Sample Packager <sample@altlinux.org>

Source: %name-%version.tar
Patch0: %name-1.0-alt-makefile-fixes.patch

%description
This specfile is provided as sample specfile for packages with programs.
It contains most of usual tags and constructions used in such specfiles.

%description -l ru_RU.UTF-8
Этот spec-файл является примером spec-файла для пакета с программой. Он содержит
основные теги и конструкции, используемые в подобных spec-файлах.

%prep
%setup
%patch0 -p1

%build
%configure
%make_build

%install
%makeinstall_std
%find_lang %name

%files -f %name.lang
```

```
%doc AUTHORS ChangeLog NEWS README THANKS TODO contrib/ manual/
%_bindir/*
%_man1dir/*

%changelog
* Sat Sep 12 3001 Sample Packager <sample@altlinux.org> 1.0-alt1
- initial build
```

6.2.2. Директивы преамбулы

В Директивы преамбулы spec-файла перечислены директивы, используемые в разделе преамбулы файла спецификации RPM.

Таблица 6.1. Директивы преамбулы spec-файла

SPEC-директива	Определение
Name	Базовое имя пакета (должно совпадать с именем spec-файла)
Version	Версия upstream-кода
Release	Релиз пакета используется для указания номера сборки при данной версии upstream-кода. Для пакетов Sisyphus поле Release должно иметь вид: - в простых случаях — altN; - в сложных — altN[суффикс]. Значение N начинается с 1 для каждой новой upstream-версии и увеличивается на 1 для каждой новой сборки: - 1.0-alt1 - 1.0-alt2
Epoch	Используется, если необходимо уменьшить версию или релиз пакета по сравнению с имеющимся в репозитории. В этом случае значение Epoch увеличивается на единицу относительно предыдущего (отсутствие поля эквивалентно значению 0), а версия и релиз устанавливаются в требуемые значени ***Epoch*** не входит в имя RPM-файла, поэтому следует избегать пакетов с одинаковыми ***Version*** и ***Release***, но разными ***Epoch***.
Summary	Краткое однострочное описание пакета. Должно начинаться с заглавной буквы и не заканчиваться точкой
License	Лицензия программного обеспечения. Должна указываться в точности так, как сформулировано в upstream-пакете. При указании лицензии рекомендуется пользоваться макросами из пакета *rpm-build-licenses*, добавив его в список ***BuildRequires***. Текст лицензии включается в пакет только если он отсутствует в **/usr/share/license** (пакет *common-licenses*). В противном случае достаточно указать название лицензии
Group	

SPEC-директива	Определение
	Категория пакета. Должна соответствовать списку групп RPM (файл **/usr/lib/rpm/GROUPS** из пакета *rpm*)
URL	Полный URL-адрес для получения дополнительной информации о программе. Рекомендуется указывать действующий URL домашней страницы проекта или другой источник исходного кода. Для директивы URL можно использовать утилиту **rpmurl**. Например, проверка доступности URL: `$ rpmurl -c пакет.spec`
Source0	Путь или URL к архиву исходного кода (без учёта патчей). Должен указывать на надёжный внешний источник (например, upstream), а не на локальное хранилище. Дополнительные источники задаются как ***Source1***, ***Source2*** и т. д.
Patch0	Название первого патча, применяемого к исходному коду. Дополнительные патчи задаются как ***Patch1***, ***Patch2*** и т. д.
BuildArch	Явное указание архитектуры, под которую собирается двоичный пакет. Если не задано, используется архитектура системы сборки (например, x86_64). Для архитектурно-независимых пакетов указывается noarch.
BuildRequires, ***Build PreReq***, ***BuildRequires(pre)***	Список пакетов, необходимых для сборки (разделяется запятыми или пробелами). Может быть несколько записей ***BuildRequires***, каждая в отдельной строке. ***BuildRequires*** обычно формируется автоматически (например, с помощью **buildreq**). Дополнительные зависимости рекомендуется указывать в ***BuildPreReq***
Requires, ***PreReq***	Список пакетов, необходимых для работы программы после установки (разделяется запятыми или пробелами). Может быть несколько записей ***Requires***, каждая в отдельной строке.
ExcludeArch	Исключает архитектуры, на которых пакет не может быть собран или работать
Conflicts	Список пакетов, конфликтующих с данным. Используется при наличии файловых, RPC- или логических конфликтов
Provides	Указывает, какую функциональность предоставляет пакет (например, альтернативное или устаревшее имя). Следует применять только в случае реальной необходимости и, как правило, в форме `Provides: something = %version-%release`

SPEC-директива	Определение
	При переименовании пакета используется совместно с ***Obsoletes***
Obsoletes	Перечисляет устаревшие пакеты/версии. Обычно применяется при переименовании и используется вместе с ***Provides***

Каждая директива разделяется от значения символом «:». Между именем директивы и двоеточием пробелы не допускаются.

Директивы ***Name***, ***Version*** и ***Release*** формируют имя RPM-пакета. Их часто называют N-V-R или NVR, поскольку имя пакета имеет формат: NAME-VERSION-RELEASE.

Увидеть пример NAME-VERSION-RELEASE можно, выполнив запрос с использованием **rpm** для конкретного пакета:

```
$ rpm -q rpmspec
rpmspec-4.13.0.1-alt40.x86_64
```

Здесь:

- rpmspec — имя пакета;
- 4.13.0.1 — версия
- alt40 — релиз
- x86_64 — сведения об архитектуре.

В отличие от NVR, архитектура не управляется напрямую сборщиком и определяется средой сборки. Исключение составляют архитектурно-независимые пакеты (noarch).

6.2.3. Директивы основной части

В Директивы основной части spec-файла перечислены директивы, используемые в основной части spec-файла. Все они, кроме ***%check***, являются обязательными.

Таблица 6.2. Директивы основной части spec-файла

SPEC-директива	Определение
%description	Полное описание программного обеспечения, входящего в состав RPM-пакета. Описание может занимать несколько строк и разбиваться на абзацы. Длина каждой строки не должна превышать 72 символа. Данное описание используется при поиске пакета через **apt-cache search** и полностью выводится при просмотре информации о пакете с помощью **apt-cache show имя_пакета**
%prep	Команда или последовательность команд для подготовки исходного кода к сборке (например, распаковка архива, указанного в Source0). Может содержать сценарий оболочки (shell скрипт).
%build	

SPEC-директива	Определение
	Команда или последовательность команд для последовательность сборки программного обеспечения в машинный код (для компилируемых языков) или байт-код (для некоторых интерпретируемых языков)
%*install*	Команды установки/копирования файлов из каталога сборки в псевдокорневой каталог. Этот раздел эмулирует установку файлов в конечную систему. Здесь происходит копирование артефактов сборки из **%*builddir*** (каталога сборки) в **%*buildroot*** (каталог, содержащий структуру файлов будущего пакета)
%*check*	Команда или последовательность команд для тестирования программного обеспечения. Обычно включает запуск модульных тестов
%*files*	Список файлов, которые будут установлены в системе конечного пользователя
%*changelog*	Журнал изменений пакета между версиями и релизами

Вне зависимости от количества двоичных пакетов, описанных в spec-файле, все они используют общие директивы **%*prep***, **%*build*** и **%*install*** (разделение на разные двоичные RPM происходит в блоках **%*files***).

6.3. RPM Макросы

Макросы RPM — это прямые текстовые подстановки, которые выполняются путем замены определенных выражений и условий на соответствующий текст во время процесса сборки пакета. Иными словами, макросы являются псевдонимами для часто используемых фрагментов текста. Их применение сокращает не только объем ввода, но и делает спецификацию легче читаемой и воспринимаемой. Имена макросов начинаются с символа «%».

Список пакетов, содержащих макросы можно получить, выполнив команду:

```
$ apt-cache search rpm | grep '^rpm-[bm]' | sort -n
rpm-build-apache2 - Набор утилит для автоматической Web серверов и приложений
rpm-build-browser-plugins - Netscape Gecko Plug-in API common packaging files
rpm-build-compat - ALT Linux compatibility macros for backport purposes
rpm-build-dmd - RPM build environment to build D lang(dmd) packages
rpm-build-emacs - Helper scripts and RPM macros to build GNU Emacs extensions
rpm-build-erlang - RPM helper scripts to calculate Erlang dependencies
rpm-build-extra-targets - Build packages for other platforms
rpm-build-fedora-compat-fonts - Build-stage rpm automation for fonts packages
rpm-build-file - Утилита для определения типов файлов
rpm-build-firefox - RPM helper macros to rebuild firefox packages
rpm-build-fonts - RPM helper scripts for building font packages
…
```

Для использования данных макросов необходимо добавить в spec-файл строку:

```
BuildRequires(pre): имя-пакета-с-макросами
```

Например:

```
BuildRequires(pre): rpm-build-java
```

Списки макросов находятся в каталоге **/usr/lib/rpm/macros.d/**.

Получить список доступных макросов и их значения можно, выполнив команду:

```
$ rpm --showrc
```

Получить значение, раскрываемое макросом, можно командой:

```
$ rpm --eval <имя_макроса>
```

Например:

```
$ rpm --eval %_sysconfdir
/etc
```

Макросы можно использовать внутри других макросов. Например, если имя архива исходных текстов формируется из имени и версии проекта (директивы ***Name*** и ***Version*** транслируются в соответствующие макросы), то директива задания пути к файлу может выглядеть следующим образом:

```
Source0: %{name}-%{version}.tar.gz
```

Примечание

Не следует использовать в spec-файлах внутренние макросы RPM, которые начинаются с двух подчеркиваний (например, %__install или %__mkdir_p).

В Макросы путей системных каталогов перечислены некоторые макросы.

Таблица 6.3. Макросы путей системных каталогов

Макрос	Описание
%homedir	Домашний каталог пользователя, вызывающего этот макрос
%_licensedir	Каталог лицензий (**/usr/share/license**)
%_controldir	Каталог control (**/etc/control.d/facilities**)
%_defaultdocdir	Каталог документации (**/usr/share/doc**)
%_defattr	Атрибуты файлов и каталогов по умолчанию для каждой секции %files и для каждого файла, включаемого в таких секциях (-,root,root,755)
%_man1dir, %_man2dir, %_man3dir, %_man4dir, %_man5dir, %_man6dir, %_man7dir, %_man8dir, %_man9dir	Каталог man-файлов (**/usr/share/man/manX**)
%java_dir, %_javadir	Каталог для некоторых jar-файлов (**/usr/share/java**)
%_rpmmacrosdir	Каталог для установки сторонних макросов (**/usr/lib/rpm/macros.d**)

С целью сокращения объёма кода и упрощения записи часто используемых команд при сборке пакета в блоках тела spec-файла существует ряд встроенных макросов. Некоторые из них рассмотрены ниже.

Макрос **%*setup*** — используется в блоке **%*prep***. Этот макрос распаковывает архив с исходным кодом (ключ **-q** подавляет подробный вывод при распаковке архива). По умолчанию распаковывается первый архив с исходным кодом (Source0); для остальных необходимо дополнительный параметр **-a X**, где X — номер Source.

```
%prep
%setup -a1 -a100 -a101 -a102 -a103 -a104 -a105
%patch1 -p1
```

В Sisyphus RPM макрос **%*setup*** использует ключ **-q** по умолчанию. Записи **%*setup*** и **%*setup -q*** эквивалентны. Для включения подробного вывода следует использовать ключ **-v**.

%*patch[X]* — используется в блоке **%*prep*** и применяет указанный патч (X — номер патча, такой же, как и при их описании в преамбуле, если патчей несколько). Патчи применяются относительно текущего каталога сборки. При необходимости можно обрезать часть пути с помощью ключа **–p** (как и у самой утилиты **patch**). Например:

```
%patch3 -p1
```

Макрос **%*autopatch*** позволяет применить все патчи, описанные в основной секции spec-файла, в порядке возрастания их номеров. Макрос поддерживает ключи **-p** и **-F**, аналогичные таким же опциям директивы **%*patch***.

Макрос **%*configure*** — используется в блоке **%*build*** и используется для упрощения запуска **./configure** с параметрами, соответствующими текущей платформе. В большинстве случаев достаточно вызвать **%*configure*** без параметров.

```
%build
%configure
%make_build
```

Если в исходниках отсутствует скрипт **configure** (например, при сборке из git), но есть **configure.ac**, перед вызовом **%*configure*** следует выполнить макрос **%*autoreconf***.

Макрос **%*make_build*** — используется в блоке **%*build*** для запуска **make** с поддержкой параллельной сборки.

Макрос **%*makeinstall_std*** — применяется в блоке **%*install*** и рекомендуется вместо **%*make_install*** с ключами install и DESTDIR=%buildroot:

```
%make_install DESTDIR=%buildroot install
```

Макрос **%*make_install*** — применяется в блоке **%*install***. Используется для упрощения установки ПО. Вызывает **make install** с указанием каталога для установки (***DESTDIR=%buildroot***) и рядом других ключей:

```
%make_install DESTDIR=%buildroot install
```

или

```
%make_install DESTDIR=%buildroot %_make_install_target
```

Макрос **%makeinstall** — применяется в блоке **%install**. Это редко используемый макрос, предназначенный для случаев, когда **Makefile** не поддерживает **DESTDIR**.

Макрос **%dir** — используется в блоке **%files** и указывает, что путь является каталогом, который должен принадлежать этому RPM. Это указание важно для того, чтобы RPM точно знал, какие каталоги нужно очистить при удалении пакета.

Макрос **%doc** — используется в блоке **%files** для создания каталога документации (**%_defaultdocdir/%name-%version**) и копирования в него указанных файлов. Пути к файлам строятся относительно каталога сборки проекта, например:

```
%doc Examples
```

Глава 7. Инструмент GEAR

Gear (Get Every Archive from git package Repository) — система для работы с произвольными архивами программ. В качестве хранилища данных gear использует git, что позволяет работать с полной историей проекта.

Gear поддерживает полный цикл организации репозиториев:

- создание репозитория или импорт существующих src.rpm-пакетов;
- обновление upstream-кода в репозиториях;
- наложение патчей и пакетирование;
- экспорт pkg.tar и src.rpm, сборка бинарных RPM-пакетов.

Основной смысл хранения исходного кода пакетов в git-репозитории заключается в более эффективной и удобной совместной разработке, а также в минимизации объёма дискового пространства, используемого для хранения архива репозитория за длительный период, и снижении трафика при обновлении исходного кода.

Идея gear заключается в том, чтобы с помощью одного файла с простыми правилами (для обработки которых достаточно **sed** и **git**) можно было собирать пакеты из произвольно устроенного git-репозитория, по аналогии с hasher, который был задуман как средство для сборки пакетов из произвольных SRPM-пакетов.

7.1. Структура репозитория

Хотя gear не накладывает ограничений на внутреннюю организацию git-репозитория (за исключением требования наличия файла с правилами), существуют рекомендации, позволяющие сделать работу более эффективной и удобной.

Одна сущность—один репозиторий

Не следует помещать в один репозиторий несколько разных пакетов, за исключением случаев, когда у этих пакетов есть общий пакет-предок.

- Плюсы: соблюдение этого правила облегчает совместную работу над пакетом, поскольку неперегруженный репозиторий легче клонировать, а инструменты git лучше подходят для работы с такими репозиториями.
- Минусы: несколько сложнее выполнять операции **`fetch`** и **`push`**, если репозиториев, которые надо обработать, много. Впрочем, выполнение **`fetch`**/**`push`** в цикле решает эту проблему.

Несжатый исходный код

Исходный код, сжатый различными средствами (**`gzip`**, **`bzip2`** и т.п.), рекомендуется хранить в git-репозитории в несжатом виде.

- Плюсы:
  - изменения удобнее отслеживать средствами git (**`git diff`**;
  - git сам хранит объекты в сжатом виде, поэтому двойное сжатие редко даёт выигрыш;
  - алгоритмы передачи данных git эффективнее работают с несжатыми данными.
- Минусы: может снижаться «нативность» исходного кода (из-за различий в способах сжатия).

Распакованный исходный код

Исходный код, упакованный архиваторами (**`tar`**, **`cpio`**, **`zip`** и т.п.), рекомендуется хранить в git-репозитории в распакованном виде.

- Плюсы:
  - существенно проще вносить изменения в конечные файлы и отслеживать их;
  - уменьшается объём передаваемых данных при обновлении.
- Минусы:
  - Git не сохраняет владельца, права доступа (кроме исполняемости) и время модификации;
  - архив, собранный из репозитория, может отличаться от оригинала;
  - в редких случаях это может повлиять на результат сборки.

Форматированный changelog

В changelog релизного коммита рекомендуется включать соответствующий текст из changelog пакета. Это делают утилиты **`gear-commit`** (обёртка к **`git commit`**, специально предназначенная для этих целей) и **`gear-srpmimport`**. Такой подход позволяет видеть изменения в очередном релизе пакета, не открывая в spec-файл.

7.2. Правила экспорта

С одной стороны, для того, чтобы srpm-пакет мог быть импортирован в git-репозиторий наиболее удобным для пользователя способом, язык правил, согласно которым производится экспорт из коммита репозитория (в форму, из которой можно однозначно изготовить srpm-пакет или запустить сборку), должен быть достаточно выразительным.

С другой стороны, для того, чтобы можно было относительно безбоязненно собирать пакеты из чужих gear-репозиториев, этот язык правил должен быть достаточно простым.

Файл правил экспорта (по умолчанию в **`.gear/rules`**) состоит из строк формата:

```
директива: параметры
```

Параметры разделяются пробелами.

Директивы позволяют экспортировать:

- любой файл из дерева, соответствующего коммиту;
- любой каталог из дерева, соответствующего коммиту, в виде tar- или zip-архива;
- unified diff между любыми каталогами, соответствующими коммитам.

Файлы на выходе могут быть сжаты разными средствами (gzip, bzip2 и т.п.). В качестве коммита может использоваться как целевой коммит (значение параметра **`-t`** утилиты **`gear`**), так и любой из его предков при соблюдении условий, гарантирующих однозначное вычисление полного имени коммита-предка по целевому коммиту.

Правила подробно описаны в документации gear-rules.

7.3. Основные типы устройства gear-репозитория

Правила экспорта реализуют основные типы устройства gear-репозитория следующим образом:

Архив с модифицированным исходным кодом

С помощью простого правила

```
tar: .
```

Все дерево исходного кода экспортируется в один tar-архив. Если у проекта есть upstream, публикующий tar-архивы, то добавление релиза в имя tar-архива, например, с помощью следующего правила позволяет избежать коллизий:

```
tar: . name=@name@-@version@-@release@
```

Архив с немодифицированным исходным кодом и патчем, содержащем локальные изменения

Если дерево с немодифицированным исходным кодом хранится в отдельном подкаталоге, а локальные изменения хранятся в gear-репозитории в виде отдельных патч-файлов, то правила экспорта могут выглядеть следующим образом:

```
tar: package_name
copy: *.patch
```

Такое устройство репозитория получается при использовании утилиты **`gear-srpmimport`**, предназначенной для быстрой миграции от srpm-файла к gear-репозиторию.

Смешанные типы

Вышеперечисленные типы устройства gear-репозитория являются основными, но не исчерпывающими. Правила экспорта достаточно выразительны для того чтобы реализовать всевозможные сочетания основных типов и создать полнофункциональный gear-репозиторий на любой вкус.

7.4. Быстрый старт gear

7.4.1. Создание gear-репозитория путем импорта SRPM-пакета

Исходные данные: srpm-пакет *foobar-1.0-alt1.src.rpm* со следующим содержимым:

```
$ rpm -qpl foobar-1.0-alt1.src.rpm
foobar-1-fix.patch
foobar-2-fix.patch
foobar.icon.png
foobar-1.0.tar.bz2
foobar-plugins.tar.gz
```

Для того чтобы сделать из этого SRPM-пакета gear-репозиторий, необходимо:

- создать каталог, в котором будет располагаться архив:

  ```
  $ mkdir foobar
  $ cd foobar
  ```

- создать новый git-репозиторий:

  ```
  $ git init
  Initialized empty Git repository in .git/
  ```

  Получившийся пустой git-репозиторий будет выглядеть примерно следующим образом:

  ```
  $ ls -dlog .*
  drwxr-xr-x 4 4096 Aug 12 34:56 .
  drwxr-xr-x 6 4096 Aug 12 34:56 ..
  drwxr-xr-x 8 4096 Aug 12 34:56 .git
  ```

  Таким образом, git-репозиторий готов для импорта SRPM-пакета.

- импортировать SRPM-пакет в git-репозиторий, воспользовавшись утилитой **gear-srpmimport**:

```
$ gear-srpmimport foobar-1.0-alt1.src.rpm
Committing initial tree deadbeefdeadbeefdeadbeefdeadbeefdeadbeef
gear-srpmimport: Imported foobar-1.0-alt1.src.rpm
gear-srpmimport: Created master branch
```

После выполнения импорта git-репозиторий будет выглядеть следующим образом:

```
$ ls -Alog
drwxr-xr-x 1 4096 Aug 12 34:56 .gear
drwxr-xr-x 1 4096 Aug 12 34:56 .git
-rw-r--r-- 1 6637 Aug 12 34:56 foobar.spec
drwxr-xr-x 3 4096 Aug 12 34:56 foobar
drwxr-xr-x 3 4096 Aug 12 34:56 foobar-plugins
-rw-r--r-- 1  791 Aug 12 34:56 foobar-1-fix.patch
-rw-r--r-- 1 3115 Aug 12 34:56 foobar-2-fix.patch
-rw-r--r-- 1  842 Aug 12 34:56 foobar.icon.png
```

при необходимости в файл правил можно вносить изменения. Например, можно убрать сжатие исходников (соответствующие изменения следует вносить и в **.gear/rules**).

7.4.2. Создание gear-репозитория на основе git-репозитория

Для того чтобы создать gear-репозиторий на основе git-репозитория необходимо:

- создать и добавить в git-репозиторий spec-файл;
- создать и добавить в git-репозиторий файл с правилами **.gear/rules**.

7.4.3. Сборка пакета из gear-репозитория

Сборка пакета при помощи hasher осуществляется командой **gear-hsh**:

```
$ gear-hsh
```

Чтобы собрать пакет, который не содержит определения тега ***Packager*** в spec-файле, следует отключить соответствующую проверку:

```
$ gear-hsh --no-sisyphus-check=gpg,packager
```

Сборка пакета при помощи **rpmbuild(8)** осуществляется командой **gear-rpm**:

```
$ gear-rpm -ba
```

7.5. Фиксация изменений в репозитории

Для выполнения коммита очередной сборки пакета рекомендуется использовать утилиту **gear-commit**, которая помогает сформировать список изменений на основе записи в spec-файле:

```
$ gear-commit -a
```

Перед первым коммитом необходимо настроить имя пользователя и адрес электронной почты. Это можно сделать глобально, например, прописав соответствующие значения в **~/.gitconfig**:

```
$ git config --global user.name 'Your Name'
$ git config --global user.email '<login>@altlinux.org'
```

Для отдельно взятого git-репозитория можно настроить имя и адрес электронной почты, прописав соответствующие значения в **.git/config** этого git-репозитория:

```
$ git config user.name 'Your Name'
$ git config user.email '<login>@altlinux.org'
```

Глава 8. Инструмент Hasher

Hasher — это инструмент безопасной и воспроизводимой сборки пакетов. Все пакеты репозитория Сизиф собираются с его помощью.

8.1. Принцип действия

Hasher — инструмент для сборки пакетов в «чистой» и контролируемой среде. Это достигается за счёт создания в chroot минимальной сборочной среды, установки в неё указанных в source-пакете сборочных зависимостей и сборке пакета в свежесозданной среде. Для сборки каждого пакета сборочная среда создается заново.

Такой принцип сборки имеет несколько следствий:

- все необходимые для сборки зависимости должны быть указаны в пакете. Для облегчения поддержания сборочных зависимостей в актуальном состоянии в Sisyphus используется инструмент ***buildreq***;

- сборка не зависит от конфигурации компьютера пользователя, собирающего пакет, и может быть повторена на другом компьютере;
- изолированность среды сборки позволяет с легкостью собирать на одном компьютере пакеты для разных дистрибутивов и веток репозитория — для этого достаточно лишь направить hasher на различные репозитории для каждого сборочного окружения.

8.2. Пакеты hasher

hasher в дистрибутиве ALT Platform Builder поставляется в пакетах *hasher*, *hasher-priv*, которые установлены по умолчанию.

8.3. Справочная информация по hasher

Для получения подробной справки по hasher можно воспользоваться командой:

```
$ man hsh
```

8.4. Настройка Hasher

8.4.1. Добавление пользователя

Hasher использует специальных вспомогательных пользователей и группу hashman для своей работы, поэтому каждого пользователя, который будет использовать hasher, перед началом работы нужно зарегистрировать:

```
# hasher-useradd <имя_пользователя>
```

Эта команда создает вспомогательных пользователей имя_пользователя_a и имя_пользователя_b и добавляет пользователя в группы hashman, имя_пользователя_a и имя_пользователя_b.

Поскольку **`hasher-useradd`** изменяет список групп, в которых состоит пользователь, пользователю перед началом работы с hasher необходимо перелогиниться.

8.4.2. Настройка сборочной среды

Для работы hasher требуется создать каталог, в котором будет размещаться сборочная среда:

```
$ mkdir ~/.hasher
```

Рабочий каталог (в данном случае **`~/.hasher`**) должен быть доступен на запись пользователю, запускающему сборку. Кроме того, его нельзя располагать на файловой системе, смонтированной с опциями **`noexec`** или **`nodev`** — в таких условиях hasher не сможет создать корректное сборочное окружение.

Команда создания сборочного окружения:

```
$ hsh --initroot-only ~/.hasher
```

Явное создание сборочного окружения необязательно — при необходимости оно будет создано при первой сборке пакета.

Hasher получает пакеты для установки из APT-источников. По умолчанию в сборочную среду копируется список источников, указанный в конфигурации APT хост-системы; также можно явно задать дополнительные репозитории, указав альтернативный файл конфигурации APT, например:

```
$ hsh --apt-config=.hasher/p11-apt.conf --initroot-only ~/.hasher
```

В таком файле конфигурации (в примере **p11-apt.conf**) необходимо указать расположение файла с APT-источниками, например:

```
Dir::Etc::SourceList "/home/user/.hasher/sources_p11.list";
```

Если необходимо создать сборочную среду, независимую по источникам от основной операционной системы, в вышеуказанный файл, помимо строки с источником, следует добавить следующую строку во избежание подключения **/etc/apt/sources.list.d/*.list**:

```
Dir::Etc::SourceParts "/var/empty";
```

По умолчанию (без указания ключа **--apt-config**) используется общесистемная конфигурация репозиториев из **/etc/apt/**. Чтобы не указывать каждый раз ключ **--apt config**, можно задать его в файле конфигурации **~/.hasher/config**, например:

```
apt_config=/home/user/.hasher/p11-apt.conf
```

Пример файла **~/.hasher/p11-apt.conf**:

```
Dir::Etc::SourceList "/home/user/.hasher/sources_p11.list";
Dir::Etc::SourceParts "/var/empty";
```

Пример файла **~/.hasher/sources_p11.list** с локальным репозиторием:

```
rpm file:/srv/public/mirror p11/branch/x86_64 classic
rpm file:/srv/public/mirror p11/branch/ x86_64-i586 classic
rpm file:/srv/public/mirror p11/branch/noarch classic
```

8.5. Сборка в hasher

Сборка выполняется от имени обычного пользователя, добавленного с помощью **hasher-useradd**:

```
$ hsh ~/.hasher path/to/package-0.0-alt0.src.rpm
```

При успешной сборке полученные пакеты будут находиться в каталоге **~/.hasher/repo/<платформа>/RPMS.hasher/**. В противном случае в стандартный вывод (stdout) будет выведена информация об ошибках сборки.

Для наблюдения за процессом сборки следует использовать ключ **-v**.

Создаваемый hasher репозиторий является обычным APT-репозиторием и может быть использован в **`sources.list`**. Он также будет использоваться при последующих сборках пакетов (это поведение можно регулировать ключом **`--without-stuff`**).

Если сборочная среда создана в tmpfs (см. ниже), каталог **`~/.hasher/repo`**, вероятно, не переживет перезагрузку системы. Репозиторий можно переместить в постоянное место, указав в файле конфигурации hasher (**`~/.hasher/config`**) параметр:

```
def_repo=постоянное_хранилище
```

или запустив hasher с ключом **`--repo`**.

8.6. Сборочные зависимости

Сборочные зависимости RPM делятся на два вида:

- необходимые для корректного создания src.rpm из spec-файла (содержащие определения RPM-макросов, используемых в spec-файле);
- все остальные (необходимые для непосредственной сборки).

Поскольку hasher собирает пакеты из src.rpm (не считая поддержки gear), для сборки в хост-системе необходимо иметь установленные сборочные зависимости первого типа. Большинство таких зависимостей (но пока не все) содержатся в пакетах с именами вида *rpm-build-**.

Сборка src.rpm либо завершается неудачно (при отсутствии сборочной зависимости первого типа), либо выполняется корректно. Поэтому собирать src.rpm-пакеты в хост-системе можно с помощью параметра **`--nodeps`**:

```
$ rpm -bs --nodeps package.spec
```

Hasher, в отличие от gear, не предъявляет требований к разделению сборочных зависимостей на первый и второй тип. Однако для совместимости с gear и для улучшения документируемости spec-файла рекомендуется распределять их следующим образом:

- в поле BuildRequires(pre) помещать сборочные зависимости, требуемые для сборки src.rpm;
- в поле BuildRequires — все остальные.

Примечание

В поле BuildRequires(pre) нельзя использовать макросы.

8.7. Монтирование файловых систем внутри hasher

Некоторым приложениям для сборки требуется смонтированная файловая система (например, **`/proc`**). Hasher поддерживает монтирование дополнительных файловых систем в сборочную среду.

Монтирование происходит при одновременном выполнении следующих условий:

- файловая система описана в файле **/etc/hasher-priv/fstab** либо является одной из предопределенных: **/proc**, **/dev/pts**, **/sys**;
- файловая система указана в опции ***allowed_mountpoints*** в конфигурации hasher-priv (**/etc/hasher-priv/system**);
- файловая система указана при запуске hasher в опции **--mountpoints** либо указана в параметре ***known_mountpoints*** конфигурационного файла hasher (**~/.hasher/config**);
- файловая система указана в сборочных зависимостях (например, ***BuildReq: /proc***) собираемого пакета — прямо или косвенно (через зависимости сборочных зависимостей пакета).

Для монтирования **/proc** необходимо:

- в **/etc/hasher-priv/system** добавить строку:

```
allowed_mountpoints=/proc
```

- в **~/.hasher/config** добавить строку (либо указывать опцию **--mountpoints=/proc** при сборке пакета):

```
known_mountpoints=/proc
```

- в spec-файле пакета указать:

```
BuildRequires: /proc
```

8.8. Использование нескольких сборочных окружений

Hasher не ограничивает пользователей одним сборочным окружением. Первый параметр, передаваемый в **hsh**, указывает на конкретное сборочное окружение, в котором необходимо выполнять сборку:

```
$ hsh --apt-conf=.hasher-p11/apt.conf.p11 ~/.hasher-p11 package 0.0 alt0.src.rpm
…
$ hsh --apt-conf=.hasher-test/apt.conf.test ~/.hasher-test package 1.0
alt0.src.rpm
…
```

По умолчанию используется каталог **~/hasher**.

8.9. Сборка пакетов на tmpfs

При наличии достаточного объёма оперативной памяти на сборочной машине сборку пакетов можно выполнять на tmpfs — такая конфигурация заметно ускоряет процесс.

Можно использовать уже смонтированный каталог **/tmp**:

```
$ mkdir -p /tmp/.private/$USER/hasher
$ hsh --repo=$HOME/hasher-repo /tmp/.private/$USER/hasher package 0.0
alt0.src.rpm
```

Каталог **/tmp/.private/$USER/hasher** необходимо создавать после каждой перезагрузки (это можно автоматизировать в файле **~/.hasher/config**, поскольку он является shell-скриптом).

Параметр **--repo** нужно указывать при каждой сборке (либо задать в **.hasher/config** параметр ***def_repo=постоянное_хранилище***).

8.10. Отключение проверок sisyphus_check

По умолчанию hasher запускает утилиту **sisyphus_check** с полным набором тестов. Она проверяет не только технические требования репозитория Sisyphus, но и организационные аспекты: сборочный хост, подпись PGP-ключом члена ALT Linux Team и т. д. Поэтому при сборке пакета вне репозитория Sisyphus может потребоваться отключение части проверок.

Для отключения части или всех проверок используется ключ **--no-sisyphus-check[=LIST]** или соответствующая опция ***no_sisyphus_check*** в конфигурационном файле.

Без аргумента этот ключ полностью отключает запуск **sisyphus_check**:

```
$ hsh --no-sisyphus-check ~/.hasher package 0.0 alt0.src.rpm
```

С аргументом (списком тестов) отключаются только указанные проверки:

```
$ hsh --no-sisyphus-check=packager,gpg ~/.hasher package 0.0 alt0.src.rpm
```

Список доступных тестов можно посмотреть в справке:

```
$ sisyphus_check --help
…
Valid options are:
…
--[no-]check-buildhost
--[no-]check-buildtime
--[no-]check-changelog
…
```

Более гибкая настройка возможна с помощью опций **--no-sisyphus-check-in** и **--no-sisyphus-check-out**, описание которых приведено в man-странице hsh(1).

8.11. Отладка в сборочном chroot

Для отладки сборки иногда полезно запустить shell в сборочном chroot. Для этого используется утилита **hsh-shell(1)**:

```
$ hsh-shell ~/.hasher
```

Можно запустить shell и с правами псевдо-root:

```
$ hsh-shell --rooter
```

8.12. Ограничение ресурсов

Hasher позволяет ограничить ресурсы, выделяемые на сборку: CPU, память, общее время исполнения и другие. Ограничения задаются в конфигурационном файле **hasher-priv**.

Полный список ограничиваемых ресурсов можно найти в man-странице **hasher-priv.conf(5)**.

8.13. Пересборка пакета без пересоздания всего chroot

Развертывание всей сборочной среды и зависимостей занимает значительное время.

Чтобы собирать один и тот же пакет до успешной сборки, можно указать hasher не пересоздавать сборочное окружение либо работать непосредственно внутри него.

8.13.1. Многократная сборка пакета в одном hasher

Если пакет не собрался, можно воспользоваться **hsh-shell** (предварительно установив текстовый редактор, shell и прочие инструменты разработчика):

```
$ hsh-install ~/.hasher vim-console less rpm-utils patchutils zsh
$ mkdir -p ~/.hasher/chroot/.in/src
$ cp -a .vim* .zprofile .zsh_aliases .zshenv .zsh_bind .zshrc .dircolors
~/.hasher/chroot/.in/src
$ hsh-run -- cp -r /.in/src /usr
$ hsh-shell --shell=/bin/zsh
```

В дереве каталогов hasher есть каталог **~/hasher/chroot/.in**, в которой пользователь может записывать файлы.

Отсутствующие сборочные зависимости можно доустановить с помощью **hsh-install** (выйдя из chroot).

8.13.2. Многократная сборка пакета при работе с gear

Если используется gear и разработка ведется в базовой системе, вместо **hsh** можно использовать **hsh-rebuild** — программу, работающую с уже сформированным сборочным окружением.

Первоначальная сборка (даже если прошла успешно, окружение не удаляется):

```
$ gear --hasher -- hsh --lazy-cleanup
```

Повторная сборка:

```
$ gear --hasher -- hsh-rebuild
```

Примечание

По окончании работы необходимо выполнить **buildreq** и забрать spec-файл:

```
$ hsh-install rpm-utils
$ echo buildreq '/usr/src/RPM/SPECS/*.spec' | hsh-shell
$ cp ~/hasher/chroot/usr/src/RPM/SPECS/*.spec .
```

Глава 9. Примеры сборки пакетов

Для примера сборки пакета используется программа для вывода системных уведомлений о текущей дате и времени. Ссылки на GitHub-репозитории с исходными текстами программ:

- C++: Notification — https://github.com/MakDaffi/notification;
- Python: DBusTimer_Example — https://github.com/danila-Skachedubov/DBusTimer_example.

Структура репозиториев для программ Notification и DBusTimer_Example идентична: главный файл (.cpp или .py) и два юнита systemd (.service и .timer):

- файл **.timer** — юнит systemd, который при истечении заданного времени вызывает скрипт .py, выводящий уведомление о дате и времени. После срабатывания таймер снова начинает отсчет времени до запуска скрипта;
- файл **.service** — содержит описание, расположение скрипта .py и интерпретатора, который будет его обрабатывать.

9.1. Пакет с исходными текстами на Python

9.1.1. Подготовка пространства

В первую очередь необходимо склонировать репозиторий в рабочий каталог, используя команду **git clone**:

```
$ git clone https://github.com/danila-Skachedubov/DBusTimer_example.git

remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 8 (delta 1), pack-reused 0
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done
```

В рабочем каталоге появится каталог с названием проекта: **DBusTimer_Example**.

Перейдите в каталог **DBusTimer_Example** и создайте в нем каталог **.gear**:

```
$ cd DBusTimer_example
$ mkdir .gear
```

В каталоге **.gear** создайте два файла: файл правил для gear — **rules** и spec-файл — **dbustimer.spec**:

```
$ touch .gear/rules .gear/dbustimer.spec
```

Содержимое каталога DBusTimer_Example в результате выполненных действий:

```
$ ls -a1
.
..
.gear
.git
script_dbus.py
script_dbus.service
script_dbus.timer
```

9.1.2. Написание spec-файла и правил gear

Следующий этап сборки — это написание spec-файла и правил для gear.

Заполните файл **.gear/rules** следующим содержимым:

```
tar: .
spec: .gear/dbustimer.spec
```

Первая строка указывает, что проект будет упакован в .tar архив. Вторая строка задаёт путь к spec-файлу.

Далее необходимо заполнить spec-файл.

В заголовке spec-файла находятся секции ***Name***, ***Version***, ***Release***, ***Summary***, ***License***, ***Group***, ***BuildArch***, ***BuildRequires***, ***Source0***.

Для данного примера можно заполнить секции заголовка следующим образом:

```
Name: dbustimer
Version: 0.4
Release: alt1

Summary: Display system time
License: GPLv3+
Group: Other
BuildArch: noarch

BuildRequires: rpm-build-python3
```

Стандартная схема ***Name-Version-Release***, содержит имя пакета, его версию и релиз сборки. Поле ***Summary*** содержит краткое описание пакета. ***License*** — лицензия, под которой распространяется ПО (в данном случае — GPLv3+). ***Group*** — категория пакета. Так как это тестовый пакет, можно указать группу «Other». ***BuildRequires*** — пакеты, необходимые для сборки. Поскольку исходный код написан на Python 3, требуется пакет *rpm-build-python3*, который содержит макросы для сборки скриптов Python. ***Source0*** — путь к архиву с исходниками (%name-%version.tar).

В основной части spec-файла описываются процесс сборки и преобразование исходных файлов.

В секции ***%description*** приводится краткое описание программы.

Секция ***%prep*** отвечает за подготовку к сборке. Макрос ***%setup*** распаковывает исходный код перед компиляцией. Пример заполнения полей ***%description*** и ***%prep***:

```
%description
This program displays notifications about the system time with a frequency of one
hour.

%prep
%setup -q
```

В секции ***%install*** указываются пути установки файлов. Чтобы не указывать пути установки файлов вручную, можно использовать предопределенные макросы. Например, ***%python3_sitelibdir_noarch*** раскрывается в путь **/usr/lib/python3/site-packages**. По этому пути будет создан каталог с именем пакета, в который будет помещен файл **script_dbus.py** с правами доступа 755. Аналогичная операция будет проведена с файлами **script_dbus.timer** и **script_dbus.service**. Они должны быть скопированы в каталог **/etc/xdg/systemd/user**. Так как макроса, раскрывающегося в данный каталог нет, можно использовать макрос ***%_sysconfdir***, который раскрывается в путь **/etc**. Пример заполнения секции ***%install***:

```
%install


mkdir -p \
    %buildroot%python3_sitelibdir_noarch/%name/
install -Dm0755 script_dbus.py \
    %buildroot%python3_sitelibdir_noarch/%name/

mkdir -p \
    %buildroot%_sysconfdir/xdg/systemd/user/
cp script_dbus.timer script_dbus.service \
    %buildroot%_sysconfdir/xdg/systemd/user/
```

Команда **mkdir -p \ %buildroot%python3_sitelibdir_noarch/%name/** создает каталог **dbustimer** в окружении buildroot по пути **/usr/lib/python3/site-packages**.

Далее файл **script_dbus.py** копируется с правами 755 в каталог **/usr/lib/python3/site-packages/dbustimer/** в окружении buildroot.

Аналогично создается каталог для systemd-юнитов **%buildroot%_sysconfdir/xdg/systemd/user/**, в который копируются файлы **.service** и **.timer**.

В секции ***%files*** описывается, какие файлы и каталоги с соответствующими атрибутами должны быть скопированы из дерева сборки в rpm-пакет, а затем будут копироваться в целевую систему при установке этого пакета. Все три файла из пакета будут распакованы по путям, описанным в секции ***%install***. Пример заполнения секции ***%files***:

```
%files
%python3_sitelibdir_noarch/%name/script_dbus.py
/etc/xdg/systemd/user/script_dbus.service
/etc/xdg/systemd/user/script_dbus.timer
```

В секции ***%changelog*** фиксируются изменения:

```
%changelog
* Thu Apr 13 2023 Danila Skachedubov <dan@altlinux.org> 0.4-alt1
- Update system
- Changed access rights
```

Итоговый spec-файл:

```
Name: dbustimer
Version: 0.4
Release: alt1

Summary: Display system time
License: GPLv3+
Group: Other
BuildArch: noarch

BuildRequires: rpm-build-python3

Source0: %name-%version.tar

%description
This program displays notifications about the system time with a frequency of one
hour.

%prep
%setup

%install

mkdir -p \
    %buildroot%python3_sitelibdir_noarch/%name/
install -Dm0755 script_dbus.py \
    %buildroot%python3_sitelibdir_noarch/%name/

mkdir -p \
    %buildroot%_sysconfdir/xdg/systemd/user/
cp script_dbus.timer script_dbus.service \
    %buildroot%_sysconfdir/xdg/systemd/user/


%files
%python3_sitelibdir_noarch/%name/script_dbus.py
/etc/xdg/systemd/user/script_dbus.service
/etc/xdg/systemd/user/script_dbus.timer
```

```
%changelog
* Thu Apr 13 2023 Danila Skachedubov <dan@altlinux.org> 0.4-alt1
- Update system
- Changed access rights
```

После заполнения файлов добавьте их в git:

```
$ git add .gear/rules .gear/dbustimer.spec
```

После добавления файлов необходимо запустить сборку:

```
$ gear-hsh ~/.hasher --no-sisyphus-check --commit –v
```

Примечание

Hasher и git должны быть предварительно настроены.

Если сборка прошла успешно, пакет *dbustimer-0.4-alt1.noarch.rpm* будет находиться в каталоге **~/.hasher/repo/x86_64/RPMS.hasher/**.

9.2. Пакет с исходными текстами на C++

Программа Notification выводит системное уведомление о текущей дате и времени в формате: день недели, месяц, число, чч:мм:сс, год.

В репозитории находятся следующие файлы:

- **.gear** — каталог с правилами gear и spec-файлом;
- **Makefile** — набор инструкций для программы **make**, которая собирает данный проект;
- **notify.cpp** — исходный код программы;
- **notify.service** — юнит данной программы для systemd;
- **notify.timer** — юнит systemd, запускающий вывод уведомления о дате и времени с периодичностью один час.

В каталоге **.gear** находятся два файла:

- **rules** — правила для упаковки архива для gear;
- **notify.spec** — файл спецификации для сборки пакета.

Содержимое файла **rules**:

```
tar: .
spec: .gear/notify.spec
```

Первая строка — указания для gear, в каком формате упаковать файлы для последующей сборки. В данном проекте архив будет иметь вид name-version.tar. Вторая строка — путь к spec-файлу с инструкциями по сборке текущего пакета.

Содержимое файла **notify.spec**:

```
Name: notify
Version: 0.1
Release: alt1

Summary: Display system time every hour
License: GPLv3+
Group: Other

BuildRequires: make
BuildRequires: gcc-c++
BuildRequires: libsystemd-devel

Source0: %name-%version.tar

%description
This test program displays system date and time every hour via notification

%prep
%setup -q

%build
%make_build

%install

mkdir -p \
    %buildroot/bin/
install -Dm0644 %name %buildroot/bin/

mkdir -p \
    %buildroot%_sysconfdir/xdg/systemd/user/
cp %name.timer %name.service \
    %buildroot%_sysconfdir/xdg/systemd/user/

%files
/bin/%name
/etc/xdg/systemd/user/%name.service
/etc/xdg/systemd/user/%name.timer

%changelog
* Thu Apr 13 2023 Sergey Okunkov <sok@altlinux.org> 0.1-alt1
- Finished my task
```

В заголовке spec-файла описаны следующие поля:

- ***Name***, ***Version***, ***Release*** — стандартная схема Name-Version-Release, содержащая имя пакета, его версию и релиз сборки;
- ***Summary*** — краткое описание пакета;
- ***License*** — лицензия, под которой распространяется данное ПО (в данном случае — GPLv3+);
- ***Group*** — категория, к которой относится пакет (в примере указана группа «Other»);
- ***BuildRequires*** — пакеты, необходимые для сборки. Поскольку исходный код написан на C++, требуется компилятор g++, система сборки make и библиотека для работы с systemd — libsystemd-devel;

- ***Source0*** — путь к архиву с исходниками (%name-%version.tar).

В основной части spec-файла описываются процесс сборки и инструкции к преобразованию исходных файлов:

- секция ***%description*** содержит описание программы. В данном примере — вывод системного уведомления с датой и временем;
- секция ***%prep*** отвечает за подготовку программы к сборке. Макрос ***%setup*** с флагом **–q** распаковывает архив, указанный в секции ***Source0***;
- секция ***%build*** описывает сборку исходного кода. Поскольку в проекте используется **Makefile** для автоматизации процесса сборки, применяется макрос ***%make_build***, использующий **Makefile** для сборки программы;
- секция ***%install*** определяет пути установки файлов. Так как файлов три, для каждого задаётся путь:
  - **notify** — скомпилированный бинарный файл. В Unix-подобных системах такие файлы обычно располагаются в каталоге **/bin**. Команда **mkdir -p %buildroot/bin** создаёт каталог **bin** в окружении buildroot. Команда **install -Dm0644 %name %buildroot/bin/** устанавливает файл **notify** в каталог **%buildroot/bin/** с правами 644;
  - **%name.timer**, **%name.service** — пользовательские юниты systemd. Они размещаются в каталоге **/etc/xdg/systemd/user/**. В окружении buildroot создается каталог:

    ```
    mkdir –p %buildroot%_sysconfdir/xdg/systemd/user/
    ```

    Здесь используется макрос ***%_sysconfdir***, который раскрывается в **/etc**. Команда:

    ```
    cp %name.timer %name.service %buildroot%_sysconfdir/xdg/systemd/user/
    ```

    копирует файлы в указанный каталог;
- секция ***%files*** — описывает файлы и каталоги, которые будут скопированы в систему при установке пакета.

Глава 10. Сборка образов с помощью mkimage-profiles

Система генерации дистрибутивов использует все преимущества банка пакетов и позволяет получать установочные образы. Для сборки образа используется утилита **mkimage**, которая использует для сборки «профиль», представляющим собой набор файлов Makefile. В результате из пакетов репозитория создается установочный диск CD/DVD. Целостность репозитория и его непротиворечивость позволяют с легкостью генерировать новые образы при необходимости.

mkimage-profiles (m-p) — система управления конфигурацией семейств дистрибутивов свободного программного обеспечения из репозиториев ALT для различных платформ.

Концепция:

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

Особенности:

- метапрофиль при сборке может быть доступен только для чтения;
- для сборки предпочтительно использовать tmpfs;
- в профиль копируются только нужные объекты (он автономен относительно метапрофиля).

Стадии работы:

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

Примечание

Сборка образов может занимать большой объем дискового пространства (например, порядка 100 Гб).

Предварительные настройки:

- пользователь с правом запуска hasher и подключения **`/proc`** к нему (см. Настройка Hasher);
- смонтированный tmpfs на несколько гигабайт (можно указать в переменной BUILDDIR):
  - например, в **`/tmp`** или **`/home/USER/hasher`**;
  - каталог из prefix в **`/etc/hasher-priv/system`**;
- настроенный **`~/.gitconfig`**.

Объекты:

- дистрибутивы и виртуальные среды/машины:
  - описываются в **`conf.d/*.mk`**;
  - могут основываться на предшественниках, расширяя их;
  - дистрибутивы могут включать один или более субпрофилей;
  - следует избегать множественного наследования;
- субпрофили:
  - список задаётся в $(SUBPROFILES);
  - базовые комплекты размещены в подкаталогах **`sub.in/`**;
  - наборы скриптов базовых комплектов могут расширяться фичами (features);

- фичи (features):
  - законченные блоки функциональности (или их наборы);
  - описываются в индивидуальных **`features.in/*/config.mk`**;
  - могут требовать другие фичи, а также субпрофили;
  - накопительный список формируется в $(FEATURES);
  - при сборке в $(BUILDDIR) содержимое фич добавляется в профиль;
- списки пакетов (*_LISTS):
  - не следует создавать отдельную фичу, если достаточно списка пакетов;
  - по возможности следует избегать дублирования (см. **`bin/pkgdups`**.);
- индивидуальные пакеты (*_PACKAGES):
  - см. **`conf.d/README`**.

Результат:

- при успешном завершении сборки образ получает имя цели и размещается в $(IMAGEDIR):
  - указанном явно;
  - **`~/out/`** (если возможно);
  - **`$(BUILDDIR)/out/`**;
- формируются отчеты, если это запрошено (переменная REPORT).

При запуске сборки принимается ряд переменных (см. **`profiles.mk.sample`**). Переменные могут быть заданы как в команде сборки (в качестве аргументов), так и в файле настроек **`$HOME/.mkimage/profiles.mk`**. Список переменных приведен в Переменные, принимаемые при сборке.

Таблица 10.1. Переменные, принимаемые при сборке

Переменная	Описание	Значение
APTCONF	Задаёт путь к apt.conf	Пусто (по умолчанию системный) либо строка
ARCH	Задаёт целевую архитектуру образов	Пусто (по умолчанию авто) либо строка
ARCHES	Задаёт набор целевых архитектур при параметрическом задании APTCONF	Пусто (по умолчанию авто) либо список через пробел
AUTOCLEAN	Включает очистку (distclean) после успешной сборки образа	Пусто (по умолчанию нет) либо любая строка
BELL	Подаёт сигнал после завершения сборки	Пусто (по умолчанию нет) либо любая строка
BRANCH		

Переменная	Описание	Значение
	Указывает, для какой ветки производится сборка	не определено — пытается определиться автоматически; пусто — присваивается значение sisyphus; имя ветки (sisyphus, p11)
BUILDDIR	Задаёт каталог генерируемого профиля и сборки	Пусто (по умолчанию авто) либо строка
BUILDDIR_PREFIX	Задаёт префикс каталога генерируемого профиля и сборки	Строка; по умолчанию выбирается алгоритмически
BUILDLOG	Задаёт путь к файлу журнала сборки/очистки	$(BUILDDIR)/build.log (по умолчанию) либо строка
CHECK	Включает режим проверки сборки конфигурации (без сборки образа)	пусто (по умолчанию) — проверка не выполняется; 0 — проверяется только конфигурация, списки пакетов не проверяются; другое значение — полная проверка
CLEAN	Экономия RAM+swap при сборке в tmpfs. Очистка рабочего каталога после успешной сборки очередной стадии	Пусто, 0, 1, 2; по умолчанию пусто при DEBUG, иначе 1
DEBUG	Включает средства отладки, может отключить очистку после сборки	Пусто (по умолчанию), 1 или 2
DISTRO_VERSION	Задаёт версию дистрибутива (если применимо)	Пусто (по умолчанию) либо любая строка
HOMEPAGE, HOMENAME, HOMEWAIT	Указывают адрес, название и таймаут перехода для домашней страницы	Корректный URL, строка, неотрицательное целое число
IMAGEDIR	Указывает путь для сохранения собранного образа	Равно $HOME/out (если существует), иначе $(BUILDDIR)/out (по умолчанию), либо другой путь
ISOHYBRID	Включает создание гибридного ISO-образа	Пусто (по умолчанию) либо любая строка
LOGDIR	Указывает путь для сохранения логов сборки	Равно $(IMAGEDIR) (по умолчанию) либо другой путь
MKIMAGE_PREFIX		

Переменная	Описание	Значение
	Указывает путь к mkimage. Если параметр не задан, используется системный mkimage	
NICE	Понижает нагрузку системы от сборочной задачи	Пусто (по умолчанию) либо любая строка
NO_SYMLINK	Не создавать символические ссылки на собранный образ	Пусто (по умолчанию) либо любая строка
QUIET	Отключает поясняющие сообщения при сборке (например, при запуске через cron)	Пусто (по умолчанию) либо любая строка
REPORT	Запрашивает создание отчётов о собранном образе. Требует включения DEBUG и отключения CHECK	- пусто (по умолчанию) — создание отчёта отключено; - 2 — создать архив из каталога отчёта; - любое другое непустое значение — создать отчёт в виде каталога
ROOTPW	Устанавливает пароль root по умолчанию для образов виртуальных машин	Пусто (по умолчанию root) либо строка
SAVE_PROFILE	Сохраняет архив сгенерированного профиля в .disk/	Пусто (по умолчанию) либо любая строка
SORTDIR	Дополнительно структурирует каталог собранных образов	Пусто (по умолчанию) либо строка (например, $(IMAGE_NAME)/$(DATE))
SQUASHFS	Определяет способ сжатия squashfs для stage2	- пусто (по умолчанию) либо normal — xz; - tight — xz с ключом `-Xbcj` по платформе (лучше, но дольше — подбор в два прохода); - fast — gzip/lzo (быстрее запаковывается и распаковывается, но степень сжатия меньше)
STATUS	Добавляет в имя образа указанный префикс	Пусто (по умолчанию) либо строка (например, «alpha», «beta»)

Переменная	Описание	Значение
STDOUT	Выводит сообщения (при включённом DEBUG) одновременно в лог и на экран	1 — включить вывод на экран, если включен DEBUG
USE_QEMU	Использовать qemu, если архитектура не совпадает	1 (по умолчанию), для отключения — любое другое значение
VM_SAVE_TARBALL	Указывает, что нужно сохранить промежуточный тарбол, из которого создается образ виртуальной машины, в заданном формате	tar tar.gz tar.xz
VM_SIZE	Задаёт размер несжатого образа виртуальной машины в байтах	Пусто (по умолчанию двойной размер chroot) или целое число

Примечание

Чтобы при указании переменной BRANCH сборка выполнялась для целевой ветки, необходимо:

- прописать в **`~/.mkimage/profiles.mk`**:

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

- создать целевые конфигурационные файлы apt по указанным путям.

Кроме того, переменная BRANCH (если задана) заменяет в имени собираемой цели слово «regular» на «alt-$BRANCH». Таким образом обеспечивается сборка стартеркитов из профиля regular для заданной ветки.

Список доступных целей:

```
$ make -C /usr/share/mkimage-profiles help
```

Список доступных образов:

```
$ make -C /usr/share/mkimage-profiles help/distro
```

Пример команды сборки образа:

```
$ make -C /usr/share/mkimage-profiles syslinux.iso
```

Пример команды сборки образа с указанием переменных:

```
$ make -C /usr/share/mkimage-profiles DEBUG=1 BRANCH=p11 APTCONF=/etc/apt/
apt.conf.local alt-server.iso
```

Содержимое **`apt.conf.local`** может выглядеть так:

```
Dir::Etc::main "/dev/null";
Dir::Etc::parts "/ver/empty";
Dir::Etc::SourceParts "/var/empty";
Dir::Etc::sourcelist "/etc/apt/sources.list.d/alt-local.list";
APT::Acquire::Retries "3";
APT::Cache-Limit 201326592;
//Acquire::http::AllowRedirect "true";

RPM
{
        Allow-Duplicated {
                // Old-style kernels.
                "^(NVIDIA_)?(kernel|alsa)[0-9]*(-adv|-linus)?($|-up|-smp|-
secure|-custom|-enterprise|-BOOT|-tape|-aureal)";
                // New-style kernels.
                "^kernel-(image|modules)-.*";
        };
        Hold {
                // Old-style kernels.
                "^(kernel|alsa)[0-9]+-source";
        };
};
```

Глава 11. JOIN

11.1. Разработка в ALT Linux Team

ALT Linux Team — команда независимых разработчиков свободного программного обеспечения, которые совместно работают над поддержкой наборов программ в репозитории Сизиф (Sisyphus).

У каждого пакета в Sisyphus есть один или несколько мейнтейнеров. Мейнтейнер — это человек, который собирает пакет для установки программы из централизованного репозитория, исправляет обнаруженные в программе ошибки, общается с разработчиками программы, старается реагировать на нужды пользователей (багрепорты, вопросы). Мейнтейнер должен быть членом ALT Linux Team (команды ALT).

Join — это процесс вступления в ALT Linux Team, результатом которого является возможность непосредственно участвовать в разработке репозитория Сизиф. После прохождения Join кандидат (человек, вступающий в ALT Linux Team) становится мейнтейнером.

Для принятия человека в Team создается небольшая команда:

- Секретарь команды. Секретарь — это административная должность в ALT Linux Team. В его обязанность входит отслеживание стадий принятия и выполнение административных действий.

- Ментор вступающего. Задача ментора — способствовать кандидату при вступлении в Team. Он помогает кандидату со вступлением, отвечает на его вопросы и принимает решение о готовности кандидата.

- Рецензент работы вступающего. Задача рецензента — оценить готовность кандидата к вступлению в Team. Рецензент проводит независимую оценку по результатам работы кандидата и подтверждает его готовность.

- Кандидат — вступающий в ALT Linux Team человек.

При взаимодействии друг с другом можно использовать номер стадии, до которой дошел кандидат. Например, секретарь регистрирует ключ кандидата и переводит процесс на стадию 2.2; или ментор решает, что его подопечный готов отправлять пакеты в Сизиф, и переводит процесс на стадию 4.0.

Официальное взаимодействие происходит в Bugzilla в содержимом особым образом оформленной ошибки (бага). Открытый баг означает заявку на вступление в Team, закрытый — само вступление или отказ в нём.

11.2. Создание заявки на вступление в ALT Linux Team

Для принятия в Team необходима следующая информация:

- имя ментора — участника команды, имеющего желание помогать в процессе приема в Team. Менторов можно искать в списках рассылки;

- псевдоним (имя пользователя) участника. Псевдоним выбирается самим участником. Имя должно начинаться с буквы, содержать только строчные латинские буквы и цифры, быть не короче трёх символов;

- адрес почты, на который будет производиться пересылка с адреса псевдоним@altlinux.org;

- SSH-ключ (ED25519 или RSA размером не менее 4096 бит). Принимающей стороне нужна публичная часть ключа. Этот ключ будет использоваться для SSH-доступа к ресурсам Sisyphus (git.alt и другие);

- GPG-ключ (RSA размером не менее 4096 бит). В ключе должны быть указаны имя в формате <First name> <Last name> и uid вида псевдоним@altlinux.org. Принимающей стороне нужна публичная часть ключа. Этот ключ будет использоваться для подписи пакетов и для удостоверения личности в почте.

Создание SSH и GPG-ключей описано в разделе Работа с ключами разработчика.

Примечание

Псевдоним — это фактически второе имя человека в команде. Его лучше выбирать коротким, запоминающимся и не перегруженным. Например, yoda — удобный псевдоним, а travellingwilburys1998 — неудобный. Список уже занятых имён можно посмотреть в пакете *alt-gpgkeys*.

Заявка на принятие в ALT Linux Team создается кандидатом в Bugzilla (https://bugzilla.altlinux.org/). В данном случае Bugzilla выступает площадкой для вступления в ALT Linux Team. Здесь ведётся официальный диалог с кандидатами. Посмотреть прогресс других кандидатов и узнать, какие задачи они взяли, можно по ключевому слову «join».

Примечание

Чтобы сообщать о новых ошибках или оставлять комментарии к существующим в Bugzilla, необходимо иметь учётную запись. Она позволяет другим пользователям идентифицировать автора, оставившего комментарии к ошибкам или изменившего их состояние. Для регистрации достаточно указать адрес электронной почты — на него будет отправлено письмо с подтверждением.

Регистрация заявки происходит следующим образом:

- на главной странице ALT Bugzilla выбрать ссылку **Зарегистрировать ошибку** или **Новая ошибка**;
- выбрать раздел **Team Accounts** (ALT Linux Team: присоединение);
- зарегистрировать заявку.

Заявка (баг) должна быть оформлена следующим образом:

- в поле **Компонент** необходимо выбрать ***join*** (в поле **Продукт** должен быть указан ***Team Accounts***);
- поле **Аннотация** предназначено для краткого описания — здесь можно указать ключевое слово ***join*** и предполагаемый адрес почты: join user@;
- значение поля **Серьезность** можно оставить по умолчанию: ***normal***;
- платформа определяется автоматически, но её можно изменить, если планируется собирать пакеты для другой платформы (в том числе указать ***all*** — все платформы);
- в теле заявки (поле **Подробности**) нужно указать:
  - псевдоним (имя пользователя) нового участника;
  - адрес пересылки почты;
  - имя ментора;
  - несколько слов о том, чем кандидат намерен заняться в ALT Linux Team на момент подачи заявки на вступление (например, «собрать для начала такой-то пакет, а потом пакеты из такой-то области», «помочь со сборкой чего-нибудь», «научиться собирать пакеты» и т. п.). От заявленной кандидатом цели могут зависеть, в частности, требования ментора и рецензента к обретаемым кандидатом навыкам и их пристрастие.

Пример комментария к открытой заявке на вступление в ALT Linux Team:

```
Псевдоним: sova
Почта: Valentin Sokolov <sova@altlinux.org>
Адрес пересылки почты: mail@mail.com
Имя ментора: Grigory Ustinov
Почта ментора: grenka@altlinux.org

Хочу научиться собирать пакеты.
```

- приложить к заявке публичный SSH-ключ и публичный GPG-ключ в виде отдельных файлов (attachments). GPG-ключ следует приложить в экспортированном виде (**`gpg --export --armor <id ключа>`**). Файлы можно прикладывать после создания заявки. Важно убедиться, что экспортирован только один ключ (gpg %путь-до-файла%);
- после создания заявки следует добавить e-mail ментора в поле **Подписчики**, чтобы он мог подтвердить свое менторство.

11.3. Обработка заявки

После получения необходимой информации секретарь проверяет приложенные ключи, создает e-mail адрес и выдает ограниченный доступ в git.alt (без возможности сборки пакетов).

Секретарь ведет процесс обработки заявки по регламенту. При переходе на новый этап он обычно указывает номер стадии в открытой кандидатом заявке.

После успешного завершения процедуры Join секретарь выдает полный доступ в git.alt. С этого момента кандидат становится полноправным членом команды.

11.4. Работа с ключами разработчика

При приёме в Team участник предоставляет два криптографических ключа, по которым он идентифицируется в дальнейшем. Ключи подписи необходимо хранить в недоступном для других месте.

При утере одного из ключей участник может заменить его, заверив вторым. При утрате обоих ключей участник обязан незамедлительно известить об этом принимающих. Его доступ в git.alt прекращается до восстановления ключей.

Ключи могут быть восстановлены либо при личной встрече с одним из принимающих, либо путём отправки их письмом, заверенным ключом одного из участников ALT Linux Team. В последнем случае ответственность за безопасность репозитория несёт участник, заверивший ключи.

11.4.1. Создание SSH-ключа

Создать SSH-ключ можно, например, следующей командой:

```
$ ssh-keygen [-t <тип ключа>] [-b <размер ключа (для RSA)>]
```

Рекомендуется указывать тип ключа ED25519 или RSA с размером не менее 4096 бит. Ключ настоятельно рекомендуется защитить паролем.

Пример создания SSH-ключа:

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_ed25519.
Your public key has been saved in /home/user/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:7gUUwuricYRlF2mQxl4429Y5Dl2xECRSxcX67yPJrdU user@platform
The key's randomart image is:
+--[ED25519 256]--+
|   .o*=**+o.     |
|    0.*o.oo.     |
|   * 0 o.+.      |
|  . = +.*        |
|   o . oSo       |
|  o o  ....  .   |
| . +    ...+. E  |
|  .    . .+.+    |
|        . .+..   |
+----[SHA256]-----+
```

На вопрос о файле сохранения ключа можно ничего не вводить, а принять путь по умолчанию, нажав **Enter**. В этом случае публичная часть ключа — файл **~/.ssh/id_ed25519.pub** для ED25519-ключа или **~/.ssh/id_rsa.pub** для RSA-ключа.

Примечание

Файлы **~/.ssh/id_ed25519** и **~/.ssh/id_rsa** — это секретные (закрытые) ключи. Никогда и никому не следует передавать секретный ключ.

Для упрощения работы с ключами, защищёнными паролем, можно использовать SSH-агент.

11.4.2. Создание GPG-ключа

Создать новый GPG-ключ можно, выполнив команду:

```
$ gpg --gen-key
```

В процессе ответа на вопросы следует выбрать:

- тип ключа — RSA и RSA;
- размер — не менее 4096 бит;
- срок действия ключа — без ограничения;
- имя — имя в формате <Имя> <Фамилия>;
- email — псевдоним@altlinux.org (желательно только строчные латинские буквы);
- комментарий — лучше оставить пустым.

Все входные данные для **gpg --gen-key** должны быть указаны в ASCII.

Пример создания GPG-ключа:

```
$ gpg --gen-key
Выберите тип ключа:
(1) RSA и RSA (по умолчанию)
(2) DSA и Elgamal
(3) DSA (только для подписи)
(4) RSA (только для подписи)
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096 бит.
Какой размер ключа Вам необходим? (2048) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
        0 = без ограничения срока действия
    <n>  = срок действия - n дней
    <n>w = срок действия - n недель
    <n>m = срок действия - n месяцев
    <n>y = срок действия - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

Для идентификации Вашего ключа необходим ID пользователя. Программа создаст его
из Вашего имени, комментария и адреса электронной почты в виде:
"Baba Yaga (pensioner) <yaga@deepforest.ru>"

Ваше настоящее имя: Valentin Sokolov
Адрес электронной почты: sova@altlinux.org
Комментарий:
Вы выбрали следующий ID пользователя:
"Valentin Sokolov <sova@altlinux.org>"

Сменить (N)Имя, (C)Комментарий, (E)адрес или (O)Принять/(Q)Выход? O
Для защиты секретного ключа необходима фраза-пароль.
```

Экспорт публичной части ключа из связки выполнятся командой:

```
$ gpg --armor --export псевдоним@altlinux.org
```

Примечание

Может быть экспортировано несколько ключей с одним uid (email), тогда как требуется один. В этом случае (например, после редактирования) следует удалить лишние ключи из связки перед экспортом:

```
$ gpg --delete-key старый/ключ
```

Для сохранения публичной части ключа в файл можно использовать:

```
$ gpg --armor --export псевдоним@altlinux.org > public.key
```

11.4.3. Модификация GPG-ключа

Необходимо убедиться, что тип ключа — RSA, а размер — не менее 4096 бит. Добавить идентификатор в ключ можно с помощью команд:

```
$ gpg --edit-key псевдоним@altlinux.org
gpg> adduid
```

Далее следует указать имя (в формате <Имя> <Фамилия>) и uid (email) вида псевдоним@altlinux.org, после чего сохранить изменения:

```
$ gpg --edit-key псевдоним@altlinux.org
gpg> save
```

11.4.4. Обновление GPG-ключа в пакете alt-gpgkeys

Для замены просроченного или недействительного ключа, используемого для подписи пакетов, необходимо:

1. Клонировать последнюю актуальную версию репозитория *alt-gpgkeys.git*.
2. Обновить в нём свой ключ.
3. Отправить (push) модифицированную версию на git.alt в своё личное пространство (private repo) — этим подтверждается аутентичность изменений.
4. Далее необходимо (пере)открыть заявку на пакет (компонент) *alt-gpgkeys* для продукта Sisyphus в Bugzilla и дождаться реакции мейнтейнера.

Также можно приложить новый ключ, экспортированный в текстовый файл, к заявке.

Глава 12. Контейнеры и OCI-реестры

Современная разработка и эксплуатация программного обеспечения всё чаще опираются на контейнеризацию. Контейнерные образы позволяют стандартизировать окружение приложений, упростить их доставку и обеспечить воспроизводимость развертывания.

В данной главе рассматривается набор инструментов для работы с контейнерными образами и OCI-совместимыми реестрами. Описываются как базовые операции (сборка, запуск, публикация образов), так и более продвинутые сценарии — зеркалирование, модификация, анализ уязвимостей и управление метаданными.

12.1. Podman

Podman — инструмент для управления контейнерами и группами контейнеров (подами), совместимый с Docker CLI и Docker-образами. **Podman** позволяет запускать контейнеры без демона (daemonless) и поддерживает rootless-режим для повышения безопасности.

Основные возможности **Podman**:

- запуск и управление отдельными контейнерами и подами (группами контейнеров, которые разделяют сетевое пространство имён и другие ресурсы);
- поддержка Docker-совместимых команд и образов;
- rootless-режим (работа без прав суперпользователя);
- отсутствие необходимости в постоянно работающем демоне, что упрощает интеграцию с systemd и CI/CD.

12.1.1. Установка и настройка Podman

Установка пакета:

```
# apt-get install podman
```

Проверка версии установленного пакета:

```
# podman --version
podman version 5.7.0
```

12.1.1.1. Настройка rootless-режима

Для работы с **Podman** в rootless-режиме необходимо выполнить ряд дополнительных действий:

1. Убедитесь, что разрешено создание пользовательских пространств имён:

   ```
   # sysctl kernel.unprivileged_userns_clone
   kernel.unprivileged_userns_clone = 0
   ```

   Если значение 0, установите пакет:

   ```
   # apt-get install sysctl-conf-userns
   ```

2. Предоставьте пользователям право запуска исполняемых файлов **`/usr/bin/newuidmap`** и **`/usr/bin/newgidmap`**:

   ```
   # control newgidmap public
   # control newuidmap public
   ```

12.1.1.2. Основные команды для работы с Podman

Таблица 12.1. Основные команды

Действие	Команда
Список контейнеров	**`podman ps`**

Действие	Команда
Создание контейнера	**podman create [опции] imageName**
Запуск контейнера	**podman start containerID**
Остановка контейнера	**podman stop containerID**
Принудительное завершение	**podman kill containerID**
Перезапуск	**podman restart containerID**
Просмотр логов контейнера	**podman logs containerID**
Выполнение команды в контейнере	**podman exec [опции] containerID command**
Информация о контейнере	**podman inspect containerID**
Список образов	**podman images**
Сборка образа	**podman build -t imageName .**
Загрузка образа в репозиторий	**podman push imageName**
Загрузка образа из репозитория	**podman pull imageName**
Удаление образа	**podman rmi imageName**
Запуск из YAML	**podman play kube file.yaml**

Примеры:

- запуск контейнера ALT:

```
$ podman run registry.altlinux.org/alt/alt:p11 uname -a
Trying to pull registry.altlinux.org/alt/alt:p11...
Getting image source signatures
Copying blob 8d382247a69f done   |
Copying blob c549b474d68c done   |
Copying config 97c70e35e5 done   |
Writing manifest to image destination
Linux 0985c00c38ea 6.12.63-6.12-alt1 #1 SMP PREEMPT_DYNAMIC Tue Dec 30
19:10:41 UTC 2025 x86_64 GNU/Linux
```

- список образов:

```
$ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED       SIZE
registry.altlinux.org/alt/alt  p11         97c70e35e575  7 months ago  125 MB
```

12.1.1.3. Настройка сети

Podman автоматически создает сеть, однако при необходимости можно создать дополнительную:

```
$ podman network create my-network
```

Список сетей:

```
$ podman network ls
NETWORK ID    NAME        DRIVER
fcb565985cc6  my-network  bridge
2f259bab93aa  podman      bridge
```

Информация о сети:

```
$ podman network inspect my-network
```

12.1.2. Тестовый запуск nginx

Запуск контейнера с nginx:

```
$ podman run -d --name nginx --network my-network -p 8081:80
registry.altlinux.org/p11/nginx
9f65b8ed6971b34f65a586800852ce695ac61176a54f318d610dacae0e5d6ce6
```

где:

- **-d** — запуск в фоновом режиме;
- **--name** — имя контейнера;
- **--network** — используемая сеть;
- **-p** — сопоставление порта хоста с портом контейнера (например, 8081:80 означает: порт 8081 на хосте → порт 80 в контейнере);
- **registry.altlinux.org/p11/nginx** — адрес образа.

Список запущенных контейнеров:

```
$ podman ps
CONTAINER ID  IMAGE                 COMMAND               CREATED
STATUS        PORTS                   NAMES
9f65b8ed6971  registry.altlinux…  nginx -g daemon o...  7 minutes ago  Up 7
minutes  0.0.0.0:8081->80/tcp  nginx
```

Проверить работу nginx можно, выполнив запрос (сервер должен вернуть код 200):

```
$ curl -I <ip_адрес>:<порт>
```

где:

- <ip адрес> — IP-адрес узла, на котором запущен контейнер;
- <порт> — порт хоста, указанный через **-p** при создании контейнера.

В данном случае возможна команда:

```
$ curl -I 192.168.0.102:8081
HTTP/1.1 200 OK
Server: nginx/1.28.1
Date: Tue, 17 Feb 2026 11:23:03 GMT
…
```

12.1.3. Работа с подами

Создание пода:

```
$ podman pod create --name my-pod --network my-network -p 8081:80
```

Список подов:

```
$ podman pod ls
POD ID        NAME        STATUS      CREATED         INFRA ID      # OF
CONTAINERS
394308217ce0  my-pod      Created     12 seconds ago  286d0da0f2dc  1
```

Добавление контейнера в под:

```
$ podman run -d --pod my-pod --name nginx registry.altlinux.org/p11/nginx
```

Контейнер с приложением **nginx** будет добавлен в под **`my-pod`**.

Подробная информация о поде:

```
$ podman pod inspect my-pod
```

Список контейнеров пода:

```
$ podman pod ps --ctr-names
POD ID        NAME        STATUS      CREATED             INFRA ID      NAMES
394308217ce0  my-pod      Running     About a minute ago  bd7dd34e327f
394308217ce0-infra,nginx
```

Управление подом:

- остановить группу контейнеров:

  ```
  $ podman pod stop my-pod
  ```

- запустить группу контейнеров:

  ```
  $ podman pod start my-pod
  ```

- удалить группу контейнеров:

  ```
  $ podman pod rm my-pod
  ```

12.2. Buildah: специфичная сборка контейнерных образов

Buildah — это инструмент командной строки из экосистемы **Podman**, предназначенный для создания, модификации и управления OCI-совместимыми образами.

В отличие от **`docker build`**, **Buildah**:

- не требует запущенного демона;
- работает от имени обычного пользователя (rootless);
- предоставляет низкоуровневый контроль над каждым слоем образа;
- поддерживает как сборку из Dockerfile, так и скриптовую (пошаговую) сборку.

Buildah подходит для использования в CI/CD, безопасной сборки в изолированных средах и создания минимальных образов.

12.2.1. Установка

Установка пакета:

```
# apt-get install buildah
```

12.2.2. Основные команды

Базовые концепции:

- контейнер (container) — изменяемая файловая система, используемая в процессе сборки;
- образ (image) — неизменяемый результат сборки.

Команды buildah **`from`**, **`run`**, **`copy`**, **`commit`** позволяют собирать образ пошагово.

Таблица 12.2. Основные команды для работы с buildah

Команда	Назначение
`buildah from <image>`	Создать рабочий контейнер из образа
`buildah run <ctr> -- <cmd>`	Выполнить команду внутри контейнера
`buildah copy <ctr> src dst`	Скопировать файлы в контейнер
`buildah config <ctr>`	Задать метаданные (ENTRYPOINT, CMD, PORT и др.)
`buildah commit <ctr> <tag>`	Сохранить контейнер как образ
`buildah bud -f Dockerfile -t tag .`	Собрать из Dockerfile (bud = Build Using Dockerfile)
`buildah images`	Список локальных образов
`buildah containers`	Список рабочих контейнеров

12.2.3. Сборка из Dockerfile

Самый простой способ собрать образ — использовать существующий Dockerfile.

Пример **`Dockerfile`**:

```
FROM alt:alt

LABEL maintainer="admin@test.alt"
LABEL description="Demo image for buildah examples"

ENV APP_DIR=/app
WORKDIR $APP_DIR

# Установка пакетов
RUN apt-get update && \
    apt-get install -y curl bash && \
    apt-get clean

# Копирование файлов в контейнер
```

```
COPY hello.sh .

# Сделать скрипт исполняемым
RUN chmod +x hello.sh

# Порт (для примера)
EXPOSE 8090

# Команда по умолчанию
CMD ["./hello.sh"]
```

Файл **`hello.sh`**:

```
#!/bin/sh
echo "Hello from Buildah demo container!"
echo "Container hostname: $(hostname)"
```

Сборка образа:

```
$ buildah bud -t myapp:latest .
```

Ожидаемый вывод:

```
STEP 1/10: FROM alt:latest
STEP 2/10: LABEL maintainer="admin@test.alt"
STEP 3/10: LABEL description="Demo image for buildah examples"
STEP 4/10: ENV APP_DIR=/app
STEP 5/10: WORKDIR $APP_DIR
STEP 6/10: RUN apt-get update && apt-get install -y curl bash && apt-get clean
...
STEP 7/10: COPY hello.sh .
STEP 8/10: RUN chmod +x hello.sh
STEP 9/10: EXPOSE 8090
STEP 10/10: CMD ["./hello.sh"]
COMMIT myalt
Getting image source signatures
Copying blob 7066ca05e439 skipped: already exists
Copying blob fb9fc20ce189 skipped: already exists
Copying blob 475224c3ec5c done   |
Copying config 5674dd2fc2 done   |
Writing manifest to image destination
--> 5674dd2fc28a
Successfully tagged localhost/myalt:latest
5674dd2fc28ac10beb78fe9ee78ffc6788f731c1560c4dd2a49edace7bacc68d
```

Примечание

По умолчанию команда **`buildah bud`** использует файл **`Dockerfile`** в текущем каталоге. Если используется файл с другим именем или расположением, необходимо указать его с помощью параметра ***`-f`***, при этом также требуется указать контекст сборки (обычно текущий каталог **`.`**).

Примечание

Ошибка **no space left on device** при работе с контейнерами часто связана с переполнением каталога **/tmp**, используемого для хранения временных файлов.

В дистрибутивах ALT каталог **/tmp** по умолчанию размещается в файловой системе tmpfs. Его размер определяется автоматически (если параметр ***size*** не задан) и обычно составляет до половины объёма оперативной памяти.

Возможные решения:

- увеличить объём доступной памяти за счёт подключения или расширения раздела подкачки (swap);
- явно задать размер tmpfs в файле **/etc/fstab**:

```
tmpfs           /tmp                    tmpfs   nosuid,size=4G
0 0
```

После внесения изменений необходимо перемонтировать файловую систему:

```
# mount -o remount /tmp
```

Альтернативный вариант — использовать другой каталог для временных файлов, например **/var/tmp**.

12.2.4. Ручная (скриптовая) сборка

Для максимального контроля процесс сборки образа можно выполнить вручную:

1. Создать базовый контейнер:

```
$ ctr=$(buildah from alt:latest)
```

2. Выполнить команды внутри контейнера:

```
$ buildah run "$ctr" -- sh -c 'apt-get update && apt-get install -y curl
bash && apt-get clean'
$ buildah copy "$ctr" hello.sh /app/hello.sh
```

3. Настроить метаданные образа:

```
$ buildah config --cmd '["/app/hello.sh"]' "$ctr"
```

4. Зафиксировать изменения в виде образа:

```
$ buildah commit "$ctr" myalt
```

5. Удалить контейнер:

```
$ buildah rm "$ctr"
```

Пример создания минимального образа:

```
#!/bin/bash
ctr=$(buildah from scratch)
buildah copy $ctr ./my-binary /app
buildah config --entrypoint '["/app"]' $ctr
buildah commit $ctr minapp:slim
buildah rm $ctr
```

Получается ультраминимальный образ (только бинарник), без OS-слоёв.

Образ scratch не содержит операционной системы, libc или shell и подходит только для статически скомпонованных бинарных файлов.

12.2.5. Отладка и инспекция

Просмотр информации об образе:

```
$ buildah inspect myalt:latest
```

Запуск временного контейнера для проверки:

```
$ podman run --rm -it myalt:latest sh
```

12.3. ALTLinux Container Registry

ALTLinux Container Registry — это публичный реестр контейнеров «Альт», предназначенный для хранения образов в формате OCI. В реестре публикуются базовые и служебные образы от членов ALT Linux Team, а также образы с популярным прикладным программным обеспечением. Все образы автоматически проверяются на наличие уязвимостей.

Реестр расположен по адресу: registry.altlinux.org

Репозитории группируются по веткам (бранчам):

- p10/ — образы на базе десятой платформы;
- p11/ — образы на базе одиннадцатой платформы;
- sisyphus/ — образы на базе репозитория Сизиф;
- c10f/ — образы на базе репозитория для дистрибутива «Альт СП».

В реестре представлены образы контейнеров различных типов:

- alt — минимальные образы;
- base — базовые образы;
- buildpack-deps — настроенная базовая среда для сборки;
- distroless — минимальные образы для запуска приложений;
- php, python, golang, ruby — образы с языками программирования и инструментами разработки;

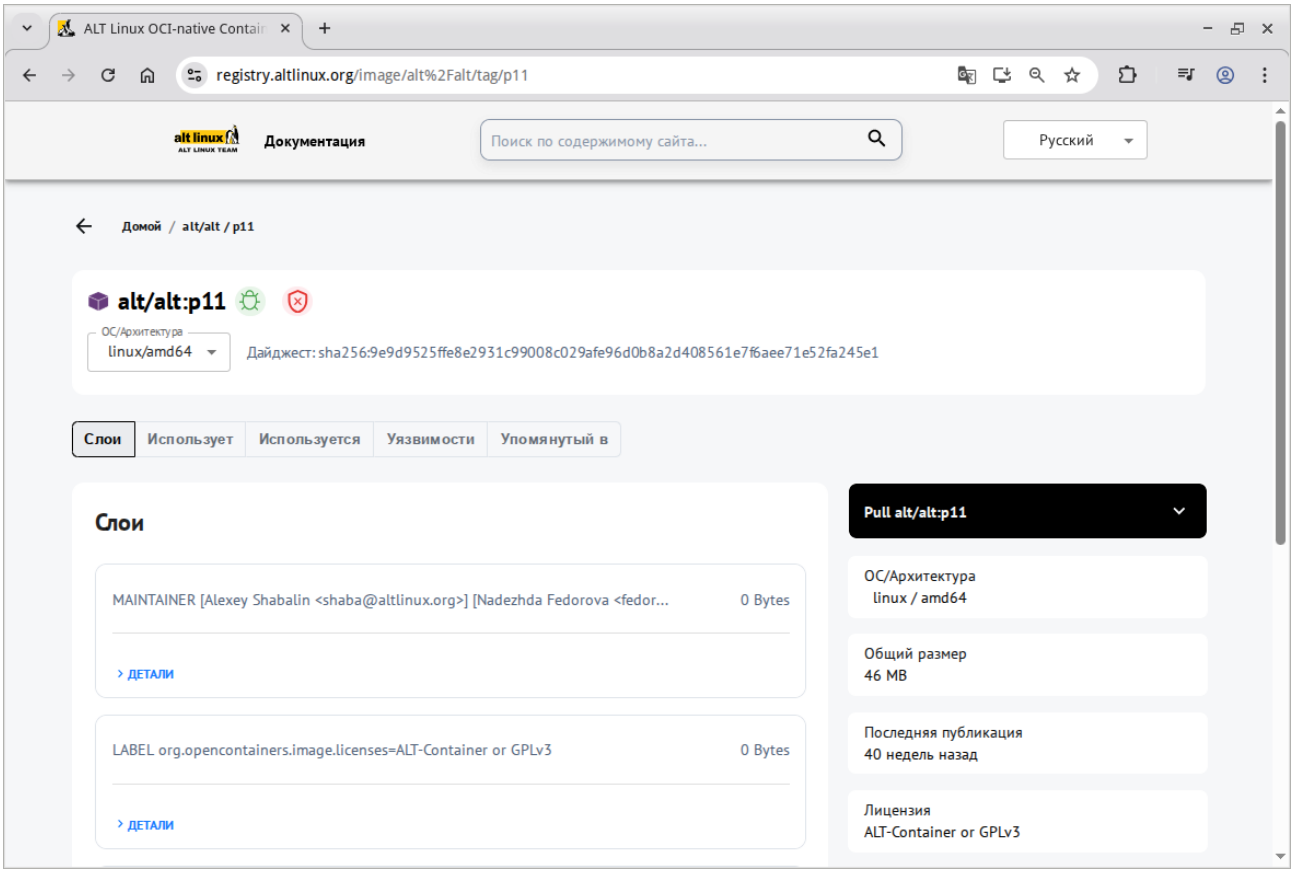
- gitea, grafana и др. — образы прикладных сервисов.

Образы версионируются с помощью тегов, в которых указываются версии приложений или дата сборки. Например:

- p11/alt:latest — последняя стабильная версия минимального образа на базе одиннадцатой платформы;
- sisyphus/python:3.13.11 — образ с интерпретатором Python указанной версии.

Для каждого образа доступны метаданные: размер, версии пакетов, дата публикации и др.

Доступ к реестру осуществляется через стандартные инструменты (Podman, Docker CLI), а также через веб-интерфейс.



В репозитории alt публикуются базовые образы. Они собираются для архитектур: amd64, arm, arm64, x86_64, и для веток: sisyphus, p10, p11.

Основные образы (вместо <branch> указывается ветка репозитория):

- минимальный образ: registry.altlinux.org/<branch>/alt;
- базовый образ с локалями и часовыми поясами: registry.altlinux.org/<branch>/base;
- веб-серверы: registry.altlinux.org/<branch>/nginx, registry.altlinux.org/<branch>/unit и registry.altlinux.org/<branch>/apache2;
- системы хранения конфигурации: registry.altlinux.org/<branch>/etcd;
- интерпретаторы для пользовательских приложений: registry.altlinux.org/<branch>/python и registry.altlinux.org/<branch>/ruby.

Также публикуются distroless-образы с префиксом distroless.

Distroless-образы — это образы, содержащие минимально необходимый набор исполняемых файлов и библиотек для запуска приложений. Они не включают полноценную пакетную систему и большинство вспомогательных утилит, присутствующих в обычной ОС.

Основные distroless-образы:

- базовый образ для запуска статических приложений: registry.altlinux.org/<branch>/distroless-static. Содержит только базовую структуру каталогов. По умолчанию используется пользователь nonroot;
- базовый образ для запуска динамически слинкованных приложений: registry.altlinux.org/<branch>/distroless-base. Содержит tzdata и основные библиотеки (например, libc, ssl, selinux);
- образ для разработки и отладки: registry.altlinux.org/<branch>/distroless-devel. Содержит дополнительные утилиты для работы с файлами и интерактивного использования. Рекомендуется использовать только на этапе разработки.

Поскольку distroless-образы не содержат средств для установки пакетов, стандартный подход с использованием **`Dockerfile`** (с установкой пакетов через пакетный менеджер) неприменим. Такие образы собираются с использованием специализированных инструментов, например: https://github.com/alt-cloud/image-forge#distroless-images

12.4. Zot: базовый функционал

Zot — легковесный высокопроизводительный OCI-совместимый реестр контейнеров, предназначенный для хранения, управления и распространения контейнерных образов и связанных артефактов.

Поддерживаемые артефакты:

- OCI- и Docker-образы;
- Helm-чарты;
- SBOM (Software Bill of Materials);
- подписи и политики доверия (Notary v2, Cosign);
- метаданные образов.

Реестр может работать как в локальном, так и в распределённом режиме.

Основные возможности:

- полная совместимость с OCI Distribution Specification;
- поддержка клиентов: Docker, Podman, containerd, Helm и др.;
- поддержка подписей образов (Notary v2, Cosign);
- управление доступом (RBAC: пользователи, группы, репозитории);
- зеркалирование удалённых реестров (pull-through cache);
- аудит и логирование;

- работа в распределённой среде (S3-совместимое хранилище);
- встроенный веб-интерфейс и REST API.

12.4.1. Установка и запуск

Установка пакета:

```
# apt-get install zot
```

Основной файл конфигурации **/etc/zot/config.json**.

Пример:

```
{
    "storage":{
        "rootDirectory":"/var/lib/zot"
    },
    "http":{
        "address":"127.0.0.1",
        "port":"5000"
    },
    "log":{
        "level":"debug"
    },
    "extensions": {
        "search": {
            "enable": true,
            "cve": {
                "trivy": {
                    "dbRepository": "altlinux.space/trivy/trivy-db"
                },
                "updateInterval": "24h"
            }
        },
        "ui": {
            "enable": true
        },
        "mgmt": {
            "enable": true
        }
    }
}
```

Для возможности доступа с других хостов необходимо для параметра ***address*** задать значение 0.0.0.0 (слушать на всех сетевых интерфейсах).

Можно также указать публичный адрес реестра (параметр ***externalUrl***), например:

```
"externalUrl": "http://registry.test.alt"
```

Примечание

DNS должен резолвиться и порт должен быть доступен.

Запуск и проверка сервиса:

```
# systemctl start zot
# systemctl status zot
```

Примечание

Для отладки или тестирования можно запустить **Zot** вручную:

```
# zot serve /etc/zot/config.json
```

После запуска реестр доступен по адресу: **http://localhost:5000**.

Примечание

Если **Zot** работает по HTTP, необходимо разрешить небезопасный реестр (insecure registry):

- добавить в **/etc/containers/registries.conf**:

```
[[registry]]
location = "localhost:5000"
insecure = true
```

- или временно:

```
$ podman push --tls-verify=false localhost:5000/alt/alt:p11
```

12.4.2. Работа с образами

12.4.2.1. Загрузка образа в реестр

Список локальных образов:

```
$ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED       SIZE
registry.altlinux.org/alt/alt   p11         97c70e35e575  9 months ago  125 MB
```

Перетегирование образа:

```
$ podman tag registry.altlinux.org/alt/alt:p11 \
  localhost:5000/alt/alt:p11
```

Загрузка в реестр:

```
$ podman push localhost:5000/alt/alt:p11
```

Проверка списка репозиториев:

```
$ curl http://localhost:5000/v2/_catalog
{"repositories":["alt/alt"]}
```

Проверка тегов:

```
$ curl http://localhost:5000/v2/alt/alt/tags/list
{"name":"alt/alt","tags":["p11"]}
```

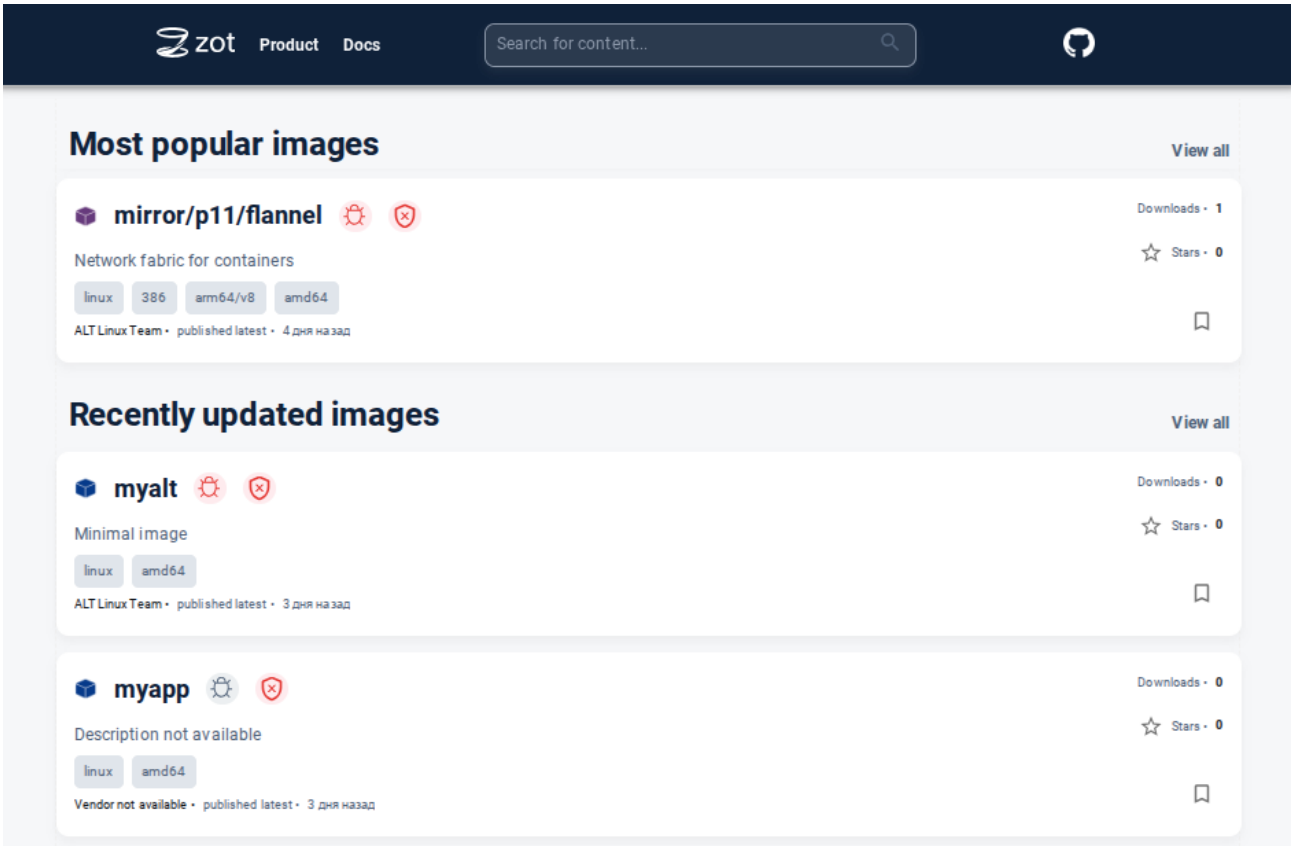
12.4.2.2. Веб-интерфейс

Встроенный веб-интерфейс доступен по адресу **`http://localhost:5000`**

В веб-интерфейсе **Zot** можно:

- просматривать репозитории и теги;
- анализировать метаданные образов;
- проверять цифровые подписи (Cosign).

В зависимости от настроек безопасности для доступа к веб-интерфейсу может потребоваться аутентификация (см. раздел Аутентификация и авторизация).



После входа отображается главная страница, на которой представлены:

- популярные образы;
- недавно обновлённые репозитории.

Все доступные образы можно просмотреть на странице **`/explore`** (доступна по ссылке **`View all`** в правом верхнем углу).

Рядом с каждым образом отображаются значки, указывающие на:

- результат сканирования на уязвимости;
- статус цифровой подписи.

Таблица 12.3. Значки и их значения

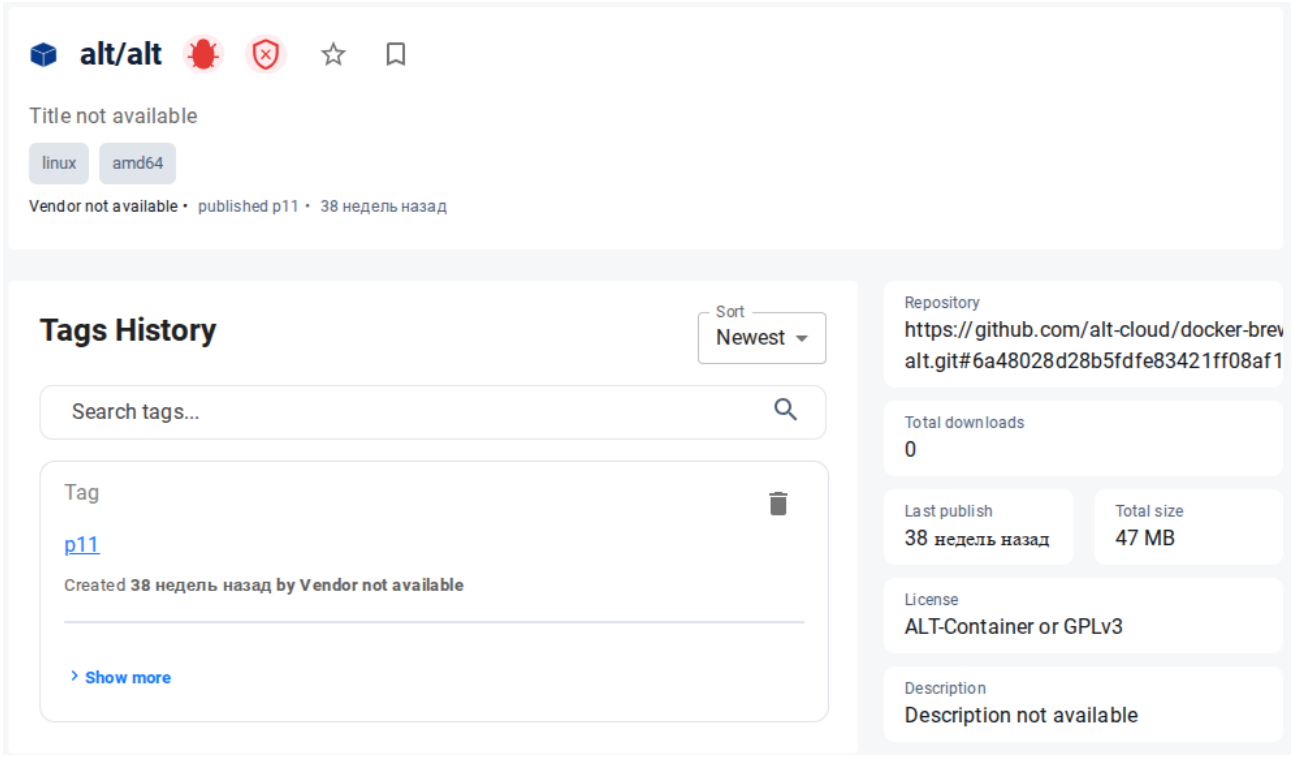
Значок	Значение
	Уязвимости не обнаружены
	Ошибка при сканировании
	Критическая уязвимость
	Высокая уязвимость
	Средняя уязвимость
	Низкая уязвимость
	Подпись проверена
	Подпись отсутствует или не проверена

Примечание

Уровни уязвимостей также отображаются на вкладке **VULNERABILITIES**.

На странице **`/explore`** доступны следующие инструменты:

- **`Поиск`** — ввод имени образа или ключевых слов;
- **`Сортировка`**:
  - по релевантности;
  - по дате обновления (сначала новые);
  - по алфавиту;
  - по количеству загрузок;
- **`Фильтры`** (фасетная навигация):
  - операционная система;
  - архитектура CPU (amd64, arm64 и др.);
  - статус подписи (подписан / не подписан).



При выборе образа открывается страница, содержащая:

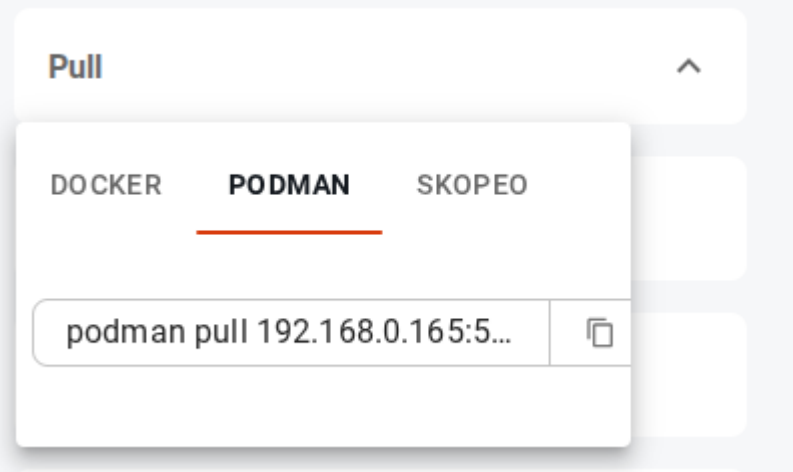
- описание образа;
- URL репозитория;
- количество загрузок;
- дату последней публикации;
- размер;
- тип лицензии;
- список доступных тегов.

На странице конкретного тега доступны вкладки:

- **LAYERS** — слои образа (команды сборки и дайджесты);
- **USES** — используемые базовые образы;
- **USED BY** — образы, использующие данный образ;
- **VULNERABILITIES** — известные уязвимости (CVE).

Примечание

На вкладке **LAYERS** можно нажать **`Details`**, чтобы увидеть команды сборки и хеши слоёв.



Чтобы загрузить образ (pull):

1. На странице тега откройте меню **Pull**.
2. Выберите инструмент:
    - Docker
    - Podman
    - Skopeo
3. Нажмите значок **Копировать** рядом с командой.
4. Вставьте команду в терминал и выполните её:

```
$ podman pull 192.168.0.165:5000/alt/alt:p11
```

12.4.2.3. Командная утилита zli

zli (Zot CLI) — утилита командной строки для работы с реестром Zot.

Утилита позволяет:

- управлять конфигурацией;
- просматривать образы;
- выполнять поиск;
- анализировать уязвимости.

Добавление реестра (задание алиаса для URL сервера):

```
$ zli config add local http://localhost:5000
$ zli config add remote http://registry.test.alt:5000
```

Вывод списка настроенных реестров:

```
$ zli config -l
local   http://localhost:5000
remote  http://registry.test.alt:5000
```

Работа с образами:

- список образов:

```
$ zli image list --config local
```

Пример вывода:

```
REPOSITORY TAG      OS/ARCH          DIGEST     SIGNED   SIZE
myalt      latest   linux/amd64      fdd06a93   false    258MB
myalt      stable   linux/amd64      fdd06a93   false    258MB
myalt      v1.2.3   linux/amd64      fdd06a93   false    258MB
myapp      latest   linux/amd64      fd6a158a   false    6.6MB
```

- информация об образе:

```
$ zli image name myalt:latest --config local
```

Пример вывода:

```
REPOSITORY TAG      OS/ARCH          DIGEST     SIGNED   SIZE
myalt      latest   linux/amd64      fdd06a93   false    258MB
```

- список репозиториев:

```
$ zli repo list --config local
```

Пример вывода:

```
Searching... 
REPOSITORY NAME
alt
alt/alt
alt-modified
busybox
myalt
myapp
```

Zot интегрируется с базами уязвимостей, что позволяет анализировать образы на наличие известных CVE:

- найти образы, затронутые конкретной уязвимостью:

```
$ zli cve affected CVE-2025-32989 --config remote
```

- список уязвимостей образа:

```
$ zli cve list alt/alt:p11 --config remote
```

- фильтрация по ID уязвимости:

```
$ zli cve list alt/alt:p11 --config remote --cve-id CVE-2025-32989
```

- подробный вывод (с описанием и пакетами):

```
$ zli cve list alt/alt:p11 --config remote --verbose
```

- вывод в формате JSON:

```
$ zli cve list alt/alt:p11 --config remote -f json
```

- найти все образы на конкретном сервере Zot, затронутые конкретной уязвимостью CVE:

```
$ zli cve affected CVE-2025-32989 --config remote --repo c3/openjdk-dev
```

- сравнение уязвимостей между версиями:

```
$ zli cve diff c3/openjdk-dev:1.0.0 c3/openjdk-dev:2.0.0 --config remote
```

Примечание

Команда покажет уязвимости, присутствующие в версии 1.0.0, но исправленные в версии 2.0.0.

- поиск исправлений:

```
$ zli cve fixed alt/alt CVE-2025-32989 --config remote
```

Примечание

Для multi-arch образов можно указывать платформу:

```
$ zli cve diff myapp:v1.0 linux/amd64 myapp:v2.0 linux/arm64 --config
remote
```

Запуск встроенного бенчмарка:

```
# zb -c 10 -s 127.0.10.0/24 -n 10 http://localhost:5000
```

Примечание

Бенчмарк позволяет оценить производительность реестра (количество запросов, задержки, нагрузку).

Команда **`search`** позволяет находить репозитории и теги по частичному совпадению:

- Поиск по подстроке:

```
$ zli search query ng --config local
```

Команда найдёт nginx, mongo, golang и др.

- Если имя завершается двоеточием — поиск строго по имени:

```
zli search query nginx: --config local
```

Команда вернёт только репозиторий nginx.

12.4.2.4. Загрузка образа из реестра

Загрузка образа:

```
$ podman pull localhost:5000/alt/alt:p11
```

12.4.3. Аутентификация и авторизация

Zot поддерживает гибкие механизмы аутентификации и тонкую настройку прав доступа (авторизации) к репозиториям.

12.4.3.1. LDAP

Интеграция с LDAP/Active Directory позволяет использовать централизованные учётные записи.

Пример конфигурации (**`config.json`**):

```
"http": {
    …
    "auth": {
        "ldap": {
            "insecure": true,
            "address": "dc1.test.alt",
            "port": 389,
            "startTLS": false,
            "baseDN": "OU=OU,DC=test,DC=alt",
            "userAttribute": "sAMAccountName",
            "userGroupAttribute": "memberOf",
            "skipVerify": false,
            "credentialsFile": "/etc/zot/ldap-creds.json"
        }
    }
}
```

где:

- ***`insecure`*** — разрешает подключение к LDAP-серверу без использования TLS/SSL. Установите значение true, если сервер не поддерживает шифрование (рекомендуется только для тестовых сред);
- ***`address`*** — IP-адрес или доменное имя LDAP-сервера;
- ***`port`*** — порт службы LDAP (обычно 389 для незашифрованного соединения, 636 для LDAPS);
- ***`startTLS`*** — включает шифрование соединения с помощью StartTLS. Установите true, если сервер поддерживает StartTLS поверх порта 389;
- ***`baseDN`*** — базовое Distinguished Name (DN), с которого начинается поиск пользователей в каталоге;

- ***userAttribute*** — имя атрибута, содержащего логин пользователя (для AD обычно используется sAMAccountName);
- ***userGroupAttribute*** — имя атрибута, содержащего информацию о группах, в которые входит пользователь (в AD это memberOf);
- ***skipVerify*** — пропускает проверку сертификата сервера при использовании TLS/StartTLS (не рекомендуется в production);
- ***credentialsFile*** — путь к файлу с учётными данными для привязки (bind) к LDAP-серверу.

Файл учётных данных (**/etc/zot/ldap-creds.json**):

```
{
  "bindDN": "cn=administrator_zot@test.alt",
  "bindPassword": "P@$$word"
}
```

где:

- ***bindDN*** — учётная запись, используемая для поиска пользователей в каталоге;
- ***bindPassword*** — пароль для учётной записи.

12.4.3.2. HTTP Basic Auth (htpasswd)

Простейший способ хранения учётных данных — локальный файл с учётными данными в формате htpasswd (хеши bcrypt).

Создание файла пользователей:

```
# htpasswd -Bbn alice mypass > /etc/zot/htpasswd
```

Конфигурация:

```
{
  "http": {
    …
    "auth": {
      "htpasswd": {
        "path": "/etc/zot/htpasswd"
      }
    }
  }
}
```

Примечание

Если пользователь существует в LDAP, используется только LDAP. Если пользователя нет в LDAP, но он есть в htpasswd, вход возможен через htpasswd — это позволяет иметь гибридную схему (например, сервисные учётные записи в файле, пользователи — в AD).

12.4.3.3. Авторизация

При использовании схемы доступа, основанной только на аутентификации, любой прошедший проверку пользователь получает полный доступ к реестру. Для более гибкого управления **Zot** поддерживает политики авторизации на уровне репозиториев, основанные на идентичности пользователя или группы. Правила доступа определяются в зависимости от репозитория, пользователя и выполняемого действия.

Правила доступа определяются в секции ***`accessControl`*** по принципу:

```
репозиторий → пользователь/группа → разрешённые действия (read, create, update, delete)
```

Поддерживаются пять типов политик, основанных на идентичности.

Таблица 12.4. Политики контроля доступа

Тип политики	Атрибут	Описание
По умолчанию	defaultPolicy	Действия, разрешённые аутентифицированному пользователю при отсутствии явных правил
Для конкретного пользователя	users, actions	Права для конкретных пользователей
Для группы	groups, actions	Права для групп
Анонимная	anonymousPolicy	Доступ для неаутентифицированных пользователей
Метрики	metrics	Доступ к endpoint метрик
Администраторская	adminPolicy	Полный доступ ко всем репозиториям

Пример конфигурации:

```
{
  "http": {
    …
    "accessControl": {
        "repositories": {
        "**": {
            "defaultPolicy": ["read"],
            "policies": [{
            "users": ["alice"],
            "actions": ["read", "create"]
            }]
        },
        "secure/*": {
            "policies": [{
            "groups": ["admins"],
            "actions": ["read", "create", "delete"]
            }],
            "defaultPolicy": ["read"]
        }
        },
        "adminPolicy": {
        "users": ["admin"],
        "actions": ["read", "create", "update", "delete"]
```

```
        }
    }
  }
}
```

Пример предоставления прав группе LDAP:

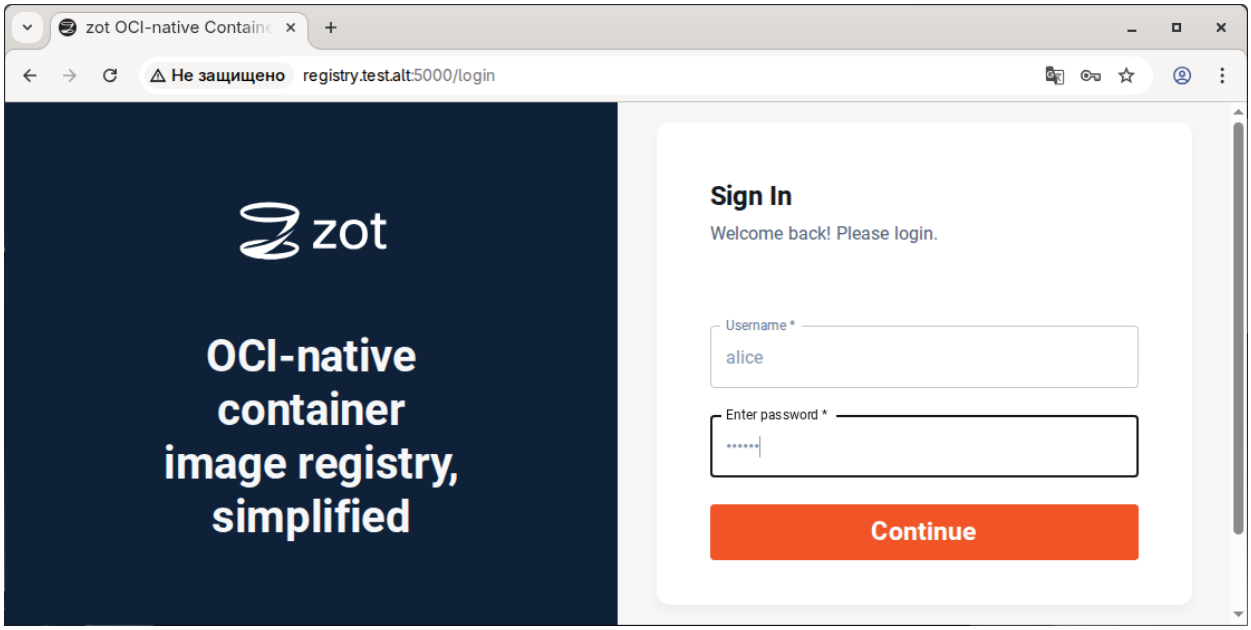
```
{
  "accessControl": {
    "repositories": {
      "myalt": {
         "policies": [{
                     "groups": ["CN=DevOps,OU=OU,DC=test,DC=alt"],
                     "actions": ["read", "create", "update"]
                     }]
           }
        }
   }
}
```

Важно

Для выполнения операций create, update и delete обязательно должно быть разрешено действие read.

Учётные данные для аутентификации могут быть указаны:

- в веб-интерфейсе:



- командной строке:

```
$ zli image list --config local --user alice:mypass
$ zli image list --config local --user 'mun:P@$$word'
```

Примечание

При использовании CLI необходимо экранировать специальные символы в пароле (например, $, !, &) или заключать строку в кавычки.

12.4.4. Зеркалирование OCI-реестров с помощью Zot

Zot поддерживает зеркалирование одного или нескольких внешних OCI-совместимых реестров. Это позволяет создавать локальные копии образов для ускорения сборки, снижения сетевых затрат и работы в изолированных (air-gapped) средах.

Важно

По умолчанию **Zot** работает как чистый OCI-реестр. При загрузке образов в формате Docker они автоматически конвертируются в OCI. В результате:

- хеш образа (digest) изменяется;
- подписи (Cosign, Notary) становятся недействительными;
- некоторые не-OCI атрибуты могут быть утеряны.

Чтобы сохранить подписи, отключите конвертацию:

```
{
  "http": { "compat": true },
  "extensions": { "sync": { "preserveDigest": false } }
}
```

В этом случае образы остаются в исходном формате, и подписи работают корректно.

Таблица 12.5. Режимы зеркалирования

Режим	Описание	Когда использовать
Полное зеркало (periodic sync)	Zot периодически опрашивает внешний реестр и заранее кеширует указанные образы	Для критически важных образов, когда требуется гарантированная доступность без задержек
Кеш по запросу (pull-through / on-demand)	Образ загружается из внешнего реестра только при первом запросе и кешируется локально. Последующие запросы обслуживаются из кеша	Для экономии места и при работе с большими реестрами (например, Docker Hub)

Примечание

Особенность Docker Hub: из-за ограничений на частоту запросов и отсутствия поддержки каталога (_catalog) рекомендуется использовать только режим on-demand. Периодический опрос (pollInterval) не рекомендуется.

Функция синхронизации (***`sync`***) настраивается в секции ***`extensions`*** файла **`config.json`**.

Пример конфигурации:

```
{
    "extensions": {
        "sync": {
            "credentialsFile": "/etc/zot/sync-auth.json",
            "registries": [
            {
                "urls": ["https://registry.altlinux.org"],
                "onDemand": true,
                "pollInterval": "12h",
                "tlsVerify": true,
                "maxRetries": 3,
                "retryDelay": "5m",
                "onlySigned": false,
                "preserveDigest": false,
                "content": [
                {
                    "prefix": "myapp/**",
                    "destination": "mirror/myapp",
                    "stripPrefix": true
                }
                ]
            }
            ]
        }
    }
}
```

Таблица 12.6. Описание параметров

Параметр	Описание
Уровень *sync*	
credentialsFile	Путь к файлу с учётными данными для внешних реестров
Уровень *registries[]*	
urls	Список URL внешних реестров (резервные адреса в порядке приоритета)
onDemand	true — включить кеширование по запросу; false — использовать только предварительно синхронизированные образы
pollInterval	Интервал полного опроса (например, "6h"). Если не задан — используется только on-demand
tlsVerify	Проверка TLS-сертификатов (true по умолчанию)
certDir	Каталог с доверенными сертификатами
maxRetries, ***retryDelay***	Параметры повторных попыток при ошибках
syncTimeout	Таймаут операции синхронизации (по умолчанию — 3 часа)
onlySigned	Синхронизировать только подписанные образы (Cosign/Notary)
preserveDigest	true – попытка сохранить digest (не гарантирует работу подписей); false — конвертировать в OCI (рекомендуется для совместимости)
Уровень *content[]*	
urls	Список URL внешних реестров (резервные адреса в порядке приоритета)

Параметр	Описание
prefix	Путь во внешнем реестре (поддерживает glob: *, **)
tags.regex	Фильтр тегов по регулярному выражению
tags.semver	Фильтр по semantic versioning
destination	Локальный путь для хранения образа
stripPrefix	Удалять префикс из prefix при сохранении (true → /repo/app → app)

Файл, указанный в ***credentialsFile***, содержит учётные данные:

```
{
    "registry.altlinux.org": {
        "username": "zot-sync",
        "password": "secret"
    },
    "gcr.io": {
        "username": "_json_key",
        "password": "{ ... }"
    }
}
```

Примеры использования:

- полное зеркало критических образов:

  ```
  {
      "onDemand": false,
      "pollInterval": "24h",
      "content": [{ "prefix": "infra/nginx" }]
  }
  ```

  Ежедневно синхронизирует только стабильные версии nginx.

- кеш по запросу для registry.altlinux.org:

  ```
  {
      "urls": ["https://registry.altlinux.org"],
      "onDemand": true,
      "content": [
      {
          "prefix": "p11/**",
          "destination": "mirror/p11",
          "stripPrefix": true
      }]
  }
  ```

  Образы загружаются только при первом обращении, например:

  ```
  $ podman pull 192.168.0.165:5000/mirror/p11/flannel:latest
  ```

- миграция реестра (перенос всего содержимого):

```
{
    "pollInterval": "12h",
    "onDemand": true,
    "content": [{ "prefix": "**" }]
}
```

После полной синхронизации можно перенаправить трафик на новый Zot.

12.5. regclient: работа с реестрами образов

regclient — это набор утилит командной строки для взаимодействия с OCI-совместимыми реестрами (Docker Hub, Harbor Zot, Quay, AWS ECR и др.). Он позволяет:

- просматривать метаданные образов и тегов;
- копировать, фильтровать и преобразовывать образы;
- проверять подписи и SBOM;
- работать с манифестами, слоями и артефактами.

Инструмент поддерживает аутентификацию, TLS, прокси и работу с частными реестрами.

12.5.1. Установка

Установка пакета:

```
# apt-get install regclient
```

12.5.2. Основные команды

Основная утилита — **`regctl`**. Также доступны **`regbot`** и **`regsync`** для автоматизации.

Таблица 12.7. Основные команды для работы с regclient

Команда	Назначение
Работа с тегами	
`regctl tag ls <repo>`	Список тегов в репозитории
`regctl tag delete <image>`	Удалить тег из репозитория
Инспекция образов	
`regctl image inspect <image>`	Показать конфигурацию и метаданные образа
`regctl manifest get <image>`	Получить манифест образа (включая multi-arch index)
`regctl image manifest <image>`	То же, что **`manifest get`**
`regctl image ratelimit <image>`	Проверить лимиты запросов к реестру (через HEAD-запрос)
Модификация образов	
`regctl image mod <image> [флаги]`	Изменить образ: метки, аннотации, слои, время создания и др.
`regctl image create <image>`	Создать новый образ из пустого состояния (scratch)

Команда	Назначение
Импорт/экспорт	
`regctl image export <image> file.tar`	Экспортировать образ в tar (OCI или Docker format)
`regctl image import <image> file.tar`	Импортировать образ из tar (docker save или OCI Layout)
Копирование и управление	
`regctl image copy <src> <dst>`	Копировать или перетегировать образ
`regctl image delete <image>`	Удалить ссылку на образ (аналог **`crane delete`**)
Работа с артефактами и SBOM	
`regctl artifact put <image> --artifact-type …`	Загрузить артефакт (SBOM, подпись и др.)
`regctl artifact list <image>`	Список привязанных артефактов (SBOM, Notary, Cosign)
Управление репозиториями	
`regctl repo ls <registry>`	Список репозиториев
Аутентификация и конфигурация	
`regctl registry login <host>`	Войти в реестр (сохраняет учётные данные в ~/.docker/config.json)
`regctl registry logout <host>`	Выйти из реестра
`regctl registry config`	Показать текущую конфигурацию
Служебные команды	
`regctl version`	Показать версию
`regctl blob get <image> <digest>`	Скачать слой (blob) по дайджесту
`regctl blob put <repo>`	Загрузить blob в репозиторий

Для получения подробной информации о команде можно использовать команду:

```
$ regctl <команда> --help
```

12.5.3. Настройка доступа

Настройка реестра для работы по HTTP:

```
$ regctl registry set --tls disabled localhost:5000
```

Это позволяет regctl принимать HTTP-ответы от сервера Zot. Если на сервере Zot включена TLS-аутентификация, эту настройку выполнять не требуется.

Для доступа к частным реестрам:

- вход через логин/пароль:

```
$ regctl registry login registry.test.alt:5000 -u username -p password
```

- через токен:

```
$ regctl registry login registry.test.alt -u username --token
```

Учётные данные сохраняются в **~/.regctl/config.json**.

Показать конфигурацию:

```
$ regctl registry config
{
  "hosts": {
    "localhost:5000": {
      "tls": "disabled",
      "hostname": "localhost:5000",
      "reqConcurrent": 3
    },
    "registry.test.alt:5000": {
      "tls": "disabled",
      "hostname": "registry.test.alt:5000",
      "user": "alice",
      "reqConcurrent": 3
    }
  }
}
```

12.5.4. Работа с regclient

12.5.4.1. Просмотр информации об образах

Список всех репозиториев:

```
$ regctl repo ls registry.test.alt:5000
myapp
```

Список тегов в репозитории:

```
$ regctl tag ls registry.test.alt:5000/myapp
latest
```

Метаданные образа:

```
$ regctl image inspect registry.test.alt:5000/myapp:latest
```

Вывод включает OCI-манифест: архитектуру, ОС, слои, размер, аннотации.

12.5.4.2. Работа с образами

Экспорт образа в tar (без Docker):

```
$ regctl image export registry.test.alt:5000/myapp:latest myapp.tar
```

Копирование образа между реестрами:

```
$ regctl image copy \
  docker.io/library/alpine:latest \
  registry.test.alt:5000/alpine:latest
```

12.5.4.3. Работа с манифестами

Получить манифест:

```
$ regctl manifest get registry.test.alt:5000/myapp
```

Отправить манифест:

```
$ regctl manifest put registry.test.alt:5000/myapp:1.0.0 \
  --format oci \
  --content-type application/vnd.oci.image.manifest.v1+json
```

Фильтрация по платформе:

```
$ regctl manifest get registry.test.alt:5000/myapp:latest --platform linux/arm64
```

12.5.4.4. Работа с артефактами

Просмотр SBOM (если загружен как OCI-артефакт):

```
$ regctl artifact list registry.test.alt:5000/myapp:latest
```

12.5.4.5. Модификация образов

Модификация образа:

- добавить аннотацию ко всем платформам:

```
$ regctl image mod registry.test.alt:5000/myapp:latest \
  --replace \
  --annotation "[*]org.opencontainers.image.created=2026-03-30T05:06:07Z"
```

- изменить entrypoint:

```
$ regctl image mod registry.test.alt:5000/myapp:latest \
  --create v1-bash \
  --config-entrypoint '["bash"]' \
  --config-cmd ""
```

Примечание

Флаг **`--replace`** перезаписывает существующий тег!

12.6. crane: модификация и управление OCI/Docker-образами

crane — это утилита командной строки из проекта go-containerregistry, предназначенная для работы с OCI- и Docker-совместимыми образами без необходимости запускать Docker или containerd. Она позволяет:

- просматривать метаданные образов;

- копировать, переименовывать и объединять образы;
- изменять слои, теги, аннотации и другие атрибуты;
- работать напрямую с реестрами (Docker Hub, Harbor, Zot и др.).

crane особенно полезен в CI/CD, скриптах автоматизации и средах без Docker.

12.6.1. Установка

Установка пакета:

```
# apt-get install go-containerregistry-crane
```

12.6.2. Основные команды

Таблица 12.8. Основные команды crane

Команда	Назначение
Работа с образами	
pull	Загрузить образ из реестра в локальное хранилище
push	Отправить локальный образ в удалённый реестр
copy	Эффективно скопировать образ между реестрами с сохранением дайджеста
tag	Создать новый тег для существующего образа в реестре
delete	Удалить ссылку на образ из реестра
export	Экспортировать файловую систему образа в tar-архив
flatten	Объединить все слои образа в один
rebase	Пересобрать образ на новой базовой основе
Инспекция и метаданные	
config	Получить конфигурацию образа (в формате JSON)
manifest	Показать манифест образа
digest	Вывести дайджест (хеш) образа
ls	Список тегов в репозитории
catalog	Список всех репозиториев в реестре
Модификация	
mutate	Изменить метки (labels) и аннотации образа
index	Редактировать манифест multi-arch образа (image index)
Работа со слоями и данными	
append	Экспортировать образ в tar (OCI или Docker format)
blob	Прочитать необработанный blob из реестра по дайджесту
Аутентификация и служебные	
auth	Управление учётными данными (логин/доступ к реестру)
validate	Проверить корректность структуры образа
version	Показать версию утилиты
completion	Сгенерировать скрипт автодополнения для оболочки
help	Справка по командам

Для получения подробной информации о команде можно использовать команду:

```
$ crane <команда> --help
```

12.6.3. Работа с crane

12.6.3.1. Работа с образами

Копирование OCI-образа в частный реестр:

```
$ crane copy --insecure alt:latest registry.test.alt:5000/alt:prod
$ crane copy alt:latest localhost:5000/alt:prod
```

Отправка (push) последней версии образа alt в локальный реестр:

```
$ crane --insecure push \
   oci/images/alpine:latest \
   localhost:5000/alpine:latest
```

Получение (pull) OCI-образа:

```
$ crane --insecure pull \
   --format oci \
   localhost:5000/alpine:latest \
   oci/images/alpine:latest
```

Просмотр конфигурации образа:

```
$ crane config alt:p11 | jq .
```

Список репозиториев и тегов:

```
$ crane catalog 192.168.0.165:5000
$ crane ls localhost:5000/myapp
```

Получение дайджеста образа:

```
$ crane digest localhost:5000/myapp:latest
```

12.6.3.2. Модификация образов

crane не редактирует образ «на лету», но позволяет пересоздать его с изменениями с помощью команды **`crane mutate`**.

Примечание

Все изменения производятся иммутабельно: создаётся новый образ с новым дайджестом.

Добавить метку (label):

```
$ crane mutate alt:latest \
  --label version=1.0 \
  --tag localhost:5000/alt:modified
```

Добавить аннотацию (OCI annotations):

```
$ crane mutate myalt:latest \
  --annotation build-system=crane \
  --tag localhost:5000/my-alt:annotated
```

Изменить entrypoint и рабочий каталог:

```
$ crane mutate app:old \
  --entrypoint '["/init"]' \
  --workdir /app \
  --tag app:new
```

12.6.3.3. Работа со слоями

Для глубокой модификации файловой системы рекомендуется использовать связку:

```
crane export → редактирование → crane append
```

Пример: добавить в образ alt:latest пользовательский скрипт **`/usr/local/bin/hello.sh`** (без пересборки через **`Dockerfile`**):

1. Экспортировать файловую систему образа в tar-архив:

   ```
   $ crane export alt:latest alt-fs.tar
   ```

   Создаётся архив **`alt-fs.tar`**, содержащий корневую ФС образа.

2. Создать рабочий каталог и распаковать архив:

   ```
   $ mkdir -p modified-root
   $ tar -xf alt-fs.tar -C modified-root
   ```

3. Добавить скрипт:

   ```
   $ cat > modified-root/usr/local/bin/hello.sh <<'EOF'
   > #!/bin/sh
   > echo "Hello from modified ALT!"
   > EOF
   ```

4. Сделать скрипт исполняемым:

   ```
   $ chmod +x modified-root/usr/local/bin/hello.sh
   ```

5. Упаковать обратно в tar:

   ```
   $ tar -cf modified-fs.tar -C modified-root .
   ```

6. Добавить изменённую ФС как новый слой к базовому образу:

```
$ crane append \
  --base alt:latest \
  --new_layer modified-fs.tar \
  --new_tag localhost:5000/alt-modified:latest
```

Создаётся новый образ, состоящий из:

- оригинального слоя alt:latest;
- нового слоя с изменениями.

Проверка результата:

```
$ podman run --rm localhost:5000/alt-modified:latest hello.sh
```

Ожидаемый вывод:

```
Hello from modified ALT!
```

12.6.3.4. Управление тегами и манифестами

Команда **`crane tag`** создаёт или переназначает тег без передачи данных (меняется только ссылка на digest).

Создать тег stable на основе latest:

```
$ crane tag registry.test.alt:5000/myalt:latest stable
```

Теперь registry.test.alt/myalt:stable указывает на тот же образ, что и latest.

Команда **`crane manifest`** выводит сырой JSON-манифест образа — полезно для отладки, анализа слоёв, проверки платформы.

Просмотр манифеста образа:

```
$ crane manifest localhost:5000/myapp:latest | jq .
```

Проверить multi-arch манифест (индекс):

```
$ crane manifest alt:p11 | jq '.manifests[].platform'
```

Для multi-arch образов возвращается image index, а не конкретный манифест.

12.6.3.5. Работа с индексами

Команда crane index позволяет управлять multi-arch образами:

- append — добавить манифесты;
- filter — отфильтровать платформы.

Примечание

Все образы должны быть уже в реестре.

Пример фильтрации (удалить все, кроме linux/amd64):

```
$ crane index filter \
  alt:p11 \
  --platform linux/amd64 \
  --tag localhost:5000/newalt:slim-multiarch
```

Пример добавления манифеста

```
$ crane index append ubuntu \
  -m hello-
world@sha256:87b9ca29151260634b95efb84d43b05335dc3ed36cc132e2b920dd1955342d20 \
  -t example.com/hello-world:weird
```

12.6.3.6. Аутентификация

crane использует те же учётные данные, что и Docker:

- **`~/.docker/config.json`**;
- переменные окружения (DOCKER_USERNAME, DOCKER_PASSWORD);
- IAM-роли (для AWS ECR, GCR и др.).

Пример явного входа в начале сессии:

```
$ crane auth login -u alice -p mypass localhost:5000
```

Примечание

При работе с HTTP-реестрами необходимо использовать флаг **`--insecure`**.

12.7. Сканер уязвимостей Trivy

Trivy — сканер уязвимостей для контейнерных образов, файловых систем и репозиториев git.

Кроме того, **Trivy** позволяет:

- выявлять ошибки в файлах конфигурации;
- находить жёстко заданные конфиденциальные данные (секреты);
- анализировать лицензии используемых компонентов и выявлять несовместимости.

12.7.1. Установка

Установка пакета:

```
# apt-get install trivy
```

12.7.2. Использование

Общий синтаксис:

```
trivy <команда> [--scanners <сканер1,сканер2>] <цель>
```

Таблица 12.9. Команды сканирования

Команда	Краткая форма	Назначение
image	**i**	Сканирование OCI/Docker-образа контейнера
filesystem	**fs**	Сканирование локальной файловой системы
config	-	Анализ файлов конфигурации (IaC: Terraform, CloudFormation, Dockerfile и др.)
repository	**repo**	Сканирование удалённого git-репозитория
rootfs	-	Сканирование корневой файловой системы (например, распакованного образа)
sbom	-	Анализ SBOM (Software Bill of Materials) — состава пакетов и зависимостей
kubernetes	**k8s**	Сканирование манифестов Kubernetes или работающего кластера
vm	-	Сканирование образов виртуальной машины (QCOW2, VMDK и др.)
aws	-	Аудит безопасности учётной записи AWS

Таблица 12.10. Команды управления и утилиты

Команда	Назначение
module	Управление модулями Trivy
plugin	Установка и управление плагинами
vex	Работа с VEX-документами (Vulnerability Exploitability eXchange)
completion	Генерация скрипта автодополнения для указанной оболочки (bash/zsh/fish)
server	Запуск Trivy в режиме сервера (для многопользовательского использования)
clean	Очистка кеша (базы уязвимостей и метаданных)
convert	Преобразование отчёта Trivy из JSON в другие форматы (SARIF, CycloneDX и др.)
registry	Управление аутентификацией в приватных реестрах (Docker Hub, ECR, GCR и др.)
version	Вывод версии

Таблица 12.11. Доступные сканеры (указываются через --scanners)

Сканер	Описание	По умолчанию
vuln	Поиск известных уязвимостей (CVE, GHSA и др.)	Да

Сканер	Описание	По умолчанию
secret	Обнаружение секретов (ключи, токены, пароли)	Да
config	Проверка IaC и конфигураций на ошибки безопасности	Нет
license	Анализ лицензий	Нет

Для получения подробной информации о команде можно использовать команду:

```
$ trivy <команда> --help
```

12.7.3. Примеры использования

12.7.3.1. Образы контейнеров

Сканирование образа на уязвимости:

```
$ trivy image alt:p11
```

Сканирование образа на наличие уязвимостей HIGH и CRITICAL с сохранением результата в формате JSON в файл:

```
$ trivy image --severity HIGH,CRITICAL -f json -o test.json alt:p11
```

Анализ лицензий:

```
$ trivy image --scanners license alt:p11
```

Проверка конфигурации только в метаданных образа:

```
$ trivy image --scanners none --image-config-scanners config alt:p11
```

Сканирование локального образа (Podman):

```
$ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED       SIZE
registry.altlinux.org/alt/nginx  latest      862baa6fbed9  3 months ago  136 MB
registry.altlinux.org/alt/alt    p11         ff2762c6c8cc  6 months ago  118 MB

$ trivy image ff2762c6c8cc
```

Примечание

Для возможности сканирования локальных образов, должен быть запущен podman.socket:

```
$ systemctl --user start podman.socket
```

12.7.3.2. Репозиторий git

Сканирование репозитория git на уязвимости и конфиденциальную информацию:

```
$ trivy repo https://github.com/altlinux/admc
```

Сканирование конкретной ветки:

```
$ trivy repo --scanners license --branch run-sh https://github.com/altlinux/admc
```

Также доступны параметры **--commit** и **--tag**.

12.7.3.3. Файловая система

Сканирование локальной файловой системы (проверка файлов конфигурации и конфиденциальной информации):

```
$ git clone git://git.altlinux.org/gears/o/openuds-tunnel.git
$ trivy fs --scanners=config ./openuds-tunnel/
```

Проверка фалов конфигурации:

```
$ trivy config ./openuds-tunnel/
```

12.7.3.4. Kubernetes

Просканировать кластер и создать сводный отчет:

```
$ trivy k8s --report=summary cluster
```

Полный отчёт по критическим уязвимостям:

```
$ trivy k8s --report=all --severity=CRITICAL cluster
```

12.7.4. Клиент/сервер

Trivy может работать в режиме клиент/сервер. В этом режиме база данных уязвимостей хранится на сервере, и клиенту Trivy не требуется загружать её локально.

При этом некоторые типы сканирования выполняются на стороне клиента даже при использовании клиент-серверного режима.

Таблица 12.12. Распределение сканеров

Сканер	Клиент/Сервер
Vulnerability	Сервер
Misconfiguration	Клиент
Secret	Клиент
License	Сервер

Проверка конфигураций (misconfiguration) и поиск секретов выполняются на стороне клиента (аналогично автономному режиму). Это связано с тем, что в противном случае клиенту пришлось бы передавать на сервер файлы, которые могут содержать конфиденциальную информацию.

Запуск сервера:

```
$ trivy server --listen localhost:8081
```

Примечание

Для возможности подключения с других хостов вместо localhost необходимо указать 0.0.0.0 или IP-адрес сервера.

Удалённое сканирование образа:

```
$ trivy image --server http://192.168.0.169:8081 alt:p11
```

Удалённое сканирование файловой системы:

```
$ trivy fs --server http://localhost:8081 --severity CRITICAL ./
```

12.7.5. Локальная база данных Trivy

Пакет *trivy-db* содержит базу данных уязвимостей для Trivy. База данных, установленная из пакета, используется только в клиент-серверном режиме (при запущенном сервере Trivy).

Для использования локальной базы данных необходимо:

1. Утановить пакеты *trivy-db* и *trivy-server*:

   ```
   # apt-get install trivy-db trivy-server
   ```

2. Запустить сервер и добавить его в автозагрузку:

   ```
   # systemctl enable --now trivy
   ```

Примеры:

- сканирование файловой системы с использованием локального сервера (на той же машине):

  ```
  $ trivy fs --server http://localhost:4954 ./
  ```

- сканирование образа с удалённой машины:

  ```
  $ trivy image --server http://192.168.0.169:4954 alt:p11
  ```

  где 192.168.0.169 — IP-адрес сервера Trivy.

Глава 13. Техническая поддержка продуктов «Базальт СПО»

13.1. Покупателям нашей продукции

«Базальт СПО» предоставляет следующие виды технической поддержки:

- Поддержка продукта входит в стоимость лицензии и включает регулярный выпуск обновлений, исправление ошибок, устранение уязвимостей в течение всего срока жизни дистрибутива.
- Поддержка пользователей обеспечивает качественную эксплуатацию продукта. Техническая поддержка эксплуатации продуктов «Базальт СПО» оказывается в объеме SLA. Доступны три уровня SLA («Базовый», «Стандартный» и «Расширенный»).

Право на получение консультационной и технической поддержки вы приобретаете при покупке большинства продуктов торговой марки Альт. Сроки и объём помощи указаны в сертификате технической поддержки.

Условия технической поддержки можно найти на странице сайта «Базальт СПО»: http://www.basealt.ru/support.

13.2. Пользователям нашей продукции

Вне зависимости от того, скачали вы или же приобрели наш дистрибутив, задавать вопросы или обсуждать их с сообществом пользователей дистрибутивов «Альт» вы можете на форуме или в списках рассылки.

Помощь сообщества:

- Документация сообщества: https://altlinux.org
- Форум: https://forum.altlinux.org
- Списки рассылки: https://lists.altlinux.org
- Сообщить об ошибке: https://bugs.altlinux.org
- Репозиторий: https://packages.altlinux.org
- Сборочная среда: https://git.altlinux.org
- Telegram-канал сообщества: https://telegram.me/alt_linux

Ресурсы компании «Базальт СПО»:

- Сайт компании: https://www.basealt.ru
- Контакты: https://basealt.ru/contacts
- Новости обновлений безопасности: https://cve.basealt.ru

Форум и списки рассылки читают опытные пользователи, профессиональные системные администраторы и разработчики «Базальт СПО». Сообщество пользователей и специалистов окажет содействие в поиске ответа на ваш вопрос или посоветует выход из сложной ситуации. При обращении к данному виду помощи у вас нет гарантии на полноту и своевременность ответа, но мы стараемся не оставлять без ответа вопросы, задаваемые в списках.